ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Гнучке управління створенням платформи для моніторингу та раннього виявлення аномалій «NOS9» для ІТ-компанії»

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 Володимир ЩЕРБІНІН

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
Щербінін Володимир Дмитрович

Тема роботи	Гнучке управління створенням платформи для моніторингу та раннього виявлення аномалій «NOS9» для IT-компанії
Номер та дата наказу про затвердження теми	№ 109-2 від 14 жовтня 2025 року.
Коротка постановка завдання	Обґрунтування бачення створюваного продукту для розв'язання проблеми в діяльності замовника на основі розробки моделі його організації. Детальний опис особливостей гнучкого управління створенням платформи «NOS9» для IT-компанії з використанням фреймворка Скрам. Розкриття особливостей інженерного менеджменту для гнучкого управління створенням платформи «NOS9».
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1. Мушинський О. Ю. Розвиток лідерства в управлінні проєктними командами. *Вчені записки Університету «КРОК». 2024. №* 76. https://doi.org/10.31732/2663-2209-2024-76-165-173 2. Lukutin O., Michkivskyy S. AI AS a catalyst of organizational agility: structural, cultural and leadership implications. Лідерство, бізнес-процеси та стале майбутнє: зб. матеріалів наук.-практ. конф. (м. Київ, 27 вересня 2025 р.). Київ: Університет економіки та права «КРОК» / Бізнес Школа КРОК, 2025. С.86-88 3. K.-B. Ooi, "The metaverse in engineering management: Overview, opportunities, challenges, and future research agenda," *IEEE Trans. Eng. Manag.*, vol. 71, pp. 13882–13889, Aug. 2024.
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 «16» жовтня 2025 р.

Керівник Олег ЛУКУТІН
Керівник Сергій МІЧКІВСЬКИЙ
Здобувач Володимир ЩЕРБІНІН

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.	14.10.2025 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу.	16.10.2025 р.	*виконано*
Основний етап			
5	Розробка концепції та змісту кваліфікаційної роботи, погодження їх з науковим керівником	06.11.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження.	08.11.2025 р.	*виконано*
7	Теоретико-методичний аналіз предметної області. Підготовка та подання керівнику розділу 1 кваліфікаційної роботи.	13.11.2025 р.	*виконано*
8	Реалізація гнучкого управління розробкою продукту. 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 урахуваннямзауважень 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 Володимир ЩЕРБІНІН

Щербінін В.Д. Гнучке управління створенням платформи для моніторингу та раннього виявлення аномалій «NOS9» для IT-компанії

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, 2026 р.

Сформульовано візію продукту NOS9 та бізнес-проблему пізнього виявлення деградації програмного коду. На основі Product Vision Board, Lean Canvas, персон і User Stories визначено потреби ключових користувачів та вимоги до продукту. Реалізовано функції гнучкого управління створенням SaaS-платформи за допомогою практик Agile, Scrum, Kanban і Lean Startup. Побудовано продуктову дорожню карту, backlog, структуру MVP та критерії прийняття інкрементів. Описано архітектуру MVP, процеси CI/CD, інструменти DevOps та підходи до alpha-тестування. Систематизовано м’які навички та управлінські компетентності, необхідні для роботи з гнучкими методами в інженерних командах.

Ключові слова: Agile, Scrum, Kanban, Lean Startup, observability, аномалії, SaaS, DevOps, інженерний менеджмент.

Табл. 15. Рис 17. Бібліограф.: 23 найм.

Shcherbinin V. Flexible management of creating the anomaly-detection and monitoring platform "NOS9" for an IT company

Qualification paper explanatory note in specialty 073 – Management (educational program – Agile Software Development Technologies), Master's Degree. – HEI «University of Economics and Law «KROK», Educational and Scientific Institute of Information and Communication Technologies, Department of Information Management, Mathematics and Statistics, Kyiv, 2026.

The business vision of the NOS9 product is formulated. The problem of late detection of codebase degradation is identified, and user needs are defined on the basis of the Product Vision Board, Lean Canvas, personas and User Stories. The functions of agile management in developing a SaaS observability platform are implemented using Scrum, Kanban and Lean Startup. A product roadmap, backlog, MVP scope and acceptance criteria are created. The MVP architecture, CI/CD processes, DevOps tooling and alpha-testing approach are described. The soft skills and managerial competencies required for agile work in engineering teams are classified.

Keywords: Agile, Scrum, Kanban, Lean Startup, observability, anomalies, SaaS, DevOps, Engineering Management.

Tabl. 15. Fig. 17. Bibliography: 23 Items.

ЗМІСТ

ВСТУП

Актуальність теми дослідження зумовлена зростанням потреб сучасних IT-компаній у впровадженні високоефективних рішень для моніторингу та раннього виявлення аномалій у роботі інформаційних систем. Ускладнення архітектури програмних продуктів, поява розподілених хмарних середовищ, а також динаміка релізних циклів спричиняють збільшення кількості прихованих дефектів, що виявляються лише на пізніх етапах експлуатації. Запізніле виявлення деградації стану кодової бази призводить до суттєвих виробничих витрат, збільшення часу інцидент-менеджменту та зниження стабільності систем. Саме тому виникає потреба у створенні інструментів, здатних забезпечувати раннє виявлення аномалій і підтримувати високу продуктивність команд розробки та експлуатації

У таких умовах застосування гнучких методів управління, зокрема Agile practices (гнучкі практики), набуває вирішального значення для швидкої адаптації до змін, інтеграції нових інструментів і мінімізації операційних ризиків. Особливу роль відіграють платформи моніторингу, що надаються за моделлю програмного забезпечення як сервісу (Software as a Service, SaaS), оскільки вони забезпечують масштабованість, швидкість розгортання та оптимізацію витрат порівняно з традиційними інструментами спостережуваності.

У рамках цієї роботи увагу зосереджено на платформі «NOS9» (Next-gen Observability System), яка розробляється як SaaS-продукт, орієнтований на оперативне виявлення та аналіз аномалій у режимі реального часу, інтеграцію в робочі процеси, а також підтримку розробників під час написання коду. Такий підхід дозволяє компаніям-користувачам платформи значно знизити операційні витрати, забезпечити проактивне управління якістю сервісів, уникнути або мінімізувати простої систем.

Мета роботи полягає у виявленні особливостей та практичному впровадженні функцій гнучкого управління для створення та інтеграції

платформи моніторингу «NOS9», що надається як SaaS-продукт, для раннього виявлення та попередження аномалій в IT-компаніях.

Для досягнення поставленої мети визначено такі **завдання дослідження**:

- вивчити сучасний стан, основні проблеми та найкращі практики моніторингу й виявлення аномалій в провідних IT-компаніях;
- запропонувати і обґрунтувати рішення платформи «NOS9» із застосуванням гнучких методів Scrum, розрахованої на SaaS-модель використання;
- описати особливості гнучкого управління створенням мінімально життєздатного продукту (MVP) платформи NOS9 та експериментально апробувати її на реальних кейсах у середовищі IT-компаній;
- провести оцінювання ефективності запропонованого рішення, його впливу на операційну продуктивність і витрати компаній-користувачів.

