Факультет комп’ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
Кафедра теорії та технології програмування

О.В.Шишацька
А.В.Криволап
А.В.Шишацький

ОСНОВИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ
ПРОЄКТНА РОБОТА: ІНІЦІАЦІЯ ТА ПЛАНУВАННЯ

Київ 2025

УДК 004:005.8(075.8)

Рецензенти

к.ф.-м.н., доцент Шкільняк О.С. (доцент кафедри факультету комп’ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка)

к.ф-м.н., доцент Ткаченко О.М. (доцент кафедри теорії та технології програмування Київського національного університету імені Тараса Шевченка)

Рекомендовано до друку вченою радою факультету комп’ютерних наук та кібернетики (протокол № 14 від 20 травня 2025 року)

Ухвалено науково-методичною комісією факультету комп’ютерних наук та кібернетики (протокол № 9 від 19 травня 2025 року)

к.ф.-м. наук, доцент Шишацька Олена Володимирівна
к.ф.-м. наук, доцент Криволап Андрій Володимирович
асистент Шишацький Андрій Вікторович

Основи управління ІТ-проєктами. Проєктна робота: ініціація та планування / О.В.Шишацька, А.В.Криволап, А.В.Шишацький. – Київ: 2025

Викладено матеріали для виконання проєктної роботи (самостійна робота) вибіркової навчальної дисципліни «Основи управління ІТ-проєктами» вибіркового блоку «Теорія та технологія програмування». У методичних вказівках наведено завдання проєктної роботи, теоретичний та практичний матеріал для їх виконання.

Для студентів четвертого курсу факультету комп’ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка, що навчаються за освітньо-професійною програмою «Інформатика» спеціальності 122 «Комп’ютерні науки».

ЗМІСТ

ПЕРЕДМОВА

Управління ІТ-проєктами – це міждисциплінарна область знань, яка поєднує принципи менеджменту, інженерії, комунікацій та аналітики для ефективного планування, реалізації та контролю ІТ-продуктів. У контексті сучасного цифрового середовища вміння управляти ІТ-проєктами є надзвичайно важливим як для спеціалістів технічних професій, так і для менеджерів, аналітиків і дизайнерів.

Метою дисципліни «Основи управління ІТ-проєктами» є ознайомлення студентів із базовими поняттями, інструментами та підходами до організації командної роботи над створенням ІТ-продуктів. Вивчення цієї дисципліни дозволяє здобути знання та практичні навички, які необхідні для запуску, ведення та завершення проєктів у сфері ІТ, зокрема в умовах обмежених ресурсів, швидких змін вимог та високої конкуренції.

Ключові поняття дисципліни включають:

- ІТ-проєкт – це комплекс взаємопов’язаних дій, які мають на меті створення унікального інформаційного продукту або послуги.
- Проєктний менеджмент – це застосування знань, навичок, інструментів і методів до проєктної діяльності з метою досягнення проєктних цілей.
- Життєвий цикл проєкту – це послідовність етапів, через які проходить проєкт: ініціація, планування, реалізація, контроль та завершення.
- Методології управління – це структуровані підходи до організації роботи: водоспадна модель (Waterfall), гнучкі методології (Agile, Scrum, Kanban), гібридні моделі.
- Команда проєкту – це група фахівців різного профілю, об’єднаних спільною метою, з чітким розподілом ролей і зон відповідальності.

- Проєктна документація – це набір артефактів, які супроводжують проєкт: цілі, вимоги, план, бюджет, ризики, статусні звіти тощо.

Окрему увагу приділено таким важливим аспектам, як постановка цілей (SMART, OKR), розподіл ролей у команді (RACI-матриця), аналіз зацікавлених сторін (stakeholders analysis), оцінка ризиків та формування комунікаційної стратегії.

Практична частина курсу полягає у виконанні командного проєкту – розробки ІТ-продукту на основі отриманих знань. Студенти працюють у складі кросфункціональних команд, обирають методологію, формують артефакти, застосовують інструменти (Jira, Trello, Miro, Google Workspace тощо) і презентують фінальний результат.

Значна увага приділяється розвитку soft skills: критичне мислення, командна взаємодія, відповідальність, здатність до самоорганізації й адаптації до змін. Завдяки цьому студенти не лише отримують знання про проєктний менеджмент, а й формують компетенції, що є необхідними для роботи в реальному ІТ-середовищі.

У методичних вказівках наведено завдання та матеріали щодо його виконання етапу ініціації та планування проєкту.

Використано матеріали курсів «Створення та розвиток ІТ-продуктів» та «Менеджмент у продуктовому ІТ» від ІТ-компанії Genesis.

Перелік практичних завдань проєктної роботи з курсу "Основи управління ІТ-проєктами" допоможе командам пройти повний цикл роботи над ІТ-продуктом, використовуючи обрану методологію (Scrum, Kanban, Waterfall тощо). Мета – створення MVP або іншого релевантного результату, максимально наближеного до реального продукту.

ПЕРЕЛІК ЕТАПІВ, ЗАВДАНЬ, ЇХ РЕЗУЛЬТАТІВ ТА ТЕРМІНІВ ВИКОНАННЯ

ЕТАП РОЗРОБКИ (блок практичних завдань)	ОЧІКУВАНІ ЗНАННЯ	ПРАКТИЧНІ НАВИЧКИ	КРИТЕРІЇ ОЦІНЮВАННЯ
Формування команди	Поняття ролей у команді, основи командної взаємодії, принципи ефективної комунікації	Розподіл ролей, організація зустрічей, фіксація домовленостей, використання каналів комунікації	Чіткість розподілу ролей; якість командної взаємодії; заповнені таблиці зустрічей
Ініціація проєкту	Життєвий цикл проєкту, структура статуту проєкту, техніки генерації ідей (брейнштормінг, SWOT)	Пошук ідеї, написання статуту, формулювання місії й мети, аналіз конкурентів	Обґрунтованість ідеї; повнота SWOT-аналізу; якість статуту та опису продукту
Планування проєкту	Планування Roadmap, методики управління (Scrum, Kanban, Waterfall), основи документації	Створення Roadmap, написання backlog, вибір моделі документації, тайм-менеджмент	Наявність логічного плану; використання відповідної методології; ведення документації

ЕТАП РОЗРОБКИ (блок практичних завдань)	ЗАВДАННЯ	РЕЗУЛЬТАТ	ГОТОВНІСТЬ
І. Формування команди	Визначити вид ролей, характерний особистості (індивідуальне завдання)	Визначено вид ролей характерний особистості (пройдено тест Бельбіна)	Другий тиждень навчального семестру
	Познайомитися в команді	Виконана «вправа на знайомство»	
	Обрати ролі та узгодити в команді (PM, дизайнер, розробник, аналітик, QA)	Попередньо обрано ролі (див. Таблиця 1. Карти ролей проєкту) та узгоджено в команді	
	Узгодити канали комунікації	Узгоджено і зафіксовано канали комунікації в команді	

II. Ініціація проєкту	Провести зустріч-брейнштормінг щодо пошуку ідеї продукту	1. Заповнена Таблиця 2. Документ для фіксації і перевірки виконання командного проєкту 2. Створена презентація ідеї продукту 3. Ідея презентована викладачам	Четвертий тиждень навчального семестру
	Сформулювати місію та цілі проєкту	Сформовано місію та цілі проєкту	
	Створити опис продукту	Заповнено таблицю опису розроблюваного продукту (Таблиця 3)	
	Оглянути конкурентів (продуктів, що мають схожий функціонал з розроблюваним)	Створено опис та порівняльний аналіз конкурентів (продуктів, що мають схожий функціонал з розроблюваним) (Таблиця). Створено презентацію, презентовано викладачам.	
	Скласти рамку Статуту проєкту	Заповнено таблиці	
III. Планування	Побудувати Roadmap	Створено Roadmap (формат довільний формат). Презентовано викладачам.	П’ятий-шостий тиждень навчального семестру
	Визначити методологію управління (Scrum, Kanban, Waterfall тощо)	Визначено методологію управління проєктом	
	Створити беглог продукту	Створено, узгоджено та затверджено беглог продукту. Зафіксовано як документ.	
	Визначитися з мінімумом для документування, моделлю документування	1. Узгоджено мінімальний перелік артефактів проєкту, інструменти організації та ведення документації 2. Обрано модель документування проєкту	

БЛОК І. ФОРМУВАННЯ КОМАНДИ

ПРАКТИЧНІ ЗАВДАННЯ

1. Визначити види ролей (індивідуальне).
2. Організувати першу зустріч в команді. Познайомитися в команді. Розподілити ролі в команді, узгодити канали комунікації.

МАТЕРІАЛИ ДЛЯ САМОСТІЙНОГО ОПРАЦЮВАННЯ

1. Product Manager vs. Project Manager: What's the Difference?
2. Project Manager: хто такий PM, чим він займається, де навчатись професії та як знайти роботу
3. 10 Difference Between Software Project and Normal Project
4. Ролі в команді (Тест Белбіна)
5. Roles and Responsibilities of a Product Manager
6. Як ефективно управляти командою. Розповідає CEO EdTech компанії Headway Антон Павловський
7. Product Team: Role and Structure
8. Communicate Effectively as a Project Manager | Google Career Certificates
9. Як екологічно давати правки, аби отримати топпродукт і не демотивувати креатора – гайд Sasquatch Digital
10. Бесіда, а не допит. 28 питань, які варто поставити керівнику на зустрічі віч-на-віч
11. What to Do and What Not to Do During Meetings

МАТЕРІАЛИ ДЛЯ ВИКОНАННЯ ПРАКТИЧНИХ ЗАВДАНЬ

1. Пройти тест Белбіна за покликанням (індивідуальне завдання)
2. Організувати та провести першу зустріч в команді.

1.1. Перед проведенням зустрічі кожен студент опрацьовує матеріал:

- Презентація «Поняття про сферу інформаційних технологій, групи IT, професії IT».
- Таблиця 1. Карти ролей в команді.

1.2. Провести першу командну зустріч, користуючись покроковою інструкцією проведення командної зустрічі та документом для фіксації і перевірки роботи над командним завданням. Результатом виконання завдання є заповнений даний документ (Таблиця 2) та розподіл і опис обов'язків в команді (Таблиця 3).

Таблиця 1 – **Карти ролей в команді**

Категорія	Назва ролі
	Performance Marketing Manager
Опис ролі	Спеціаліст, що запускає та оптимізує рекламні кампанії у Facebook, Google та на інших платформах. Розробляє стратегію просування продукту та звітує про результати, які завжди можна виміряти в цифрах – охоплення рекламної публікації, ціна кліка на пост, кількість людей, що зробили цільову дію. Види маркетологів у продуктовому IT: Performance Marketing, Affiliate Marketing, E-mail & Push Marketing, SEO / ASO, Brand Marketing, Influence Marketing, Creative Marketing
Зони відповідальності	Перед запуском - дослідження ринку та визначення ЦА продукту, аналіз конкурентів для покращення метрик продукту Під час запуску - відповідальність за стратегію виходу на ринок, дорожня карта просування продукту, визначення ключових меседжів, контроль комунікації про продукт Після запуску - дослідження та реакція на фідбек юзерів, контроль метрик, додаткові кампанії з розвитку та вдосконалення продукту, маркетинг нових функцій
Hard Skills	- Аналітичне мислення та робота з інструментами аналітики - Основи статистики для правильного проведення A/B-тестів - Дослідження ринку та цільової аудиторії продукту - Визначення ключових меседжів - Робота з метриками (CPM, CPC, CPL, ROI, Retention тощо) - Базове розуміння дизайну
Soft Skills	- Сильні комунікаційні навички - Вирішення проблем та аналітика - Емпатія - Робота в команді (особливо в кросфункціональних) Вміння слухати та аналізувати
Інструменти	Аналіз ринку (SimilarWeb, Typeform, Google Analytics, UserTesting, SensorTower, Amplitude, Segment) Аналітика (Tableau, Microsoft Excel, Google Sheets, SQL) Проєктний менеджмент (Asana, Slack, Trello) Створення контенту (Figma, Sketch) Рекламні майданчики (Google, Facebook, Snapchat, TikTok)

