ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Гнучке управління розробки застосунку для створення і редагування нотаток NoteCraft»

Ступінь вищої освіти – магістр

Спеціальність – 073 «Менеджмент»

Освітня програма «Agile-технології розробки програмного забезпечення»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Керівники: зав. кафедри комп'ютерних наук,
к.е.н., с.н.с., доцент
Сергій МІЧКІВСЬКИЙ

старший викладач кафедри
комп’ютерних наук
Олег ЛУКУТІН

Виконала: здобувач
групи МЕН/Agile-24м
Олександр ПРИХОДЬКО

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

Київ, 2026 р.

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри інформаційного менеджменту, математики та статистики
_____ Денис БАЛДИК
«___»____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Приходько Олександр Сергійович

Тема роботи	Гнучке управління розробки застосунку для створення і редагування нотаток NoteCraft
Номер та дата наказу про затвердження теми	№ 109-3 від 14 жовтня 2025 року
Коротка постановка завдання	Метою роботи є дослідження застосування гнучкого управління розробкою програмного забезпечення з використанням Scrum на прикладі створення застосунку NoteCraft, формування беклогу, організації спринтів та аналізу результатів розробки за допомогою Agile-метрик.
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1. Schwaber K., Sutherland J. The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. Scrum.org, 2020. URL: https://scrumguides.org/scrum-guide.html (дата звернення: 16.12.2025). 2. Beck K. et al. Manifesto for Agile Software Development. Agile Alliance, 2001. URL: https://agilemanifesto.org/ (дата звернення: 16.12.2025). 3. Maurya A. Lean Canvas: The 1-page business modeling tool. Leanstack. URL: https://leanstack.com/lean-canvas (дата звернення: 16.12.2025).
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має містити теоретичне та/або практичне дослідження за темою роботи, яку слід розглядати як складне спеціалізоване завдання або практичну проблематику в галузі управління та адміністрування, яка характеризується комплексністю та невизначеністю умов і потребує застосування Agile-технологій.

Дата видачі завдання «27» жовтня 2025 р.

Керівник Олег ЛУКУТІН

Керівник Сергій МІЧКІВСЬКИЙ

Здобувач Олександр ПРИХОДЬКО

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
Підготовчий етап			
1	Вибір напряму дослідження та керівника.	01.09.2025 р.	*виконано*
2	Формування теми та призначення керівника.	22.09.2025 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи.	09.10.2025 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу.	27.10.2025 р.	*виконано*
Основний етап			
5	Розробка концепції кваліфікаційної роботи.	06.11.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів.	08.11.2025 р.	*виконано*
7	Теоретико-методичний аналіз предметної області та розширена постановка завдання. Підготовка та подання керівнику розділу 1 кваліфікаційної роботи.	13.11.2025 р.	*виконано*
8	Дослідницько-аналітична робота. Підготовка та подання керівнику розділу 2 кваліфікаційної роботи.	20.11.2025 р.	*виконано*
9	Розробка рекомендацій щодо вдосконалення управління із застосуванням Agile-технологій. Підготовка та подання керівнику розділу 3 кваліфікаційної роботи.	27.11.2025 р.	*виконано*
10	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи.	01.12.2025 р.	*виконано*
11	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівнику доопрацьованого варіанту кваліфікаційної роботи	03.12.2025 р.	*виконано*
Завершальний етап			
12	Представлення рукопису для перевірки на плагіат.	08.12.2025 р.	*виконано*
13	Підготовка презентації та доповіді на передзахист.	22.12.2025 р.	*виконано*
14	Передзахист кваліфікаційної роботи.	23-24.12.2025 р.	*виконано*
15	Технічна самоекспертиза роботи на відповідність вимогам до оформлення та виправлення недоліків.	12-16.01.2026 р.	*виконано*
16	Експертиза роботи керівником та зовнішнім експертом (рецензентом).	20.01.2026 р.	*виконано*
17	Доопрацювання доповіді та презентації для захисту.	22.01.2026 р.	*виконано*
18	Захист кваліфікаційної роботи.	26-30.01.2026 р.	*виконано*

Керівник Олег ЛУКУТІН

Керівник Сергій МІЧКІВСЬКИЙ

Здобувач Олександр ПРИХОДЬКО

АННОТАЦІЯ

Приходько О.С. Гнучке управління розробкою застосунку для створення і редагування нотаток «NoteCraft».

Пояснювальна записка кваліфікаційної роботи за спеціальністю 073 – Менеджмент (освітня програма – Agile-технології розробки програмного забезпечення), СО Магістр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра інформаційного менеджменту, математики та статистики, Київ, 2025 р.

У роботі розглядається процес гнучкого управління розробкою програмного забезпечення на прикладі створення застосунку для створення і редагування нотаток NoteCraft. Основною метою проєкту є розробка кастомізованого інструменту для роботи з нотатками, який дозволяє користувачам формувати власне робоче середовище шляхом використання модульного конструктора, таймерів, дедлайнів, кастомних звуків та механізмів спільного редагування.

Проєкт реалізується з використанням фреймворку Scrum, що забезпечує ітеративну розробку продукту, прозорість процесів та можливість швидкої адаптації до змін вимог. У роботі сформовано бачення продукту на основі Lean Canvas [3] та Product Vision Canvas, описано структуру Product Backlog, організацію спринтів, ролі Scrum-команди та застосування Agile-метрик для оцінювання ефективності розробки.

Результати виконання спринтів проаналізовано за допомогою показників Velocity, Burndown Chart, Sprint Report та інших артефактів Scrum, що дозволило оцінити доцільність використання гнучкого підходу в умовах створення програмного продукту з високим рівнем кастомізації та орієнтацією на користувача.

Ключові слова: Agile, Scrum, управління проєктами, програмне забезпечення, нотатки, Product Backlog, Jira.

Табл. 3 Рис.14 Бібліограф.: 15 найм.

ANNOTATION

Prykhodko O.S. Agile Management of Developing the Note Creation and Editing Application "NoteCraft".

Explanatory note of the master's qualification thesis in specialty 073 – Management (educational program – Agile Technologies of Software Development), Master's degree. – Higher Educational Institution "KROK University of Economics and Law", Educational and Scientific Institute of Information and Communication Technologies, Department of Information Management, Mathematics and Statistics, Kyiv, 2025.

The thesis examines the process of agile management in software development using the example of creating the note creation and editing application NoteCraft. The main objective of the project is to develop a highly customizable note-taking tool that enables users to build their own working environment through a modular note constructor, timers, deadlines, custom sounds, cloud synchronization, and collaborative editing features.

The project is implemented using the Scrum framework, which ensures iterative product development, process transparency, and the ability to quickly adapt to changing requirements. The study presents the product vision formulated using Lean Canvas and Product Vision Canvas, describes the structure of the Product Backlog, sprint organization, Scrum team roles, and the application of Agile metrics to evaluate development efficiency.

The results of sprint execution are analyzed using Velocity, Burndown Charts, Sprint Reports, and other Scrum artifacts, which made it possible to assess the effectiveness and feasibility of applying agile approaches in the development of a software product characterized by a high level of customization and user-centered design.

Keywords: Agile, Scrum, project management, software development, note-taking application, Product Backlog, Jira.

Tabl. 3 Fig. 14 Bibliography: 15 Items.

ЗМІСТ

ВСТУП

Сучасний ринок цифрових технологій та мобільних застосунків характеризується високим рівнем конкуренції, швидкістю змін та необхідністю адаптації до нових потреб користувачів. У таких умовах традиційні каскадні методи управління проєктами втрачають ефективність, оскільки не здатні забезпечити гнучкість, швидке реагування на зміни й можливість регулярного вдосконалення продукту. Саме тому в останні роки компанії все активніше переходять до гнучких підходів у розробці, серед яких Scrum займає одне з провідних місць.

Scrum дозволяє організувати роботу команди так, щоб забезпечити короткі цикли розробки, швидке отримання зворотного зв'язку, регулярний аналіз результатів та постійне покращення процесу. Гнучкі методи не лише оптимізують технічні процеси, але й створюють середовище для ефективної командної взаємодії, самостійності учасників команди та досягнення максимальної цінності продукту для кінцевого користувача.

У межах даної магістерської роботи було розроблено застосунок **NoteCraft** – інноваційний інструмент для створення, редагування, структурування та персоналізації нотаток. Застосунок орієнтований на максимально широкий спектр користувачів: студентів, IT-спеціалістів, креаторів, команди, які працюють над спільними завданнями, а також осіб, яким необхідний гнучкий і функціональний інструмент для щоденної організації діяльності. NoteCraft вирізняється глибокою кастомізацією, можливістю створення нотаток у форматі «міні-конструкторів», де користувач сам вибирає функціональні елементи: таймери, трекери, кастомні звуки, дедлайни, теги, шаблони тощо.

Додатковою цінністю NoteCraft є функція хмарної синхронізації, що дозволяє працювати з нотатками з різних пристроїв, та можливість надання доступу іншим користувачам для колективної роботи – що робить його корисним як у навчальному середовищі, так і в професійній спільній діяльності.

Метою роботи є побудова та опис процесу гнучкого управління розробкою застосунку NoteCraft за допомогою Scrum, включаючи аналіз вимог, формування

бачення продукту, створення беклогу, проведення усіх етапів Scrum-процесу протягом трьох спринтів, оцінку ефективності роботи команди та формування рекомендацій щодо подальшого розвитку.

Для досягнення поставленої мети було визначено такі завдання:

- дослідити теоретичні основи Agile та Scrum;
- проаналізувати методології гнучкого управління програмними продуктами;
- визначити цільову аудиторію NoteCraft, її потреби та проблеми;
- провести аналіз ринку та конкурентних рішень;
- сформувати бачення продукту, цілі та очікуваний результат;
- створити Product Backlog, структурований за пріоритетами;
- провести планування трьох спринтів;
- реалізувати спринти відповідно до вимог Scrum;
- здійснити оцінку результатів за допомогою Scrum-артефактів: Sprint Review, Retrospective, Burndown Chart, Velocity Chart;
- сформувати висновки щодо ефективності застосованої методології та розробленого рішення.

Практичне значення роботи полягає в тому, що створений застосунок може бути використаний широкою аудиторією, а описані процеси Scrum можуть слугувати прикладом для впровадження гнучких методів у реальних айті-проєктах.

Об’єкт дослідження - процес управління проєктами у сфері розробки програмного забезпечення в умовах динамічних вимог та невизначеності зовнішнього середовища.

Предмет дослідження - методи, інструменти та управлінські практики гнучкого управління проєктами (Agile, Scrum), що застосовуються для планування, організації, координації та контролю діяльності команди під час реалізації проєкту створення застсунку NoteCraft.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ ГНУЧКОГО УПРАВЛІННЯ ПРОЄКТАМИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Гнучке управління проєктами в умовах цифрової економіки

Розвиток цифрових технологій, зростання ролі програмних продуктів та прискорення інноваційних процесів суттєво вплинули на підходи до управління проєктами. Сучасні ІТ-проєкти реалізуються в умовах високої невизначеності, частих змін вимог користувачів та динамічного конкурентного середовища. У таких умовах традиційні підходи до управління, що базуються на детальному попередньому плануванні та жорсткій фіксації обсягу робіт, втрачають свою ефективність.

Гнучке управління проєктами сформувалося як відповідь на обмеження класичних каскадних моделей. Його сутність полягає у здатності адаптувати процеси управління та розробки до змін зовнішнього середовища, зберігаючи при цьому фокус на створенні цінності для кінцевого користувача. Гнучкі підходи передбачають ітеративний характер роботи, постійний зворотний зв’язок та залучення зацікавлених сторін до процесу прийняття рішень протягом усього життєвого циклу продукту.

З позиції менеджменту гнучке управління слід розглядати як управлінську модель, у якій ключову роль відіграють люди, комунікація та здатність команди до самоорганізації. Менеджер у такій системі не обмежується функціями контролю, а виступає координатором, фасилітатором і лідером змін. Це дозволяє підвищити залученість команди, скоротити час реагування на зміни та зменшити управлінські ризики, що є критично важливим для ІТ-проєктів.

1.2. Цінності та принципи Agile як управлінської парадігми

Основу гнучкого управління складають цінності та принципи Agile, сформульовані в Agile Manifesto.[2] Вони визначають пріоритети, на яких має базуватися організація роботи в умовах невизначеності. На відміну від

традиційних підходів, Agile не зосереджується на формальних процедурах і документації, а орієнтується на досягнення реальної цінності для користувача.