Об'єктом дослідження виступає гнучке управління процесами створення, впровадження та інтеграції SaaS-продуктів у діяльності сучасних IT-компаній.

Предметом дослідження є процеси управління розробкою, інтеграцією та експлуатацією SaaS-платформи моніторингу й раннього виявлення аномалій у роботі інформаційних систем («NOS9») в умовах використання Agile-методологій.

Методи дослідження включають аналіз теоретичних джерел щодо Agile-методологій та SaaS-рішень, системний аналіз артефактів продукту NOS9, моделювання бізнес-процесів, застосування HADI-циклів, використання інструментів продуктового та проєктного менеджменту (roadmap, backlog, OKR), а також експериментальне тестування MVP у реальному середовищі експлуатації. з подальшим аналізом отриманих результатів.

Практична значущість дослідження полягає у створенні ефективної платформи «NOS9», здатної скоротити операційні витрати компаній до 30% завдяки своєчасному виявленню та попередженню проблем у роботі програмних систем. Впровадження NOS9 за моделлю SaaS дозволить компаніям-користувачам швидко адаптувати платформу до власних потреб з мінімальними інфраструктурними витратами.

Апробація результатів здійснювалась під час проведення Міжнародної V Наукової конференції Університету «КРОК» «Сучасний менеджмент організації: витоки, реалії та перспективи розвитку» (м. Київ, 17 квітня 2025 року).

Структура та обсяг роботи. Робота складається зі вступу, трьох розділів, висновків до розділів, загального висновку, списку посилань та додатків. Загальний обсяг роботи 68 сторінок, обсяг основного тексту 65 сторінок.

РОЗДІЛ 1
ДИЗАЙН БІЗНЕСУ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз підприємства

Компанія, що розробляє продукт NOS9 є міжнародною організацією, яка розробляє SaaS платформу спостережності (*observability platform*) нового покоління. Основна мета продукту полягає у створенні технологічних інструментів, які допомагають розробникам і технічним командам аналізувати стан своїх систем у реальному часі, швидко виявляти аномалії, зменшувати кількість інцидентів і забезпечувати стабільність програмних продуктів.

Внутрішня робоча мова команди – англійська, що зумовлено її міжнародним складом. Саме тому використовується двомовна термінологія: ключові поняття, якими послуговується команда, наводяться англійською із відповідниками українською в дужках. Такий підхід відповідає реаліям сучасних технологічних компаній, де розробка, комунікація та документація ведуться англійською незалежно від географії учасників.

Компанія має чітку місію – допомагати організаціям підтримувати стабільність і високу якість своїх цифрових продуктів через раннє виявлення технічних проблем. NOS9 прагне зробити складні технології спостережності доступними, зручними й такими, що не потребують глибоких налаштувань. Продукт дозволяє компаніям заощаджувати ресурси, попереджуючи інциденти до того, як вони вплинуть на кінцевих користувачів.

Довгострокові амбіції (Aspirations and Quests) компанії – це створення продукту, який буде провідним постачальником спостережності для середнього бізнесу та стартапів, орієнтуючись на North Star Metric – скорочення виробничих інцидентів і часу простою щонайменше на 30% для клієнтів протягом першого року користування.

Бізнес-цілі розвитку продукту NOS9 формуються відповідно до бачення, визначеного у Product Vision Board (Дошка бачення продукту) (рис. 1-1), яка використовується як стратегічний інструмент узгодження ціннісної пропозиції продукту, сегментів користувачів, проблем, що підлягають вирішенню, та

очікуваних бізнес-результатів. Product Vision Board систематизує ключові припущення щодо користувацьких потреб, визначає пріоритети продуктового розвитку та забезпечує спільне розуміння між стейкголдерами щодо того, для чого створюється продукт, яку цінність він має приносити та яким чином буде оцінюватися його успішність. У практиці створення NOS9 цей інструмент використовується для формування чіткої логіки продуктового розвитку: від виявлення проблеми пізнього виявлення деградації продуктивності до визначення очікуваних результатів у вигляді зниження витрат, підвищення стабільності та зростання користувацької бази.

Головні стратегічні напрямки:

1. Developer Experience (Покращення досвіду розробників) – забезпечення глибокої інтеграції NOS9 у професійні робочі середовища розробників. Створення плагінів для середовищ розробки (IDE), зокрема Visual Studio Code, що дозволяють отримувати аналітичні підказки безпосередньо під час роботи з кодом.

2. Модель Freemium – залучення незалежних розробників та стартапів, забезпечуючи низький поріг входу та швидкий початок роботи з платформою через безкоштовний базовий рівень доступу до з можливістю масштабування.

3. Відкриті стандарти – інтеграція з екосистемою OpenTelemetry, Prometheus та іншими відкритими технологіями спостережності, що забезпечують прозорість збору телеметрії, технологічну незалежність та сумісність NOS9 із поширеними галузевими підходами. Використання відкритих стандартів дозволяє створити уніфікований механізм отримання метрик, логів і трасувань, спрощує впровадження платформи в середовищах різного масштабу, зменшує витрати на інтеграцію та забезпечує відповідність очікуванням інженерних команд, які вже працюють з інструментами open-source-стеку.

4. Побудова спільноти (Community Building) – механізм підтримки, співтворення та вдосконалення продукту на основі зворотного зв’язку через Slack та Discord, проведення вебінарів, хакатонів і програм спільної розробки (*co-creation*).

