ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА
КАФЕДРА ТЕОРІЇ ТА ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

"ПРОГРАМУВАННЯ: ТЕОРІЯ ТА ПРАКТИКА"

ЗБІРНИК МАТЕРІАЛІВ
ЗА РЕЗУЛЬТАТАМИ ІТ-ПРОЄКТУ
МІЖДИСЦИПЛІНАРНОЇ ІНТЕГРАЦІЇ
2020-2021 навчальний рік

КИЇВ, 2021

УДК 004.9
ББК 32.973
П78

Рекомендовано до друку вченою радою факультету комп’ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка (Протокол № 15 від 07 червня 2021 року).

Рецензенти:
Марченко О.О., доктор фізико-математичних наук, професор кафедри математичної інформатики
Шкільняк О.С., кандидат фізико-математичних наук, доцент кафедри інтелектуальних програмних систем

Загальна редакція:
Омельчук Л.Л., кандидат фізико-математичних наук, доцент кафедри теорії та технології програмування
Ткаченко О.М., кандидат технічних наук, доцент кафедри теорії та технології програмування
Шишацька О.В., кандидат фізико-математичних наук, асистент кафедри теорії та технології програмування

Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції / За редакцією Л.Л. Омельчук, О.М. Ткаченка, О.В. Шишацької. – Одеса: Айс Принт, 2021. – 161 с.

Збірник містить звіти про студентські роботи, виконані в рамках міждисциплінарного проєкту-експерименту інтеграції на прикладі теоретично орієнтованих курсів "Методи специфікації програм", "Коректність програм та логіки програмування" і практично орієнтованого курсу "Інструментальні середовища та технології програмування" (спеціальність 122 "Комп’ютерні науки").

Матеріали подано в авторській редакції, відповідальність за достовірність фактів, цитат, посилань на джерела та вживання назв документів, власних імен тощо несуть автори публікацій.

ISBN

ЗМІСТ

ВПРОВАДЖЕННЯ МЕТОДИКИ ІНТЕГРАЦІЇ НАВЧАЛЬНИХ КУРСІВ НА ОСНОВІ ПРОЄКТНОГО ПІДХОДУ ТА ГНУЧКИХ МЕТОДОЛОГІЙ УПРАВЛІННЯ

Людмила Омельчук, Олексій Ткаченко, Олена Шишацька

Актуальність роботи. Результати "Аналізу ІТ-освіти у вишах України" [1], розробленого експертами Офісу ефективного регулювання BRDO в лютому 2021 року, свідчать, що наразі попит на нових ІТ-фахівців в Україні складає 30-50 тисяч осіб на рік. У свою чергу, заклади вищої освіти України щороку випускають у середньому 16,2 тисяч бакалаврів ІТ-спеціальностей. Разом з тим, лише 44% випускників українських закладів вищої освіти після отримання диплому працюють за фахом [2], решта не відповідають рівню своєї кваліфікації. Зазначене свідчить про наявність проблем у професійній підготовці ІТ-фахівців, зокрема, у відповідності рівня фахової підготовки випускників сучасним потребам ІТ-ринку.

Як свідчать опитування стейкголдерів освітнього процесу, при підготовці фахівців для ІТ-галузі основними проблемами є [1, 2]:

(1) існування невідповідності між очікуваннями роботодавців в галузі ІТ та практичними уміннями і соціальними навичками випускників ІТ-факультетів і, як наслідок, висока потреба у підвищенні рівня зазначених компетентностей у здобувачів вищої освіти;

(2) недостатній зв’язок між теоретичним і практичним матеріалом навчальних дисциплін;

(3) розмитість міждисциплінарних зв’язків;

(4) недостатнє системне розуміння студентами життєвого циклу розробки програмного забезпечення.

Інтеграція навчальних курсів в галузі інформаційних технологій на основі проєктного підходу та гнучких методологій управління повинна сприяти усуненню зазначених проблем. Розробка та впровадження методики проведення інтегрованих курсів (далі – проєкт-експеримент) здійснюється в межах освітньо-професійної програми "Інформатика" бакалаврського рівня вищої освіти, що реалізується на факультеті комп’ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка за спеціальністю 122 "Комп’ютерні науки". В рамках спільного проєктного завдання для студентів другого та четвертого років навчання було поєднано практичні частини наступних дисциплін:

- "Методи специфікації програм" (вибіркова дисципліна, 4 рік навчання, викладач: к.ф.-м.н. Шишацька О.В., задіяно 25 студентів);
- "Коректність програм та логіки програмування" (вибіркова дисципліна, 4 рік навчання, викладач: к.т.н. Ткаченко О.М., задіяно 25 студентів);
- "Інструментальні середовища та технології програмування (обов’язкова дисципліна, 2 рік навчання, викладач: к.ф.-м.н. Омельчук Л.Л., задіяно 8 студентів).

В проєкті-експерименті передбачалося досягнення таких цілей:

(1) посилення практичної складової навчальних дисциплін, задіяних у проєкті-експерименті;

(2) підвищення у студентів рівня розуміння:

- усіх етапів життєвого циклу програмного забезпечення;
- підходів до управління ІТ-проєктами;
- міждисциплінарних зв’язків;
- зв’язків між теоретичним і практичним матеріалом;

(3) підвищення рівня професійних та соціальних навичок;

(4) врахування результатів проєкту-експерименту при перегляді та оновленні освітньої програми та її компонентів.

Для досягнення цілей проєкту-експерименту було поставлено такі задачі:

(1) інтеграція на прикладі теоретично орієнтованих курсів "Методи специфікації програм", "Коректність програм та логіки програмування" та практично орієнтованої курсу "Інструментальні середовища та технології програмування";

(2) проведення експерименту, в рамках якого для виконання проєктних завдань формуються команди за різними критеріями: володіння технологіями розробки ПЗ, методологіями управління ІТ-проєктом, складом (віковий, соціально-спрямованими вподобаннями, ін.);

(3) розробка критеріїв оцінювання успішності запропонованого підходу до інтеграції дисциплін;

(4) аналіз результатів успішності проєкту-експерименту, формування пропозицій щодо його вдосконалення та впровадження як кейсу на постійній основі;

(5) розробка програмного продукту підтримки інтеграції навчальних дисциплін.

Цільовою аудиторією проєкту-експерименту є:

- студенти та викладачі бакалаврської освітньої програми "Інформатика" за спеціальністю 122 "Комп'ютерні науки", що реалізується на факультеті комп'ютерних наук та кібернетики КНУ імені Тараса Шевченка, які задіяні в рамках дисциплін "Методи специфікації програм", "Коректність програм та логіки програмування" та "Інструментальні середовища та технології програмування" (пряма аудиторія);
- ІТ-компанії, студенти і викладачі інших освітніх програм за ІТ-спеціальностями (опосередкована аудиторія).

Проєкт-експеримент було організовано у другому семестрі 2020/2021 навчального року. Було визначено наступні показники досягнення цілей:

(1) підвищення рівня професійних та соціальних навичок у студентів, задіяних в проєкті-експерименті (інструмент перевірки – вихідне опитування);

(2) наявність програмних систем, розроблених студентськими командами;

(3) наявність звітів за результатами роботи кожної студентської команди;

(4) наявність звіту викладачів про результати проєкту-експерименту;

(5) оприлюднення результатів проєкту-експерименту.

Проведення проєкту-експерименту. Для студентів четвертого курсу участь в проєкті була обов'язковою. Студентам другого курсу було запропоновано на добровільних засадах долучитися до проєкту-експерименту. При цьому було враховано результати навчання (підсумковий бал більше 80) з обов'язкової дисципліни "Об'єктно-орієнтоване програмування", яку ці студенти вивчали в попередньому семестрі.

На початку семестру студентам четвертого року навчання було запропоновано пройти анкетування, яке включало такі питання:

- **чи маєте Ви досвід роботи в реальних колективних проєктах (не навчальних)** (*був досвід, не маю досвіду, зараз в проєкті);*
- **запропонуйте декілька варіантів предметних областей або конкретних систем Вашого майбутнього проєкту;**
- **вкажіть не більше, ніж 3 людини, з якими Ви б хотіли працювати в команді;**

- **на Вашу думку, в якій ролі в командній роботі Ви би** *(почувалися впевнено, НЕ хотіли себе бачити, хотіли "прокачати" себе в проєкті / керівник проєкту, модератор, фахівець з документації, аналітик, фронтенд, бекенд, тестувальник);*
- **які компетентності Ви би хотіли розвинути в рамках майбутнього проєкту** (поглибити теоретичні знання, розвинути практичні вміння, програмувати, оформляти документацію, працювати в команді, планувати і організовувати свій час, вміння презентувати результати діяльності, комунікаційні навички, інше);
- **досвід роботи з якою платформою і/або методологією управління ІТ-проєктами Ви маєте**;
- **в якій системі управління ІТ-проєктами (методології) Ви б хотіли працювати в проєкті**;
- н**а Вашу думку, використання якої мови для Вас є** *(близькою, Ви нею володієте на впевненому рівні, не бажаною, Ви не впевнені, що володієте нею на хорошому рівні, бажаною в проєкті, бо Ви хочете "прокачати" себе в ній / C#, C++, Java, PHP, Python, інше)*;
- **ваші пропозиції щодо організації проєкту.**

На рис. 1-3 наведено деякі результати вхідного анкетування.

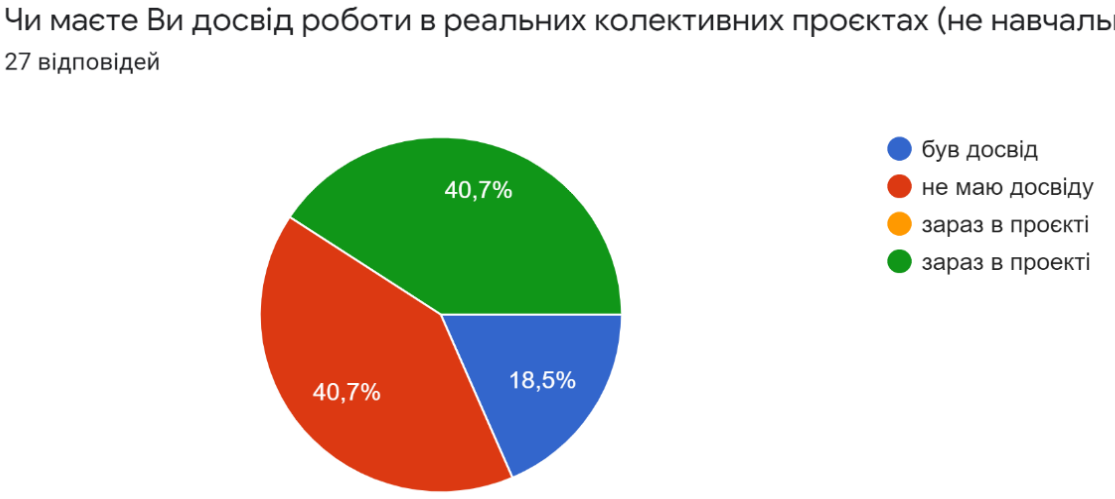


Рисунок 1. Результати анкетування щодо досвіду участі в колективних проєктах

Рисунок 2. Результати анкетування щодо бажаних компетентностей

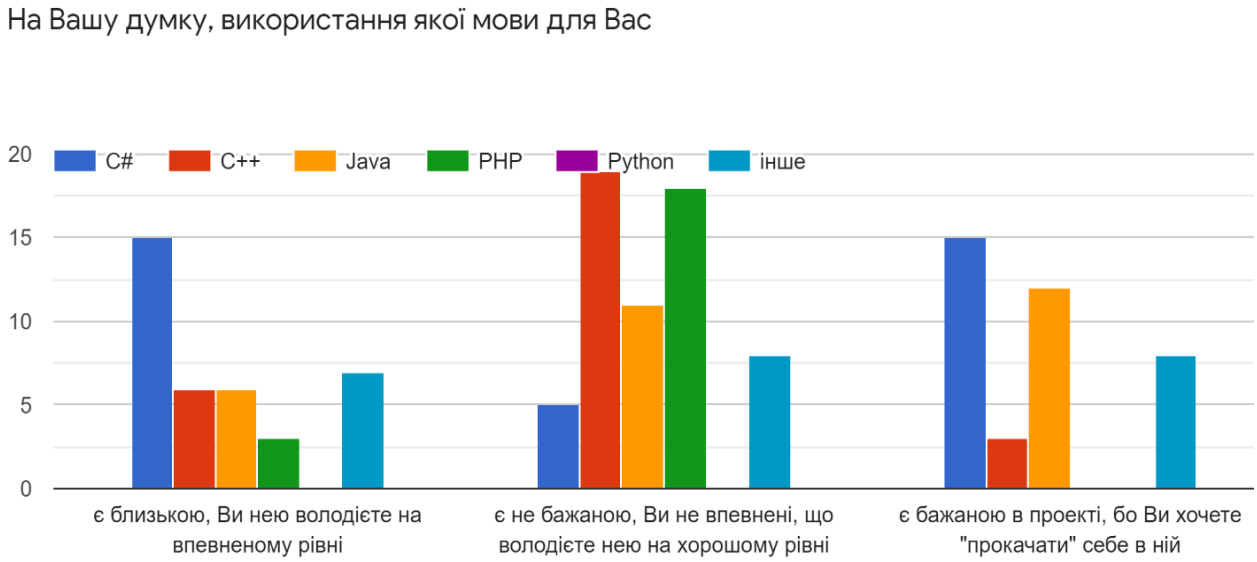


Рисунок 3. Результати анкетування щодо рівня володіння мовами програмування

За результатами анкетування для формування команд було побудовано соціометричний орієнтований граф, який враховував досвід роботи в командних проєктах, ступінь володіння технологіями та вибір колег. З метою експерименту для формування команд застосовувалися різні підходи:

- за роком навчання: команди, що складалися виключно зі студентів четвертого року навчання та команди, що складалися зі студентів четвертого та другого років навчання;
- за соціальними вподобаннями (з урахуванням вибору бажаних партнерів по проєкту): команди з тісними соціальними зв'язками (взаємний вибір) та команди з відсутніми соціальними зв'язками;
- за досвідом участі в реальних колективних проєктах: усі учасники мали досвід, ніхто з учасників не мав досвіду, змішані команда;
- за технологіями: команди зі спільними побажаннями щодо технологій розробки та команди з різними вподобаннями.

В результаті було сформовано вісім команд, з яких сім успішно виконали проєкт. Варто зазначити, що команда, яка не завершила проєкт складалася зі студентів одного року навчання, які мали взаємний вибір (тісні соціальні зв'язки) та однакові технологічні вподобання.

В рамках проєкту-експерименту здійснювалися щотижневі зустрічі учасників студентських команд з викладачами та організовано фінальний публічний захист студентських розробок. Звіти про виконання колективних проєктів, розроблених в межах проєкту-експерименту, наведено в даному збірнику.

По завершенню проєкту-експерименту було проведено два вихідних опитування: анонімне та авторизоване. Деякі результати цих опитувань наведено на рис. 4-7.

Рисунок 4. Результати вихідного опитування щодо розвинених компетентностей

Рисунок 5. Результати вихідного опитування щодо бажаних компетентностей

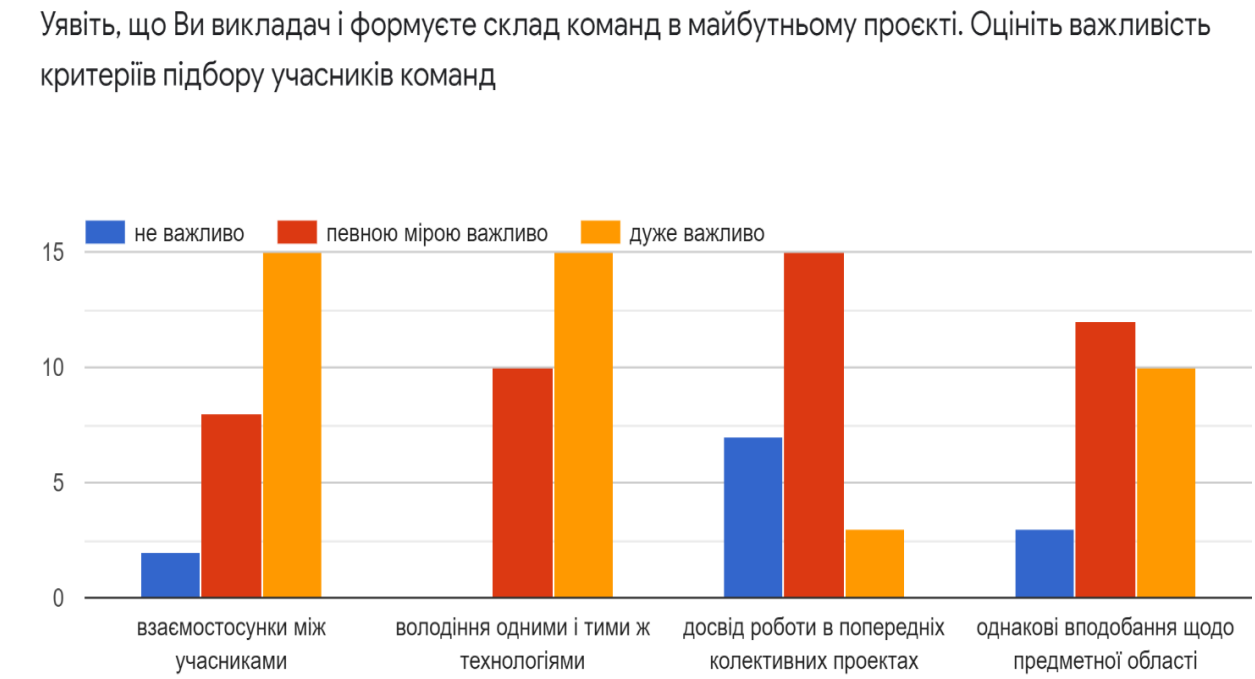


Рисунок 6. Результати вихідного опитування щодо важливості критеріїв формування команд

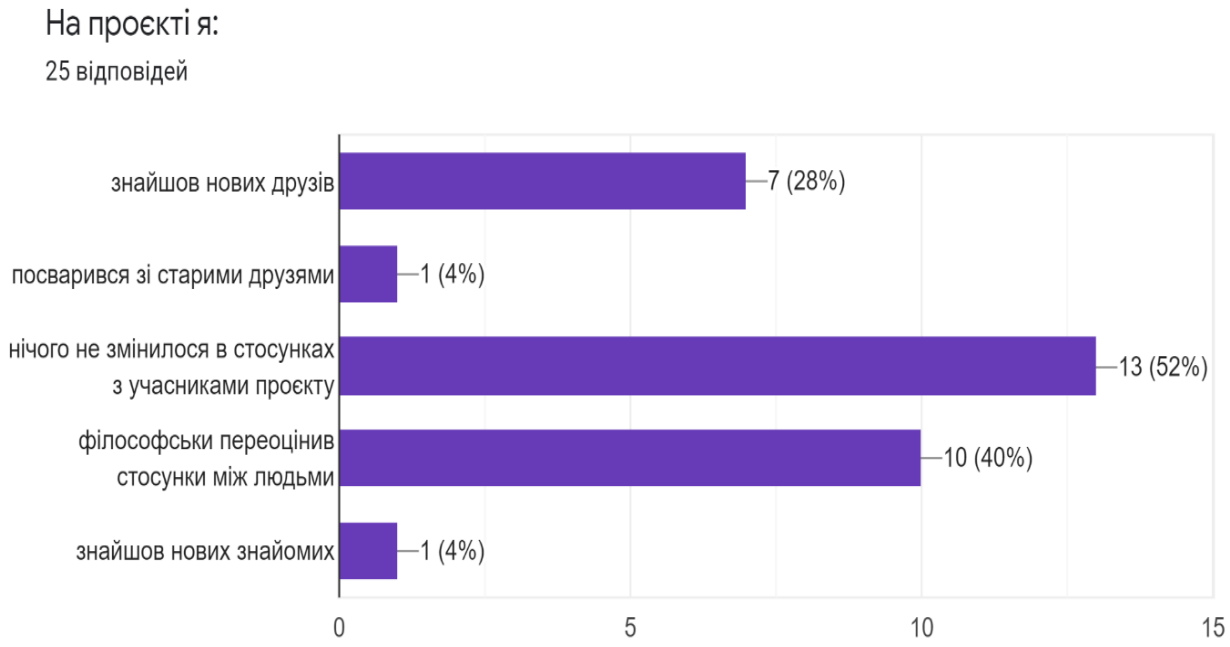


Рисунок 7. Результати вихідного опитування – рефлексія

Висновки. В результаті проведення проєкту-експерименту досягнуто наступні результати:

(1) розроблено програмні продукти підтримки інтеграції навчальних дисциплін;

(2) підвищено рівень професійних та соціальних навичок у студентів, задіяних в проєкті-експерименті;

(3) підвищено у задіяних в проєкті-експерименті студентів рівень розуміння всіх етапів життєвого циклу програмного забезпечення, підходів до управління ІТ-проєктами; міждисциплінарних зв'язків у межах освітньої програми; зв'язків між теоретичним та практичним матеріалом;

(4) посилено практичну складову навчальних дисциплін, задіяних в експерименті;

(5) сформовано пропозиції щодо оновлення робочих програм навчальних дисциплін, задіяних в експерименті, та освітньої програми в цілому: впроваджувати міждисциплінарний проєктний підхід з використанням гнучких методологій управління проєктами;

Подальшими кроками будуть:

(1) розробка та опис методики інтеграції навчальних курсів в галузі інформаційних технологій на основі проєктного підходу та гнучких методологій управління;

(2) вироблення рекомендацій щодо впровадження методики інтеграції навчальних курсів в галузі інформаційних технологій на основі проєктного підходу та гнучких методологій управління;

(3) поширення досвіду експерименту на інші навчальні дисципліни;

(4) впровадження програмного продукту підтримки інтеграції навчальних дисциплін, розробленого в межах експерименту;

(5) посилення практичної складової теоретично орієнтованих курсів;

(6) підвищення якості освітньої програми, зокрема, в контексті формування професійних та соціальних навичок у випускників ІТ-спеціальностей.

Література:

1. Лебедєв Д. (2021) Аналіз ІТ-освіти у ВИШах України / Д. Лебедєв, І. Самоходський, https://brdo.com.ua/wp-content/uploads/2021/02/Analiz_IT_osvity_u_vyshah_Ukraiiny_Print.pdf, останній доступ: 2021/05/22.

2. Експерт: 44% випускників вишів працюють не за фахом // Укрінформ https://www.ukrinform.ua/rubric-society/2737000-ekspert-44-vipusknikiv-visiv-pracuut-ne-za-fahom.html, останній доступ: 2021/05/22.

СТВОРЕННЯ СИСТЕМИ ПІДТРИМКИ ІНТЕГРОВАНОГО КУРСУ T-COLLAB

Едуард Андращук, Андрій Клячкін, Владислав Некряч,
Ярослав Приходько, Карина Скоробагатько

ПОСТАНОВКА ЗАДАЧІ

Огляд ринку. Нині на ринку немає прямих аналогів T-Collab. Звісно, все одно є сервіси, що частково мають функціонал нашої розробки, тож тут можна прочитати порівняння з найпопулярнішими такими платформами. Наприклад, з *Google Classroom*.

Із спільного:

– авторизація (в нас дещо змінена - можна реєструватись лише із поштової скриньки в домені @knu.ua)
– окремі класи, необмежена кількість
– стрічка (потік) повідомлень класу
–можливість спільної роботи

Недоліки (для інтегрованого курсу):

– немає виставлення повідомлень, завдань для декількох класів одночасно
– обов'язковий аккаунт Google

Переваги:

– оффлайн доступ до інформації
– можливість прикріплювати файли різних типів у відповідь на завдання
– можливість прикріплювати файли різних типів до повідомлення

Аналог системи для навчання ***Moodle***, спільне:

– можливість авторизації (в нас з додатковими обмеженнями)
– створення необмеженої кількості груп
– можливість здавати та отримувати виконані завдання
– можливість прикріплювати файли різних типів як виконане завдання

Недоліки (для інтегрованого курсу):

– відсутність можливості коментувати завдання, дивитись коментарі одногрупників
– відсутні нагадування про дедлайни
– відсутня можливість поділитись файлами з одногрупниками (одним чи декількома)
– відсутня можливість взаємодії з групою взагалі

Наші переваги:

– можливість створювати анкети для учнів з метою їх розподілу по групам (класам)
– розумний алгоритм розподілу учасників на групи, який спирається на подану інформацію в анкетах
– можливість коригувати результати алгоритму
– автоматизоване створення класів з правильним набором учасників
– збереження інформації про учнів, вчителів
– панель вчителя для моніторингу прогресу, результатів

Інструментарій (технології проєктування, розробки, верифікації та тестування). Використані технології проєктування: Miro (roadmap), Columns (дошка), draw.io (UML, ER-diagram), Figma, Google диск з Google Docs. Використані технології розробки: C#, ASP.NET

Web API, Python, F#, React/Redux, JS, PostgreSQL, GitHub, TS, Heroku.Використані технології верифікації та тестування: XUnit, Moq, React Testing Library, інструменти Visual Studio, Jest, Cypress.io, Postman.

Методологія та система управління проєктами. Команда обрала працювати за методологією Scrum. На першій зустрічі було вирішено проводити Daily Meetings, Retrospective та Planning Session. Відведений час на виконання проєкту ми розбили на 4 спрінти по два тижні, тож раз на два тижні ми проводили ретроспективу минулого спрінта та планування наступного. Для цього ми використовували Kanban дошку та Miro для зображення подальших планів та Roadmap. Також тричі на тиждень ми проводили Daily Meetings, але оскільки для такого роду проєкту виділено надто мало часу - в них не було багато сенсу, адже часто, звісно, були періоди коли нічого не було зроблено, наприклад за вихідні, а колись, навпаки, був надто важкий тиждень з масою модульних робіт і основний обсяг робіт виконувався на вихідний. Саме тому ми прийняли рішення що Daily Meetings не підходять нашому проєкту і було вирішено замінити їх на більш інформативні речі. Команда домовилась, що кожен її учасник писатиме в чат команди (у нас був створений час в телеграмі), що він зробив на кінець дня, якщо щось робив. Таким чином ми не отримуватимемо такого зайвого фідбеку як "ну я нічого не робив", а отримуватимемо лише інформацію про те, що робилось. Такий підхід видався набагато ефективнішим.

Отже, для швидкої комунікації був створений чат в Telegram в якому ми зберігали посилання на всі необхідні нам в роботі ресурси. Спілкувались ми у відведеному нам Google Meets, а також у Discord. У нас був своя папка на Google Drive, в якій ми зберігали необхідні письмові роботи, документи, скріншоти. Як вже було сказано вище, ми використовували дошку Miro для спільної роботи та планування компонентів. Для розробки дизайну ми використовували Figma. Для зберігання коду та забезпечення можливості одночасної розробки ми використовувати GitHub. Ретроспективи проводились за допомогою Jamboard, де кожен міг написати свої думки та пропозиції для їх обговорення. Для проєктування різних схем процесів та діаграми бази даних ми використовували draw.io. У кожного члена команди був доступ до кожного з ресурсів, також вчителі були додані до них.

Для вибору Kanban дошки було проаналізовано на протестовано 5 онлайн ресурсів: Jira, Trello, Asana, Azure DevOps та Columns. Головними критеріями вибору для нас стали доступність (ціна ресурсу), просте запрошення та вхід (мінімум дій для створення дошки та реєстрації в ній команди), мінімум функцій (нам не потрібно створювати важкі логіку задач, прописувати що є підзадачами, а що головним "епіком" для невеликого проєкту, так само як і не треба можливість писати workflow для задач та їх поведінки), гнучкі правила створення дошок, найпростіший не відволікаючий дизайн та мінімальна кількість дій для отримання максимальної вигоди.

Для огляду та аналізу було створено тестові дошки у кожній з систем, продемонстровані на першій зустрічі усім членам команди. На першій зустрічі команда обговорила та встановила вимоги до дошки та обрала найбільш відповіду до існуючих вимог.

– Jira має дуже багато налаштувань, вона важко налаштовується під гнучкі, часто змінні, вимоги до короткого проєкту. Вона підходить для більших команд та для більш тривалого процесу розробки. Також більшість розширень є для Jira, та сама вона не безкоштовна;

– Trello має дуже широкий функціонал як для безкоштовного ресурсу, не дивлячись на можливість розширити функціонал окремо за гроші. Нам хотілося знайти щось просте,

що б не обтяжувало та не ускладнювало роботу з дошкою. Trello видався нам трохи обтяжуючим та дещо менш гнучким у налаштуванні дизайну фону та самих задач;

– Asana має дещо ускладнену систему реєстрації та додавання нових користувачів у систему, також обтяжена зайвими для нашого конкретного проєкту можливостями;

– Azure DevOps - складна система реєстрації та налаштування дошки, багато зайвих функцій. Більше підійде для довготривалих проєктів з великою кількістю команд;

– Columns - ми обрали саме цей ресурс, тому про нього буде більше описано нижче.

Ідеальним вибором команді здалась Columns, до того ж вона є маловідомою, точніше, відомою лише у вузьких колах. Тож це було цікавим викликом було робити там, де ніхто інший робити не буде. Великим плюсом став мінімалістичний дизайн. В цій системі дійсно дуже просте створення дошки (1 клік) та дуже простий процес доєднання членів команди (такий самий як надати доступ до документу в Google, необхідно мати лише пошту). Оскільки проєкт університетський, то звісно нам не потрібен був розподіл доступів за ролями в самій дошці, тож те, що його не передбачено в системі Columns, стало первагою, адже ми економимо на налаштуваннях. Тож, завдяки цьому вибору, вже через 5 хвилин почали створювати разом задачі. Columns у порівнянні з іншими онлайн Kanban дошками має помітно обмежений функціонал, але все необхідне там є. А саме, ми використовували створення колонок для дошки, створення задач, фарбування задач в різні кольори, в залежності від виду діяльності (розробка, планування, написання звітів, документації, тестування тощо), також ми виставляли кінцевий термін для задач і відповідальних за задачу. Columns також дозволяє додавати до задач опис, файли, посилання, та навіть має можливість вести переписку в самій задачі. Приємний бонус: в Columns можна створювати окремі секції для задач та підзадачі.

Розподіл ролей. Склад команди та розподіл ролей можна переглянути в табл. 1.

Таблиця 1. Розподіл ролей у команді

Учасник	**Роль учасника на початку проєкту**	**Демо 1**	**Демо 2**	**Роль по завер-шенню проєкту**
Приходько	модератор, аналітик, дизайнер	фронтенд	бекенд	бекенд
Андращук	аналітик, модератор, дизайнер, фронтенд	фронтенд	фронтенд	фронтенд
Некряч	модератор, аналітик,, дизайнер	фулстек	фулстек	фулстек
Клячкін	модератор, аналітик, дизайнер	тестувальник, фахівець зі специфікації	Тестуваль-ник, бекенд	тестувальник, бекенд
Скоробагатько	керівник проєкту, аналітик	керівник проєкту, аналітик, модератор, дизайнер, фахівець з документації, дизайнер	керівник проєкту, аналітик, модератор	керівник проєкту, модера-тор, фахівець з докумен-тації, дизайнер

ТЕХНІЧНІ ВИМОГИ

Метою розробки інтеграційної платформи "T-Collab" є створення сайту з можливістю організувати та провести з його допомогою командні проєкти. Це система з можливістю авторизації, створення групових проєктів на їх супровід. Спершу бажаючі взяти участь у груповому проєкті повинні зареєструватись та заповнити відповідну анкету (анкету створює вчитель). Потім, коли вчителю зручно, він самостійно запускає розподіл – дивиться, як сформувались групи та має змогу вплинути на результат (за бажанням). Після розподілу учні отримують доступ до своїх класних кімнат, у яких буде змогла публікувати виконані завдання, розміщувати оголошення. Вчитель матиме змогу оцінювати як командну роботу загалом, так і кожного її учасника окремо.

Розробка компонентів програмного забезпечення. В рамках розробки інтегрованої системи були розроблені наступні програмні компоненти:

І блок – Розробка програмного забезпечення

1. Стартова сторінка з відгуками. Сторінка з відгуками учнів, що вже пройшли курс.
2. Сторінка для реєстрації учня. Реєстрація з введенням інформації про себе та відповідями на питання про команду на проєкті.
3. Сторінка для реєстрації вчителя. Реєстрація разом з введенням інформації про курс.
4. Сторінка для входу вже зареєстрованих користувачів. Вхід до власних кабінетів особам, що вже проходили реєстрацію на сайті.
5. Сторінка для вчителя, чий статус ще не підтверджений. Якщо людина зареєструвалась як вчитель, отримуємо підтвердження від адміністратора.
6. Початкова сторінка для вчителя, чий статус вже підтверджений. Тут інформація про те, що необхідно створити свій курс.
7. Сторінка для створення нового інтегрованого курсу. Тут поля для введення інформації щодо можливих мов програмування, ролей та іншої загальної інформації.
8. Програмний компонент "Кабінет вчителя". Забезпечено:
    - 8.1. відображення особистої інформації
    - 8.2. кнопка для запуску розподілу для кожного курсу окремо
    - 8.3. екран перегляду курсу, основної інформації по ньому (кількість зареєстрованих учнів, максимальна кількість учнів) з можливістю відредагувати його
    - 8.4. можливість створення ще одного курсу
    - 8.5. можливість редагування склад команд
    - 8.6. можливість переглянути детальну інформацію по кожному курсу
    - 8.7. можливість додати коментарі до будь-якого свого курсу
    - 8.8. можливість додати завдання до будь-якого свого курсу, редагувати та оцінювати їх
    - 8.9. можливість перегляду неоціненних робіт
9. Сторінка для учня, для якого розподілу ще не було. Якщо людина зареєструвалась як учень - необхідно дочекатись, коли ваш вчитель запустить розподіл по групах щоб побачити власний кабінет.
10. Прикладний програмний компонент "Кабінет учня". Забезпечено:
    - 10.1. перегляд курсів, на які зареєстрований
    - 10.2. перегляд їх основної інформації та найближчих дедлайнів
    - 10.3. перегляд оцінених завдань в конкретному курсі з оцінкою
    - 10.4. перегляд неоціненних завдань в конкретному курсі
    - 10.5. можливість додавати коментарі та вкладення до існуючих завдань.

Вимоги до розробки системи

Загальні вимоги. Розробка повинна будуватись з використанням підходів ООП/ФП, з уніфікацією програмно-технічних засобів розробки прикладної функціональності з використанням сучасних веб-портальних, сервісно-орієнтованих технологій.

Функціонально розробка повинні бути веб-застосунком та уніфіковане з точки зору програмно-апаратної платформи.

Базовими компонентами розробки мають бути програмні комплекси сервісів, що забезпечують реалізацію функціональності T-Collab.

Розробка повинна забезпечити уніфікований та комфортний, максимально простий та інтуїтивно зрозумілий інтерфейс користувача.

Технологічна гнучкість, надійність роботи при розробці та вибору функціонального складу, скорочення часу та сукупних витрат на розробку та підтримку проєкту повинен досягатись за рахунок реалізації принципів стандартизації та уніфікації, а саме:

- уніфікованих правил структурної побудови та організації прикладних програмних компонентів, їх взаємодії між собою;
- стандартизації вимог до побудови єдиної централізованої бази даних, формування єдиних вимог до класифікації об’єктів та їх атрибутивного складу;
- уніфікації правил побудови інформаційної взаємодії з іншими системами.

Вимоги до технічної та інформаційної архітектури. Розробка повинна вбудовуватись у Microsoft Visual Studio, Visual Studio Code та Webstorm, Pycharm з використанням підходу Database first. Обмін інформацією між сервером (БД) та клієнтською частиною системи та іншими зовнішніми системами повинен реалізовуватись у форматі JSON із застосуванням технології веб-сервісів. Компонент серверу застосувань реалізації бізнес-логіки призначені для створення серверних служб доступу до об’єктів та бізнес-логіки прикладної функціональності у відповідності до функціональних задач. Компонент серверу застосувань сервісів інформаційної взаємодії призначені для забезпечення ведення регламентів взаємодії та механізмів інформаційного обміну. Компонент серверу застосувань сервісів обробки та управління інформаційними даними призначений для формування, актуалізації даних та виконання запитів до бази даних.

Вимоги до програмного забезпечення. Програмне забезпечення (ПЗ) повинне складатися із:

- загальносистемного програмного забезпечення (ЗПЗ);
- прикладного програмного забезпечення (ППЗ).

Програмне забезпечення повинно відображати специфіку автоматизованих функціональних задач користувачів та забезпечувати:

- підтримку загальноприйнятих сучасних міжнародних стандартів до відкритих систем;
- сумісність та інтегрованість;
- підтримку функціонування в різнорідному апаратному і програмному середовищах;
- вмонтованість механізму захисту від помилок і підтримки цілісності;
- мінімальні витрати на їх закупівлю та експлуатацію.

Детальний склад загальносистемного програмного забезпечення визначається на етапі розробки технічного завдання. Рішення зі складу загальносистемного програмного забезпечення повинні бути технічно та економічно обґрунтовані з точку зору забезпечення повноти забезпечення застосування доопрацювань та його компонентів за призначенням та

мінімізації витрат на закупівлю та супровід.

До прикладного програмного забезпечення повинні відноситись програмне забезпечення, що розробляється та налаштовується під час розробки. За результатами розробки програмний код прикладного програмного забезпечення повинен бути переданий Виконавцем Замовнику в електронному вигляді. Розробка прикладного програмного забезпечення повинна проводитись за допомогою сучасних інструментальних засобів програмної інженерії проєктування і генерації розподілених баз даних (CASE-засобів).

При розробці ППЗ повинні використовуватися принципи модульності та типовості, які забезпечать послідовне нарощування функціональних можливостей доопрацювань за рахунок створення, впровадження та тиражування функціонально завершених програмних компонентів.

Детальні вимоги щодо створеного програмного забезпечення доопрацювань повинні бути уточнені за результатами обстеження об'єктів автоматизації та викладені в Технічному завданні.

Вимоги до чисельності, кваліфікації технічного персоналу та режиму роботи. Виконавцем повинні бути запропоновані рішення щодо чисельності та кваліфікації обслуговуючого персоналу оновленого рішення. Пропозиція повинна бути обґрунтована та мати оптимізований склад обслуговуючого персоналу.

Технічна підтримка програмного забезпечення системи буде здійснюватися за окремими договорами спеціалізованими організаціями, підприємствами чи установами.

Вимоги до показників навантаження. Розробка має забезпечувати:

- обслуговувати одночасну роботу до 1000 користувачів;
- швидкість обробки запитів та відображення інформації на сторінці – не більше ніж 6 сек;
- первісне завантаження будь-якої веб-сторінки – не більше 5 сек.

Вимоги до надійності. Надійність розробки повинна бути забезпечена за наступними напрямками:

- забезпечення працездатності компонентів програмно-технічної платформи;
- збереження даних.

Збереження працездатності повинне забезпечувати надійність роботи при відмові одного або декількох компонентів за рахунок їх резервування. При цьому повинна вимагатися мінімальна увага з боку адміністратора щодо реакції на усунення наслідків відмов компонентів, а також програмно-апаратними засобами повинно бути забезпечене збереження даних.

Збереження даних повинно забезпечувати збереження цілісності даних при програмно-апаратних збоях, відмовах, помилках, шляхом використання відповідних програмно-апаратних засобів та рішень, резервного копіювання, транзакційності при змінах даних.

Деталізовані вимоги щодо розробки повинні бути уточнені в ході розробки Технічного завдання.

Вимоги до ергономіки. Рішення щодо ергономіки розробки повинні відповідати вимогам технічної естетики та інженерної психології для забезпечення гармонійного зв'язку між параметрами технічних засобів і психофізичними можливостями людини із врахуванням створення єдиного об'ємно-просторового і кольорового рішень відповідно до ДСТУ 2489-94, ДСТУ 2506-94.

Вимоги до захисту інформації від несанкціонованого доступ. Створення Комплексної системи захисту інформації не є предметом розробки T-Collab та повинно бути виконано в рамках іншого договору.

Комплексна система захисту інформації повинна реалізовувати сукупність узгоджених за часом та місцем застосування організаційних та технічних заходів, які ґрунтуються на принципах доцільності, неперервності та комплексності і включають:

- блокування несанкціонованого доступу до інформації чи її носіїв;
- перевірку справності та працездатності технічних засобів і систем обробки інформації та життєзабезпечення
- блокування витоку інформації технічними каналами.

Вимоги до організаційного забезпечення. Розробка системи повинна підвищити ефективність проведення командних проєктів.

Організаційне забезпечення, що створюється у межах розробки, повинно включати кабінети, які відображають автоматизований процес обробки інформації та ведення інтегрованих курсів (спільних проєктів) для користувачів системи.

Календарний план. На першій зустрічі перед початком розробки було сформовано календарний план проєкту (табл. 2).

Таблиця 2. Календарний план проєкту

№ з/п	**Назва послуг за етапами етапу**	**Термін**	**Результат**	**Вартість послуги без ПДВ, грн.**
1	Створення технічного завдання на розробку	5 робочих днів **	Технічне завдання	гарні оцінки
2	Розробка дизайну	7 робочих днів**	Інтерактивний зразок програмного забезпечення у figma.com	гарні оцінки
3	Розробка програмного забезпечення	24 робочих днів **	Дослідний зразок програмного забезпечення (MVP)	гарні оцінки
4	Загальна модернізація програмного забезпечення	5 робочих днів **	Програмне забезпечення на останній стадії до тестування	гарні оцінки
5	Тестування програмного забезпечення	5 робочих днів **	Програма та методика приймальних випробувань; Опис системи; Посібник користувача; Посбник адміністратора ; Завершене програмне забезпечення	гарні оцінки + приз

** Замовник може надавати власний варіант календарного плану.*

*** Строк розробки та порядок надання ТЗ може бути змінено, робочими днями вважаємо середу, четвер та п'ятницю*

МЕТОДОЛОГІЯ РОЗРОБКИ

Діаграма компонентів. Для проєкту було розроблено приблизний шлях користувачів у веб-застосунку, можливі дії та відповідні сторінки. На основі цього було створено діаграму компонент (рис. 1)

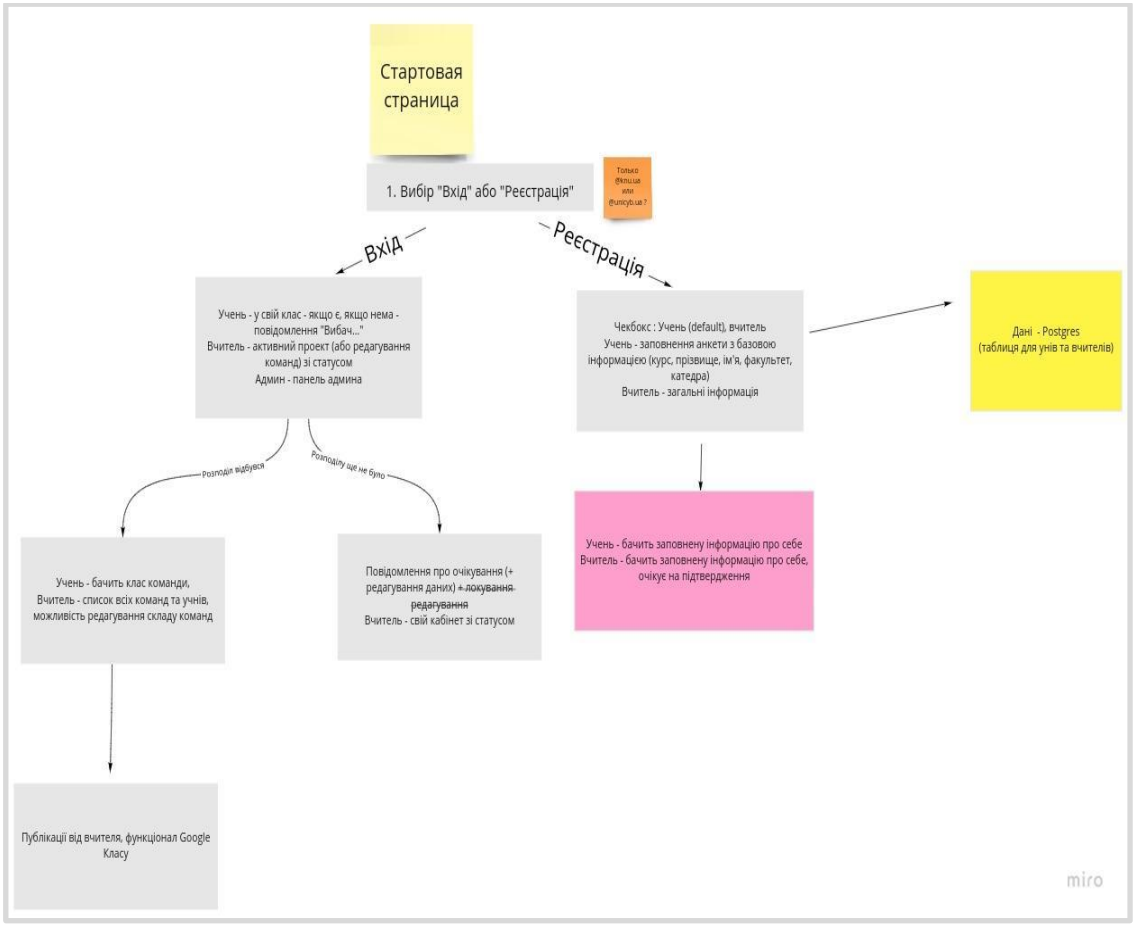


Рисунок 1. Діаграма компонентів

Діаграма Ганта. Для проєкту було створено діаграму Ганта (рис. 2), що показує заплановані етапи розробки, їх тривалість та приблизну вартість.

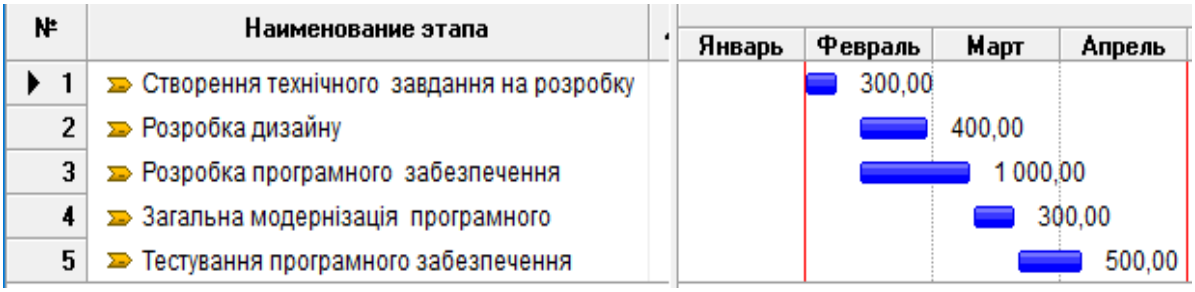


Рисунок 2. Діаграма Ганта

Діаграма послідовностей. Перед початком розробки було сформовано діаграму послідовностей (рис. 3), яку потім модифікували для використання cookies замість JWT.

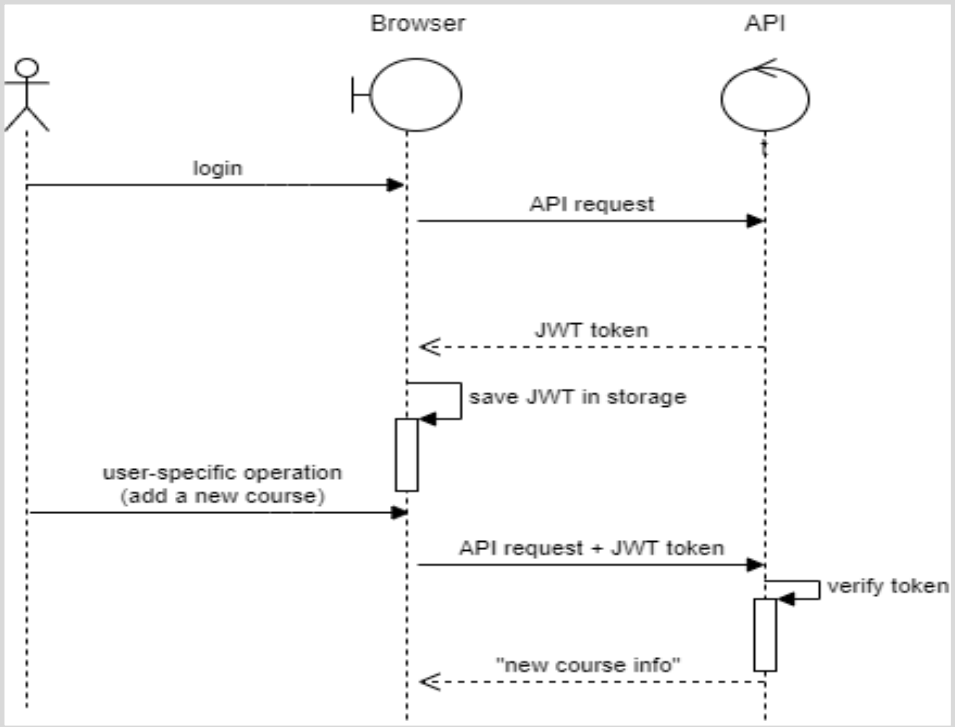


Рисунок 3. Діаграма послідовностей

Діаграма баз даних. Для проєктування веб-застосунку було сформовано діаграму бази даних. Насправді, ця діаграма змінювалась декілька разів у зв'язку з новими вимогами, розширенням функціоналу, більш детального осмислення бажаного результату від проєкту. Також діаграма бази даних була дещо змінена вже після початку розробки та виявлення нових вимог. Остаточну діаграму баз даних можна знайти у додатку А.

Daily Meetings. Щоденно ми проводили ранкові зустрічі до 15 хвилин, на яких всі члени команди відповідали на 3 питання, та з'ясовували чи необхідно нам додатково щось обговорити. Відповіді давались на наступні питання:

1. Що було зроблено в минулий робочий день?
2. Що планується в роботу на сьогодні?
3. Чи є якісь проблеми, що гальмують процес?

Результати деяких таких зустрічей ми записували в Miro. Такі зустрічі допомагають зрозуміти, як йде прогрес виконання задач, скоригувати пріоритети задач, виявити проблеми одразу, перед початком роботи, виявити необхідність у додаткових зустрічах.

Приклад результатів Daily Meetings можно побачити на рис. 4.

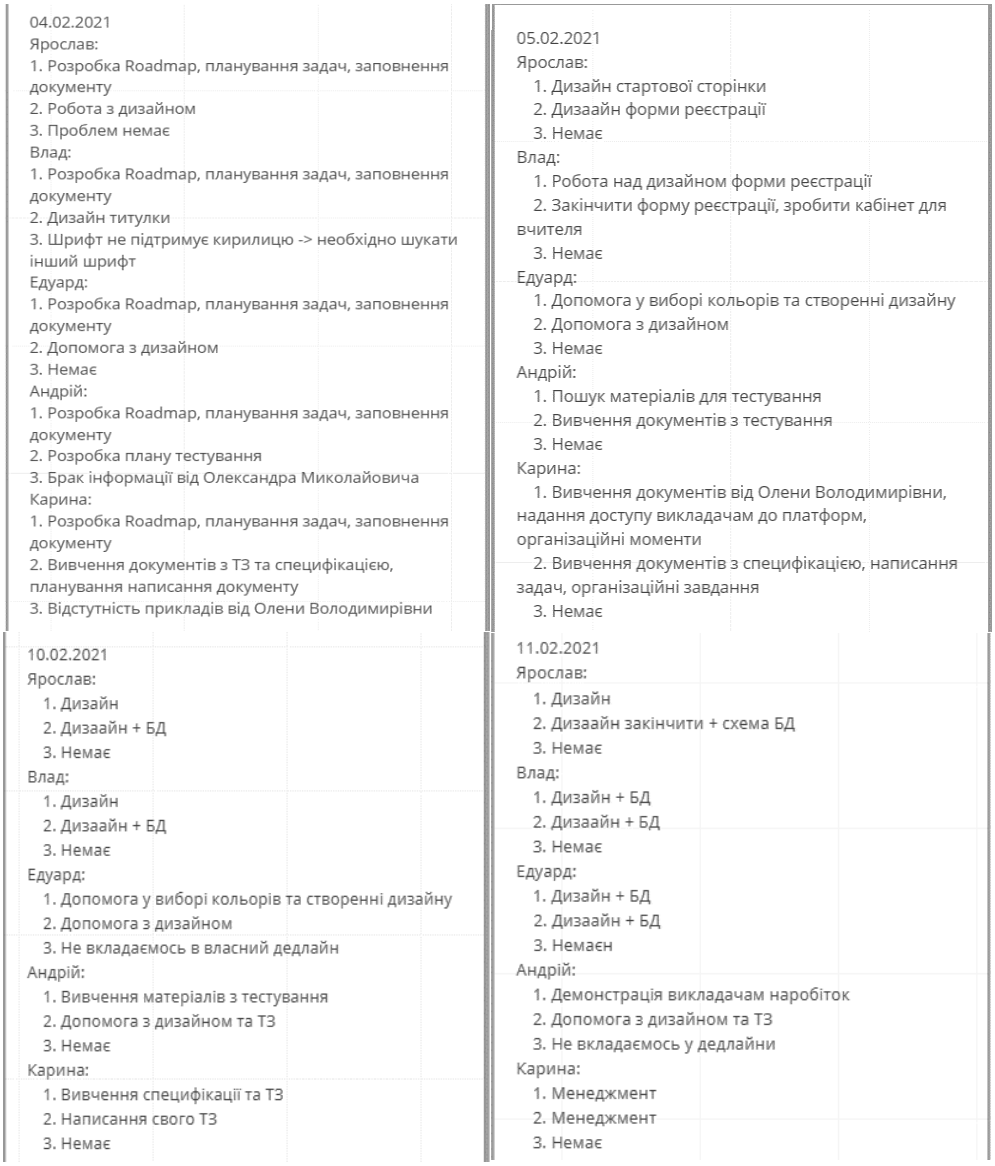


Рисунок 4. Приклад результатів Daily Meetings зустрічей

Ретроспективи. Нашу роботу за проєктом ми розділили на 4 двотижневих спрінти. Наш перший спрінт почався у середу 03.02.2021 та тривав до середи 17.02.2021. Дрий спрінт тривав з 17.02.2021 до 03.03.2021. Третій з 03.03.2021 до 17.03.2021, і останній спрінт тривав з 17.03.2021 по 31.03.2021. Під час ретроспективи, по-перше, йде обговорення настроїв членів команди, по-друге, обговорення процесу взаємодії, проблем, інших питань тощо, що стосуються завершеного спрінта. Під час формування результатів ретроспективи для оцінки настроїв ми використовували Miro, для обговорення проблем, того, що варто покращити, а що припинити робити, ми використовували Jamboard від Google.