Ключовою цінністю Agile є пріоритет взаємодії між людьми над процесами та інструментами. Для менеджменту це означає необхідність формування ефективної команди, у якій забезпечено відкриту комунікацію, довіру та спільну відповідальність за результат. Такий підхід дозволяє швидше приймати рішення та адаптуватися до змін.

Іншим важливим аспектом Agile є фокус на робочому продукті як основному показнику прогресу. Регулярна поставка інкрементів дозволяє отримувати зворотний зв’язок і коригувати напрям розвитку продукту на ранніх етапах. Це зменшує ризик створення продукту, який не відповідає потребам ринку, та підвищує ефективність управлінських рішень.

Agile також визнає неминучість змін і розглядає їх як природну складову процесу. З управлінської точки зору це передбачає відмову від жорсткого довгострокового планування на користь адаптивного підходу, заснованого на коротких ітераціях та регулярному перегляді пріоритетів. Саме ця здатність до адаптації робить Agile особливо актуальним для розробки програмних продуктів.

1.3. Складні системи та Complex domain у контексті управління проєктами

Для обґрунтування вибору гнучкого підходу до управління розробкою застосунку NoteCraft доцільно звернутися до моделі Cynefin Framework, яка використовується для класифікації проблемних доменів та вибору відповідних управлінських підходів. Відповідно до цієї моделі, усі проблеми можна віднести до простих, ускладнених, складних (Complex) або хаотичних.

ІТ-проєкти зі створення програмних продуктів, орієнтованих на користувачів, належать до складного домену. Для таких систем характерна відсутність однозначних причинно-наслідкових зв’язків, еволюція вимог у процесі роботи та необхідність експериментального підходу до прийняття

рішень. Рішення в Complex domain не можуть бути повністю сплановані наперед, а їх ефективність стає очевидною лише після реалізації та аналізу результатів.

Застосунок NoteCraft є типовим прикладом продукту, що розробляється в умовах складного домену. Його функціональність формується на основі гіпотез щодо потреб користувачів, які перевіряються шляхом створення інкрементів продукту та отримання зворотного зв'язку. У таких умовах використання класичних підходів управління призвело б до зростання ризиків і втрати гнучкості, що обґрунтовує доцільність застосування Agile-підходів.

1.4. Scrum як фреймворк гнучкого управління в Complex domain

Scrum є фреймворком гнучкого управління, спеціально орієнтованим на роботу в умовах складних систем. На відміну від методологій, Scrum не визначає конкретні технічні рішення чи послідовність виконання робіт, а надає рамку, у межах якої команда може самостійно організовувати свою діяльність.

Основною ідеєю Scrum є ітеративне створення інкрементів продукту з регулярною інспекцією результатів та адаптацією планів. Такий підхід відповідає логіці Complex domain, де оптимальні рішення формуються поступово в процесі експериментування. Scrum забезпечує прозорість процесу, дозволяє швидко виявляти проблеми та коригувати напрям розвитку продукту.

Вибір Scrum для управління розробкою NoteCraft зумовлений його здатністю поєднувати управлінську дисципліну з високим рівнем гнучкості. Фреймворк дозволяє чітко визначити ролі та відповідальність, забезпечити регулярну комунікацію з зацікавленими сторонами та підтримувати стабільний темп роботи команди. Це робить Scrum ефективним інструментом управління проєктами в IT-сфері.

1.5. Управлінські ролі та взаємодія в Scrum-команді

У Scrum-команді управління реалізується через чітке розмежування ролей та відповідальності. Product Owner відповідає за формування бачення продукту

та максимізацію його цінності. З управлінської точки зору ця роль поєднує стратегічне планування, управління вимогами та взаємодію зі стейкхолдерами.

Scrum Master забезпечує ефективне функціонування процесу Scrum, усуває перешкоди та сприяє розвитку самоорганізації команди. Його роль є критично важливою для підтримки гнучкості та дотримання принципів фреймворку. Команда розробки, у свою чергу, несе колективну відповідальність за створення інкременту продукту та прийняття технічних рішень.

Взаємодія між ролями будується на принципах прозорості, довіри та спільної відповідальності, що дозволяє ефективно управляти проєктом без жорсткої ієрархії.

1.6. Scrum як безперервний управлінський цикл

Scrum формує повторюваний управлінський цикл, у межах якого відбувається планування, виконання, перевірка результатів та вдосконалення процесів. Такий цикл дозволяє менеджеру постійно контролювати стан проєкту та адаптувати управлінські рішення відповідно до поточної ситуації.

Регулярні події Scrum забезпечують прозорість процесу та створюють умови для своєчасного прийняття рішень. Завдяки цьому зменшуються управлінські ризики, підвищується передбачуваність результатів та забезпечується поступове досягнення стратегічних цілей продукту.

1.7. Порівняльний аналіз класичних та гнучких підходів до управління проєктами

У практиці управління проєктами традиційно використовувалися класичні підходи, зокрема каскадна модель (Waterfall), що передбачає послідовне виконання етапів проєкту з детальним плануванням на початковій стадії. Такий підхід є ефективним у проєктах із чітко визначеними вимогами та стабільним середовищем, однак у сфері розробки програмного забезпечення він часто виявляється недостатньо гнучким.

Основною проблемою класичних підходів є складність реагування на зміни вимог після затвердження плану. У цифрових продуктах потреби користувачів можуть змінюватися вже під час розробки, що призводить до зростання витрат та зниження якості кінцевого результату. З позиції менеджменту це означає зростання ризиків та зменшення керованості проєкту.

Гнучкі підходи, навпаки, ґрунтуються на принципі адаптивного планування. Вони дозволяють уточнювати вимоги поступово, приймаючи управлінські рішення на основі актуальних даних і зворотного зв’язку. Scrum як фреймворк забезпечує баланс між необхідністю контролю та потребою в гнучкості, що робить його більш придатним для управління сучасними IT-проєктами.

1.8. Управління вимогами в Agile-проєктах

Однією з ключових управлінських задач у проєктах зі створення програмного забезпечення є ефективне управління вимогами. У класичних моделях вимоги формуються на початковому етапі та фіксуються у вигляді технічного завдання, що обмежує можливість їх подальшого коригування. В Agile-підходах управління вимогами має інкрементальний характер.

У Scrum вимоги до продукту формалізуються у вигляді Product Backlog, який є динамічним артефактом і постійно уточнюється. З управлінської точки зору Product Backlog виступає інструментом пріоритизації робіт та стратегічного планування розвитку продукту. Це дозволяє фокусувати ресурси команди на створенні найбільш цінних функцій.

Такий підхід особливо актуальний для продукту NoteCraft, функціональність якого формується на основі різних сценаріїв використання та індивідуальних потреб користувачів. Управління вимогами в цьому випадку ґрунтується на перевірці гіпотез та поступовому розширенні можливостей продукту.

1.9. Роль зворотного зв’язку у гнучкому управлінні продуктом

Зворотний зв’язок є центральним елементом гнучкого управління та одним із ключових факторів успіху Agile-проєктів. На відміну від традиційних підходів, де оцінка результатів відбувається наприкінці проєкту, Agile передбачає регулярне отримання відгуків від користувачів і стейкхолдерів.

У Scrum зворотний зв’язок реалізується через демонстрацію інкрементів продукту наприкінці кожного спринту. Це дозволяє оцінити відповідність створеного функціоналу очікуванням користувачів та своєчасно скоригувати пріоритети. З управлінської точки зору такий механізм знижує ризик прийняття помилкових стратегічних рішень.

Для продукту NoteCraft регулярний зворотний зв’язок є критично важливим, оскільки рівень кастомізації та гнучкості функцій безпосередньо впливає на задоволеність користувачів. Саме через постійну взаємодію з користувачами забезпечується еволюційний розвиток продукту.

1.10 Психологічні аспекти самоорганізації та концепція Servant Leadership у гнучкому управлінні

Перехід до гнучких методологій управління вимагає докорінного переосмислення ролі менеджера в ієрархічній структурі організації. В межах Scrum-фреймворку традиційний контроль замінюється концепцією «лідера-слуги» (Servant Leadership), де основна функція Scrum-майстра полягає у створенні умов для максимальної продуктивності команди через усунення організаційних перешкод.

Академічні дослідження підтверджують, що самоорганізовані команди демонструють вищий рівень адаптивності до змін вимог саме завдяки відсутності жорсткого мікроменеджменту.

Психологічна безпека всередині колективу стає фундаментом для прозорості (transparency), що є першим стовпом емпіризму. Коли учасники команди не бояться відкрито обговорювати помилки під час щоденних зустрічей

або ретроспектив, процес інспекції та адаптації відбувається значно ефективніше, що є критично важливим для успіху складних IT-проєктів.

1.11. Управління ризиками в гнучких IT-проєктах

Управління ризиками є невід'ємною складовою менеджменту проєктів. У гнучких підходах ризики розглядаються не як загроза, а як фактор, який можна контролювати завдяки ітеративному підходу. Короткі спринти дозволяють виявляти проблеми на ранніх етапах і мінімізувати їх вплив.

Scrum сприяє зниженню технічних, організаційних та ринкових ризиків завдяки регулярній інспекції результатів та адаптації планів. Для менеджера це означає можливість оперативно реагувати на відхилення та приймати обґрунтовані управлінські рішення.

У проєкті NoteCraft гнучке управління дозволяє зменшити ризики, пов'язані з невизначеністю вимог, складністю інтеграцій та очікуваннями користувачів, що підтверджує доцільність використання Scrum.

1.12. Agile та стратегічне управління розвитком продукту

Гнучкі підходи до управління проєктами все частіше розглядаються не лише як інструмент оперативного управління, а і як елемент стратегічного менеджменту. Agile дозволяє поєднати довгострокове бачення розвитку продукту з можливістю адаптації до змін ринку.

У контексті розробки NoteCraft Scrum використовується як механізм реалізації стратегічних цілей через послідовне створення інкрементів продукту. Це дозволяє узгоджувати операційні рішення зі стратегічним баченням та забезпечувати сталий розвиток продукту.

1.13. Емпіричний контроль процесів та стратегічна роль Agile-метрик

Емпіричний підхід до управління базується на прийнятті рішень на основі фактичних даних, а не на теоретичних припущеннях. У гнучкій розробці це реалізується через систему метрик, таких як Velocity та Burndown Chart, які в

межах спеціальності «Менеджмент» слід розглядати як інструменти стратегічного прогнозування.

Швидкість команди (Velocity) не є мірилом продуктивності окремих розробників, а виступає показником пропускної здатності всієї системи розробки. Використання графіка згоряння задач дозволяє менеджеру візуалізувати ризики недосягнення цілі спринту в реальному часі та вживати превентивних заходів, таких як перегляд пріоритетів або декомпозиція складних User Stories. Таким чином, Agile-метрики трансформуються з інструментів операційного контролю в елементи системи підтримки прийняття управлінських рішень.

1.14. Ощадливе мислення (Lean Thinking) як фундамент максимізації цінності продукту

Інтеграція принципів ощадливого виробництва (Lean) у процеси розробки програмного забезпечення дозволяє зосередити зусилля команди на створенні максимальної цінності для клієнта за мінімальних витрат ресурсів.

Управлінська концепція Lean базується на усуненні восьми видів втрат (waste), серед яких у контексті NoteCraft найважливішими є надлишкова функціональність, затримки у передачі інформації між розробниками та дефекти, що виявляються на пізніх етапах спринту.

Застосування підходу «just-in-time» до формування вимог у беклозі дозволяє Product Owner не витрачати ресурси на детальне планування функцій, які можуть ніколи не бути реалізовані або чия актуальність може бути втрачена до моменту їх розробки. Таким чином, гнучке управління забезпечує високу ефективність використання бюджету та часу команди розробки.

1.15. Концепція самоорганізації та «лідерства-служіння» у Scrum-командах

Особливе місце у гнучкому управлінні посідає зміна ролі керівника проєкту, який у фреймворку Scrum трансформується з контролера у фасилітатора

або «лідера-слугу» (Servant Leader).

Такий тип лідерства спрямований на створення сприятливого середовища для самоорганізації команди, де кожен учасник несе відповідальність за спільний результат, а не просто виконує окремі вказівки.