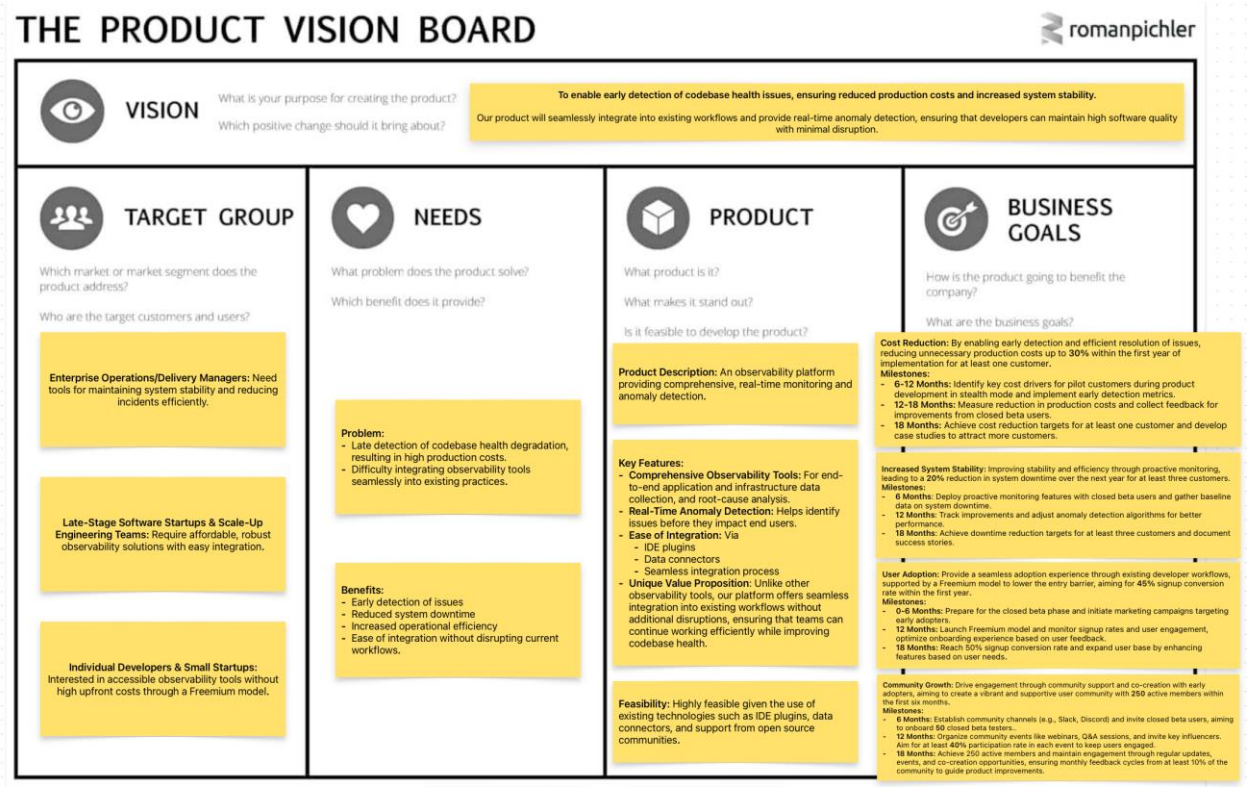

Рисунок 1.1 – The product vision board.

Джерело: розроблено автором

Сфера спостережності за програмними системами сьогодні переживає стрімкий розвиток. Сегмент observability-інструментів розвивається під впливом кількох структурних чинників:

- ускладнення архітектур програмних систем - зростання кількості мікросервісів, контейнерних платформ і CI/CD-потоків призводить до збільшення обсягу телеметричних даних і підвищення ризику аномалій;
- зростання вартості інцидентів - за результатами емпіричного дослідження Ponemon Institute «Cost of Data Center Outages» середня вартість простою дата-центру становить близько 8 851 дол. США за хвилину, причому з 2010 по 2016 рр. цей показник зріс на [1]. Огляд Atlassian з посиланням на Gartner оцінюють середню вартість простою IT-системи на рівні 5 600 дол. США за хвилину [2]. Це підтверджує, що деградація продуктивності та простої мають суттєвий фінансовий вплив, а отже раннє виявлення аномалій є критично необхідним для сучасних IT-компаній.

- навантаження на інженерні команди - команди DevOps, SRE та розробники не мають змоги витрачати значну кількість часу на інтеграцію складних observability-рішень, що відкриває можливість для простих, доступних продуктів з мінімальним порогом входу.

1.2. Обґрунтування продукту

На етапі *Discovery* здійснено аналіз бізнес-моделі NOS9 з використанням двох взаємодоповнювальних інструментів - Persona Canvases та Jobs-to-be-Done (JTBD). Обидва інструменти є базовими в сучасному продуктово-орієнтованому управлінні, оскільки забезпечують коректне формування бізнес-вимог, визначення ключових сценаріїв використання та валідацію ціннісної пропозиції продукту.

Persona Canvas (персона) – це структурований опис узагальненого представника певного сегмента користувачів, який дозволив сформувати структуроване розуміння трьох ключових клієнтських сегментів, їхніх професійних контекстів, болів та очікувань. Це забезпечило основу для продуктового аналізу, формування вимог і побудови Customer Journey.

JTBD-підхід (Jobs-to-be-Done, Робота, яку намагається виконати користувач) використовується для опису потреби не через функції продукту, а через очікуваний результат, який користувач намагається отримати. Роль JTBD у формуванні продукту полягає в обґрунтуванні ціннісної пропозиції, пріоритизації функціоналу, формування User Stories, валідації MVP у рамках HADI-гіпотез та побудові Customer Journey.

Було визначено три основні сегменти клієнтів. Першим сегментом є Enterprise Operations Managers (рис 1.2), управлінці, які відповідають за стабільність великих корпоративних систем і прагнуть зменшити кількість інцидентів без збільшення витрат. Персона: Karen. JTBD: “забезпечити стабільність системи та зменшити кількість інцидентів”.