Категорія	Назва ролі	
	Product Manager	**DevOps Engineer**
Опис ролі	Спеціаліст, що працює на стику бізнесу, маркетингу і технологій та синхронізує роботу всіх департаментів. Головна мета – створити прибутковий продукт, що відповідатиме очікуванням та потребам користувачів	Спеціаліст, основне завдання якого – автоматизувати процеси та створити єдине операційне середовище для команд
Зони відповідальності	- Дослідження ринку, конкурентів, потреб користувача - Управління беклогом - Планування роботи продуктової команди - Пріоритетування завдань - Планування релізу нових фіч - Розробка стратегії - Висування та перевірка гіпотез	- Автоматизація та оптимізація процесів на стороні розробки - Оцінка інфраструктури проєкту - Допомога в розгортанні продукту - Контроль продуктивності процесів і усунення багів - Створення єдиного операційного середовища для команд
Hard Skills	- Робота з аналітикою та базами даних - Розробка дорожньої карти продукту - Знання методологій управління командами - Дослідження ринку та цільової аудиторії - Базове розуміння процесу технічної розробки продукту - Проведення A/B-тестувань - Розробка та створення MVP - Базове розуміння коду -Розробка стратегії розвитку продукту - UX/UI-дизайн - Маркетинг продукту	- Знання основ топології сіток, протоколів TCP/IP (IP, TCP, UDP, HTTP/HTTPS) - Знання мов програмування: Python, JavaScript/TypeScript чи Go - Розуміння концепцій API (REST, gRPC, GraphQL) - Опанування CI/CD - Знання концепції Infrastructure-as-a-Code, систем моніторингу, агрегації логів - Розуміння баз даних, вебсерверів, проксі, балансувальників навантаження - Уміння працювати з операційною системою Linux - Розуміння життєвого циклу програми (SDLC) - Навички роботи з хмарними провайдерами
Soft Skills	- Вміння комунікувати, слухати, домовлятися, переконувати - Тактичне та стратегічне мислення - Командна робота - Прийняття рішень - Тайм-менеджмент - Емпатія - Helicopter view	- Ефективна комунікація - Критичне мислення - Командна робота - Вміння швидко адаптовуватись - Стресостійкість - Прийняття рішень
Інструменти	Figma, Notion, Google Analytics, Slack, Jira, Miro, Microsoft Excel, Asana, Tableau, Amplitude,Confluence	Python, Go, TypeScript, AWS (Amazon Web Services), Google Cloud, Microsoft Azure, New Relic, Linux, Automic, Jenkins, Git, Flyway, Terraform, Jira

Категорія	Назва ролі	
	Product Analyst	**Product Designer**
Опис ролі	Продуктовий аналітик займається пошуком інсайтів у даних про поведінку користувача. Ці знання допомагають бізнесу отримати більше прибутку та оптимізувати окупність затрат. Включає аналітику A/B тестів, змін у додатку, нових релізів застосунку тощо	UI/UX (або продактдизайнери) – фахівці, які проєктують рішення залежно від потреб бізнесу та проблем користувачів. Вони відповідають за те, як саме інтерфейс комунікує з юзерами, розробляють структуру застосунку, його зовнішній вигляд, кольори, шрифти та іконки
Зони відповідальності	- Вивчення метрик продукту, перевірка гіпотез - Проведення досліджень цільової аудиторії продукту - Збір, аналіз, інтерпретація та візуалізація даних - Аналіз поведінки користувачів - Пошук точок зростання продукту - Розвиток продукту для задоволення потреб користувачів - Дослідження ринку та конкурентний аналіз	- Розробка робочого прототипу продукту - Проведення якісних та кількісних користувацьких досліджень - Взаємодія з тестувальниками, внесення правок / змін за результатами - Створення user flow та user journey map - Створення портрету ЦА
Hard Skills	- Аналіз даних та інструменти продуктової аналітики - Математика та статистика - Робота з базами даних та їхня візуалізація - Продуктові метрики та A/B тестування - Юніт-економіка	- Методи кількісних та якісних користувацьких досліджень - Алгоритми й бізнес-процеси - Дослідження проблем та болей юзерів, планування рішень, тестування та вдосконалення продуктів - Прототипування інтерфейсів відповідно до потреб користувачів - Базове розуміння коду для взаємодії з технічною командою - Технічне документування - Побудова та валідація гіпотез на основі даних, використання інструментів аналітики - Основи графічного та motion-дизайну
Soft Skills	- Аналітичне та системне мислення - Швидке вивчення нової інформації - Прийняття рішень - Вміння домовлятися - Командна робота - Критичне мислення - Ініціативність, проактивність - Допитливість та комунікація	- Комунікація та командна робота - Допитливість і постійне навчання - Емпатія - Критичне мислення - Дизайн-мислення - Гнучкість у прийнятті рішень
Інструменти	Figma, Adobe XD, Adobe Photoshop, Adobe Illustrator, Sketch, Balsamiq, Google Analytics, Framer, InVision	Figma, Adobe XD, Photoshop, Illustrator, Sketch, Balsamiq, Google Analytics, Framer, InVision

Категорія	Назва ролі	
	Data Scientist	**QA Engineer**
Опис ролі	Фахівці, які аналізують дані і «тренують» ML-модель, яка може визначати певні типи закономірностей. Зокрема, розпізнати і класифікувати об'єкти на картинках, рекомендувати товар в онлайн-магазині тощо.	Спеціаліст, який слідкує за виконанням вимог до продукту: запускає тести та застосовує інші підходи для перевірки його якості
Зони відповідальності	- Тренування ML-моделі для перевірки гіпотез та реалізації бізнес-цілей продукту - Створення гіпотез, прогнозування поведінки користувачів - Візуалізація зібраних даних для подальшого опрацювання командою - Аналіз Big Data - Проєктування користувацьких функцій - Збір даних, аналіз поведінки користувачів	- Проведення усіх видів тестування - Розробка сценаріїв та процедур тестування - Участь в процесі формування вимог до продукту - Виявлення та опис багів продукту, аналіз знайдених проблем, контроль процесу їх усунення - Ведення тестової документації, програми bug-трекінгу
Hard Skills	- Знання Python (Pandas, Numpy, scikit-learn, LightGBM, CatBoost, TensorFlow) або R - SQL для роботи з даними - Дискретна математика, статистика, теорія ймовірностей - Розуміння основних алгоритмів машинного навчання	- Знання основ мов програмування (Java, Python, C#, Ruby) - Розуміння життєвого циклу та етапів розробки ПЗ (PDLC) - Уміння працювати з протоколом HTTP - Володіння інструментами тестування (ручне і автоматизоване) - Писати чеклисти, тест-кейси - Знання SQL, HTML, JSON
Soft Skills	- Ефективна комунікація - Прийняття рішень - Критичне мислення - Командна робота - Креативність - Стресостійкість	- Командна робота - Уміння давати та приймати фідбек - Ефективна комунікація - Прийняття рішень - Критичне мислення - Уважність до деталей
Інструменти	Python, TensorFlow, Pandas, Numpy, SQL	DevTools (інструменти розробника в браузерах), CrossBrowserTesting, Git (система контролю версій), SQL, MySQL, Jira (трекер задач), Excel, AWS (Amazon Web Services), Postman (інструмент для API тестування), Linux, JavaScript, LightShot (інструмент для зняття скриншотів), Redmine (система керування проєктами)

Категорія	Назва ролі		
	Front-end Developer	**Back-end Developer**	**Full-stack Developer**
Опис ролі	спеціаліст, основне завдання якого створення всі зображення, кнопки, тексти, віконця на платформі, тобто «зовнішня» частина продукту	Фахівці, що створюють програмно-апаратну частину («внутрішню») продукту, яка відбувається на стороні сервера та невидима користувачу	фахівці, які можуть виконувати роботу як front-end-, так і back-end-розробників
Зони відповідальності	- Створення інтерфейсу продукту - Адаптивна та кросплатформна верстка - Налаштування форм, кнопок, слайдерів - Програмування інтерактивних елементів сайту - Забезпечення якості користувацького інтерфейсу - Інтеграція з серверною частиною	- Забезпечення роботи «невидимих» функцій продукту - Проєктування архітектури функціоналу - Налаштування резервного копіювання - Інтеграція з зовнішніми сервісами (наприклад, платіжними) - Налаштування збереження даних, переходів між сторінками	- Проєктування клієнтської і серверної частин продукту - Планування, розгортання та тестування продукту - Контроль якості, запуск у продакшн
Hard Skills	- HTML, CSS, JavaScript - CSS-фреймворки, UI-бібліотеки - ReactJS, VueJS, AngularJS - Webpack, Git, API, Docker - UI/UX дизайн, CSS-препроцесори	- PHP, Node.js, Golang, Java, Python - MySQL, API, Docker - Git, протоколи TCP, HTTP/HTTPS - Сервери (Nginx), патерни та фреймворки (Nest, Express, Django)	- Володіння як front-end, так і back-end стеком - Робота з базами даних, API, Docker - Повне розуміння життєвого циклу вебзастосунку
Soft Skills	- Командна робота - Адаптивність - Ефективна комунікація - Дисципліна та структурованість - Прийняття та надання конструктивного фідбеку	- Командна робота - Адаптивність - Ефективна комунікація - Дисципліна та структурованість - Прийняття та надання конструктивного фідбеку	- Командна робота - Адаптивність - Ефективна комунікація - Дисципліна та структурованість - Прийняття та надання конструктивного фідбеку
Інструменти	Мови та фреймворки (HTML5, CSS3, JavaScript, AngularJS, Vue.js,React) Інструменти розробки (GitHub, Bootstrap, Stack Overflow,VS Code) Інструменти дизайну: (Figma)	Мови програмування (PHP, C / C++, Node.js, Golang, Java, Python) Бази даних (MongoDB, MySQL) Системи керування версіями (Git)	Мови та фреймворки (HTML5, CSS3, JavaScript, AngularJS, Vue.js,React, PHP, C / C++, Node.js, Golang, Java, Python) Інструменти розробки (GitHub, Bootstrap, Stack Overflow,VS Code) Бази даних (MongoDB, MySQL) Інструменти дизайну: (Figma) Системи керування версіями (Git)

Категорія	Назва ролі с		
	GAMEDEV SPECIALISTS		
	Unity Developer	**Game Designer**	**Game Artist**
Зони відповідальності	- Створення мультимедійних цифрових ігор для різних платформ - Робота з ігровим фреймворком Unity та C# - Участь у тестуванні, підтримці та оновленнях гри	- Розробка концепту гри, наративів, механік, рівнів - Побудова економіки гри, монетизація - Сценарії, правила, логіка гри	- Розробка стилю гри, створення ескізів, 2D/3D моделей - Робота над візуальною історією гри
Hard Skills	- Основи C#, ООП, .NET (базовий рівень) - Знання Unity API (MonoBehaviour, SceneManager, Application) - Математика: тригонометрія, дискретна математика, лінійна алгебра - Особливості Android та iOS платформ	- Ігрова аналітика, метрики - Google Sheets, Excel - Маркетинг, користувацька психологія - Теорія ймовірності, статистика	- Малювання, візуалізація, сторітелінг - UX/UI, графічний дизайн - Анімація, робота зі світлом та перспективою - 3D-моделювання
Soft Skills	- Креативність - Командна робота - Любов до ігор	- Аналітичне мислення, - Критичне мислення - Командна робота - Комунікація - Прийняття рішень	- Креативність - Адаптивність - Командність - Комунікація - Дисциплінованість
Інструменти	Unity, NUKE, Hammer Editor	Google Sheets ,Miro, Jira, Figma	Adobe Photoshop, Adobe Illustrator, Adobe After Effects, Autodesk Maya, Blender, Nuke, Unity, Autodesk 3ds Max, C++, Python, Houdini, Unreal Engine

Таблиця 2 – **Покрокова інструкція проведення командної зустрічі документ для фіксації і перевірки роботи над командним завданням**