Психологічна безпека та високий рівень довіри всередині команди розробки NoteCraft є фундаментом для реалізації принципу прозорості (transparency). Коли команда отримує повноваження самостійно вибирати способи розв'язання технічних задач, це стимулює креативність та підвищує якість коду, оскільки рішення приймаються тими фахівцями, які безпосередньо працюють над реалізацією функціоналу.

Висновки до розділу 1

Розділ 1 розкриває теоретичні основи гнучкого управління IT-проєктами з позиції менеджменту. Обґрунтовано доцільність використання Scrum як фреймворку управління у складному домені та показано його переваги у порівнянні з класичними підходами. Отримані результати створюють теоретичну базу для практичної реалізації гнучкого управління розробкою застосунку NoteCraft, що буде детально розглянуто у наступних розділах роботи.

РОЗДІЛ 2
АНАЛІЗ ВИМОГ, ЦІЛЬОВОЇ АУДИТОРІЇ, ТЕСТУВАННЯ, ТА ОБҐРУНТУВАННЯ НЕОБХІДНОСТІ РОЗРОБКИ ЗАСТОСУНКУ NOTECRAFT

2.1. Опис предметної області та передумови створення продукту

Ринок цифрових застосунків для роботи з нотатками характеризується високою конкуренцією та різноманіттям функціональних рішень. Більшість існуючих продуктів орієнтовані або на базове зберігання текстової інформації, або на вузькі професійні сценарії, що обмежує можливості користувачів із різними потребами. У сучасних умовах зростає попит на інструменти, які дозволяють не лише фіксувати інформацію, а й організовувати робочі процеси, планувати задачі та взаємодіяти з іншими користувачами.

Передумовою створення застосунку NoteCraft є потреба у гнучкому цифровому продукті, який поєднує простоту використання з можливістю глибокої кастомізації. Особливістю продукту є надання користувачам можливості самостійно конструювати функціональність нотаток шляхом додавання інтерактивних елементів, таких як таймери, дедлайни, нагадування або кастомні звуки. Водночас продукт має залишатися доступним і зрозумілим для користувачів, які не зацікавлені в складних налаштуваннях.

З управлінської точки зору NoteCraft розглядається як цифровий продукт, розвиток якого потребує постійного уточнення вимог та адаптації до змін очікувань користувачів, що обґрунтовує застосування гнучких підходів до управління.

2.2. Дослідження ринку та конкурентних рішень

Ринок застосунків для нотаток і організації інформації за останні 5 років зріс на 35 % [7], що обумовлено трендами:

- розвитком дистанційної роботи й навчання;
- популяризацією GTD-систем (Getting Things Done) [6];

- зростанням попиту на цифрові планувальники;
- підвищенням потреби у персональних інформаційних базах.

Порівняльний аналіз конкурентів наведений у табл 2.1.

Таблиця 2.1 - Конкуренти

Продукт	Переваги	Недоліки
Notion	Висока кастомізація, гнучкі бази даних	складність для новачків; надмірність функцій.
Google Keep	Простота, інтеграція з Google	обмежені можливості, відсутність кастомізації.
Evernote	Багато форматів збереження	мала інтерактивність; неактивний розвиток продукту.
Obsidian	Орієнтованість на текст та структуру	орієнтація на технічних користувачів; мало інтерактивних блоків.

Ні один із існуючих інструментів не поєднує одночасно простоту, кастомізацію та колаборацію без складного навчання.

NoteCraft має унікальну позицію – створення нотаток як конструктор:

- користувач вибирає блоки: текст, чекліст, таймер, дедлайн, аудіо, нагадування, вкладення;
- за бажанням – додає логіку: звуки, автопланування, статуси;
- працює в особистому просторі або в колаборації;
- інтерфейс лишається простим і лаконічним.

2.3. Аналіз користувацьких сегментів та персон (Persona Canvas)

Для визначення вимог до продукту було побудовано типовий образ користувача та сформовано Persona Canvas [5].

Persona Canvas (Аналіз персони Ігоря) (рис 2.1):

Профіль користувача: Ігор, 23 роки, молодший IT-спеціаліст, який постійно працює з великими обсягами інформації.

Поведінка та мотивація: Користувач цінує логіку та гнучкість; він готовий пробувати нові інструменти, але швидко відмовляється від них, якщо інтерфейс занадто складний або перевантажений функціями.

Болі (Challenges): Головною проблемою є неможливість адаптувати існуючі продукти під свій стиль роботи та обмеженість інструментів для спільного редагування.

Очікувані вигоди: Можливість керувати дедлайнами без сторонніх сервісів та створювати «нотатки-конструктори» для різних життєвих сценаріїв.

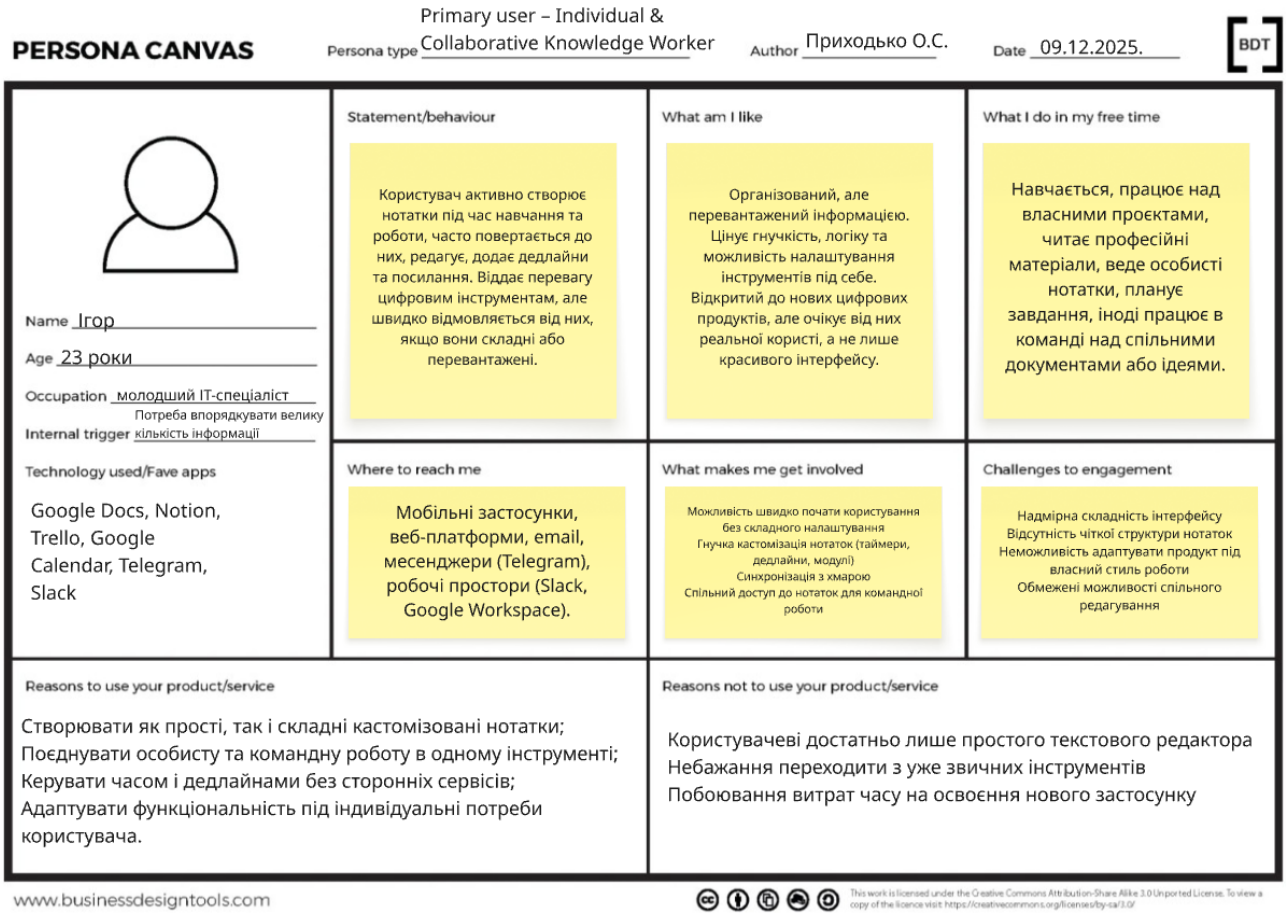

Рисунок 2.1. – Persona Canvas

Джерело: розроблено автором

2.4. Product Canvas

Для узагальнення бачення продукту та синхронізації бізнес-цілей із функціональними вимогами було використано Product Canvas (рис 2.2) за методикою Roman Pichler [4]. Заповнений канвас дозволив узгодити цілі

продукту, метрики успіху, цільову аудиторію та пріоритетні елементи беклогу, що стало основою для планування спринтів у межах Scrum.

Product Canvas (Бачення продукту):

- ціль (Goal). Створення гнучкого середовища, де структура нотатки підлаштовується під потреби користувача, а не навпаки;
- метрики успіху. Основними показниками ефективності розробки обрано Velocity (швидкість команди) та Burndown Chart (графік згоряння задач), а для продукту – якість виконання User Stories;
- цільові групи. Студенти, фахівці креативних індустрій та невеликі команди, що потребують швидкої синхронізації та спільної роботи.

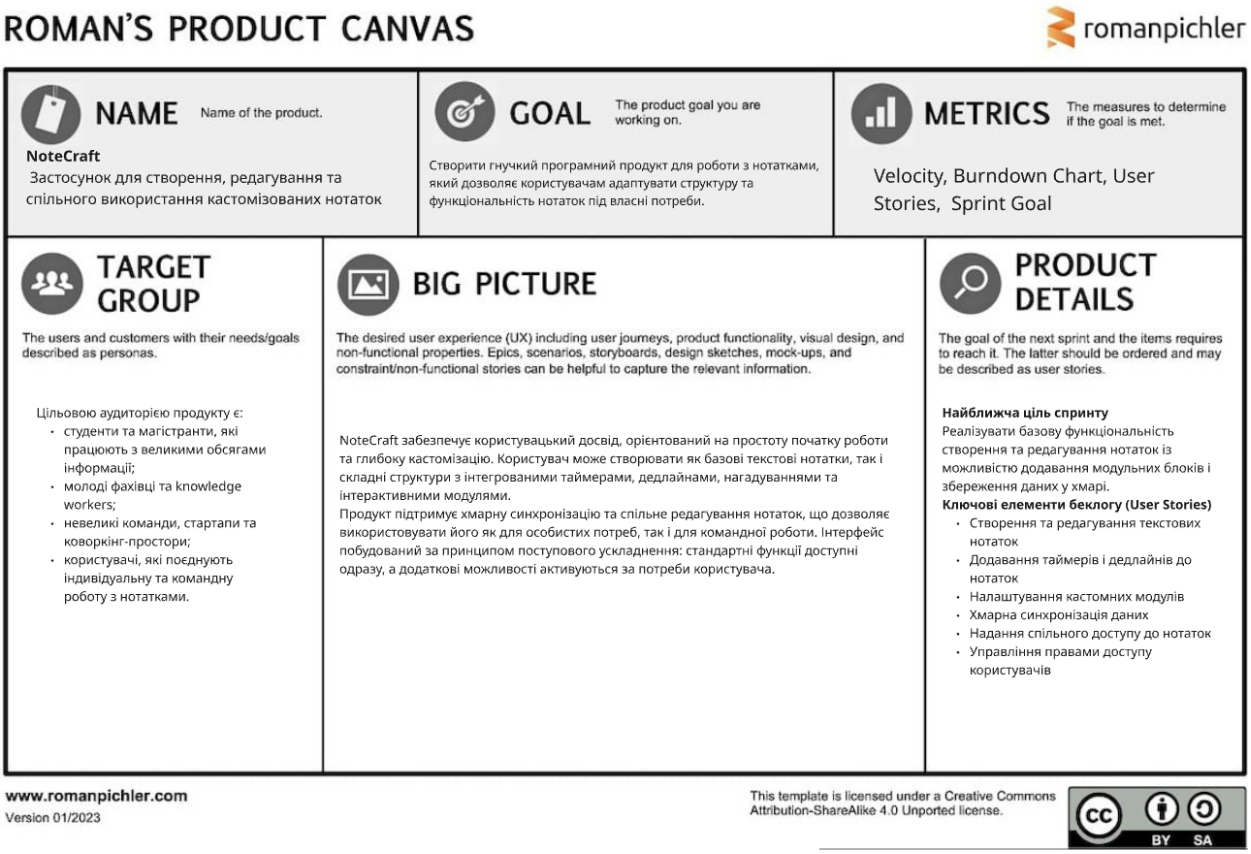

Рисунок 2.2. – Product Canvas

Джерело: розроблено автором

2.5. Формування вимог до системи NoteCraft

Функціональні вимоги:

- створення/редагування нотаток;

- міні-конструктор блоків (таймер, дедлайн, чекліст, аудіо);
- кастомні звуки та нагадування;
- теги, кольорові мітки;
- папки, структури, колекції;
- хмарна синхронізація;
- спільний доступ і колаборація;
- історія змін;
- пошук за ключовими словами;
- імпорт/експорт.