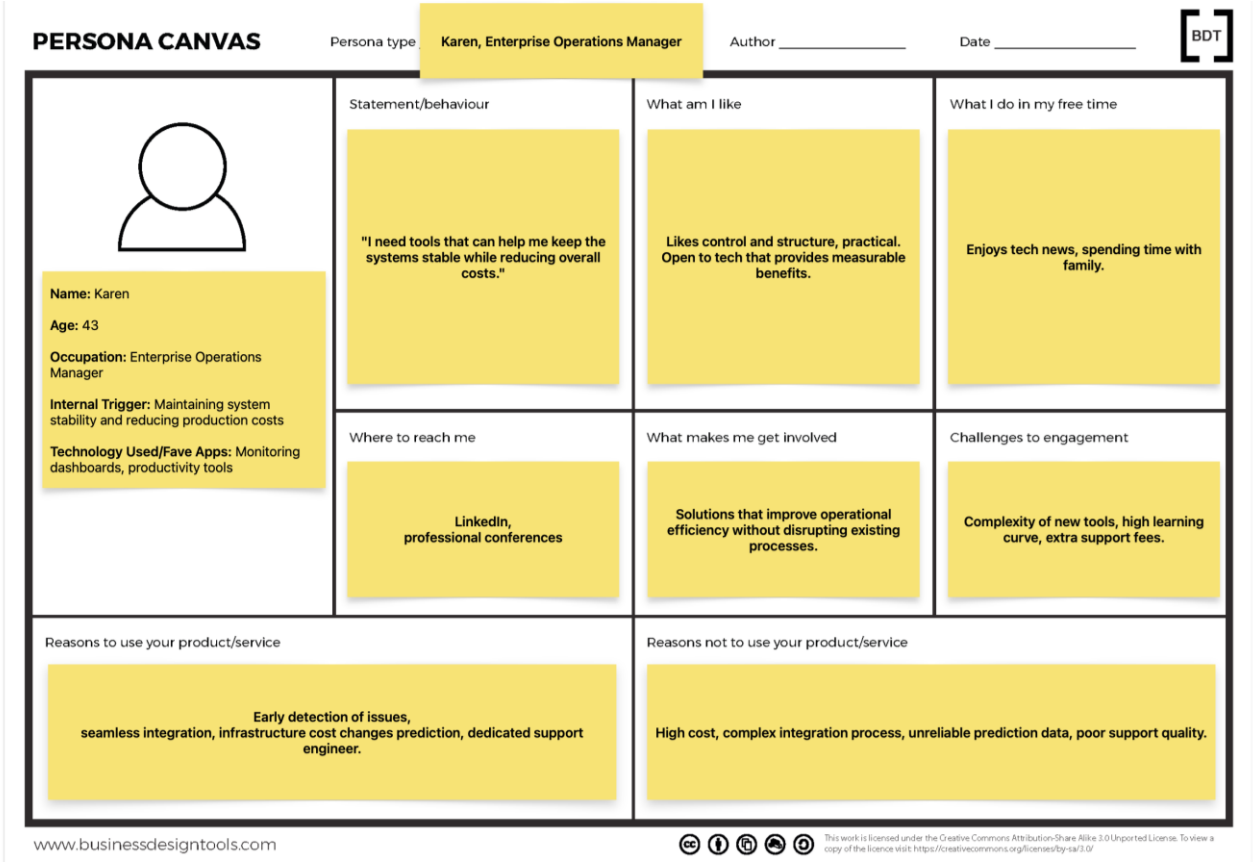

Рисунок 1.2 – Persona canvas: Karen.

Джерело: розроблено автором

Другим сегментом є технічні команди (рис 1.3) швидкозростаючих компаній (Scale-up Engineering Teams)**,** що потребують швидких інтеграцій без порушення існуючих процесів. Персона: Jamie. JTBD: “отримувати інсайти без зміни робочого процесу”.

Останнім сегментом є Individual Developers and Startups (рис 1.4)**,** невеликі команди або індивідуальні розробники, які шукають доступні рішення з низьким порогом входу. Персона: Darren. JTBD: “підтримувати якість продукту при мінімальних витратах”.

Розроблені Persona Canvas та JTBD-структура підтверджує необхідність універсального, але легкого у впровадженні рішення, та дозволяють сформулювати головну проблему наступним чином:

«Пізнє виявлення деградації здоров’я кодової бази, що викликає збільшення витрат на інциденти та ускладнює підтримання стабільності складних систем».

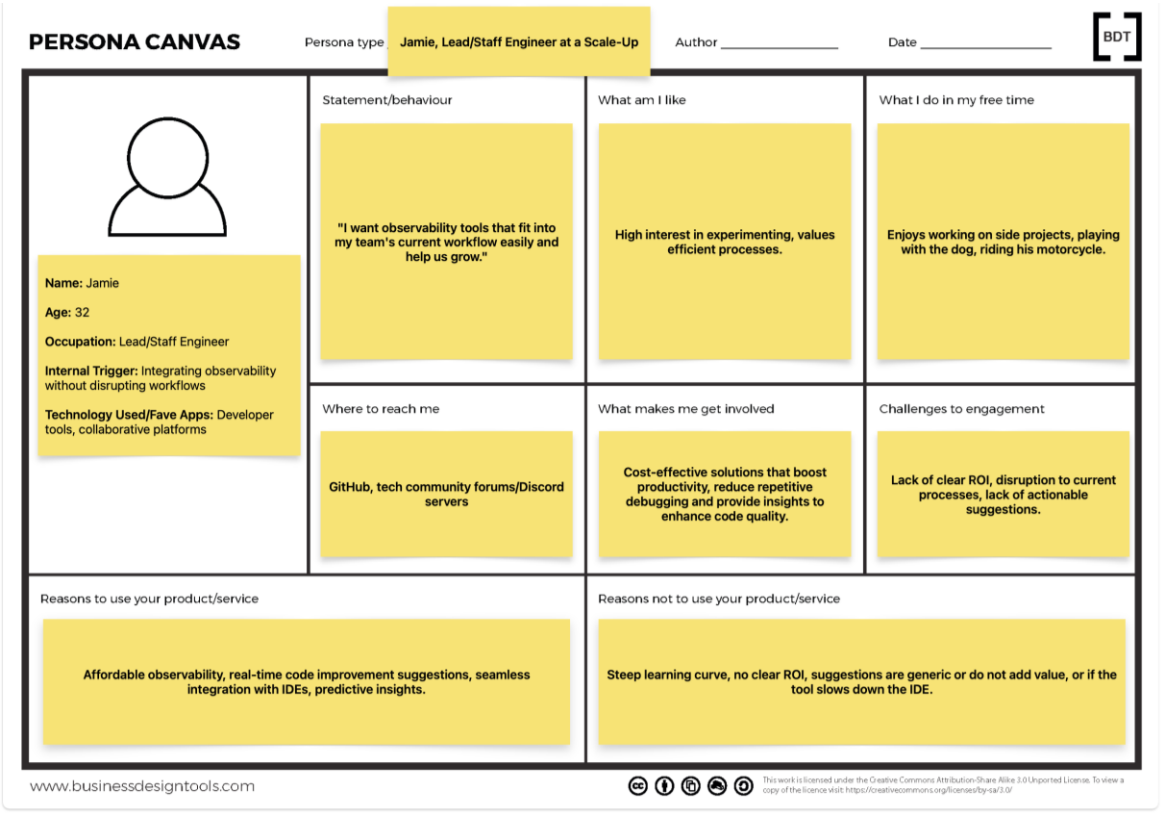

Рисунок 1.3 – Persona canvas: Jamie.