Результати першої та другої ретроспектив представлені на рисунках 5–7. Результати третьої та четвертої ретроспектив представлені на рисунках 8-10.

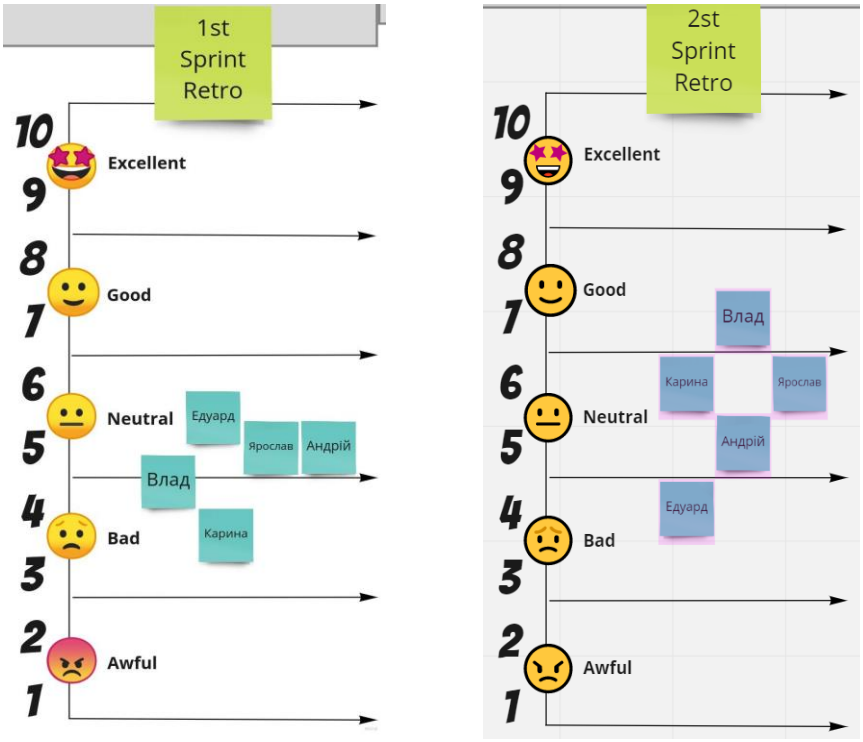


Рисунок 5. Результати першої (ліворуч) і другої (праворуч) ретроспективи (настрої)

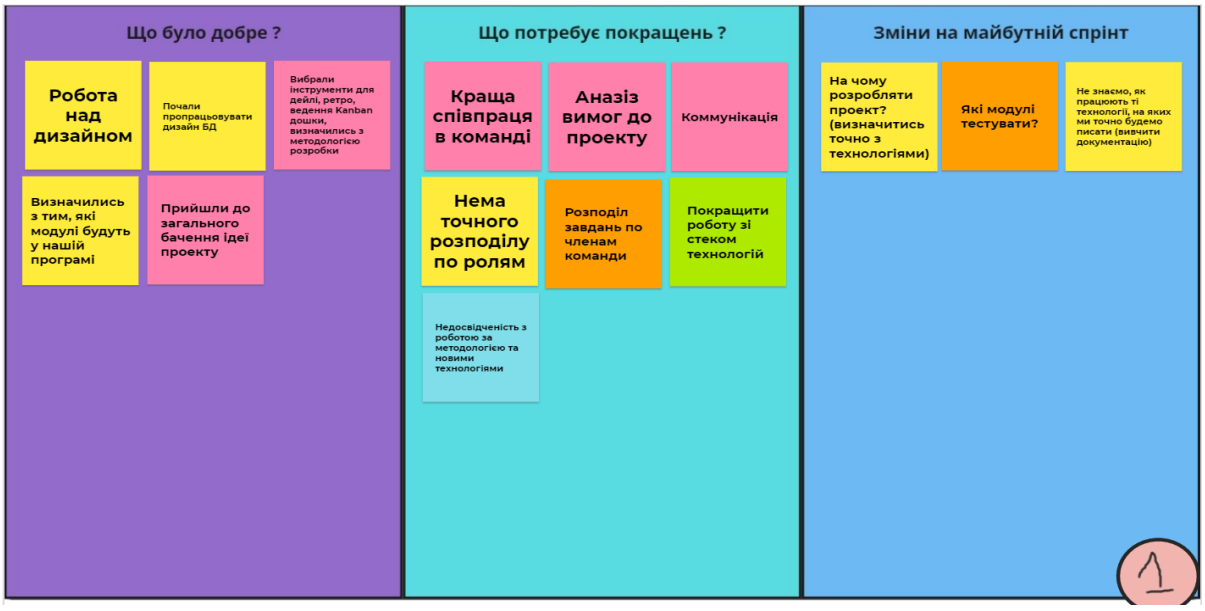


Рисунок 6. Результати першої ретроспективи (аналіз)

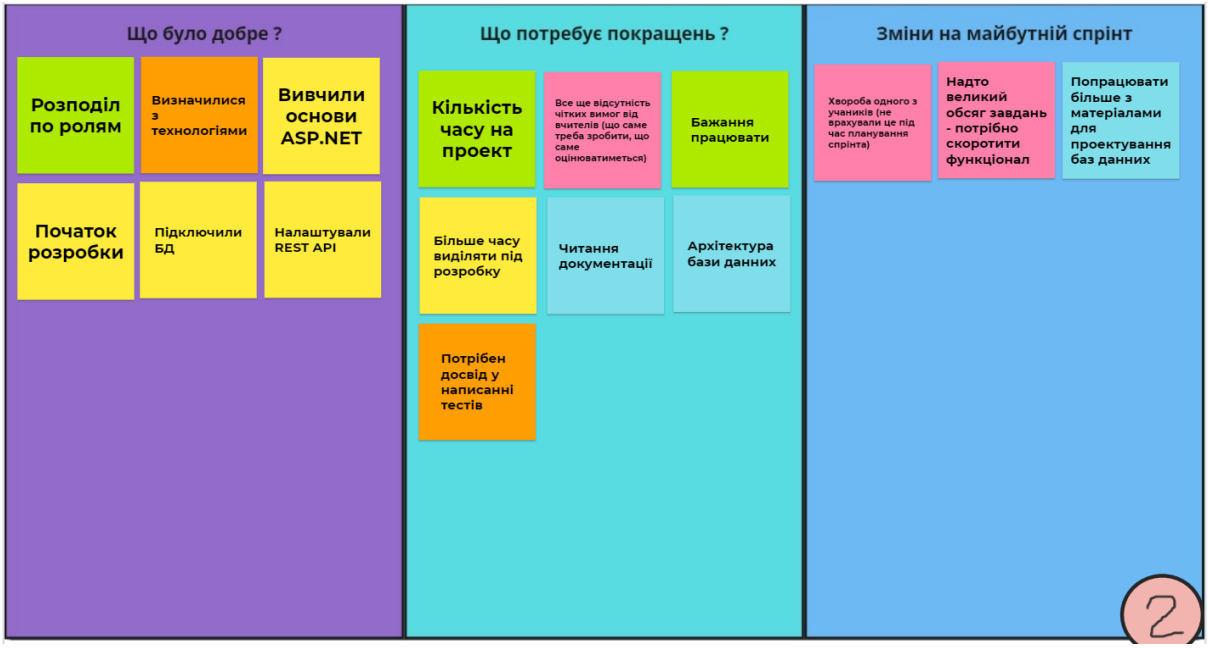


Рисунок 7. Результати другої ретроспективи (аналіз)

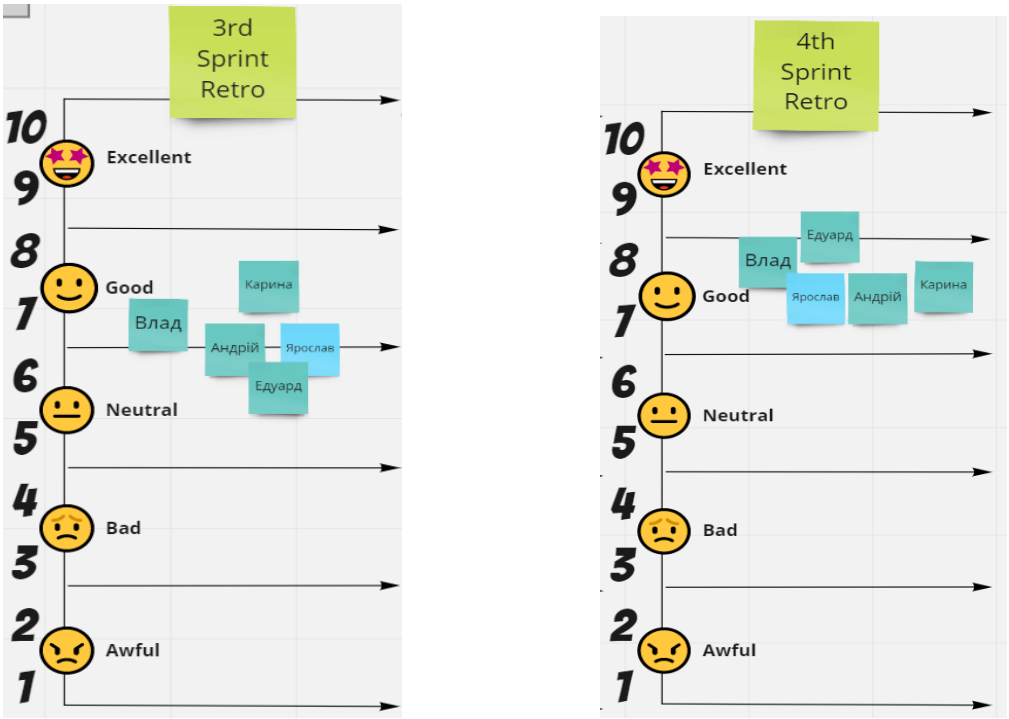


Рисунок 8. Результати третьої (ліворуч) і четверної (праворуч) ретроспектив

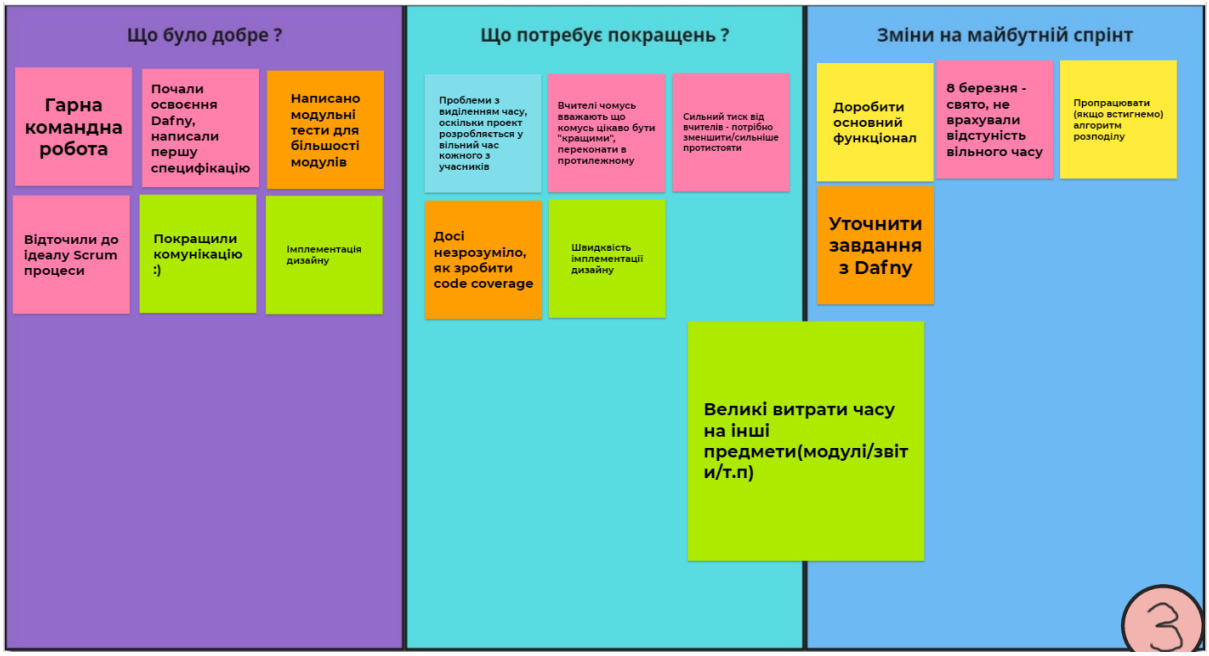


Рисунок 9. Результати третьої ретроспективи (аналіз)

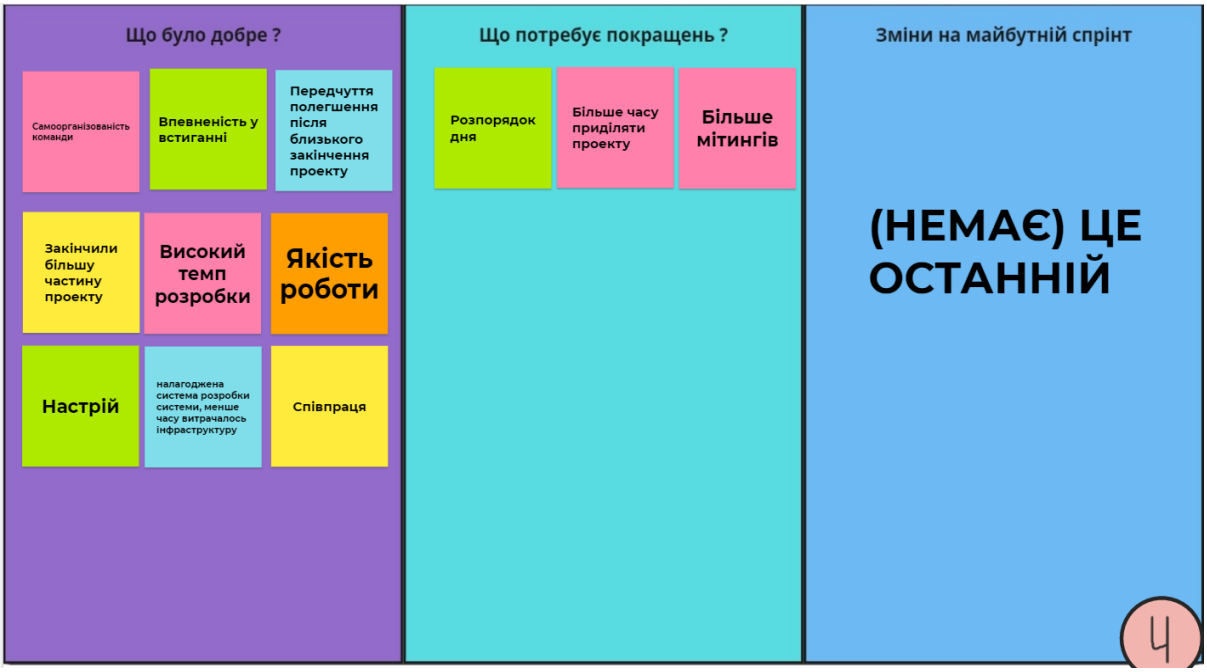


Рисунок 10. Результати четвертої ретроспективи (аналіз)

Закриті задачі. У кінці кожного спрінта ми також передивлялась задачі, що були закриті за цей період, та ті, що ми не встигли зробити. Після аналізу та перевірки виконаних задач ми переносили їх в колонку "виконані в Х спрінті", щоб мати історію того, скільки за який спрінт ми встигли зробити. Задачі, що були закриті в 1–4 спрінтах, подані в додатках Б-Д.

СПЕЦИФІКАЦІЯ ТА ВЕРИФІКАЦІЯ

Специфікація. Специфікацію було розроблено на мові Dafny. Специфікація проводилася для логіки реєстрації користувача. Для цього було розроблено клас User та клас Program.

Клас Program містить набір зареєстрованих користувачів та методи, що перевіряють, чи виконуються умови реєстрації. Користувач не має бути присутнім у переліку зареєстрованих, користувач має мати валідну електронну пошту та пошта має бути з домену knu.ua.

```
class User {
    var Email: string;
    var Password: string;
    constructor(email: string, pwd: string)
    {
        this.Email := email;
        this.Password := pwd;
    }
}
```

Рисунок 11. Dafny-код класу User

```
class Program {
    var users: set<User>;

    constructor (users: set<User>)
    modifies users
    {
        this.users := users;
    }

    static method isUnique(user: User, users: set<User>) returns (r: bool)
    {
        r:= user !in users;
    }

    static method strictEndsWith(substr: string, b: string) returns (r: bool)
        {
            r := |substr| < |b| && b[|b|-|substr|..] == substr;
        }

    static method containtsCharsInString(a: string, banned: string) returns (r: bool)
    {
        r := exists bannedChar :: bannedChar in banned && bannedChar in a;
    }

    method RegisterUser(user: User)
    returns (res: bool)
    modifies this
    {
        var unique := isUnique(user, this.users);
        var correctDomain := strictEndsWith("@knu.ua", user.Email);
        var correctChars := containtsCharsInString(user.Email, "/[]{} \\");
        if(unique || correctDomain || correctChars){
            return false;
        }
        this.users := this.users + {user};
        res := user in this.users;
    }
```

Рисунок 12. Dafny-код класу Program

Верифікація. Для верифікації коду було використано Code Contracts.

```
1 reference | 1/1 passing
public Algorithm(IEnumerable<Questionnaire> _questionnaires, int minGroupSize, int maxGroupSize)
{
    Contract.Requires(_questionnaires != null);
    Contract.Requires(_questionnaires.Count() > 0);
    Contract.Requires(minGroupSize > 0 && maxGroupSize > 0);
    Contract.Requires(minGroupSize < maxGroupSize);
```

Рисунок 13. Code Contracts у конструкторі класу алгоритму розподілу

```
1 reference
public List<Chromosome> Crossover(List<Chromosome> parents)
{
    Contract.Requires(parents.Count >= 2);
    var children = new List<Chromosome>();
    Contract.Ensures(children.Count == (Population.Count - toNextGen));
```

Рисунок 14. Перед- і післяумови в алгоритмі розподілу

ТЕСТУВАННЯ

Тестування проводилося за стратегією Bottom-Up. Її було обрано з двох причин:

1. Стратегія підходить для поступового опанування методів та технологій тестування недосвідченим тестувальником: елементи системи тестуються в порядку зростання їх складності та, відповідно, складності написання тестів для них;
2. Під час розробки системи елементи найнижчого рівня переважно розроблялися першими, і використання цієї стратегії давало можливість відразу проводити їх тестування.

Було проведено модульне тестування контролерів та алгоритму розподілу зі сторони Backend та декількох компонентів зі сторони Frontend.

Також було розроблено класи тестів інтеграції, проведено тестування інтеграції.

Проведено мануальні тести системи і остаточні тести. Відповідність ТЗ визначалася за допомогою мануальної перевірки і опитувань учасників проєкту.

Тестування бекенду. Для тестування бекенду було використано бібліотеку Xunit, бібліотеку Moq та засоби тестування ASP.NET Core. Тести було виділено в окремий проєкт, для тестування кожного з контролерів розроблено окремий клас.

Приклади модульних тестів:

```
#region GET tests
[Fact]
0 references
public async System.Threading.Tasks.Task StudentsGetSuccesfull()
{
    //Arrange
    var context = TestingUtilities.CreateInMemoryDatabaseContext("TestDatabaseGetStudents");
    var data = new List<Student>
    {
        new Student() { Id=1},
        new Student() { Id=2},
        new Student() {Id=3}
    };
    context.Students.AddRange(data);
    context.SaveChanges();
    var cont = new StudentsController(context);
    //Act
    var result = await cont.GetStudents();
    var resValue = result.Value;
    var resResult = result.Result;
    //Assert
    Assert.Equal(data, resValue);
    Assert.Null(resResult);
    context.Database.EnsureDeleted();
}
```

```
[Fact]
0 references
public async System.Threading.Tasks.Task UserRegistrationSuccesfull()
{
    var controllerContext = CreateControllerContext(isAuth: false);
    // Use a clean instance of context for the test
    var context = CreateDatabaseContext("TestDatabaseLogin1");
    var userController = new UsersController(context)
    {
        ControllerContext = controllerContext
    };
    //Act
    var result = await userController.Register(new User { Id = 5, Email = "new.email@knu.ua", Password = "1", Role = UserRole.Student });
    //Assert
    Assert.IsType<CreatedAtActionResult>(result.Result);
    context.Database.EnsureDeleted();
}
```

Приклади тестів інтеграції:

```
public async Task Get_LoggedInUser_ShouldReturnDTO()
{
    // Arrange
    var url = "/api/users/logged_in";
    using var scope = base._factory.Services.CreateScope();
    // Arrange
    var client = _factory.WithWebHostBuilder(builder =>
    {
        builder.ConfigureTestServices(services =>
        {
            services.AddAuthentication("Test")
                .AddScheme<AuthenticationSchemeOptions, TestAuthHandler>(
                    "Test", options => { });
        });
    })
        .CreateClient(new WebApplicationFactoryClientOptions
        {
            AllowAutoRedirect = false,
        });
    client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Test");
    var dbContext = scope.ServiceProvider.GetRequiredService<IntegratedCourseSystemContext>();
    var baseAddress = _factory.Server.BaseAddress;
    var request = new HttpRequestMessage()
    {
        Method = HttpMethod.Post,
        RequestUri = new System.Uri(baseAddress, url),
    };
    // Act
    var response = client.SendAsync(request).Result;
    var content = await response.Content.ReadAsStringAsync();
    // Assert
    response.EnsureSuccessStatusCode();
    Assert.Equal("{\"id\":1,\"role\":2,\"email\":\"a@knu.ua\",\"name\":\"Andrii\",\"surname\":\"Kliachkin\"}", content);
}
[Fact]
0 references
public async Task Login_RegisteredUser_ShouldReturnDTO()
{
    // Arrange
    var url = "/api/users/login";
    using var scope = this._factory.Services.CreateScope();
    var dbContext = scope.ServiceProvider.GetRequiredService<IntegratedCourseSystemContext>();
    var baseAddress = _factory.Server.BaseAddress;
    // Register a user
    var newUser = new DataBase.Models.User()
    {
        Email = "asdfw@knu.ua",
        Password = Convert.ToBase64String(Encoding.UTF8.GetBytes("5")),
        Role = DataBase.Models.UserRole.Student
    };
    var jsonStudent = JsonConvert.SerializeObject(newUser);
    var registerRequest = new HttpRequestMessage()
    {
        Method = HttpMethod.Post,
        RequestUri = new Uri(baseAddress, "/api/users/register"),
        Content = new StringContent(jsonStudent, Encoding.UTF8, "application/json"),
    };
    var registerResponse = httpClient.SendAsync(registerRequest).Result;
    registerResponse.EnsureSuccessStatusCode();
    // Try to log in
    var request = new HttpRequestMessage()
    {
        Method = HttpMethod.Post,
        RequestUri = new Uri(baseAddress, url),
        Content = new StringContent(jsonStudent, Encoding.UTF8, "application/json"),
    };
    // Act
    var response = httpClient.SendAsync(request).Result;
    var content = await response.Content.ReadAsStringAsync();
    // Assert
    response.EnsureSuccessStatusCode();
}
```

На прикладі двох тестів інтеграції можна побачити використання стратегії Bottom-Up: спочатку тестується реєстрація користувача, потім за допомогою реєстрації тестується вхід в систему вже зареєстрованого користувача.

Тестування фронтенду. Тестування фронтенду здебільшого проводилося мануально, але було написано модульні тести для деяких компонентів. Було використано бібліотеки Jest та Enzyme.

Приклад баг-репорту після мануального тестування:

Тема: учню доступна кнопка «Додати завдання» на сторінці групи.

Опис: користувачу, що авторизований як учень, доступна кнопка «Додати завдання» над переліком завдань на сторінці групи.

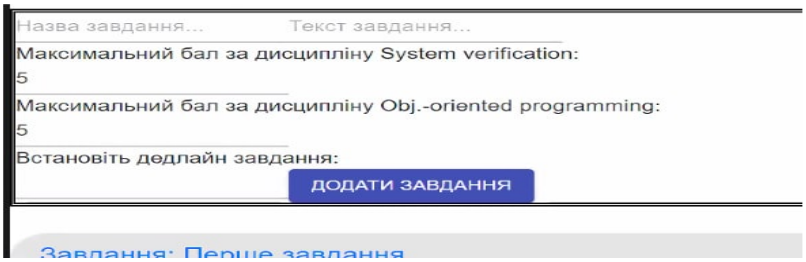
Як відтворити:

1. Зайти на сайт проекту.
2. Увійти в систему як учень.
3. Зайти на сторінку групи, що має хоча б одне завдання.
4. Звернути увагу на низ сторінки над переліком завдань.

Реальний результат: користувачеві, що авторизований як учень, відображається кнопка «Додати завдання».

Очікуваний результат: користувачеві, що авторизований як учень, не має відображатися кнопка «Додати завдання».

Прикріплені файли:



Приклади автоматичного тестування:

```
describe("RegistrationForm interaction", () => {
  let wrapper: any, mockStore, initialState, store, mockSubmit;
  beforeEach(() => {
    mockSubmit = jest.fn();
    mockStore = configureStore();
    initialState = {}
    store = mockStore(initialState);
    wrapper = enzyme.mount(<Provider store={store}><RegistrationForm/></Provider>);
  });

  it("Shows passwords don't match error message", () => {

    expect(wrapper.findWhere((n: any) => n.text() === 'Passwords don\'t match!')).toHaveLength(0);

    const textFields = wrapper.find(TextField);
    const emailInput = textFields.at(0).find('input');
    emailInput.simulate('change', {target: {value: 'a@knu.ua'}});
    const pwdInput = textFields.at(1).find('input');
    pwdInput.simulate('change', {target: {value: '1'}});
    const pwdRepeatInput = textFields.at(2).find('input');
    pwdRepeatInput.simulate('change', {target: {value: '2'}});

    wrapper.find('form').simulate('submit');
    expect(wrapper.findWhere((n: any) => n.text() === 'Passwords don\'t match!'));
  });
});
```

```
test('regex test on @knu.ua', () => {
    expect(validateEmail("andrew@knu.ua")).toBe(true);
    expect(validateEmail("andrew.kl@knu.ua")).toBe(true);
    expect(validateEmail("derevianchenko.....@knu.ua")).toBe(true);
    expect(validateEmail("rr@knu.ua")).toBe(true);
    expect(validateEmail("as df@knu.ua")).toBe(false);
    expect(validateEmail("@knu.ua")).toBe(false);
  });

test('regex test on not @knu.ua', () => {
    expect(validateEmail("andrew@gmail.com")).toBe(false);
  });
```

РЕЗУЛЬТАТИ РОЗРОБКИ

Загальні результати розробки. В процесі розробки було реалізовано:

- 22 базових класи моделей та 2 додаткових для безпечної передачі даних про користувача по мережі та визначення його ролі
- 21 клас REST Web API контролерів
- 24 React-компоненти для відображення/логіки користувацького інтерфейсу
- 31 Typescript-файли для зберігання state користувацького інтерфейсу та відправки даних на сервер
- 1 клас розширення функціональності інтерфейсу IServiceCollection
- 5 класів міграції Entity Framework Core
- 1 клас контексту бази даних
- 2 інфраструктурних класи ASP.NET Core Startup.cs, Program.cs
- 11 класів тестів бекенду, в яких в свою чергу реалізовано
    - 8 тестів інтеграції
    - 50 Unit-тестів, з них:
        - 49 тестів Web API
        - 1 тест генетичного алгоритму розподілу.

Для розробки було використано наступні технології:

- .NET 5
- ASP.NET Web API
- React.js
- Redux
- Typescript
- Coverlet.selector
- XUnit
- PostgreSQL
- Entity Framework Core
- Material-UI

Робота з базою даних. База даних використана для створення інтегрованого курсу знаходиться на одному з серверів Heroku. Heroku – це хмарна PaaS-платформа, що підтримує ряд мов програмування, це одна з перших хмарних платформ. Програми, що працюють на Heroku, використовують DNS-сервер Heroku (зазвичай додатки мають доменне ім'я виду "ім'я_додатку.herokuapp.com"). Для кожної програми виділяється кілька незалежних

віртуальних процесів, які називаються "dynos". Вони розподілені по спеціальній віртуальній сітці ("dynos grid"), яка складається з декількох серверів.

Heroku також підтримує систему контролю версій Git та Docker.

У інструкції описано, як розгорнути базу даних на сервісі Heroku. Якщо ви користуєтесь іншим сервісом, у більшості випадків, інструкія не буде мати великої кількості відмінностей від тої, яку ви читаєте зараз.

1. Створіть аккаунт Heroku та свій додаток з базою даних Postgres. Більше деталей за посиланням: https://dev.to/prisma/how-to-setup-a-free-postgresql-database-on-heroku-1dc1

Будь ласка, якщо ви користуєтесь іншим провайдером, зверніться до його документації для того, щоб виконати розгортання.

2. Клонуйте репозиторій проєкту з сервісу Github на ваш комп'ютер у інтерфейсі командного рядка з допомогою команди: git clone https://github.com/ukitta555/IntegratedCourseSystem.git

3. Запустіть скрипт Database.py, змінивши рядок підключення до бази даних (який ми отримали на 1 кроці) в скріпті на той, що ви отримали з сервісу Heroku.

4. База даних створена на хмарному сервісі і готова до використання у програмі.

Структура бази даних за замовчуванням зберігається у файлі DBManipulations/createDB.sql, за необхідності можна її змінювати, але якщо зміни внесено після розгортання бази даних, перед перерозгортанням потрібно виконати запит з файлу DBManipulations/deleteDB.sql

Back-end. Для розробки серверної частини була використана технологія .NET Core Web API, яка приймає запити з клієнтської частини з допомогою REST API контролерів, оброблює їх, надсилає клієнту відповідь і, якщо потрібно, робить запити до БД.

Більшість класів, які використовує система, це вище згадані контролери та класи-моделі для ORM-фреймоворку EF Core, який спрощує роботу з БД для розробників.

Приклади коду моделі та REST API контролера:

```
public partial class Questionnaire
{
    public Questionnaire()
    {
        RolePreferences = new HashSet<RolePreference>();
        StudentRolePeriods = new HashSet<StudentRolePeriod>();
        SubjectQuestionnaires = new HashSet<SubjectQuestionnaire>();
        TeammateAntipreferences = new HashSet<TeammateAntipreference>();
        TeammatePreferences = new HashSet<TeammatePreference>();
        TechPreferences = new HashSet<TechPreference>();
    }

    public int Id { get; set; }
    public int StudentId { get; set; }
    public int ClassId { get; set; }

    public virtual Class Class { get; set; }
    public virtual Student Student { get; set; }
    public virtual ICollection<RolePreference> RolePreferences { get; set; }
    public virtual ICollection<StudentRolePeriod> StudentRolePeriods { get; set; }
    public virtual ICollection<SubjectQuestionnaire> SubjectQuestionnaires { get; set; }
    public virtual ICollection<TeammateAntipreference> TeammateAntipreferences { get; set; }
    public virtual ICollection<TeammatePreference> TeammatePreferences { get; set; }
    public virtual ICollection<TechPreference> TechPreferences { get; set; }
}
}
```

```
namespace IntegratedCourseSystem.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UsersController : ControllerBase
    {
        private readonly IntegratedCourseSystemContext _context;

        public UsersController(IntegratedCourseSystemContext context)
        {
            _context = context;
        }

        [HttpPost]
        [Route("logged_in")]
        public async Task<ActionResult<UserDTO>> GetInfoOnEnteringSite()
        {
            if (User.Identity.IsAuthenticated)
            {
                UserDTO userDTO = new UserDTO();
                var userByEmail = await _context
                    .Users
                    .FirstOrDefaultAsync(entry => entry.Email == User.Identity.Name);


                if (userByEmail.Role == UserRole.Student)
                {
                    var student = await  context
```

```
                if (!ModelState.IsValid)
                {
                    return BadRequest(ModelState);
                }


                _context.Entry(user).State = EntityState.Modified;

                await _context.SaveChangesAsync();

                return new ObjectResult(user);
            }
            else
            {
                return BadRequest(ModelState);
            }
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<UserDTO>>> GetUsers()
        {
            return await _context
                .Users
                .Select(item => ItemToDTO(item))
                .ToListAsync();
        }
```

Front-end. Для розробки клієнтської частини було використано стек React/Redux/Material-UI. Головна для цього причина – бажання зробити додаток у стилі Single Page Application, коли клієнтський інтерфейс відображається без перезавантажень сторінки.

Технологія Redux зав’язана на архітектурному патерні Flux, який моделює обіг даних в одному напрямку. Він допомагає відокремити логіку інтерфейсу від логіки передачі даних та їх обробки перед тим, як представити користувачу.

Було створено багато service-файлів, які відповідають за зв’язок між клієнтським інтерфейсом та серверною частиною.

Приклади файлів-компонентів, які відповідають за інтерфейс:

```
import rolePreferenceService from '../../services/rolePreferenceService'
import techPreferenceService from '../../services/techPreferenceService'
import teammatePreferenceService from '../../services/teammatePreferenceService'
import enemyPreferenceService from '../../services/enemyPreferenceService'
import { useHistory } from 'react-router'
import { updateCourseRegStatus } from '../../reducers/userReducer/userActionCreators'


const CourseRegistrationForm = () => {
  const NOT_SELECTED = -1
  const user = useSelector ((state: {user: UserState}) => state.user)
  const dispatch = useDispatch()
  //const history = useHistory()

  const [name, setName] = useState<string>(user.name ? user.name :  "")
  const [surname, setSurname] = useState<string> (user.surname ? user.surname : "")
  const [classId, setClassId] = useState<number | null> (user.currentCourseId ? user.currentCourseId : null)

  const [classSubjects, setClassSubjects] =  useState<ClassSubject[]> ([])

  const [selectedFriends, setSelectedFriends] = useState<Student[]> ([])
  const [selectedEnemies, setSelectedEnemies] = useState<Student[]> ([])

  const [subjectsChecked, setSubjectsChecked] = useState<SubjectPreference[]> ([])
  const [rolePreferences, setRolePreferences] = useState<RolePreference[]> ([])
  const [techPreferences, setTechPreferences] = useState<TechPreference[]> ([])

  const [classStudents, setClassStudents] = useState<Student[]>([])

  const handleSubmit = async (event: React.FormEvent<HTMLFormElement>) => {
    event.preventDefault()

    const questionnaireForCourse = await questionnaireService.getQuestionnaire({classId: classId, studentId: user.id})
    const prefSubjectsRequest = subjectsChecked.map (sub => {
      return {
        classSubjectId: sub.id,
        questionnaireId: questionnaireForCourse.id
```

```
return (
    <ThemeProvider theme={light}>
    <form onSubmit = {onSubmit} style={{padding: `0 ${spacing * 4}px`}}>
      <Grid container style={GridWrapperStyle} justify="space-around" spacing={spacing}>
        <Grid item xs={5}>
          <Box maxWidth="lg" bgcolor="" style={documentIconStyle} children={false} />
        </Grid>
        <Grid item container justify="center" alignItems="center" xs={5} style={titleTextWrapperStyle}>
          <Typography component="h6" style = {titleTextStyle}>
            Створення нового інтегрованого курсу
          </Typography>
        </Grid>
        <Grid item xs={5}>
          <Box maxWidth="lg" bgcolor="theme_grey.main" style={pythonIconStyle} children={false} />
        </Grid>
        <Grid item xs={5}>
          <Box color="theme_black.main">
            <p>Мови програмування</p>
            <Box bgcolor="theme_green.main" style={textFieldsWrapperStyle}>
              <Grid container direction="column" justify="center" alignItems="center">
                {languages.map( item =>
                    <DeletableTextField
                        onDelete={handleDeleteListItem(languages, setLanguages)(item.id)}
                        id={`lang-${item.id}`}
                        key={item.id}
                        value = {item.value}
                        onChange = {
                          (event: React.ChangeEvent<HTMLInputElement>) => setLanguages(
                              languages.map(language =>
                                  language.id !== item.id
                                      ? language
                                      : {value: event.target.value, id: language.id}))
                        }
```

Приклад сервіс-файлу для відправки HTTP-запитів до REST API:

```
const addTechs = async (requestData: {teches: {name: string}[]}) => {
  try {
    const promisesArray = requestData.teches.map (tech =>
      axios.post (
        baseURL,
        tech,
        {withCredentials : true}
      )
    )
    const response = await Promise.all(promisesArray)
    return response.map (r => r.data)
  }
  catch (error) {
    console.log (error.response.data)
    return []
  }
}

const getTech = async (techId: number) => {
  try {
    const response = await axios.get (
        `${baseURL}/${techId}`,
        {withCredentials : true}
      )
    return response.data
  }
  catch (error) {
    console.log (error.response.data)
    return null
  }
}


export default {
  addTechs,
  getTech
```

Генетичний алгоритм. Для розподілення студентів за групами було розроблено генетичний алгоритм. Вхідними даними алгоритму є список об’єктів класу Questionnaire -

результатів опитування студентів під час запису на курс. Результатом роботи алгоритму є розподіл за групами зазначеного розміру з урахуванням побажань кожного зі студентів.

Алгоритм створює фіксовану кількість випадкових розподілів по групам, після чого починається процес:

- Обираються найкращі варіанти розподілу
- З них утворюється наступне покоління розподілів
- Кожна група з наступного покоління має шанс "мутувати" – зазнати слабкої зміни якогось параметру
- Групи перевіряються на наявність "жорстких" порушень – замалий чи завеликий розмір групи чи кількість технологій у групі
- Попереднє покоління, за винятком фіксованої кількості найкращих особин, замінюється новим
- Процес починається спочатку.

Експериментально було визначено, що 200 ітерацій є оптимальною кількістю.

Також для роботи алгоритму було розроблено два допоміжні класи: Chromosome і GroupWithQuestionnaires.

Приклади коду:

```
1 reference
private List<Chromosome> SelectForReproduction()
{
    Contract.Requires(Population.Count > 0);
    var parents = new List<Chromosome>();
    Contract.Requires(parents.Count == (Population.Count - toNextGen)*2);

    double minFitness = Population.Min().Fitness;
    double toAdd = minFitness < 0 ? minFitness : 0;
    double sumFitness = Population.Select(x => x.Fitness + toAdd).Sum();
    var wheel = new List<Tuple<double, Chromosome>>();
    foreach (var s in Population)
    {
        wheel.Add(new Tuple<double, Chromosome>((s.Fitness + toAdd) / sumFitness, s));
    }
    for (int i = 0; i < (Population.Count - toNextGen) * 2; i++)
    {
        var d = rng.NextDouble();
        var upper = 0.0;
        for (int j = 0; j < wheel.Count; j++)
        {
            upper += wheel[j].Item1;
            if (d < upper)
            {
                parents.Add(wheel[j].Item2);
                break;
            }
        }
    }
    return parents;
}
```

```
2 references
public void Fix(Chromosome chromosome)
{
    foreach (var groupToFix in chromosome.Groups)
    {
        while(groupToFix.students.Count > maxStudentsInGroup)
        {
            var groupsBySizeAsc = chromosome.Groups.OrderBy(g => g.students.Count).ToList();
            var studentToGive = groupToFix.students[rng.Next(groupToFix.students.Count)];
            var qToGive = groupToFix.questionnaires.First(q => q.Student == studentToGive);
            groupToFix.RemoveStudent(studentToGive);
            groupsBySizeAsc[0].AddStudentFromQuestionnaire(qToGive);
        }
        while (groupToFix.students.Count < minStudentsInGroup)
        {
            var populatedGroups = chromosome.Groups.Where(g => g.students.Count > minStudentsInGroup);
            if (populatedGroups.Count() == 0)
            {
                throw new Exception("Not enough students to form groups!");
            }
            var groupToTakeFrom = populatedGroups.ElementAt(rng.Next(populatedGroups.Count()));
            var newStudent = groupToTakeFrom.students.First(s => !groupToFix.students.Contains(s));
            var newQuestionnaire = groupToTakeFrom.questionnaires.First(q => q.Student == newStudent);
            groupToTakeFrom.RemoveStudent(newStudent);
            groupToFix.AddStudentFromQuestionnaire(newQuestionnaire);
        }
    }
}
```

```
1 reference | 1/1 passing
public List<Group> Run(Class @class)
{
    Population = RandomInitialize(questionnaires);
    foreach(var c in Population)
    {
        Fix(c);
    }
    for(int gen=0; gen < 200; gen++)
    {
        var parents = SelectForReproduction();
        var children = Crossover(parents);
        foreach (var ch in children)
        {
            Fix(ch);
        }
        foreach (var child in children)
        {
            Mutate(child);
        }
        Population.Sort();
        for(int i = 0; i < populationSize - toNextGen; i++)
        {
            Population[i] = children[i];
        }
    }
    List<Group> res = new List<Group>();
    foreach(var group in Population.Max().Groups)
    {
        res.Add(group.ToDbGroup(@class));
    }
    return res;
}
```

Інструкція користувача

Для створення курсу необхідно
- Увійти в систему як вчитель
- Натиснути на кнопку "Створити курс"
- Заповнити Інформацію про курс
- Натиснути кнопку "Створити"

Для запуску розподілу для курсу необхідно:
- Увійти в систему як вчитель. У вас має бути створений хоча б один курс
- Натиснути на кнопку "Запустити розподіл"

Для реєстрації на курс як учень необхідно:
- Увійти в систему як учень
- Зайти на сторінку "Додати курс"
- Ввести необхідну інформацію про свої побажання
- Натиснути на кнопку "Далі"

Для створення нового завдання необхідно:
- Увійти в систему як вчитель. У вас має бути створений хоча б один курс
- Перейти на сторінку курсу (посилання можна знайти на сторінці з усіма курсами)
- Перейти на сторінку групи, для якої треба створити завдання
- Ввести текст завдання, дедлайн, та максимальну оцінку
- Натиснути на кнопку "Додати"

Для того, щоб залишити коментар до завдання:
- Увійти до системи
- Перейти до сторінки з завданням (Сторінка з курсами => сторінка курсу(для вчителя) => сторінка групи => сторінка завдання)
- Ввести коментар
- Натисунти кнопку "Додати".

ПРОБЛЕМИ, ПОМИЛКИ, ПЛАНИ РОЗРОБКИ

Помилки та проблеми під час розробки проєкту

– Вибраний стек технологій не був достатньо засвоєний на початку проєкту – довелось вчити паралельно з написанням проєкту.

– Переоцінка своїх можливостей – на етапі розробки прототипу проєкту було спроєктовано більше можливостей, ніж можна зробити за відведений проміжок часу.

– Не була проведена ретельна перевірка відповідності дизайну БД до прототипу – довелося змінювати дизайн системи та БД під час розробки.

– Через помилку вище довелось розробляти одну з сторінок з самого початку.

– Зроблений на початку проєкту розподіл за ролями виявився незручним для розробників – довелося під час проєкту його змінювати.

– Взаємодія з незнайомими бібліотеками – додаткові витрати часу на конфігурацію, а саме: налаштування code coverage для оцінки кількості протестованого коду.

– Обраний стандартний формат проведення Daily Meetings не підходив для цього проєкту та був змінений та адаптований для нашого проєкту.

– Час, відведений на розробку проєкту часто дорівнює одній парі на тиждень, адже очікувалось, що написання проєкту буде у інший вільний час, але не було враховано що інші предмети потребують той самий вільний час також, тому іноді через велике навантаження з інших дисциплін не було можливості розробляти проєкт. Очікуємо що в наступному році учням виділятиметься більше часу або складність проєкту та кількість обов'язкових завдань буде значно зменшена.

– Вільний час у учасників команди не завжди співпадав для розробки та спілкування.

– На початку роботи над проєктом не були чітко поставлені вимоги, не було чіткого розуміння що саме вимагається, як буде оцінено тощо.

– Іноді через перевантаження виникали проблеми з мотивацією (для 2-го курсу написати лабу і брати участь у проєкті – це зовсім незрівнянне навантаження, у той час, для 4-го курсу виділяти так багато часу на проєкт, маючи дипломну роботу, також не зручно, адже доводиться робити вибір в сторону диплому (дедлайни для дипломної та проєкту майже співпадають).

– Брак необхідного досвіду вимагає більше часу на розробку/тестування з першого погляду легких речей.

– Життєві обставини (хвороби, свята) впливали на продуктивність всієї команди.

– Не всі лекції у 4-го курсу співпадають з тим, що необхідно зробити в проєкті.

Плани подальшої розробки проєкту. Оскільки даний груповий проєкт є обмеженим в часі, розроблений веб-застосунок є не повною реалізацією усіх ідей. В зв'язку з цим, було розроблено план для подальшої розробки:

– Додати захист сайту від атак, підвищення рівня безпеки сайту, перевірка на уразливість
– Темна тема
– Мануальна зміна результатів алгоритму
– Рефакторинг коду
– Інтеграція з Github
– Адмін-панель
– Можливість додавати файли до створених завдань
– Інтеграція з Google Drive
– Влаштована Kanban дошка
– Трекер часу
– Зміна мови інтерфейсу
– Аналітика сайту
– Утиліта для ретроспектив
– Інтеграція з Discord-каналом
– Окремий модуль для змагань (хакатони)
– Збільшити кількість налаштувань профілю
– Нейтралітет до високих навантажень
– Оптимізувати кількість часу на обробку кожного з запитів до серверу та БД
– Можливість взаємодії команд
– Більш широка аудиторія (не лише для розробників)

ВИСНОВКИ

Кожен із учасників команди навчився багато новому та поглибив свої знання. У висновках ми б хотіли детальніше розрики аспекти, на які повпливав проєкт для кожного.

Для Владислава це був перший проєкт такого розміру, і він виділив для себе наступні переваги:

- Досвід роботи у команді
- Важливість проєктування на початкових стадіях проєкту
- Навчився як працювати з новими технологіями у майбутньому
- Відпочинок має таку ж важливість, як і робота.

Едуард виділив для себе наступні переваги:

- Відточення навичок роботи з обраними технологіями
- Навички працювати з великими проєктами
- Досвід роботи в команді
- Досвід роботи з інструментами для роботи в команді.

Ярослав набув наступних навичок:

- Досвід роботи зі студентами старших курсів в одній команді
- Досвід роботи з методологіями розробки програмного забезпечення
- Досвід роботи з технологіями
- Кілька пунктів для CV
- Досвід проєктування програмного забезпечення великих розмірів.

Андрій зазначив наступні пункти як переваги для себе:

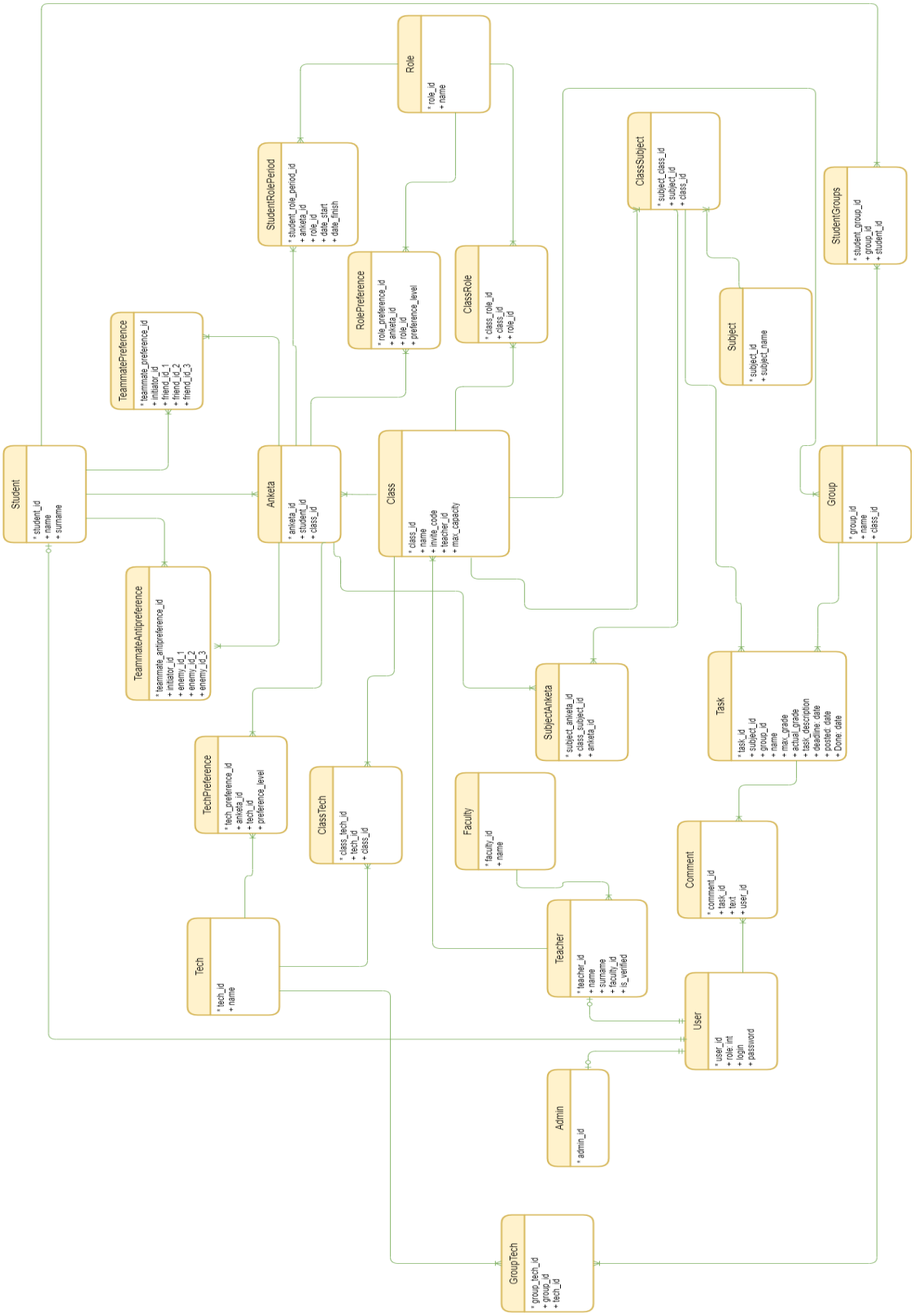
- Покращення навичок командної роботи
- Поглиблення знань з технологій, що використовуються у тестуванні
- Досвід роботи у ролі тестувальника
- Знайомство з новими технологіями (Dafny, деякі технології з front-end проєкту).

Карина виділила для себе наступні переваги:

- Нові знайомства
- Поглиблення навичок керівника (управління, планування, комунікація)
- Покращення навичок написання технічної документації
- Покращення організаторських навичок
- Додала впевненість у знаннях та силах
- Покращення знань з написання звітностей, створення презентацій, ведення зустрічей (Daily, Retro, Planning)

Отже, кожен учасник проєкту отримав для себе новий досвід та покращив свої знання та навички у різних сферах.