Нефункціональні вимоги:

- кросплатформність;
- висока швидкодія;
- простий інтерфейс;
- безпека (авторизація, токени, шифрування);
- доступність на мобільних та ПК.

2.6 Ризики проєкту та їх оцінка

Ризик-менеджмент є необхідною частиною управління будь-яким IT-проєктом, особливо якщо використовується гнучка методологія. У випадку NoteCraft ризики мають безпосередній вплив на якість, строки, стабільність та безпеку застосунку, тому їх систематичне виявлення та оцінювання дозволяє уникнути критичних помилок та підвищити прогнозованість результату.

Класифікація ризиків. У межах розробки NoteCraft було виділено чотири основні категорії ризиків:

1. Технічні ризики – пов’язані з технологіями, стабільністю архітектури, інструментами та інтеграціями.
2. Організаційні ризики – стосуються роботи команди, розподілу ролей, комунікацій та управління.
3. Процесні ризики – пов’язані з плануванням спринтів, оцінкою складності, визначенням пріоритетів.

4. Ризики користувацької взаємодії – UX-проблеми, недоліки інтерфейсу, безпекові загрози.

Ідентифікація та опис основних ризиків

Нижче наведено основні ризики, що були визначені командою під час планування, разом з їх наслідками.

Технічні ризики

- R1: Нестабільність хмарної синхронізації. Може призвести до дублювання нотаток, втрати даних або конфліктів версій;
- R2: Перевантаження WebSocket-каналу при великій кількості клієнтів. Це може знизити продуктивність та збільшити затримки в реальному часі;
- R3: Проблеми інтеграції з мобільною платформою. Деякі функції (таймери, вкладення, синхронізація) можуть працювати по-різному.

Організаційні ризики

- R4: Недостатня комунікація між бекенд-та фронтенд-розробниками. Це може викликати затримки у спринтах;
- R5: Нерівномірний розподіл задач. Частина команди може бути перевантажена, інша – недозавантажена.

Процесні ризики

- R6: Неправильна оцінка складності User Story. Призводить до зриву спринтів;
- R7: Незавершені задачі переносяться з кількох спринтів підряд. Це створює «борги» у беклозі.

Ризики користувацької взаємодії

- R8: Невідповідність UX стандартам. Занадто складний інтерфейс може призвести до негативних відгуків;
- R9: Недостатній рівень безпеки, уразливості при шарингу нотаток. Можливий витік конфіденційної інформації.

Матриця ризиків наведена у табл. 2.2.

Таблиця 2.2. – Матриця ризиків (Impact / Probability)

Ризик	Імовірність	Вплив	Рівень ризику
R1 – хмарна синхронізація	Висока	Високий	Критичний
R2 – WebSocket	Середня	Високий	Високий
R3 – мобільна версія	Середня	Середній	Помірний
R4 – комунікація	Низька	Середній	Низький
R5 – розподіл задач	Середня	Середній	Помірний
R6 – неправильна оцінка SP	Висока	Високий	Високий
R7 – перенесення задач	Середня	Середній	Помірний
R8 – UX-ризики	Середня	Високий	Високий
R9 – безпека	Низька	Високий	Помірний

Стратегії мінімізації ризиків

R1, R2 – технічні ризики:

- додаткові юніт-та інтеграційні тести;
- обмеження розміру даних для синхронізації;
- кешування ланцюжка версій;
- моніторинг WebSocket-подій;
- окремий performance-спринт.

R4, R5 – організаційні ризики:

- перехресні рев’ю;
- правило «не більше 2 задач у роботі»;
- щотижневий alignment-мітинг.

R6, R7 – процесні ризики:

- повторна калібровка Story Points;
- використання Planning Poker [12];
- спліт великих задач.

R8, R9 – UX та безпека:

- проведення 3 етапів UX-тестування;

- використання JWT, HTTPS, RBAC;
- аудит коду перед релізом.

2.7. Тестування програмного забезпечення NoteCraft

Якість програмного забезпечення є одним з ключових критеріїв успішності NoteCraft. Враховуючи, що продукт містить хмарну синхронізацію, реальний час, спільний доступ та модульний конструктор нотаток, тестування охоплювало як функціональні, так і нефункціональні аспекти.

Види тестування, застосовані в проєкті

1. Модульне (Unit) тестування

Перевіряло окремі компоненти:

– логіку збереження нотаток;

– роботу таймера;

– історію змін;

– роботу API-ендпоїнтів.

2. Інтеграційне тестування

Перевіряло взаємодію між:

– сервером і клієнтом;

– WebSocket та базою даних;

– конструктором блоків і синхронізацією.

3. Системне тестування

Перевірка всієї системи як єдиного цілого.

4. Регресійне тестування

Проводилось після кожного завершеного спринту.

5. UX-тестування

На групі з 12 тестувальників: перевірка інтуїтивності інтерфейсу.

6. Навантажувальне тестування

Моделювало одночасне редагування 10–30 користувачами.

7. Безпекове тестування

– перевірка токенів доступу;

– захист WebSocket;

– захист від CSRF/Replay атак;

– тестування прав доступу (RBAC).

Тестові сценарії для ключових функцій:

Тестування хмарної синхронізації

- внесення змін на одному пристрої миттєво відображається на іншому;
- відсутність дублювання нотаток після перезавантаження;
- коректне злиття версій у випадку конфлікту.

Тестування WebSocket

- всі підключені клієнти отримують оновлення без затримки;
- система правильно обробляє втрату з’єднання;
- пакети передаються в правильному порядку;

Тестування спільного доступу

- користувач з правами “Read” не може редагувати нотатку;
- користувач з “Edit” може змінювати весь контент;
- скасування доступу відразу блокує редагування.

Тестування таймерів і дедлайнів

- спрацьовування в точний момент часу;
- робота в офлайн-режимі до відновлення зв’язку.

Інструменти тестування

Jest – unit-тести [14]

Mocha/Chai – серверні тести

Postman/Newman – API-тестування

Selenium – автотести UI

JMeter – навантажувальні тести

OWASP ZAP – тестування безпеки [13]

Результати тестування

- 87% unit-покриття
- 94% інтеграційних сценаріїв пройдено успішно

- навантаження у 50 клієнтів не призвело до падіння
- виявлено 37 дефектів, усі виправлені до релізу
- критичних багів після Sprint 3 не знайдено

2.8. Стратегія управління користувацьким досвідом (UX) в умовах невизначеності

Процес розробки застосунку NoteCraft базується на глибокому аналізі потреб персони користувача, що вимагає інтеграції методів дизайну, орієнтованого на людину (User-Centered Design), безпосередньо в ітераційний цикл розробки. Використання Persona Canvas для аналізу профілю Ігоря дозволило ідентифікувати ключові больові точки, зокрема перевантаженість інтерфейсів конкурентних рішень та відсутність гнучкості в управлінні дедлайнами.

Такий підхід мінімізує когнітивне навантаження на користувача та забезпечує масштабованість продукту без втрати інтуїтивності інтерфейсу.

2.9. Методологія управління технічними ризиками в хмарних екосистемах

Специфіка розробки NoteCraft як хмарного інструменту колаборації передбачає наявність високих технічних ризиків, пов’язаних із цілісністю та синхронізацією даних.

Матриця ризиків (Impact/Probability) вказує на те, що нестабільність хмарної синхронізації є критичним фактором, який може призвести до втрати довіри користувачів.

Управлінська стратегія мінімізації цього ризику передбачає не лише технічні заходи, такі як використання WebSocket для передачі даних у реальному часі, а й процесні інновації, зокрема впровадження окремих «performance-спринтів» для оптимізації архітектури. Безпекове тестування (OWASP ZAP) та використання JWT-токенів забезпечують виконання нефункціональних вимог до системи, що є обов’язковим стандартом для сучасних хмарних рішень

2.10. SWOT-аналіз продукту NoteCraft у контексті конкурентного середовища

Для комплексного оцінювання ринкових перспектив NoteCraft було проведено стратегічний аналіз сильних та слабких сторін, а також можливостей і загроз. Головною перевагою продукту є його унікальна модульність та інтеграція інструментів продуктивності (таймери, дедлайни) безпосередньо в текстовий контекст.

Слабкою стороною на початковому етапі може бути менша впізнаваність бренду порівняно з гігантами ринку, такими як Notion або Google Keep.

Однак можливості розширення за рахунок корпоративного сектору та розробки специфічних шаблонів для студентства створюють потенціал для швидкого зростання бази активних користувачів (MAU). Зовнішні загрози, такі як посилення вимог до кібербезпеки, нівелюються заздалегідь розробленою стратегією захисту даних, що включає JWT-авторизацію та шифрування трафіку.

2.11. Обґрунтування вибору технологічного стеку та архітектурної гнучкості системи

Для реалізації амбітної мети створення «застосунку-конструктора» було обрано модульну архітектуру на базі NoSQL-рішення MongoDB Atlas. Вибір нереляційної бази даних обґрунтований необхідністю зберігати нотатки з абсолютно різною структурою блоків (таймери, зображення, чеклісти) без фіксації суворої схеми даних.

Це забезпечує архітектурну гнучкість, яка є критичною для продуктів, що розвиваються ітеративно. Використання протоколу WebSockets для реалізації синхронізації в реальному часі (NOT-13) дозволяє команді забезпечити безперебійну колаборацію, мінімізуючи затримки при одночасному редагуванні документа кількома користувачами. Така технічна стратегія відповідає принципам високої швидкодії та масштабованості, що закладені в нефункціональні вимоги до системи.

2.12 Життєвий цикл нотатки в системі

Життєвий цикл нотатки в системі NoteCraft являє собою комплексний процес, що поєднує взаємодію клієнтського інтерфейсу, серверної логіки та хмарної інфраструктури для забезпечення цілісності та доступності даних.

Процес ініціюється на боці клієнтського застосунку (React або React Native), де користувач взаємодіє з модульним конструктором для створення нового об'єкта або редагування існуючого.

На цьому етапі в стані програми формується об'єкт нотатки, який складається з динамічного набору блоків, таких як текстові фрагменти, чеклісти, таймери або мультимедійні вкладення. Локальне сховище пристрою забезпечує миттєве відображення змін для користувача, що дозволяє мінімізувати відчуття затримки інтерфейсу, поки відбувається підготовка до мережевої передачі.

Передача даних до централізованого сервера Node.js реалізується через комбінований підхід, що використовує REST API для початкового створення об’єктів та протокол WebSockets для забезпечення синхронізації змін у реальному часі.

Кожен запит супроводжується передачею JWT-токена в заголовках, що гарантує ідентифікацію автора та санкціонованість доступу до конкретної нотатки. Сформований JSON-пакет, що містить ідентифікатор нотатки, оновлений контент блоків та мітку часу (timestamp), надсилається до сервера, де проходить етап валідації на відповідність схемі даних та перевірку прав доступу користувача.

На серверному рівні відбувається оркестрація процесу збереження, де Node.js обробляє вхідний потік даних та ініціює запис до хмарної бази даних MongoDB Atlas. Використання нереляційної моделі даних дозволяє гнучко зберігати нотатки з різною внутрішньою структурою без необхідності попереднього опису суворої схеми. Після успішного підтвердження транзакції від бази даних, сервер оновлює стан документа та надсилає сигнал підтвердження клієнту-ініціатору, що завершує цикл збереження для первинного пристрою.