Джерело: розроблено автором

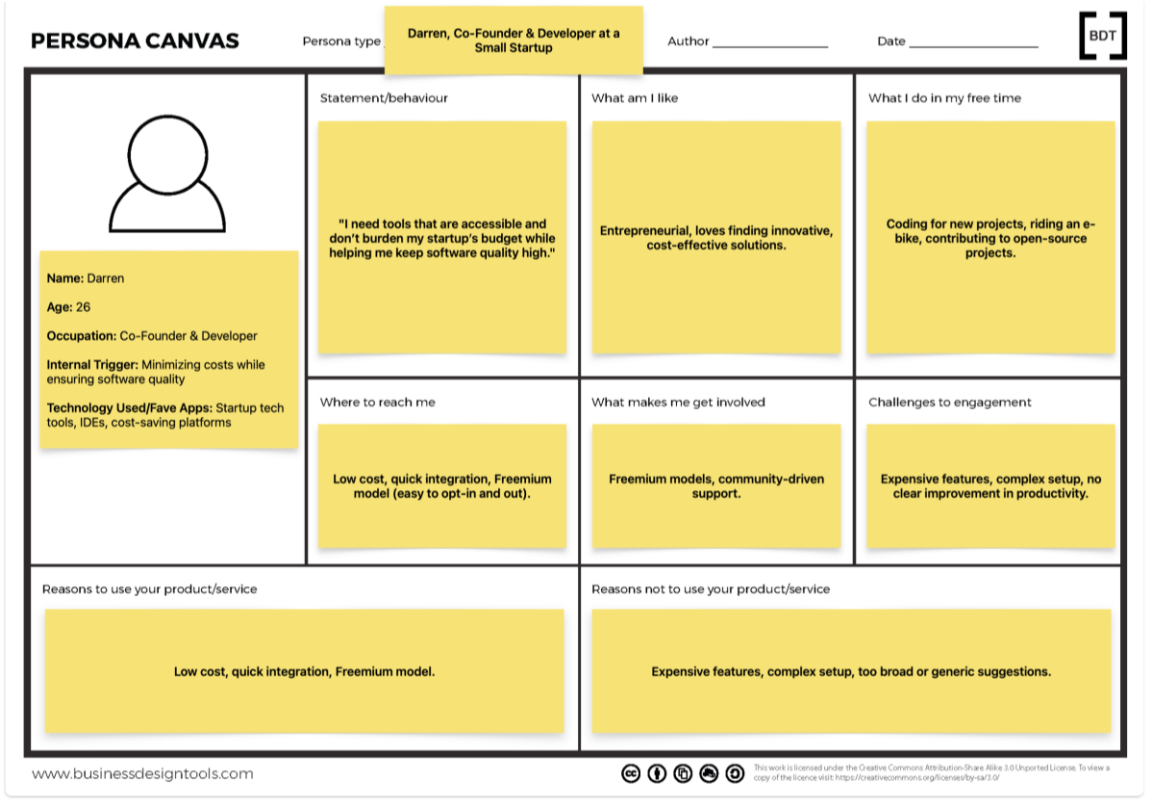

Рисунок 1.4 – Persona canvas: Darren.

Джерело: розроблено автором

Проблема ускладнюється наступними чинниками:

- розкиданістю даних між різними системами моніторингу;
- високою складністю інтеграції наявних APM-платформ;
- надмірним навантаженням на інженерні команди;
- відсутністю доступних інструментів для стартапів.

Таким чином, більшість інструментів спостережності складно впроваджувати у вже налагоджені DevOps-процеси. Це призводить до пізнього виявлення проблем у коді, підвищення витрат на підтримку і, як наслідок, втрати бізнес-ефективності.

Мета продукту NOS9 – усунути ці обмеження, забезпечивши раннє попередження про деградацію системи, що дозволяє діяти превентивно.

NOS9 пропонує інтегровану платформу для спостережності з можливістю збору, аналізу й візуалізації метрик у режимі реального часу. Її унікальність полягає у простоті впровадження: замість складних налаштувань користувач може розпочати роботу безпосередньо з IDE, використовуючи плагін, який автоматично підключається до аналітичного ядра. Таким чином, NOS9 не змінює існуючий робочий процес розробника, а лише підсилює його ефективність.

Моделювання канвасів (Business Model Canvas (рис 1.5) [3], Lean Product Canvas, Persona Canvas (рис 1.6)) дозволило чітко окреслити цільові сегменти користувачів, їхні ключові «болі» та очікувані результати та виявити, що критичною умовою розв’язання проблеми пізнього виявлення деградації систем є створення єдиної платформи, що поєднує моніторинг, аналітику та інтеграцію з робочим середовищем розробника. Така платформа має забезпечувати безперервний цикл «спостереження – інтерпретація – дія» для всіх визначених персон (Karen, Jamie, Darren), надаючи їм релевантні інсайти без порушення звичних процесів розробки, експлуатації та підтримки. У термінах продуктової стратегії це означає перехід від фрагментованого використання розрізнених інструментів спостережуваності до цілісного SaaS-рішення NOS9, вбудованого в IDE та інженерну інфраструктуру клієнта, що мінімізує бар’єри впровадження, скорочує час реакції на інциденти та створює основу для подальшого масштабування продукту.