ДОДАТОК А Діаграма бази даних



ДОДАТОК Б Завершені задачі за перший спрінт

Sprint 1 finished

Примечание

- [x] Написати очікування Feb 17
- [x] Уточнена постановка задачі
- [x] Заповними 1 етап для Команда 1
- [ ] Дизайн адміністративної панелі Feb 17
- [ ] Дизайн елементів з дошки (завдання, оголошення...) Feb 17
- [ ] Дизайн кабінету команди Feb 17
- [ ] Розібратись шо треба з тестування Feb 17
- [ ] Сформувати Roadmap
- [ ] Дизайн кабінету вчителя Feb 17
- [ ] Продумати кабінет учня + функції Feb 17
- [ ] Дизайн для редагування інформації учнем Feb 17
- [ ] Продумати кабінет вчителя + функції Feb 17
- [ ] Дизайн кабінету учня Feb 17
- [ ] Дизайн дошки вчителя Feb 17
- [ ] Спроектувати БД для проекту Feb 17
- [x] Дизайн стартової сторінки Feb 17
- [ ] Дизайн сторінки регістрації(для усіх користувачів) Feb 16
- [ ] Дизайн повідомлення про очікування Feb 17
- [ ] Дизайн анкети вчителя Feb 17
- [ ] Дизайн анкети учня Feb 17

ДОДАТОК В Завершені задачі за другий спрінт

ДОДАТОК Г Завершені задачі за третій спрінт

Sprint 3 finished

(03.03.2021 - 17.03.2021)

- [x] Залить на Гитхаб все картинки из Фигмы (для общего доступа) Mar 17
- [x] Застосувати тему до хедера
- [x] Застосувати кольорову тему до сторінки логіну
- [x] Застосувати кольорову тему до сторінки реєстрації
- [x] Тестування анкети вчителя Mar 17
- [x] Тестування бекенду анкети учня
- [x] Тестування бекенду користувачів
- [x] Тестування бекенду анкети вчителя
- [x] Тестування кабінету учня Mar 17
- [x] Верифікація на Дафні
- [x] Тестування кабінету вчителя Mar 17
- [x] Тестування анкети учня Mar 17
- [x] Написати розділи для глобального звіту
- [x] Написати половину глобального звіту
- [x] Перенести ретроспективи в Jamboard
- [x] Написати міграції для створення суміжних таблиць в БД
- [x] Бекенд анкети учня Mar 17
- [x] Бекенд екрана очікування учня Mar 17
- [x] Бекенд кабінету вчителя Mar 17
- [x] Бекенд акнекти вчителя Mar 17
- [x] Бекенд акнекти вчителя Mar 17
- [x] Бэкенд для обработки анкеты ученика Mar 17
- [x] Бекенд екрану очікування викладача Mar 17
- [x] Бэкенд для обработки анкеты учителя Mar 17
- [x] Додати endpoint до арі для витягування курсів для конкретного вчителя
- [x] Бекенд сторінки курсу Mar 17
- [x] Фронтенд анкети Mar 17
- [x] Сторінка з переліком курсів вчителя - показувати лише ті курси, які цей вчитель створив
- [x] Редіректи на потрібну сторінку після логіну для вчителя
- [x] Фронтенд сторінки створення курсу
- [x] Зробити кольорову тему
- [x] Знайти документацію на Dafny
- [x] Вивчити документацію на Dafny
- [ ] Написати майбутні плани розробки в звіті
- [x] Привести хедер до задуманого дизайну Mar 17
- [x] Змінити структуру звіту, змінити розділи, додати нові
- [x] Демо для вчителів 2шт.
- [x] Пододавати необхідні скріншоти в звіт

ДОДАТОК Д Завершені задачі за четвертий спрінт

Sprint 4 finished

(17.03.2021 - 31.03.2021)

- [x] Створити стиль презентації
- [ ] МАЙЖЕ Закінчити глобалний звіт
- [ ] Перенести всі результати ретро в звіт
- [x] Бекенд форми реєстрації на курс
- [x] Розробка алгоритму вибору команд
- [ ] Бекенд сторінки класу
- [ ] Бекенд кабінету учня Mar 31
- [ ] Бекенд сторінки групи (вчитель)
- [ ] Бекенд компонента завдання Mar 31
- [ ] Бекенд сторінки групи(учень) Mar 31
- [ ] Persistent login (backend)
- [ ] Тести інтеграції для сторінки учня Mar 31
- [ ] Тести інтеграції для сторінки групи Mar 31
- [ ] Тести інтеграції для сторінки викладача Mar 31
- [ ] Тести інтеграції для сторінки класу Mar 31
- [ ] Тести інтеграції для анкети учня Mar 31
- [ ] Тести інтеграції для анкети вчителя Mar 31
- [ ] Тести системи - usability Mar 31
- [ ] Тести системи - performance testing Mar 31
- [ ] Остаточні тести Mar 31
- [ ] Верифікація на Dafny Mar 31
- [x] Тестування БД Mar 24
- [ ] Logout
- [ ] Фронтенд кабінету учня Mar 31
- [ ] Редіректи після логіну на потрібну сторінку
- [ ] Persistent login(frontend)
- [ ] Фронтенд компонента завдання Mar 31
- [ ] Фронтенд сторінки з курсами (вчитель)
- [ ] Фронтенд сторінки з курсами (учень)
- [ ] Фронтенд сторінки класу Mar 31
- [x] Фронтенд сторінки групи(вчитель) Mar 31
- [ ] Фронтенд кабінету вчителя Mar 31
- [ ] Зробити схематично презентацію (1 версія)
- [ ] Додати результати ретро та дошки в звіт
- [ ] Загальний звіт за всім проектом
- [ ] Презентація Power Point онлайн платформи
- [ ] Написати підсумки на ретро
- [ ] Презентація (слова) онлайн платформи

INTEGRATED LEARNING COURSE SUPPORT SYSTEM

Tetiana Zverieva, Maxim Karpenko, Dmytro Kuntso, Lybomyr Maevskiy, Dina Formakidova

GLOSARY

Term	Description
CRM	Customer Relationship Managementa is a technology for managing all your company's relationships and interactions with customers and potential customers.
CRM system	Corporate informational system where all stages of working with customers are automated.
Account	Legal entity with which you cooperate. It can be a customer, partner, supplier, competitor and so on. Company's subsidiaries and company itself are also accounts.
Contact	Individual with which you cooperate. It can be an independent contact or a representative of a company. Company's employees are also contacts.
Activity	Item of the history of cooperation with customer. Activities can be of three types: Email, Call, and Task. Task is a particular detailed action that can be performed during the specified period of time. For example, preparing a document, participating in a meeting, helping client with some questions and doing other activities.
System Administrator	A system administrator has full authority to access all system elements. System administrators form user groups, allocate access rights, and define system settings.
User	Users have limited access rights to system elements. Their access rights are granted by system administrators.

PROJECT GOALS AND BUSINESS TASKS

1. Provide a platform to create, store and manage all user-related information in one place:
   - Contacts and organizations served by the bank
   - Bank employees related data
   - Bank clients related data
   - Bank products and services information
   - Transactions and activities carried out in the form of client-client, as well as client-bank
2. Enable possibility to create, read, update and delete data according to roles responsibility.
3. Provide implementation for different language regions.

ROLES

The following user roles can be assigned in the system:

Role	Acronym	Description
System Administrator	SA	Technical admin of system
Key Customer Representatives manager	KCR	Assist customers in opening new bank accounts, modifying existing accounts and dealing with other tasks related to user needs. Responsible for development assigned accounts.
Customer Representatives manager	CR	Assist customers in opening new bank accounts, modifying existing accounts and dealing with other tasks related to user needs.
Sales	Sa	Telling customers about the bank's products and services.
Clerks	Cl	Maintain financial records for the bank by posting financial transaction, check records for accuracy, reconcile entries and balances and create reports from the data.
Loan Officers	LO	Bank loan officers help applicants complete loan applications, verify the information and make the final decision on loan approval or denial.

USER STORIES

Customer management. Customer section divided into 2 units:

Account Section

Section contains information on the Legal Entities, Company, Partners. It allows to see and edit information of the existing Legal entities and create new Accounts.

Contact Section

Section contains information on employees and other physical persons that can be connected with existing Accounts. Allows to see and edit information on the existing Persons and create new Contacts.

ID	As	I want to	So that / In order to	Acceptance criteria
CM-1	KCR, CR, Sa, Cl, LO	See accounts	Know account information according to access right	Account information is stored in fields and linked details in account page
CM-2	KCR, CR	Add accounts	Store and manage all information about new account in one page	Add values in all required fields and linked details in account page
CM-3	KCR, CR	Edit accounts status	Store up to date status about account data	Trigger validation process after adding new account or change existing account status

ID	As	I want to	So that / In order to	Acceptance criteria
CM-4	KCR, CR, Sa, Cl, LO	See account contacts	Know which contacts linked to Account	See all contacts on detail Contacts in Contacts. Have possibility to open Contact page from account and see contact information
CM-5	KCR, CR	Add account contacts	Linked existing contacts with Account	Add values in all required fields and linked details in account page
CM-6	KCR, CR, Sa, Cl, LO	See responsible CR for account	Know which CR contact linked to Account	See responsible CR managers on basic detail in Account info tab. CR manager is filled in automatically after creating Account
CM-7	KCR	Assign account to managers	Edit responsible manager to account	Edit CR in Account info tab
CM-8	KCR, CR, Sa, Cl, LO	See accounts actions history	See which last had an Account	Account actions such as creating new card, taking loan, changing some data in account are stored in detail History options in Account info tab
CM-9	KCR, CR, Sa	See/Add account communication options	See which interaction channels I have with the Account	Communication options such as email, phone, web details are stored in detail Communication options in Account info tab
CM-10	KCR, CR, Sa	Filter existing accounts and contacts	Find necessary entry	Filter on Account and Contact general tab, that accept search by name, ID, IBAN, phone number

Transaction management.

Transaction section contains all information about completed transactions. Transactions is not editable field. Only Cl can edit transactions as exception.

Type of filtering transactions:

- User transactions
- Bank transactions
- All transactions.

ID	As	I want to	So that / In order to	Acceptance criteria
TM-1	Cl, LO, KCR, CR	See transactions	Know transaction information according to access right	Transaction information is stored in fields and linked details in account page
TM-2	Cl	Create a reverse transaction based on existing	Make a refund	Create new transaction with reversed recipient and sender fields and editable money amount field
TM-3	Cl	Download report by period	Check records for accuracy, reconcile entries and balances and create reports from the data	Select date period and download all transaction records on this period

Card management. Card section contains information about account cards. Managers have opportunity to change statuses

ID	As	I want to	So that / In order to	Acceptance criteria
CdM-1	KCR, CR, CL, LO	See user`s cards	Know cards information according to access right	Cards information is stored and linked details in account page
CdM-2	KCR, CR	Add new card	Store and manage all information about new cards in one page	Add values in all required fields and linked details in account page
CdM-3	KCR, CR	Change status	Store up to date status about cards data	Opportunity to change card status to one from the list (blocked, expired, closed, etc.)

Loan management

ID	As	I want to	So that / In order to	Acceptance criteria
Lo-1	LO, CR, KCR	See bank loans information	Know loans information according to access right	Loans information is stored and linked to account page
Lo-2	LO	Asset new loan for account	Create new connection with account and loan	Loans information is stored and linked to account page

Activity management

ID	As	I want to	So that / In order to	Acceptance criteria
A-1	KCR, CR ,LO, Cl, Sa	See/edit activities	Track time of activity and planning the day	Page with activities filtered by date, type and users

Access rights are configured for the following system objects:

- Account
- Contact
- Transaction
- Card
- Loan
- Activity

The following levels of object access rights are available in the system:

- [Read] – access to view the information.
- [Create] – access to add new records.
- [Edit] – access to change and save record.
- [Delete] – access to delete records.

Role	Account	Contact	Transaction	Card	Loan	Activity
SA	R/E/D	R/E/D	R/E/D	R/E/D	R/E/D	R/E/D
KCR	R/E	R/E	R	R/E	R	R/E/D
CR	R/E	R/E	R	R/E	R	R/E/D
Sa	R	R	-	-	-	R/E/D
Cl	R	R	R/E	R	R	R/E/D
LO	R	R	R	R	R/E	R/E/D

REQUIREMENTS

Functional requirements. General fuctional requirements

- The service is implemented in three languages: English, Russian, Ukrainian.
- Structured interface
- Clear names of elements / infoblocks

Required page fields:

Table describes necessary fields and tabs on pages:

Page	Fields	Tabs
Accounts page	- name - registered date - responsible CR	
Detailed account page	- name - registered date - responsible CR	- Contacts - Transaction - Card - Loan - Activity - History
Contact page	- name - email - phone	
Detailed contact page	- name - email - phone - birthday - address	- Accounts - Activity

Registration & Login pages flow

Authentification service should have 2 controllers: Auth controller and Home controller.

[Login]

When user on Login page enter login and password fields and click [Login] button, auth controller react and send request to Database with user credentials(login,password). The database return userID to AuthController.

IF userID is not null - AuthController send request to HomeController that is redirect user on home page on UI.

IF UserID is null - AuthController return error message to UI.

[Registration]

1. User on Login page click [Registration] button, auth controller react and return registration page on UI.

2. User input required field on Registration page (login, password,email) click register button, auth controller react and send request to Database with user credentials (login,password) to check if user exists and return userID.

IF userID is not null - AuthController return error message that user already exist in system to UI.

IF UserID is null - AuthController send request to Database to create new User. Database return newly created userID on AuthController.

AuthController send email confirmation and return result to itself. AuthController return confirm email message.

When user click on button [Go to HomePage] HomeController react and redirect user to home page on UI.

Requirements for information and software compatibility

Requirements for information structures and solution methods. The database is managed using PostgreSQL.

Requirements for user queries of data from databases. Users work with the database through the Web interface.Users should be able to edit, view and delete data.

Requirements for source codes and programming languages. The service must be implemented using the ASP.NET Core framework.

Requirements for the development of user interface functionality. The development of the UI functionality must be implemented using React.

Requirements for system integration. System integration is carried out using Rest API.

Deployments. Use RDS for Postgres Database deployment. Use EC2 instance (Linux) for hosting web API.

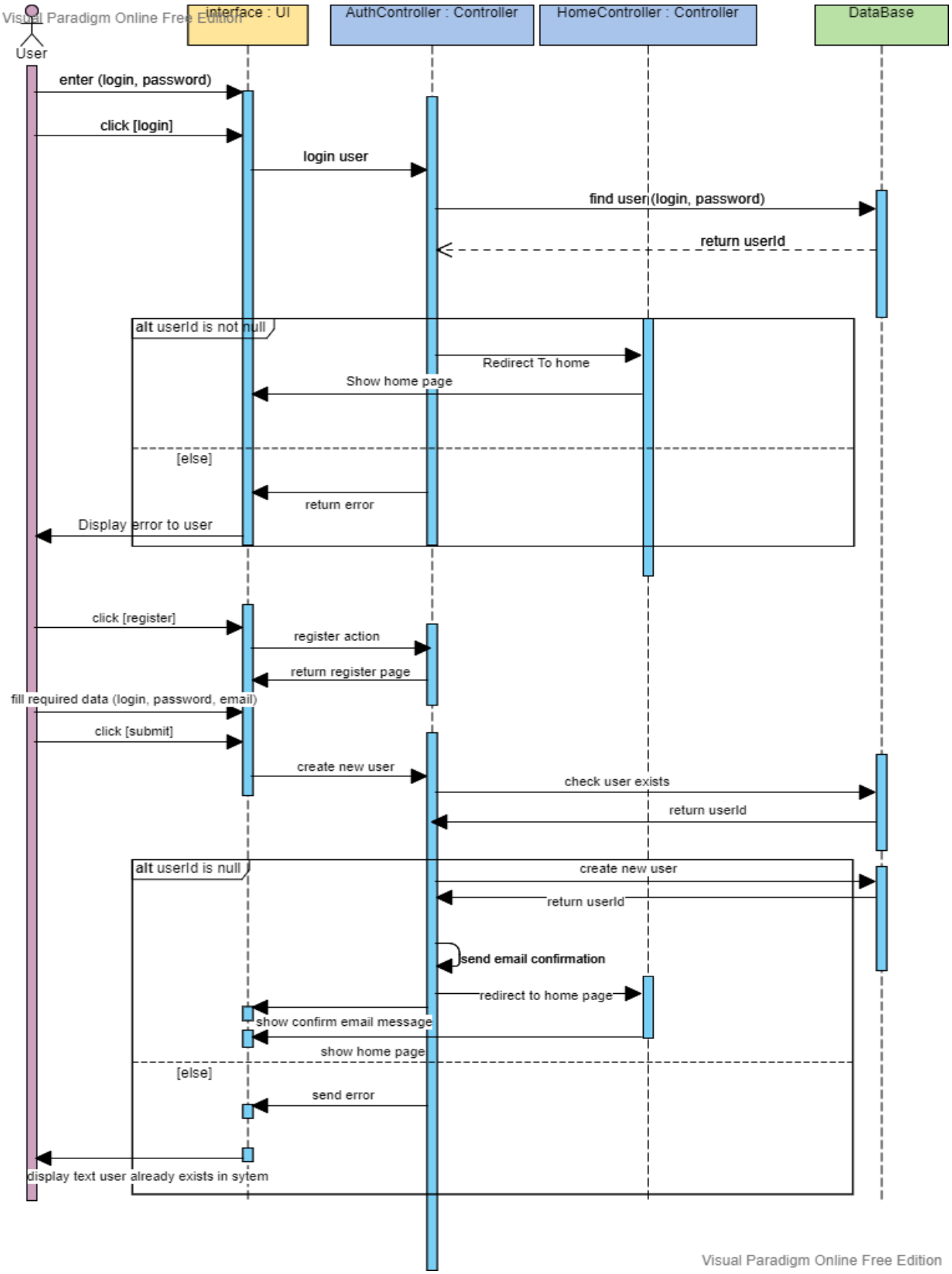


Figure 1. Sequence diagram

Non functional requirements

Design prototype. The design is created at the discretion of the developers, in the absence of specific wishes from the customer.

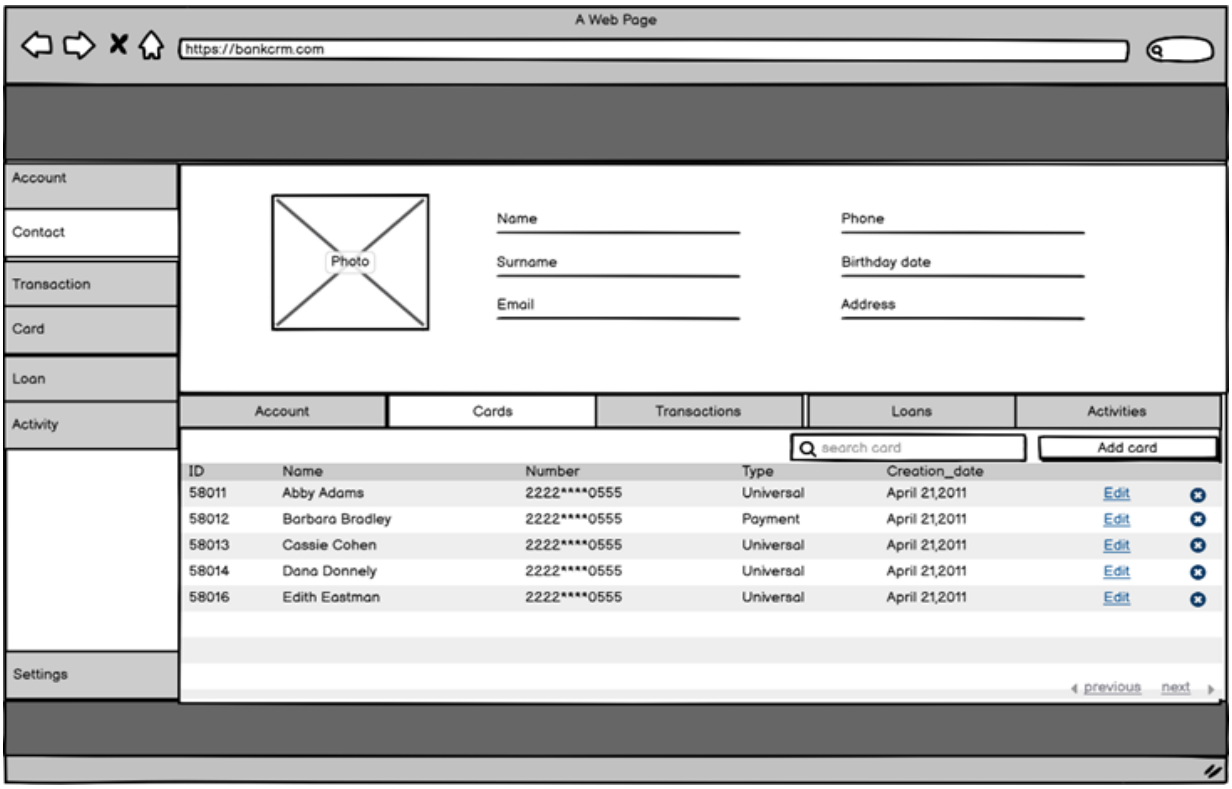


Figure 2. Contacts

ARCHITECTURE & INTEGRATIONS

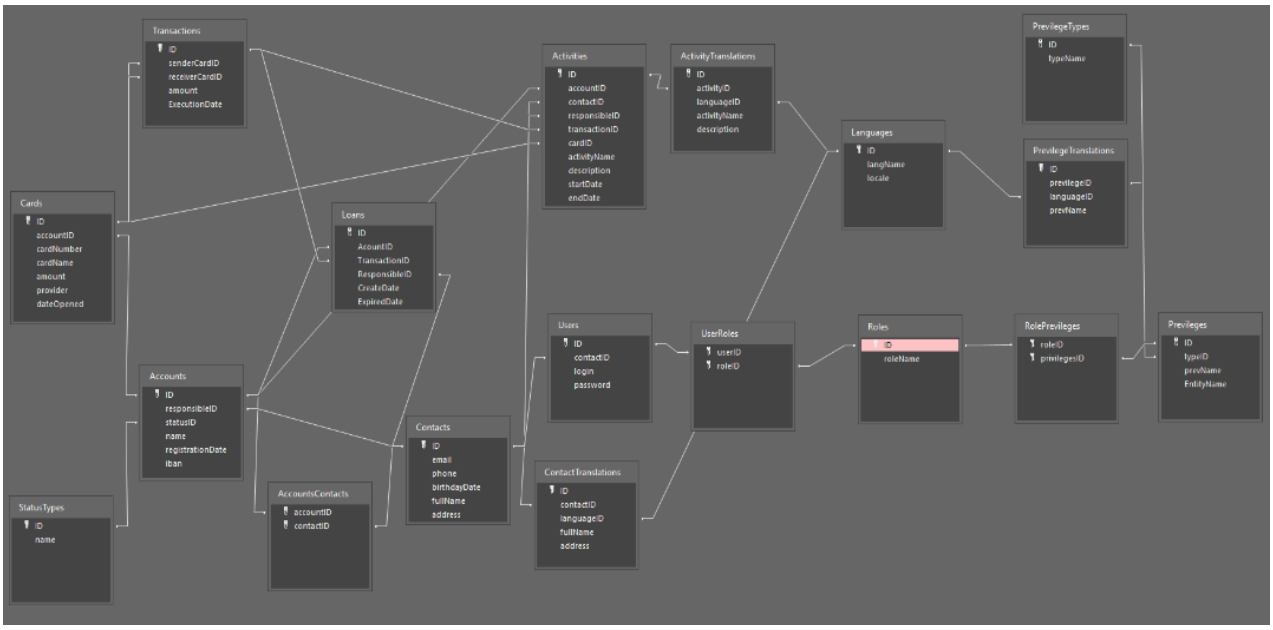


Figure 3. Data model

BACK-END

API: receiving and transmitting data

In order to receive and transmit data from the backend to the frontend we use Swagger.

http://bankcrm-dev.eu-central-1.elasticbeanstalk.com/swagger/index.html

Swagger UI allows users to visualize and interact with the API's resources simply from your OpenAPI Specification. It makes it easy for back end implementation and client side consumption with visual documentation.

We can get either all the data or find it via special request e.g. id, name, etc.

We have controllers for all the tables and can easily access all the data from the frontend.

Besides the data we can do authorization, registration and also role management.

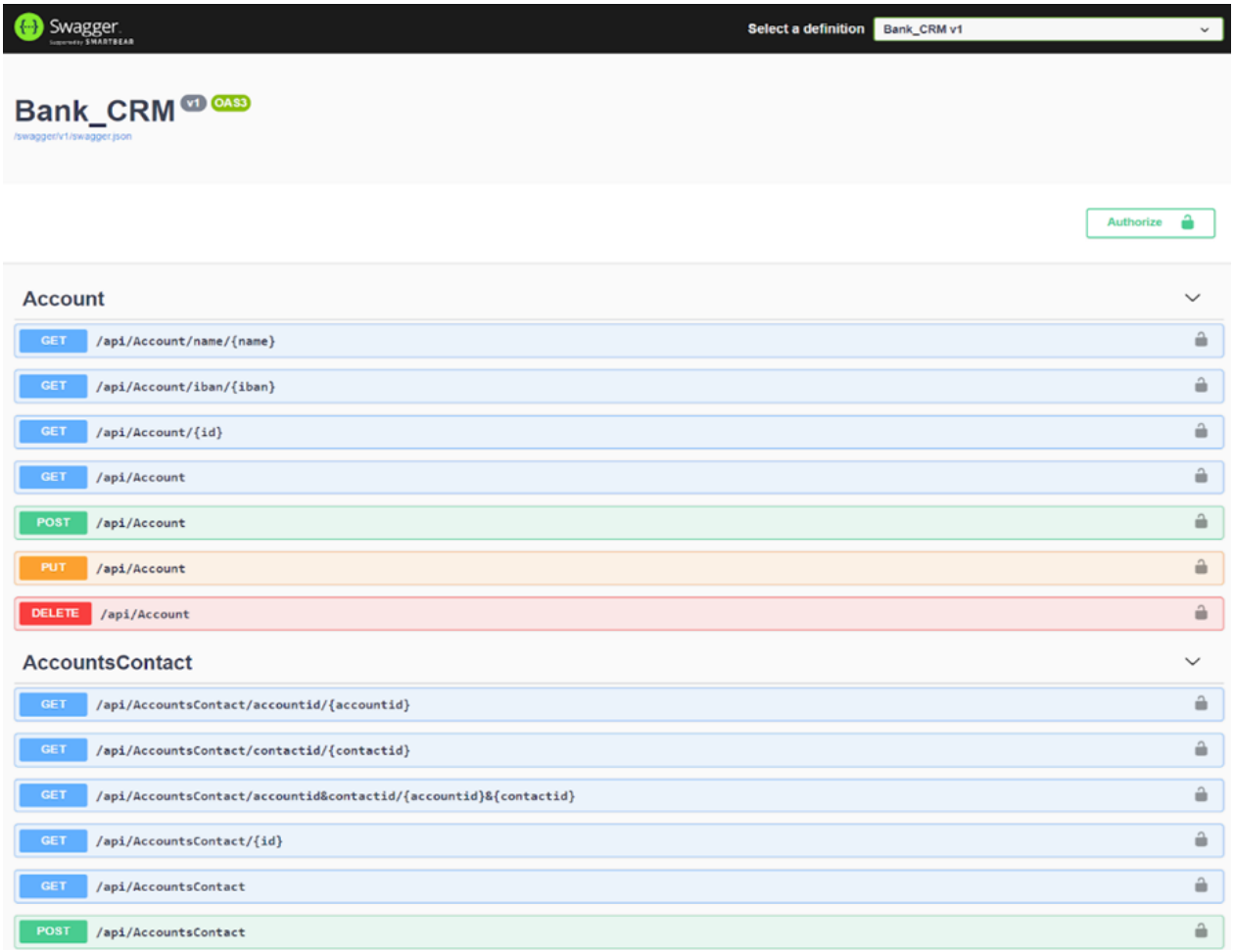


Figure 4. Swagger page

Controllers & data. We use The Repository pattern. Repositories are classes or components that encapsulate the logic required to access data sources. They centralize common data access functionality, providing better maintainability and decoupling the infrastructure or technology used to access databases from the domain model layer. A repository performs the tasks of an intermediary between the domain model layers and data mapping, acting in a similar way to a set of domain objects in memory.

In the IBaseRepository we declare methods that are used to work with controllers. It is done to enable dependency injection, due to which you can easily change the implementation.

This provides work with all models. Then in BaseRepository class we give appropriate body for methods.

```
43 references
public interface IBaseRepository<TDbModel> where TDbModel : BaseModel
{
    18 references
    public List<TDbModel> GetAll();
    4 references
    public TDbModel Get(int? id);
    3 references
    public TDbModel Create(TDbModel model);
    2 references
    public TDbModel Update(TDbModel model);
    2 references
    public void Delete(int? id);
}
```

```
19 references
public class BaseRepository<TDbModel> : IBaseRepository<TDbModel> where TDbModel : BaseModel
{
    12 references
    private Bank_CRMContext Context { get; set; }

    0 references
    public BaseRepository(Bank_CRMContext context)
    {
        Context = context;
    }

    3 references
    public TDbModel Create(TDbModel model)
    {
        Context.Set<TDbModel>().Add(model);
        Context.SaveChanges();
        return model;
    }

    4 references
    public TDbModel Get(int? id)
    {
        return Context.Set<TDbModel>().FirstOrDefault(m => m.Id == id);
    }

    18 references
    public List<TDbModel> GetAll()
    {
        return Context.Set<TDbModel>().ToList();
    }
```

Figure 5. Repository pattern

In Startup class we inject our models in the ConfigureServices method.

In the ConfigureServices method, the system will pass instances of the BaseRepository class in place of the IBaseRepository interface objects.

Once added to ConfigureServices, services can be retrieved and used anywhere in the application. And through the parameter of the Configure method, we can get the service and use it.

```
#region Transients
services.AddTransient<IBaseRepository<Account>, BaseRepository<Account>>();
services.AddTransient<IBaseRepository<Accountscontact>, BaseRepository<Accountscontact>>();
services.AddTransient<IBaseRepository<Activitiestranslation>, BaseRepository<Activitiestranslation>>();
services.AddTransient<IBaseRepository<Activity>, BaseRepository<Activity>>();
services.AddTransient<IBaseRepository<Card>, BaseRepository<Card>>();
services.AddTransient<IBaseRepository<Contact>, BaseRepository<Contact>>();
services.AddTransient<IBaseRepository<Contactstranslation>, BaseRepository<Contactstranslation>>();
services.AddTransient<IBaseRepository<Language>, BaseRepository<Language>>();
services.AddTransient<IBaseRepository<Loan>, BaseRepository<Loan>>();
services.AddTransient<IBaseRepository<Previlege>, BaseRepository<Previlege>>();
services.AddTransient<IBaseRepository<Previlegestranslation>, BaseRepository<Previlegestranslation>>();
services.AddTransient<IBaseRepository<Previlegestype>, BaseRepository<Previlegestype>>();
services.AddTransient<IBaseRepository<Role>, BaseRepository<Role>>();
services.AddTransient<IBaseRepository<Rolesprevilege>, BaseRepository<Rolesprevilege>>();
services.AddTransient<IBaseRepository<Statustype>, BaseRepository<Statustype>>();
services.AddTransient<IBaseRepository<Transaction>, BaseRepository<Transaction>>();
services.AddTransient<IBaseRepository<Usersrole>, BaseRepository<Usersrole>>();
services.AddTransient<IEmailSenderService, EmailSenderService>();
#endregion Transients
```

Figure 6. Transients

Login & Registration

In order to create user login and registration based on roles access we use cookies.

In the login field we check whether a user exists in the system, accuracy of entered password/email, confirmation of email with UserManager class.

If yes, we log him in the system with appropriate role access rights.

```
[HttpPost("api/Auth/login/{email}&{password}")]
0 references
public async Task<IActionResult> Login(string email, string password)
{
    if (!ModelState.IsValid)
        return new JsonResult($"ERROR: Model State Is Not Valid");

    User user = await userManager.FindByEmailAsync(email);

    if (user != null)
    {
        bool IsValidPassword = await userManager.CheckPasswordAsync(user, password);

        if (user.EmailConfirmed && IsValidPassword)
        {
            var emailCookie = new CookieHeaderValue("email", user.Email);

            HttpContext.Response.Cookies.Append("email", user.Email);
            HttpContext.Response.Cookies.Append("passwordHash", user.PasswordHash);

            return new JsonResult($"{email} {user.PasswordHash}");
        }
        else
            return new JsonResult("Unconfirmed email or Invalid password");
    }
    return new JsonResult("User does not exist");
}
```

Figure 7. Log in

In order to register, users need to enter creds and confirm email via a specific link. If the link expires, users can register with the same creds and get a new confirmation link.

We use the ModelState property of a Controller object, which represents a collection of name and value pairs that were submitted to the server during a POST. If the user did not enter required fields the error will be caught.

It is checked by UserManager whether the user already exists and if so, it will report the corresponding information.

After that step SA appoints him role access and this user is being added to the database.

```
[HttpPost("api/Auth/registration/{email}&{password}")]
0 references
public async Task<IActionResult> Registration(string email, string password)
{
    if (!ModelState.IsValid)
        return new JsonResult($"ERROR: Model State Is Not Valid");

    User user = await userManager.FindByEmailAsync(email);
    if (user != null)
    {
        if (user.EmailConfirmed)
            return new JsonResult($"{email} exists in AspNetUsers' table");
        else if (await userManager.CheckPasswordAsync(user, password))
        {
            bool emailResponse = SendEmailConfirmation(user).Result;

            if (emailResponse)
                return new JsonResult("Another confirmation is sent");
            else
                return new JsonResult("Email was not sent");
        }
        else
            return new JsonResult("Invalid password for the User(Email is not confirmed)");
    }

    User newUser = new User
    {
        UserName = email,
        Email = email
    };
    IdentityResult result = await userManager.CreateAsync(newUser, password);

    if (result.Succeeded)
    {
        bool emailResponse = SendEmailConfirmation(newUser).Result;

        if (emailResponse)
            return new JsonResult("Confirmation is sent");
        else
            return new JsonResult("Email was not sent");
    }
    else
    {
        foreach (IdentityError error in result.Errors)
            ModelState.AddModelError("", error.Description);
    }
    return new JsonResult(result.Errors.ToList());

}
```

Figure 8. Registration

The confirmation link is sent by EmailSenderService.

```
2 references
private async Task<bool> SendEmailConfirmation(User user)
{
    var token = await userManager.GenerateEmailConfirmationTokenAsync(user);
    var confirmationLink = Url.Action("ConfirmEmail", "Auth", new { token, email = user.Email }, Request.Scheme);
    EmailSenderService emailSender = new EmailSenderService();
    return emailSender.SendEmail(user.Email, confirmationLink);
}
```

Figure 9. Confirmation link

In IdentityHostingStartup we have a configuration which checks correctness of password and imposes restrictions for the password.

```
1 reference
public class IdentityHostingStartup : IHostingStartup
{
    0 references
    public void Configure(IWebHostBuilder builder)
    {
        builder.ConfigureServices((context, services) => {

            services.AddDbContext<IndentityBank_CRMContext>(options =>
                options.UseNpgsql(
                    context.Configuration.GetConnectionString("IndentityBank_CRMContextConnection")));

            services.AddDefaultIdentity<User>(options => options.SignIn.RequireConfirmedAccount = true)
                .AddEntityFrameworkStores<IndentityBank_CRMContext>();

            services.Configure<IdentityOptions>(opts =>
            {
                opts.User.RequireUniqueEmail = true;
                opts.User.AllowedUserNameCharacters =
                    "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-._@+ ";

                opts.SignIn.RequireConfirmedEmail = true;

                opts.Password.RequireNonAlphanumeric = false;
                opts.Password.RequiredLength = 6;
                opts.Password.RequireDigit = true;
                opts.Password.RequireUppercase = true;
                opts.Password.RequireLowercase = true;

            });
        });
    }
}
```

Figure 10. Password correctness checking

Login is based on roles authentication. When user enters credentials, Filter checks his access level. Authorization filter determine whether the user is authorized to fulfill the current request. If the user is not authorized to access the resource, then the filter ends processing the request. All roles and privileges are taken from the database.

FRONTEND

Login form. When user pressed on the sign up button, the registration page is loaded. To authenticate, a user need to enter email and password. Also on the login page a validation exists.

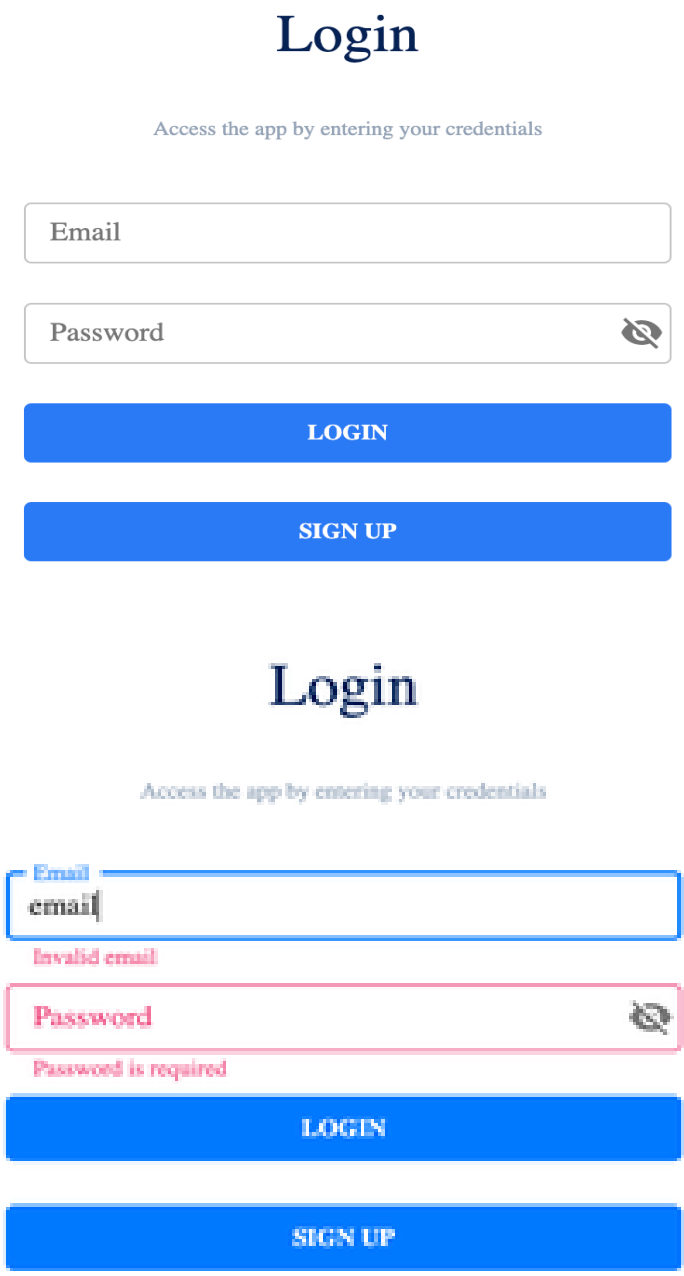


Figure 11. Login form with checking

Registration form. To register a user need to enter email and password. After that, user get email with a link to activate the account. After activation he will have an opportunity to log in.

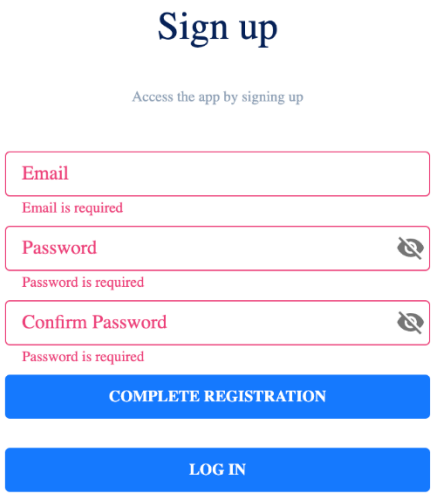
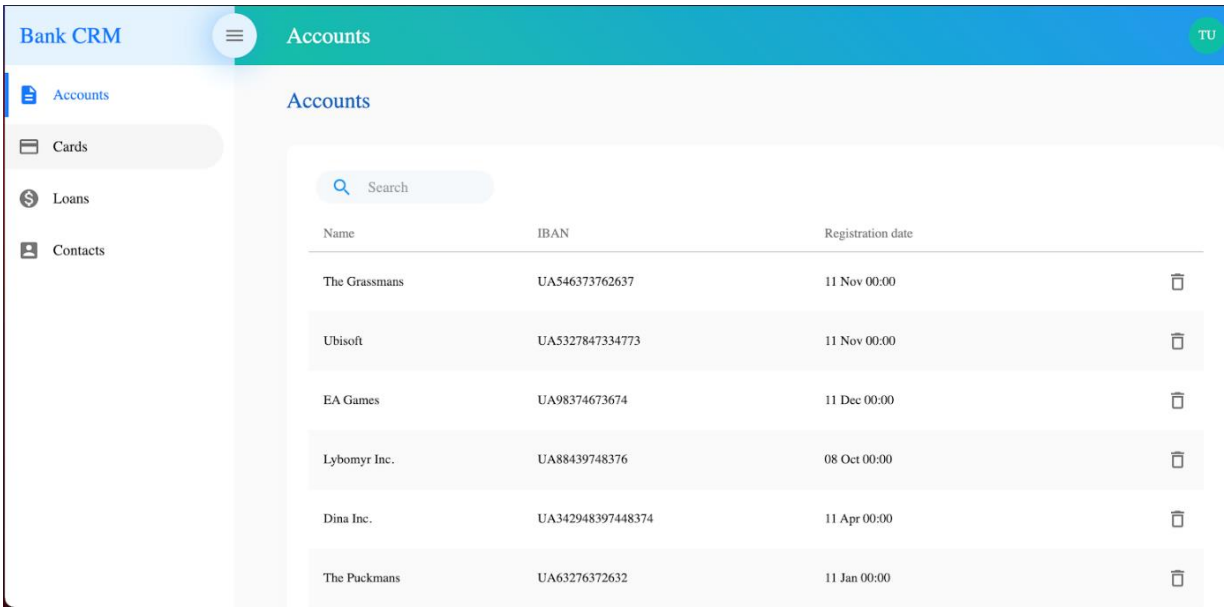


Figure 12. Registration form

```
class SignIn extends React.Component {
  static propTypes = {
    history: RouterPropTypes.history.isRequired,
    login: PropTypes.func.isRequired,
  };

  static defaultProps = {};

  componentDidMount() {}

  onSubmit = async (values, formikActions) => {
    const { login, history } = this.props;
    const { email, password } = values;
    if (await login(email, password, formikActions)) {
      history.push(baseCrmPath);
    }
  };

  render() {
    return (
      <Formik
        initialValues={{
          email: '',
          password: '',      Maksym Karpenko, 3 weeks ago • feat: initial setup
        }}
        validationSchema={validation.LoginSchema}
        render={formikProps => <SignInForm formikProps={formikProps} />}
        onSubmit={this.onSubmit}
      />
    );
  }
}

export default connect(mapStateToProps, mapDispatchToProps)(SignIn);
```

Figure 13. Code for login page

Project start page. Here we have main information about accounts in a table view.



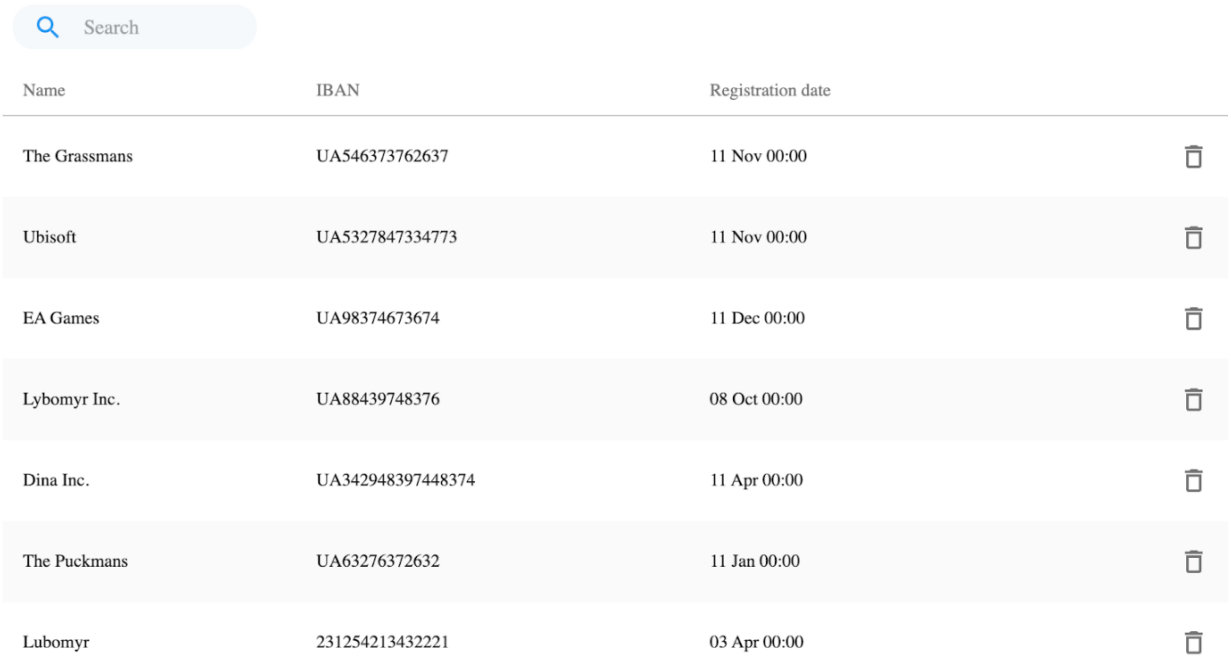


Figure 14. Accounts info

```
return (
  <ContentBox className={classes.root} {...props}>
    <div className={classes.tableWrapper}>
      <div className={classes.tableHeaderWrapper}>
        <Search
          placeholder="Search"
          className={classes.search}
          onChange={onSearch}
          inputStyle={{ width: searchWidth }}
        />
      </div>
      <CustomTable>
        <Header className={classes.header}>
          <TableRow>
            {columns.map(column => (
              <HeaderCell
                className={classes.tableHeader}
                key={uuid()}
                row={column}
                order={direction}
                orderBy={sort}
                onRequestSort={handleRequestSort}
              />
            ))}
          </TableRow>
        </Header>
        <Content>{tableContent}</Content>
      </CustomTable>
    </div>
    <Pagination
      labelRowsPerPage="Rows per page"
      rowsCount={total}
      rowsPerPage={perPage}
      page={page}
      handleChangePage={handlePageChange}
      handleChangeRowsPerPage={handleChangeRowsPerPage}
    />
  </ContentBox>
```

Figure 15. Tables code

Clicking on the delete icon a modal window with confirmation will be opened. Here we can confirm or reject the deletion of the entity.

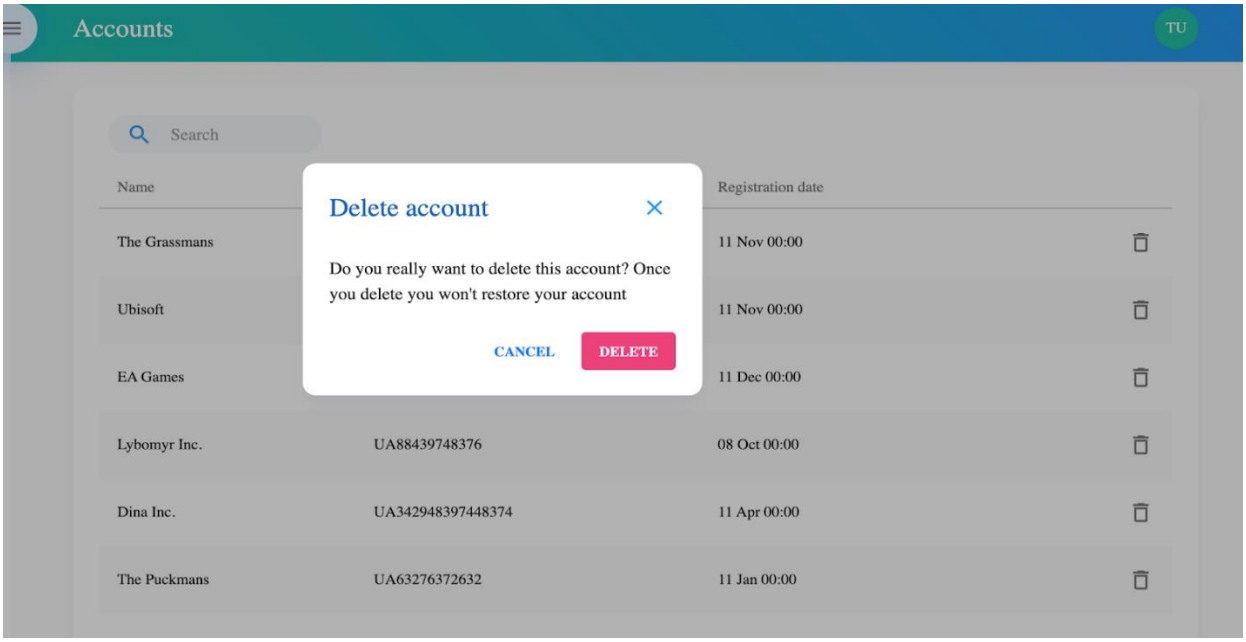


Figure 16. Deletion of the entity

After that, a notification is displayed that the removal was successful.

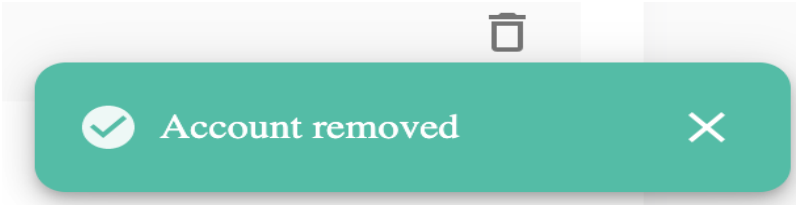


Figure 17. Deletion confirmation

By clicking on a row in the table, we can go to the page with detailed information of the entity.

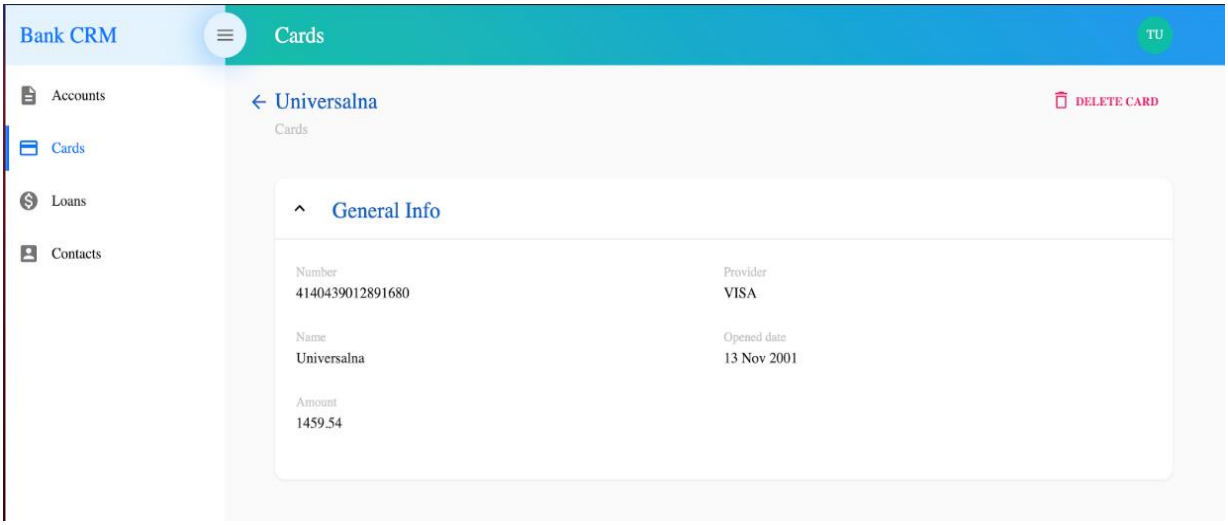


Figure 18. Detailed info

TESTING

For testing our project we use Unit Testing. We use NUnit library and Moq library.

```
using Bank_CRM.Controllers.Base;
using Bank_CRM.Models;
using NUnit.Framework;
using System;
using System.Collections.Generic;
using System.Text;

namespace Bank_CRM.Tests {
    [TestFixture]
    class BaseControllerTests {

        public static IEnumerable<IGenericControllerTestCase> TestCases() {
            yield return new GenericControllerTestCase<Account>();
            yield return new GenericControllerTestCase<Contact>();
            yield return new GenericControllerTestCase<Activity>();
        }

        [Test]
        [TestCaseSource("TestCases")]
        public void GetAll_ExistsAtLeastOneEntity_RetrunModels(IGenericControllerTestCase testCase) {
            testCase.GetAll_ExistsAtLeastOneEntity_RetrunModels();
        }
        [Test]
        [TestCaseSource("TestCases")]
        public void GetAll_NotExistsAtLeastOneEntity_RetrunEmptyList(IGenericControllerTestCase testCase) {
            testCase.GetAll_NotExistsAtLeastOneEntity_RetrunEmptyList();
        }
        [Test]
        [TestCaseSource("TestCases")]
        public void Get_ProvideIdExistingInDb_ReturnModel(IGenericControllerTestCase testCase) {
            testCase.Get_ProvideIdExistingInDb_ReturnModel();
        }
        [Test]
        [TestCaseSource("TestCases")]
        public void Get_ProvideIdNotExistingInDb_ReturnNull(IGenericControllerTestCase testCase) {
            testCase.Get_ProvideIdNotExistingInDb_ReturnNull();
        }
        [Test]
        [TestCaseSource("TestCases")]
        public void Post_ModelCreated_ReturnSuccess(IGenericControllerTestCase testCase) {
            testCase.Post_ModelCreated_ReturnSuccess();
```

Figure 19. Unit testing

```
using Bank_CRM.Controllers.Base;
using Bank_CRM.Models.Base;
using Bank_CRM.Repositories.Interfaces;
using Moq;
using NUnit.Framework;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Bank_CRM.Tests {
    public class GenericControllerTestCase<T> : IGenericControllerTestCase
        where T : BaseModel, new() {

        private readonly BaseController<T> _baseController;
        private readonly Mock<IBaseRepository<T>> _repositoryMock;

        public GenericControllerTestCase() {
            _repositoryMock = new Mock<IBaseRepository<T>>();
            _baseController = new BaseController<T>(_repositoryMock.Object);
        }

        private void Initialize() {

        }

        public void Delete_ExceptionRaise_ReturnNotSuccess() {
            var model = new T {
                Id = 1
            };
            _repositoryMock.Setup(r => r.Get(model.Id)).Returns(model);
            _repositoryMock.Setup(r => r.Delete(model.Id)).Throws(new ArgumentException());

            var result = _baseController.Delete(model.Id);

            Assert.That(result.Value.ToString().Equals("Delete was not successful"));
        }

        public void Delete_ModelDeleted_ReturnSuccess() {
            var model = new T {
                Id = 1
            };
            _repositoryMock.Setup(r => r.Get(model.Id)).Returns(model);

            var result = _baseController.Delete(model.Id);

            Assert.That(result.Value.ToString().Equals("Delete successful"));
        }
```

Figure 20. Unit testing using Moq library

Also we create a formal verification and specification for Login page with the help of Code Contracts.

```
[HttpPost("api/Auth/registration/{email}&{password}")]
0 references
public async Task<IActionResult> Registration(string email, string password)
{
    // Contracts start


    // Precondition
    Contract.Requires(!string.IsNullOrEmpty(email));
    Contract.Requires(!string.IsNullOrEmpty(password));
    Contract.Requires(ModelState.IsValid, "Invalid Model state");

    //Postcondition

    Contract.Ensures(Contract.Result<JsonResult>() != null);

    //Contracts end
```

```
[ContractInvariantMethod]
0 references
private void ValidateUserManager() {
    Contract.Invariant(this.userManager != null);
}
```

```
2 references
private async Task<bool> SendEmailConfirmation(User user)
{
    Contract.Requires(user != null);
    Contract.Ensures(Contract.Result<bool>() == true);
    var token = await userManager.GenerateEmailConfirmationTokenAsync(user);
    var confirmationLink = Url.Action("ConfirmEmail", "Auth", new { token, email = user.Email }, Request.Scheme);
    EmailSenderService emailSender = new EmailSenderService();
    return emailSender.SendEmail(user.Email, confirmationLink);
}
```

Figure 21 .Verification for the logging

STAGES OF DEVELOPMENT

At the stage of detailed design, the following stages of work should be completed:

1. program development
   - 1.1 Creating a database model
   - 1.2 Database creation
   - 1.3 Service development on the ASP.NET Core platform
   - 1.4 Implementation of user interface functionality
   - 1.5 Implementation of integration
2. program testing
3. report preparing

Deadlines. https://crm-bank-knu.atlassian.net/jira/software/c/projects/BC/boards/1/roadmap

Control and acceptance procedure. Control over the progress of work is done using the Jira service, where the executing party notes all current tasks, sprints, reports on the work done, etc.