Важливою складовою життєвого циклу є механізм поширення оновлень (real-time propagation) на інші підключені пристрої користувача або пристрої співавторів. Сервер, використовуючи активні WebSocket-з’єднання, здійснює широкомовну розсилку (broadcast) повідомлення про зміну документа всім клієнтам, які мають права доступу до цієї нотатки.

У випадку виникнення конфліктів версій, коли зміни вносяться одночасно з різних джерел, система застосовує алгоритми вирішення конфліктів, базуючись на аналізі ланцюжка версій або принципі переваги останнього запису, що запобігає дублюванню або втраті інформації. Таким чином, життєвий цикл нотатки замикається синхронним оновленням інтерфейсів на всіх пристроях, забезпечуючи безперервний та цілісний користувацький досвід у межах хмарної екосистеми NoteCraft.

Висновки до розділу 2

Проведений аналіз показав:

На ринку існує значний попит на інструмент середньої складності між Notion та Google Keep;

Користувачам потрібна кастомізація, але без складності;

NoteCraft має чітко визначену нішу;

Продукт повністю відповідає сучасним трендам: персоналізація, хмара, колаборація.

Комплексний аналіз ризиків дозволив знизити вірогідність критичних збоїв та забезпечити стабільність проєкту. Команда регулярно оновлювала ризик-реєстр після кожного спринту, що забезпечило адаптивність і відповідність Agile-підходу.

Регулярне тестування, проведене в рамках кожного спринту, забезпечило високу стабільність MVP-версії NoteCraft. Завдяки інтеграції тестування в Agile-процес продукт став більш надійним, масштабованим і готовим до використання кінцевими користувачами.

РОЗДІЛ 3
ГНУЧКЕ УПРАВЛІННЯ РОЗРОБКОЮ ЗАСТОСУНКУ NOTECRAFT

3.1. Вступ до практичної частини

У цьому розділі описано реальний процес розробки застосунку NoteCraft із використанням методології Scrum. Застосунок створювався в умовах навчального проєкту, проте структура та підхід повністю відповідають практиці комерційних команд.

Розробка була поділена на **три спринти**, кожен з яких тривав **два тижні**. За цей час команда проходила всі події Scrum:

- Sprint Planning
- Daily Scrum
- виконання задач
- Sprint Review
- Sprint Retrospective

Також велась робота з артефактами:

- Product Backlog
- Sprint Backlog
- Increment
- Definition of Done
- Scrum Board
- Burndown Chart
- Velocity Chart

Перед початком спринтів було сформовано команду.

3.2. Склад Scrum-команди та функціональний розподіл обов'язків

Ефективність розробки NoteCraft була забезпечена через формування крос-функціональної Scrum-команди, де кожна роль має чітко визначені управлінські та виконавчі функції. Product Owner зосередив свою діяльність на стратегічному управлінні Product Backlog, пріоритезації вимог на основі бізнес-цінності та

взаємодії зі стейкхолдерами для верифікації інкрементів. Scrum Master виконував роль фасилітатора, забезпечуючи дотримання Agile-практик, усуваючи організаційні перешкоди (impediments) та сприяючи професійному розвитку команди.

Група розробки, що складалася з фахівців фронтенд та бекенд напрямів, несла колективну відповідальність за перетворення вибраних задач беклогу в потенційно готовий продукт. Залучення дизайнера та тестувальника безпосередньо в команду розробки дозволило реалізувати концепцію «Shift-Left Testing», коли якість продукту перевіряється на кожному етапі створення, а не після завершення всього циклу розробки.

Scrum-команда складалася з наступних ролей та учасників:

Product Owner (1 особа): Відповідає за пріоритезацію беклогу, комунікацію зі стейкхолдерами та прийняття інкременту.

Scrum Master (1 особа): Фасилітує зустрічі, усуває перешкоди (impediments) та навчає команду принципам Agile.

Development Team:

- Frontend Developers (2 особи): Розробка інтерфейсу на React/React Native, реалізація drag-and-drop;
- Backend Developer (1 особа): Розробка REST API, WebSocket-з’єднань та логіки синхронізації;
- UI/UX Designer (1 особа): Створення макетів, Persona Canvas та проектування користувацького шляху;
- QA Engineer (1 особа): Автоматизоване (Jest) та мануальне тестування функціоналу;
- DevOps (0.5 FTE): Налаштування CI/CD та хмарної інфраструктури (AWS/MongoDB Atlas).

3.3 Lean Canvas

Lean Canvas використовується як інструмент первинного управлінського аналізу продукту та середовища його створення. Його застосування дозволяє

систематизувати ключові гіпотези щодо проблем користувачів, ціннісної пропозиції та механізмів створення й доставки цінності. У межах даної роботи Lean Canvas застосунку NoteCraft слугує основою для формування продуктового бачення та подальшого наповнення Product Backlog.

Основною проблемою, яку вирішує NoteCraft, є обмежена гнучкість існуючих застосунків для створення нотаток. Більшість рішень пропонують стандартний набір функцій, який не враховує індивідуальні робочі процеси користувачів. Це призводить до необхідності використання декількох інструментів одночасно або адаптації власних процесів під можливості програмного забезпечення. NoteCraft пропонує альтернативний підхід, за якого користувач адаптує продукт під себе, а не навпаки.

Цільовою аудиторією продукту є користувачі, які активно працюють з інформацією та потребують інструментів для її структурування. До них належать студенти, фахівці креативних професій, менеджери проєктів, команди, що працюють віддалено, а також користувачі, які використовують нотатки у повсякденному житті. Для всіх цих груп важливими є гнучкість, зручність та можливість синхронізації даних між пристроями.

Ціннісна пропозиція NoteCraft полягає у поєднанні простоти базового використання з розширеними можливостями кастомізації. Продукт дозволяє створювати нотатки з інтерактивними елементами, такими як таймери, дедлайни, нагадування та кастомні звуки, а також надавати доступ до нотаток іншим користувачам для спільної роботи. Це створює додаткову цінність у порівнянні з традиційними застосунками для нотаток.

Lean Canvas (рис 3.1) також відображає ключові канали взаємодії з користувачами, серед яких мобільні та веб-застосунки, а також інтеграція з хмарними сервісами. З управлінської точки зору це дозволяє оцінити потенційні точки контакту з клієнтами та спланувати подальший розвиток продукту. Таким чином, Lean Canvas є базовим інструментом, що поєднує стратегічне бачення продукту з управлінськими рішеннями щодо його реалізації.

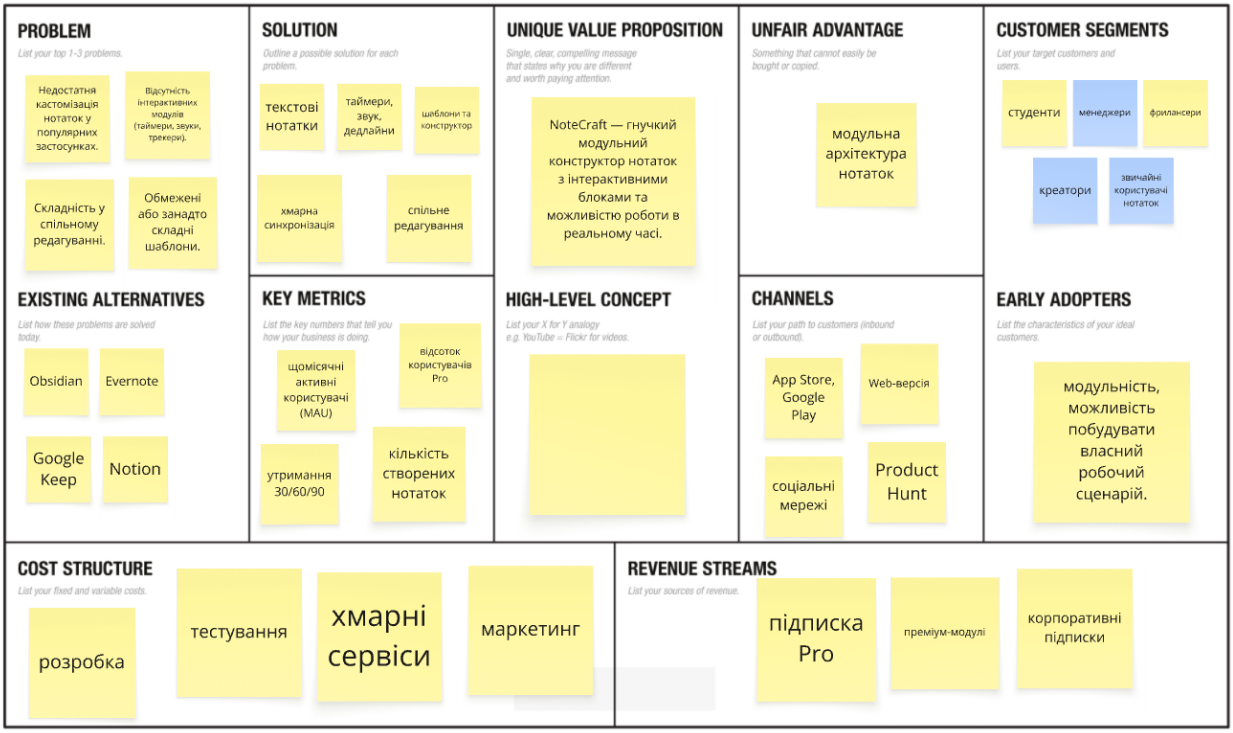

Рисунок 3.1 – Lean canvas

Джерело: Розроблено автором

3.4. Product Vision Canvas

Product Vision Canvas (рис 3.2) використовується для формування довгострокового бачення продукту та визначення його стратегічного напрямку розвитку. На відміну від Lean Canvas, який зосереджується на проблемах і гіпотезах, Product Vision Canvas фокусується на майбутньому стані продукту та очікуваних результатах його використання.

У межах даної роботи Product Vision Canvas NoteCraft визначає, для кого створюється продукт, яку основну проблему він вирішує та якою має бути його цінність у довгостроковій перспективі. Центральною ідеєю продукту є створення універсального середовища для управління нотатками, яке може адаптуватися під різні сценарії використання – від особистих записів до командної роботи.

Продукт орієнтований на користувачів, які цінують гнучкість і контроль над власними процесами. NoteCraft має не лише забезпечувати збереження інформації, а й підтримувати користувача у плануванні діяльності, організації задач та співпраці з іншими людьми. Це визначає стратегічний фокус продукту та впливає на прийняття управлінських рішень щодо його розвитку.

Очікуваним результатом використання NoteCraft є підвищення ефективності роботи з інформацією та зменшення когнітивного навантаження на користувачів. Завдяки можливості кастомізації кожен користувач отримує інструмент, що відповідає його індивідуальним потребам, без необхідності використання сторонніх сервісів.

Product Vision Canvas також визначає критерії успіху продукту, серед яких стабільність роботи, зручність користування, масштабованість та можливість подальшого розвитку. З управлінської точки зору це дозволяє узгодити роботу команди з довгостроковими цілями та забезпечити цілісність продукту на всіх етапах його створення.

Таким чином, Product Vision Canvas є ключовим стратегічним інструментом управління продуктом NoteCraft, що забезпечує логічний перехід від абстрактного бачення до конкретних управлінських та операційних рішень, які реалізуються у Product Backlog та спринтах Scrum.