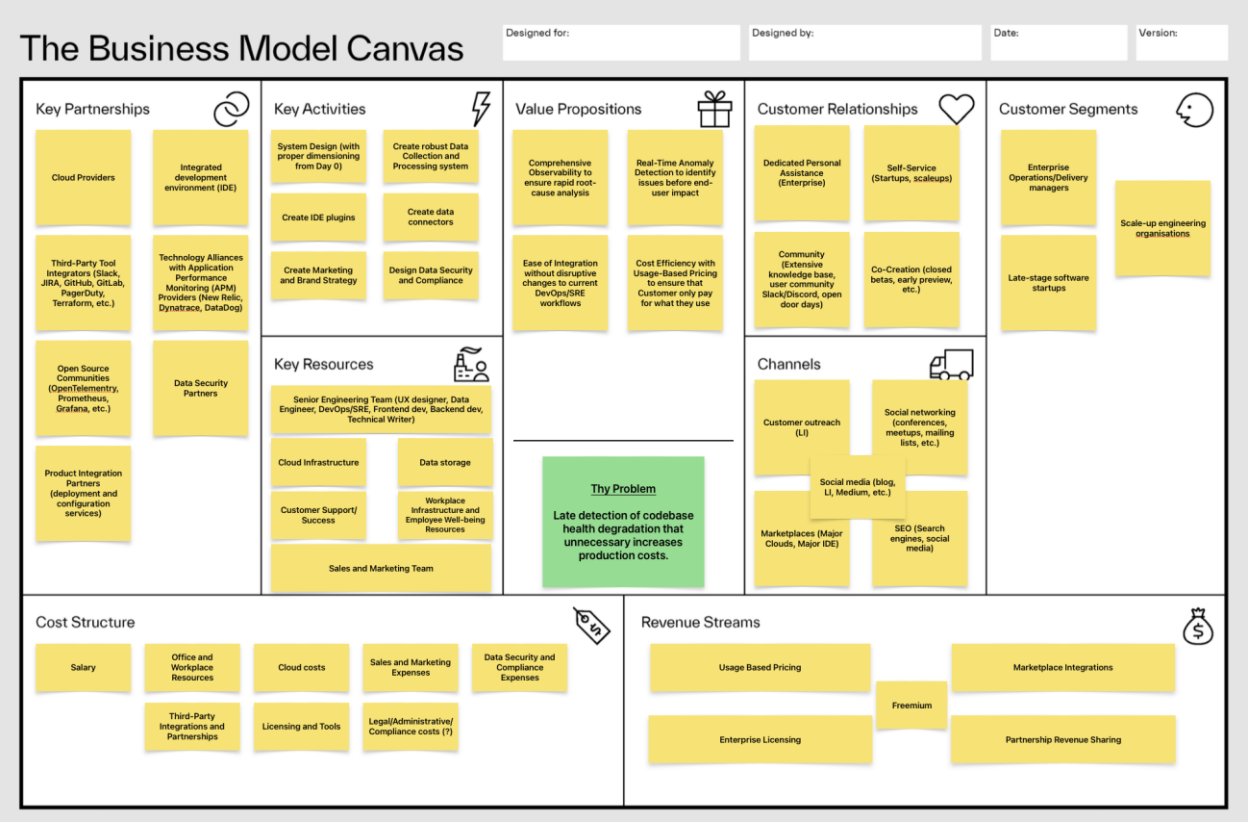

Рисунок 1.5 – The Business Model Canvas

Джерело: розроблено автором

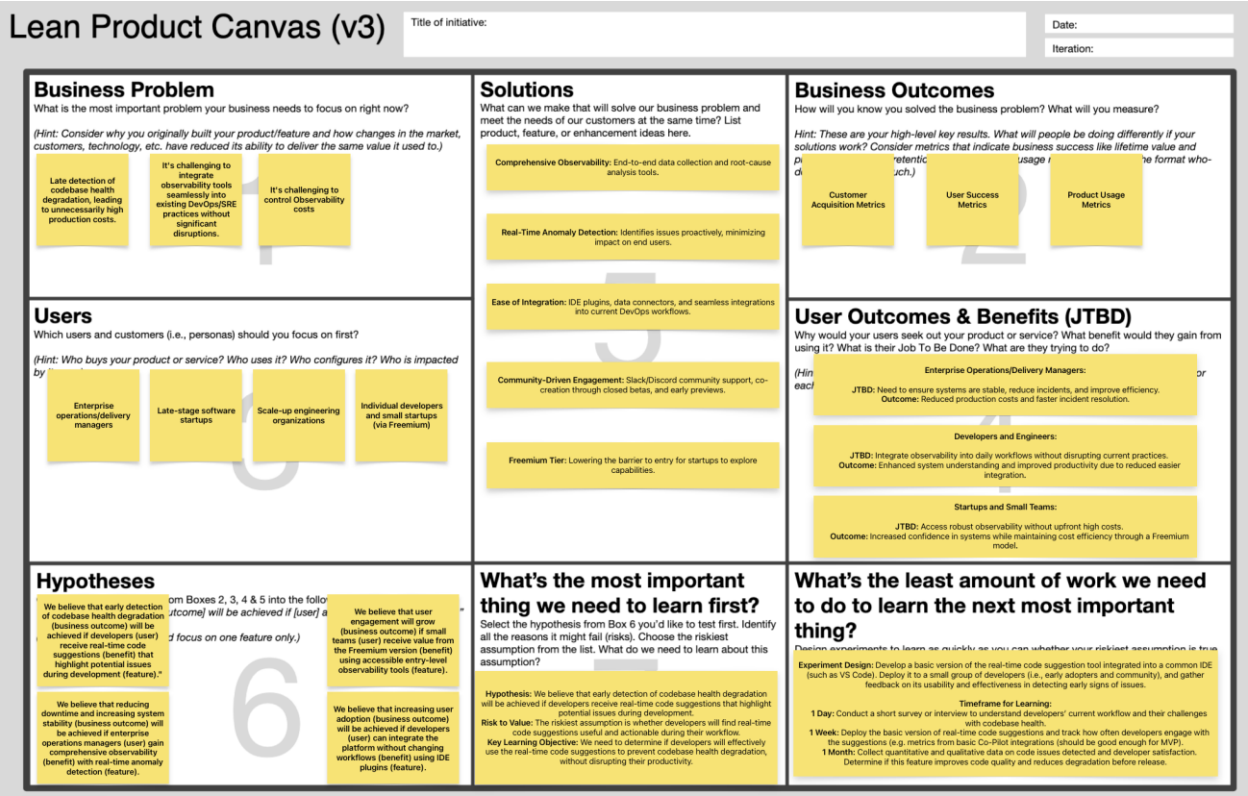

Рисунок 1.6 – Lean Product Canvas

Джерело: розроблено автором

1.3. Опис домену за моделлю Cynefin Framework

Зважаючи на аналіз продукту NOS9 наведеного вище, можна встановити що створення та розвиток такого продукту ефективніше буде відбуватися у домені Complex System, відповідно до моделі Cynefin Framework (рис 1.7), запропонованої Дейвом Сноуденом [4].

Рис. 1.7 – Візуалізація моделі Cynefin.

Джерело: [4]

Це середовище, у якому неможливо передбачити всі наслідки рішень наперед, оскільки фактори взаємодіють нелінійно. Замість жорсткого планування використовуються експерименти, аналіз і адаптація до нових умов.

Команда працює за принципом *probe–sense–respond* – тобто «дослідити – зрозуміти – відреагувати». Такий підхід відповідає природі ринку IT-продуктів, де зміни відбуваються швидко, а результат залежить від реального користувацького досвіду.

Agile-підхід у цьому контексті є найдоцільнішим, оскільки дає змогу поступово розвивати продукт, перевіряючи гіпотези, отримуючи дані, збираючи фідбек і вдосконалюючи рішення в межах коротких ітерацій. Саме цей принцип лежить в основі подальшої роботи команди NOS9 над створенням мінімально життєздатного продукту (MVP).

Висновки до розділу 1

У першому розділі було сформовано цілісний опис бізнес-контексту, сегментації клієнтів і проблемного простору, що обґрунтовує необхідність створення платформи NOS9. На основі Product Vision Board, Persona Canvases, Business Model Canvas та Lean Canvas визначено ключові проблеми – відсутність цілісної спостережуваності у більшості команд, пізнє виявлення аномалій у кодовій базі та інфраструктурі, висока інвазивність існуючих інструментів спостережності, високий ціновий бар’єр існуючих продуктів для стартапів і невеликих команд. Сформовано продуктову візію, ціннісну пропозицію, бізнес-вимоги та обсяг проєкту, що створюють методичну основу для формування продуктової стратегії, визначення HADI-гіпотез, архітектури MVP та подальшої імплементації гнучких практик управління у процесі розроблення NOS9.

РОЗДІЛ 2
ГНУЧКЕ УПРАВЛІННЯ СТВОРЕННЯМ ПРОДУКТУ NOS9

2.1. Особливості гнучкого управління розробкою продукту

Управління розробкою платформи NOS9 здійснюється відповідно до гібридного характеру проєкту, Scrum [5] використовується для розробки та підтримки інфраструктури, у поєднанні з Lean Startup (методикою ощадливого запуску продукту) для експериментальної перевірки гіпотез щодо поведінки користувачів.

Обґрунтування використання Scrum у NOS9 спирається на наступні чинники:

- нерівномірність вимог користувачів Enterprise Managers (рис. 1.2.), Scale-up Engineers (рис. 1.3.), Startup Developers (рис. 1.4.);
- високу технологічну невизначеність, що віднесено до квадранту Complex моделі Cynefin;
- необхідність коротких циклів інкрементального постачання згідно Product Vision Board (Дошки бачення продукту) (рис. 1-1.).

Управління проєктом за фреймом Scrum передбачає наявність трьох ключових ролей: Scrum Master, Product Owner та Development Team. Scrum Master забезпечує коректне застосування процесів Scrum, сприяє командній ефективності та усуває перешкоди, що можуть впливати на виконання завдань. Product Owner відповідає за формування бачення продукту, визначення та пріоритизацію вимог, а також за планування інкрементів і релізів. Development Team – це самоорганізована кросфункціональна команда фахівців, яка безпосередньо виконує роботи зі створення програмного продукту.