The team works according to the Scrum project management method, thereby allowing you to observe progress in each sprint.

For our project we use 2-weeks sprints and meetings twice a week. At these meetings we discussed the current progress of the team, encountered problems and possible solution options.

At the first sprint we defined an optimal number of tasks and so, for the next sprints we could planning an amount of tasks to do and further work process.

At the end of the each sprint we had a retrospective, where we discussed such questions as:

"What was good in a team?", "What was bad in a team?", "What could we do better in next sprint?" and also review tasks and move technical debt into backlog.

Also at the start of the sprint we have a sprint planning where we prioritize a backlog and choose tasks that we will do at this sprint.

Roles in project

Name	Role
Kuntso Dmytro	CTO / Scrum master
Zverieva Tetiana	Business analyst / Project Manager
Karpenko Maxim	Frontend developer
Maevskiy Lybomyr	Backend developer
Formakidova Dina	Backend developer

Github
Frontend: https://github.com/makskarp/bank-crm-fe
Backend: https://github.com/Dymasik/Bank-CRM

РОЗРОБКА СИСТЕМИ ПІДТРИМКИ ВИКЛАДАННЯ ІНТЕГРОВАНОГО КУРСУ

Андрій Кліщ, Олена Намака, Оксана Савонік, Юлія Токан, Софія Ярмоленко

ВСТУП

Актуальність. Школа, внз, навчальний центр, навіть мовні курси або автошкола постійно працюють з великою кількістю людей, які змінюються з певною періодичністю (по закінченню кожного навчального терміну). Для будь-якої навчальної організації важливо мати систему управління, що дозволяє моніторити відвідуваність та успішність учнів, надходження оплати за навчання, перевіряти домашні завдання, відповідати на питання в загальному чаті й особисто.

Також для якісної роботи дуже важливо формувати звіти, як державного зразка, так і для власного управлінського аналізу та обліку. І все це можна робити на одній платформі.

Автоматизація навчального процесу відчутно полегшує навчання для учнів і робочий процес для вчителів. Наприклад, усі необхідні дані знаходяться в готових папках із простим і зручним доступом, виконані і нові завдання сортуються автоматично, рейтинг кращих учнів формується автоматично, фіксується кількість спроб скласти іспит і всі результати. Також можна автоматизувати будь-які процеси, потрібні конкретно вам: складати звіти натисканням декількох кнопок і т. п.

Управління системою відбувається шляхом створення груп (класів), до яких прикріплюється учитель, куратор і набір курсів, що відповідає даній групі. Учень підключається до системи в залежності від його форми навчання, матеріалів, курсів, що повинен освоїти у процесі навчання. Учитель же може контролювати виконання всіх завдань: чи встигає учень за розкладом і послідовністю тем, наскільки він їх зрозумів і яку оцінку заробив.

Мета й завдання роботи. Метою роботи є розробка веб застосунку, який міститиме систему управління навчального процесу, що зможе забезпечити можливістю вести учбовий процес онлайн. Для досягнення цієї мети поставлено такі завдання:

- дослідити існуючі застосунки для управління навчальним процесом на ринку;
- дослідити застосування різних технологій для проєктування та реалізації веб сайту;
- розробити технічне завдання до програмного продукту;
- розробити інтерфейс та дизайн веб застосунку.

Об'єкт, методи й засоби розробки. Об’єктом розроблення програми є процес ведення навчального процесу. Предметом роботи є веб застосунок для забезпечення можливості ведення контролю знань дистанційно.

Розробці програмного засобу передував аналіз готових рішень та способи їх застосування. В якості інструменту створення програмного засобу було обрано IntelliJ IDEA IDE – інтегроване середовище розробки мовою програмування Java від компанії JetBrains.

ОГЛЯД ІСНУЮЧИХ НА РИНКУ СИСТЕМ

1. Google Classroom [1]. Веб сервіс для надання послуг навчальним закладам. Значно спрощує обмін файлами між учнями і вчителями шляхом об’єднання у собі різних інших веб систем від Google (рис. 1).

Рисунок 1. Google Classroom

2. Moodle [2]. Дослівний переклад абревіатури звучить так - модульне об'єктно-орієнтоване динамічне навчальне середовище (рис. 2). Система для організації навчального процесу. Підходить для організації дистанційних курсів, очного навчання , тощо.

Рисунок 2. Логотип системи Moodle

3. ILIAS [3]. Програмне забезпечення для підтримки навчального процесу. Використовується переважно у німецьких закладах (рис. 3).

Рисунок 3. Логотип системи ILIAS

4. **Unicraft [4].** Система для створення курсів та автоматизації навчання в компаніях (рис. 4).

Рисунок 4. Логотип Unictaft

5. **Schoology [7].** Приклад системи для полегшення організації дистанційного навчання (рис. 5).

Рисунок 5. Логотип Schoology

Отже, в Інтернеті існує уже багато веб-систем для організації навчального процесу . Слід зазначити, що наша система буде більш заточена на наш факультет - комп’ютерних наук та кібернетики, і також матиме деякі особливості (як система антиплагіату).

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ТА ТЕХНІЧНІ ВИМОГИ

Вибір технологій. Java [6] – це мова комп'ютерного програмування. Вона дозволяє програмістам писати комп'ютерні інструкції, використовуючи англійські команди, замість того, щоб писати в числових кодах. Він відомий як мова "високого рівня", тому що його легко читати і писати люди. Як і будь-яка мова, Java має набір правил, які визначають написання інструкцій. Ці правила відомі як його "синтаксис". Після написання програми інструкції високого рівня переводяться в числові коди, які комп'ютери можуть зрозуміти і виконати.

Apache Maven [7] – це інструмент автоматизації побудови проєктів Java. Подумайте про Ant, або Make, але набагато потужніше і простіше у використанні. Якщо вам доводилося мати справу зі створенням Java-проєкту з залежностями або спеціальними вимогами для збирання, ви, мабуть, переживали розчарування, які Maven прагне усунути. Maven був проєктом з відкритим кодом під Apache [8] з 2003 року, починаючи з Sonatype до цього. Враховуючи його сильну підтримку і величезну популярність, Maven є дуже стабільним і багатофункціональним, забезпечуючи численні плагіни, які можуть зробити що-небудь від генерації PDF-версій документації вашого проєкту до створення списку останніх змін з вашого SCM. І все, що потрібно, щоб додати цю функціональність - це невелика кількість додаткового XML або додатковий параметр командного рядка. Є багато залежностей? Без проблем. Maven підключається до віддалених сховищ (або ви можете налаштувати власні місцеві репозитарії) і автоматично завантажує всі залежності, необхідні для створення вашого проєкту.

Spring Framework [9] – це платформа Java, яка забезпечує всебічну підтримку інфраструктури для розробки додатків Java. Spring [10] обробляє інфраструктуру, щоб ви могли зосередитися на вашому додатку. Spring дозволяє створювати програми з "звичайних старих Java-об'єктів" (POJO) і застосовувати корпоративні послуги неінвазивно до POJO. Ця можливість застосовується до моделі програмування Java SE і до повної і часткової Java EE. Приклади того, як ви, як розробник додатків, можете використовувати перевагу платформи Spring: зробіть метод Java виконаним у транзакції бази даних без необхідності мати справу з API транзакцій; зробити локальний метод Java віддаленою процедурою без роботи з віддаленими API; зробіть локальний метод Java операцією керування без необхідності мати справу з JMX API [11]; зробіть локальний метод Java обробником повідомлень, мати справу з JMS API. Spring Framework складається з функцій, організованих у близько 20 модулів. Ці модулі згруповані в контейнер Core, доступ до даних / інтеграцію, Web, AOP (програмне забезпечення, орієнтоване на аспект), приладобудування і тест.

Core контейнер – основний контейнер складається з модулів Core, Beans, Context і Language Expression. Модулі Core та Beans забезпечують основні частини рамки, включаючи особливості IoC та Dependency Injection. BeanFactory являє собою складну реалізацію заводського шаблону. Це усуває потребу в програмних синглетонах і дозволяє відділити конфігурацію і специфікацію залежностей від фактичної логіки програми. Модуль "Контекст" будується на твердій основі, що надається модулями Core і Beans: це засіб доступу до об'єктів у стилі, що відповідає рамкам, подібним до реєстру JNDI. Модуль "Контекст" успадковує його функції від модуля Beans і додає підтримку інтернаціоналізації (використовуючи, наприклад, пакети ресурсів), поширення подій, завантаження ресурсів і прозоре створення контекстів, наприклад, контейнером сервлетів. Модуль Context також підтримує функції Java EE, такі як EJB, JMX і базові віддалені можливості. Інтерфейс ApplicationContext є фокусною точкою модуля Context.

Trello [12] - це одна з найпопулярніших систем управління проєктами в режимі онлайн, яка користується особливим попитом серед невеликих компаній і стартапів. Вона дозволяє ефективно організовувати роботу по японській методології канбан-дощок.

Dafny [13] – це засіб перевірки програм, що включає мову програмування та специфікацію конструкції. Користувач Dafny створює та перевіряє специфікації та реалізації. Мова програмування Dafny є об'єктною, імперативною, послідовною і підтримує загальні класи і динамічне виділення. Конструкції специфікації включають стандартні перед- та пост-умови, конструктивні рамки та метрики завершення.

Технічні вимоги. Розробки повинні будуватись з використанням підходів централізовано програмно-технологічної платформи, з уніфікацією програмно-технічних засобів розробки (модернізації) прикладної функціональності з використанням сучасних веб-портальних, сервісно-орієнтованих технологій.

Базовими компонентами розробки мають бути програмні комплекси сервісів, що забезпечують реалізацію функціональності ІС ПВК.

Розробка повинна забезпечувати уніфікований та комфортний, максимально простий та інтуїтивно зрозумілий інтерфейс користувача.

Технологічна гнучкість, надійність роботи при розробці функціонального складу повинен досягатись за рахунок реалізації принципів стандартизації та уніфікації, а саме:

- уніфікованих правил структурної побудови та організації прикладних програмних компонент, їх взаємодії між собою;
- стандартизації вимог до побудови єдиної централізованої бази даних, формування єдиних вимог до класифікації об’єктів та їх атрибутивного складу;
- уніфікації правил побудови інформаційної взаємодії з іншими інформаційними системами.

Система повинна мати архітектуру, побудовану на сучасних технологіях зберігання, обробки, аналізу даних та доступу до них; забезпечувати одночасну роботу декількох користувачів. Рішення щодо побудови Системи повинні базуватися на:

- застосуванні сучасних інформаційних технологій; - реалізації концепції створення єдиного інформаційного простору;
- застосуванні правила централізованого накопичення, зберігання та обробки інформації.
- підтримці актуальності, повноти, несуперечності, цілісності та доступності інформації;
- забезпеченні надійного захисту інформації від порушення її цілісності, витоку та блокування згідно з вимогами нормативно-правових документів в галузі захисту інформації;
- забезпеченні надійності, резервування компонентів технічного забезпечення Системи;
- забезпеченні централізованого управління, безперервного контролю функціонування та централізованого налаштування Системи і модулів її компонентів;
- використанні сучасних засобів програмної інженерії при розробці програмного прикладного забезпечення.

Система представляє собою комплекс інформаційних, програмних, технічних, організаційно-методичних та інших необхідних засобів, що забезпечують збір, обробку, зберігання та передачу даних. Архітектура Системи повинна передбачати максимальну незалежність програмно технічних модулів від Виконавця.

Інформаційна архітектура Системи повинна відповідати сучасним вимогам щодо побудови інтерфейсів користувача.

ОПИС ОРГАНІЗАЦІЙНОЇ ІНФОРМАЦІЙНОЇ БАЗИ

Логічна структури бази даних. users – таблиця користувачів веб-додатку. roles – таблиця ролей користувачів. courses – таблиця курсів. posts – таблиця постів. tasks - таблиця завдань, які потрібно виконати користувачу з роллю студент у певному курсі.

to_do_items - таблиця для збереження особистого списку речей, що треба зробити для кожного користувача. courses_role - зв'язна таблиця між курсом і роллю користувача. courses_users - зв'язна таблиця між курсом і користувачем. user_to_do_items- зв'язна таблиця між списком речей і користувачем. documents - таблиця для збереження файлів. comments - таблиця для збереження коментарів до завдань.

Опис таблиць. У таблицях 1-8 представлені основні таблиці інформаційної бази.

Таблиця 1. users

id	bigint	Ключ користувача
Name	varchar(50)	Ім'я користувача
Surname	varchar(100)	Прізвище користувача
Email	varchar(100)	Електронна пошта
Password	varchar(100)	Пароль аккаунту

Таблиця 2. roles

id	bigint	Ключ ролі
name	varchar(255)	Назва ролі

Таблиця 3. courses

id	bigint	Ключ курсу
name	varchar(255)	Назва

Таблиця 4. tasks

id	bigint	Ключ завдання
title	varchar(255)	Тема або назва завдання
task	text	Текст завдання
courses_id	bigint	Ключ курсу
time	varchar(20)	Час

Таблиця 5. to_do_items

id	bigint	Ключ задач
title	varchar(255)	Текст задачі
done	tinyint	Чи виконано
user_id	bigint	Ключ користувача

Таблиця 6. tasks

id	bigint	Ключ поста
title	varchar(255)	Тема або назва поста
post	text	Текст поста
courses_id	bigint	Ключ курсу
announce	varchar(255)	Короткий опис

Таблиця 7.comments

id	bigint	Ключ коментаря
comment	text	Текст коментаря
task_id	bigint	Ключ завдання
user_id	bigint	Ключ користувача

Таблиця 8.documents

id	bigint	Ключ файлу
name	varchar(255)	Назва файлу
size	bigint	Розмір файлу
upload_time	datetime(6)	Час завантаження
content	longblob	Власне файл
courses_id	bigint	Ключ курсу
user_id	bigint	Ключ користувача
task_id	bigint	Ключ завдання
post_id	bigint	Ключ поста

У таблицях 9-11 представлені таблиці зв’язків.

Таблиця 9. courses_users

id	bigint	Ключ зв’язку
courses_id	bigint	Ключ курсу
user_id	bigint	Ключ користувача
courses_roles_id	bigint	Ключ зв’язку ролі користувача

Таблиця 10. courses_roles

id	bigint	Ключ зв’язку
courses_id	bigint	Ключ курсу
roles_id	bigint	Ключ ролі

Таблиця 11. user_to_do_items

id	bigint	Ключ зв’язку
user_id	bigint	Ключ користувача
to_do_items_id	bigint	Ключ списку завдань

ФОРМАЛЬНА СПЕЦИФІКАЦІЯ

Для формальної специфікації та верифікації була обрана мова Dafny. Dafny - це комплексний засіб, що включає в себе мову програмування та конструкції специфікації та використовується для перевірки функціональної коректності програм. Як задачу було поставлено підрахунок балів студента за окремий курс. Бали студента зберігаються в масиві цілих чисельних значень. Підрахунок балів - це підрахунок суми всіх чисел масиву.

Класс Subject відповідає за предмет. Він зберігає назву предмету та оцінки за цей предмет. Клас Student має такі поля як email та subject. Email для ідентифікації студента, а subject показує приналежність студента до певного предмету. Відповідно так кожному студенту присвоюються бали за окремий предмет.

Функція Sum підраховує суму балів будь-якого масиву. На вхід має подаватися не порожній масив. Функція Calculate використовуючи Sum знаходить підсумковий бал для конкретного студента, що вказаний в параметрах функції.

Код програми:

```
class Subject{
  var Name: string;
  var Grade: array<int>;
  constructor(name: string, grade: array<int>){
    this.Name := name;
    this.Grade := grade;
  }
}

class Student{
  var Email: string;
  var subject: Subject;
  constructor(subject: Subject, email: string)
  modifies subject
  {
    this.subject := subject;
    this.Email := email;
  }

  function RecursiveSum(a: array<int>, to: nat) : int
    reads a
    requires a != null
    requires to <= a.Length
  {
    if to == 0 then 0
    else a[to-1] + RecursiveSum(a, to-1)
  }

  method Sum(a: array<int>) returns (s: int)
    requires a != null
    ensures s == RecursiveSum(a, a.Length)
```

```
{
  s := 0;
  var i : nat := 0;
  while (i < a.Length)
    invariant  0 <= i <= a.Length && s==RecursiveSum(a,i) ;
  {
    s, i := s+a[i], i+1;
  }
}

method Calculate(student: Student) returns (s:int){
  var subj := student.subject;
  var a: array<int> := subj.Grade;
  s:= Sum(a);
  print student.Email;
  print ": ";
  print student.subject.Name;
  print " ";
  print s;
}
```

Результат наведено на рисунку 4.2.1.

```
Dafny program verifier finished with 6 verified, 0 errors
Running...

StudentA@gmail.com: Subject1 100
```

Рисунок 4.1. Результат формальної специфікації

Також для на етапі специфікації проєкту було спроєктовано різні види діаграм для кращого розуміння специфіки системи (див. додаток А).

ЕТАП ТЕСТУВАННЯ

Тестування системи. Тестування програмного забезпечення – це етап життєвого циклу розробки програмного продукту, на якому здійснюються перевірка відповідності між реальною поведінкою програми та поведінкою, яку очікує клієнт.

Цей етап допомагає команді зрозуміти в якому стані зараз знаходиться проєкт, на що треба звернути увагу, і як планувати подальшу розробку. Наразі за розробку більшості веб-застосувань відповідають цілі команди кваліфікованих співробітників.

Автоматичне тестування це процес на етапі контролю якості життєвого циклу розробки програмної розробки. Для розробки автоматичних тестів, їх запуску, використовують різні сучасні технології та програмні засоби що допомагає скоротити час тестування і спростити його процес.

Сучасні технології призначені для автоматизації процесів тестування постійно розвиваються. Існують технології, що дозволяють записувати дії користувача в браузері та конвертувати їх в кроки автоматичного тесту, що звісно може значно спростити процес розробки автоматичних тестів, але все одно, об’єм роботи над створенням набору тестів для

хоча б мінімального покриття основного функціонала, вимагатиме значних ресурсів та часу на її розробку.

Тестування програмного забезпечення - перевірка відповідності між отриманою на виході після закінчення етапу розробки і очікуваною поведінкою програми, що здійснюється на кінцевому наборі тестів, обраному певним чином, спеціалістом з тестування.

Класичний процес тестування програмного продукту виглядатиме наступним чином. Після отримання перших специфікацій, QA спеціаліст починаєте писати тест план та розробляти тест кейси, оцінюється необхідність використання автоматизації, причому як автоматизації функціонального тестування, так і навантажувального.

Мануальне тестування (Manual тестування) – є по факту найпростішим видом тестування, коли програмний продукт перевіряється вручну спеціалістами Manual Quality Assurance на відповідність вимогам програмного забезпечення та стандартам якості продукту.

У більшості проєктів мануальне тестування є основним видом тестування. Навіть при реалізації автоматизації основних тестових сценаріїв, виключення даного етапу неможливе. Всі автоматичні тести пишуться на основі мануальних кейсів і спершу перевіряються в ручну.

Дослідницьке тестування є одним із підходів до тестування, найчастіше мануального. У деяких проєктах воно може виявитися більш продуктивним, ніж звичайне тестування за сценаріями, під час яких пишуться тест плани та тест кейси.

Автоматичне тестування це частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення. Воно використовує програмні засоби для виконання тестів і перевірки результатів виконання, що допомагає скоротити час тестування і спростити його процес.

Головна мета автоматизації - прискорити процес тестування без втрати якості, а також позбавити ручного тестувальника від рутини проходження регресійних сценаріїв, дозволивши сфокусуватися на перевірці бізнес-логіки додатка з точки зору кінцевого користувача.

Існує кілька рівнів автоматичного тестування:

– Unit tests;
– Integration tests;
– GUI tests.

Аналіз результатів тестування.

Таблиця 12. Тест "testEmailKnu"

Показник	Значення
Опис тесту	Перевіряється на валідність пошта
Початкові умови	email = "test.11@gmail.com"
Очікуваний результат	Значення параметру email не відповідає стандартам knu.ua
Результат тесту	Тест пройдений – помилок не виявлено. Отримали повідомлення "Not knu user"

Таблиця 13. Тест "testEmailError"

Показник	Значення
Опис тесту	Перевіряється на валідність пошта
Початкові умови	email = "@gmail.com"
Очікуваний результат	Значення параметру email невалідне
Результат тесту	Тест пройдений – помилок не виявлено. Отримали повідомлення "Email should be valid"

Таблиця 14. Тест "testEmptyEmail"

Показник	Значення
Опис тесту	Перевіряється введення пошти
Початкові умови	email = ""
Очікуваний результат	Значення параметру email порожнє
Результат тесту	Тест пройдений – помилок не виявлено. Отримали повідомлення "Email cannot be empty"

Таблиця 15. Тест "testEmptyPassword"

Показник	Значення
Опис тесту	Перевіряється на введення пароль
Початкові умови	password = ""
Очікуваний результат	Значення параметру password порожнє
Результат тесту	Тест пройдений – помилок не виявлено. Отримали повідомлення "Password cannot be empty"

Таблиця 16. Тест "testPasswordError"

Показник	Значення
Опис тесту	Перевіряється на коректність пароль
Початкові умови	password = "qwe123"
Очікуваний результат	Значення параметру password невалідне
Результат тесту	Тест пройдений – помилок не виявлено. Отримали повідомлення "Password must be not less than 7"

Таблиця 17. Тест "testEmptyName"

Показник	Значення
Опис тесту	Перевіряється на введення ім’я
Початкові умови	name = ""
Очікуваний результат	Значення параметру name порожнє
Результат тесту	Тест пройдений – помилок не виявлено. Отримали повідомлення "Name cannot be empty"

Таблиця 18. Тест "testNameError"

Показник	Значення
Опис тесту	Перевіряється на коректність ім’я
Початкові умови	name = "Yu"
Очікуваний результат	Значення параметру name невалідне
Результат тесту	Тест пройдений – помилок не виявлено. Отримали повідомлення "Name must be between 2 and 30 characters"

Таблиця 19. Тест "validateRegistrationTestError"

Показник	Значення
Опис тесту	Перевіряється на коректність результат надсилання запиту на реєстрацію
Початкові умови	email = "yulia.tokan@knu.ua" password = "qwe" name="Yulia" surname="Tokan"
Очікуваний результат	4xxClientError
Результат тесту	Тест пройдений – помилок не виявлено. Отримали 400 помилку

Таблиця 20. Тест "validateRegistrationTest"

Показник	Значення
Опис тесту	Перевіряється на коректність результат надсилання запиту на реєстрацію
Початкові умови	email = "yulia.tokan@knu.ua" password = "qwe12345" name="Yulia" surname="Tokan"
Очікуваний результат	2xxSuccessful
Результат тесту	Тест пройдений – помилок не виявлено. Отримали код результату 200

Таблиця 21. Тест "incorrectEmailUserLogin"

Показник	Значення
Опис тесту	Перевіряється на коректність результат надсилання запиту на реєстрацію та інтерфейс користувача
Початкові умови	email = "yulia.tokan.11@gmail.com" password="qwe12345" name="johny" surname="test"
Очікуваний результат	Поява червоного сповіщення
Результат тесту	Тест пройдений – помилок не виявлено. Отримали червоне повідомлення "Not knu user"

Таблиця 22. Тест "correctUserLogin"

Показник	Значення
Опис тесту	Перевіряється на коректність результат надсилання запиту на реєстрацію та інтерфейс користувача
Початкові умови	email = "yulia.tokan.11@knu.ua" password="qwe12345" name="johny" surname="test"
Очікуваний результат	Перехід на головну сторінку після успішної реєстрації
Результат тесту	Тест пройдений – помилок не виявлено

ІНСТРУКЦІЯ КОРИСТУВАЧА

"Smart Cource" допомагає викладачам та учням легше підтримувати навчальний процес при дистанційному навчанні. Також ця система містить функції для саморозвитку.

Доступ до головної сторінки є у всіх користувачів. Натиснувши на клавішу "Pomodoro" можна перейти на сторінку з функціями pomodoro timer та використовувати її для ефективнішої роботи над завданнями. Також на сторінці є можливість подивитись випадкову сторінку з вікіпедії, натиснувши на "View" під блоками "Something interesting". Сторінки генеруються випадково щоб зробити різноманітнішим світогляд користувача.

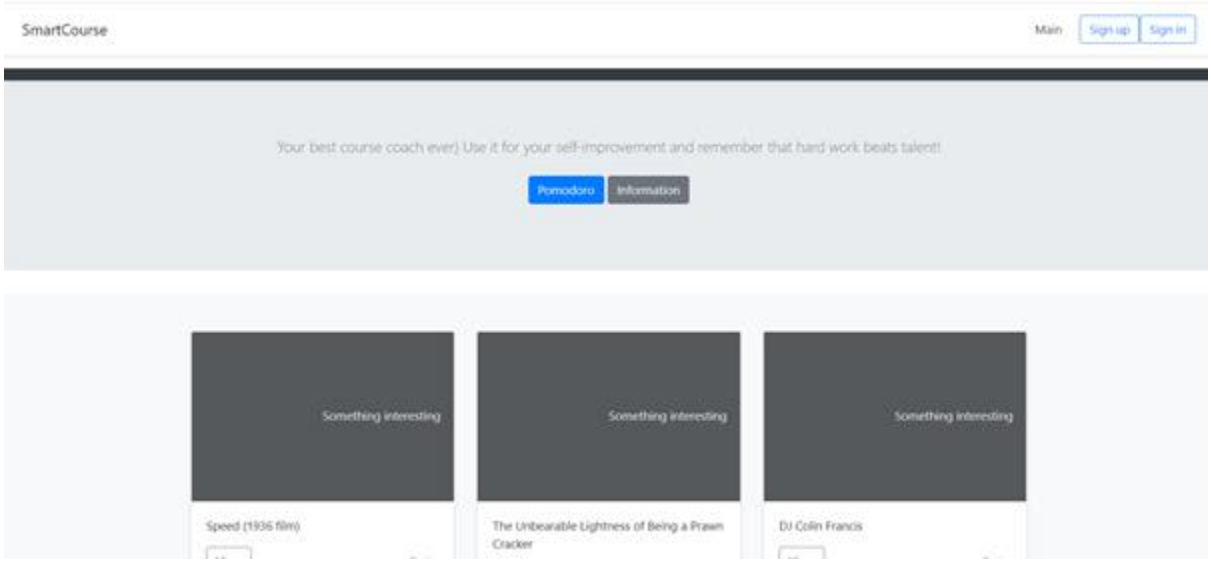


Рисунок 6. Головна сторінка сайту

Щоб мати доступ до розширеного функціоналу потрібно увійти як користувач. Якщо ще нема акаунта, то зареєструватись можна, натиснувши кнопку "Sign Up" і потім ввести потрібні дані. Для них працює перевірка значень, тому дотримуйтесь коректності в даних та майте на увазі, що зареєструватись можуть тільки користувачі, в яких є пошта з доменом knu.ua.

Рисунок 7. Форма реєстрації

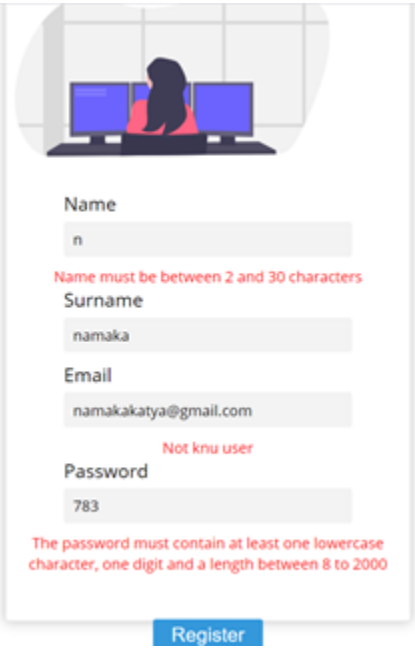


Рисунок 8. Форма реєстрації з некоректними даними

Якщо ви вже маєте аккаунт, то просто введіть пароль та логін і система вас упізнає)

Рисунок 9. Форма входу з правильними значеннями

Якщо дані коректні, то ви потрапляєте на свою власну сторінку, де вже є витягнутий список ваших справ, які потрібно зробити.

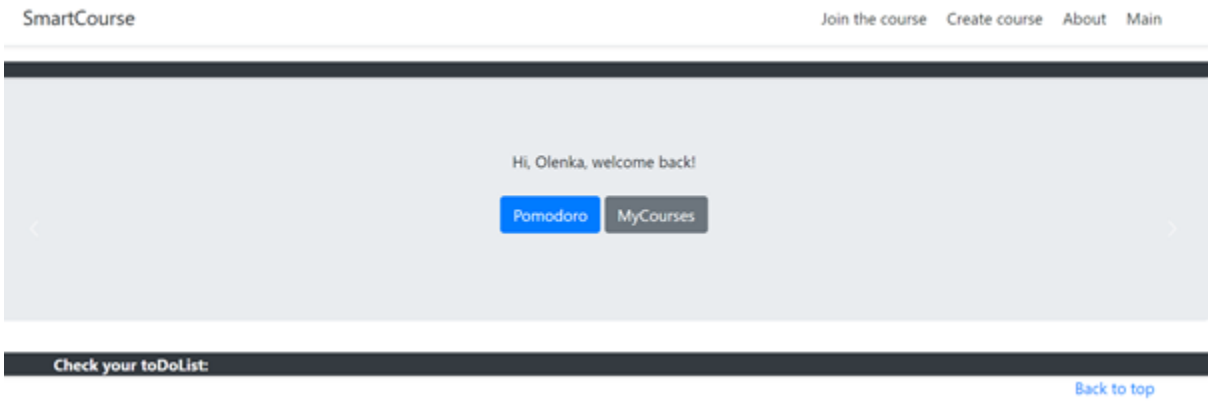


Рисунок 10. Головна сторінка користувача

Якщо хочете переглянути свої курси, то натисніть на "MyCourses". Якщо ви вчитель в якомусь курсі, або просто хочете створити свій курс, то натисніть на "Create course" у верхньому правому куті. Потім ви побачите поле, де потрібно ввести назву курсу:

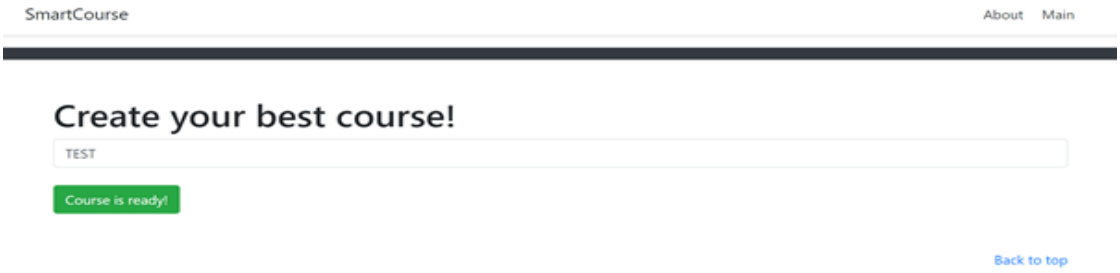


Рисунок 11. Сторінка створення курсу

Натисніть "Course is ready!" і ваш курс готовий) Його головна сторінка буде виглядати так (На сторінці будуть всі пости, якими ви захочете поділитись з учнями):

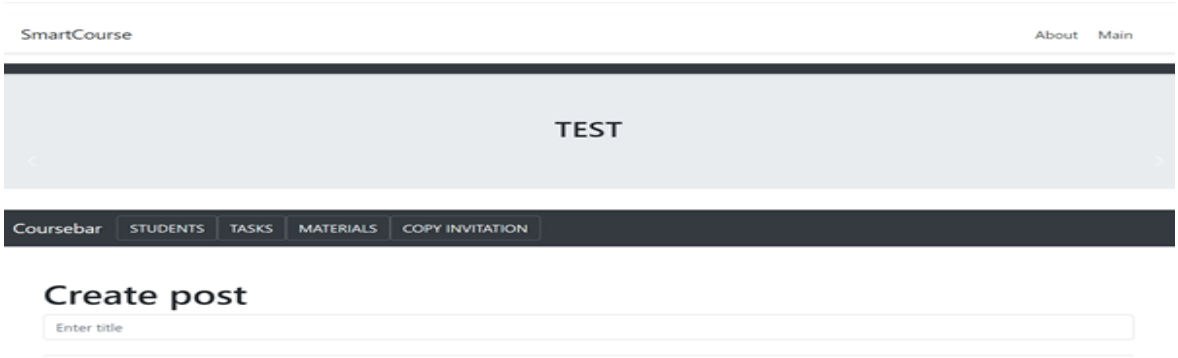


Рисунок 12. Головна сторінка курсу для вчителя

Щоб створити новий пост, просто введіть всі потрібні дані стосовно нього, прикріпіть документ, якщо це потрібно і потім натисність "ADD POST" і він автоматично з’явиться на головній сторінці курсу для вас і ваших учнів.

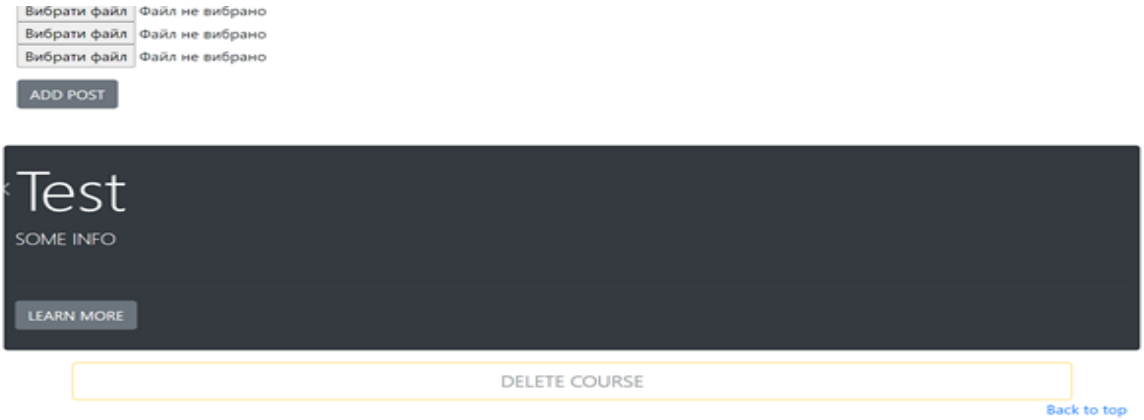


Рисунок 13. Створення поста

Якщо захочете видалити курс, то просто натисніть "DELETE COURSE".

Щоб дати можливість учням приєднуватись до курса натисніть "COPY INVITATION", скопіюйте згенероване посилання та надішліть його учням. Саме за ним вони зможуть приєднатися до курсу.

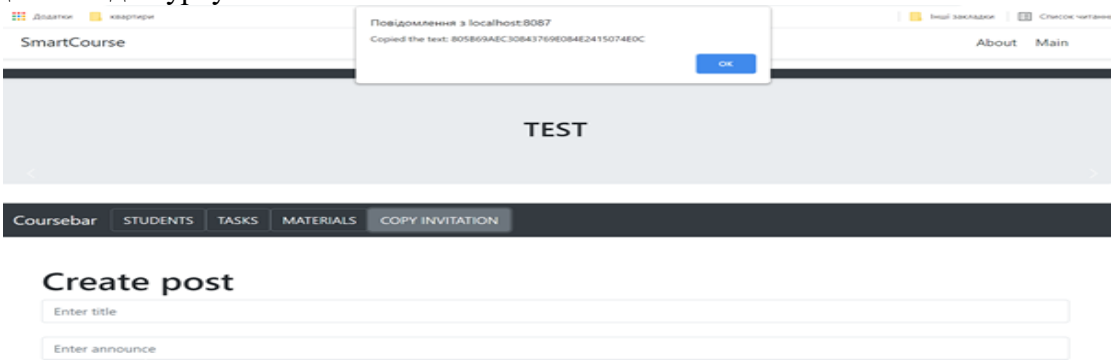


Рисунок 14. Сторінка з запрошенням до курсу

Якщо ви учень та хочете приєднатися, то скопіюйте посилання, яке кинув вчитель, натисніть "Join course" на своїй головній сторінці та вставте посилання у віконце.

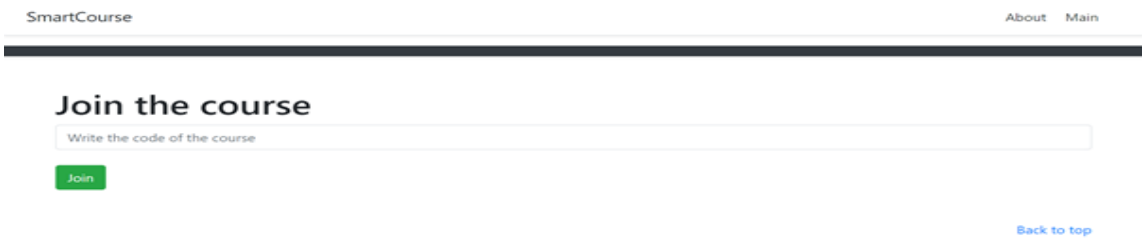


Рисунок 15. Сторінка приєднання до курсу

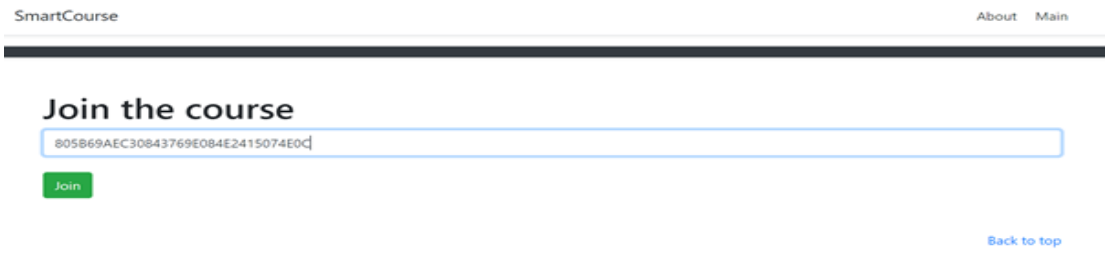


Рисунок 16. Сторінка приєднання до курсу

Після цього ви зайдете на свою сторінку курсу, де зможете переглянути всі пости вчителя. Також, натиснувши на "TASKS", переглянути всі завдання, які потрібно здати. І якщо воно готове, то завантажити зроблене).

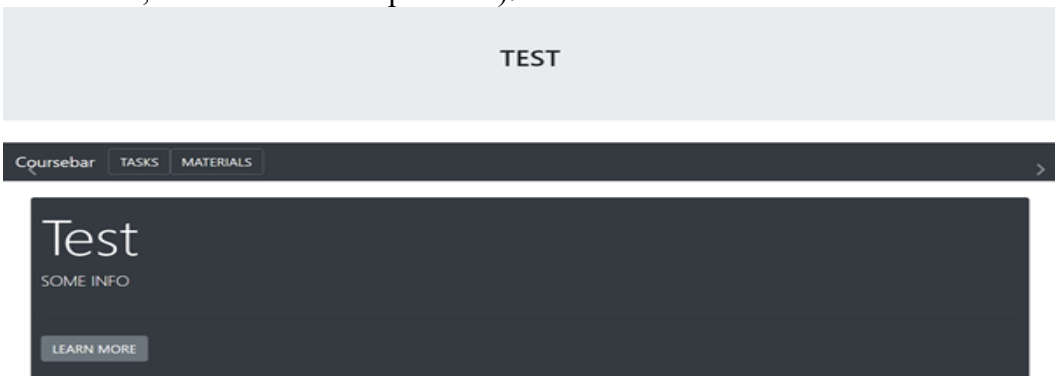


Рисунок 17. Головна сторінка курсу для учня

Якщо ви вчитель, то створити завдання можна натиснувши на "TASKS" та створити нову. Якщо потрібно файли, то є можливість їх завантажити щоб потім учень міг їх собі завантажити. Також є можливість поставити дедлайн.

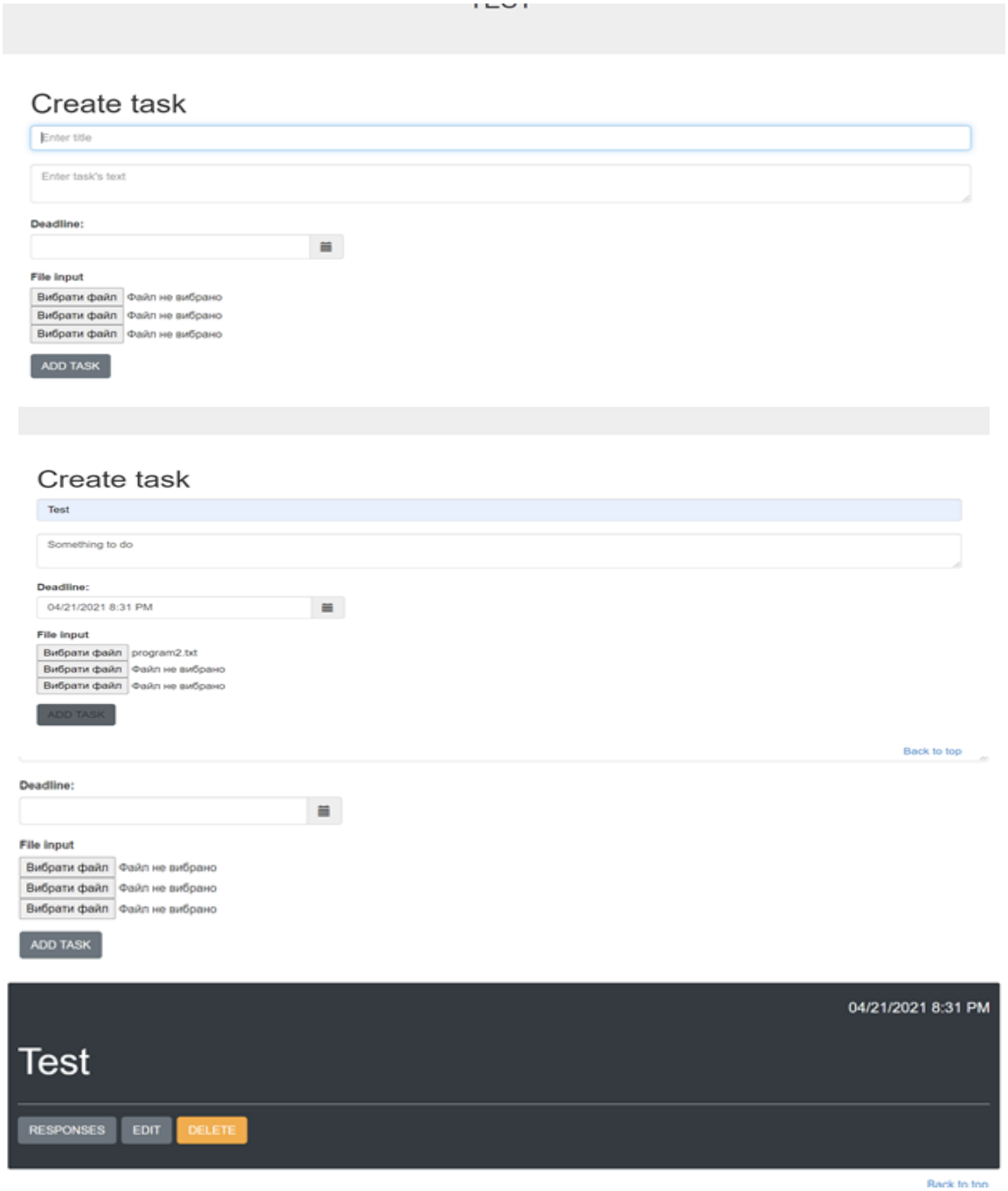


Рисунок 18. Створення завдання

Після створення ви все одно можете редагувати завдання та видаляти його. Щоб переглянути відповіді – просто натисніть "RESPONSES".

З боку учня це виглядатиме так:

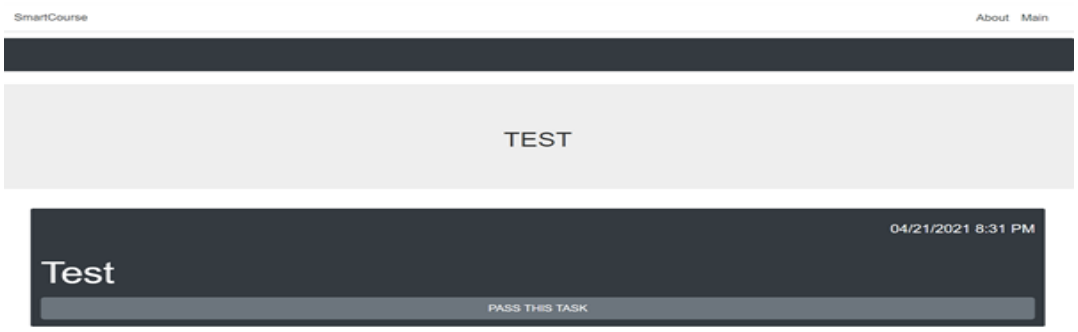


Рисунок 19. Головна сторінка з завданнями для учня

Щоб здати завдання потрібно натиснути "PASS THIS TASK" І введіть всі необхідні поля та завантажте всі файли. Також ви можете залишити коментар до цього завдання. Можливо вчитель вам відповість).

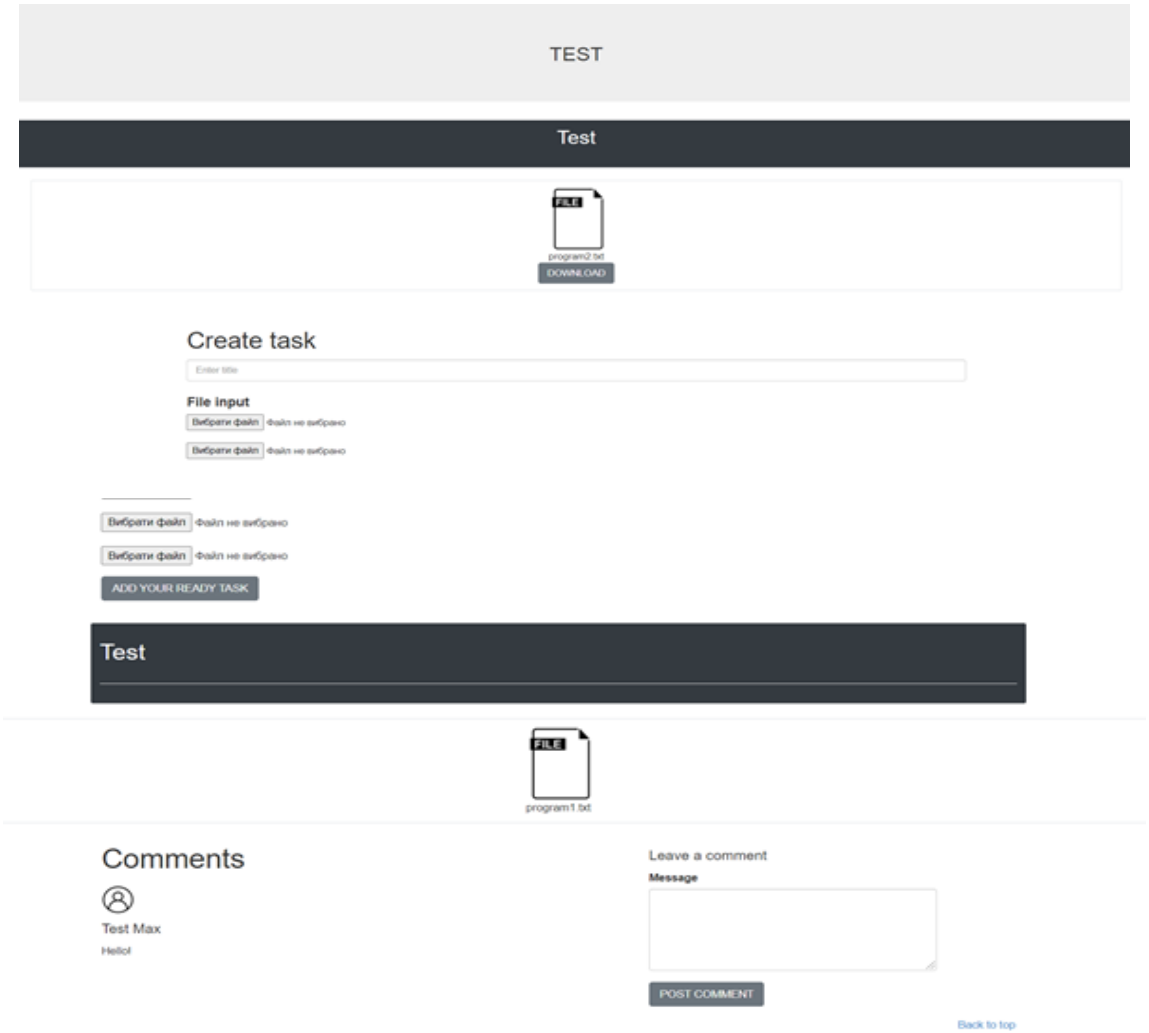


Рисунок 20. Здача завдання

Коли завдання здано, то у вчителя буде клавіша "Estimate", де можна буде оцінити учня оцінкою з потрібного проміжку. Це допомагає уникнути помилок, коли максимум 15, а у учня стоїть 20) Також є дуже корисний антиплагіат, який перевіряє на списування. Особливо це корисно при здачі коду).

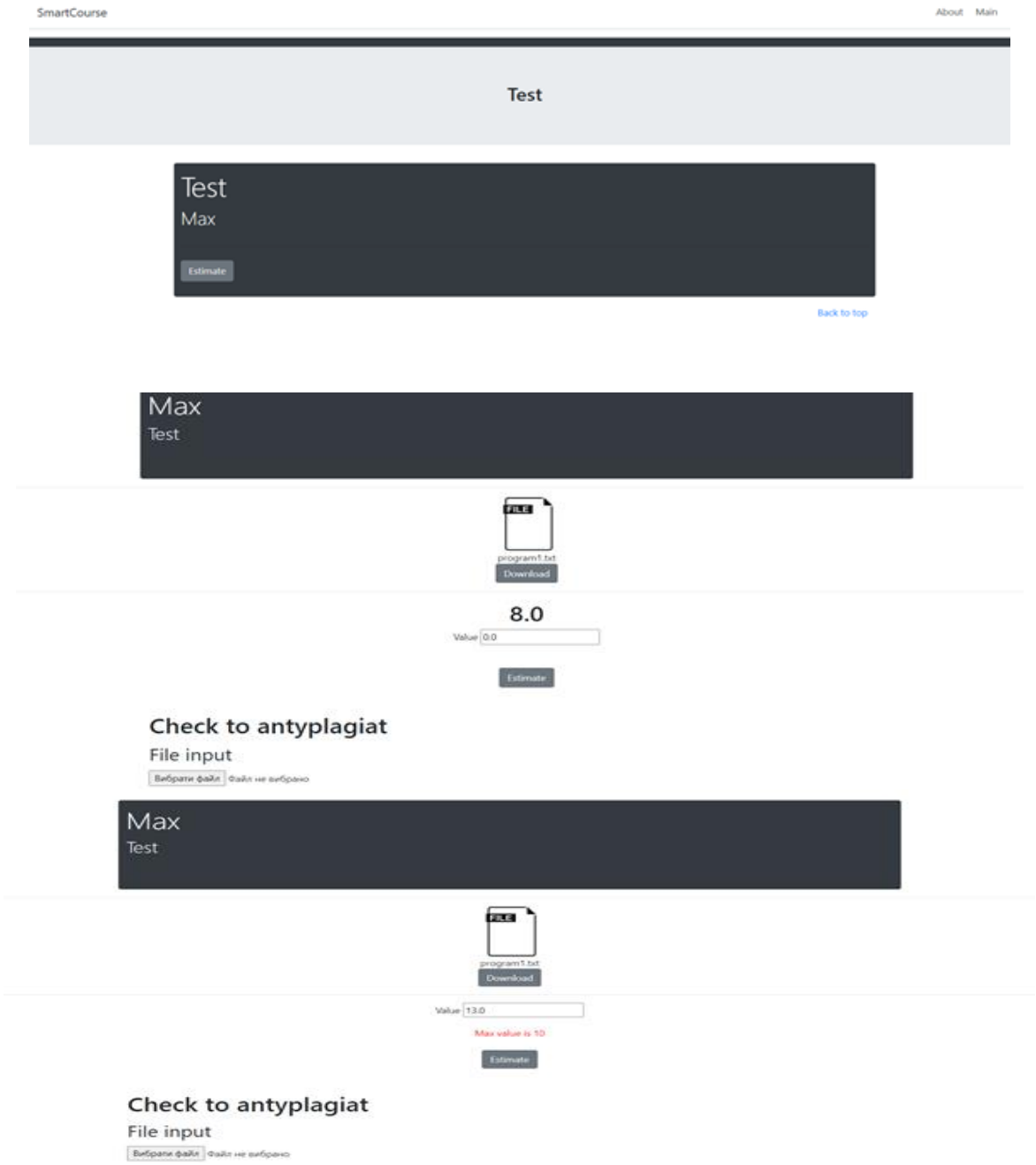


Рисунок 21. Оцінка завдання

Щоб переглянути список всіх учасників курсу потрібно натиснути на "Course participants".