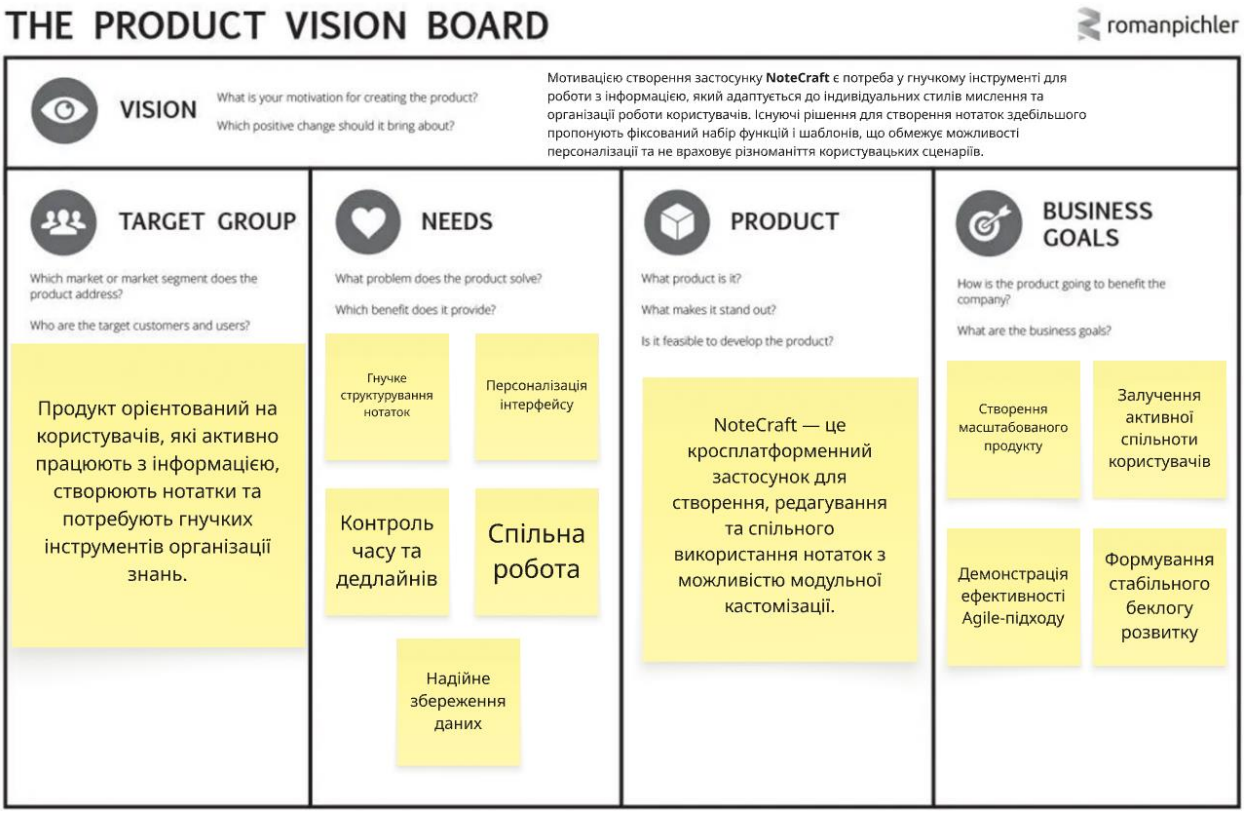

Рисунок 3.2 – Product Vision Canvas

Джерело: Розроблено автором

3.5 Формування дерева продукту на основі Product Vision

На основі Product Vision було сформовано дерево продукту, яке відображає логічну структуру функціональності NoteCraft. Дерево продукту дозволяє візуалізувати продукт як систему взаємопов’язаних компонентів і забезпечує перехід від абстрактного бачення до конкретних функцій.

Коренем дерева продукту є основна цінність NoteCraft – гнучке управління нотатками. Від неї відходять ключові функціональні напрями, зокрема створення та редагування нотаток, кастомізація, спільна робота та хмарна синхронізація. Кожен із цих напрямів деталізується на рівні конкретних можливостей, які згодом трансформуються у користувацькі історії.

Використання дерева продукту з управлінської точки зору дозволяє забезпечити узгодженість між стратегічним баченням і операційними завданнями команди. Саме на основі цієї структури формується Product Backlog. Дерево продукту візуалізує ієрархію функцій, виходячи з Product Vision.

- Корінь: NoteCraft – система гнучкого управління нотатками.
    - Гілка 1: Ядро (Core)
        - Авторизація та безпека (JWT, профілі);
        - Текстовий редактор (створення, редагування, видалення);
        - Локальне сховище.
    - Гілка 2: Модульний конструктор (Customization)
        - Блоки таймерів та дедлайнів;
        - Чеклісти та списки;
        - Мультимедійні вкладення (аудіо, зображення).
    - Гілка 3: Колаборація та Хмара (Infrastructure)
        - Хмарна синхронізація (MongoDB Atlas);
        - Спільний доступ (права: Read/Edit);
        - WebSocket-з’єднання для реального часу.

3.6. Формування Product Backlog та Product Roadmap

Product Backlog є центральним артефактом Scrum, який містить перелік усіх вимог до продукту у вигляді користувацьких історій. Для NoteCraft Product Backlog сформовано як логічне продовження Product Vision та дерева продукту.

Усі елементи беклогу орієнтовані на створення цінності для користувача та відображають різні сценарії використання продукту. Пріоритизація беклогу здійснюється Product Owner з урахуванням стратегічних цілей продукту, складності реалізації та очікуваного ефекту для користувачів.

Таким чином, Product Backlog виступає не лише переліком задач, а й інструментом стратегічного управління розвитком NoteCraft. Нижче наведено Product Backlog (рис 3.3).

Product Roadmap наведена у табл 3.1.

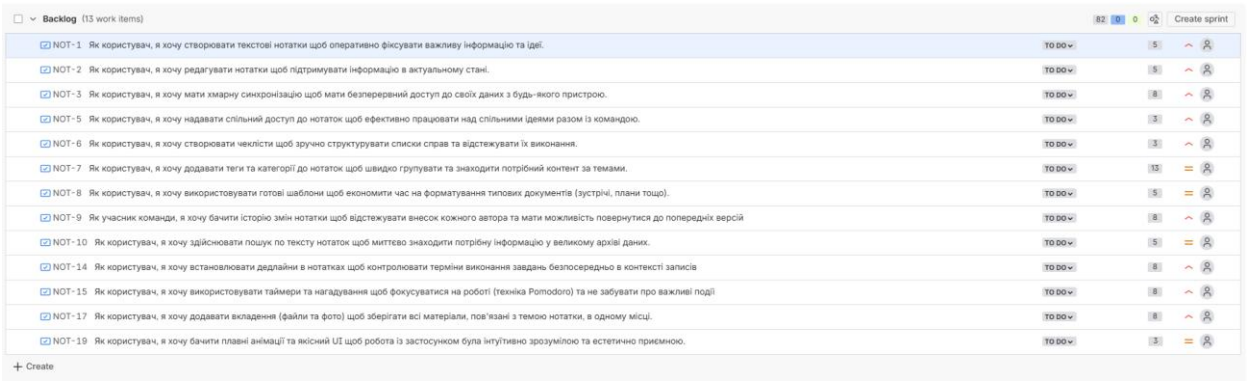

Рисунок 3.3 – Product Backlog

Джерело: розроблено автором

Таблиця 3.1 – Product Roadmap

Етап розробки	Віха (milestone)	Ціль спринту та ключові результати	Ключові User Stories (Backlog)
Sprint 1	Foundational MVP	Побудова архітектурного ядра та реалізація базового циклу роботи з текстовим контентом.	NOT-1, NOT-2, NOT-6

Sprint 2	Connected Ecosystem	Перетворення локального інструменту на хмарну платформу з можливістю колективної роботи.	NOT-3, NOT-5, NOT-7, NOT-10
Sprint 3	Advanced Productivity	Впровадження інтерактивних модулів управління часом та фінальна стабілізація продукту.	NOT-14, NOT-15, NOT-17, NOT-18

Опис віх розробки (Milestones)

Milestone 1: Foundational MVP (Sprint 1)

Цей етап відповідає «кореню» та першій гілці дерева продукту – створенню надійного ядра системи.

- Ціль. Сформувати працездатний прототип, що забезпечує безпеку даних та базові операції.
- Результати:
  - Налаштована архітектура клієнт-сервер та база даних MongoDB;
  - Система авторизації користувачів через JWT-токени;
  - Функціонал створення, редагування та локального збереження текстових нотаток і чеклістів .

Milestone 2: Connected Ecosystem (Sprint 2)

Віха фокусується на гілці «Інфраструктура», розширюючи можливості застосунку до рівня мережевого інструменту.

- Ціль. Впровадити безперервну синхронізацію та інструменти для командної взаємодії.
- Результати:
  - стабільна хмарна синхронізація даних між різними пристроями користувача;
  - механізм спільного доступу (права: Read/Edit) та WebSocket-з’єднання для редагування в реальному часі;
  - система тегів, категорій та глобальний пошук по тексту для швидкої навігації.

Milestone 3: Advanced Productivity & Launch (Sprint 3)

Фінальна віха реалізує гілку «Модульний конструктор», додаючи унікальну ціннісну пропозицію продукту.

- Ціль. Завершити MVP-версію через додавання інтерактивних блоків та оптимізацію інтерфейсу.
- Результати:
  - модулі таймерів та дедлайнів із системою сповіщень;
  - бібліотека готових шаблонів та можливість додавання мультимедійних вкладень;
  - повністю стабілізований бекенд та оптимізований UI з плавними анімаціями .

3.7. Приклад деталізованої User Story

Story. NOT-14 – Впровадження дедлайнів у нотатках

Опис. Як користувач, я хочу мати можливість встановлювати дату та час завершення для окремих блоків нотатки, щоб я міг візуально контролювати свою завантаженість та отримувати сповіщення про наближення термінів.

Критерії прийняття (Acceptance Criteria): 1. Користувач може вибрати дату та час через віджет календаря всередині нотатки. 2. Встановлений дедлайн відображається у верхній частині нотатки як кольоровий тег (зелений – вчасно, червоний – прострочено). 3. За 15 хвилин до настання дедлайну система має надіслати push-сповіщення (на мобільний) або системне сповіщення (у браузері). 4. Користувач може видалити або змінити дедлайн у будь-який момент. 5. Інформація про дедлайн успішно синхронізується з хмарною базою даних MongoDB.

Технічні замітки:

- використати бібліотеку date-fns для обробки форматів часу;
- реалізувати серверне сповіщення через Node.js Cron Jobs.

Оцінка: 8 Story Points.

3.8. Sprint 1: Опис та реалізація

Sprint 1: Sprint Goal (рис 3.4). Основною метою першої ітерації було закладення технічного фундаменту системи та реалізація базового циклу роботи з контентом . У межах спринту команда фокусувалася на історіях NOT-1 та NOT-2, щоб користувач міг оперативно фіксувати ідеї та підтримувати їх в актуальному стані. Також важливо було впровадити історію NOT-6, що дозволило структурувати списки справ безпосередньо в нотатках.

На Sprint Planning команда обрала такі user stories із Product Backlog:

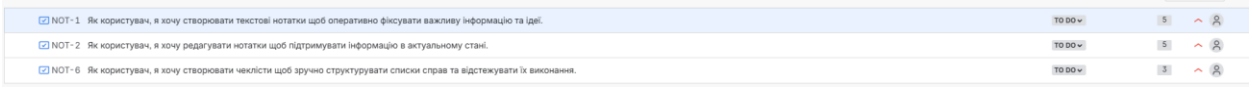

Рисунок 3.4 – Sprint 1

Джерело: Розроблено автором

Sprint Planing. Під час планування команда провела декомпозицію обраних User Stories на конкретні технічні завдання: розробку API-ендпоїнтів, створення React-компонентів та налаштування схем у MongoDB. Було визначено загальну складність спринту у 34 story points. Окрему увагу приділили налаштуванню сервісу авторизації (NOT-12 – реєстрація та логін), що стало критичним кроком для безпечного збереження даних користувачів.

Daily Scrum. Щоденні мітинги забезпечували синхронізацію між розробниками та вчасне виявлення перешкод. Зокрема, обговорювалися питання інтеграції UI з бекендом та тестування JWT-токена . Одним із блокерів стала помилка у JWT після реєстрації, яку вдалося швидко усунути завдяки фасилітації Scrum-майстра, що організував коротку зустріч для уточнення форматів даних між backend та frontend спеціалістами.

Definition of Done (Критерії готовності) Робота вважалася завершеною за умови повної інтеграції клієнтської частини з базою даних та успішного проходження юніт-тестів. Кожна функція мала відповідати UI-макетам та стабільно зберігати дані локально.

Burndown Chart Sprint 1 (рис 3.5). Burndown Chart демонструє швидкість виконання задач протягом спринту[15]. Аналіз графіка показав стрімку динаміку виконання завдань: команда завершила всі активності на 12-й день замість 14-ти. Це свідчить про високу якість попереднього планування та відсутність серйозних технічних проблем на старті.