	Запитання	**Результат**
1	Дата та час проведення зустрічі. *Вкажіть дату та час фактичного проведення зустрічі.*	*Заповнювати перед початком зустрічі.*
2	Модерував\ла проведення зустрічі. *Вкажіть модератора зустрічі/ ПІБ студента\ки.*	*Заповнювати перед початком зустрічі.*
3	Склад команди. *Вкажіть повний склад команди.*	*Заповнювати перед початком зустрічі.*
4	Були присутні на зустрічі? *Вкажіть, хто з учасників команди був присутнім на зустрічі.*	*Заповнюється на початку зустрічі.*
5	**Виконати «вправу на знайомство».** **Модератор запитуєкожного учасника:** • Розкажіть, що вважаєте своєю сильною стороною (дозволить визначити роль в команді та розподілити в подальшому завдання) • Розкажіть, яка сфера і чому саме вона є для вас цікавою (дозволить зрозуміти те, що ви вже знаєте про сферу і підкреслить спільне між вами) • Знайдіть спільні риси між вами Знайдіть 4-5 спільних рис між вами. Що надихає вас? Чим ви схожі?	*Зафіксовані відповіді учасників.*
6	**Обговорити вибір сфери розробки.** **Студент-модератор запитує кожного учасника:** Розкажіть, чому саме ця сфера є для вас цікавою? Чому вибрали саме цю сферу? Це дозволить зрозуміти те, що ви вже знаєте про сферу, і підкреслить спільне між вами для створення контакту. *Фіксуємо коротко відповіді членів команди.*	*Зафіксовані відповіді учасників..*
7	Команда вибирає спосіб прийняття рішень у разі спірних питань. **Модератор фіксує, який із варіантів вибрали учасники.** **1. Демократичний,** голосуванням (*всі рішення приймає більшість – не рекомендовано при парному числі учасників*). **2. Лідерський демократичний** (*загальні рішення приймаються шляхом домовленостей. Рішення щодо спірних питань приймає лідер команди, попередньо опитавши команду*). **3. Лідерский** (*лідер приймає ключові рішення, попередньо опитавши команду, але може бути усунутий голосуванням всіх членів команди*). **4. Ваш варіант** (*вкажіть свій спосіб, який обирає ваша команда для прийняття рішень*).	*Зафіксовано про про який варіант домовилась команда. ПІБ лідера команди*
8	Команда обговорює сферу ІТ та продукт, над яким	*Зафіксовано про що*

	працюватиме команда.	*домовилась команда.*
9	Члени команди вибирають ролі в проєкті. Використовуючи таблицю 1. Карти родей в команді.	*Зафіксовано ролі кожного учасника команди в Таблиці 3. Описано, які обов'язки зазначений фахівець виконуватиме у проєкті. Може бути відкореговано відповідно до планів побудови команди.*
10	Команда формудює місію та намір команди вашого продукту. Місія визначає причини існування команди та встановлює межі того, що буде або не буде робитися. Відповідає на питання, чому команда існує. Намір же визначає, як саме команда досягає цієї місії та які стратегії використовує.	*Місія команди – ...* *Намір команди –...*
111	Команда вибирає канали комунікації команди (наприклад, листи на email, повідомлення у Slack, зустрічі в Zoom, надання звітів у Asana, Jira).	*Перелічено канали комунікації*
12	Модератор-студент дякує участникам і завершує зустріч.	

Таблиця 3 – **Обов'язки в команді**

Фахівець	Прізвище члена команди	Опишіть, які обов'язки фахівець виконує у вашому продукті
Product Manager		
Project Manager		
Operations Manager		
Back-end Developer		
Front-end Developer		
DevOps Engineer		
iOS Developer		
Product Analyst		
Product Designer		
Talent Acquisition Specialist		
HR People Partner		
Marketing Manager		
PR&Partnerships Specialist		
Customer Support Specialist		

Чеклист для оцінювання виконання Блоку І проєктної роботи

- Чітко визначено та розподілено ролі в команді.
- Проведено першу зустріч, заповнено відповідні таблиці.
- Узгоджено канали комунікації в команді.
- Наявний заповнений документ із результатами знайомства.
- Виконано індивідуальне завдання (тест Бельбіна).

Рефлексія до Блоку ІІІ

- Яку роль я виконував(-ла) у команді? Чи відповідає вона моїм навичкам і перевагам?
- Які труднощі виникли при розподілі ролей?
- Чи вдалося налагодити ефективну комунікацію в команді?
- Який внесок я зробив(-ла) у першу зустріч команди?
- Що б я зробив(-ла) інакше при формуванні команди?

БЛОК ІІ. ІНІЦІАЦІЯ ПРОЄКТУ

ПРАКТИЧНІ ЗАВДАННЯ

1. Провести зустріч-брейнштормінг пошуку ідеї продукту. Заповнити таблицю 4. Створити презентацію ідеї продукту. Презентувати ідею викладачам.
2. Сформулювати місію та цілі проєкту.
3. Створити опис та порівняльний аналіз конкурентів (продуктів, що мають схожий функціонал з розроблюваним).
4. Створити та затвердити Статут проєкту.

МАТЕРІАЛИ ДЛЯ САМОСТІЙНОГО ОПРАЦЮВАННЯ

1. Що таке мозковий штурм та які техніки бувають
2. Project Initiation Phase
3. 5 phases of the project life cycle: An end-to-end guide
4. The project management life cycle explained
5. What Is The Project Life Cycle: The 5 Phases Explained
6. Project Life Cycle: A Guide to What it is and the 5 Life Cycle Stages
7. Project Initiation Ultimate Guide: 6 Steps To Start Projects Right
8. Project Initiation Phase - Importance and Roles Involved
9. Goals, Objectives and Deliverables in Project Initiation - Laying The Right Foundations for Your Project
10. Project Initiation Phase – From Idea to Action – Full Guide
11. 6 project constraints and how to manage them for project success
12. Project Initiation : Setting goals

Project Charter: Guide With Examples and Template

МАТЕРІАЛИ ДЛЯ ВИКОНАННЯ ПРАКТИЧНИХ ЗАВДАНЬ

1. Підготувати та провести зустріч - брейнштормінг пошуку ідеї продукту. Заповнити таблицю 4. Створити презентацію ідеї продукту та презентувати викладачам.

3.1.3. Підготовка проєктного менеджера до зустрічі.

Підготувати організаційні деталі зустрічі заздалегідь. Слід продумати:

- Час, дату, місце та тривалість зустрічі.
- Головні цілі зустрічі, яких необхідно досягти.

- Адженду (план пунктів до обговорення). Запланувати час для відповідей на них, щоб не виходити за таймінг.
- Наступні кроки, яким варто буде слідувати після завершення зустрічі.

Сформулювати мету проєкту. Перед тим, як обговорити конкретні завдання з командою, поясніть високорівневу ціль проєкту та його вплив на ІТ-продукт загалом. Це підвищує мотивацію фахівців, адже вони чітко розуміють на що впливатимуть, а отже, навіщо варто закривати свої зони відповідальності якісно.

Призначити члена команди, який вестиме нотатки зустрічі. Завдання цього спеціаліста полягатиме в тому, щоб записати ключові деталі зустрічі та поділитися ними з командою пізніше. Для нотаток можна використовувати онлайн-інструменти, у яких легко буде ставити запитання або лишати коментарі. Наприклад, FigJam.

Забезпечити встановлення контакту між членами команди проєкту. У випадку, якщо до команди долучилися нові фахівці, які раніше не працювали разом. Щоб допомогти членам команди проєкт познайомитися, запропонуй учасникам коротко розповісти трохи про себе, наприклад, ім’я, позиція в проєкті, зони відповідальності, цікавий факт чи хобі. Після створення доброзичливої атмосфери колегам буде легше приступати до серйозних тем.

Уніфікувати очікування команди. Після огляду мети та обсягу завдань проєкту важливо пояснити, що тобі, як проєктному менеджеру, потрібно від кожного члена команди. Кілька запитань, які у цьому допоможуть:

- Хто відповідатиме за результати ключових етапів проєкту?
- Коли потрібно виконати кожен етап?
- З ким слід консультуватися щодо деталей?
- Як часто ви очікуєте звітів про стан проєкту?

- Через який канал комунікації відбуватиметься більшість спілкування в команді?

Надихнути команду на нові звершення. Поясніть цінність майбутньої роботи для команди.

1. Проведення зустрічі-брейнштормінгу пошуку ідеї продукту. Заповнення Таблиці 4 – Пошук ідеї продукту.

Таблиця 4 – **Пошук ідеї продукту**

Крок	**Запитання**	**Результат**
1	Дата та час проведення брейнштормінгу.	
2	Модерував\ла командний брейнштормінг…?	*Вкажіть модератора, ПІБ студента\ки.*
3	Склад команди…?	*Вкажіть повний склад команди на момент проведення.*
4	Присутні на брейнштормінгу.	*Вкажіть, хто із учасників команди був присутнім на зустрічі.*
5	Модератор окреслює ціль брейнштормінгу: **1. Обрати сферу для створення власного продукту** **2. Пошук ідеї власного продукту у вибраній сфері**	
6	Модератор окреслює чіткий час на брейнштормінг.	*Дедлайн, до якого ідея повинна бути вибрана*
7	Модератор окреслює правила бренгштормінгу 2. І проводить перший етап. 1) **Брейштормінг буде проходити в 4 етапи:** • Генерація ідей (сфера + продукт); • Відбір; • Доопрацювання ідей. 2) Кожен учасник команди повинен обов'язково запропонувати варіанти по черзі і має не більше 1 хвилини на власну пропозицію. 3) Всі учасники уважно слухають варіанти один одного, не критикуючи і не перебиваючи один одного. 4) Модератор фіксує кожну ідею.	*Модератор записує всі ідеї першого етапу*
8	На другому етапі учасники розглядають ідеї і	*Модератор фіксує мінімум три*

	вибирають, яка із ідей найкраще відповідає критеріям пошуку: 1) Має достатньо великий обсяг ринку. 2) Можлива для реалізації і вже має певний ринок і ЦА. 3) У учасників як у команди, наявні переваги для реалізації саме цього продукту. (Інші критерії про успішність ІТ-продуктів)	*ідей, які вибрали учасники*
9	На третьому етапі учасники виступають в ролі критиків і захисників ідеї, опрацьовують відповіді на найбільш «небезпечні запитання» і ризики за моделлю SWOT: • Сильні сторони ідеї; • Слабкі сторони; • Ризики; • Можливості. *Почитати про SWOT-аналіз ІТ-продукту можете тут*	*Модератор фіксує SWOT-аналіз для розбору кожної з мінімум трьох ідей*
10	**Підготовка до презентації ідей та їх SWOT-аналіз викладачам).** • 2 хв на презентацію кожної ідеї; • максимум 5 хв на запитання.	*Результатом має бути презентація ідей* Обсяг презентації – мінімум один слайд для однієї ідеї Структура слайду: 1.сфера для створення продукту 2.опис ідеї 3. SWOT-модель

2. Створити опис продукту. Заповнити Таблицю 5 – Опис продукту.

3. Створити порівняльний аналіз конкурентів (продуктів, що мають схожий функціонал з розроблюваним). Категорії порівняння – категорії опису продукту. Обов’язкова умова – порівняння містить опис переваг розроблюваного продукту над існуючими. Результат представити в таблиці довільного формату.

Таблиця 5 – **Опис продукту**

Категорія	Приклад заповнення	Ваш продукт
Назва	QuickScan	
Місія	Підвищити продуктивність користувачів, зекономивши час шляхом оперативної роботи з документами.	
Основні функції	1) **Сканування документів.** Застосунок надає можливість відсканувати будь-який паперовий документ за допомогою камери смартфона та перевести його в електронний формат PDF або JPEG. 2) **Редагування відсканованого документа.** У додатку є можливість вносити зміни в електронну версію документа. Наприклад, видалити або виділити кольоровим фоном важливі частини документа. Є й опція електронного підпису, який формується юзером, зберігається і додається одним кліком у файл. 3) **Збереження відсканованого документа локально на пристрої.** Після сканування документа, він автоматично зберігається у самому застосунку. Додатково можна обрати функцію завантаження файлу на пам'ять пристрою. Це дозволяє користуватись документами й у режимі офлайн. 4) **Робота з багатосторінковими документами та їх обробка.** Додаток дозволяє сканувати документи від 2-ох сторінок і групувати їх в один файл для обробки та зберігання. Це допомагає уникнути плутанини зі збереженням кожної сторінки як окремого файлу під час роботи з об'ємними документами. 5) **Розпізнавання тексту на документі та можливість працювати з ним окремо у форматі Word (Optical Character Recognition).** Друкований та рукописний текст документа виділяється та розпізнається застосунком. Це надає можливість копіювати текст та змінювати його у будь-якому текстовому редакторі. 6) **Додавання коментарів до частин відсканованих документів.** У ситуаціях, коли не потрібно корегувати безпосередньо документ, а лише залишити нотатку для колег або майбутньої роботи, є функція додавання коментарів до відсканованого документа. 7) **Пересилання-шеринг відсканованого документа через пошту або месенджер.** Документ, що ви відсканували та уже перевели у формат PDF або JPEG, можна миттєво надіслати поштою або за допомогою месенджерів. Або працювати з документом прямо у застосунку разом з колегою, надавши доступ до редагування файлу через пошту.	
Унікальна пропозиція	QuickScan – це комплексне рішення для перетворення паперових документів у електронну форму легко та ефективно. Користувачі можуть сканувати документи, зберігати їх на своєму пристрої у форматах PDF або JPEG та вносити зміни до файлів у режимі редагування. Інтуїтивно зрозумілий інтерфейс забезпечує доступність у використанні як для користувачів, що мають досвід електронного документообігу, так і для тих, хто тільки починає працювати з цим. QuickScan оптимізує роботу з особистими документами та робочою документацією для власників малого та середнього бізнесу.	