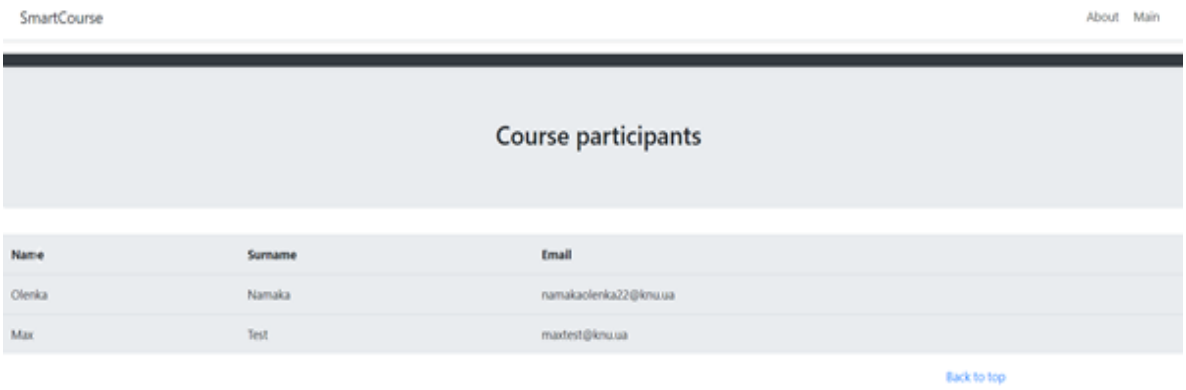


Рисунок 22. Список учасників

АНТИПЛАГІАТНА СИСТЕМА

Нормалізація і токенізація. Завдання нормалізації полягає в перетворенні програмного коду в деякий універсальний стандартний текстовий вигляд [14] , який можна досліджувати стандартними рядковими методами. . Так як одним із способів приховування плагіату є використання подібних ідентифікаторів мови програмування (наприклад, int і long), то було б зручно представляти ці ключові слова в нормалізованому тексті однаковим способом. Для цього потрібно попередньо виконати лексичний аналіз мови, знайти всі ключові слова й визначити їх тип. По іншому це можна назвати токенізацією коду[15].

Для початку необхідно визначити всі ключові слова мови та згрупувати подібні. Після цього виконати заміну в коді цих слів однаковими ідентифікаторами. Найкращим способом є заміна всіх ключових слів на символи.

Алгоритм нормалізації повинен бути стійким щодо перейменування змінних та функцій. Тому необхідно кожній змінній надавати власний ідентифікатор. Якщо ідентифікатори будуть різними для кожної змінною , то процент плагіату можна буде зменшити шляхом перестановки рядків, а якщо для всіх змінних зробити один ідентифікатор це навпаки призведе до збільшення рівня плагіату. Виходячи з цього було вирішено виконувати токенізацію змінних та функцій тим токеном, який відповідає типу змінної або типу результату, який повертає функція.

У реалізації було створено універсальний файл для мов С-подібних мов та Java, у якому зберігаються всі роздільники та ключові слова, які можна використати при написанні коду.

Із всього вищесказаного отримали наступні кроки алгоритму нормалізації програмного коду:

1) зчитати файл із граматикою мови);

2) видалити з коду коментарі та не програмний текст;

3) знайти та замінити всі ключові слова на відповідні їм токени;

4) знайти та замінити числа на токен, який відповідає числовому типу даних;

5) решту використаних слів, які не є ключовими для даної мови, замінити на токен, який знаходиться зліва чи справа від цього слова.

Метод пошуку найдовшого рядка. Найдовший спільний підрядок [16] – підрядок двох і більше рядків, котрий має найбільшу довжину. Формально, найдовшим спільним підрядком рядків s_1, s_2,…s_n називається рядок w*, для якого виконується умова ||w*|| = max

({||w|| | w ⊂ s_i, i = 1,…n}), операція w ⊂ s_i позначає те, що рядок w є підрядком рядка s_i. Для знаходження найдовшого спільного підрядка двох рядків s_1 і s_2, довжина яких m і n відповідно, необхідно заповнити матрицю A_{ij} розміром (m+1) x (n+1), дотримуючись певного алгоритму (рис. 23).

$$\begin{cases} A_{0j} = A_{i0} = 0, & j = 0 \dots n, \quad i = 0 \dots m, \\ A_{ij} = 0, \quad s_1[i] \neq s_2[j], & i \neq 0, \quad j \neq 0, \\ A_{ij} = A_{i-1,j-1}, \quad s_1[i] = s_2[j], & i \neq 0, \quad j \neq 0. \end{cases}$$

Рисунок 23. Алгоритм для заповнення матриці A_{ij}.

Найбільше число A_{uv} в матриці являє собою довжину найдовшого спільного підрядка. Сам підрядок знаходиться за наступним правилом: s_1[u-A_{uv}+1]…s_1[u] та s_2[v-A_{uv}+1]…s_2[v].

Рівень плагіату визначається відношенням отриманої довжини спільного підрядка до довжини рядка, який досліджуємо на плагіат(рис. 24).

$$r(s_1, s_2) = 1 - \frac{|LCS(s_1, s_2)|}{|s_1|}$$

Рисунок 24. Формула для рівня плагіату в методі найдовшого спільного рядка

Проте у цього алгоритму є два основні недоліки: використання великої кількості пам’яті (що нівелюється скороченням коду програми в процесі нормалізації) та нестійкість перед доданими у код непотрібними символами. Щоб зробити алгоритм стійким у даному випадку, повторюватимемо пошук спільного підрядка й видалятимемо його з коду до тих пір, поки не залишиться лише оригінальна частина.

Метод шинглів. Алгоритм шинглів [17] – алгоритм, розроблений для пошуку копій та дублікатів тексту у веб-документі. Засіб для виявлення плагіату . Шингли (англ. shingles – луски) – виділені з тексту підпослідовності слів . Алгоритм складається з наступних кроків:

1) нормалізація тексту;
2) розбиття тексту на шингли;
3) обрахування хеш-значень отриманих шинглів;
4) випадкова вибірка певної кількості значень контрольних сум;
5) порівняння отриманих вибірок.

Рівень плагіату знаходиться за формулою зображеною на рис. 25, де S(s) є множиною обраних хеш-значень шинглів рядка.

$$r(s_1, s_2) = \frac{|S(s_1) \cap S(s_2)|}{|S(s_1) \cup S(s_2)|}$$

Рисунок 25. Рівень плагіату для методу шинглів.

У алгоритмі прийшлося зробити декілька змін. По-перше, випадкову вибірку було вирішено не обирати, щоб зробити результат виконання алгоритму сталим для однакових вхідних даних. По-друге, Формулу обрахунку коефіцієнту плагіату було змінено (рис.7.3.2).

$$r(s_1, s_2) = \frac{|S(s_1) \cap S(s_2)|}{|S(s_1)|}.$$

Рисунок 26. Нова формула рівня плагіату.

Це дозволяє оцінювати оригінальність роботи без огляду на розмір роботи, яку було використано за основу , оскільки інколи студент може використати у своїй роботі не весь код оригінальної програми, а лише частину.

Оскільки обидва методи повертають хоч якийсь відсоток плагіату, потрібно було обрати мінімальну довжину спільного підрядка для методу пошуку найдовшого спільного рядка та розмір шинглів у другому. Розмірковуючи над цим питанням було вирішено підібрати структуру коду, яка б використовувалася часто у різних програмах на С-подібних мовах. Найбільш очевидною структурою став цикл for. Довжина рядка об'явлення такого циклу складає 16 символів. Наприклад, for (int i = 0; i < 20; i++) перетвориться у f(nn=n;n<n;n++).

ОРГАНІЗАЦІЯ РОБОТИ КОМАНДИ

Для організації командної роботи було обрано методику Scrum. Scrum - це методика, яка допомагає командам вести спільну роботу. Методику Scrum найчастіше застосовують команди розробників додатків, але принципи та досвід її використання можна застосувати до командної роботи будь-якого роду. Це одна з причин такої популярності методики. Учасники команди Scrum проводять збори, використовують спеціальні інструменти і приймають на себе особливі ролі, щоб організувати роботу і керувати нею. Робота команди складалася з планування спринта, зборів команди для наради та ретроспективи.

Спринт - це фактичний проміжок часу, протягом якого команда Scrum спільно працює над вирішенням певної задачі, яка була запланована на цей спринт. У нашій команді тривалість спринтів займала від одного до двох тижнів. На цей період кожному учаснику команди ставилися конкретні задачі, вирішення яких призвело до фінального продукту.

Scrum-нарада - це дуже коротке щоденне зібрання, яке для зручності проводиться в один і той же час і в одному і тому ж місці. Багато команд намагаються вкластися в 15 хвилин, однак це лише рекомендація. У нашому випадку було прийнято рішення збиратися 3-4 рази на тиждень для того, щоб кожен учасник команди був в курсі того, що відбувається і не відхилявся від мети. Під час цих зустрічей кожен учасник команди доповідав, що було зроблено з минулої зустрічі, що він планує зробити до наступної зустрічі і чи виникали у нього якісь проблеми. Також кожного тижня проводилася нарада, де обговорювалося виконання спринта, які проблеми виникали і що можна вдосконалити у наступному спринті.

Ретроспектива проводиться, щоб команда зафіксувала і обговорила всі успіхи і невдачі спринту, проєкту, учасників та їх взаємовідносин, інструментів тощо. Мета ретроспективи - створити умови, щоб команда могла приділити увагу всьому, що вдалося і що потрібно поліпшити в наступний раз, і не зациклювалася на невдачах.

Під час роботи над проєктом було допущено такі помилки:

- команда відразу прийнялася за розробку, без формулювання усіх вимог і моделювання проєкту, що призвело до уповільнення роботи над проєктом і певних проблем з реалізацією

- не правильно сформульована задача для формальної специфікації, що призвело до потреби модифікації готового коду і затягнуло час виконання задачі

- не правильно спроєктована база даних, що призвело до затрат часу на зміну логіки проєкту

Завдяки методиці scrum, усі помилки було виявлено, обговорено і більше не допущено в наступних спринтах. Ця методика значно полегшила роботу команди і допомогла вчасно довести проєкт у фінальну стадію.

Що з запланованого не було реалізовано:

- середовище для розробки лабораторних робіт з хімії та фізики
- підтвердження пошти під час реєстрації
- впровадження капчи під час реєстрації
- завантаження картинок до статті з вікіпедії

Задачі не були виконані, через те, що вони були менш пріоритетними. Через те, більше уваги приділялося на основний функціонал, і за браку часу вище перераховані задачі були усунені від подальшої розробки.

Результати усіх зустрічей було записано до таблиці (див. додаток Б).

ВИСНОВКИ

Електронне навчання, як і будь-який навчальний процес, крім змістовної частини обов'язково включає організаційний компонент. Система LMS, в ідеалі, повинна надавати кожному студентові персональні можливості для найбільш ефективного вивчення матеріалу, а менеджерові навчального процесу - необхідні інструменти для формування навчальних програм, контролю їх проходження, складання звітів про результативність навчання, організації комунікацій між студентами і викладачами.

Студент отримує можливості доступу до навчального порталу, який є відправною точкою для доставки всього навчального контенту, вибору відповідних учбових треків на основі попереднього і проміжних тестувань, використання додаткових матеріалів за допомогою спеціальних посилань.

Адміністративні функції охоплюють декілька базових областей. Управління студентами включає в себе завдання реєстрації і контролю доступу користувачів до системи і до учбового контенту, організацію слухачів в групи для надання їм загальних курсів і складання звітності, управління аудиторними і викладацькими ресурсами. Відповідає також за інтеграцію додаткових елементів навчального процесу (практичні заняття, лабораторні роботи, тести, засоби спільної роботи, посилання на зовнішні матеріали і ін.)

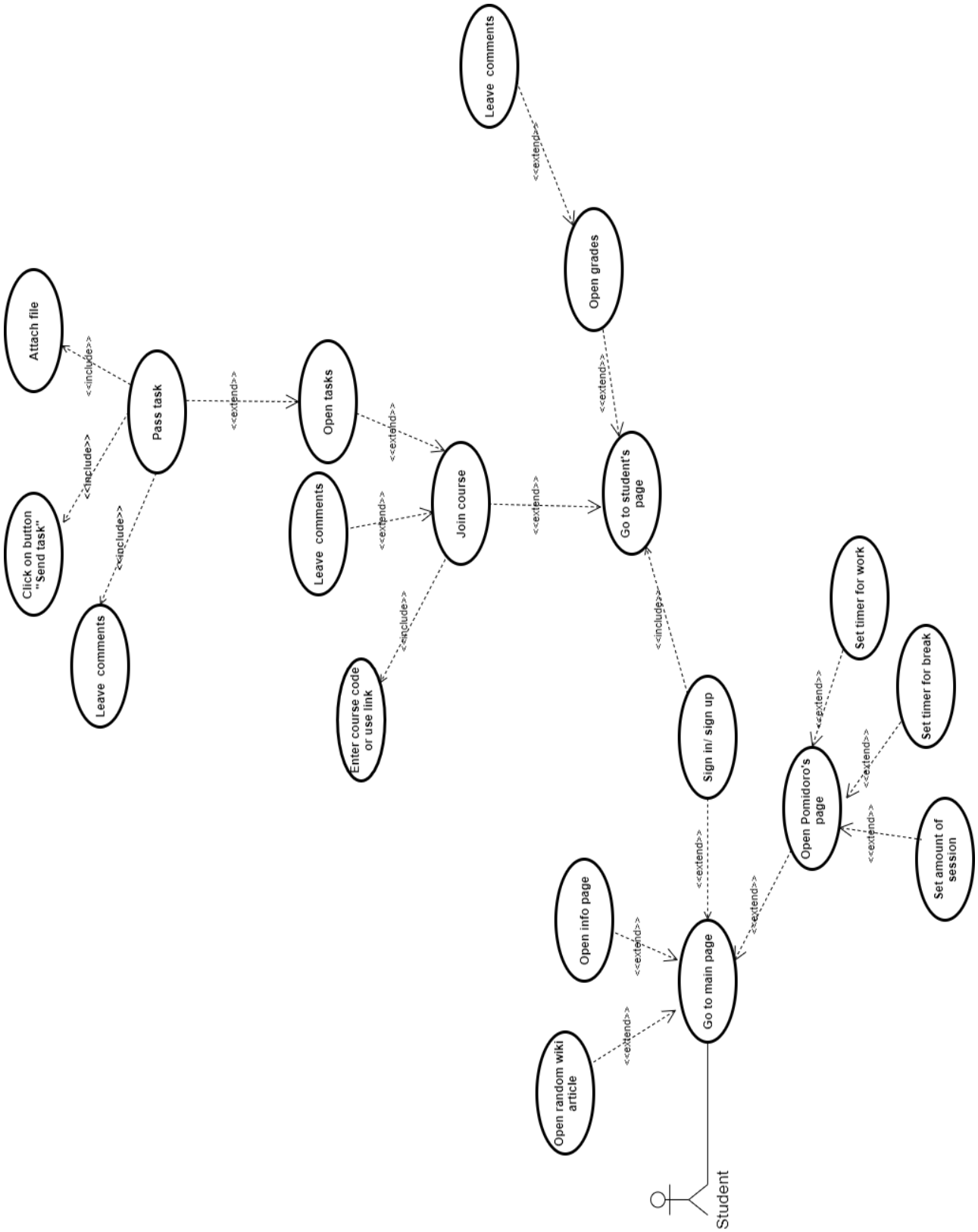
Ринок LMS поки ще достатньо фрагментований, що свідчить про його незрілості, проте він швидко розвивається; системи цього класу стають все більш затребуваними і розглядаються не просто як необхідна інфраструктура для e-Learning.

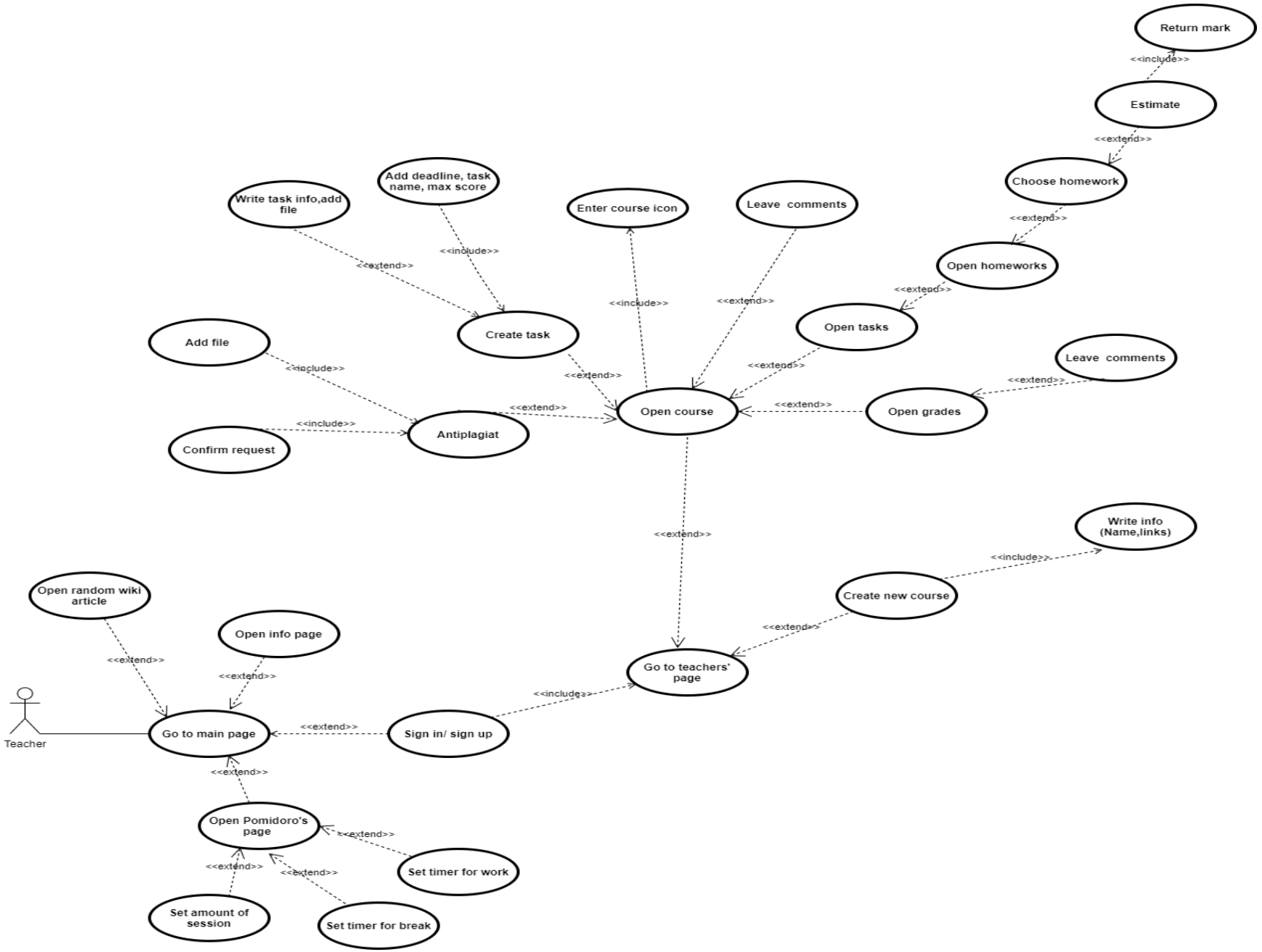
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Classroom [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://classroom.google.com/ .
2. Moodle [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://moodle.org/
3. ILIAS [Електронний ресурс] – Режим доступу до ресурсу: https://www.ilias.de/

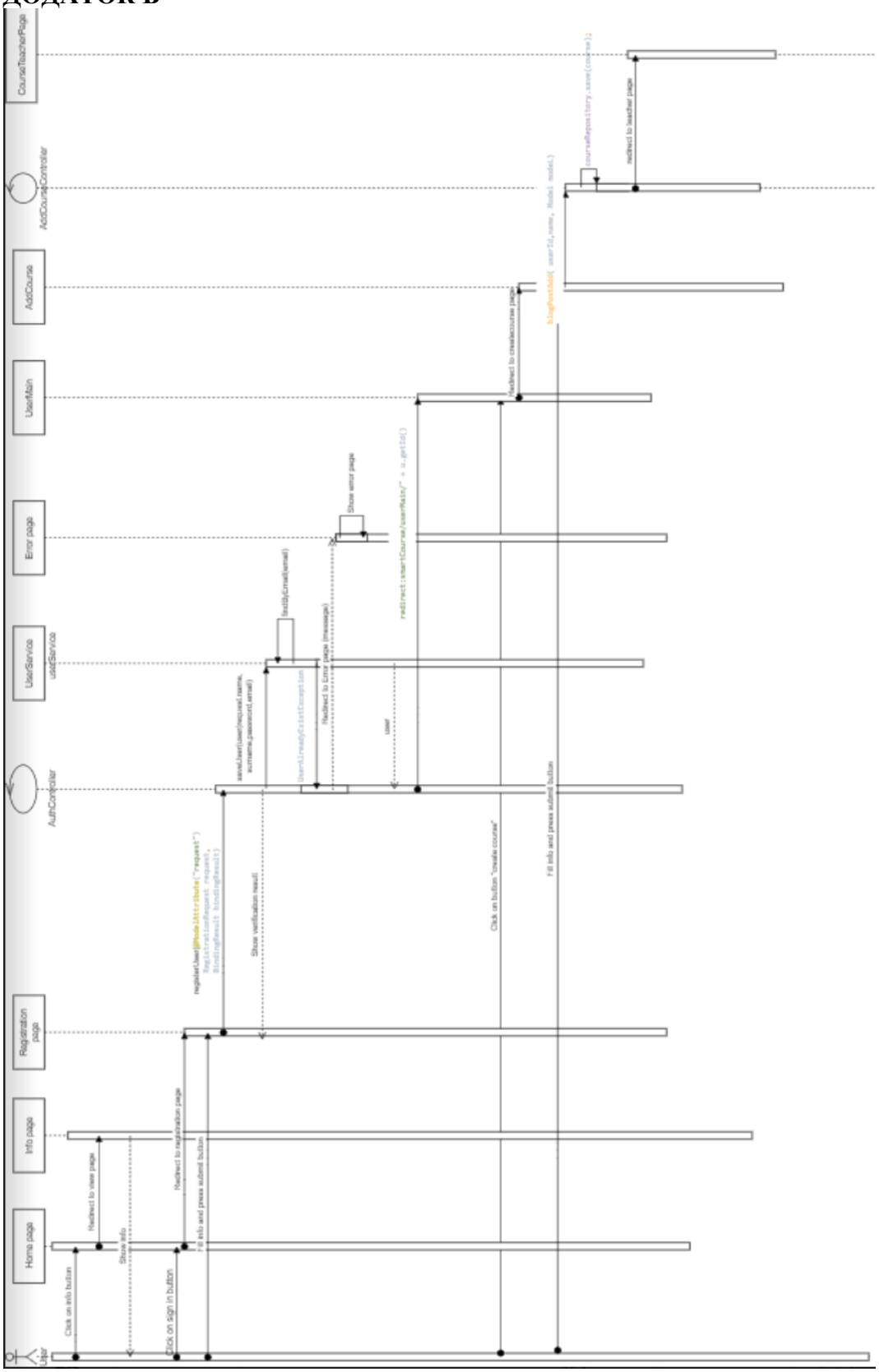
4. Unicraft [Електронний ресурс] – Режим доступу до ресурсу: https://www.unicraft.org/ .
5. Schoology [Електронний ресурс] – Режим доступу до ресурсу: https://www.schoology.com/ .
6. Java [Електронний ресурс] – Режим доступу до ресурсу: https://www.java.com/
7. Apache Maven [Електронний ресурс] – Режим доступу до ресурсу: https://maven.apache.org
8. Apache [Електронний ресурс] – Режим доступу до ресурсу: https://httpd.apache.org/
9. Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/projects/spring-framework
10. Spring [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/
11. JMX API [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javase/7/docs/technotes/guides/jmx/index.html
12. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/
13. Dafny [Електронний ресурс] – Режим доступу до ресурсу: https://rise4fun.com/Dafny
14. Совершенный код: нормализация данных [Електронний ресурс] – Режим доступу до ресурсу: https://ru.hexlet.io/blog/posts/sovershennyy-kod-normalizatsiya-dannyh
15. Lexical analysis [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Lexical_analysis
16. Longest common substring problem [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Longest_common_substring_problem
17. w-shingling [Електронний ресурс] – Режим доступу до ресурсу: https://nlp.stanford.edu/IR-book/html/htmledition/near-duplicates-and-shingling-1.html

ДОДАТОК А

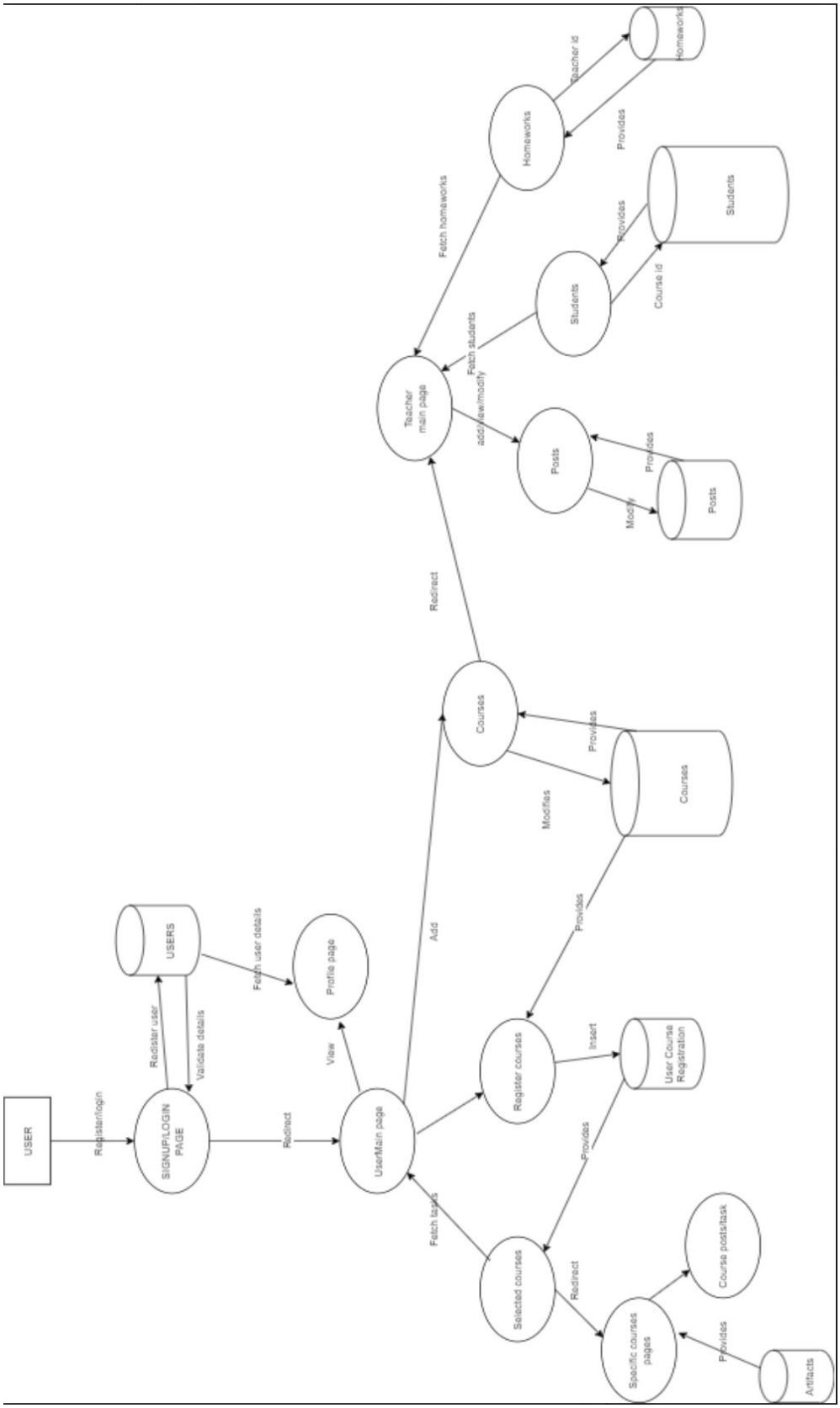

Leave comments
<<extend>>
Open grades
<<extend>>
Go to student's page
<<include>>
Sign in/ sign up
<<extend>>
Go to main page
Student
Open info page
<<extend>>
Open random wiki article
<<extend>>
<<extend>>
Open Pomidoro's page
<<extend>>
Set timer for work
<<extend>>
Set timer for break
<<extend>>
Set amount of session
<<extend>>
<<extend>>
Join course
<<include>>
Enter course code or use link
<<extend>>
Leave comments
<<extend>>
Open tasks
<<extend>>
Pass task
<<include>>
Leave comments
<<include>>
Click on button "Send task"
<<include>>
Attach file


Return mark
<<include>>
Estimate
<<extend>>
Choose homework
<<extend>>
Open homeworks
<<extend>>
Open tasks
<<extend>>
Leave comments
<<extend>>
Open grades
<<extend>>
Open course
<<extend>>
Enter course icon
<<include>>
Leave comments
<<extend>>
Add deadline, task name, max score
<<include>>
Write task info,add file
<<extend>>
Create task
<<include>>
Add file
<<include>>
Antiplagiat
<<include>>
Confirm request
<<extend>>
Write info (Name,links)
<<include>>
Create new course
<<extend>>
Go to teachers' page
<<include>>
Sign in/ sign up
<<extend>>
Go to main page
<<extend>>
Open random wiki article
<<extend>>
Open info page
<<extend>>
Teacher
<<extend>>
Open Pomidoro's page
<<extend>>
Set timer for work
<<extend>>
Set timer for break
<<extend>>
Set amount of session

ДОДАТОК Б


CourseTeacherPage
AddCourseController
AddCourse
UserMain
Error page
UserService
AuthController
Registration page
Info page
Home page
User
Click on info button
Redirect to view page
Show info
Click on sign in button
Redirect to registration page
Fill info and press submit button
Show verification result
Click on button "create course"
Fill info and press submit button
Redirect to createcourse page
Show error page
redirect:/smartCourse/userMain/
Redirect to CourseTeacherPage
redirect to teacher page

ДОДАТОК В


USER
Register/login
SIGNUP/LOGIN PAGE
Register user
Validate details
USERS
Redirect
Fetch user details
Profile page
View
User/Main page
Add
Courses
Redirect
Teacher main page
Fetch homeworks
Homeworks
Teacher id
Homeworks
Provides
Fetch students
Students
Provides
Students
Course id
add/view/modify
Posts
Provides
Posts
Modify
Provides
Courses
Modifies
Courses
Register courses
Insert
User Course Registration
Provides
Fetch tasks
Selected courses
Provides
Redirect
Specific courses pages
Course posts/task
Provides
Artifacts

ДОДАТОК Г

Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
03/02/2021	Знайомство. Основні ідеї для проєкту за темою "Облік лабораторних робіт". Обговорення інструментарію для проєктування та написання проєкту.Вибір початкових ролей команди. Спілкування з замовниками.	Обдумати ідеї над проєктом. Обдумати базу даних.	Обдумати ідеї над проєктом. Обдумати базу даних.	Обдумати ідеї над проєктом. Обдумати базу даних.	Обдумати ідеї над проєктом. Обдумати базу даних.	Обдумати ідеї над проєктом. Обдумати базу даних.	Відбулося знайомтсво учасників команди. Було обрано ролі в команді для кожного учасника. Вибрано інструментарій для написання проєкту(MySQL, Spring Framework, JAVA). Установлено основні ідеї нашого проєкту(схожий до класруму + антиплагіат система)
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
05/02/2021	Замовники уточнили умови проєкту. Обговорення таблиць та полів бази даних.	Обдумати краще базу даних.	Обдумати краще базу даних. Початок роботи над специфікацією.	Обдумати краще базу даних.	Обдумати краще базу даних.	Обдумати краще базу даних.	Установилося загальне бачення того як має виглядати проєкт(робота з організацією курсів, перевіркою домашніх завдань різних типів, антиплагіат)Розроблена перша версія бази даних. Створена дошка завдань на Trevo.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
08/02/2021	Коротке обговорення бази даних і деяких нюансів пов'язаних з нею. Обговорення деяких моментів специфікації.	Розробити макет вигляду сайту. Специфікація.	Специфікація	Обдумати краще базу даних. Початок праці над кодом.	Обдумати краще базу даних. Початок праці над кодом.	Специфікація	Створено другу версію бази даних. Обговорення програмних компонентів проєкту. Обговорено деякі програмні компоненти програми.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
09/02/2021	Обговорення специфікації та макету дизайну проєкту. Обговорення бази даних.	Специфікація	Специфікація	Обдумати краще базу даних. Початок праці над кодом.	Обдумати краще базу даних. Початок праці над кодом.	Специфікація	Установлення бета версії вигляду сайту. Обговорено основні можливості рлей студента і вчителя. Продовження обговорення програмних компонентів.

Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
10/02/2021	Розмова з замовниками. Відкат назад. Обговорення діаграм для проєкту та програмних компонентів.	Специфікація. Дана таблиця)	Специфікація	UML-діаграми. Робота над кодом.	UML-діаграми. Робота над кодом.	Специфікація	Переосмислення потреб замовників. Установленя нового курсу для роботи.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
12/02/2021	Обговорення головної сторінки. Генерування рандомних статей з вікіпедії. ПРОБЛЕМА - Як вставити картинку зі статті: 1) не у всіх статей вона є; 2) не ясно як це зробити (як правильно розпарсити відповідь GET запиту і знайти картинку, як вставити картинку на сайт)	Допрацювання документу технічних вимог	Допрацювання документу технічних вимог, розпис календарного плану	Допрацювання головної сторінки	Допрацювання головної сторінки	Допрацювання документу технічних вимог	Вирішено проблему таким чином: робити GET-запит, отримувати у відповідь html, розпарсити відповідь, отримати тег img, взяти з нього src, вставляєму цю картинку собі на сайт.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
17/02/2021	Обговорення ідей як зробити можливість скачувати файли	Допрацювання документу технічних вимог	Допрацювання документу технічних вимог, розпис календарного плану	Допрацювання головної сторінки	Розібратися з підключенням mysql і spring security	Допрацювання документу технічних вимог	Вирішено зберігати файл як масив байтів в базі даних, для відправки файлу: коли надсилається запит, то він має output stream, в нього записується цей файл як масив байтів, потім поставити content-type відповідно до формату файлу
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
22/02/2021	Зміни в базі даних	Допрацювання документу технічних вимог	Допрацювання документу технічних вимог	Допрацювання бази даних	Допрацювання бази даних	Допрацювання документу технічних вимог	Треба зробити перехідну табличку між юзером і курсом, бо оцінок може і не бути. Краще зробити табличку з ролями. Треба передбачити можливість, що юзер в одному курсі може бути вчителем, а в іншому учнем.

Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
26/02/2021	Деталі програмування проєкту. План тестування логіки системи.	Визначення порядку тестування. Допомога з реалізацією основної логіки проєкту.	Визначення порядку тестування. Допомога з реалізацією основної логіки проєкту.	Побудова базової архітектури системи.	Побудова базової архітектури системи.	Визначення порядку тестування. Допомога з реалізацією основної логіки проєкту.	Потрібно доробити базовий функціонал системи, після чого приступати до тестування.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
01/03/2021	Було розглянуто головну сторінку, сторінку реєстрації, логіну, було обговорено спосіб долучатися студентам до курсу	Дослідити методи формальної специфікації	Дослідити методи формальної специфікації	Допрацювання багів, об'єднання двох частин проєкту	Допрацювання верифікації при реєстрації, об'єднання двох частин проєкту	Дослідити методи формальної специфікації	Потрібно об'єднати дві частини проєкту та повиправляти баги, після чого приступати до тестування реєстрації. Потрібно дослідити методи формальної специфікації, визначитися з тим, що підходить нам.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
02/03/2021	Було розглянуто об'єднаний проєкт. Обговорювалось тестування. Обговорення яку задачу слід обрати для формальної специфікації.	Дослідити методи формальної специфікації. Тестування реєстрації. Дослідити теоретичну частину методів антиплагіату	Дослідити методи формальної специфікації. Тестування реєстрації.	Допрацювання багів. Початок робота над власне курсами в проєкті.	Допрацювання верифікації при реєстрації, початок роботи над власне курсами в проєкті	Дослідити методи формальної специфікації. Тестування реєстрації.	Обрано задачу для формальної специфікації (антиплагіат). Обговорено подальші дії.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення

03/03/2021	Обговорення специфікації. Зустріч з клієнтами. Обговорення діаграми потоку та послідовностей..	Дослідити методи формальної специфікації. Дослідити теоретичну частину методів антиплагіату	Дослідити методи формальної специфікації. Тестування реєстрації.	Допрацювання багів. Початок робота над власне курсами в проєкті. Діаграми	Допрацювання верифікації при реєстрації, початок роботи над власне курсами в проєкті. Діаграми	Дослідити методи формальної специфікації. Тестування реєстрації.	Змінено задачу специфікації(підрахунок суми балів.) Обговорено які діаграми ще потрібно зробити. Обрано подальші дії для кожного члена команди
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
05/03/2021	Обговорення специфікації та сторінки учителя.	Формальна специфікація. Дослідити теоретичну частину методів антиплагіату	Формальної специфікація. .	Робота над сторінкою вчителя. Діаграми	Робота над сторінкою вчителя. Діаграми	Тестування реєстрації.	Обговорено сторінку вчителя. Розбиралися у мові для специфікації Dafny.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
09/03/2021	Обговорення специфікації та тестування. Розглянуто , шо уже готова по сторінці вчителя. Розглянуто діаграму потоків і послідовностей.	Формальна специфікація.	Формальна специфікація. .	Робота над сторінкою вчителя.	Робота над сторінкою вчителя.	Тестування реєстрації.	Зроблено початок програми для специфікації. (сума масиву оцінок) Розроблено дагграму потоків та послідовностей. Фронт сторінки вчителя готовий.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
10/03/2021	Обговорення специфікації. Обговорення помилок , що були виконані на етапі проєктування.	Формальна специфікація. Допомога іншим членам команди	Формальна специфікація. Допомога іншим членам команди	Робота над сторінкою вчителя.	Робота над сторінкою вчителя.	Планування тестування сторінки вчителя. Допомога іншим членам команди	Обдумано помилки що були зроблені (початок не з проєктування а з бази даних, ігнор умл діаграм). Створені тести для реєстрації.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
12/03/2021	Обговорення довершення специфікації. Перші варіанти сторінки учителя.	Формальна специфікація. Допомога іншим	Формальна специфікація. Допомога іншим	Робота над сторінкою вчителя.	Робота над сторінкою вчителя.	Тестування сторінки вчителя. Допомога іншим членам команди	Прописано створення курсів і деякі можливості вчителя.

		членам команди	членам команди				
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
16/03/2021	Обговорення сторінки вчителя. Початок обдумування звіту.	Звіт	Звіт	Робота над сторінкою вчителя.	Робота над сторінкою вчителя.	Тестування реєстрації. Звіт	Додано створення постів в курсі. Написано ще тести. Початок роботинад звітом(огляд існуючих систем)
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
17/03/2021	Обговорення сторінки вчителя. Повернення до роботи над формальною специфікацією	Звіт. Формальна специфікація	Звіт. Формальна специфікація	Робота над сторінкою вчителя.	Робота над сторінкою вчителя.	Інтеграційні тести. Звіт	Обговорено проблеми , що виникли під час формальної специфікації
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
19/03/2021	Обговорення роботи з завданнями(обмін тасками між вчиелем і учнем	Звіт. Розробка системи антиплагіату	Звіт. Формальна специфікація	Робота над сторінкою вчителя.	Робота над сторінкою вчителя.	Інтеграційні тести. Звіт	Створену сторінку з завданнями для вчителя. Продовження роботи над звітом (огляд використаних технологій і вимог)
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
23/03/2021	Обговорення варіації тестів та написання формальної специфікації. Створення звіту для керівників	Звіт. Розробка системи антиплагіату.	Формальна специфікація	Робота над сторінкою студента.	Робота над сторінкою студента.	Тестування інтерфейсу користувача. Звіт	Створену сторінку учня. Продовження роботи над звітом (тестування пп1)
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
26/03/2021	Обговорення специфікація і розділу про бд в звіті	Звіт. Розробка системи антиплагіату.	Формальна сцифікація. Звіт	Робота над сторінкою студента.(головна сторінка)	Робота над фільтром з генерацією токена .	Тестування інтерфейсу користувача. Звіт	Рефакторинг коду. Продовження роботи над звітом (розділ бд)
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення

29/03/2021	Обговорення розділу про специфікацію. Вирішення питання з merge.	Звіт. Розробка системи антиплагіату.	Формальна сцифікація та валідація. Звіт	Сторінка тасок для студента	Розробка логіки вза'ємодії вчителя та унів	Звіт	Merge. Продовження роботи над звітом (формальна специфікація)
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
31/03/2021	Зустріч з керівниками проєкту. Розмова про те що ще слід доробити.	Звіт. Розробка системи антиплагіату.	Валідація. Звіт	Розробка логіки здачі лабораторних робіт	Розробка логіки вза'ємодії вчителя та унів	Редагування тестів. Звіт	Написано нормалізацію та токінізацію по антиплагіат системі. Закінчено зі специфікацією. В звіті описана база даних.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
02/04/2021	Обговорення антиплагіату, валідації та майбутньої презентації.	Звіт	Звіт	Розробка логіки здачі лабораторних робіт	Кнопка для приєднання до курсу та список курсів юзера	Звіт	Закінчено нормалізацію та токінізацію по антиплагіат системі. Закінчено зі специфікацією.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
05/04/2021	Підведення підсумків. Обговорення реалізованих та не реалізованих задач. Створення презентації. Звітування про організацію роботи в команді.	Звіт. Презентація.	Звіт. Презентація.	Звіт. Презентація.	Звіт. Презентація.	Звіт. Презентація.	Написаний розділ про організацію роботи в команді та інструкцію користувача.
Дата	Що обговорювалося	Кліщ Андрій	Савонік Оксана	Ярмоленко Софія	Намака Олена	Токан Юлія	Результат обговорення
06/04/2021	Підготовка до презентації. Обговорення доповіді.	Звіт. Презентація.	Звіт. Презентація.	Звіт. Презентація.	Звіт. Презентація.	Звіт. Презентація.	Створені звіт та презентація. Підготовлена доповідь.

ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ ПІДТРИМКИ НАВЧАЛЬНОГО ПРОЦЕСУ EDUCATION MANAGEMENT SYSTEM

Владислав Артюх, Валентина Пікуза, Дмитро Пінковський, Ігор Піонтковський, Богдан Юркевич

ТЕМАТИКА ПРОЄКТУ

На початку роботи учасникам команди було поставлене таке завдання - розробити систему підтримку інтегрованого курсу. В ході онлайн-зустрічей з викладачами, було уточнено тематику проєкту та необхідний функціонал для програми.

В результаті своєї роботи учасники команди спроєктували та розробили систему підтримки навчального процесу "Education management system" (далі – програмний продукт). Даний програмний продукт розроблено як є комплексне рішення для повної організації навчального процесу в університеті в онлайн режимі.

Структура та опис програмного продукту. Система має 3 основні ролі:
Адміністратор;
Лектор;
Студент.
Адміністратор має можливість:

- редагувати текстовий контент (там, де це передбачено);
- додавати контент;
- переглядати і редагувати дані користувачів;
- блокувати , видаляти користувачів;
- формувати групи, курси та команди.

Лектор має можливість:

- створювати групу, курс, підгрупу;
- додавати , видаляти користувачів (інших лекторів та студентів);
- відправляти особисті повідомлення учасникам навчального процесу;
- створювати завдання;
- виставляти оцінку за виконані завдання;
- додавати інформацію необхідну для навчального процесу;
- редагувати особисту інформацію.

Студент має можливість:

- відправляти особисті повідомлення учасникам навчального процесу;
- виконувати завдання;
- додавати інформацію необхідну для навчального процесу;
- залишати коментарі;
- редагувати особисту інформацію.

Зареєстрований користувач має доступ до функціоналу відповідно до його ролі. Незареєстрований користувач не має можливості проглядати та долучатись до груп, курсів, тощо.

Для програмного продукту було розроблено технічне завдання проєкту. Детальніше можна переглянути за посиланням: https://docs.google.com/document/d/1l9Si08wejFuiytamb-Mo4jwgRtGVYWH-

Актуальність проєкту. У зв'язку зі всесвітньої пандемією, задля забезпечення безпеки громадян, освітній процес в університетах наразі проводиться в онлайн режимі. Тому подібна тематика проєкту є надзвичайно актуальною в сучасних реаліях. Адже проведення навчання у дистанційній формі потребує зручних систем, в яких можна організувати весь навчальний процес. Також оскільки світова тенденція напрямлена на діджиталізацію дана розробка не втратить своєї значимості і під час повернення до очної освіти, адже допомагає систематизувати процеси, публікувати завдань, матеріалів проведення їх перевірки оцінювання.

ТЕХНІЧНЕ ЗАВДАННЯ

Загальна інформація

Мета і актуальність проєкту

Система підтримки навчального проєкту "Education management system" (далі – програмний продукт) розробляється як є комплексне рішення для організації навчального процесу в університеті в онлайн режимі. Актуальність проєкту в першу чергу полягає у необхідності проведення навчання у дистанційній формі, а також може бути допоміжним ресурсом як і для очної освіти, адже допомагає систематизувати освітній процес, публікувати завдання та проводити оцінювання у онлайн-режимі.

Абревіатури, скорочення. Для спрощення читання і розуміння даного документа будуть використовуватися наведені нижче абревіатури або скорочення, які можуть стосуватися як технічних моментів, так і позначення персон або термінів.

Таблиця 1. Абревіатури та скорочення

№ з/п	Термін	Абревіатури, скорочення	Коментар
1	Зареєстрований / незареєстрований користувач	користувач	-
2	Програмний продукт	Система	-
3	База даних	БД	-
4	Технічне завдання	ТЗ	-
5	CRUD-операції	CRUD	4 базові функції управління даними: "створення, зчитування, зміна і видалення"
6	Адміністратор Системи	Адміністратор	Особа або група осіб, що мають повне уявлення про функціональну та програмно-апаратну структуру Системи і контролюють її проєктування та використання

№ з/п	Термін	Абревіатури, скорочення	Коментар
5	Модератор Системи	Модератор	Користувач, який має ширші права порівняно зі звичайними внутрішніми користувачами
6	Розробник	Виконавець	Установа/заклад/організація, що забезпечує розробку, супровід та впровадження Системи
7	Facebook, Youtube, Twitter, Linkedin	Соціальні мережі	Найбільші соціальні мережі
8	Модуль	–	Функціонально закінчений фрагмент програми, призначений для використання в інших програмах
9	Проєктна платформа	платформа	Сукупність програмних модулів

Ролі в проєкті було розподілено наступним чином: Артюх – керівник проєкту, модератор; Піонтковський – бекенд; Пінковський – фронтенд; Юркевич –тестувальник; Пікуза – аналітик, фахівець з документації.

Концепція та структура

Мета створення Системи

Система підтримки інтегрованого навчального курсу створена для зручного освітнього процесу між студентами та викладачами групи.

Цілі і переваги користувача

Система повинні забезпечувати реалізацію наступних завдань:

- можливість доступу до інформації та документів, розміщених на платформах;
- можливість створювати групи, курси;
- можливість розподілити студентів на групи
- можливість створювати завдання для оцінювання навчального процесу;
- можливість спілкуватись із учасниками навчального процесу всередині системи.

Користувач / цільова група

Цільовою аудиторією є користувачі Системи. Цільова аудиторія представлена наступними групами користувачів які задіяні у навчальному процесі:

- лектори, які бажають створити групу, курс;
- студенти, які повинні навчатись у обраній викладачами системі.

Системні вимоги:

Операційна система: Linux, Windows;

Мова програмування веб-платформи: React.js, або аналоги;

СУБД: MySQL або аналоги;

Поштовий сервер: Gmail.

Ресурси

Основні конкуренти/приклади, на які потрібно орієнтуватись:

при створенні платформи з функціональними можливостями Classroom на https://classroom.google.com/;

при створенні платформи з функціональними можливостями Moodle на https://moodle.org/.

Вимоги та опис

Вимоги до стилістичного оформлення Системи. Стилістичне оформлення Системи має відповідати дизайн-макету, погодженого з керівником Замовника через керівника проєкту. Використання колірних схем, графічних елементів, шрифтів погодженого дизайн-макету в процесі створення Системи та МБ є обов'язковим.

Вимоги до графічного дизайну Системи. Система та усі їх компоненти не повинні бути графічно перенасиченими.

Інтерфейс має бути зручним у навігації, інтуїтивно зрозумілим, більшість ключових елементів мають бути доступні через дві-три прості дії.

Дизайн Системи має враховувати стилістику, кольори відповідно до лого платформи (див. рис. 1). В процесі проєктування дизайну та структури елементів заохочується аналіз найкращих практик побудови веб-ресурсів.

Повинна бути присутня мінімальна кількість графіки, більшість елементів повинні бути реалізовані засобами CSS).

Рисунок 1. Лого платформи (Education Management System)

Вимоги до верстки і програмування. Система повинна бути оптимізованою і зверстаною у відповідності зі стандартами W3C для роботи в різних стабільних версіях браузерів (без помилок). Всі JS скрипти повинні бути оптимізовані на максимальну продуктивність системи і заховані в зовнішні файли JS. Верстка сторінок Системи повинна бути виконана з урахуванням максимальної продуктивності Системи; код чистий без помилок і без зайвих тегів.

Система повинна містити інформацію про права інтелектуальної власності, яка: розташована внизу на кожній сторінці із вмістом та помітна; стверджує, що матеріали Системи є об'єктами права інтелектуальної власності, яке захищається відповідно до законодавства;

Типографія Системи повинна відповідати наступним вимогам:

- використовувати заголовки рівнів від h1 до h4 під час формування структури веб сторінок Системи, при цьому кожна веб-сторінка повинна містити лише один заголовок рівня h1. Заголовок h1 розташовується якомога ближче до відкритого тегу <body>. Для цього код сторінки формується таким чином, щоб основний контент був вище в коді сторінки;
- шрифт основного тексту − не менш як 16 пікселів;
- міжрядковий інтервал − у 1,5 рази більший, ніж розмір шрифту;
- вирівнювання основного тексту − по лівому краю;
- довжина рядка − не менше 40 та не більше 80 символів;
- абзаци повинні відокремлюватися один від одного відступами, вдвічі більшими, ніж розмір шрифту;
- не допускається одночасне використання в тексті підкреслення, курсиву та заголовних літер;
- розмір шрифту тексту, за винятком титрів, повинен змінюватися в межах до 200 відсотків без використання допоміжних технологій та втрати інформаційного наповнення або функціональності Системи;
- елементи інтерфейсу розміщуються на сторінці Системи через серію рядків та колонок з використанням модульної сітки; рекомендовано використовувати 12- колонкову адаптивну сітку;
- відступи між елементами інтерфейсу повинні бути кратними 5 пікселям. Відступ вмісту від меж екрану − не менше 15 пікселів для мобільного екрану та 30 пікселів для монітору;
- поля вводу інформації повинні бути підписаними, мати короткі, інформативні та однозначні назви, які розташовуються над полем. Поля вводу інформації повинні бути вирівняні одне під одним, по одному в рядок, ширина поля повинна відповідати вмісту, який вводиться;
- під час введення інформації користувачам необхідно надавати помітні та зрозумілі підписи та/або інструкції;
- помилки введення повинні виявлятися автоматично і, якщо відомо, як їх виправити, то користувачеві повинні надаватися підказки щодо їх виправлення.

Вимоги до вікна Системи. Система повинна забезпечувати коректне відображення даних в наступних браузерах актуальних версій: Google Chrome; Internet Explorer; Opera; Mozilla Firefox; Safari.

Вимоги до контенту і наповненню Системи. Контент системи регулюється самими користувачами, а саме лектором або адмністратором. Їм доступні текстові та графічні матеріали, а також коментарі, що стосуються їх змісту, обсягу, оформлення і розміщення.

Вимоги до компонування сторінок Системи. Компонування сторінок Системи повинно забезпечувати автоматичне масштабування сторінок в залежності від ширини робочого поля браузера користувача. Мінімальний розмір (ширина) робочого поля браузера, при якому необхідно забезпечити повноцінне відображення сторінок (без смуги горизонтальної прокрутки), становить 1170 пікселів, який у разі потреби може бути збільшений.

Вимоги в цілому. Система повинна мати архітектуру, побудовану на сучасних технологіях зберігання, обробки, аналізу даних та доступу до них; забезпечувати одночасну роботу декількох користувачів.