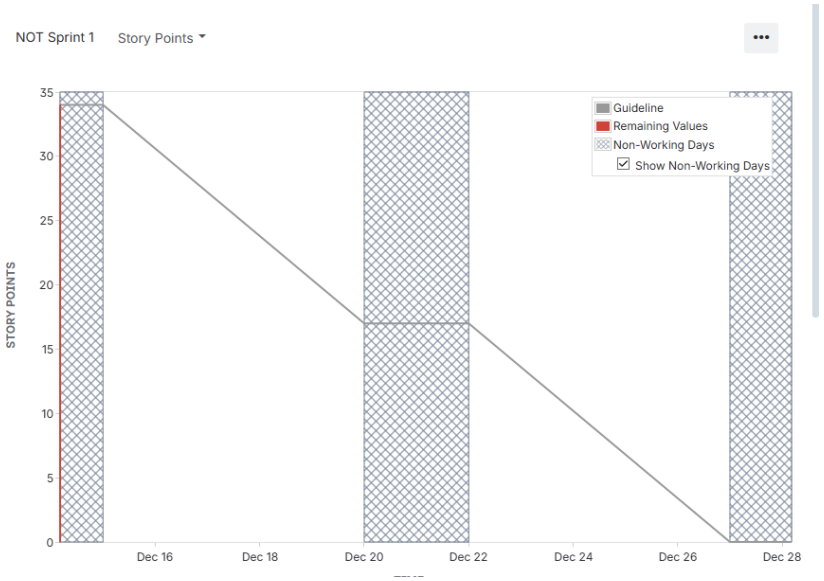

Рисунок 3.5 – Burndown Chart Sprint 1

Джерело: Розроблено автором

Velocity (Спринт 1) (рис 3.6). Пропускна здатність команди склала 34 Story Points . Цей показник став еталонним для планування подальшого навантаження.

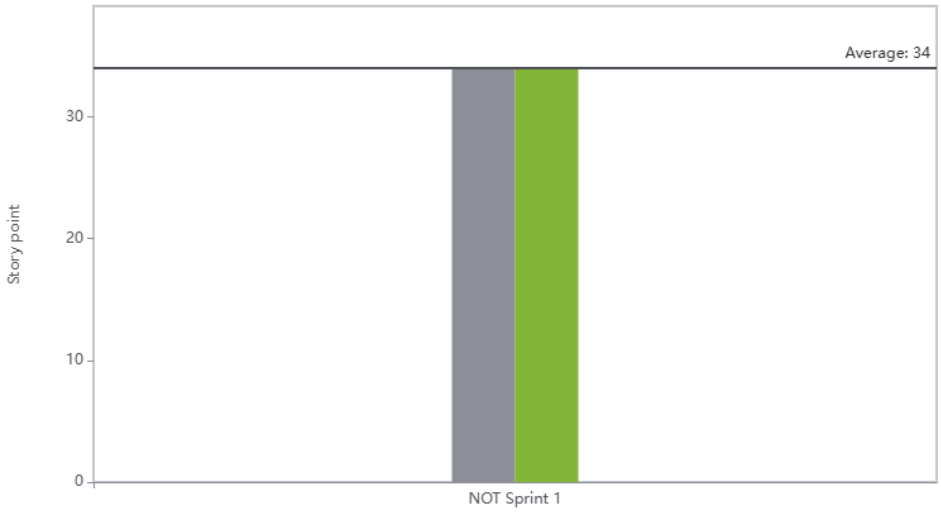

Рисунок 3.6 – Velocity після першого спринта

Джерело: Розроблено автором

Sprint Review (Підсумки Sprint 1). Sprint Review першого спринту було присвячено демонстрації створеного інкременту продукту та перевірці досягнення поставленої цілі спринту. Команда представила базовий функціонал застосунку, що дозволяв створювати та редагувати нотатки, а також продемонструвала загальну архітектуру рішення. Product Owner оцінив відповідність реалізованого функціоналу вимогам беклогу та надав зворотний зв’язок щодо подальших напрямів розвитку продукту. Результати Sprint Review стали основою для оновлення Product Backlog та уточнення пріоритетів наступного спринту.

Feedback від Product Owner:

- додати більше варіантів шрифтів;
- змінити колір кнопок відповідно до бренду;
- покращити відтворення чеклістів.

Стейкхолдери схвалили роботу, і було вирішено інтегрувати синхронізацію та колаборацію у наступному спринті.

Sprint Retrospective

Під час Sprint Retrospective команда проаналізувала власну роботу в межах першого спринту, зосередившись як на досягненнях, так і на труднощах. Було відзначено позитивний вплив регулярної комунікації та спільного прийняття рішень. Водночас команда дійшла висновку про необхідність покращення оцінювання завдань та більш раннього врахування аспектів тестування. Напрацьовані висновки стали основою для вдосконалення процесів у наступних спринтах.

3.9. Sprint 2: Реалізація синхронізації, колаборації та пошуку

Sprint Goal. Планування другого спринту відбувалося з урахуванням результатів попереднього етапу та зворотного зв’язку, отриманого під час Sprint Review. Основною ціллю спринту було розширення функціональності NoteCraft шляхом впровадження хмарної синхронізації та можливостей спільного доступу

до нотаток. Під час Sprint Planning Product Backlog було повторно пріоритезовано, а до Sprint Backlog включено завдання, що мали найбільшу цінність для користувачів. Команда також приділила увагу потенційним технічним ризикам та узгодила підхід до їх мінімізації.

Sprint Planning. Сесія планування була тривалішою (2 години) через високу складність архітектурних рішень для WebSockets та синхронізації в реальному часі (NOT-13) . Було заплановано навантаження у 48 SP, що перевищувало попередню швидкість команди, проте розробники погодилися на це, враховуючи наявні напрацювання з першого спринту.

На Sprint Planning команда відібрала такі user stories (рис 3.7).

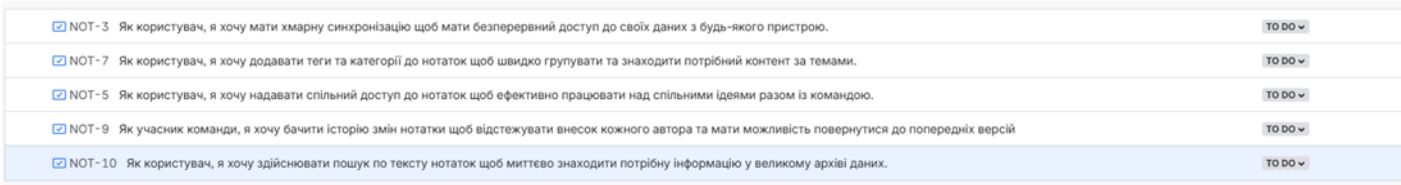

Рисунок 3.7 – Sprint Backlog 2

Джерело: Розроблено автором

Daily Scrum (Щоденні зустрічі) У цьому спринті Daily Scrum стали критичним інструментом управління технічними ризиками. Команда координувала дії щодо впровадження хмарних функцій та обробки конфліктів версій при синхронізації . Саме під час щоденної зустрічі було виявлено критичний дефект – дублювання нотаток на різних пристроях, що дозволило негайно пріоритезувати виправлення цього багу.

Definition of Done

- хмарна синхронізація працює стабільно на різних пристроях;
- WebSocket-підключення стабільне;
- спільний доступ реалізований через списки дозволів;
- теги зберігаються та фільтруються;
- пошук працює мінімум за заголовками й тегами;
- усі модулі пройшли тестування;

– без критичних багів.

Burndown Chart Sprint 2 (рис 3.8). Графік мав більш напружений характер, відображаючи труднощі з реалізацією синхронізації. Проте команда змогла вирівняти прогрес до кінця другої ітерації.

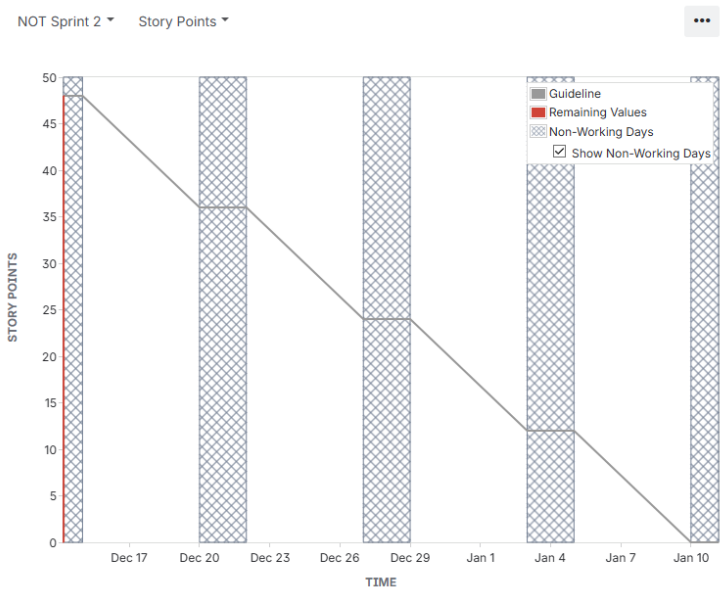

Рисунок 3.8 - Burndown Chart Sprint 2

Джерело: Розроблено автором

Velocity Sprint 2. (рис 3.9). Швидкість розробки зросла до 44 Story Points, що свідчить про покращення командної взаємодії та зрілість процесів розробки .

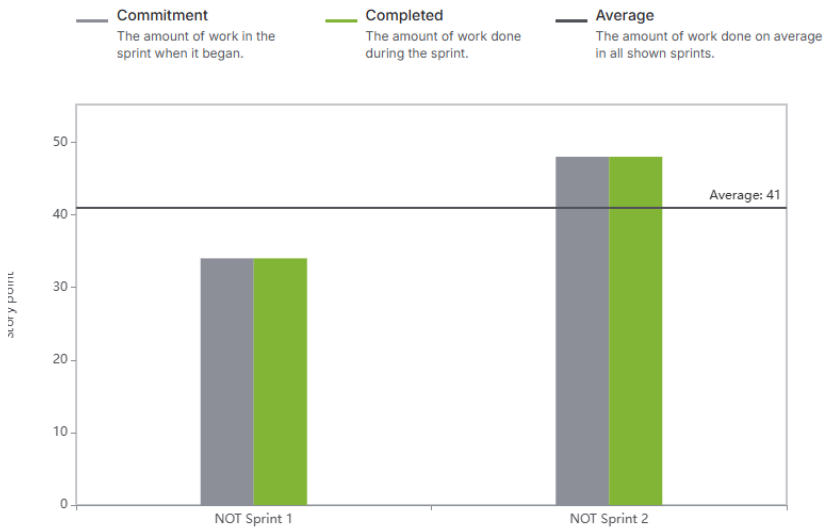

Рисунок 3.9 – Velocity після спринту 2

Джерело: Розроблено автором

Sprint Review Sprint 2. Sprint Review другого спринту було спрямовано на оцінювання нових можливостей продукту та їх відповідності очікуванням користувачів. Команда продемонструвала реалізовану хмарну синхронізацію та механізми спільного доступу до нотаток. Отриманий зворотний зв'язок дозволив виявити додаткові вимоги до безпеки та зручності використання, які були зафіксовані в Product Backlog для подальшої реалізації.

Sprint Retrospective Sprint 2. Ретроспектива другого спринту показала зростання зрілості команди та покращення процесів взаємодії. Було відзначено ефективність спільного вирішення складних завдань, а також необхідність більш детального технічного планування. Команда дійшла висновку, що попереднє опрацювання архітектурних рішень дозволяє зменшити кількість переробок і підвищити якість кінцевого результату.

3.10. Sprint 3: Таймери, дедлайни, покращення UI та стабілізація

Sprint Goal. Метою фінальної ітерації було завершення MVP-версії через впровадження інструментів контролю часу: дедлайнів (NOT-14) та таймерів Pomodoro (NOT-15).

Sprint Planning (Планування спринту) Планування було зосереджене на якості та стабільності продукту. Команда свідомо обмежила обсяг нових функцій, зосередившись на реалізації шаблонів (NOT-16), вкладень (NOT-17) та UI-полірингу (NOT-18), щоб забезпечити завершеність MVP. Планове навантаження склало 37 SP. Sprint Backlog 3 наведено на рис 3.10.