Однією з основних характеристик Scrum є система регулярних командних подій, що забезпечують прозорість, адаптивність і безперервну координацію роботи протягом спринту.

Таким чином Scrum використовується для організації процесу розроблення продукту, забезпечуючи ітеративне постачання інкрементів та прозорість командної роботи. Водночас методологія Lean Startup застосовується як підхід

до експериментальної перевірки гіпотез щодо поведінки користувачів та цінності функціональності продукту. Lean Startup передбачає побудову циклу «побудувати – виміряти – навчитися», у межах якого команда створює мінімальний функціонал, збирає дані про його реальне використання та формує інсайти для подальших рішень. Такий підхід дає змогу виявляти цінність функцій на ранніх етапах, зменшуючи ризики помилкових припущень. Поєднання Scrum та Lean Startup дає змогу інтегрувати інкрементальну розробку з науково обґрунтованою перевіркою продуктового бачення та забезпечує умови для реалізації HADI-гіпотез (Hypothesis – Action – Data – Insights) з JTBD-підходами, визначеними під час Discovery-фази продукту, що є критичним для створення інноваційної SaaS-платформи у домені спостережності.

2.2. Планування змісту, тривалості та вартості проєкту

2.2.1. Обов'язкові роботи

Обов'язковими роботами визначені такі, які мають бути реалізовані у будь-якому випадку, що формують технологічну базу продукту та не залежать від результатів експериментів.

Для обов'язкових робіт сформульовано Use Cases (сценарії використання) – опис поведінки системи, як вона відповідає на зовнішні запити. Іншими словами, сценарії використання описує, «хто» і «що» може зробити з розглянутою системою. Методика різновидів використання застосовується для виявлення вимог до поведінки системи, відомих також як функціональні вимоги.

Таким чином, до обов'язкових робіт належать:

1. Створення системи зберігання та обробки даних у реальному часі**.** Реалізується окремим етапом, оскільки є центральним компонентом продукту. Вирішення проблеми пізнього виявлення деградації можливе лише за умови наявності високопродуктивного ядра обробки. Не має окремого різновиду використання – системна функціональність без прямого користувацького сценарію.

2. Побудова інфраструктури збору даних (Data Collection Pipelines) (табл. 2.1)**,** яка включає механізми збору метрик, логів та трасувань, інтеграційні точки для Kubernetes, Docker або хмарних провайдерів. Основний набір метрик і базові алгоритми аномалій мають бути доступні всім Persona-групам.

Таблиця 2.1 – Різновид використання 1. Налаштування інфраструктури збору даних.

Різновид використання	Налаштування інфраструктури збору даних
Актори	DevOps, SRE, Data Engineer
Попередні умови	
- створено організаційний обліковий запис NOS9; - цільова інфраструктура (кластер Kubernetes, хмарне середовище або Docker) доступна для інтеграції; - налаштовані облікові дані доступу до інфраструктури.	
Основний сценарій	
- DevOps Engineer авторизується в NOS9 та переходить до модуля налаштування data collection pipelines; - обирає тип інтеграції (наприклад, Kubernetes або хмарний провайдер) та додає необхідні параметри підключення; - вмикає збір базових метрик (CPU, пам’ять, час відповіді, помилки) згідно зі скоупом MVP; - підтверджує конфігурацію та запускає тестовий збір даних; - перевіряє, що зібрані метрики коректно надходять до сховища та відображаються в системі.	
Результат	
- Інфраструктура збору метрик і логів успішно підключена; - платформа NOS9 отримує потік даних у реальному часі для подальшої обробки.	

3. Розроблення базової панелі моніторингу (Core Dashboard) (табл. 2.2)**,** яка має відображати ключові метрики, логування змін інфраструктури та базову аналітику.

Таблиця 2.2 – Різновид використання 2. Перегляд операційним менеджером стану системи на дашборді.

Різновид використання	Перегляд операційним менеджером стану системи на дашборді
Актор	Карен (Enterprise Operations Manager)
Попередні умови	
- інтеграцію з інфраструктурою налаштовано (таб. 2.1);	

- зібрані метрики зберігаються та обробляються продуктом; - користувач має роль із доступом до операційного дашборду.
Основний сценарій
- Operations Manager входить до NOS9 та відкриває панель моніторингу; - Обирає потрібний сервіс, кластер або User Journey для аналізу; - Переглядає ключові показники (CPU, пам’ять, час відповіді, помилки) до та після останніх змін інфраструктури; - За потреби змінює часовий інтервал або деталізацію даних; - Приймає управлінське рішення (масштабування, оптимізація ресурсів тощо) на основі отриманої інформації.
Результат
- Operations Manager отримує цілісне уявлення про стан системи в реальному часі; - Ухвалює обґрунтовані рішення щодо стабільності та продуктивності.

4. Реалізація RBAC. RBAC (табл 2.3) необхідний для розмежування рівнів доступу, що дозволить легко будувати різні рівні доступа для користувачів продукту, базуючись на ролі, або комерційній складовій.

Таблиця 2.3 – Різновид використання 3. Керування доступом та тарифами.

Різновид використання	Керування доступом та тарифами
Актори	Account Owner (Адміністратор організації), Карен / Джеймі / Даррен як кінцеві користувачі.
Попередні умови	
- Організаційний акаунт створено; - Freemium-модель і ролі доступу реалізовані в системі.	
Основний сценарій	
- Адміністратор входить до NOS9 та відкриває модуль керування користувачами й тарифами; - Створює ролі доступу (наприклад, «Operations Manager», «Lead Engineer», «Developer»), пов’язуючи їх із наборами прав (перегляд дашбордів, керування інтеграціями, адміністративні дії); - Додає нових користувачів (Karen, Jamie, Darren) та призначає їм відповідні ролі; - За потреби оновлює тарифний план з Freemium до платного рівня, без втрати історичних даних, забезпечуючи безперервний перехід між планами; - Перевіряє, що користувачі мають коректний доступ і бачать релевантний функціонал.	
Результат	
- Ролі та права доступу налаштовано; - Користувачі мають відповідний до їхньої функції рівень доступу; - Перехід між тарифами відбувається без втрати даних.	

5. Створення базової системи автоматизованого тестування (QA Pipelines) (табл. 2.4). Система має пройти тестування на продуктивність, стабільність і коректність збору метрик.

Таблиця 2.4 – Різновид використання 4. Запуск і аналіз автоматизованих тестів якості платформи.