Структура та опис Системи. Use-case діаграми системи наведено на рис. 2 та 3.

Рисунок 2. Use-case діаграма можливостей адміністратора

Рисунок 3. Use-case діаграма можливостей лектора та студента

Мова Системи. Основна мова Системи − англійська.

Ролі в Системі. Система має 3 основні ролі:

- Адміністратор;
- Лектор;
- Студент.

Адміністратор має можливість:

- редагувати текстовий контент (там, де це передбачено);
- додавати та редагувати контент;
- переглядати і редагувати дані користувачів;
- блокувати , видаляти користувачів;
- формувати групи, курси та команди.

Лектор має можливість:

- створювати групу, курс, підгрупу.
- додавати , видаляти користувачів (інших лекторів та студентів);
- відправляти особисті повідомлення учасникам навчального процесу;
- створювати завдання;
- виставляти оцінку за виконані завдання;
- додавати інформацію необхідну для навчального процесу;
- редагувати особисту інформацію.

Студент має можливість:

- відправляти особисті повідомлення учасникам навчального процесу;
- виконувати завдання;
- додавати інформацію необхідну для навчального процесу;
- залишати коментарі;
- редагувати особисту інформацію.

Зареєстрований користувач має доступ до функціоналу відповідно до його ролі. Незареєстрований користувач не має можливості проглядати та долучатись до груп, курсів, тощо.

Реєстрація. Користувач може зареєструватися в Системі.

Сторінка реєстрації містить:

- текст заголовок: "Реєстрація";
- вибір ролі для реєстрації (лектор або студент)
- текст і поля для реєстрації:
- прізвище*;
- ім’я*;
- пароль*;
- електронна пошта.
- кнопка "Register";
- кнопка переходу до авторизації "Login".

Після натискання на кнопку "Register" користувачеві відображається сторінка з написом "Посилання для активації відправлено на ваш e-mail: 'name@mail.com', будь ласка, перевірте свою електронну пошту та перейдіть за посиланням, для того, щоб активувати свій обліковий запис.

На e-mail користувача надходить лист з info@site.com, з текстом *"Ласкаво просимо в …! Для того, щоб активізувати свій обліковий запис перейдіть за посиланням …. Будь ласка, не пересилайте та не відповідайте на це повідомлення.*

З повагою, команда … ".

Перейшовши за цим посиланням, користувач потрапляє на головну сторінку Системи вже авторизованим.

* **поля обов'язкові для заповнення.**

** **на всіх полях повинна бути відповідна валідація.**

Авторизація. Користувач може авторизуватися в Системі

Сторінка авторизації містить:

- текст заголовок: "Login";
- кнопка переход до реєстрації "Register";
- інпути:
- пошта (username)*;
- пароль*.

Форма авторизації містить активне посилання: - кнопка "Вхід".

* **на всіх полях повинна бути відповідна валідація.**

Відновлення пароля. Користувач може відновити пароль до свого облікового запису в Системі. Сторінка відновлення пароля містить:

- текст заголовок: "Відновити пароль";
- текст: "Введіть адресу Вашої пошти";
- інпути:
- пошта (адреса електронної пошти)*;
- кнопка "Відновити".

на полі "Пошта" повинна бути відповідна валідація через "@"

Після натискання на кнопку "Відновити" відкривається сторінка "Відновлення пароля" з вмістом:

"Ми відправили вам лист з посиланням для зміни пароля. Будь ласка, перевірте свою електронну пошту і натисніть на посилання, щоб продовжити. "

Користувачеві приходить лист з наступним вмістом:

"Вітаємо, Name!

Ви отримали цей лист, тому що Ви запросили відновлення пароля на … . Щоб скинути пароль, перейдіть за наступним посиланням або скопіюйте і вставте його в веб-браузер: посилання

Якщо ви не хочете змінювати свій пароль, видаліть цей лист. "

Перейшовши за цим посиланням, користувач потрапляє на сторінку "Відновлення пароля", яка містить:

- текст "Введіть новий пароль нижче";
- інпути:
- новий пароль;
- пароль ще раз.
- кнопка "Підтвердити".

Після натискання кнопки "Підтвердити" завантажується головна сторінка, користувач автоматично авторизований.

* **на всіх полях повинна бути відповідна валідація.**

Огляд системи для Лектора (викладача). Система має основне функціональне бокове меню з активним підписом EMS, при натисканні якого можна проглянути основні можливості програми , а саме такі пункти:

- іконка користувача, перейшовши по якій можна перейти до модулю Особистий профіль;

- завдання на перевірку;
- особисті повідомлення;
- список доступних груп;
- список курсів;
- список команд (підгруп).

Модуль системи, сторінка "Особистий профіль". Основна сторінка яка відкривається після реєстрації це сторінка Особистого профілю. Зліва доступне функціональне меню. На Основній частині сторінки є такі доступні поля:

- фотографія;
- ім'я;
- по-батькові ;
- пароль;
- електронна адреса.

Кнопки: "Редагувати", при натисканні якої, відкривається вікно з вказаними вище полями та відповідно кнопка "Зберегти"; "Створити" з можливими варіантами вибору (група, курс, команда), при натисканні відкривається вікно з полями Назва (відповідно групи, курсу або команди) та з кнопкою "Підтвердити", натиснувши яку користувач перенаправляється на відповідну сторінку створеного об'єкта; "Долучитись", при натисканні якої можна ввести відомий користувачу код групи та долучитись до неї.

Нижній футер (пропонується Виконавцем, погоджується з керівником проєкту).

Модуль системи, сторінка "Особисті повідомлення". Зліва доступне функціональне меню

В основній частині сторінки наявне поле пошуку , в якому можна знайти викладача / студента з яким він знаходиться в одній групі/на одному курсі / в одній команді

В особистому діалозі з учасником навчального процесу, кожен з учасників може відправляти текстові повідомлення та прикріпити необхідні файли (документи, фотографії, презентації, тощо).

Модуль системи, сторінка "Групи". Оскільки лектор може викладати не лише в одній групі, при натисканні на вкладку групи з'являється випадаючий список доступних груп.

Хедер. На сторінці хедера є вкладки:

- матеріали групи, де викладачі публікуватимуть питання що стосуються загального навчального процесу, студенти мають можливість коментувати, для уточнення питань. Публікації мають вигляд опублікованих дописів, є можливість додавати текст, вкладати фотографії або необхідні файли. Дописи можна коментувати, та автору дописа або адміністратору можна його видаляти;
- люди, у вигляді списку всіх викладачів групи та її студентів. Викладачі можуть видаляти/додавати студентів групи;
- налаштування (у налаштуваннях можна знайти посилання на групу або код за яких можна долучитись, редагувати дані групи).

Модуль системи, сторінка "Курси". Кожен з викладачів групи має можливість створити курс, до якого автоматично будуть додані всі студенти групи

Хедер. На сторінці хедера є вкладки:

- матеріали курсу;

- завдання, де викладач може переглянути список завдань конкретного курсу і додати нові натиснувши кнопку "створити завдання" і відповідно перейшовши до вікна "Створення завдання";
- люди, у вигляді списку всіх викладачів групи та її студентів. Викладачі можуть видаляти/додавати студентів або інших викладачів курсу.

Модуль системи, сторінка "Створення Завдання". Спочатку необхідно обрати тип завдання (Завдання, Тест , Розподіл по групах)

На сторінці є текстове поле, в якому необхідно описати задачу, а також можливість вкласти додаткові файли.

Також необхідно вказати значення у полях

для кого (випадаючий список з наявними групами командами і курсами , де викладає лектор);

оцінка (виставляється максимально можлива кількість набраних балів за дане завдання);

дедлайн завдання.

І відповідно кнопка "Опублікувати завдання".

Модуль системи, сторінка "Команди". Викладач також має можливість розбити свою групу на підгрупи (команди).

Для цього йому необхідно визначити кількість команд на яку необхідно розділити групу та обрати одну з опцій одну із опцій:

1. Розбити випадковим чином.
2. Опублікувати завдання для опитування студентів, де вони обиратимуть критерії задані викладачем, за якими система після завершення тестування сформує команди.
3. Розбити на групи за власним бажанням (у попередніх опціях є можливість коригувати сформовані групи).

Сторінка підгрупи матиме такий ж вигляд як і сторінка курсу.

Модуль системи, сторінка "Завдання на перевірку". На сторінці доступний список всіх завдань опублікованих конкретним викладачем і також кнопка "Створити завдання" натиснувши яку викладач буде перенаправлений на відповідну сторінку.

Огляд системи для Студента. Студент може долучитись до групи за посиланням або кодом наданим викладачем

Система має основне функціональне бокове меню з активним підписом EMS, при натисканні якого можна переглянути такі пункти:

- іконка користувача, перейшовши по якій можна перейти до модулю Особистий профіль;
- призначені завдання;
- особисті повідомлення;
- список доступних груп;
- список курсів;
- список команд (підгруп).

Модуль системи, сторінка "Особистий профіль". Основна сторінка яка відкривається після реєстрації це сторінка Особистого профілю Зліва доступне функціональне меню

На Основній частині сторінки є такі доступні поля:

- фотографія;
- ім'я;
- по-батькові;

- пароль;
- електронна адреса.

Кнопки: "Редагувати", при натисканні якої, відкривається вікно з вказаними вище полями та відповідно кнопка "Зберегти".

"Долучитись", при натисканні якої можна ввести відомий користувачу код групи та долучитись до неї.

Модуль системи, сторінка "Особисті Повідомлення". Для студента Особисті повідомлення мають такий ж вигляд як і для викладача.

Модуль системи, сторінка "Групи". Оскільки лектор може викладати не лише в одній групі, при натисканні на вкладку групи з'являється випадаючий список доступних груп.

Хедер. На сторінці хедера є вкладки:

- матеріали групи, де викладачі публікуватимуть питання що стосуються загального навчального процесу, студенти мають можливість коментувати, для уточнення питань. Студенти мають можливість коментувати їх;
- люди, у вигляді списку всіх викладачів групи та її студентів. Викладачі можуть видаляти/додавати студентів групи;
- налаштування (у налаштуваннях можна знайти посилання на групу або код за яких можна долучитись, редагувати дані групи).

Модуль системи, сторінка "Курси". Студент групи буде автоматично доданий до курсу

Хедер. На сторінці хедера є вкладки:

- матеріали курсу;
- завдання, де студент може переглянути завдання призначенні йому з конкретного курсу та здати конкретну задачу натиснувши на неї, відповідно перейшовш на сторінку самого завдання;
- люди, у вигляді списку всіх та її студентів курсу та викладачів.

Модуль системи, сторінка "Команди". Хедер. На сторінці хедера є вкладки:

- завдання для команди;
- люди, у вигляді списку своїх товаришів по команді групи та викладачів;
- чат, де студенти команди можуть спілкуватися для вирішення спільних питань.

Модуль системи, сторінка "Призначені завдання". На сторінці можна переглянути список завдань із усіх груп та курсів, та переглянути оцінки за виконані курси. Завдання відсортовані по вкладкам Виконані і Невиконані. І всі завдання розташовані відповідно по наближенню делайнів. Чим скоріше час здачі , тим вище розташоване завдання

Модуль системи, сторінка здачі завдання. Студент може натиснути на конкретне завдання і на сторінці будуть наявні такі елемент:

- текст завдання;
- прикріпленні файли;
- форма для прикріплення документів для здачі завдання;
- кнопка "Здати завдання".

Ризики. Необхідно спроєктувати БД і оптимізувати код, щоб максимально зменшити кількість запитів до БД.

Тести. Для тестування буде використано junit и spring integration tests.

МЕТОДОЛОГІЯ ТА СИСТЕМА УПРАВЛІННЯ ПРОЄКТАМИ

Розподіл ролей в команді. Для успішного виконання проєкту учасникам групи необхідно було розділити між собою обов'язки за такими ролями: керівник проєкту, модератор, аналітик, бекенд-розробник, фронтенд-розробник, фахівець з документації та тестувальник. Після проведення онлайн-зустрічей ролі були розподілені між студентами:

- Артюх Владислав – керівник проєкту, модератор;
- Піонтковський Ігор – бекенд;
- Пінковський Дмитро – фронтенд;
- Юркевич Богдан – тестувальник, спеціаліст з специфікації;
- Пікуза Валентина – аналітик, фахівець з документації.

Методологія та система управління проєктами. В якості методологію управління проєктами було обрано **Scrum**.

Щотижня проводились зустрічі, на яких учасники команди обговорювали такі питання:

- прогрес, виконання поставлених задач;
- які є проблеми на даному етапі та варіанти їх вирішень?
- план задач на наступний тиждень.

Для оперативного вирішення завдань та можливості швидко зв'язатись з учасниками команди був створений груповий чат у месенджері Telegram.

За систему управління було прийнято рішення обрати Jira, де відповідно всім учасникам команди призначались завдання та відстежувався процес їх виконання (рис. 4)

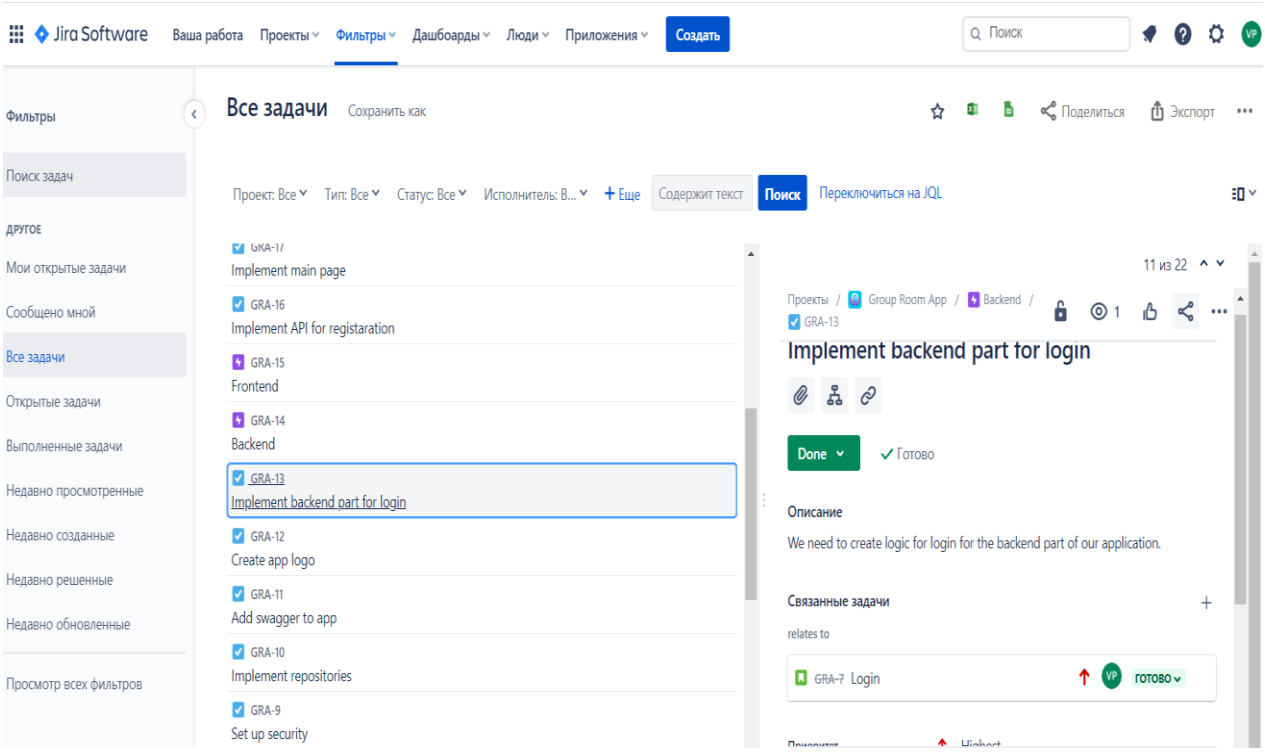


Рисунок 4. Задачі в Jira

ФОРМАЛЬНА СПЕЦИФІКАЦІЯ

Предметом формальної специфікації було обрано метод, який використовується під час розподілення учнів по групам, а саме такий математичний метод як знаходження найкоротшого шляху між вершинами в графі.

Для реалізації методу був вибраний алгоритм пошуку на графах: пошук в ширину (BFS).

Специфікацію реалізовано мовою Dafny.

Алгоритм методу BFS. Якщо задано граф $G = (V, E)$ та початкову вершину s, алгоритм пошуку в ширину систематично обходить всі досяжні із s вершини. На першому кроці вершина s позначається, як пройдена, а в список додаються всі вершини, досяжні з s без відвідування проміжних вершин. На кожному наступному кроці всі поточні вершини списку відмічаються, як пройдені, а новий список формується із вершин, котрі є ще не пройденими сусідами поточних вершин списку. Для реалізації списку вершин найчастіше використовується черга та принцип зміщення. Виконання алгоритму продовжується до досягнення шуканої вершини або до того часу, коли на певному кроці в список не включається жодна вершина. Другий випадок означає, що всі вершини, доступні з початкової, уже відмічені, як пройдені, а шлях до цільової вершини не знайдений.

Код методу BFS:

```
method BFS(G : Graph, s : int) returns (d : array<int>)
  requires 0 <= s < |G.adjList|
  requires forall u :: 0 <= u < |G.adjList| ==>
            forall v :: 0 <= v < |G.adjList[u]| ==> 0 <= G.adjList[u][v] < |G.adjList|
  requires forall u :: 0 <= u < |G.adjList| ==>
            forall v,w :: 0 <= v < w < |G.adjList[u]| ==> G.adjList[u][v] != G.adjList[u][w]
{   var i := 0;
  var j := 0;
  var u : int;
  var Q : seq<int>;
  var iterations := G.adjList[0];
  var n := |G.adjList|;
  var color : array<bool>;
  color := new bool[n];
  d          := new int [n];
  i := 0; while (i < n)
  {
            color[i] := true;
            d[i] := -1;
            i := i + 1;
  } Q := [s];
  while (Q != [])
  decreases TrueIndices(color), |Q|
  invariant forall x | x in Q :: 0 <= x < |G.adjList|  // invariant (1)
  {
            ghost var top_of_loop_indices := TrueIndices(color);
```

```
        ghost var top_of_loop_Q := Q;
        // u <- Dequeue(Q)
        u := Q[0]; Q := Q[1..];
        assert u in top_of_loop_Q;  // trigger invariant (1) for u
        // help Dafny see that dequeueing is ok
    assert forall x | x in Q :: x in top_of_loop_Q;
        // foreach v in adjList[u]
        i := 0; while i < |G.adjList[u]|
        invariant forall x | x in Q :: 0 <= x < |G.adjList|  // invariant (2)
        invariant  // invariant (3)
        || TrueIndices(color) < top_of_loop_indices
        || (TrueIndices(color) == top_of_loop_indices && |Q| < |top_of_loop_Q|)
        {
        var v := G.adjList[u][i];
        if (color[v])
        {
        // help Dafny see that v was newly colored false
        assert v in TrueIndices(color);
        color[v] := false;
        d[v]       := d[u] + 1;
        Q          := Q + [v];
        }
        i := i + 1; }}}
```

ПРОГРАМНА РЕАЛІЗАЦІЯ

Frontend частина. Для розробки Frontend частини було прийнято використовувати бібліотека для створення інтерфейсів користувача React.js, фреймворк Material-UI, Sass, та середовищем розробки було обрано VS Code.

На рис. 5 та рис.6 зображено зовнішні вигляд блоків Реєстрації та Входу в систему.

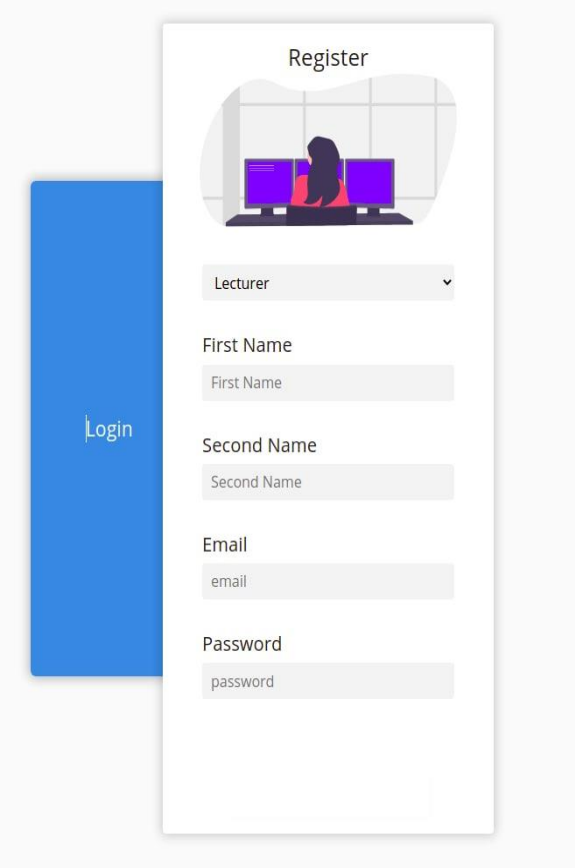


Рисунок 5. Зовнішній вигляд реєстрації

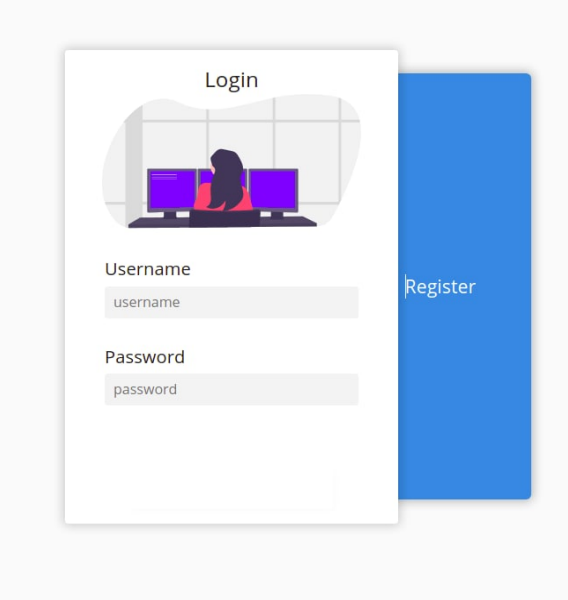


Рисунок 6. Зовнішній вигляд входу в системи

На рис. 7 зображена основна сторінка зі списком курсів та груп.

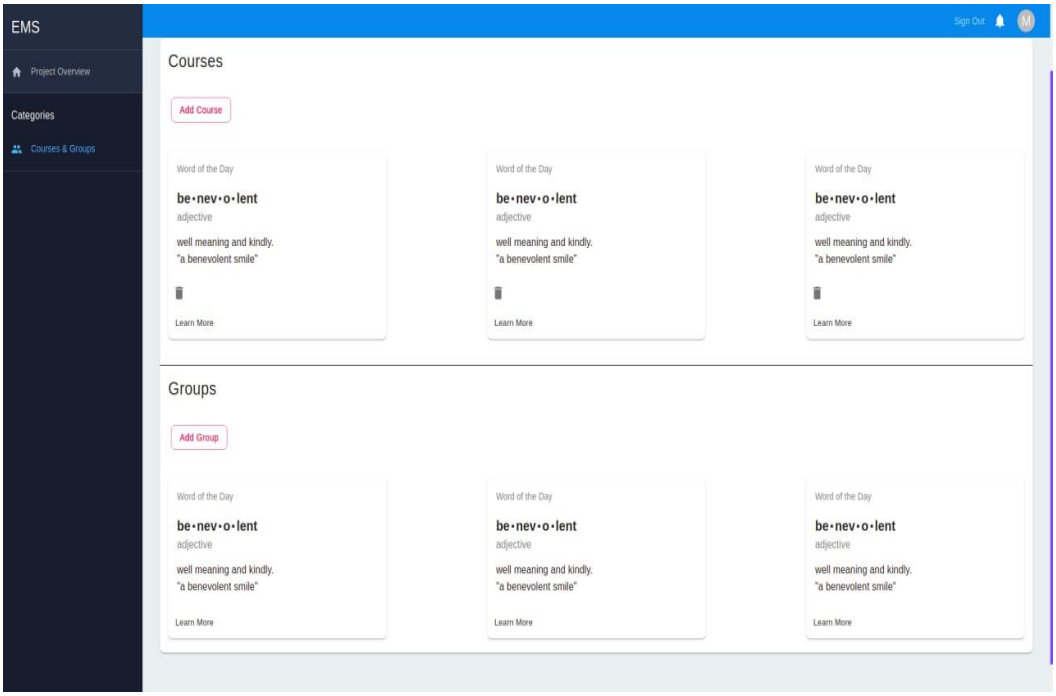


Рисунок **7.** Основна сторінка зі списком груп та курсів

На рис 8. зображена сторінка при переході на конкретну груп, на якій відповідно є вкладки Post, де публікуються загальна інформація для групи і Task (рис. 9) із списком завдань.

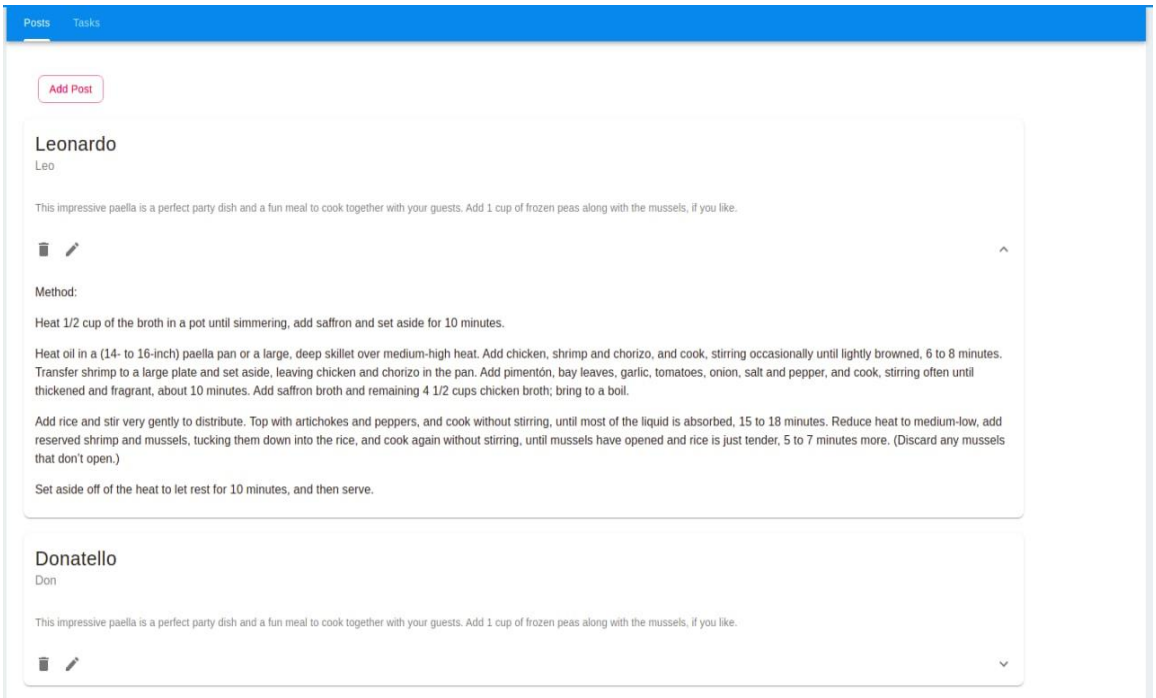


Рисунок 8. Сторінка групи

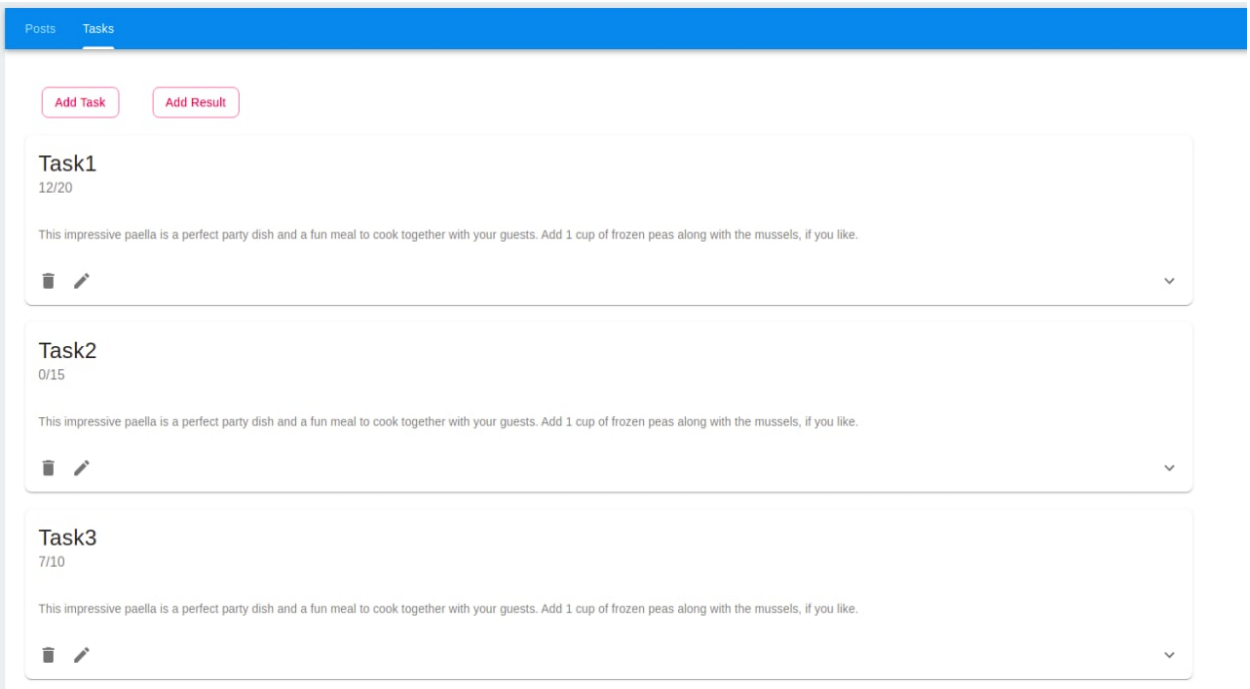


Рисунок 9. Сторінка зі списком завдань

При натисканні на кнопку створення завдання відкривається відповідна форма (рис.10), де вказується назва опис завдання та оцінка за його виконання

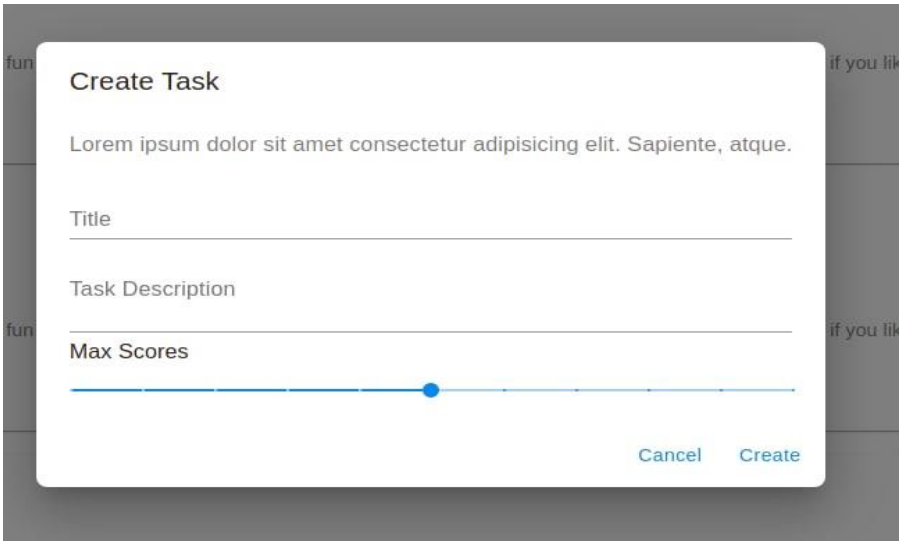


Рисунок 10. Сторінка створення завдання

На рис 11. зображено,як виглядає особистий чат учасника навчального процесу.

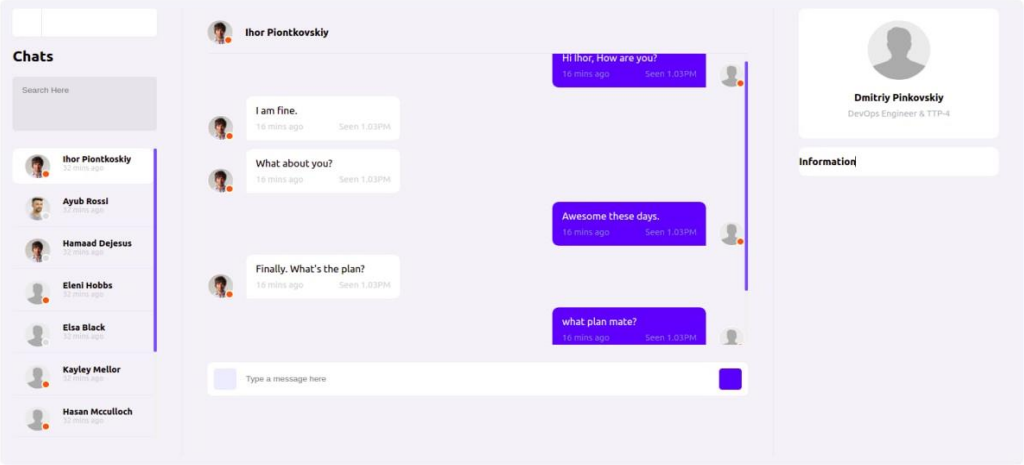


Рисунок 11. Особистий чат учасника навчального процесу

Backend частина

Проєктування, модельний шар, база даних

Проаналізувавши основні вимоги до програмного продукту було обрано реляційну систему керування базами даних MySQL (ClearSQL для віддаленого середовища) – оскільки у застосунку потрібно зберігати велику кількість структурованих даних.

Оскільки основною мовою програмування для backend-у було обрано Java 11.0.9 то ми прийшли до думки що буде правильно обрати Spring Framework для розробки RESTful сервісу. Таким чином у застосунку буде використовуватись Spring Framework та його компоненти(а саме Spring JPA, Spring Security, Spring Validation, Spring Web) версії 2.4.3.

Оскільки backend частина застосунку буде представляти собою RESTful сервіс то ми прийшли до логічного висновку що правильно буде використати деяку систему документування запитів сервісу, а саме у цьому застосунку було викростано springfox-swagger2 версії 2.6.1.

Допоміжні ресурси які були використані Lombok, jdbc(mysql-connector-java), bcrypt.

Для побудови застосунку було обрано Gradle Build Tool.

Система контролю версій - GitHub https://github.com/ihPiontkovskyi/ems-service.

Система розгортування та безперервної інтеграції - Heroku https://service-ems.herokuapp.com/.

Після того як було обрані основні компоненти застосунку ми почали проєктувати модельний шар застосунку, а саме було спроєктовано ER діаграму для бази даних (рис. 12), по ній побудовано Java класи (рис. 13, 14), а потім за допомогою JPA було згенеровано схему бази даних як для локального так і для віддаленого середовища.

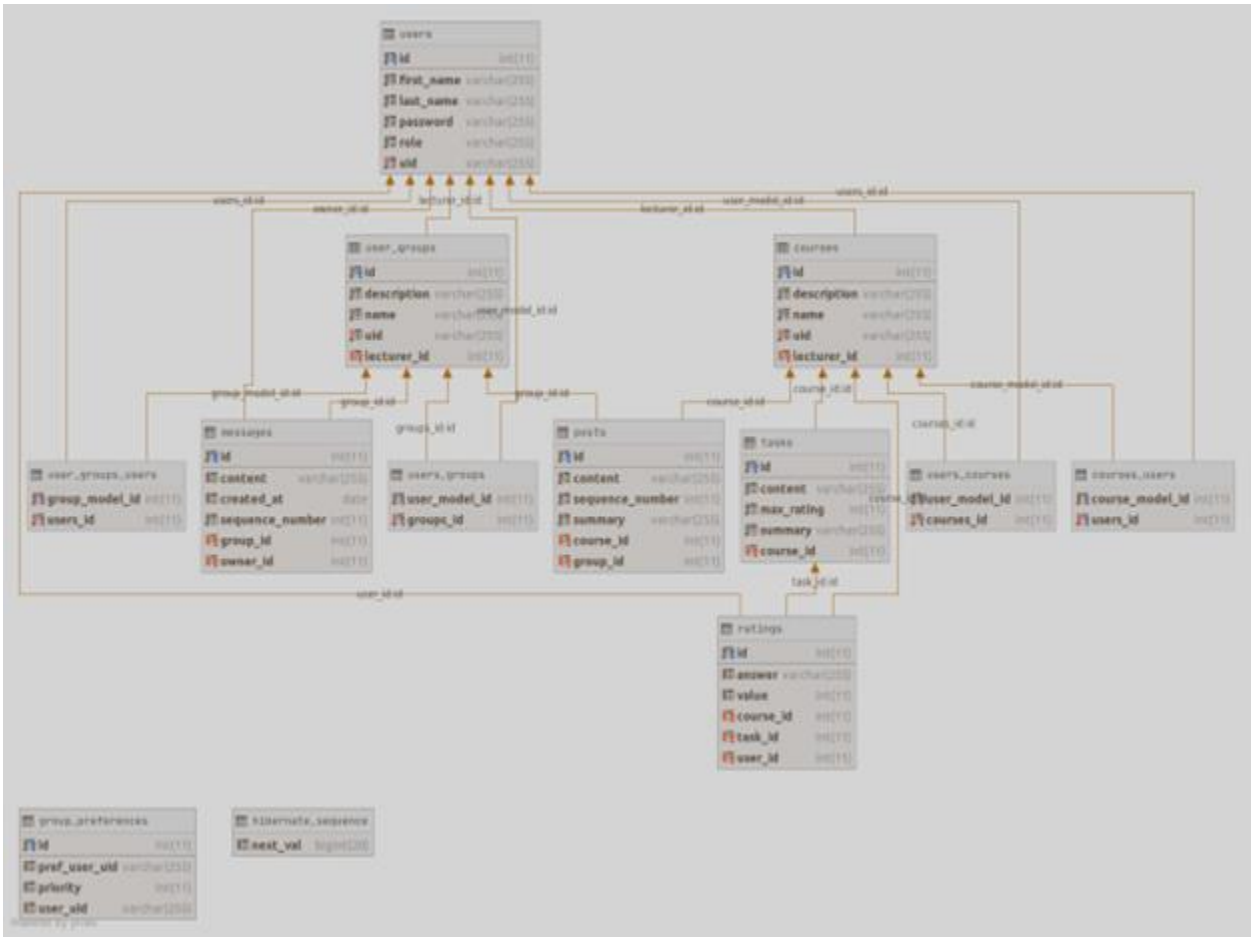


Рисунок 12. ER діаграма бази даних

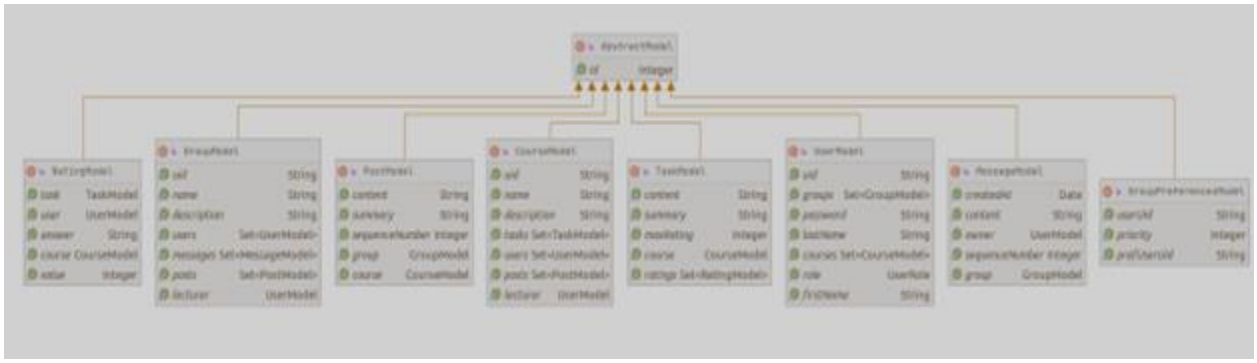
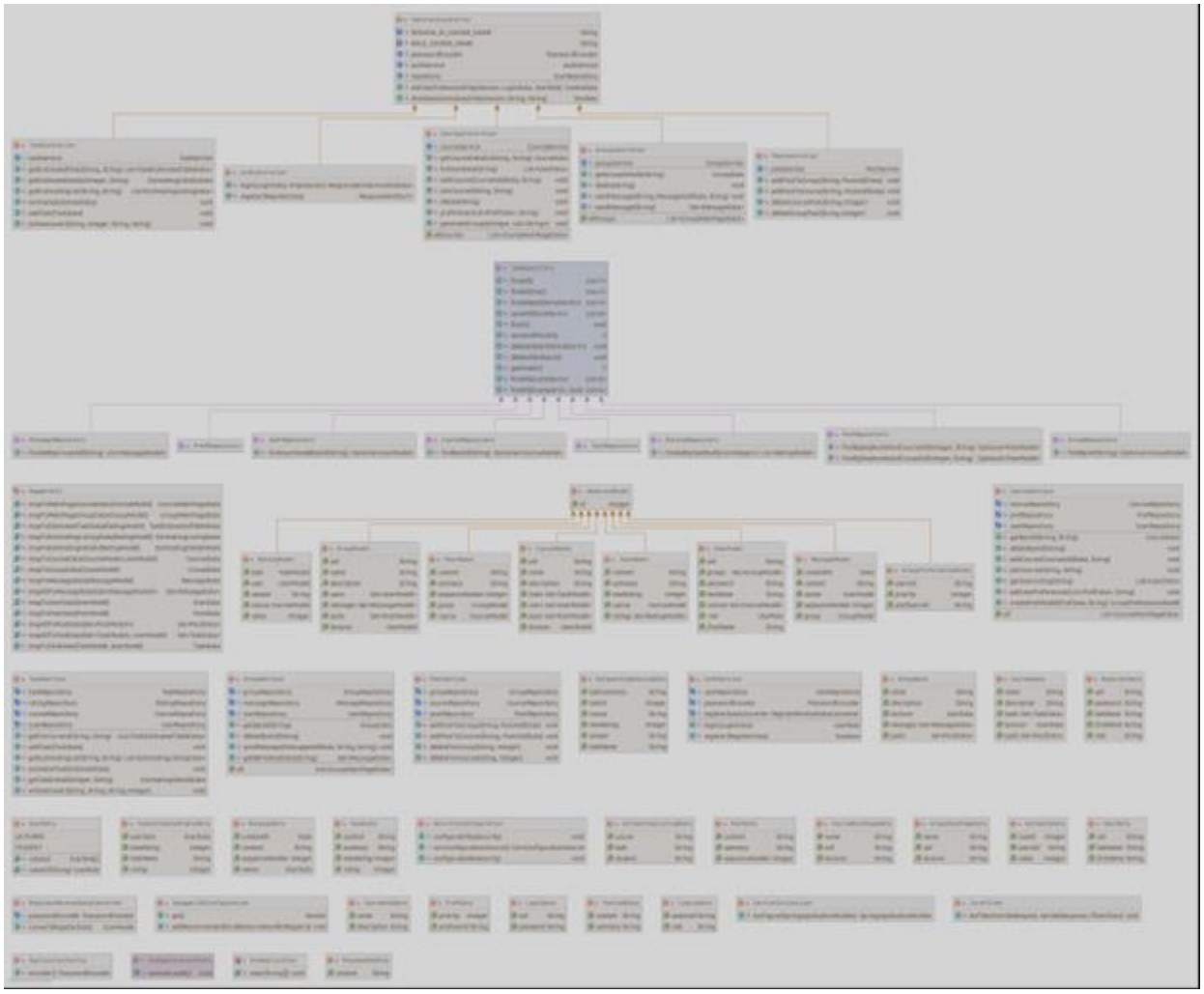


Рисунок 13. Діаграма класів модельного шару



Додаток 14. Загальна діаграма класів

Розробка ендпоінтів та сервісного шару. Розробивши основні компоненти ми приступили до розробки перших ендпоінтів, ними стали авторизація та реєстрація (див. рис 15).

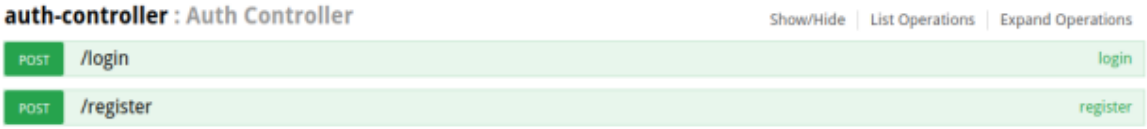


Рисунок 15**.** Ендпоінт авторизації та реєстрації

Під час розробки цих ендпоінтів стало питання безпеки, тому було обрано варіант автентифікації наступним чином:

користувач надсилає свої дані для входу(в мережі дані пересилаються в зашифрованому вигляді), було обрано base64 формат шифрування даних які пересилаються мережею;

якщо дані валідні то в відповіді сервіс надсилає токен, та роль юзера (див. рис 16), які зберігаються в куках (час життя – 30 хв).

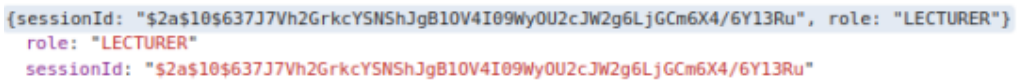


Рисунок 16. Приклад відповіді при успішній авторизації

Для цієї логіки було розроблено деякі класи з DTO шару, а також деякі методи сервісного шару, додано ендопоінти до контроллера а також було налаштовано роботу CORS фільтрів для коректної взаємодії з рест сервісом. Було додано конфігурацію для Swagger2 та конфігурацію для Spring Security. Після розробки логіки авторизації та реєстрації було розроблено ендпоінт для взаємодії з курсами (рис. 17):

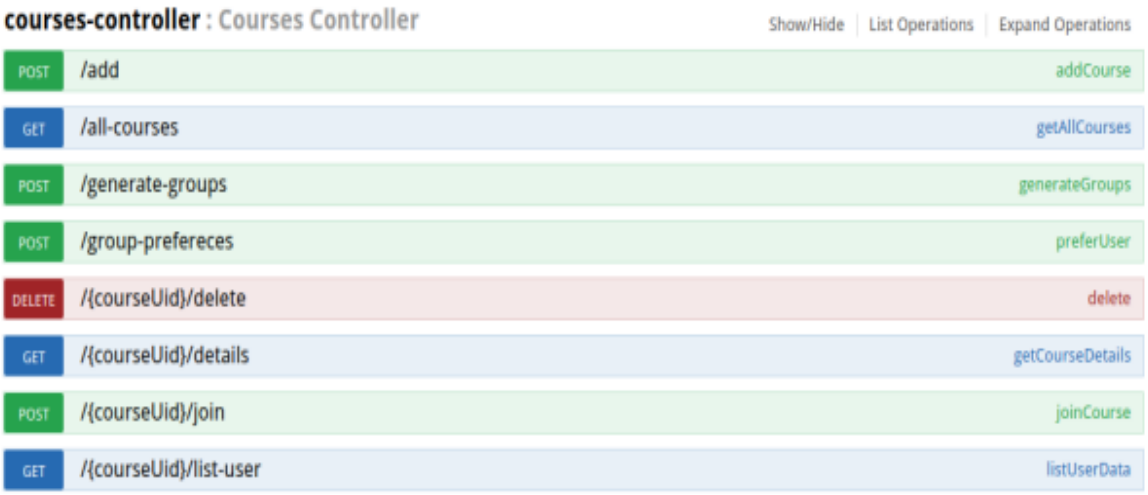


Рисунок 17. Ендпоинт для взаємодії з курсами

POST: /webapi/courses/add - ендпойнт, за допомогою якого можна створити курс, доступний тільки для ролей LECTURER та ADMIN, отримує JSON вигляду:

```
{
    "description": "string",
    "name": "string"
}
```

GET: /webapi/courses/all-courses - повертає всю інформацію про доступні користувачеві курси, приклад респонсу:

```
[
{
"lecturer": "string",
"name": "string",
"uid": "string"
 }
 ]
```

POST: /webapi/courses/generate-groups - на основі наведеної кількості груп та юзерів генерує групи. (RequestParam Integer numOfGroups, RequestParam String[] uids)

POST: /webapi/courses/group-preferences - доступний тільки для ролі STUDENT, вказуючи юзера та числове значення пріоритету формуються вподобання юзерів бути в групах з тими чи іншими людьми.

DELETE: /webapi/courses/{courseUid}/delete - доступний тільки юзерам з роллю ADMIN, LECTURER; видаляє курс.

GET: /webapi/courses/{courseUid}/details - повертає всю інформацію щодо конкретного курсу:


```
{
  "description": "string",
  "lecturer": {
    "firstName": "string",
    "lastName": "string",
    "uid": "string"
  },
  "name": "string",
  "posts": [
    {
      "content": "string",
      "sequenceNumber": 0,
      "summary": "string"
    }
  ],
  "tasks": [
    {
      "content": "string",
      "maxRating": 0,
      "rating": 0,
      "summary": "string"
    }
  ]
}
```


POST: /webapi/courses/{courseUid}/join - за допомогою uid можна приєднатися до курсу

GET: /webapi/courses/{courseUid}/list-user - відображає список юзерів учасників курсу

Для розробки усих ендпоінтів було розширено сервісний шар, DTO , шар контролерів, а також розроблені утиліти для конвертування об’єктів з моделей в DTO.

А також було розроблено інші енд поніти для взаємодією з групами, завданнями, постами, повідомленнями (див. рис 18). Їх призначення інтуїтивно зрозуміле, їх конфігурацію можна переглянути за посиланням: https://service-ems.herokuapp.com/swagger-ui.html.

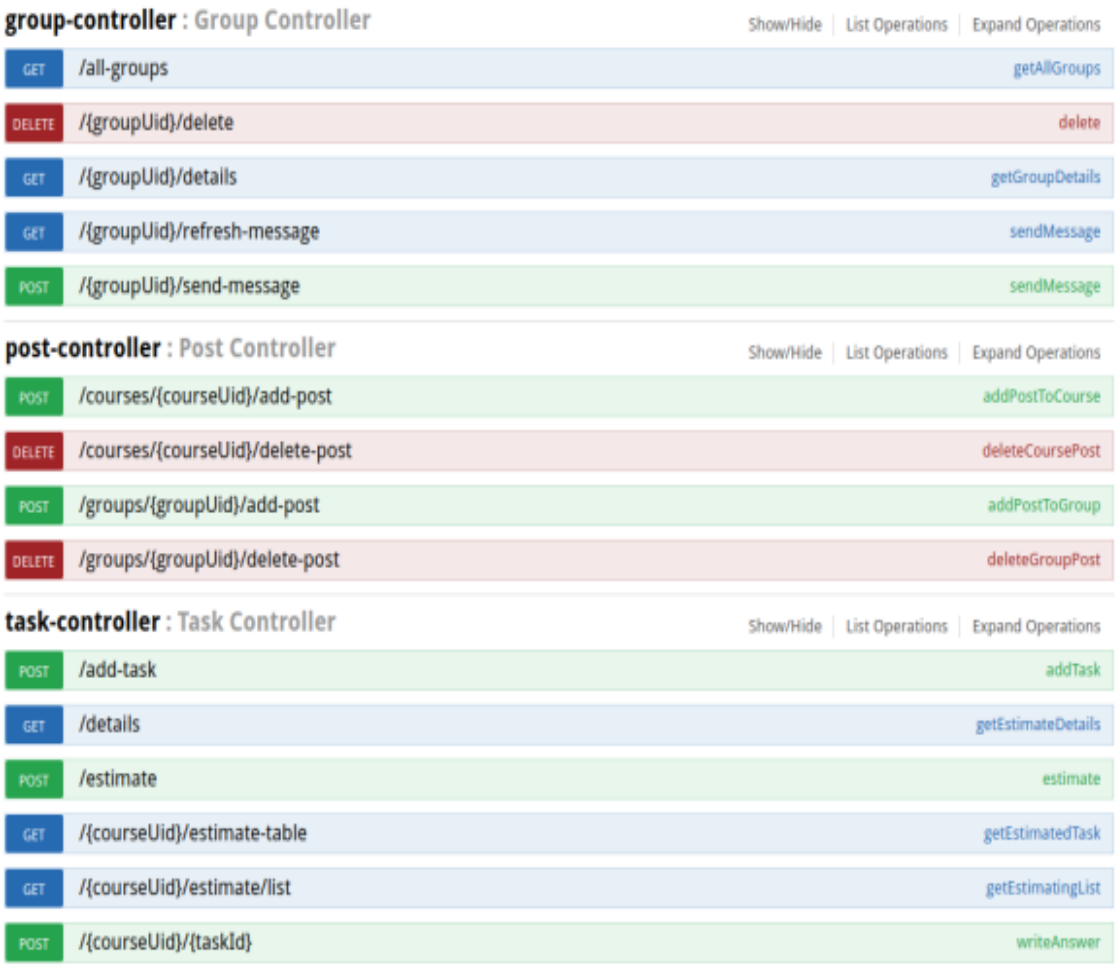


Рисунок 18. Ендпоінти

Скористатися Swagger-ом для зміни даних у БД не є можливим оскільки уся логіка роботи з ендпоінтами привязана до куків та сессій.

Тестування та верифікація. Тестування проводилося за технологією "чорний ящик". Вона була обрана через такі причини:

1) можливість незалежної розробки та тестування системи без знання її конкретної реалізації;

2) вона дозволяє тестувальникам працювати окремо від розробників майже без ніякої синхронізації та взаємодії;

3) простота використання, можливість швидкого пристосування до специфікації.