Tone of Voice	- Орієнтований на користувачів та їхній фідбек. - Пропагує постійний розвиток та вдосконалення в легкому та ненав'язливому форматі. - Інклюзивний та зрозумілий для будь-кого. - Надає впевненість, що будь-які перешкоди в роботі з документами можна подолати. - Транслює надійність та безпеку, адже особиста інформація користувачів захищена.	
Формат (застосунок / веб / застосунок + веб)	Застосунок для iOS, заплановано – Android. Вебсайт для редагування відсканованих документів.	
Ринок (на яку географію користувачів націлений продукт)	Україна, США	
Мовні локалізації	Українська, англійська	
Спосіб монетизації	Для застосунку та вебсайту – **Freemium**.	
Умови монетизаційної моделі	Усі описані функції додатка є безоплатними, однак у такому режимі користувач може працювати лише із 3-ма документами. Також відсутня функція шерингу документа із колегами для спільного редагування. У платній підписці кількість документів для сканування, зберігання та редагування – необмежена. Також можна надавати доступ за поштою і працювати над документом разом з колегами. Аналогічно для роботи на вебсайті.	
Цільова аудиторія (всі сегменти)	- Власники малого та середнього бізнесу - Фрілансери - Менеджери проєктів, команд та інші управлінці - Люди, які часто подорожують - Студенти та науковці Спільна риса цих груп – потреба часто оперувати великою кількістю особистих або робочих документів, нотаток та файлів.	

Больові точки аудиторії	1) **Складність зберігання особистих документів.** Користувачі зіштовхуються з проблемою організації єдиного сховища для всіх паперових документів. Це ускладнює їх упорядкування та пошук. Розв'язання проблеми – надання юзерам ефективного та організованого помічника для зберігання документів. 2) **Непередбачена потреба використати документ.** Юзери потрапляють у ситуації, коли їм несподівано потрібні певні документи, які недоступні у фізичній формі. Застосунок надає швидкий доступ до електронних версій документів, гарантуючи, що користувачі зможуть отримати необхідну інформацію без затримок. 3) **Проблеми з якістю електронних документів на основі фотографій.** Робота з документами у форматі фото, а не сканів, призводить до зниження їхньої якості. Наприклад, спотворений текст, низька роздільна здатність і потрапляння зайвих предметів у кадр. Застосунок використовує вдосконалені методи обробки зображень, щоб не допустити такі помилки. 4) **Неможливість редагувати фотоверсії.** Фотоверсіям бракує гнучкості відсканованих файлів, оскільки користувачі не можуть легко змінювати їх. Додаток слугує інструментом безперешкодного редагування та приміток до своїх електронних документів. 5) **Передруковування тексту з паперових документів.** Щоб перенести текст із паперової версії документа в електронну, користувачі змушені передруковувати текст вручну. Це знижує продуктивність і створює ризик помилок. Застосунок містить технологію OCR (оптичного розпізнавання символів), щоб автоматично «витягувати» текст із паперових документів у електронний формат. 6) **Відсутність єдиного середовища для документів у власників малого та середнього бізнесу.** Застосунок пропонує уніфіковане рішення, що дозволяє власнику бізнесу класифікувати, керувати та ефективно редагувати документи, а також долучати до цього інших співробітників.	

4. Створити рамку Статуту проєкту. Заповнити Таблицю 6 – Статут проєкту.

СТАТУТ ПРОЄКТУ ТА ЙОГО СТВОРЕННЯ

Статут проєкту – офіційний документ, який:

• підтверджує початок проєкту.

• визначає обсяг, цілі, бюджет, часові рамки, учасників і ключових зацікавлених сторін;

• надає керівнику проєкту (Project Manager, PM) повноваження на управління ресурсами та рішеннями в межах проєкту.

Статут розробляється на початку життєвого циклу проєкту та є критично важливим для правильного запуску ініціативи.

Статут проєкту **потрібен для**:

• офіційного схвалення проєкту керівництвом або спонсорами;

• узгодження очікувань між усіма зацікавленими сторонами;

• надання повноважень керівнику проєкту;

• встановлення стратегічного напрямку реалізації проєкту;

• запобігання ризикам непорозумінь, неефективного розподілу ресурсів і розширення обсягу завдань без контролю ("scope creep").

Різновид документації проєкту, що створюється на етапі ініціації. Має на меті викласти основні характеристики проєкту.

Створення Статуту проєкту (інструкція)

1. Зрозумійте ключові цілі та завдання проєкту.

Почніть з чіткого визначення мети проєкту. Залучіть ключові зацікавлені сторони, щоб зібрати інформацію та очікування. Сформулюйте 3 – 5 цілей, яких потрібно досягти в ході реалізації проєкту. Переконайтеся, що вони є конкретними, вимірюваними, досяжними, актуальними та обмеженими в часі. Іншими словами, вони повинні бути SMART. Проаналізуйте, як проєкт може вплинути на бізнес-показники організації.

2. Визначте організацію проєкту.

Скорегуйте структуру проєктної команди та ролі й обов'язки, визначені на першій командній зустрічі з урахуванням цілей та завдань проєкту. Перегляньте та остаточно узгодьте керівника проєкту, ролі членів команди, а також визначення структури підпорядкування та каналів комунікації. В команді З'ясування цих питань на ранній стадії допомагає запобігти плутанині, встановити рамки підзвітності та повноважень в рамках проєкту.

3. Створіть план реалізації.

Складіть детальний план, в якому буде вказано, як будуть досягнуті цілі проєкту. Він має включати етапи проєкту, ключові заходи, проміжні етапи та терміни виконання. План реалізації повинен також детально описувати ресурси, необхідні для кожного етапу проєкту, зокрема, час, бюджет і людські ресурси.

4. Перелічіть потенційні проблемні зони.

Проведіть оцінку ризиків для визначення потенційних проблем, таких як розподіл ресурсів, відставання від графіків або зовнішні фактори. Розробіть стратегії пом'якшення цих ризиків. Такий підхід не лише готує команду до подолання труднощів, а й демонструє ретельне планування та прогнозування.

Дії після створення статуту проєкту

1. Авторизація проєкту

Першим кроком після створення статуту проєкту є отримання офіційного затвердження. Зазвичай це означає розгляд і схвалення статуту проєкту ключовими зацікавленими сторонами, такими як спонсори проєкту, вищий керівний склад або рада проєкту.

2. Розробка технічного завдання

Розробка детального технічного завдання є критично важливим етапом після затвердження проєкту. Цей документ містить детальніший опис результатів, меж

і критеріїв прийняття результатів проєкту, що ґрунтується на початковому обсязі робіт, визначеному в Статуті.

Важливі особливості складання Статуту

- Лаконічність: оптимальний обсяг – 1–2 сторінки.
- Чіткість формулювань: використовувати просту і недвозначну мову.
- Актуальність: оновлювати Статут у разі суттєвих змін у проєкті.
- Залучення команди: розробляти Статут спільно з ключовими членами проєкту.

Різниця Між Статутом та Планом проєкту

Параметр	Статут проєкту	План проєкту
Мета	Офіційний старт і загальні рамки	Детальне керівництво виконанням
Деталізація	Загальна інформація	Конкретні завдання, ресурси, розклади
Етап створення	На стадії ініціації	Після затвердження статуту

Основні ризики проєкту без статуту

- Невизначеність очікувань і цілей.
- Відсутність формальної підтримки проєкту.
- Неконтрольоване розширення обсягу робіт.
- Втрата фокусу команди на стратегічних цілях.

Таблиця 6 – **Статут проєкту**

	Опис пункту	Коментарі
Назва проєкту Коротка, зрозуміла, відображає суть проєкту		
Зміст проєкту Що буде виконано, а що – ні		
Проєктний менеджер Ім'я, роль і відповідальність керівника		
Дата початку проєкту		

Дата завершення проєкту		
Бізнес-кейс		
Ключові точки проєкту Етапи, стадії, ключові моменти		
Критерії успіху проєкту Які результати будуть визнані як успішні		
Ризики проєкту Основні загрози та способи їх мінімізації		
Команда проєкту Перелік членів команди, їх ролі на проєкті та обов’язки		
Бюджет Приблизна вартість проєкту		
Очікувані вигоди Цінність і користь від реалізації проєкту		
Обмеження проєкту		

РОЗРОБКА НА ОСНОВІ ГІПОТЕЗ

«Більшість компаній зазнають краху через те, що створюють продукт, який нікому не потрібен» – Ерік Ріс.

Суть підходу:

Розробка на основі гіпотез (Hypothesis-Driven Development) – це метод створення продуктів, що базується на систематичному тестуванні припущень. Вона дає змогу командам ітеративно формувати, перевіряти та вдосконалювати ідеї, орієнтуючись на реальні потреби користувачів.

Ключові характеристики:

- Продукт не проектується "в стіл", а поступово формується на основі зворотного зв’язку.

- Кожне рішення перевіряється через експеримент або MVP.
- Усі допущення, зроблені під час проєктування, мають бути або підтверджені, або спростовані.

Переваги підходу:

- Дає змогу мінімізувати ризики ще на ранніх етапах розробки.
- Сприяє створенню продукту, який відповідає очікуванням і реальним потребам цільової аудиторії.
- Забезпечує швидкий цикл навчання та корекції помилок.

Ключові запитання, які слід поставити в процесі:

1. Яке припущення у нашому проєкті є найбільш ризикованим або невизначеним?
2. Який найшвидший і найменш витратний спосіб перевірити це припущення?

Приклад застосування фреймворку

Таблиця 7 – **Створення сервісу з продажу товарів із доставкою**

Ситуація:	Команда планує запустити новий онлайн-сервіс з продажу товарів і розглядає можливість додати опцію доставки. Ідеї включають: найм кур'єрів, придбання брендованого одягу та сумок, можливо – транспорту. Проте перед впровадженням цього функціоналу команда вирішує перевірити гіпотези.
Крок 1: Формулювання гіпотез	• **Основна гіпотеза:** клієнти дійсно потребують послугу доставки. • **Додаткові гіпотези:** користувачам важливі вартість, швидкість і зручність доставки.
Крок 2: Попередня перевірка ідей	• Проведення **інтерв'ю** та **опитувань** користувачів з метою виявлення актуальних потреб. • Визначення бажаних **параметрів доставки**: оптимальний час, ціна, спосіб отримання.
Крок 3: Мінімальна реалізація функціоналу (MVP)	• Розробка простої вебсторінки або форми для введення адреси та бажаної дати доставки. • Створення **імітаційної кнопки "Доставка"** в інтерфейсі (без реального функціоналу), щоб перевірити інтерес – скільки користувачів натиснуть її.
Крок 4:	• Усі замовлення автоматично зберігаються у базі даних.

Ручна симуляція процесу	• Менеджер вручну обробляє ці замовлення, оформлюючи доставку через сторонній веб-сервіс. • Таким чином перевіряється логістика **без складної технічної інтеграції** на початковому етапі.
Крок 5: Аналіз і рішення	• На основі кількості натискань, зворотного зв'язку та конверсії приймається рішення: o чи потрібна послуга доставки взагалі; o яку **вартість доставки** користувачі вважають прийнятною; o які очікування щодо **швидкості**.

Мета:

- виявити функцію, яка принесе найбільшу цінність саме зараз;
- перевірити цю функцію якомога простішим способом, без повної розробки;
- **намагатися якнайшвидше спростувати кожну свою гіпотезу.**

Це непросто психологічно – довести самому собі, що ідея не має сенсу, – але це надзвичайно ефективно для бізнесу, оскільки дозволяє не витрачати ресурси на створення непотрібного продукту.

Типові помилки під час перевірки гіпотез

1. Розробка без перевірки гіпотез. Команда повністю впевнена у своїй ідеї та одразу переходить до створення продукту. Головний ризик – ігнорування потреб ринку та користувачів.

2. Пошук підтвердження, а не перевірка. Замість об'єктивного тестування – підсвідома спроба знайти докази на користь власної ідеї. Це когнітивне упередження виникає з двох причин:

- Засліплення ентузіазмом засновників (стартапи).
- Формалізований підхід до гіпотез (корпорації), де перевірка сприймається як крок, що автоматично веде до розробки, навіть якщо результат негативний.