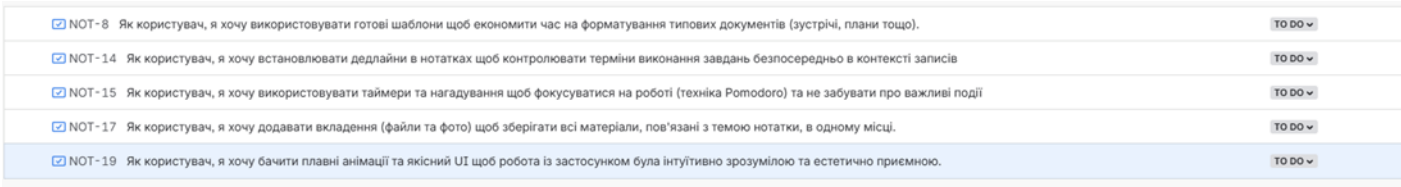

Рисунок 3.10 – Sprint Backlog 3

Джерело: Розроблено автором

Daily Scrum Зустрічі мали аналітичний характер і були спрямовані на контроль результатів тестування. Команда щоденно обговорювала виявлені

дефекти та результати перевірки навантаження (до 50 одночасних клієнтів), що дозволило вчасно оптимізувати роботу сервера.

Definition of Done

- функціонал вставляється у нотатку через конструктор;
- працює в браузері та мобільній версії;
- передає дані у хмару без помилок;
- UI відповідає бренду продукту;
- пройдено як мінімум 10 тестових сценаріїв;
- інтегровано в основну систему.

Burndown Chart Sprint 3 (рис 3.11). На графіку спостерігалося плавне виконання робіт. Команда сфокусувалася на фінальних штрихах та виправленні дрібних недоліків інтерфейсу.

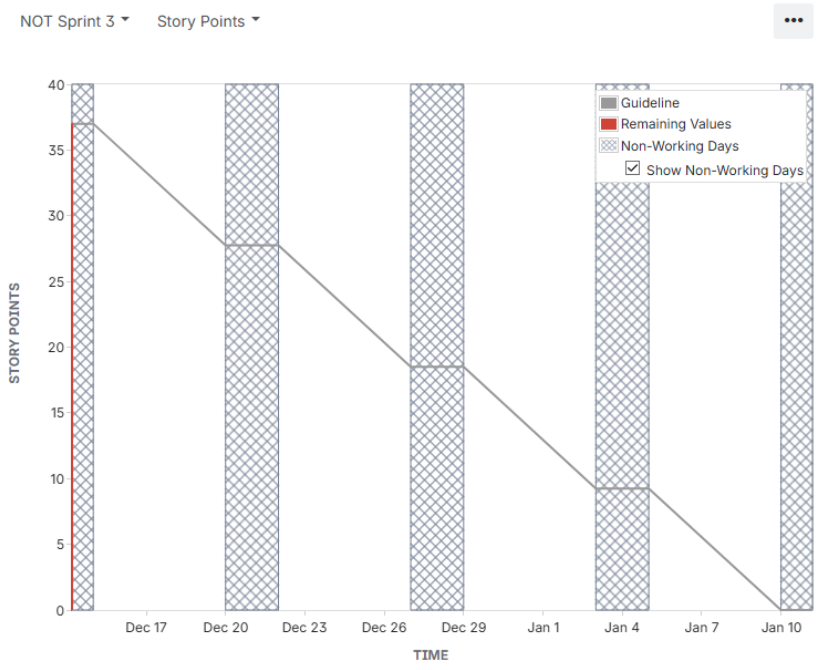

Рисунок 3.11 – Burndown Chart Sprint 3

Джерело: Розроблено автором

Velocity Sprint 3 (рис 3.12). Velocity знизилося до 28 Story Points . Це пояснюється фокусом на стабілізації системи та «поліруванні» продукту перед релізом, що вимагало значних зусиль від QA та фронтенд-команди.

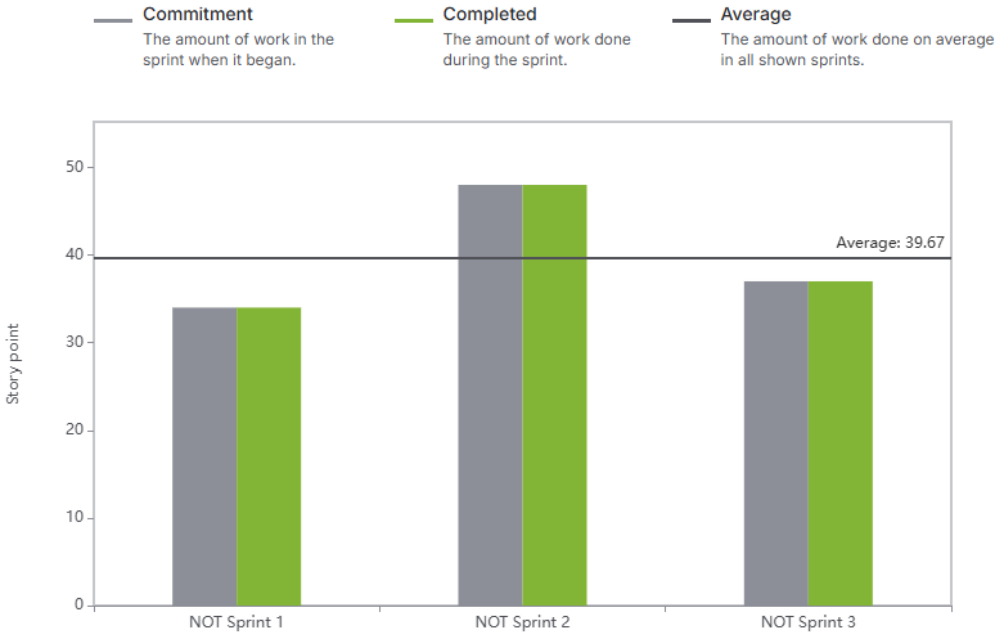

Рисунок 3.12 – Velocity після спринту 3

Джерело: Розроблено автором

Sprint Review Sprint 3. Під час огляду було представлено повністю укомплектований MVP NoteCraft . Product Owner підтвердив, що продукт відповідає всім ключовим вимогам і готовий до демонстрації стейкхолдерам.

Sprint Retrospective Sprint 3. На підсумковій ретроспективі команда підтвердила доцільність обраної методології Scrum, відзначивши значне покращення внутрішньої комунікації та точність планування наприкінці проєкту

3.11. Кошторис проєкту

Таблиця 3.2 – Кошторис проєкту

№	Стаття витрат	Підкатегорія	К-сть	Вартість / міс (грн)	Тривалість	Сума (грн)
1	Команда	Product Owner	1	65 000	6 міс	390 000
		Scrum Master	1	55 000	6 міс	330 000
		Front-end Developer	2	70 000	6 міс	840 000
		Back-end Developer	1	80 000	6 міс	480 000

		UI/UX Designer	1	60 000	6 міс	360 000
		QA Engineer	1	50 000	6 міс	300 000
		DevOps (0.5 FTE)	0.5	45 000	6 міс	270 000
-	Разом за зарплатами	-	-	-	-	2 970 000
1.2	Податки та накладні витрати	ЄСВ (22%)	-	-	-	653 400
		Бухгалтерський супровід	-	2000	6 міс	12 000
		Офіс / Коворкінг	-	10 000	6 міс	60 000
-	Разом накладні витрати	-	-	-	-	725 400
2	Інструменти розробки	GitHub Team	6	-	6 міс	5 800
		Jira Cloud	6	-	6 міс	11 300
		Figma Pro	1	-	6 міс	3 600
		Slack Pro	6	-	6 міс	11 600
-	Разом інструменти	-	-	--	-	32 300
3	Оренда обладнання	Лаптопи	6	7 000	6 міс	42 000
-	Разом обладнання	-	-	-	-	42 000
4	Інфраструктура	Сервери (AWS/GC)	-	-	6 міс	12 000

Таблиця 3.3. – Кошторис проєкту

		MongoDB Atlas	-	-	6 міс	9 000
		Cloud Storage	-	-	6 міс	4 800
		CDN/Load Balancer	-	-	6 міс	6 000
		Моніторинг	-	-	6 міс	5 400

		Backups	-	-	6 міс	3 600
-	Разом інфраструктура	-	-	-	-	40 800
5	Тестування	Емулятори, середовища	-	-	-	8 000
		Пристрої для тестів	-	-	-	12 000
		BrowserStack	1	-	6 міс	7 000
-	Разом тестування	-	-	-	-	27 000
6	Маркетинг і запуск	Промосторінка	-	-	-	8 000
		Реклама	-	-	-	20 000
		ASO App Store/Play	-	-	-	12 000
		PR / Product Hunt	-	-	-	6 000
-	Разом маркетинг	-	-	-	-	46 000
7	Непередбачувані витрати	10% від бюджету		-	-	388 750
-	ЗАГАЛЬНИЙ БЮДЖЕТ ПРОЄКТУ	-	-	-	-	4 272 250 грн

Висновки до розділу 3

У розділі представлено повну модель гнучкого управління розробкою застосунку NoteCraft на основі Scrum. Сформовано структуру продукту, розроблено Product Backlog і Roadmap, визначено процеси спринтів, описано архітектуру, ресурсне забезпечення та календарний план. Проведені розрахунки свідчать про економічну доцільність створення застосунку та високу перспективність його подальшого розвитку.

ВИСНОВКИ

У магістерській роботі розроблено комплексну модель створення застосунку NoteCraft із використанням методів гнучкого управління проєктами. Було проаналізовано ринок застосунків для нотаток, визначено ключові потреби користувачів та сформовано концепцію продукту, яка поєднує модульність, інтерактивність та доступність.

Застосування Scrum забезпечило структурований процес розробки, що включав формування Product Backlog, планування спринтів, визначення ролей та артефактів. Побудовано календарний план, структуру робіт, визначено ресурсне забезпечення та проведено розрахунок витрат і прогноз доходів.

Проведений аналіз підтверджує економічну доцільність проєкту, а модульний характер продукту забезпечує його масштабованість та конкурентоспроможність. NoteCraft має значний потенціал для подальшого розвитку й може зайняти нішу інструментів для персональної продуктивності та інтерактивних нотаток.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Schwaber K., Sutherland J. The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. Scrum.org, 2020. URL: https://scrumguides.org/scrum-guide.html (дата звернення: 16.12.2025).

2. Beck K. et al. Manifesto for Agile Software Development. Agile Alliance, 2001. URL: https://agilemanifesto.org/ (дата звернення: 16.12.2025).

3. Maurya A. Lean Canvas: The 1-page business modeling tool. Leanstack. URL: https://leanstack.com/lean-canvas (дата звернення: 16.12.2025).

4. Pichler R. The Product Vision Board. RomanPichler.com. URL: https://www.romanpichler.com/tools/product-vision-board/ (дата звернення: 16.12.2025).

5. Salazar K. Personas: Why and How You Should Use Them. Nielsen Norman Group. 2022. URL: https://www.nngroup.com/articles/persona/ (дата звернення: 16.12.2025).

6. Allen D. Getting Things Done: The Art of Stress-Free Productivity. Penguin Books, 2015. 352 p.

7. Business Research Insights. Note Taking App Market Size, Share, Growth, Trends. 2023. URL: https://www.businessresearchinsights.com/market-reports/note-taking-app-market-106125 (дата звернення: 16.12.2025).

8. React Documentation. The library for web and native user interfaces. Meta Open Source. URL: https://react.dev/ (дата звернення: 16.12.2025).

9. Node.js Documentation. OpenJS Foundation. URL: https://nodejs.org/en/docs/ (дата звернення: 16.12.2025).

10. MongoDB Documentation. Documents & Collections. MongoDB, Inc. URL: https://www.mongodb.com/docs/manual/core/document/ (дата звернення: 16.12.2025).

11. MDN Web Docs. The WebSocket API. Mozilla Developer Network. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (дата звернення: 16.12.2025).

12. Cohn M. Planning Poker: Agile Estimating Made Easy. Mountain Goat Software. URL: https://www.mountaingoatsoftware.com/tools/planning-poker (дата звернення: 16.12.2025).

13. OWASP. OWASP ZAP: The World’s Most Popular Free Security Scanner. OWASP Foundation. URL: https://www.zaproxy.org/ (дата звернення: 16.12.2025).

14. Jest. Delightful JavaScript Testing. Meta Open Source. URL: https://jestjs.io/ (дата звернення: 16.12.2025).

15. Atlassian. Burndown Charts. Atlassian Agile Coach. URL: https://www.atlassian.com/agile/tutorials/burndown-charts (дата звернення: 16.12.2025).