Також використовувалася технологія тестування BDD. Тестування поведінковою розробкою (BDD) - це розділ тестової розробки (TDD). BDD використовує зручні для читання описи вимог користувачів програмного забезпечення як основу для тестування програмного забезпечення. Подібно доменному дизайну (DDD), раннім кроком у BDD є визначення спільного словника між зацікавленими сторонами, експертами доменів та інженерами. Цей процес включає визначення сутностей, подій та результатів, про які турбуються користувачі, та надання їм імен, щодо яких усі можуть погодитись.

Для тестування використовувалися технології: JUnit, Mockito, H2, Serenity, Spring Boot Test. Було проведено мануальне тестування аутентифікації та авторизації зі сторони Front-

end. В основному тести проводилися мануально, тобто за оцінкою правильності тестувальниками. Також було протестовано Back-end частину системи інтеграційними тестами (див. рис 19).

Тести були перевірені, проаналізовані та верифіковані учасниками команди та головним тестувальником.

Було проведено ряд регресійних функціональних тестувань демо версій проєкту, зокрема фінальної. Під час таких тестувань було знайдено та виправлено ряд помилок, наприклад, Bug in authorization.

```
@Before
public void setUp() {
    user.setPassword("pass");
    user.setUid("uid");
    user.setRole(UserRole.LECTURER);
    user.setLastName("lastName");
    user.setFirstName("firstName");

    group.setDescription("Description");
    group.setLecturer(new UserModel());
    group.setName("group");
    group.setUid("uidX");
    group.setLecturer(user);

    message.setGroup(group);
    message.setOwner(user);
    message.setContent("content");
    message.setSequenceNumber(1);


    course.setDescription("Description");
    course.setName("course");
    course.setUid("uidY");
    course.setLecturer(user);

    coursePost.setCourse(course);
    coursePost.setSummary("Summary_post1");
    coursePost.setContent("Content_post1");
    coursePost.setSequenceNumber(1);

    groupPost.setGroup(group);
    groupPost.setSummary("Summary_post1");
    groupPost.setContent("Content_post1");
    groupPost.setSequenceNumber(1);

    task.setMaxRating(20);
    task.setSummary("summary");
    task.setContent("content");
    task.setCourse(course);

    rating.setCourse(course);
    rating.setTask(task);
    rating.setUser(user);
    rating.setAnswer("asnwer-link");
    rating.setValue(18);
}

@Test
public void testNativeQueryInMessageRepository() {
    userRepository.save(user);
    groupRepository.save(group);
    messageRepository.save(message);

    List<MessageModel> messages = messageRepository.findAllByGroupUId("uidX");

    assertThat(messages, hasSize(1));
    assertThat(messages.get(0).getContent(), equalTo("content"));
    assertThat(messages.get(0).getSequenceNumber(), equalTo(1));
}

@Test
public void testNativeQueryInPostRepositoryGroupCase() {
    userRepository.save(user);
    groupRepository.save(group);
    postRepository.save(groupPost);

    Optional<PostModel> post = postRepository.findBySeqNumbAndGroupUid(1, "uidX");

    assertTrue(post.isPresent());
    assertThat(post.get().getContent(), equalTo("Content_post1"));
    assertThat(post.get().getSummary(), equalTo("Summary_post1"));
}

@Test
public void testNativeQueryInPostRepositoryCourseCase() {
    userRepository.save(user);
    courseRepository.save(course);
    postRepository.save(coursePost);
```

Рисунок 19. Інтеграційне тестування SQL запитів

ВИСНОВКИ

В результаті роботи над груповим проєктом був розроблений повноцінний програмний продукт "Education management system" - система для організації освітнього процесу в університеті в онлайн режимі. Створена командою розробка відповідає наданим технічним вимогам та реалізує основні функціональності необхідні для навчання: опублікування матеріалів, призначення завдань та їх оцінювання. До особливостей та переваг розробленого програмного продукту слів віднести можливість розбиття навчальних груп на команди для групової роботи, а також наявність особистих повідомлень, де студенти та викладачі можуть листуватись для вирішення поточних питань.

У процесі роботи в команді учасники досягли бажаних результатів. Для частини студентів робота в команді над розробкою проєкту була вперше, тому це був хороший досвід, адже вони змогли дізнатися більше про повний процес розробки: від спілкування з замовником, проєктування, розробки і до презентації. Студенти набули досвіду в оформленні формальної специфікації та документації. Частина студентів, яка уже мала досвід роботи в проєктах, змогла поглибити свої знання в даній області, набула нового досвіду в розробці проєктів, використання BDD, покращила навички у застосування технологій верифікацій та тестування.

РОЗРОБКА ПЛАТФОРМИ ПІДТРИМКИ ВИКЛАДАННЯ ДИСЦИПЛІНИ "ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Дмитро Безух, Олександр Гутаревич, Софія Соняк, Надія Ярошевська

ВСТУП

Початок пандемії коронавірусної інфекції став "кризою століття" для різних галузей. Не стала й виключенням освіта. В сучасних умовах складність опанування матеріалу студентами вищих навчальних закладів була значно збільшеною. Окрім того, викладачі теж зіткнулися із певними проблемами при викладенні матеріалу. Звичні аудиторні заняття стали неможливими. Це призвело до ускладнення взаємодії між викладачем та студентами.

Розробка "Освітньо-тестової платформи з курсу ООП" забезпечить створення повного циклу роботи викладача із студентами: від розміщення лекційного матеріалу до проходження модульного тестування.

Система освітньо-тестової платформи з курсу ООП розглядається, як функціонал для подання матеріалу, проходження тематичних тестів, модулів, перевірки роботи студента та відстеження успішності студента.

ОРГАНІЗАЦІЯ РОБОТИ В КОМАНДІ

Учасники команди та розподіл ролей.

До команди увійшло 4 учасники, студенти 4 курсу: Безух Дмитро, Гутаревич Олександр, Соняк Софія, Ярошевська Надія. Розподілей ролей (див. табл.1) відбувався за побажаннями студентів і був збережений протягом усієї роботи над проєктом.

Таблиця 1 – розподіл ролей у групі

Учасник	Роль учасника
Безух Дмитро	Бекенд, тестувальник.
Гутаревич Олександр	Бекенд, тестувальник.
Соняк Софія	Керівник проєкту, модератор, фахівець з документації, аналітик.
Ярошевська Надія	Фронтенд.

Методологія та система управління проєктами. Для управління проєктом було обрано Scrum. Зустрічі проводилися на початку реалізації проєкту кожного дня (іноді двічі на день). По мірі виконання завдань, кількість зустрічей була зменшена. Проводився запис беклогу (рис. 1) за допомогою таблиці в Excel (https://docs.google.com/spreadsheets/d/1H7qA99Flnj192xnHSVvvW2mdkJyY2DaDOvdUEUfbBu4/edit?usp=sharing).

	03.02.2021			04.02.2021		
Учасник	Що зроблено?	Що буде зроблено?	Які проблеми?	Що зроблено?	Що буде зроблено?	Які проблеми?
Безух Дмитро		Дослідження і вибір технології		Порівняння деяких із технологій	Остаточне затвердження технології	Бажання замовника (уточнення)
Гутаревич Олександр		Дослідження і вибір технології		Порівняння деяких із технологій	Вибір типу тестування та способу його реалізації	Бажання замовника (уточнення)
Соняк Софія		Аналіз існуючих схожих систем + trello		Відібрані найбільш схожі системи, виділено їх переваги та недоліки + trello	Звіт по відібраним системам	
Ярошевська Надія		Макет дизайну головної сторінки		Ескіз головної сторінки	Варіанти заповнення сторінки	
	08.02.2021			09.02.2021		
Учасник	Що зроблено?	Що буде зроблено?	Які проблеми?	Що зроблено?	Що буде зроблено?	Які проблеми?
Безух Дмитро	Затвердження технології програмування	Програмний розподіл на викладача та студента+реєстрація/вхід до системи		Програмний розподіл на викладача та студента+реєстрація/вхід до системи	Підтвердження студентів викладачем	
Гутаревич Олександр	Затвердження типу тестування	Розробка алгоритму тестування		Алгоритм типового тестування	Розробка чорнового варіанту тестування	Бажання замовника (уточнення)
Соняк Софія	Звіт по відібраним системам	Формування ТЗ	Бажання замовника (уточнення)	Чорновий варіант ТЗ	Чорновий варіант специфікації	Бажання замовника (уточнення)
Ярошевська Надія	Текстові варіанти заповнення	Оформлення реєтраційної форми		Оформлення реєстраційної форми	Оформлення підтвердження	
	10.02.2021			11.02.2021		
Учасник	Що зроблено?	Що буде зроблено?	Які проблеми?	Що зроблено?	Що буде зроблено?	Які проблеми?
Безух Дмитро	Підтвердження студентів викладачем	Сторінка викладача зі студентами		Сторінка викладача зі студентами	Перехід на сторінку студента	
Гутаревич Олександр	Чорновий варіант тестування	Розробка алгоритму тестування		Алгоритм тестування	Тестування студентів (демо-1)	
Соняк Софія	Чорновий варіант специфікації	Заповнення документації + діаграми		Документація + діаграми	Алгоритм тестування продукту	
Ярошевська Надія	Оформлення підтвердження	Оформлення сторінки викладача		Оформлення сторінки викладача	Оформлення сторінки студента	

Рисунок 1. Беклог групи

Окрім того, для розстановки завдань та дедлайнів використовувалася система Trello (рис. 2).

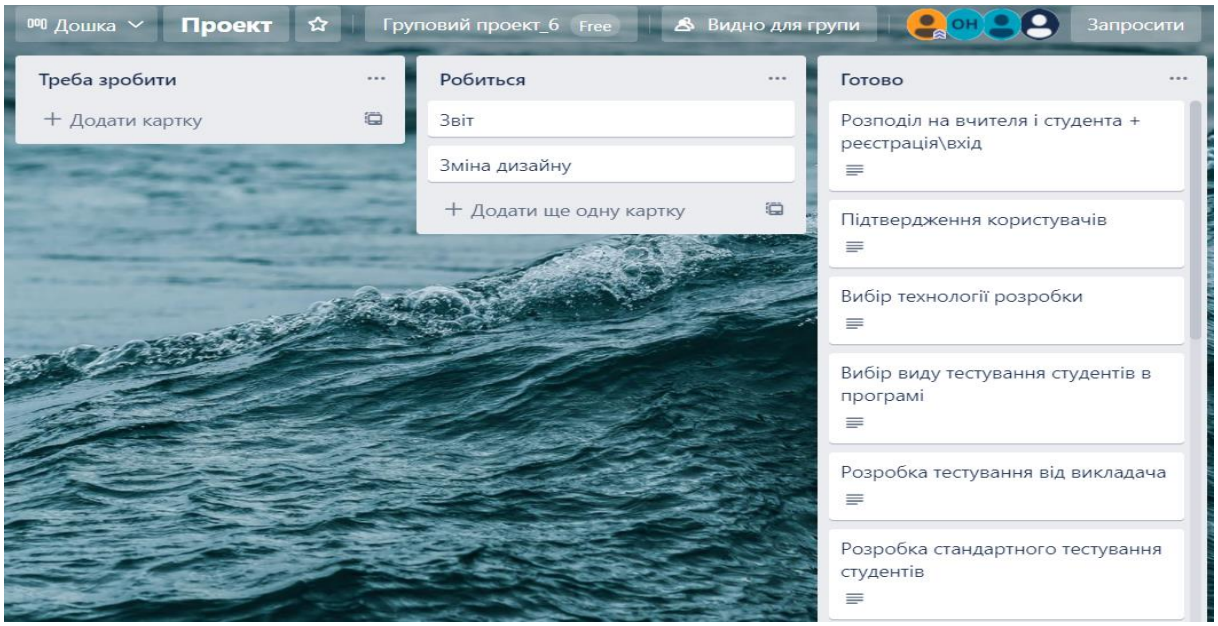


Рисунок 2. Дошка Trello

ПІДГОТОВКА ДО РЕАЛІЗАЦІЇ ПРОЄКТУ

Аналіз ринку. Було розглянуто два найвідоміших ресурси, які використовуються для навчання, зокрема, Moodle та Google Classroom, з метою виділення переваг та недоліків кожної із систем для підготовки підґрунтя для власного проєкту (рис. 3).

	Clasroom	Moodle	Пояснення	Clasroom	Moodle
Простота інтерфейсу	+	+-		Один курс - один клас; вся інформація по курсу в одному класі	Курс для всіх предметів; не зовсім логічне розташування вкладок; на головній сторінці - все і відразу
Можливість створювати тести прямо в системі	-	+		Тести створюються за допомогою іншого ресурсу і зберігаються на драйві	Можна створювати тести
Можливість викладати лекційний матеріал	+	+		Можна викладати за тематиками	Можна викладати за тематиками
Можливість автоматичного підрахування балів та їх перенесення	+-	+		Так як результати зберігаються на диску, не зважди відбувається коректне підтягнення результатів	Бали за тестування автоматично підраховуються
Можливість виведення загального балу за курс	+	+-		Бали сумуються для кожного студента	Не завжди інтуїтивно зрозуміла розбаловка через наявність безлічі не зовсім зрозумілих вкладок
Простота приєднання до курсу	+	+-		Студент приєднується за посиланням із уже наявної пошти, або ж викладач приєднує за списком	Викладач повинен створити власноруч список студентів та надати їм дані акаунту
Повідомлення на пошту	+	-		Студенту приходять сповіщення на привязану пошту про виставлення балів, оновлення інформації	Такої функції немає.

Рисунок 3. Порівняльна характеристика систем

Такий порівняльний аналіз допоміг виявити необхідні аспекти, яким потрібно приділити більше уваги.

Проєктування системи. Для проєктування системи було використано декілька типів діаграм. Варто виділити діаграму прецедентів, яка описує роботу системи повністю (див. рис. 4).

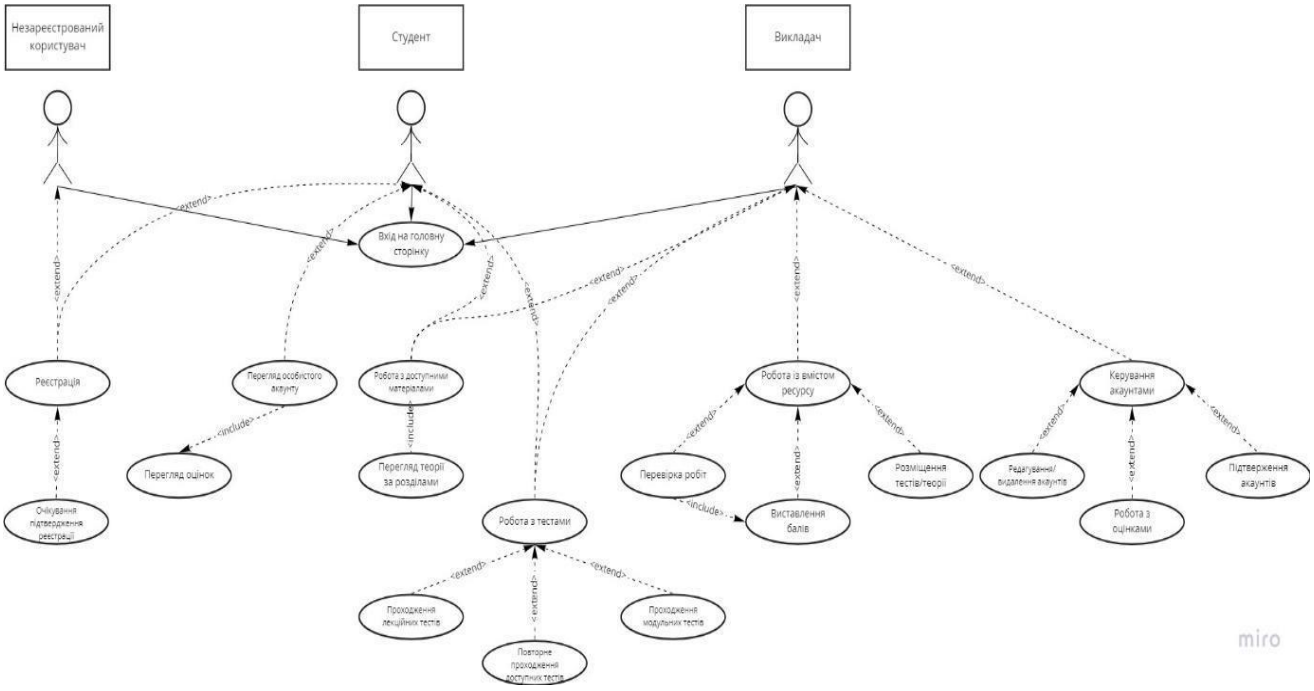


Рисунок 4. Діаграма прецедентів

На діаграмі представлено три ролі, які доступні в системі, а саме: незареєстрований користувач, студент та викладач та їх функції.

Написання технічного завдання. Наступним етапом роботи стало написання технічного завдання. За його основу було взято вимоги замовників, а також порівняльну характеристику систем.

РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЧАСТИНИ

Програма частина проєкту реалізована мовою C# (.Net framework). Із кодом програми можна ознайомитися на GitHub за посиланням https://drive.google.com/drive/folders/1jOM7O3c9mjIMKIlYnSOnseiAQGhSrx3g?usp=sharing.

ТЕСТУВАННЯ, ВЕРИФІКАЦІЯ ТА СПЕЦИФІКАЦІЯ

Тестування в проєкті відбувалося за допомогою Unit тестування, а також було використано інтегроване тестування.

Для тестування було обрано частину програмної логіки системи, а саме перевірку правильності проходження тесту користувачем та прохідність порогу модульного тестування. Тести покривають практично всі можливі сценарії. Також варто додати те, що цей тестований модуль використовується в частині коду, де відбувається процес тестування студентів.

Для інтегрованого тестування було обрано програмну частину реєстрації. В процесі цього тестування створюється окрема база даних.

Усі тести успішно проходять (рис. 5).

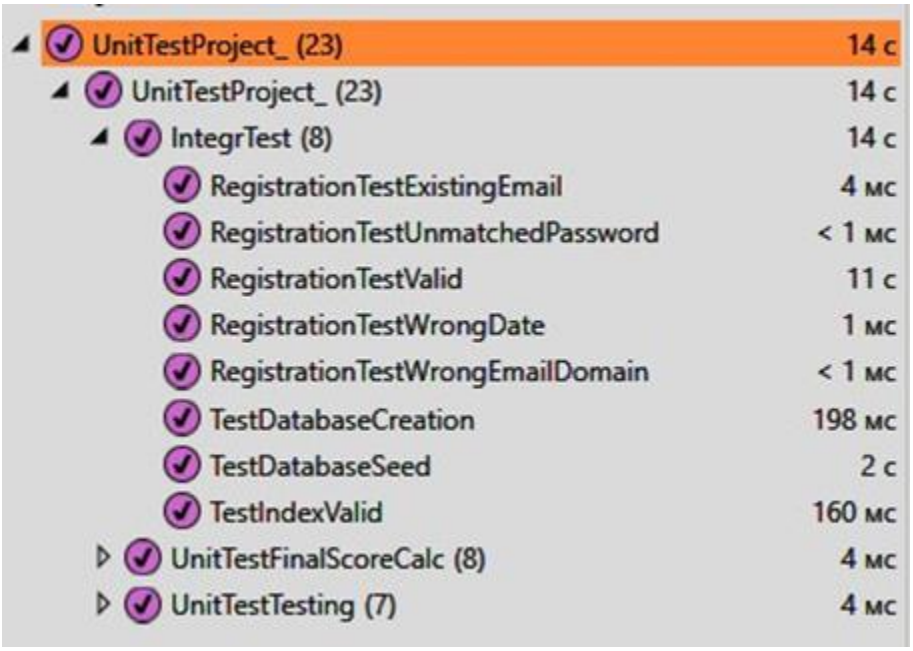


Рисунок 5. Проходження тестів

Верифікація програми відбувалася на мові Code Contracts. В процесі верифікації відбулася перевірка сумування балів в тестах. Код тесту:

```
public static int Sum(string numbers)
{
    Contract.Requires(numbers != null);

    if (numbers.Trim() == string.Empty)
        return 0;

    return numbers
        .Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries)
        .Select(int.Parse).Sum();
}
```

Для специфікації було обрано частину, яка підраховує суму балів за пройдений тест. У процесі специфікації використовувалася мова Dafny. У процесі специфікації помилок виявлено не було, а кількість балів було підраховано правильно (рис. 6).

```
61 }
62
63 method Main() {
64
65     var que1 := new Test("qwer", 2);
66     var que2 := new Test("qwert", 1);
67
68     var a := new Student({que1,que2},"qwer");
69     var sum := a.TransformToArray(a);
70   }
71
72
```

tutorial home video permalink

'▶' shortcut: Alt+B

		Description	Line	Column
	1	the type of the other operand is a non-null type, so this comparison with 'null' will always return 'true' (to make it possible for variable 'a' to have the value 'null', declare its type to be 'array?<int>')	22	15
	2	the type of the other operand is a non-null type, so this comparison with 'null' will always return 'true' (to make it possible for variable 'a' to have the value 'null', declare its type to be 'array?<int>')	30	15

```
stdin.dfy(22,15): Warning: the type of the other operand is a non-null type, so this comparison with 'null' will always return 'true' (to make it possible for variable
stdin.dfy(30,15): Warning: the type of the other operand is a non-null type, so this comparison with 'null' will always return 'true' (to make it possible for variable

Dafny program verifier finished with 5 verified, 0 errors
Running...

3
```

Рисунок 6. Специфікація тестування

ІНСТРУКЦІЯ КОРИСТУВАЧА

Як зазначалося раніше, в системі є три ролі. Далі розглядатиметься кожна із них. При відкритті, кожен користувач потрапляє на головну сторінку (рис. 7).

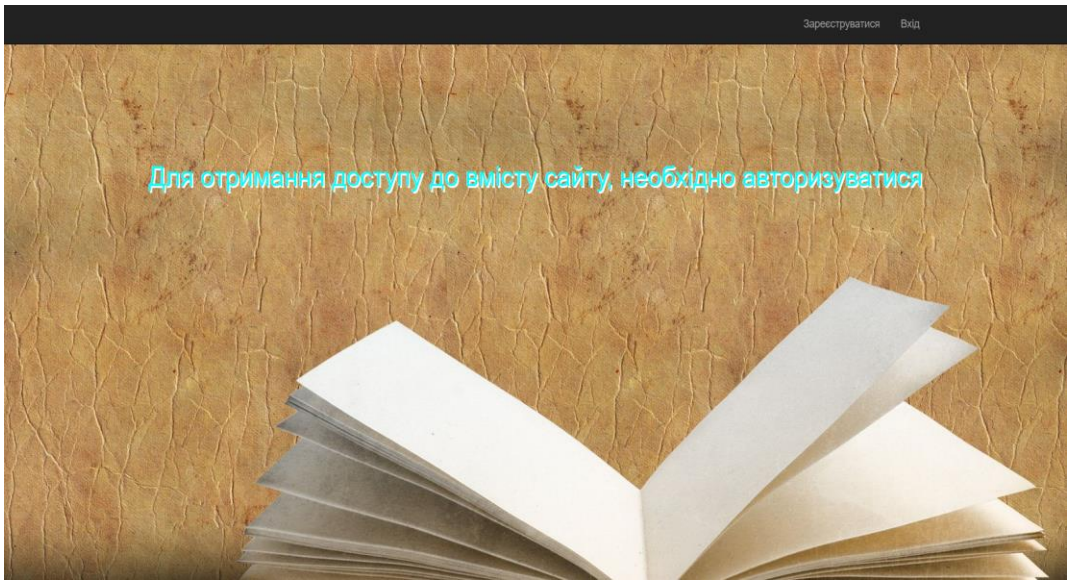


Рисунок 7. Головна сторінка сайту

Незареєстрований користувач. Для незареєстрованого користувача функціонал недоступний. При реєстрації необхідно заповнити такі поля: ім’я, прізвище, пошта, пароль, його підтвердження та дату народження (рис. 8).

Рисунок 8. Реєстрація користувача

Після заповнення, необхідно очікувати підтвердження від викладача. Коли акаунт буде активованим, прийде повідомлення на електронну пошту.

Викладач. У викладача досить широкий функціонал, зокрема, зміна статусу профілю та його видалення, додавання користувачів (рис. 9).

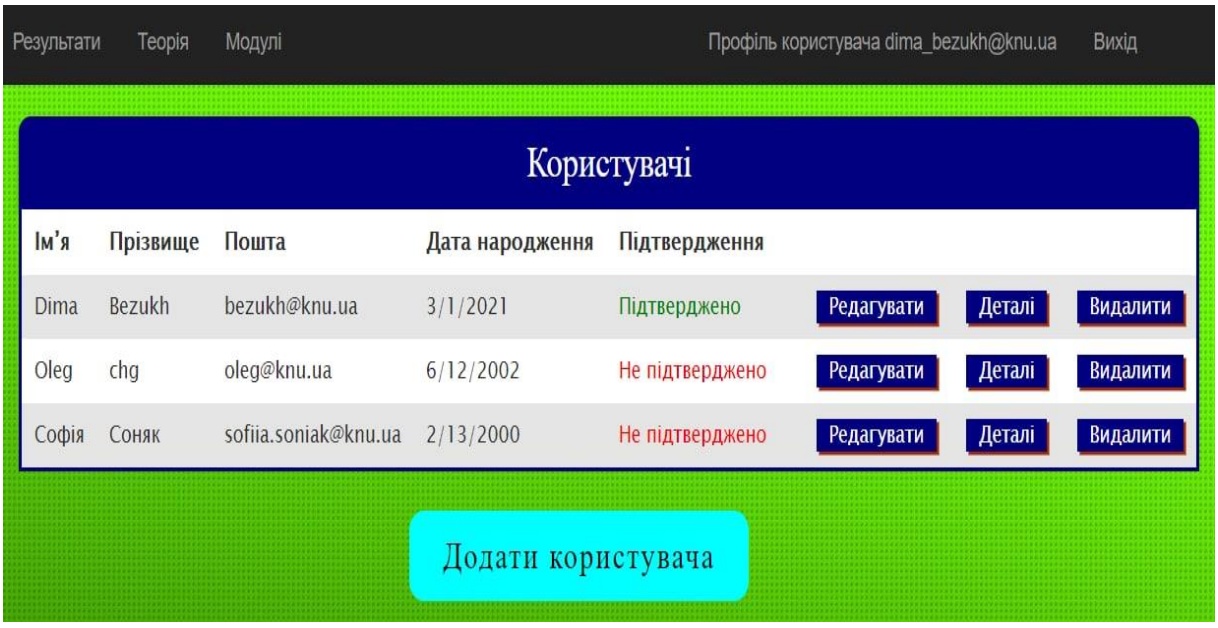


Рисунок 9. Робота із профілями користувачів

Викладач може розміщувати тести, редагувати їх та налаштування (рис. 10).

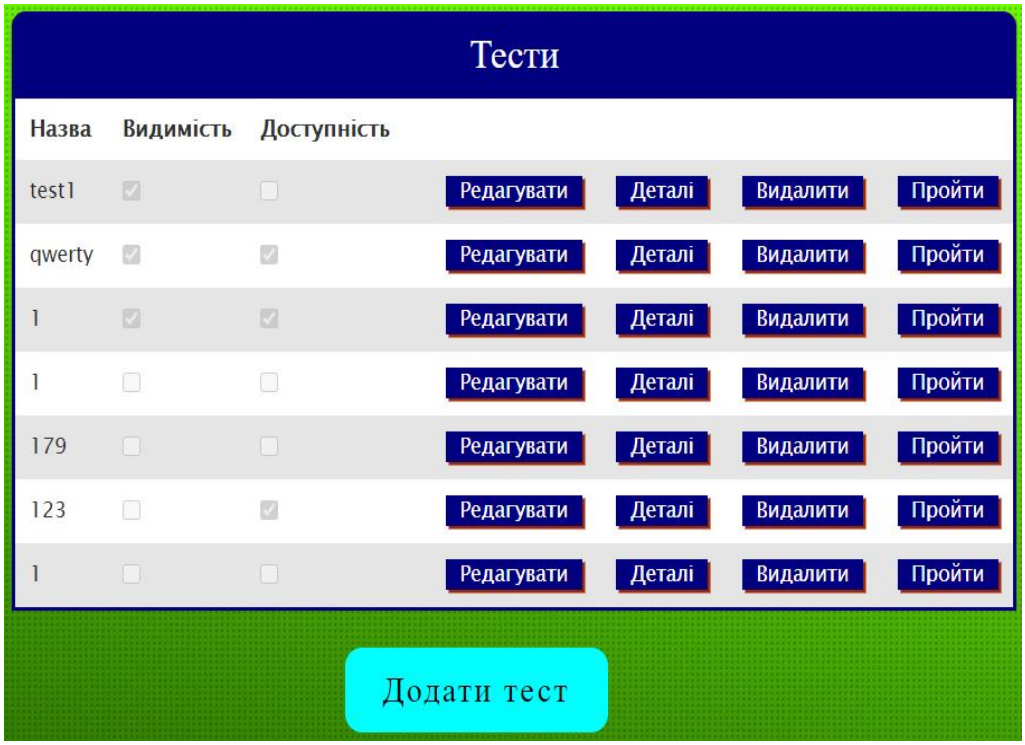


Рисунок 10. Робота з тестами

Окрім того, викладачу доступне проходження тестів. Також викладач може переглядати результати усіх тестувань студентів та деталі проходження (рис. 11).

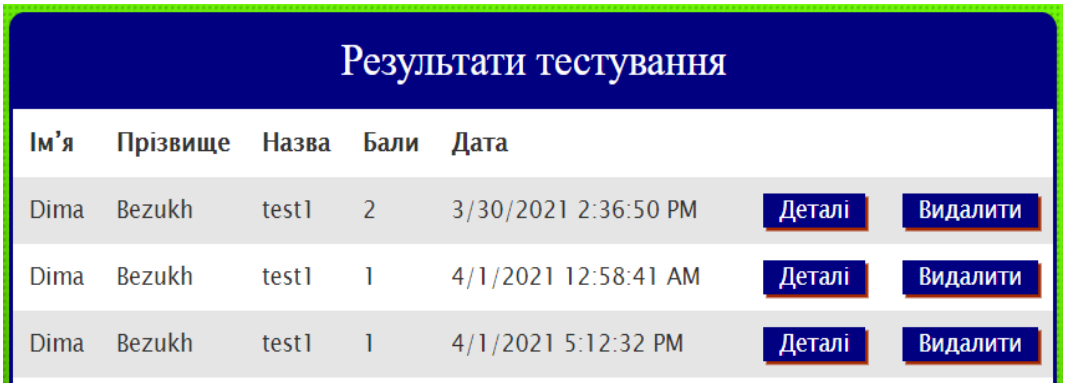


Рисунок 11. Результати тестувань

Однією із основних функцій є розміщення лекцій та прив'язка їх до відповідних модульних тестувань, де встановлений поріг для проходження до наступних тем курсу (рис. 12).

Рисунок 12. Робота з модулями

Студент. Для студента ж функціонал дещо звужений, зокрема, студент може проходити тестування, переглядати лекції (якщо попередні здані на достатньому рівні для допуску). В особистому кабінеті студента показані результати його тестувань, доступна функція зміни паролю (рис. 13). Видаляти акаунти може лише викладач.

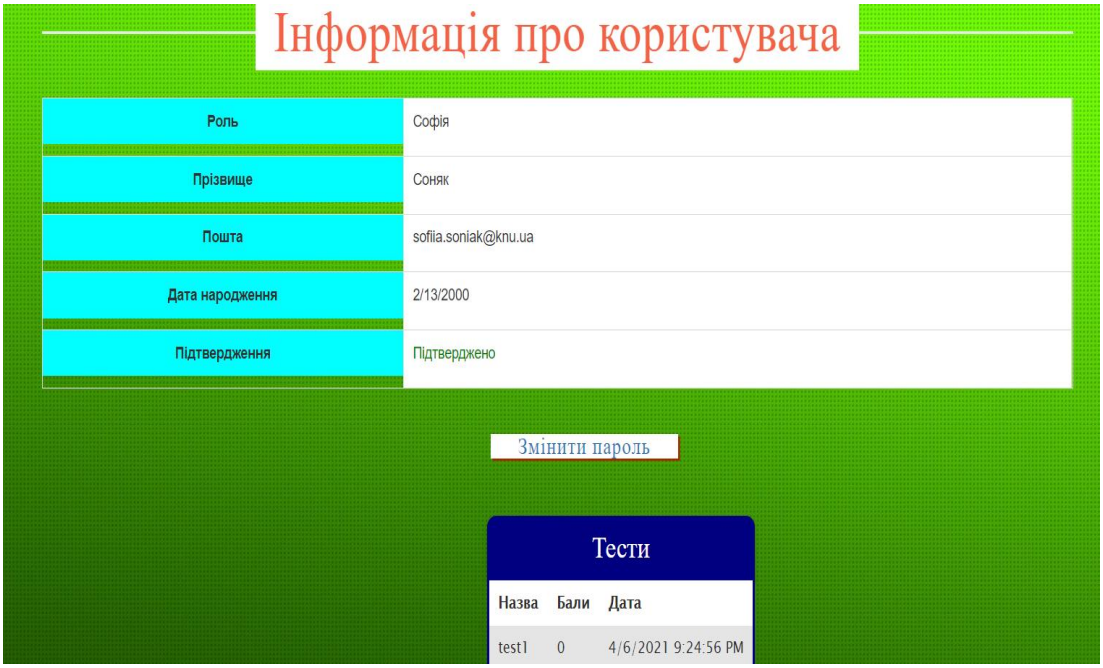


Рисунок 13. Особистий кабінет користувача

ВИСНОВКИ

У процесі розробки було створено систему, яка повністю відповідає технічному завданню. Окрім того, в процесі розробки були додані деякі модулі для покращення роботи системи, які не були передбачені на початку розробки даної системи.

МОДУЛЬ УПРАВЛІННЯ КУРСАМИ ЯК РОЗШИРЕННЯ ПЛАТФОРМИ GOOGLE CLASSROM

Віталій Борисов, Владислав Каплюк, Віталій Кльоз, Вікторія Рудзей

ПОСТАНОВКА ЗАДАЧІ

Інформаційно-комунікаційна "Система інтеграції Google Classroom" (далі – програмний продукт, Система) розробляється як єдине комплексне рішення з метою створення зручного інструмента для взаємодії користувача з ресурсом Google Classroom.

Інструментарій. Використані технології проєктування: Jira, Google диск з Google Docs. Використані технології розробки:

- Сервіс: Node js, typescript, Express, Mongodb, axios, oauth, jwt
- Деплоймент: Heroku, Atlas
- Фронтенд: JS, Velo

Використані технології верифікації та тестування: Jest, Nock.

Методологія та система управління проєктом. Для зручного управління та введення проєкту було використано методологію Scrum, системою управління була Jira Software. Було створено 3 Епіка:

- Front-End;
- Back-End;
- Документація.

Це сприяло розподілу задач та відображенню прогресу виконання кожного їх блоку.

ТЕХНІЧНІ ВИМОГИ

Система повинна забезпечити можливості доступу користувачів до системи Google Classroom та використання основних функцій, наявних у Google Classroom.

Система повинна забезпечувати реалізацію наступних завдань:

- можливість створювати клас у Google Classroom;
- можливість переглянути списки завдань та людей;
- можливість додавати завдання;
- можливість відправляти повідомлення декільком класам.

Цільовою аудиторією є користувачі Системи. Цільова аудиторія - викладачі, що створюють декілька класів у Google Classroom.

Розробка компонентів програмного забезпечення. В рамках розробки інтегрованої системи були розроблені наступні програмні компоненти:

І блок – Розробка програмного забезпечення

1. Стартова сторінка з авторизацією. Google авторизація.
2. Сторінка створення курсу.
3. Сторінка для додавання студентів.
4. Сторінка для групування студентів.
5. Сторінка поширення анонсів.
6. Сторінка поширення завдань
7. Сторінка для показу всіх курсів.
8. Створення сторінки для вибору дій.

Вимоги до розробки системи. Система повинна бути оптимізованою і зверстаною у відповідності зі стандартами W3C для роботи в різних стабільних версіях браузерів (без помилок). Для правильного відображення інтерфейсу Системи на різних пристроях і браузерах повинна бути коректна, адаптивна верстка. Адаптивність верстки повинна бути забезпечена за допомогою засобів Angular. Всі стилі потрібно максимально прибрати в зовнішні файли CSS, включаючи зображення, що відносяться до дизайну. Всі JS-скрипти повинні бути оптимізовані на максимальну продуктивність системи і заховані в зовнішні файли JS. Сторінки Системи не повинні містити флеш-контент. На всі сторінки, файли і зображення повинні бути ЛЗП.

Система повинна забезпечувати коректне відображення даних в наступних браузерах актуальних версій: Google Chrome; Internet Explorer; Opera; Edge; Mozilla Firefox; Safari.

Загальні вимоги. Система повинна мати архітектуру, побудовану на сучасних технологіях зберігання, обробки, аналізу даних та доступу до них; забезпечувати одночасну роботу декількох користувачів. Рішення щодо побудови Системи повинні базуватися на:

- застосуванні сучасних інформаційних технологій;
- реалізації концепції створення єдиного інформаційного простору;
- застосуванні правила централізованого накопичення, зберігання та обробки інформації.
- підтримці актуальності, повноти, несуперечності, цілісності та доступності інформації;
- забезпеченні надійного захисту інформації від порушення її цілісності, витоку та блокування згідно з вимогами нормативно-правових документів в галузі захисту інформації;
- забезпеченні надійності, резервування компонентів технічного забезпечення Системи;
- забезпеченні централізованого управління, безперервного контролю функціонування та централізованого налаштування Системи і модулів її компонентів;
- використанні сучасних засобів програмної інженерії при розробці програмного прикладного забезпечення.

Система представляє собою комплекс інформаційних, програмних, технічних, організаційно-методичних та інших необхідних засобів, що забезпечують збір, обробку, зберігання та передачу даних.

Архітектура Системи повинна передбачати максимальну незалежність програмно-технічних модулів від Виконавця.

Інформаційна архітектура Системи повинна відповідати сучасним вимогам щодо побудови інтерфейсів користувачів.

Вимоги до режимів функціонування. Цілодобове безперервне повноцінне функціонування відповідно до заявленого фунціоналу при наявності хостинга зі сторони клієнта. Експлуатація Системи повинна передбачати такі режими:

- основний режим – режим штатного функціонування всіх модулів та компонентів за призначенням;
- нештатний режим – режим нештатного функціонування всіх компонентів Системи;
- режим адміністрування – режим здійснення централізованого автоматизованого налагоджування та автоматизованого оновлення Системи одночасно із роботою решти користувачів в основному режимі;

- режим регламентного обслуговування – режим регламентного технічного обслуговування та відновлення працездатності технічних засобів компонентів Системи.

Вимоги до надійності. Надійність Системи повинна бути забезпечена за наступними напрямками:

- забезпечення працездатності компонентів Системи. Механізми із збереження працездатності повинні забезпечувати надійність роботи при відмові одного або декількох компонентів за рахунок їх резервування. При цьому повинна вимагатися мінімальна увага з боку Адміністратора щодо реакції на усунення наслідків відмов компонентів, а також програмно-апаратними засобами повинно бути забезпечене збереження даних;
- збереження даних повинно забезпечувати збереження цілісності даних при програмно-апаратних відмовах, помилках шляхом використання відповідних програмно-апаратних засобів та рішень, резервного копіювання, транзакційності при змінах даних.

Збереження даних має забезпечуватися у випадках:

- вимкнення живлення;
- відмови технічних засобів обробки інформації;
- помилки, збоїв або руйнування програмного забезпечення;
- тимчасової відмови ліній зв'язку.

Надійність функціонування Системи повинна забезпечуватися:

- використанням сучасних технологій розробки (модернізації) прикладного
- програмного забезпечення та забезпеченням якісного його тестування;
- резервуванням основних компонентів;
- регламентом організації резервного копіювання та архівного зберігання інформації; - оперативністю заміни програмно-технічних засобів, що вийшли з ладу; - сумісністю технічних засобів та програмного забезпечення.

Повинно бути реалізованим гаряче резервування, у відповідності до якого дублюючі компоненти знаходяться у режимі "гарячого" резерву. У разі відсутності відклику основного компонента здійснюється перенаправлення трафіку на резервну систему. Вимоги щодо надійності Системи можуть бути уточнені Виконавцем.

Вимоги до ергономіки. Рішення щодо ергономіки повинно забезпечувати:

- зрозумілу логічну побудову сторінок та переходів відповідно до інформаційної архітектури;
- вбудовані механізми валідації значень, що визначаються для окремих полів, комбінацій полів (контекстно-залежний контроль), контроль значень полів за довідниками/класифікаторами.

Вимоги до захисту інформації від несанкціонованого доступу. Базові вимоги із забезпечення захисту інформації від несанкціонованого доступу повинні бути реалізовані організаційно-адміністративними заходами, апаратно-програмним та інженерно-технічним забезпеченням.

Повинні бути реалізовані наступні механізми:

- контроль цілісності програмного забезпечення;
- тестування на правильність функціонування та блокування роботи в разі виявлення порушень;
- захист від порушення конфіденційності інформації внаслідок помилкових дій користувача або в разі відхилень у роботі складових елементів Системи;

- захист ключових даних на їх носіях від несанкціонованого зчитування;
- захист від здійснення порушником навмисного зовнішнього впливу;
- захисту від порушення конфіденційності та цілісності ключових даних у ключових документах.

Повинні бути реалізовані наступні функції:

- захист трафіку на рівні автентифікації/шифрування мережевих пакетів з використанням сучасних протоколів;
- забезпечення конфіденційності та цілісності інформації шляхом здійснення її криптографічного перетворення (виконання функцій шифрування та дешифрування) та здійснювати розмежування доступу до такої інформації;
- пакетної фільтрації трафіку з використанням інформації в полях заголовків мережевого і транспортного рівнів;
- закриття всіх мережевих портів, крім тих, що необхідні для функціонування Системи;
- доступ до серверів виключно через SSH та виключно за ssl-ключами; - класифікації та маркування трафіку;
- реалізації заданого протоколу взаємодії (автентифікацію та/або захист трафіку) для кожного захищеного з'єднання, доступ в заданому захищеному режимі тільки для зареєстрованих партнерів по взаємодії;
- регульованої стійкості захисту трафіку;
- підтримки списку відкликаних сертифікатів (CRL – Certificate Revocation List); - реєстрації подій.

Вимоги до патентної чистоти. Патентна чистота модифікації Системи має бути забезпечена Виконавцем.

Вимоги до стандартизації та уніфікації. Стандартизація та уніфікація функцій модулів та компонентів Системи повинна бути забезпечена за рахунок використання сучасних інструментальних програмних засобів, які підтримують єдину технологію проєктування і розробки функціонального, інформаційного та програмного забезпечень.

У процесі розробки модулів та компонентів Системи повинні бути сформовані вимоги до розробки прикладного програмного забезпечення, які уніфікують процедуру обробки інформації, ідентифікацію програмних модулів та баз даних, типізують окремі програмні модулі відповідно до свого призначення.

Вимоги до інформаційного забезпечення. Інформаційне забезпечення повинно відповідати таким вимогам та можливостям:

- забезпечення фізичної та логічної цілісності даних;
- мінімізація надмірності даних, що зберігаються;
- стандартизація представлення даних;
- достовірність та актуальність даних.

Календарний план. На першій зустрічі перед початком розробки було сформовано календарний план проєкту (табл. 1).

Таблиця 1. Календарний план проєкту

№ з/п	Назва послуг за етапами етапу	Термін	Результат	Вартість послуги без ПДВ, грн.
1	Створення технічного завдання на розробку	5 робочих днів **	Технічне завдання	гарні оцінки
2	Розробка дизайну	7 робочих днів**	Інтерактивний зразок програмного забезпечення	гарні оцінки
3	Розробка програмного забезпечення	24 робочих днів **	Дослідний зразок програмного забезпечення	гарні оцінки
4	Загальна модернізація програмного забезпечення	5 робочих днів **	Програмне забезпечення на останній стадії до тестування	гарні оцінки
5	Тестування програмного забезпечення	5 робочих днів **	Програма та методика приймальних випробувань Опис системи Керівництво користувача Керівництво адміністратора Завершене програмне забезпечення	гарні оцінки + приз

** Замовник може надавати власний варіант календарного плану.*

*** Строк розробки та порядок надання ТЗ може бути змінено, робочими днями вважаємо середу, четвер та п'ятницю*

ВЕРИФІКАЦІЯ

Оскільки розробка велась на основі node.js мова написання серверної частини додатку була typescript, то довелось знаходити альтернативи які можна було б використати для верифікації.

Все ж знайшлась альтернатива code contracts .NET для typescript називається typescript code contracts (https://www.npmjs.com/package/ts-code-contracts) .

Приклад верифікації сутності яка працює з базою даних:

```
export class IntegratedCourseDaoImpl implements IntegratedCourseDao {
    private readonly collection: Collection<IntegratedCourse>;
    constructor(mongoClient: MongoClient) {
        this.collection = mongoClient.db("my-db").collection<IntegratedCourse>("my-collection");
    }

    async get(id: string): Promise<IntegratedCourse> {
        requires(isDefined(id) && id !== "", "IntegratedCourseId must be populated");
        return ensuresNonNullish(this.collection.findOne({ id }, { projection: { _id: 0 }}), "Integrated course with such id does not exist");
    }

    async create(params: Omit<IntegratedCourse, "id">): Promise<IntegratedCourse> {
        requires(params.courses.length > 0, "Integrated course should contain at least 1 course");

        const course: IntegratedCourse = {
            ...params,
            id: uuid(),
        }
        await this.collection.insertOne(course, {forceServerObjectId: true});
        return course;
    }
```

Рисунок 1. Верифікація IntegratedCourseDAO

Приклад верифікації одного з методів сервісу інтегрованих курсів:

```
async createIntegratedCourse(oauthClient: OAuth2Client, integratedCourseParams: IntegratedCourseCreationParams): Promise<string> {

    requires(isDefined(integratedCourseParams), "Integrated course cannot be null or undefined");
    requires(isDefined(integratedCourseParams.name) && integratedCourseParams.name !== "", "Integrated course name required");
    requires(integratedCourseParams.studentGroups.length > 0 , "Integrated course should contain at least 1 group");
    requires(integratedCourseParams.teachers.length > 0, "Integrated course should contain at least 1 teacher");

    const userEmail = await this.classroomFacade.getUserEmail(oauthClient);
    const teachersWithoutOwner = integratedCourseParams.teachers.filter(item=> item !== userEmail);

    const createGroupPromises = integratedCourseParams.studentGroups.map(async(studentGroup)=>{
        return this.createGroup(oauthClient, userEmail, teachersWithoutOwner, studentGroup);
    });

    const createdGroups = await Promise.all(createGroupPromises);
    const { id } = await this.coursesDao.create({
        owner: userEmail,
        name: integratedCourseParams.name,
        courses: createdGroups.map((classroomCourseId)=>({classroomCourseId}))
    });
    return id;
}
```

Рисунок 2. Верифікація IntegratedCourseManagerService

Приклад верифікації фасаду взаємодії з Google classroom:

```
public async updateCourseState(oAuth2Client: OAuth2Client, courseId: string, courseName: string, courseState: CourseState) : Promise<void>{
    requires(isDefined(courseName) && courseName !=="" , "Course Name cannot be null or empty");
    requires(isDefined(courseId) && courseId !=="" , "Course Id cannot be null or empty");
    requires(isDefined(courseState), "Course Id cannot be null or undefined");

    const response =  await this.classroom.courses.update({
        auth: oAuth2Client,
        id: courseId,
        requestBody: {
            name: courseName,
            courseState,
            id: courseId
        }
    });
    return;
}

public async createCourse(oAuth2Client: OAuth2Client, teacherOwnerEmail: string, courseName: string) : Promise<string> {
    requires(isDefined(courseName) && courseName !=="" , "Course Name cannot be null or empty");
    requires(isDefined(teacherOwnerEmail) && teacherOwnerEmail !=="" , "Teacher Owner Email cannot be null or empty");

    //TODO DOnot have permission to create show proper error
    const response = await this.classroom.courses.create({
        auth: oAuth2Client,
        requestBody: {
            ownerId: teacherOwnerEmail,
            name: courseName
        }
    });
    await this.updateCourseState(oAuth2Client, response.data.id, courseName, CourseState.Active);
    return response.data.id;
}

public async addCourseTeacher(oAuth2Client: OAuth2Client, teacherEmail: string, courseId: string) : Promise<void> {
    requires(isDefined(teacherEmail) && teacherEmail !=="" , "Teacher email cannot be null or empty");
    requires(isDefined(courseId) && courseId !=="" , "Course Id cannot be null or empty");

    const response = await this.classroom.courses.teachers.create({
        auth: oAuth2Client,
```

Рисунок 3. Верифікація GoogleClassroomFacade

Приклад верифікації фасаду взаємодії з Google Drive:

```
export class GoogleDriveFacade {

    private readonly drive: drive_v2.Drive;

    constructor() {
        this.drive = google.drive("v2");
    }

    async getFileName(oAuth2Client: OAuth2Client, fileId: string) : Promise<string> {
        requires(isDefined(fileId) && fileId !=="", "fileId cannot be null or empty");

        const touchedFile = await this.drive.files.touch({
            auth: oAuth2Client,
            fileId: fileId
        });
        return touchedFile.data.title;
    }

    async copyFile(oAuth2Client: OAuth2Client, fileId: string, distDirectoryId: string, title: string) : Promise<FileDto> {
        requires(isDefined(fileId) && fileId !=="", "File Id cannot be null or empty");
        requires(isDefined(distDirectoryId) && distDirectoryId !=="", "Distanation directory Id cannot be null or empty");
        requires(isDefined(title) && title !=="", "File title cannot be null or empty");

        const copiedFile = await this.drive.files.copy({
            auth: oAuth2Client,
            fileId,
            requestBody: {
                title,
                parents:[{
                    id: distDirectoryId
                }]
            }
        });
        return {
            id: copiedFile.data.id,
            title: copiedFile.data.title,
            link: copiedFile.data.alternateLink
        }
    }
```

Рисунок 4. Верифікація GoogleDriveFacade

ТЕСТУВАННЯ

Тестування бекенду. Оскільки основною роллю сервісу було взаємодіяти з сторонніми сервісами та базою даних, при тестуванні використовувались в більшості end to end та інтеграційні тести, в комбінації з статичним тестуванням.

Хоча end to end тестування зазвичай є найбільш затратним типом тестування по часу, завдяки паралельному запуску тестів, перевикористанню тестового середовища та тому, що сервіс є невеликим, різниця в швидкості не була помітною.

Проте цей вибір дав нам високу надійність та зручність у розробці й тестуванні.

Для створення зручних, коротких та читабельних тестів були створені окремі класи-драйвери, що імплементують функції для моків зовнішніх сервісів GoogleClassroom, GoogleDrive, GoogleOAuth; моків бази даних; виклику кожного з наших ендпоїнтів з та без авторизаційного токену. Приклади коду драйверів:

```
given = {
    courses: (courses: CourseMock[]) => {
        const response: Schema$ListCoursesResponse = { courses }
        nock( basePath: 'https://classroom.googleapis.com') Scope
            .persist() Scope
            .get( uri: '/v1/courses') Interceptor
            .query( matcher: () => true) Interceptor
            .reply( responseCode: 200, response)
    }
```

Рисунок 5. Мок відповіді від GoogleClassroom, GoogleClassroomDriver.ts

```
given = {
    userWithValidToken: (email: string) => {
        this.credentials = {
            refresh_token: 'refresh_token',
            access_token: 'access_token',
            id_token: `${email}-id_token`
        };

        const response: GetTokenResponse = {
            tokens: this.credentials,
            res: null
        }
        nock( basePath: 'https://oauth2.googleapis.com')
            .persist() Scope
            .post( uri: '/token') Interceptor
            .reply( responseCode: 200, response)
    }
```

Рисунок 6. Мок відповіді від GoogleAuth та зберігання цього токену з метою виклику нашого ендпоїнта з авторизованим токеном, ServerDriver.ts

```
when = {
    listYourCourses: () => {
        return request(this.server) SuperTest<Test>
            .get( url: `/list-your-courses`) Test
            .set('Authorization', this.jwtAuthToken ?? '') T
            .send();
    }
```

Рисунок 7. Функція для виклику listYourCourses ендпоїнту з авторизаційним токеном, якщо перед цим був викликаний метод userWithValidToken, ServerDriver.ts]

Приклади тестів з використанням описаних драйверів:

```
describe( name: '/list-your-courses', fn: () => {
    it( name: 'should return "no valid token" when jwt token is not passed', fn: async () => {
        await driver.given.courses( courses: []);

        const res = await driver.when.listYourCourses();

        expect(res.status).toEqual( expected: 401);
        expect(res.text).toMatchJSON( b: { message: 'No valid jwt oauth token provided' });
    })
```

Рисунок 8. Тестування повернення помилки при виклику listYourCourses, коли в запиті відсутній авторизаційний токен

```
it( name: 'should return all Classroom courses of the authorized user', fn: async () => {
    const userEmail = 'test@gmail.com';
    const courses = [
        {id: '1', name: 'course1', alternateLink: '1234'},
        {id: '2', name: 'course2', alternateLink: '123456'}
    ];

    await driver.given.userWithValidToken(userEmail)
    await driver.given.courses(courses);

    const res = await driver.when.listYourCourses();

    const expectedCourses: CourseDto[] =
        courses.map(({id : string , name : string , alternateLink : string }) => ({ id, name, link: alternateLink, students: []}))
    expect(res.text).toMatchJSON(expectedCourses)
})
```

Рисунок 9. Тестування повернення курсів при виклику listYourCourses, коли в запиті є валідний авторизаційний токен

Використані технології: тестувальні фреймворки Jest та Jasmine, бібліотека для http моків Nock та драйвер для тестів mongDbTestkit.