3. Перевірка другорядного замість головного. Замість того, щоб протестувати ключову гіпотезу про попит або цінність, команди зосереджуються на менш важливих деталях – кольорі кнопок, іконках, стилях інтерфейсу. У результаті – втрата часу без реальної перевірки.

Гіпотеза попиту (потреби / проблеми)

Це одне з найбільш критичних припущень:

«Чи справді існує у користувача проблема, яку наш продукт має вирішити?»

Як перевірити попит без розробки?

- Landing Page – створіть просту посадкову сторінку з описом продукту, ілюстраціями та кнопкою дії.
- Оголошення – розмістіть оголошення на сайтах, де є ваша цільова аудиторія (OLX, Prom, Facebook).
- Імітація процесу – використовуйте рекламу або форми збору контактів, навіть якщо продукт ще не існує.
- Оболонка без функціоналу – якщо можливо, створіть "пусту" версію інтерфейсу, щоб перевірити реакцію.
- Відео-демо – створіть відео з імітацією функцій майбутнього продукту. Це дозволяє перевірити інтерес і зібрати зворотний зв’язок без програмування.

Гіпотеза цінності

Це припущення про те, що продукт не лише цікавий, а реально корисний для користувача і кращий за наявні альтернативи.

«Чи буде користувач готовий перейти на наш продукт?»

Як перевірити цінність без програмування?

- Інтерактивне тестування через сторонні сервіси – використовуйте інструменти для симуляції складної логіки (наприклад, no-code рішення).

- Ручна імітація сервісу – відтворіть логіку продукту вручну (наприклад, менеджер сам передає замовлення кур'єру).
- Тестування прототипів – створіть клікабельний прототип (Figma, Marvel) і проведіть usability-тестування. Опитування після тестів дозволяє оцінити, чи є в продукті реальна цінність.

Таблиця 8 – **Коротко: як тестувати ідеї без витрат?**

Гіпотеза	**Що перевіряємо**	**Як перевірити без розробки**
Попит	Чи потрібен продукт?	Лендінг, реклама, демо-відео
Цінність	Чи вирішує проблему краще за інші?	Прототип, ручна симуляція, no-code

Як створити та перевірити гіпотези для продукту: методологія HADI

HADI – це проста, але ефективна методологія перевірки ідей, яка допомагає перетворювати припущення на практичні дії, збирати дані та робити висновки. Назва походить від англійських слів:

- **H**ypothesis – гіпотеза
- **A**ction – дія (експеримент)
- **D**ata – дані
- **I**nsights – висновки

Цикл HADI: покрокова інструкція

КРОК 1. Формулювання гіпотези (Hypothesis)

Починається з чіткого припущення, яке можна протестувати. Структура гіпотези має вигляд:

«Якщо ми [виконаємо дію], то отримаємо [очікуваний результат]»

Приклади:

- Якщо ми додамо кнопку “Порівняти ціни”, то частка користувачів, які додають товар у кошик, зросте на 10%.

- Якщо ми надішлемо електронний лист із промокодом, то кількість переходів на сайт зросте вдвічі.

КРОК 2. Запуск експерименту (Action)

Організовується мінімальне втручання або експеримент, щоб перевірити гіпотезу.

Приклади дій:

- Додати кнопку/банер на сайт.
- Провести email-розсилку.
- Запустити рекламу.
- Впровадити опитування або форму зворотного зв’язку.

КРОК 3. Збір даних (Data)

Визначається період і метрики, за якими оцінюється результат експерименту.

Що слід фіксувати:

- Кількість кліків / переглядів / переходів.
- Конверсію в дію (наприклад, заповнення форми).
- Відгуки користувачів або рівень задоволеності.

КРОК 4. Аналіз результатів (Insights)

На основі зібраних даних робиться чіткий висновок:

- Чи справдилася гіпотеза?
- Що варто змінити?
- Який наступний експеримент запустити?

Цей крок замикає цикл та створює підґрунтя для нової гіпотези.

Переваги HADI-циклу:

- Легкість у застосуванні – підходить навіть початківцям.
- Структурований підхід до тестування ідей.
- Висока швидкість ітерацій.
- Підходить як для перевірки **попиту**, так і **цінності**.

Таблиця 9 – **Шаблон для HADI-гіпотези**

Елемент	Приклад
H	Якщо ми додамо FAQ у кошик, то зменшиться кількість кинутих кошиків
A	Додаємо блок FAQ на сторінку замовлення
D	Відстежуємо показник завершення покупки протягом 7 днів
I	Частка завершених покупок зросла на 15% → гіпотеза підтвердилась

Метод HADI: Як формувати та перевіряти гіпотези продукту

Методологія **HADI** (Hypothesis – Action – Data – Insights) – це ітеративний цикл, який дозволяє продуктологам і командам розробників системно перевіряти припущення про продукт. Кожен цикл складається з чотирьох логічних кроків:

КРОК 1. Формулювання гіпотези (Hypothesis)

Це початкова точка: **що саме ви хочете з'ясувати**?

Гіпотеза формулюється за шаблоном: **"Якщо [дія], то [результат]"**

Який рівень продукту ви тестуєте?

1. Рівень цінності – перевірка, чи вирішує продукт реальну проблему користувача.
2. Рівень функціональності – оцінка того, чи дозволяє певна функція швидко усвідомити цінність продукту.
3. Рівень дизайну – перевірка зручності та інтуїтивності взаємодії з інтерфейсом.
4. Рівень технічної здійсненності – перевірка можливості реалізації ідеї з технічної точки зору.

Як визначити пріоритет гіпотез?

Скористайтесь **ICE Scoring Model**, яка оцінює кожну гіпотезу за трьома критеріями:

Таблиця 10 – **ICE Scoring Model**

Критерій	Питання для оцінки
Impact (Вплив)	Наскільки сильно це покращить метрику? (конверсія, LTV тощо)
Confidence (Впевненість)	Наскільки ми впевнені в оцінці впливу та легкості реалізації?
Ease (Легкість)	Скільки зусиль, часу й ресурсів це потребує?

Формула: ***ICE = (Impact + Confidence + Ease)/3***

(усі значення оцінюються на одній шкалі, наприклад, від 1 до 10)

Приклад гіпотези:

Якщо ми додамо кнопку "Задати питання експерту", то частка користувачів, які завершують реєстрацію, зросте на 15%.

КРОК 2. Проведення дії (Action)

Після обрання пріоритетних гіпотез слід запустити перевірку. Це може бути:

- A/B тестування (розділення трафіку)
- Кількісні опитування (Google Forms, SurveyMonkey)
- Якісні інтерв'ю користувачів (Customer Development)
- Створення прототипу (Figma, Marvel)
- Фічі з поступовим запуском (Feature Flags)
- Обмежене оновлення для окремої групи користувачів (Beta)

Порада: обирайте найшвидший і найменш витратний спосіб тестування. Вашою метою не є створити ідеальний продукт, а зібрати найцінніші дані якнайшвидше.

КРОК 3. Збір даних (Data)

На цьому етапі ви збираєте **кількісні або якісні дані**, що виникають у результаті експерименту:

- Час взаємодії з елементами
- Кількість натискань/переходів

- Рівень задоволеності
- Фідбек із відкритими відповідями

Забезпечте логічну структуру вашої роботи над продуктом:

- Одні гіпотези – у стані планування.
- Інші – в активному тестуванні або аналізі.

КРОК 4. Інтерпретація результатів (Insights)

Після завершення тесту прийміть рішення щодо кожної гіпотези:

- **Підтверджена:** результат виявився позитивним.
- **Спростована:** очікування не виправдалися.
- **Відмова:** тест не має сенсу продовжувати (надто дорогий або малозначущий).

Починайте з гіпотез **найбільшого ризику** – це дозволить швидко і безпечно перевірити найслабші місця ідеї.

Де та як фіксувати гіпотези? Чому це важливо

Для ефективної перевірки гіпотез важливо мати **єдине джерело істини** – централізоване місце, де зберігаються всі ідеї, експерименти, результати та висновки. Це:

- забезпечує **прозорість** для всієї команди;
- полегшує **зворотний зв'язок** зі стейкголдерами;
- дозволяє швидко повернутися до **архіву тестів** під час ухвалення рішень.

Інструменти для збереження гіпотез

- **Google Sheets / Google Docs** – найпростіший формат із розділами: гіпотеза, план, дія, дані, висновки.
- **Coda / Notion** – більш гнучкі інструменти для створення структурованих баз знань, можна додавати таблиці, теги, коментарі.

- **Miro** – для візуального відображення HADI-циклів і командного брейнштормінгу.
- **Jira / Trello** – інтеграція гіпотез у потік задач.

Що має містити запис гіпотези

1. Формулювання гіпотези (у форматі: якщо..., то...).
2. Мета перевірки (який показник очікуєте покращити?).
3. Кроки дії / експерименту.
4. Період тестування та сегмент користувачів.
5. Метрики та зібрані дані.
6. Скріншоти, посилання, артефакти.
7. Висновки та рішення: підтверджена, спростована чи перенесена.
8. Коментарі учасників або нотатки з обговорень.

Як ефективно працювати з гіпотезами. Поради з організації процесу

1. Перевірка – це не просто “так/ні”. Навіть якщо гіпотеза не підтвердилась, вона дала нові знання. Це – не поразка, а етап навчання.

2. Не перевіряйте все підряд. Фокусуйтеся на важливому: проблемах, що впливають на попит, цінність і поведінку користувача. Не витрачайте час на “косметику”, якщо є серйозніші гіпотези.

3. Контролюйте сегментацію. Усі тести мають бути спрямовані на чітко визначену аудиторію. Інакше дані можуть ввести в оману.

4. Зберігайте об’єктивність. Не фільтруйте або не ігноруйте дані, які “не пасують” під очікування. Наприклад, якщо у вас є дані лише по буднях – не домальовуйте позитивну картину, додаючи вихідні навмання.

5. Фіксуйте процес і докази. Записуйте все: від скрінів прототипів до результатів опитувань. Це дозволяє іншим командним учасникам або новачкам зрозуміти хід вашого мислення.

6. Діліться результатами з командою. Обговорюйте гіпотези разом, шукайте зв’язки між ідеями та експериментами – це формує культуру продуктивної перевірки ідей.

Шаблон HADI-циклу формулювання гіпотез

Цей шаблон дозволяє командам формулювати гіпотези, планувати дії, збирати дані та робити висновки в структурованому форматі.

Таблиця 11 – **Шаблон формулювання гіпотез**

Hypothesis (Гіпотеза): Чітке формулювання припущення у форматі "Якщо..., то..."	
Action (Дія): План експерименту або дії для перевірки гіпотези.	
Data (Дані): Метрики та показники, які будуть зібрані.	
Insights (Висновки): Результати аналізу даних та подальші кроки.	
Impact (Вплив): Оцінка потенційного впливу на продукт або користувачів.	
Confidence (Впевненість): Рівень впевненості в гіпотезі.	
Ease (Легкість): Оцінка складності реалізації дії	

Рекомендації для командної роботи

1. **Копіюйте шаблон:** Створіть копію шаблону для своєї команди.
2. **Заповніть стовпці:** Разом з командою заповніть всі стовпці для кожної гіпотези.
3. **Пріоритизуйте гіпотези:** Використовуйте оцінки Impact, Confidence та Ease для визначення пріоритетів.

4. **Відстежуйте прогрес:** Оновлюйте шаблон по мірі виконання дій та отримання даних. Ведіть **архів гіпотез**, щоб уникнути повторів і враховувати попередній досвід.

Чеклист для оцінювання виконання Блоку ІІ проєктної роботи

- Проведено брейнштормінг, створено презентацію ідеї.
- Сформульовано місію та цілі проєкту.
- Описано продукт у визначених категоріях.
- Проведено аналіз конкурентів з таблицею порівняння.
- Заповнено та узгоджено Статут проєкту.

Рефлексія до Блоку ІІ

- Наскільки вдалою була ідея, яку обрала наша команда?
- Чи була презентація ідеї переконливою та логічною?
- Як ми сформулювали місію й цілі проєкту? Що вдалося, а що – ні?
- Який мій особистий внесок у SWOT-аналіз і опис продукту?
- Чи було зрозуміло, яку цінність має продукт для користувачів?

БЛОК ІІІ. ПЛАНУВАННЯ

ПРАКТИЧНІ ЗАВДАННЯ

1. Побудувати Roadmap продукту.
2. Обрати та обгрунтувати методологію та методику управління проєктом (Scrum, Kanban, Waterfall тощо)
3. Обрати модель документування проєкту. Визначитися з мінімальним переліком для документування. Створити документ-маршрутизатор документування проєкту.

МАТЕРІАЛИ ДЛЯ САМОСТІЙНОГО ОПРАЦЮВАННЯ

1. Gantt chart guide: How to create and use one.
2. What is strategic planning? A 5-step guide.
3. Strategic Planning.
4. Learn how to do operational planning the right way.
5. The Eisenhower Matrix.
6. Фреймворки пріоритизації (Prioritization Frameworks).
7. Asana.
8. Як створювати продуктові гіпотези: покрокова інструкція
9. Як створити успішний продукт за допомогою розробки на основі гіпотез. Детальний гайд від Devlight
10. Як генерувати та перевіряти гіпотези під час розробки продукту. Інструкція продакт-менеджерки
11. Вимоги в SAFe: як організувати ваш беклог?
12. Як налаштувати Jira для управління беклогом: покрокова інструкція
13. Планування спринту: пам’ятайте, що це спринт, а не марафон
14. Як розставити пріоритети для вашого продукту та досягти максимального успіху в бізнесі
15. User Story Mapping: від ідеї до релізу

МАТЕРІАЛИ ДЛЯ ВИКОНАННЯ ПРАКТИЧНИХ ЗАВДАНЬ

ВАЖЛИВО розрізняти поняття:

Метод – це спосіб або сукупність прийомів виконання певної дії; тобто техніка реалізації завдання або роботи.

Методологія – це система принципів, підходів і цінностей, що базуються на теорії та використовуються для вирішення проєктних завдань. Її іноді називають моделлю організації процесу.

Методика (або фреймворк) – це структурований, готовий до використання алгоритм дій, що описує, як поєднати різні методи й техніки для досягнення визначеної мети.

ПЛАН, АБО ДОРОЖНЯ МАПА ПРОЄКТУ (ROADMAP)

План, або дорожня мапа проєкту (Roadmap) – це стратегічний документ, що коротко і чітко описує план розвитку продукту, проєкту, компанії чи ринку. Використовується на початковому етапі проєкту або для впровадження змін у вже існуючих структурах. Складається з цілей (глобальні задачі проєкту), етапів (послідовність дій і досягнення цілей), термінів (опціонально - орієнтовні дати виконання етапів). Візуалізує стратегію розвитку, служить базою для планування, інформує всіх зацікавлених осіб, узгоджує дії команди.

Потрібен для інвесторів (розуміння інвестицій і термінів), ТОП-менеджменту (управління стратегією), співробітників (планування змін), команди розробки (координація технічних етапів). Види Roadmap: продуктові (стратегія розвитку продукту), проєктні (план виконання проєкту), галузеві (розвиток ринку), корпоративні (трансформація компанії), освітні (розвиток навичок).

Основними цілями Roadmap: є створення плану розвитку, узгодження цілей, підвищення прозорості, формування основи для планування і контролю.

Дорожня мапа проєкту може містити:

- **Project overview** – загальний огляд проєкту. Тут зазначають цілі, яких треба досягти, та scope statement.

- **Scope statement** – це зміст завдань та очікувані результати проєкту, що вкажуть на його успішність.
- **Work breakdown structure** – детальну декомпозицію усіх завдань проєкту.
- **RACI-матрицю** – для розподілу ролей на проєкті.
- **Таймлайн проєкту.** Наприклад, діаграму Ганта.
- Оцінені витрати та часові межі проєкту.

Ціль проєкту – це результат, який ми хочемо отримати по зевершенню проєкту. Вони формуються відповідно до бізнес-потреб.

Важливо розрізняти **верхньорівневі цілі** з глобальним впливом на розвиток продукту та нижчі за рівнем – **інструменти досягнення**.

Наприклад, верхньорівнева ціль – збільшити утримання користувачів на 20% за 3 місяці. А нижча ціль-інструмент – створення нової фічі.

Work breakdown structure – ієрархічна структура проєкту – це інструмент для розподілу завдань проєкту.

Він пропонує ділити ключові цілі на нижчі рівні. Така декомпозиція завдань допомагає краще оцінити, скільки ресурсів потрібно для виконання цих завдань.

- Перший рівень – **deliverables** (результати роботи) – відповідає на питання «що?»
- Наступні рівні – **work packages** (пакети робіт) – відповідають на питання «як?»

ЕТАПИ ПЛАНУВАННЯ ПРОЄКТУ

1. **Робота із зацікавленими сторонами (стейкхолдерами).** Хто зацікавлений в успіху проєкту? Описати, хто і в чому саме зацікавлений, які

задачі будуть виконувати ці люди. Якщо зацікавлені сторони не знайомі між собою, це варто виправити і провести зустріч: офлайн чи онлайн. На ній мають бути обговорені очікування людей від роботи один з одним

2. **Опис цілей.** Чітко окреслитися цілі проєкту. Описати, що саме вважається результатом роботи

3. **Створення списку задач.** На основі цілей. Великі етапи слід розбити на менші, прописати конкретні завдання, їх послідовність, пріоритетність, взаємопов'язаність (діаграма Ганта)

4. **Складання розкладу.** План роботи над проєктом має включати ключові дати в календарному вигляді: коли починається робота над тим чи іншим завданням, коли відбудеться проміжний контроль, коли завдання буде завершено і що буде далі

5. **Оцінка ризиків і за необхідності коригування планів.** Коли є детальний план, простіше побачити потенційні причини, з яких робота може піти не так, як хотілося б. Дещо можна виправити ще до старту. Також не завадить закласти у план трохи зайвого часу на випадок форс-мажорів

ШАБЛОН РОБОТИ ІЗ ЗАВДАННЯМИ:

- вказується проєкт, частиною якого воно є;
- прямо у заголовку стисло описується сутність завдання;
- призначається відповідальний за нього;
- вказується дедлайн;
- додається розгорнутий опис, якщо це потрібно, а також корисні/необхідні для виконання задачі файли;
- якщо завдання ділиться на кілька частин, можна зробити чек-лист, в якому одразу буде видно, що вже зроблено, а що ні.

РОБОТА З ВИМОГАМИ (БЕГЛОГ). МЕТОДИКА SAFe

Контекст і розуміння ролі беклогу в SAFe

У методології SAFe (Scaled Agile Framework) backlog — це впорядкований список елементів роботи, який визначає, що саме має бути реалізовано для досягнення цілей ART (Agile Release Train) або конкретної команди. Беклог — це не просто перелік завдань. Це інструмент планування, пріоритезації, комунікації та стратегічного вирівнювання.

Розрізняють:

- Portfolio backlog — містить Epic'и, що підтримують стратегічні теми портфелю;
- Solution backlog — для великих систем, містить Capabilities;
- ART backlog — містить Features для Agile Release Train;
- Team backlog — містить User Stories для окремої команди.

Командний беклог — це головний об'єкт управління для кожної Scrum-команди в SAFe. Він наповнюється задачами, які витікають із Features ART-беклогу. Управління командним беклогом відбувається в рамках Scrum або Kanban-процесів.

Важливо: створення якісного беклогу передбачає розуміння бізнес-цілей, взаємодію з Product Owner'ом та іншими зацікавленими сторонами (stakeholders), участь в PI Planning, використання WSJF (Weighted Shortest Job First) для пріоритезації та постійну ревізію та оновлення backlog refinement.

Вхідні дані для формування командного беклогу в SAFe

Перед тим як створити backlog команди, необхідно зібрати всі вхідні дані з відповідних рівнів SAFe:

1. Features з ART backlog. Це основні функціональні блоки, які має реалізувати команда. Їх формулюють Product Manager та System Architect, і вони є вихідними точками для створення User Stories.

2. Результати PI Planning. Планування інкременту (Program Increment) дозволяє команді визначити обсяг роботи на наступні 8–12 тижнів. Саме під час цього заходу формуються цілі інкременту та початкове наповнення беклогу.

3. Вимоги бізнесу та стейкхолдерів. Product Owner має збирати та узагальнювати побажання замовників, користувачів, бізнес-аналітиків тощо. Часто це оформлюється у вигляді Epic Hypothesis Statements або Lean Business Case.

4. Нефункціональні вимоги (NFRs). Вимоги до надійності, продуктивності, безпеки, відповідності стандартам тощо мають бути включені у форматі умов приймання (Acceptance Criteria) або як окремі user stories.

5. Ретроспективи та технічні борги. Під час ретроспектив команда визначає внутрішні задачі на вдосконалення процесів, інфраструктури чи виправлення помилок, які також мають бути включені в беклог.

6. Командна доступність і velocity. Для реалістичного формування беклогу необхідно враховувати кількість доступних членів команди, відпустки, відрядження та історичну швидкість команди (кількість story points, що команда може завершити за ітерацію).

Усі ці вхідні дані фіксуються, узгоджуються з командою та Product Owner’ом і лягають в основу backlog refinement.

Формування командного беклогу

Створення командного беклогу в SAFe виконується командою спільно з Product Owner’ом і передбачає кілька послідовних етапів:

1. Аналіз Features. Команда знайомиться з Features, визначеними в ART backlog, і дізнається контекст: для чого створюється продукт, яка користь для бізнесу, які існують обмеження.

2. Декомпозиція Features на User Stories. Кожна функціональність розбивається на окремі user stories, які мають відповідати критеріям INVEST:

- I — Independent
- N — Negotiable
- V — Valuable
- E — Estimable
- S — Small
- T — Testable

3. Формулювання Acceptance Criteria. Для кожної story Product Owner спільно з командою визначає умови приймання — чіткі критерії, що вказують, коли роботу можна вважати завершеною.

4. Оцінка обсягу роботи (Story Points). За допомогою Planning Poker або аналогічної техніки команда оцінює складність кожної user story. Це потрібно для планування обсягу роботи на ітерації.

5. Пріоритезація backlog. Найпоширеніший підхід у SAFe — WSJF (Weighted Shortest Job First), який враховує:

- Business Value
- Time Criticality
- Risk Reduction/Opportunity Enablement
- Job Size

6. Формування backlog refinement графіка. Команда домовляється про регулярність перегляду та оновлення беклогу (зазвичай — раз на спринт або частіше).

7. Уточнення залежностей. Визначаються залежності з іншими командами, ART або зовнішніми учасниками. Вони фіксуються в беклозі як спеціальні завдання або позначки.

Результатом є чітко сформований командний backlog, з яким команда працює впродовж інкременту, поступово адаптуючи його до змін бізнес-пріоритетів.

Підтримка, актуалізація та уточнення беклогу (Backlog Refinement)

Backlog refinement або backlog grooming — це постійний процес оновлення та удосконалення беклогу. У SAFe цей процес є ключовим елементом командної відповідальності та безперервного планування.

Основні завдання backlog refinement:

1. Додавання нових user stories. На основі нових запитів бізнесу, зворотного зв'язку користувачів, технічних потреб або PI Objectives додаються нові user stories або tasks.

2. Уточнення існуючих записів. Команда разом із Product Owner’ом аналізує, чи достатньо детально описані завдання, чи відповідають вони принципам INVEST, чи зрозумілі Acceptance Criteria.

3. Видалення або перенесення неактуальних пунктів. Змінюється бізнес-орієнтація або з’являються інші пріоритети — деякі записи можуть бути видалені або відкладені.

4. Оновлення пріоритетів. Залежно від нових бізнес-ризиків, критичності задач або стратегічних змін перераховується WSJF-пріоритетність.

5. Переоцінка складності. Якщо з’явились нові знання або змінено спосіб реалізації — команда переоцінює user story (story points).

6. Управління залежностями. Нові залежності між задачами або з іншими командами фіксуються та враховуються в плануванні.

Як проводиться refinement:

- Один або кілька разів на спринт (як окрема зустріч або частина планування).
- За участі всієї команди, Product Owner’а та, за потреби, архітектора або бізнес-аналітика.
- Використовується backlog board (наприклад, в Jira або іншому інструменті).

Результати:

- Підготовлений до планування backlog наступної ітерації.
- Узгоджені пріоритети, оцінки та критерії приймання.
- Відсутність неактуальних або «сирих» записів у backlog.

Цей процес забезпечує прозорість, адаптивність і відповідність планів команди стратегічним цілям організації.