Тестування фронтенду. Тестування фронтенду проводилось мануально при додаванні нових елементів / даних. Оскільки код та елементи різних по змісту сторінок були розділені між собою, тестувати потрібно було лише певну частину.

ПРОБЛЕМИ, ПОМИЛКИ, ПЛАНИ РОЗРОБКИ

Помилки та проблеми під час розробки проєкту, початкова концепція. Початковою концепцією було створення телеграм боту для Classrom, з можливістю нотифікації для студентів, тобто оповіщення про кожен новий допис створений викладачем повинен приходити в телеграм-бот.

Було витрачено багато часу на побудову моделі реалізації та авторизації, займались дослідженням API, що могло б стати в нагоді для подальшої реалізації задумки.

Зіштовхнулись з тим, що Google Classroom API не надає можливості отримувати дані про зміни станів та оновлень в Classroom в реальному часі: це стосується оголошення завдання, самі завдання, склад людей, тощо.

Єдиний варіант такої інтеграції було би опитування (polling) (див.рис.6.1) всіх ендпоїнтів з певної періодичністю для отримання актуальних даних, що не задовольнило нас через низьку продуктивність, надмірні витрати за великий трафік в Classroom API та обмежену розширюваність.

Це призвело нас до наступної задумки проєкту, який ми презентуємо. Ми використали модуль авторизації, який був розроблений на початковому етапі та дійшли до нової ідеї.

Плани подальшої розробки проєкту. Для покращення роботи проєкту плануємо:

1. Розробити можливість створювати зустрічі в Google Meet для кожного з курсів. Поки що дана ініціатива на етапі дослідження можливостей;
2. Можливість інтеграції з github;
3. Покращення UI/UX front-end.

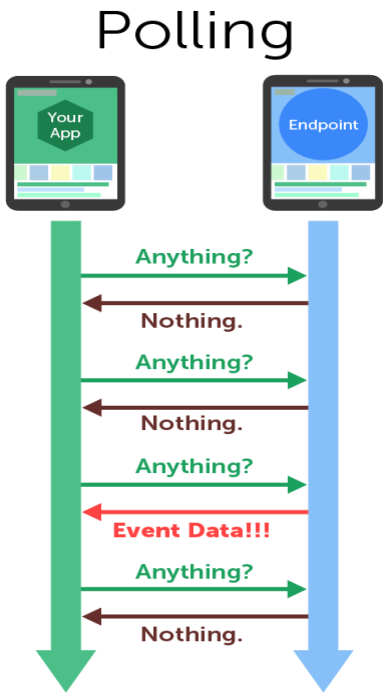


Рисунок 10. Пояснення роботи Polling

ВИСНОВКИ

Було створено інформаційно-комунікаційну система інтеграції Google Classroom як єдине комплексне рішення з метою створення зручного інструмента для взаємодії користувача з ресурсом Google Classroom.

Поглибнено знання в використанні таких технологій розробки:

Сервіс: Node js, typescript, Express, Mongodb, axios, oauth, jwt; Деплоймент: Heroku, Atlas; Фронтенд: JS, Velo.

Покращилась взаємодія один з одним та командна робота над одним проєктом була досить продуктивною. Кожен студент почерпнув для себе нові знання та навички, вдосконалив та покращив старі.

РОЗРОБКА СИСТЕМИ ПІДТРИМКИ ІНТЕГРОВАНОГО КУРСУ

Дмитро Винник, Максим Капелянович, Анастасія Полянська, Євгеній Романенко

ПОСТАНОВКА ЗАДАЧІ

Цілі та поставлені задачі. У сьогоднішніх реаліях гостро постає проблема двостороннього контакту між вчителями та учнями, а також проблема знаходження продуктивних засобів, методів та інструментів, котрі забезпечують навчальний процес. Існує багато доступних сервісів та систем, які так чи інакше допомагають у вирішенні вище описаних проблем. Наша команда провела детальний аналіз проблемної області та наявних на ринку рішень, після чого було вирішено розробити власну систему підтримки інтегрованого курсу з усуненням недоліків та забезпечення зручності у користуванні обом сторонам навчального процесу. Для реалізації цього завдання було прийняте рішення створити і веб сайт, і мобільний клієнт, оскільки так завжди можна бути в курсі новододаних завдань, оголошень.

Аудиторія. Інформаційна система "Grouper" (далі – Система) розроблена як єдине комплексне рішення з метою запровадження інструментів менеджменту групами для вчителів. Розрахована на викладачів та студентів для підтримки навчального процесу.

Інструментарій (технології розробки). Сервер системи розроблений на мові C# із використанням .NET5.0, AspNetCore Web API, Entity Framework5. База даних - MS SQL Server з використанням T-SQL, SQL Profiler. Для написання серверу використовувались Visual Studio та VS Code. Сервер запущений з використанням технологій Docker та CI/CD.

Для розробки мобільного додатку була використана мова Java та IDE Android Studio. Для реалізації зв’язку з сервером була використана бібліотека Retrofit.

Сайт написаний на JS з використанням Angular, HTML/CSS, TypeScript. Для паралельної розробки елементів однієї системи було використано система контролю версій Git та GitHub.

Розподіл ролей. При роботі над спільним проєктом у нас був розподіл ролей, який подано в табл. 1.

Таблиця 1. Розподіл ролей в команді

Роль	Член команди	Коментар
Мобільний розробник, фахівець з документації	Анастасія Полянська	Розробка мобільного додатку, інтерфейсу та створення шаблонів документації
Скрам-мастер, бекенд разробник	Романенко Євгеній	Комунікація команди, написання бекенд серверу
Дизайнер, фронтедн розробник	Дмитро Винник	Розробка дизайну інтерфейса, розробка веб інтерфейсу
Розробка мобільного додатку, архітектор	Максим Капелянович	Розробка архітектури мобільного додатку, та його інтерфейсу

Кожен мав свої задачі та основний напрям роботи. Вдалося покрити весь спектр задач. Обрані ролі базуються на основних навичках членів команди, доповнюючи один одного.

Методологія та системи управління проєктами. Під час розробки використовувалася методологія Scrum (рис. 1).

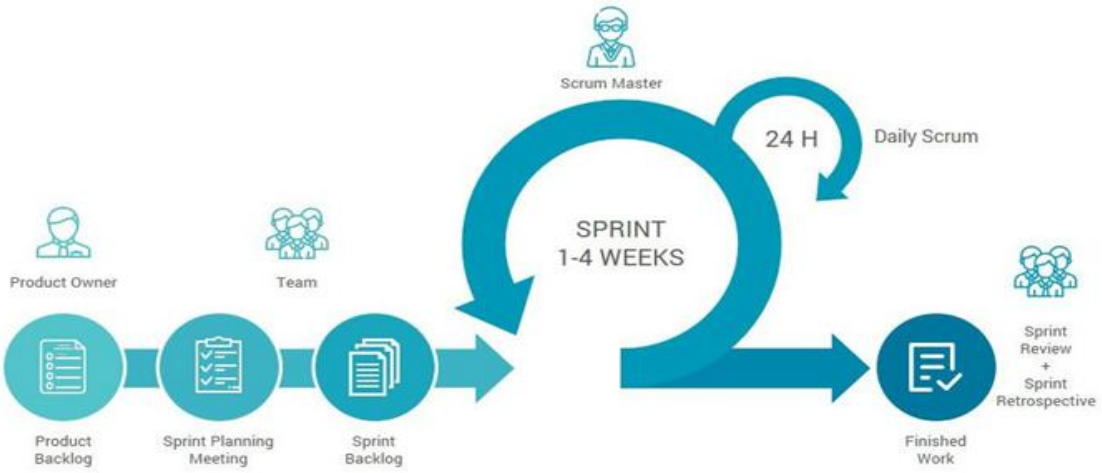


Рисунок 1. Цикл розробки ПЗ за методологією Scrum

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи — Scrum Master, той хто опікується процесами, веде їх і працює як модератор проєкту, Product Owner — людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Scrum Team, яка включає розробників.

У нашому випадку Product Owners були вчителі, ScrumMaster був Романенко Євгеній. Спринт тривав 1 тиждень, демо було кожну середу. У четвер проводилися планування спринта та ретроспектива. Стендап відбувався 2-4 рази на тиждень, у платформі Google Meets. Поточні задачі та беклог знаходилися у Jira. Завдання створювалися у скоупі 4-х епіків: Документація/Планування, Бекенд сервер, Фронтенд інтерфейс, Мобільний додаток. Кожне з завдань відноситься до одного з них.

Використання гнучкої методології розробки найбільш точно підійшло під час виконання групового проєкту. Під час роботи вимоги до продукту змінювалися, і використання Scrum дозволило нам швидко підлаштовуватися під ситуацію. У спринт йшли задачі з найвищим пріоритетом. Наявність спільної дошки з завданнями спрощувало комунікацію, завжди було зрозуміло хто над якими задачами працює в даний момент. Також задачі додавалися у спринт на основі фідбеку від викладачів. Також налаштована інтеграція Jira з Github, що дає змогу бачити у описі задачі пов’язані з нею зміни в коді.

Загалом усі члени команди отримали навички роботи у команді, які дуже цінуються на справжній роботі.

ТЕХНІЧНІ ВИМОГИ

Формулювання вимог реалізації. Веб клієнт та мобільний додаток повинні забезпечувати реалізацію наступних завдань:

- можливість створювати групи;

- можливість створювати пости (завдання) у групах;
- можливість залишати коментарі у постах;
- можливість залишати відповіді на завдання у постах;
- можливість бачити користувачів, які подивилися пост;

Веб клієнт повинен мати архітектуру, побудовану на сучасних технологіях зберігання, обробки, аналізу даних та доступу до них; забезпечувати одночасну роботу декількох користувачів. Рішення щодо побудови веб клієнта повинні базуватися на:

- застосуванні сучасних інформаційних технологій;
- застосуванні правила централізованого накопичення, зберігання та обробки інформації;
- підтримці актуальності, повноти, цілісності та доступності інформації;
- використанні сучасних засобів програмної інженерії при розробці програмного прикладного забезпечення.

Вимоги до стилістичного оформлення веб клієнта та мобільного додатка. Стилістичне оформлення веб-клієнта та мобільного додатка відповідають дизайн-макету, погодженого усіма розробниками. Використання колірних схем, графічних елементів, шрифтів погоджено в процесі створення системи. Інтерфейс є зручним у навігації, інтуїтивно зрозумілим, більшість ключових елементів є доступними через дві-три прості дії.

Дизайн повністю є адаптивним: склад, розміри і взаємне розташування елементів всіх шаблонів динамічно змінюються в залежності від розміру вікна браузера/розміру екрану телефона, в якому відображається інформація.

Вимоги до верстки і програмування. Система повинна бути оптимізованою і зверстаною у відповідності зі стандартами W3C для роботи в різних стабільних версіях браузерів (без помилок). Для правильного відображення інтерфейсу Системи на різних пристроях і браузерах повинна бути коректна, адаптивна верстка. Адаптивність верстки повинна бути забезпечена за допомогою CSS медіа запитів "@Media", або ж за допомогою використання бібліотек (наприклад Bootstrap), які надають можливість реалізовувати адаптивну верстку на більш високому рівні абстракції. Всі стилі потрібно максимально прибрати в зовнішні файли CSS, включаючи зображення, що відносяться до дизайну. Проєкт має мати добре структуровану, масштабовану архітектуру. Всі TS-компоненти повинні бути оптимізовані на максимальну продуктивність. Верстка сторінок Системи повинна бути виконана з урахуванням максимальної продуктивності Системи: код чистий без помилок і без зайвих тегів, мобільний додаток має бути адаптованим для роботи як на великих, так і на малих дисплеях. При розробці мобільного додатка має бути проведена оптимізація задля зменшення використання мобільного трафіку та операційної пам’яті телефону.

Вимоги до дизайну. Система повинна містити інформацію про права інтелектуальної власності, яка:

- розташована внизу на кожній сторінці із вмістом та помітна;
- стверджує, що матеріали системи є об’єктами права інтелектуальної власності, яке захищається відповідно до законодавства;
- Типографія системи повинна відповідати наступним вимогам:
- шрифт основного тексту − не менш як 16 пікселів;
- елементи інтерфейсу розміщуються на сторінці веб-клієнта через елементи div.
- поля вводу інформації повинні бути підписаними, мати короткі, інформативні та однозначні назви, які розташовуються над полем;

- помилки введення повинні виявлятися автоматично і, якщо відомо, як їх виправити, то користувачеві повинні надаватися підказки щодо їх виправлення.

Вимоги до вікна веб клієнта та мобільного додатка. Система повинна забезпечувати коректне відображення даних в наступних браузерах актуальних версій:

- Google Chrome;
- Mozilla Firefox;
- Safari;

Мобільний додаток мають коректно відображатись та функціонувати на ОС Android версії 4 і вище.

СТРУКТУРА ПРОЄКТУ

Проєкт складається з веб сайта, мобільного додатка та сервера, який надає можливість взаємодіяти клієнтам між собою за допомогою запитів через базу даних. Мобільний клієнт розроблявся для найбільш поширеної мобільної операційної системи Android. Він працює та функціонує на версії 4 і вище. Веб клієнт забезпечує коректне відображення даних в таких браузерах як Google Chrome, Mozilla Firefox, Safari.

Сервер - частина проєкту, з якою взаємодіє і веб-інтерфейс, і мобільний додаток. В свою чергу сервер взаємодіє з базою даних. Тут відбуваються усі маніпуляції з даними. Список усіх доступних методів можна знайти за посиланням (https://grouper-8team-api.herokuapp.com/swagger/index.html)

Рисунок 2. Swagger

Діаграма класів сервера:

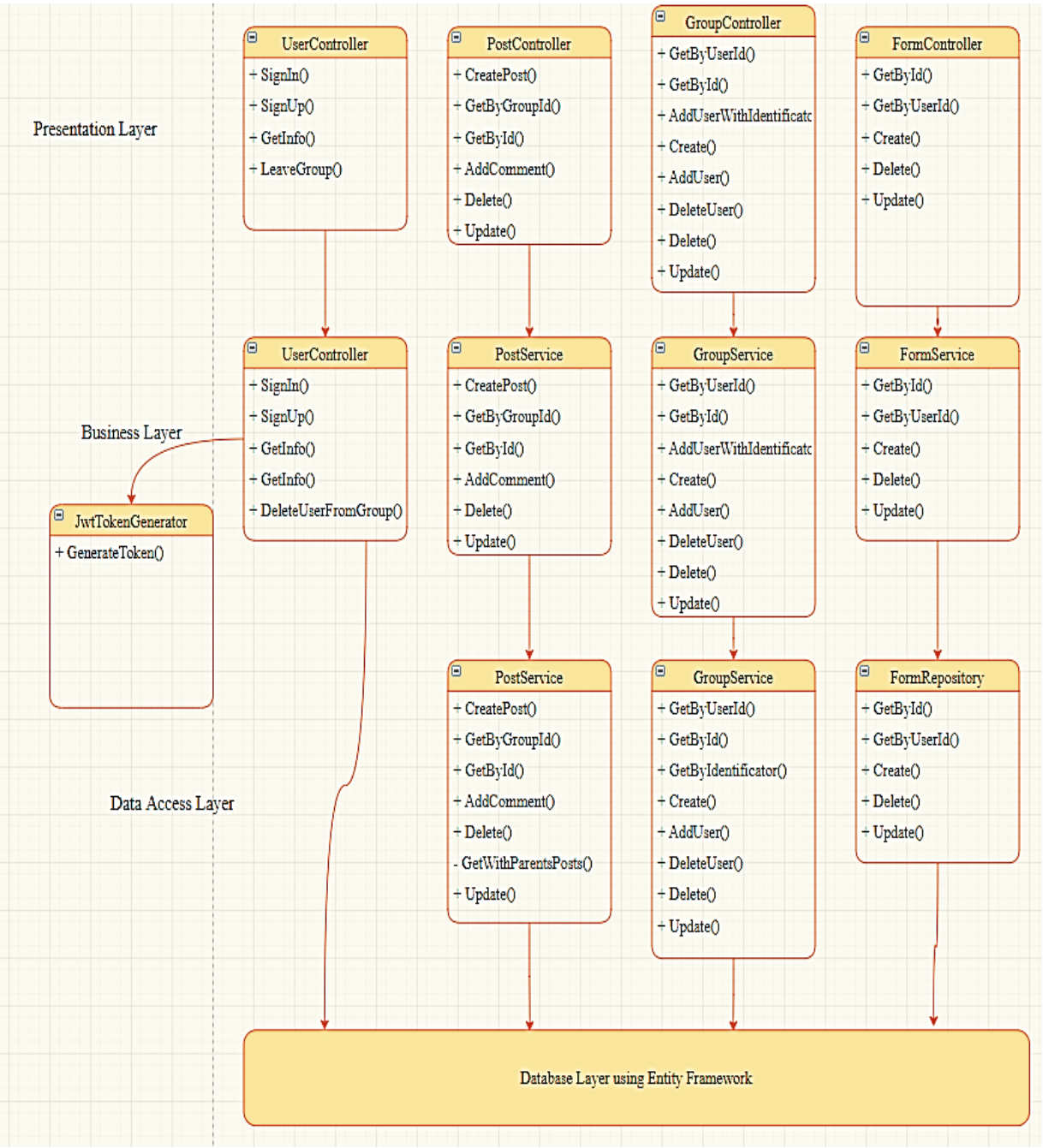


Рисунок 3. Діаграма класів сервера

Також на кожному рівні реалізовані свої DTO (Data Transfer Object), для того щоб на кожному рівні оперувати спеціальними моделями. Ці моделі відповідають сутностям бази даних. Наведемо також діаграму бази даних:

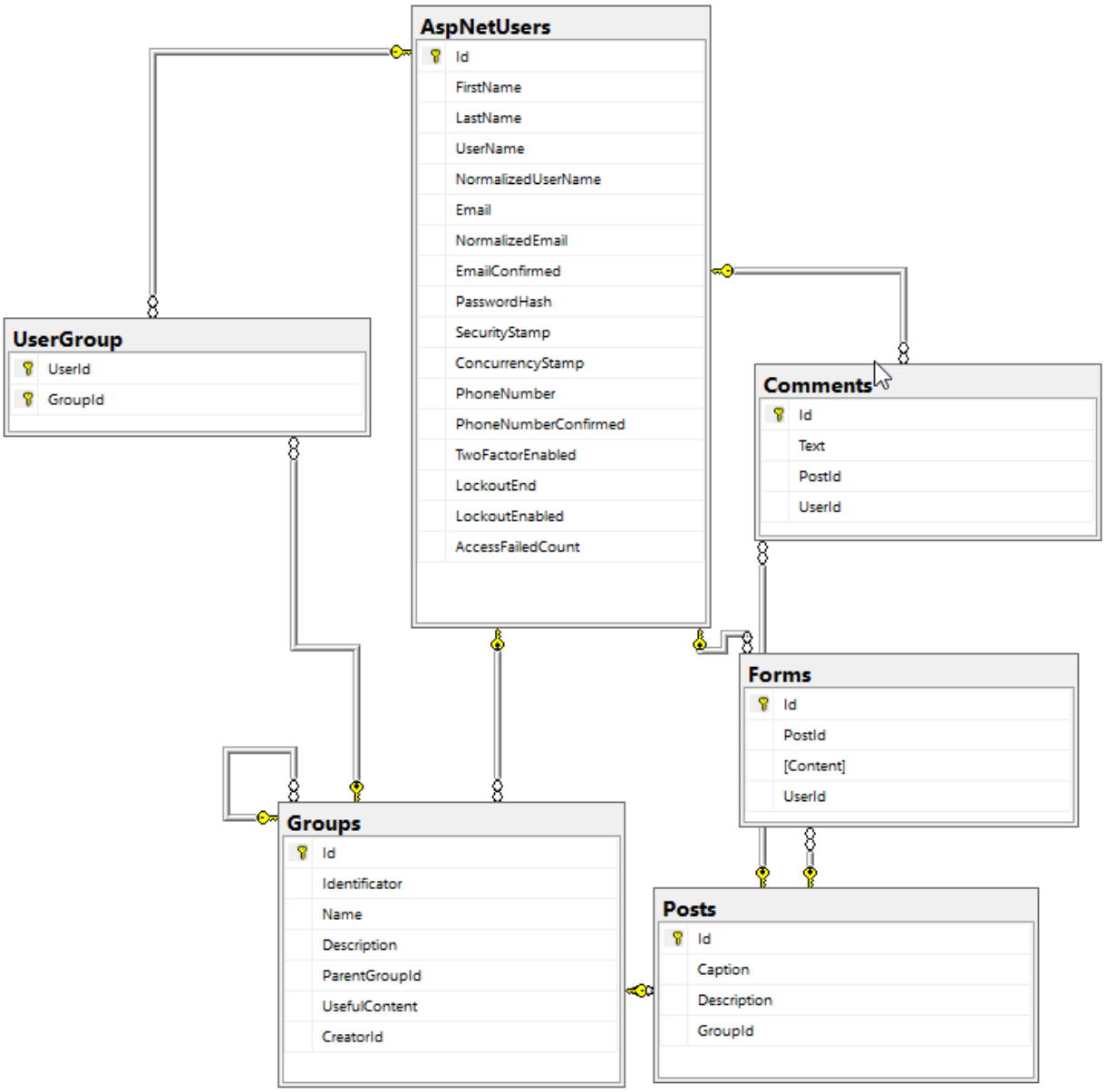


Рисунок 4. Схема даних

Деталі реалізації.

- Сервер написаний на мові C# (9 версія), на платформі .NET (.NET5 - найновіша версія), з використанням фреймворку ASP.NET.
- Для роботи з базою даних використовується Entity Framework.
- Авторизація відбувається на основі токену. В залежності від ролі користувач має або не має доступу до методів.
- Для передачі моделей між рівнями використовується Automapper, який дозволяє швидко з об’єкта одного класу отримати об’єкт іншого класу.
- Використовується патерн проєктування UnitOfWork для роботи з базою даних. Створено окремий репозиторій для кожної сутності.
- Як система контроля версій використовується Git. Віддалене сховище - GitHub.

- Налаштований віддалений сервер на основі heroku.com, за допомогою Docker.

Налаштований CI/CD процес. На основі Github Actions налаштований процес інтеграції та деплойменту. Після коміта в основну гілку (Develop) відбувається запуск збірки та тестування проєкту. Якщо збірка та тестування виконались успішно, запускається деплоймент на віддалений сервер grouper-8team-api.herokuapp.com.

МОБІЛЬНИЙ КЛІЄНТ ТА ВЕБ-КЛІЄНТИ

Після встановлення та запуску мобільного додатку користувач бачить сторінку логіна (рис. 5), де він зобов'язаний ввести свою пошту та пароль, з якими він був зареєстрований в системі. Якщо ж користувач ще не зареєстрований, то він повинен перейти на відповідну сторінку, ввести своє прізвище та ім'я, пошту, пароль і натиснути кнопку **Реєстрація**. Додаток відправить запит реєстрації на сервер і, за умови його успішного виконання, в систему буде додано нового користувача, а в мобільному клієнті відкриється сторінка логіна. При помилковому відкритті сторінки реєстрації є можливість повернутися на сторінку логіна.

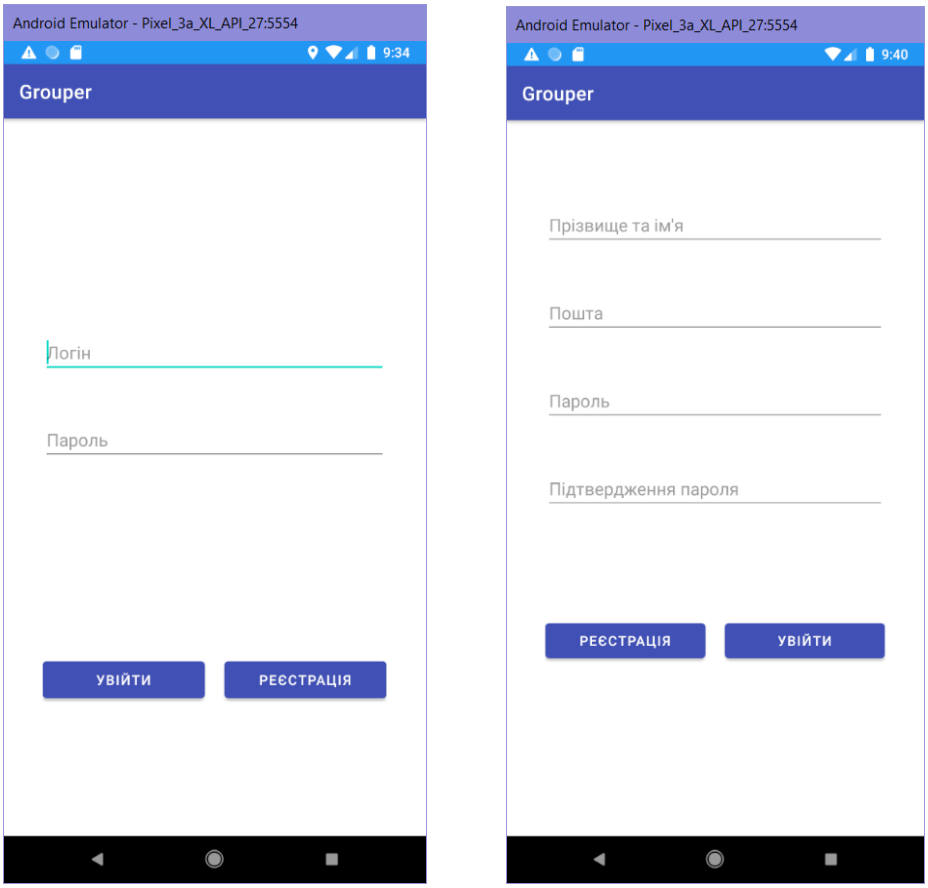


Рисунок 5. Форма для логіна (зліва) та форма для реєстрації (справа)

Після успішного входу в систему мобільний додаток зберігає поточного користувача за допомогою SharedPreferences. Якщо користувач вимкне додаток, а потім знову його відкриє, то спочатку буде перевірено наявність користувача в SharedPreferences і, якщо він

є, то буде здійснений запит входу в систему на сервер, інакше користувачу відкриється сторінка логіна. Взаємодія із SharedPreferences реалізована за допомогою класу SharedPreferencesHelper.

Після сторінки логіна користувач потрапляє на сторінку груп, в які він доданий (рис. 6). На цій сторінці у вигляді горизонтального списку кнопок розміщуються групи, перейти в які можна натиснувши на відповідну кнопку. Зовнішній вигляд власне сторінки для студента та викладача не відрізняється.

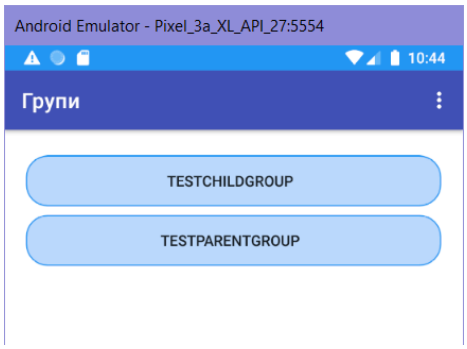


Рисунок 6. Сторінка всіх груп

Відмінності наявні лише у функціоналі меню (рис. 7).

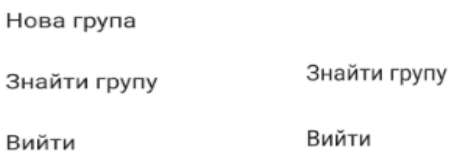


Рисунок 7. Меню груп вчителя (ліворуч) та груп студента

Вчитель може створити нову групу (елемент **Нова група**) та вступити в існуючу (елемент **Знайти групу**). Студент може тільки вступити в існуючу групу (елемент **Знайти групу**). Елемент **Вийти** присутній на всіх меню додатку, він видаляє користувача з SharedPreferences та завершує поточну сесію з сервером.

Елемент **Нова група** відкриває сторінку створення групи (рис. 8).

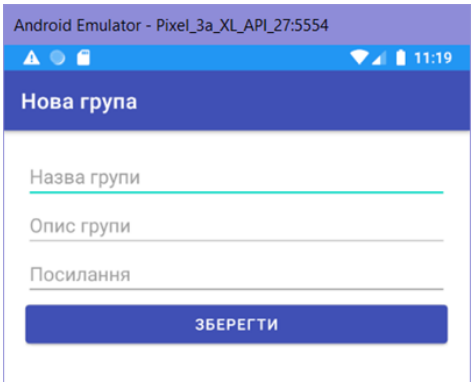


Рисунок 8. Сторінка створення групи

Викладач повинен ввести у відповідні поля назву, опис групи, посилання на корисні матеріали, Zoom, Google Meets. Обов'язковим є тільки поле **Назва групи**. Після заповнення інформації викладач повинен натиснути кнопку **Зберегти**, додаток відправить запит збереження групи на сервер. Якщо запит виконався успішно, викладач повернеться на оновлену сторінку груп з наявною щойно доданою групою. Інакше буде виведено повідомлення про помилку і викладач залишиться на сторінці створення групи.

Елемент **Знайти групу** відкриває сторінку (рис. 9) з полем вводу ідентифікатора групи, до якої хоче вступити користувач. Відповідно щоб приєднатись до групи треба просто ввести її ідентифікатор та натиснути кнопку **Вступити в групу**.

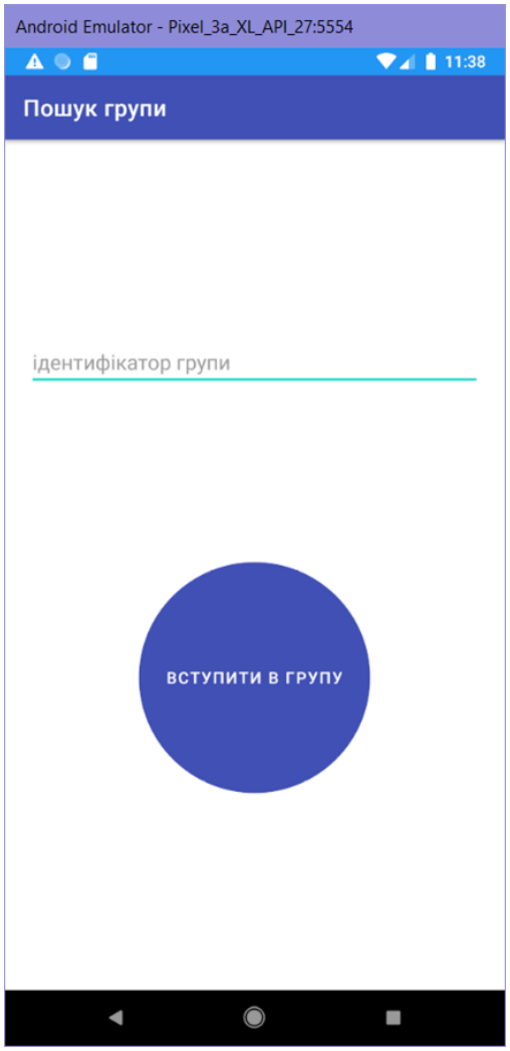


Рисунок 9. Сторінка пошуку групи

Сторінка конкретної групи (рис. 10) містить список всіх постів, створених вчителями. Перейти у відповідний пост можна натиснувши на нього. Сторінка групи спільна як для викладача, так і для студента.

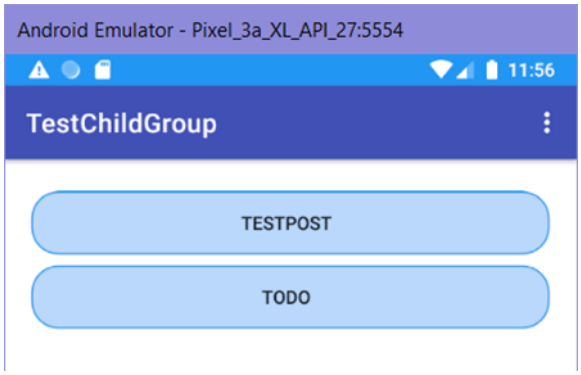


Рисунок 10. Сторінка групи

Залежно від ролі поточного користувача змінюється функціональність меню сторінки (рис. 11). Студенту доступні тільки елементи **Опис групи** та **Вийти**.

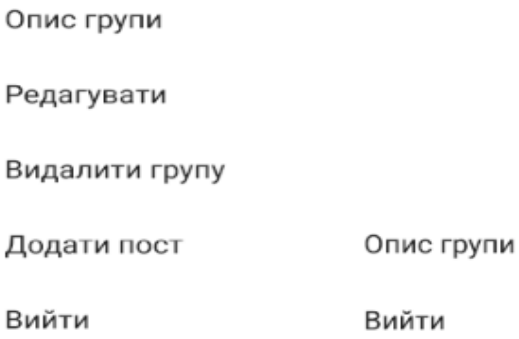


Рисунок 11. Меню групи викладача (ліворуч) та студента

Елемент **Опис групи** спільний для користувачів обох ролей та відкриває одну й ту саму сторінку (рис 12). У відповідних віджетах TextView міститься інформація про назву групи, її ідентифікатор (можна копіювати), опис, корисні посилання (всі коректні посилання клікабельні та відкривають відповідну сторінку у браузері за замовченням) та учасників.

Елемент **Змінити опис** доступний тільки для викладача і відкриває сторінку (рис. 13), із заповненими відповідними полями групи, готовими до редагування. Проте, викладач не може редагувати ідентифікатор групи, так як він генерується автоматично на сервері при створенні групи. На цій сторінці можна також видаляти студентів з групи, просто знявши галочку біля відповідного імені. Для збереження змін потрібно натиснути кнопку **Зберегти**. Якщо запит виконався успішно, викладач повернеться на оновлену сторінку групи. Інакше буде виведено повідомлення про помилку і викладач залишиться на сторінці редагування групи.

Рисунок 12. Сторінка опису групи

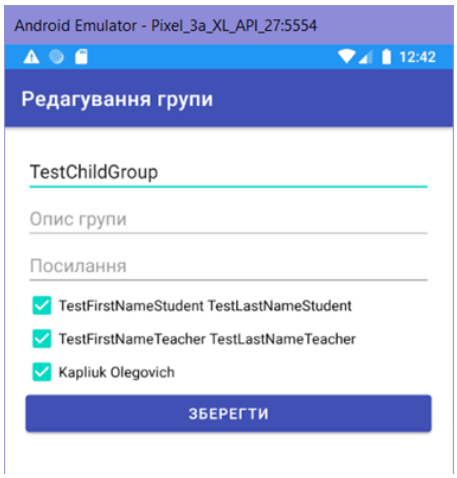


Рисунок 13. Сторінка редагування опису групи

Елемент **Видалити групу** надсилає запит видалення поточної групи на сервер. Якщо запит виконався успішно, для викладача відкриється оновлена сторінка всіх груп, без щойно видаленої. Інакше буде виведено повідомлення про помилку і викладач залишиться на сторінці групи.

Елемент **Додати пост** відкриває нову сторінку (рис. 14) для створення посту. На сторінці наявні 2 поля для вводу завдання (назва посту) та опис (що треба зробити). Для збереження треба натиснути кнопку **Зберегти**, яка посилає запит створення нового посту на сервер. Якщо запит був виконаний успішно, то викладачу відкриється оновлена сторінка групи з наявним щойно доданим постом. Інакше буде виведено повідомлення про помилку і викладач залишиться на сторінці групи.

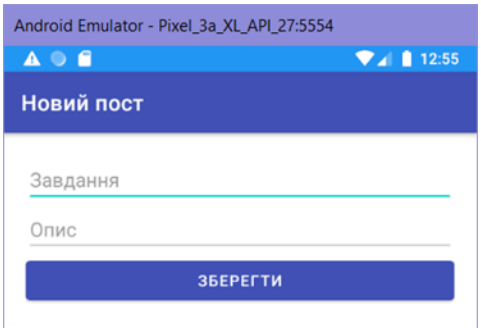


Рисунок 14. Сторінка створення посту

Сторінка поста (рис. 15) містить назву поста (у вигляді кнопки), опис (текст змінює характеристику видимості при натисканні на вищезгадану кнопку), поле для відображення та поле додавання коментарів. Коментарі поточного користувача здвигаються вправо та не містять прізвища відправника. Всі інші коментарі розміщені зліва та містять прізвище користувача, який його відправив.

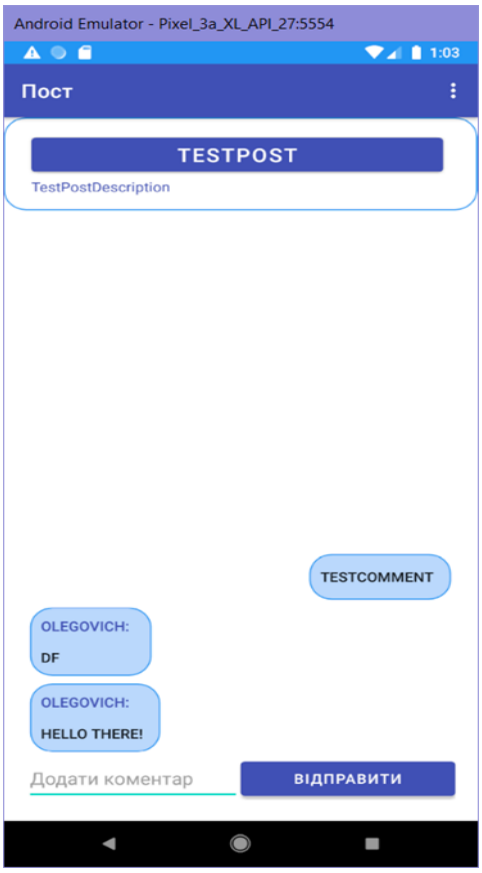


Рисунок 15 Сторінка поста

Сторінка поста спільна для викладача та студента. Залежно від ролі поточного користувача змінюється функціональність меню сторінки (рис 16).

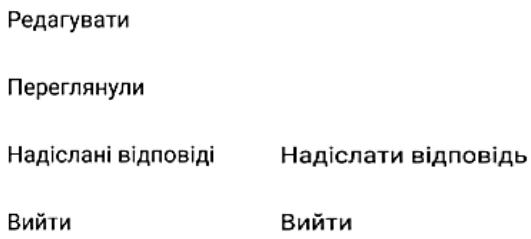


Рисунок 16. Меню поста викладача (ліворуч) та студента

Елемент **Редагувати** відкриває сторінку редагування поста (рис. 17) з заповненими відповідними полями готовими до редагування. Для збереження відредагованого поста потрібно натиснути кнопку **Зберегти**, яка посилаю запит зміни існуючого поста на сервер. Якщо запит був виконаний успішно, то викладачу відкриється оновлена сторінка поста. Інакше буде виведено повідомлення про помилку і викладач залишиться на сторінці редагування поста.

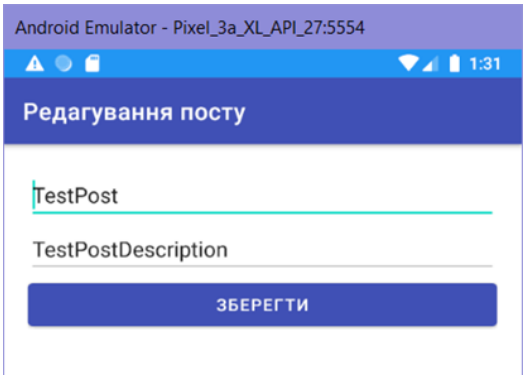


Рисунок 17 Сторінка редагування поста

Елемент **Переглянули** відкриває сторінку (рис. 18) з двома списками студентів (з групи поста): які переглядали даний пост та які не переглядали його.

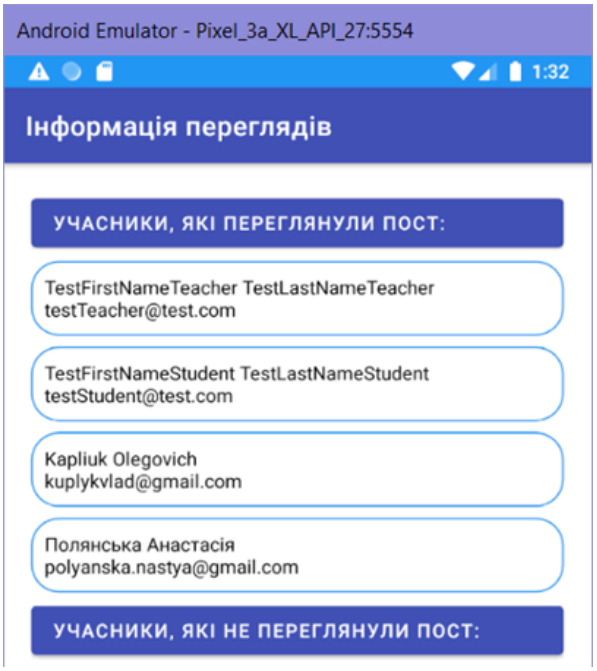


Рисунок 18. Сторінка з інформацією про студентів, які переглядали пост або ні

Елемент **Надіслані відповіді** відкриває сторінку із списком всіх надісланих студентами форм для даного поста (рис. 19). Вчитель може натиснути на відповідь конкретного студента і текст відповіді стане видимим, а активна форма стане зеленою. Текст підтримує посилання, якими можна переходити.

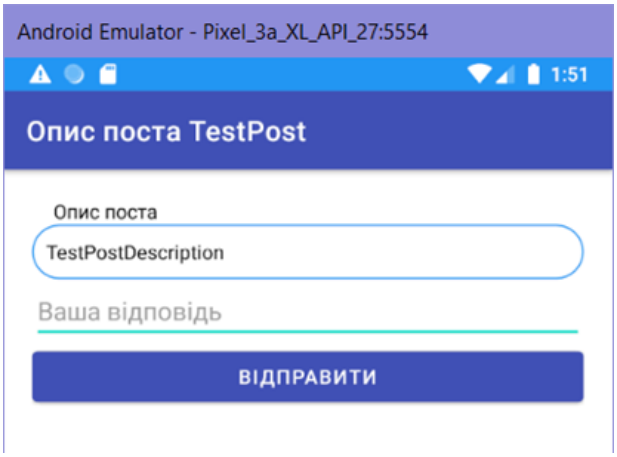


Рисунок 19. Сторінка надісланих відповідей

Елемент **Надіслати відповідь** відкриває сторінку створення форми (рис. 20). Сторінка дублює назву поста та опис, а також містить поле для вводу відповіді на завдання. Для збереження форми треба натиснути кнопку **Відправити**, яка відправляє запит створення форми на сервер. Якщо запит був виконаний успішно, то студенту відкриється сторінка поста та буде виведено повідомлення про успішне збереження відповіді. Інакше буде виведено повідомлення про помилку і студент залишиться на сторінці створення форми.

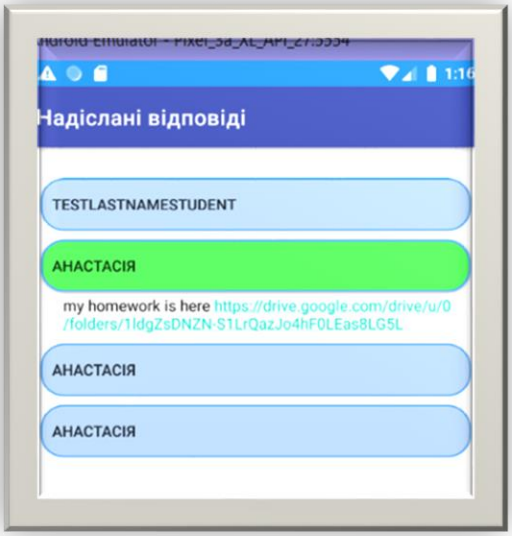


Рисунок 20. Сторінка створення форми

Веб-клієнт розроблено з використанням фреймворку Angular 11. Ключовою перевагою фреймворку є розбиття на компоненти з ізольованим стилями, кодом, та HTML шаблонами. Стилі проєкту побудовано на базі Angular Material.

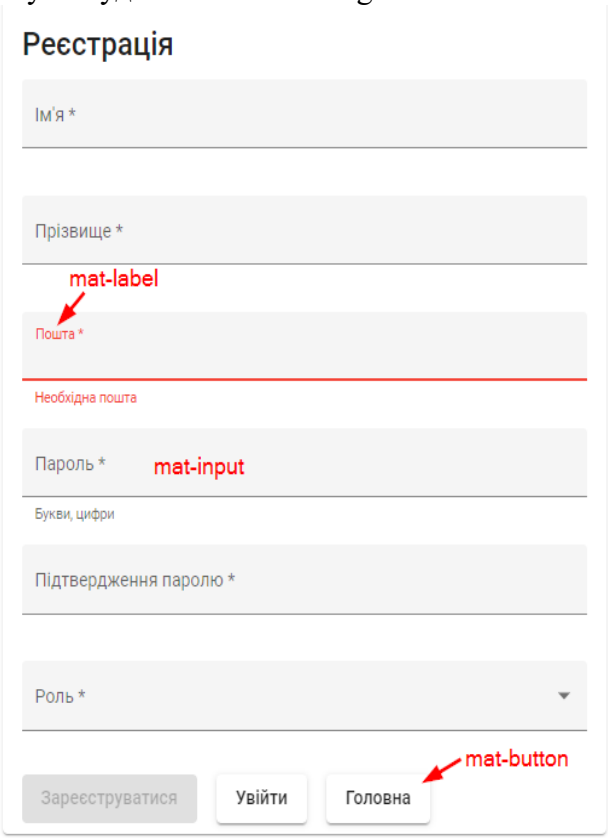


Рисунок 21. Приклад Angular Material компонент

ВЕРИФІКАЦІЯ

Для верифікації програми використовуються Unit тести. Під час тестів також застосовуються такі технології як Mock. Тести запускаються при кожному коміті в основну гілку. Результат виконання тестів можна бачити у Github.

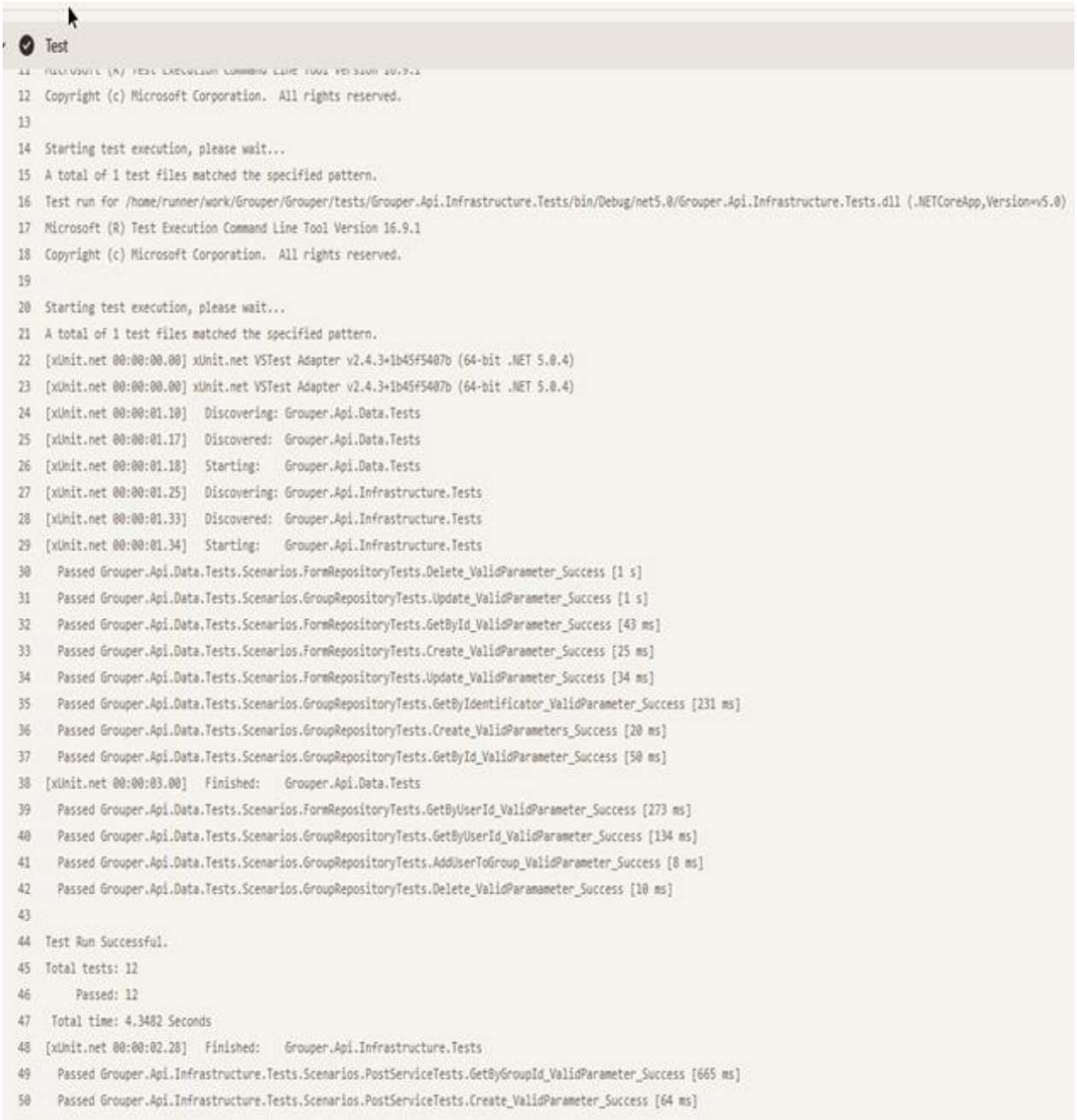


Рисунок 22. Результат виконання тестів

Наявність гарних тестів підтверджує відповідність вимогам, а також дозволяє більш безпечно оновлювати код проєкту.

Специфікація. Верифікація програмного коду реалізована за допомогою бібліотеки CodeContracts, яка дозволяє визначити передумови, післяумови та інваріанти. Приклади верифікації серверної частини проєкту:

```
[ContractClassFor(typeof(IGroupService))]
  internal abstract class IGroupServiceContract : IGroupService
  {
    public Task AddUser(int groupId, string userEmail)
    {
      Contract.Requires<ApiException>(groupId > 0, $"{nameof(groupId)} less than 0");
      Contract.Requires<ApiException>(!string.IsNullOrEmpty(userEmail),
        $"{nameof(userEmail)} is null or empty");

      return default;
    }
    public Task AddUserWithIdentificator(string identificator, string userId)
    {
      Contract.Requires<ApiException>(!string.IsNullOrEmpty(identificator),
        $"{nameof(identificator)} is null or empty");
      Contract.Requires<ApiException>(!string.IsNullOrEmpty(userId),
        $"{nameof(userId)} is null or empty");

      return default;
    }
    public Task<int> Create(GroupDto groupDto)
    {
      Contract.Requires<ApiException>(groupDto is not null,
        $"{nameof(groupDto)} is null");

      Contract.Ensures(Contract.Result<int>() > 0);

      return default;
    }

    public Task Delete(int id)
    {
      Contract.Requires<ApiException>(id > 0, $"{nameof(id)} less than 0");

      return default;
    }
     public Task DeleteUsers(int groupId, List<string> userEmails)
     {
      Contract.Requires<ApiException>(groupId > 0, $"{nameof(groupId)} less than 0");
      Contract.Requires(Contract.ForAll(userEmails, userEmail => !string.IsNullOrEmpty(userEmail)));

      return default;
    }

    public Task<GroupDto> GetById(int id)
```

```
        {
            Contract.Requires<ApiException>(id > 0, $"{nameof(id)} less than 0");
            Contract.Ensures(Contract.Result<GroupDto>().Id > 0);

            return default;
        }

        public Task<List<GroupDto>> GetByUserId(string userId)
        {
            Contract.Requires(!string.IsNullOrEmpty(userId));
            Contract.Ensures(Contract.ForAll(Contract.Result<List<GroupDto>>(), group => group.Id > 0));

            return default;
        }

        public Task Update(GroupDto groupDto)
        {
            Contract.Requires(groupDto.Id > 0);

            return default;
        }
    }

[ContractClassFor(typeof(IUserService))]
    class IUserServiceContract : IUserService
    {
        public Task<UserDto> GetInfo(string id)
        {
            Contract.Requires(!string.IsNullOrEmpty(id));
            Contract.Ensures(!string.IsNullOrEmpty(Contract.Result<UserDto>()?.Id));

            return default;
        }

        public Task<string> SignIn(UserDto userDto)
        {
            Contract.Requires(!string.IsNullOrEmpty(userDto.Email) && !string.IsNullOrEmpty(userDto.Password),
"email or password is null");

            return default;
        }

        public Task SignUp(UserDto userDto)
        {
            Contract.Requires(!string.IsNullOrEmpty(userDto.Email) && !string.IsNullOrEmpty(userDto.Password));

            return default;
        }
    }
```

"ПРОГРАМУВАННЯ: ТЕОРІЯ ТА ПРАКТИКА"

ЗБІРНИК МАТЕРІАЛІВ
ЗА РЕЗУЛЬТАТАМИ ІТ-ПРОЄКТУ
МІЖДИСЦИПЛІНАРНОЇ ІНТЕГРАЦІЇ
2020-2021 навчальний рік

До 50-річчя кафедри
теорії та технології програмування
факультету комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка

Підписано до друку 18.06.2021 р.
Формат 64x90/8.
Папір офс. Друк офс. Ум. друк арк. 18,6
Гарнітура Times New Roman. Наклад: 300 прим.,
Замовлення № 18/06/21

Виготовлювач: Типографія «Айс Принт»
Свідоцтво серія ДК № 3534 від 24.07.2009 р.
Тел: +38 (099) 192-00-33, +38 (048) 706-92-82
Site: www.ice-print.com.ua
E-mail:info@ice-print.com.ua