Таблиця 12 – **Приклад SAFe Backlog Template**

ID	Feature	User Story	Acceptance Criteria	Story Points	WSJF Score	Priority	Dependencies	Status	Sprint	Comments
US-001	Авторизація користувача	Як користувач, я хочу увійти до системи, щоб отримати доступ до функцій.	1. Пароль не менше 8 символів2. Повідомлення про помилку	3	21	High	Залежить від модуля реєстрації	To Do	Sprint 1	UI макет уже затверджено
US-002	Створення профілю	Як користувач, я хочу створити профіль, щоб персоналізувати досвід.	1. Заповнені всі поля2. Дані зберігаються в базі	5	18	Medium	Немає	In Progress	Sprint 2	Потрібен бекенд API
US-003	Інтеграція з платіжною системою	Як клієнт, я хочу оплатити підписку, щоб активувати сервіс.	1. Підтримка VISA/PayPal2. Підтвердження платежу на email	8	15	High	API банку, безпечне з'єднання	To Do	Sprint 3	Обговорити з юридичним відділом

Пояснення полів:

- **ID**: унікальний ідентифікатор для user story;
- **Feature**: функціональність з Program backlog;
- **User Story**: опис потреби користувача;
- **Acceptance Criteria**: умови, за яких результат буде прийнятий;
- **Story Points**: оцінка обсягу роботи;
- **WSJF Score**: приоритезація за формулою Weighted Shortest Job First;
- **Priority**: рівень важливості (High/Medium/Low);
- **Dependencies**: залежності від інших команд, компонентів або етапів;
- **Status**: To Do, In Progress, Done;
- **Sprint**: номер ітерації;
- **Comments**: додаткові нотатки.

МЕТОДИКИ ВСТАНОВЛЕННЯ ПРІОРИТЕТУВАННЯ ВИМОГ

1. **Метод односторонніх дверей** використовує для пріоритетування засновник компанії Amazon Джефф Безос. Коли ризики великі, а ресурси обмежені, він запитує себе: «Які завдання видаються такими, ніби це єдиний шанс досягти успіху?». Так він фокусує свою увагу на найважливіших завданнях, відкидаючи менш критичні.

2. **Матриця Айзенхауера** – це спосіб швидкого сортування завдань і справ. Потрібно відповісти на два прості запитання – «чи це терміново?» та «чи це важливо?».

- Комірка A – важливі та термінові завдання. Це ті завдання, які треба виконати ASAP – as soon as possible, тобто якнайшвидше.
- Комірка B – важливі та нетермінові завдання. Тут розміщуються важливі завдання з некритичними дедлайнами, а також важлива щоденна діяльність.
- Комірка – термінові та неважливі завдання. Це завдання, які можуть не наближати до цілі, але їх наявність може заважати зосередитися на важливому. Часто їх можна делегувати, щоб звільнити ресурси для важливіших завдань.
- Комірка D – нетермінові та неважливі завдання. Це справи, які не несуть жодної користі для досягнення результату.

	Термінові	**Нетермінові**
Важливі завдання	**A** Виправлення критичних багів застосунку. Усунення збоїв на сервері. Витік персональних даних користувачів.	**B** Áналіз конкуретнів. Сесія стратегічного планування. Листи партнерам з результатами за квартал.

Неважливі завдання	**C** Написання підсумків зустрічі. Оплата підрядників Некритичні адміністративнізавдання.	**D** Дзвінки або листи без визначеної мети й результату. Зустрічі, в яких не передбачена твоя участь. Перегляд соцмереж у робочий час

3. **Impact-Effort Matrix (вплив / зусилля)** – це спеціальна матриця для оцінювання пріоритетності завдань, де Impact – вплив, який матиме реалізація конкретної фічі, а Effort – зусилля, які потрібні для її реалізації.

Для **оцінки впливу** спочатку оціни потенційний вплив кожної функції на ключові показники, задоволеність користувачів і бізнес-цілі компанії. Потім розглянь потенційні позитивні результати – збільшення залучення користувачів, їх утримання або збільшення доходу компанії.

А **оцінка зусиль** охоплює оцінку часу розробки, ресурсів та будь-яких потенційних технічних проблем для реалізації кожної функції. І також складності та залежностей, пов’язаних з кожною функцією.

ВПЛИВ		**Невеликі зусилля**	**Великі зусилля**
	Великий вплив	Швидкі перемоги	Важливі проєкти/завдання
	Невеликий вплив	Незначні завдання	Даремна втрата часу
	ЗУСИЛЛЯ		

Таймлайн проєкту

Діаграма Ганта – інструмент планування, який використовується для візуалізації плану та графіка робіт за будь-яким проєктом.

Діаграма складається зі стрічок, розміщених вздовж шкали часу. Кожна стрічка – це окреме завдання. А горизонтальні смужки вказують, як довго триватиме їх виконання.

Практичні рекомендації зі створення діаграми Ґанта

1. Складіть перелік усіх завдань вашого проєкту. Розбийте великі задачі на менші — це полегшить контроль виконання.

2. Визначте початок і кінець кожного завдання. Ураховуйте залежності: деякі задачі можна починати лише після завершення попередніх.

3. Встановіть тривалість кожного завдання у календарних днях або тижнях.

4. Визначте відповідального за кожну задачу — це забезпечить прозорість та контроль.

5. Виберіть інструмент для побудови діаграми (наприклад, Google Sheets, Microsoft Project, GanttPRO, ClickUp).

6. Побудуйте діаграму: кожне завдання — це окремий рядок, а тривалість позначається горизонтальними відрізками.

7. Регулярно оновлюйте діаграму під час виконання проєкту, щоб відображати реальний прогрес і коригування.

ОГЛЯД ОСНОВНИХ МЕТОДОЛОГІЙ РОЗРОБКИ ІТ-ПРОЄКТІВ

I. Waterfall (Каскадна модель)

Загальний опис: Waterfall – це класична послідовна модель управління проєктами. Кожен етап проєкту завершується повністю до початку наступного. Найчастіше використовується в проєктах, які можна чітко розділити на логічні фази.

Основні етапи:

1. Збір та аналіз вимог.
2. Проєктування (створення планів, прототипів).
3. Реалізація (розробка згідно з ТЗ).
4. Тестування (виявлення та виправлення помилок).
5. Впровадження (запуск продукту).
6. Підтримка (супровід у процесі експлуатації).

Ключові характеристики:

- Повна послідовність етапів, повернення до попереднього – неможливе.
- Велика увага до документації.
- Замовник залучається лише на фінальному етапі.

Переваги:

- Структурованість і передбачуваність.
- Простота управління.
- Чітке планування бюджету й строків.

Недоліки:

- Високий ризик невідповідності очікувань, якщо вимоги змінюються.
- Низька адаптивність.

Коли застосовувати:

- Великі або короткострокові проєкти з чіткими вимогами.

- Проєкти з обмеженим бюджетом або термінами.
- Будівництво, виробництво, проєкти з фізичними об’єктами.

II. Agile (Гнучка методологія)

Загальний опис: Agile – це реакція на обмеження Waterfall. Вперше представлений у "Agile Manifesto" (2001). Містить чотири ключові цінності:

- Люди важливіші за процеси.
- Працюючий продукт важливіший за документацію.
- Співпраця з клієнтом важливіша за контрактні умови.
- Готовність до змін важливіша за початковий план.

Особливості реалізації:

- Розробка ведеться короткими ітераціями.
- Продукт демонструється замовнику після кожного циклу.
- Постійне вдосконалення на основі зворотного зв’язку.

Переваги:

- Гнучкість і адаптивність до змін.
- Постійна комунікація з клієнтом.
- Підвищена мотивація команди.

Недоліки:

- Відсутність жорсткого плану може призвести до неефективності.
- Потребує високої залученості всіх учасників.

Коли застосовувати:

- У проєктах без чіткої візії кінцевого продукту.
- У стартапах та інноваційних ініціативах.

Agile Frameworks

Scrum

- Командна робота зі спринтами (2–4 тижні).
- Пріоритетування завдань у беклозі.
- Щоденні стендапи та демонстрація результату в кінці спринту.

- Оптимально для команд 5–10 осіб.

Kanban

- Візуалізація завдань на дошці (Trello, Jira).
- Завдання розміщуються в колонках (прийнято, в роботі, тестується, готово).
- Відсутність спринтів, але чіткі ліміти активних задач.

Scrumban

- Гібрид Scrum та Kanban.
- Поєднання спринтів та візуалізації.
- Підвищена гнучкість та адаптивність.
- Підходить для команд, що вже мають досвід Agile.

Extreme Programming (XP)

- Призначений для динамічних проєктів.
- Акцент на простоті, зворотному зв’язку та якості коду.
- Застосовується у складних проєктах з часто змінюваними вимогами.

III. Lean (філософія мінімальних витрат)

Загальний опис: Lean – концептуальний підхід, мета якого – створення цінності для користувача з мінімальними витратами. Часто застосовується у формі створення MVP (мінімально життєздатного продукту).

Принципи:

- Тестування ідей у реальних умовах.
- Збір зворотного зв’язку та поступове вдосконалення.
- Мінімізація невиправданих витрат.

Переваги:

- Ідеальний для стартапів.
- Підтримує ітеративну розробку.

IV. PRINCE2

Загальний опис: Офіційна методологія з Великої Британії, орієнтована на детальне планування та контроль. Часто використовується в соціальних та державних IT-проєктах.

Базується на:

- 7 принципах (доцільність, досвід, ролі, етапність, фокус на результат, адаптивність, економія комунікацій).
- 7 темах (обґрунтування, організація, якість, ризики, плани, контроль, зміни).
- 7 процесах (ініціація, управління, контроль, закриття тощо).

Переваги:

- Повна регламентація.
- Документальне забезпечення.
- Високий рівень контролю.

Недоліки:

- Складнощі з адаптацією до змін.
- Мінімальна гнучкість.
- Мала роль командної взаємодії.

Коли застосовувати:

- Великі проєкти з фіксованими вимогами.
- Сфера державного управління.

V. Six Sigma

Загальний опис: Методологія, зосереджена на контролі якості. Основна мета – усунення дефектів і постійне покращення процесів.

Основні етапи (DMAIC):

1. Define – визначення цілей.
2. Measure – збір вимог.
3. Analyze – аналіз проблем.

4. Improve – усунення недоліків.
5. Control – перевірка результатів.

Принципи:

- Орієнтація на клієнта.
- Аналіз на основі фактів.
- Прозорість і командна робота.
- Випередження помилок.

VI. Critical Path Method (CPM)

Загальний опис: Інструмент планування, що дозволяє визначити найважливіші задачі у проєкті. Побудований на оцінці тривалості всіх завдань та їх взаємозалежності.

Призначення:

- Визначення критичного шляху.
- Встановлення залежностей і паралельних дій.
- Контроль хронології.

Обмеження:

- Слабкий облік ресурсів.
- Варто поєднувати з іншими методами (наприклад, Waterfall).

VII. Як обрати відповідну методологію?

Необхідно врахувати:

- Сферу діяльності. В IT – переважно Agile, у промисловості – Waterfall.
- Бюджет. Фіксований чи гнучкий?
- Складність і розмір команди. Наприклад, PRINCE2 – для великих, Scrum – для малих.
- Рівень комунікації. Який ступінь взаємодії з замовниками?
- Пріоритети. Що важливіше – строки, якість, інновації чи стандарти?

ДОКУМЕТУВАННЯ ПРОЄКТУ (для AGILE. ФРЕЙМВОРК SCRUM)

НЕОБХІДНИЙ МІНІМУМ ДЛЯ ДОКУМЕНТУВАННЯ

БЛОК 1

Документ-маршрутизатор – єдина точка входу - місце, звідки можна дістатися всіх артефактів проекту (сторінка в Confluence, Notion простір, просто Google-документ тощо).

Артефакти проектних процесів – флоу процесу розробки, статусна модель завдань, дошки з беклогом, план-графік проекту, список потрібних контактів тощо.

Глосарій – фіксація єдиного понятійного поля.

Артефакти бізнес-потреби та бізнес-процесу

Agile-маніфест: "Працюючий продукт важливіший за вичерпну документацію". Але не можна розробити працюючий як треба продукт без чіткого розуміння, що ж ми все-таки робимо і навіщо. Бізнес-вимоги, схема процесу або просто лист із постановкою від замовника – щось має бути.

Концептуальна модель системи – опис основних сутностей, верхньорівнева функціональність та взаємодія з іншими системами. UML, IDEF0, "доріжки" BPMN, просто схеми або текстовий перелік – головне позначити принцип вашої системи.

Класи користувачів та рівні доступу – дуже допоможе при розробці вимог - завжди треба розуміти, для кого і що робиться - плутанина і в результаті невірний доступ – потенційно великі проблеми.

Сценарії використання – сценарій роботи системи – інструкція для користувачів чи тех. Підтримки, послідовність викликів із параметрами, якщо йдеться про взаємодію систем тощо.

Логіка роботи системи – дані так чи інакше передаються розробникам під час постановки завдання - документ, що читається, а в завданнях вставляти тільки посилання на місце в цьому документі. До того ж документа прикріплювати тест-кейси.

Опис АПІ – якщо з вами інтегруються зовнішні компанії - не обійтися без докладного опису параметрів.

Тестові дані – середовище, облікові записи і все, що потрібно, щоб не зводити з розуму тестувальника одноманітними питаннями.

Обмеження/нефункціональні вимоги – Ви домовилися, що розраховуєте максимум 100 користувачів? Робите імпорт довідників раз на добу о першій годині ночі? Зовнішня система не вміє набувати якихось значень? І багато чого ще... Зробіть приємно собі в майбутньому – запишіть усі ці деталі.

БЛОК 2

Архітектура системи – детальний опис сервісів та їх взаємодії (приклад, якщо сервісів кілька і вони взаємопов'язані, або архітектура мікросервісна.

Вимоги до даних – логічна модель, вимоги до складу та формату даних, особливості роботи з ними тощо (не завжди покривається характеристиками БД - –зафіксувати їх в окремому документі); угоди про формат БД – принципи найменування таблиць і атрибутів, використовувані типи даних тощо.

UX/UI макети та прототипи – збирати в одному відомому місці і тримати в актуальному стані. (уявіть, що у вас хоча б 3 екрани та у кожного хоча б по 5 станів – аналітик, дизайнер та розробники дивляться кожен у свій проект у Figma) – ПОРЯДОК В МАКЕТАХ – ПОРЯДОК НА ПРОЄКТІ.

Опис інтеграцій – протоколи інтеграцій, специфікації АПІ, процеси та потоки даних і все, що необхідно для уникнення непорозумінь при взаємодії з іншими системами.

Безпека – параметри безпеки можуть бути описані разом з доступами або нефункціональними вимогами, або успадковуватись від загального контуру IT компанії. У системах з пріоритетом чи особливим підходом до цієї характеристики – окремий документ.

Зовнішня документація – документи для користувачів, партнерів, наглядових органів та інші артефакти, якими система спілкується із зовнішнім світом.

ЯК ПИСАТИ ДОКУМЕНТАЦІЮ

Не забувайте про актуальність – пишіть лише ту документацію, яку ви зможете підтримувати у актуальному стані (неактуальна документація гірше, ніж відсутність документації).

Неоформлені артефакти – теж документація – складайте в одне місце ті артефакти, які не можна обробити і добре оформити (листи з домовленостями, приклади даних, посилання тощо – не забудете, зможете підтвердити якесь погодження).

Культивуйте культуру документування в команді – не обов'язково все має документувати одна людина, особливо якщо в команді немає спеціальної ролі для цього.

Домовтеся, як документація підтримуватиметься - коментарі в коді не завжди рятують – структура коду не повинна повторювати структуру процесу/функціоналу і, тим більше, бізнес-і користувацьких вимог. Код – «**ЯК**» працює система, проєктна документація – «**НАВІЩО?**», «**ЧОМУ?**» і «**ЯК МАЄ БУТИ?**».

Плануйте та декомпонуйте роботу з документацією – документування – це завдання, яке легко розбити на невеликі відрізки часу і розмазати по спринту.

Закривайте техборг – при введенні функціоналу в експлуатацію виділяйте час на дооформлення «хвостів». Зрозуміти, чого не вистачає просто: уявіть, що

ви приходите на цей проект зараз, і ніхто з попередньої команди з вами поговорити не може.

Більше схем та діаграм – візуальна інформація сприймається легше та швидше, краще запам'ятовується. Чим більше даних ви надасте графічно – тим доступнішою і коротшою буде документація (графічні нотації та UML).

Вчіться писати нехудожні тексти – навичка написання текстів сильно прискорює процес документування та підвищує його якість.

МЕТОДИКА СТВОРЕННЯ ДОКУМЕНТАЦІЇ «ЯК СТВОРИТИ «ХОРОШУ ДОКУ» (СТАТТЮ)»

Відповіді на блоки питань допоможуть у створенні документації проєкту.

Блок ЦІЛІ

1. Для кого я пишу статтю? Хто майбутній читач: користувач, адміністратор, розробник?

2. Які завдання перед ним (jobs to be done)? Чи є опис особи?

3. Який рівень підготовки цього користувача? Що він знає? Що йому неочевидно?

4. Як можна пояснити це користувачу-початківцю і при цьому не злити просунутого поясненням елементарних речей?

5. Що ще потрібно пояснити користувачеві, щоб він зрозумів основний зміст статті?

6. Який розділ документації підійде ця стаття?

7. Цю статтю чи її частину треба продублювати в інших розділах?

8. На які статті слід посилатися?

9. Можливо, цю статтю слід супроводжувати відеоінструкцією?

Блок ДЖЕРЕЛА ІНФОРМАЦІЇ

10. Чи поточні користувачі мають проблеми, пов'язані з темою статті?

11. Як зараз підтримка пояснює, що треба зробити?

12. Відділ маркетингу писав на цю тему статті та новини у блог? Чи можна «піддивитися» формулювання, структуру та ін.?

13. Чи є розділи на сайті, присвячені цій темі?

14. Що в сценарій закладав UX та продакт-менеджер? Чому це так?

15. Як це питання описано у конкурентів?

16. У яких сферах ще можна подивитися найкращі практики??

Блок ПЕРЕВІРКА ЗМІСТУ

17. Чи вдалося досягти мети статті?

18. Чи все буде зрозуміло більш просунутому користувачеві?

19. Чи все буде зрозуміло початківцю?

20. Все логічно та послідовно? Немає «стрибків» та прірв?

21. Послідовність дій правильна? Чи зможе користувач досягти мети, дотримуючись лише цієї інструкції?

22. Ми врахували всі кейси/шляхи користувача?

23. Чи вписується стаття у вибраний розділ?

Блок ПЕРЕВІРКА ВЕРСТКИ

24. Чи є нечитабельні простирадла тексту? Чи можна замінити схемою?

25. Чи є довгі абзаци?

26. Чи є надто короткі абзаци?

27. Чи є надто довгі списки?

28. Чи є надто вкладені складні для сприйняття списки (ті, в яких більше двох-трьох рівнів)?

29. Зображень достатньо?

30. Зображення не надто багато? Чи не ілюструємо ми надто очевидні кроки?

31. Якщо є схеми, вони зрозумілі?

32. Таблиці не складні сприйняття?

33. Сторінка загалом виглядає добре?

Блок ЛІТЕРАТУРНЕ РЕДАГУВАННЯ

34. Все оформлено на гайду?

35. Чи відповідає стиль решти документації?

36. Чи є пропозиції, які можна спростити?

37. Чи є складні терміни, які потребують пояснень?

38. Чи є канцеляризми?

39. Чи є повтори?

40. Нічого не ріже слух?

Блок ФІНАЛЬНА ВИЧИТКА

41. Чи немає помилок у правописі та пунктуації?

42. З перенесеннями, абзацами та розділами все гаразд?

43. Усі зображення підписано?

44. Елементи інтерфейсу названо правильно?

45. Чи скрізь стоять посилання? Вони працюють і ведуть куди треба?

Блок ПІСЛЯ ПУБЛІКАЦІЇ

46. Чи є у статті розділи, які «підтягуються» в інші статті? Вони оформлені макросами, щоб зміни до однієї статті автоматично застосовувалися до інших?

47. На цю статтю треба послатись з інших розділів? Якщо так, то з яких?

48. Чи потрібно додати до продукту швидке посилання на цю статтю?

49. Чи потрібно надіслати посилання на підтримку, маркетинг або інші відділи?

50. Чи потрібно віддати статтю перекладу?

АННОТАЦІЯ ДО ПРОЄКТУ

Анотація (іноді — резюме, summary, abstract) — це короткий, структурований виклад основної інформації про проєкт, наукову роботу або дослідження.

Основне призначення анотації:

- надати стисле уявлення про зміст роботи;
- допомогти читачеві зрозуміти, чи релевантна йому тема;
- коротко представити мету, методи, результати та значення дослідження.

Ключові ознаки анотації:

- Лаконічність: зазвичай 150–300 слів (для наукових статей), або 1000–2000 знаків для проєктів;
- Об'єктивність: не містить оцінок, висновків, емоційних суджень;
- Інформативність: не дублює назву, а розкриває суть;
- Стислість викладу без зайвих деталей.

Структура аннотації до проєкту:

1. Назва проєкту

- Має бути чіткою, інформативною та відображати предмет дослідження або проєктної діяльності.
- Уникайте загальних формулювань або надмірно довгих заголовків.

2. Мета дослідження / проєкту

- Опишіть проблему або потребу, яку вирішує ваш IT-продукт.
- Вкажіть, чому саме цей проєкт є важливим. Сформулюйте мету лаконічно та конкретно, підкресліть суспільну / наукову / галузеву цінність.

3. Цільова аудиторія / користувач

- Хто основний користувач системи?
- Для якої сфери або типу організацій вона призначена?

4. Функціональні можливості (що реалізовано)

- Коротко, але інформативно: перерахуйте основні модулі або компоненти, які були реалізовані.

5. Використані технології

- Які мови програмування, фреймворки, бази даних, середовища розробки або бібліотеки були використані?

6. Етапи розробки / архітектура

- За бажанням — коротко зазначте послідовність: проектування, розробка, тестування, запуск;
- або особливості архітектури (MVC, мікросервіси тощо).

7. Основні результати / інновації

- Що стало підсумком?
- Чим ваш проєкт відрізняється?
- Чи містить він нові підходи, зручні інтерфейси, інтеграції, автоматизацію?

8. Практична значущість / можливості застосування

- Як і де може бути використаний проєкт?
- Яку користь приносить?

9. Ключові слова

- 5–7 ключових термінів (для індексації, каталогу, журналу)

Чеклист для оцінювання виконання Блоку ІІІ проєктної роботи

- ▪ Створено Roadmap і презентовано його.
- ▪ Вибрано методологію управління проєктом (Scrum, Kanban, тощо).
- ▪ Сформовано беклог продукту у вигляді документа.
- ▪ Обрано модель документування, зазначено мінімальні артефакти.
- ▪ Узгоджено таймлайн із зазначенням ключових етапів.

Рефлексія до Блоку ІІІ

- ▪ Наскільки реалістичним був наш план роботи?
- ▪ Чи була обрана методологія (Scrum, Kanban) ефективною для нашої команди?
- ▪ Які задачі я ініціював(-ла) або описав(-ла) у беклозі?
- ▪ Чи вдалося дотриматися термінів?
- ▪ Що я дізнався(-лась) про командне планування та управління?

ПіІДСУМКИ ТА ПОДАЛЬШІ КРОКИ

Методичні вказівки охоплюють три ключові етапи проєктної роботи: формування команди, ініціацію проєкту та планування. У межах цих блоків студенти здобувають знання і навички, необхідні для старту командного ІТ-проєкту, формулювання його мети, вибору ідеї, розподілу обов'язків і створення базового плану реалізації.

Цілеспрямоване проходження цих етапів формує основу для практичної реалізації проєкту у наступних етапах. Зокрема, розробка MVP, його тестування, підбиття підсумків та ретроспектива будуть висвітлені у другій частині підручника.

У подальших розділах буде детально розглянуто процеси реалізації програмного продукту, взаємодію з користувачами, збір зворотного зв'язку, а також підготовку фінальної презентації. Це дозволить студентам отримати повний досвід проходження повного життєвого циклу ІТ-проєкту — від ідеї до реального результату.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. A Guide to the Project Management Body of Knowledge (PMBOK®Guide). – 7th ed. – Newtown Square, PA : Project Management Institute, 2021. – 370 p.
2. Schwalbe, K. Information Technology Project Management. – 9th ed. – Boston : Cengage Learning, 2018. – 672 p.
3. Sommerville, I. Software Engineering. – 10th ed. – Boston : Pearson, 2016. – 792 p.
4. Pressman, R. S., Maxim, B. R. Software Engineering: A Practitioner's Approach. – 9th ed. – New York : McGraw-Hill, 2019. – 944 p.
5. Highsmith, J. Agile Project Management: Creating Innovative Products. – 2nd ed. – Boston : Addison-Wesley, 2009. – 432 p.
6. Larman, C., Vodde, B. Practices for Scaling Lean & Agile Development. – Boston : Addison-Wesley, 2010. – 384 p.
7. Kerzner, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. – 12th ed. – Hoboken, NJ : Wiley, 2017. – 1104 p.