Різновид використання	Запуск і аналіз автоматизованих тестів якості платформи
Актори	NOS9 QA Engineer.
Попередні умови	
- Налаштовані CI/CD-пайплайни; - Створено набір автоматизованих тестів для основних сценаріїв (збір даних, робота дашборду, продуктивність плагіну, базова аномалія).	
Основний сценарій	
- QA Engineer ініціює прогін автоматизованих тестів (ручним запуском або через push/merge у репозиторії або за розкладом); - Система виконує тести для ключових сценаріїв: ○ коректність збору метрик; ○ стабільність real-time engine під навантаженням; ○ швидкодія дашборду; ○ робота VS Code-плагіну на тестовому репозиторії. - Після завершення тестів формується звіт із результатами, включно з помилками та показниками продуктивності. - QA Engineer аналізує результати і, за потреби, створює задачі у Product Backlog для усунення дефектів.	
Результат	
- Підтверджено базову технічну готовність платформи; - Дефекти зафіксовано й передано в розробку; - Знижено ризик випуску нестабільного інкременту MVP.	

6. Організація спільноти користувачів (Slack/Discord) – формування активної спільноти як механізму зворотного зв’язку і driver adoption.

Таблиця 2.5 – Різновид використання 6. Взаємодія користувача зі спільнотою підтримки.

Різновид використання	Взаємодія користувача зі спільнотою підтримки
Актор	Будь який користувач
Попередні умови	
- Створено спільноту (Slack/Discord); - У продукті доступний канал переходу до спільноти та механізм подання зворотного зв’язку (feedback mechanisms) .	

Основний сценарій
- Користувач, зіткнувшись із питанням або проблемою інтеграції, відкриває розділ «Допомога/Спільнота» в інтерфейсі NOS9; - Переходить до Slack/Discord-каналу спільноти або створює запит через вбудовану форму зворотного зв’язку; - Формулює запит (опис проблеми, контекст, середовище) та надсилає його; - Support Engineer отримує запит, аналізує його та надає рекомендації або рішення; - За потреби Support Engineer створює новий елемент у Product Backlog (наприклад, покращення документації або інтерфейсу), якщо проблема носить системний характер.
Результат
- Користувач отримує відповідь або рішення щодо свого запиту; - Накопичується база знань і зворотного зв’язку для подальшого розвитку продукту.

2.2.2. Роботи, що виконуються за HADI-циклом

Роботи, не визначені як фундаментальні або архітектурно неминучі, виконуються через експерименти відповідно до HADI (Hypothesis – Action – Data – Insight) – ітераційної моделі, що базується на принципі експериментального підходу до розробки (Hypothesis-Driven Development).

Кожна гіпотеза проходить оцінку за параметрами Impact (вплив), Confidence (впевненість) і Ease (легкість реалізації), що дозволяє правильно визначити пріоритети в беклозі.

Для формулювання гіпотез було застосовано шаблон представлений в роботі [6].

Було сформовано набір гіпотез, що охоплюють три основні напрями розвитку продукту:

- технічна якість (Гіпотеза 1, Гіпотеза 4) – зниження кількості помилок і підвищення стабільності через автоматизацію (табл 2.6, 2.9);
- бізнес-ефективність (Гіпотеза 2) – зменшення MTTR і втрат від простоїв табл (2.7);
- ринкова цінність (Гіпотеза 3) – залучення нових користувачів через Freemium-модель. (табл 2.8)

Таблиця 2.6 – Гіпотеза 1. IDE-підказки знижують кількість помилок у коді

Елемент	*Опис*
Hypothesis	Якщо розробники отримуватимуть рекомендації щодо якості коду безпосередньо у VS Code, кількість технічних інцидентів у кодовій базі зменшиться на 15%
Action	Розробити MVP-плагін для VS Code, інтегрувати його з бекендом NOS9, провести внутрішнє тестування серед розробників
Data (Метрики)	- Usage Rate (UR) ≥ 70% активних користувачів; - Adoption Latency (AL) ≤ 10 хв після інсталяції; - Error Reduction (ER) ≥ 15% зниження кількості помилок у комітах.
Insights	Збір аналітики IDE покаже, наскільки часто користувачі приймають підказки, і дозволить визначити найкорисніші типи рекомендацій
Impact	Високий – безпосередньо підвищує якість коду та зменшує регресії
Confidence	4/5
Ease	3/5 – середня складність інтеграції з IDE та аналітикою

Таблиця 2.7 – Гіпотеза 2. Раннє виявлення аномалій скорочує час реакції на інциденти

Елемент	Опис
Hypothesis	Якщо операційні менеджери отримуватимуть сповіщення про аномалії в реальному часі, середній час реакції (MTTR) скоротиться на 20%
Action	Реалізувати модуль збору базових метрик (CPU, Memory, Latency), побудувати дашборд, виміряти MTTD та MTTR
Data (Метрики)	- Mean Time to Detect (MTTD) ≤ 30 секунд; - Mean Time to Recover (MTTR) ≤ 15 хв; - Downtime Reduction (DR) ≥ 20%.
Insights	Порівняння MTTR до й після впровадження покаже реальний вплив NOS9 на бізнес-процеси клієнтів
Impact	Високий – напряму впливає на стабільність систем
Confidence	3/5
Ease	4/5 – використовується наявна інфраструктура моніторингу

Таблиця 2.8 – Гіпотеза 3. Freemium-модель підвищить залучення користувачів

Елемент	Опис
Hypothesis	Якщо впровадити Freemium-рівень доступу, кількість активних користувачів зросте на 45%, а Retention Rate перевищить 60%
Action	Створити базовий тариф із обмеженим функціоналом, провести тестову кампанію серед невеликих команд і стартапів
Data (Метрики)	- Conversion Rate (CR) ≥ 10%; - Retention Rate (RR) ≥ 60%; - Net Promoter Score (NPS) ≥ +30.
Insights	Аналіз показників CR і RR допоможе оптимізувати тарифну політику перед запуском Beta-релізу
Impact	Середній – впливає на розширення користувацької бази
Confidence	3/5
Ease	4/5 – невисока складність, мінімальні технічні ризики

Таблиця 2.9 – Гіпотеза 4. Автоматизація CI/CD підвищує ефективність команди

Елемент	Опис
Hypothesis	Якщо процеси CI/CD будуть автоматизовані, кількість регресій після релізів скоротиться щонайменше на 15%, а стабільність релізів зросте.
Action	Розробити CI/CD-пайплайни на GitHub Actions, додати автоматичне тестування й деплой у staging.
Data (Метрики)	- Deployment Frequency (DF) ≥ 3 релізи/тиждень; - Change Failure Rate (CFR) ≤ 5%; - Lead Time for Changes (LTC) ≤ 1 година.
Insights	Порівняння DF і CFR до/після впровадження покаже ефект DevOps-автоматизації.
Impact	Високий – прискорює релізний цикл і зменшує ризики помилок.
Confidence	5/5
Ease	2/5 – реалізація потребує високої технічної експертизи.

2.3. Product Roadmap (дорожня карта продукту) та Product Backlog (продуктовий беклог)

Для координації дій команди та поступового переходу від ідеї до готового рішення було створено Product Roadmap (дорожню карту продукту**,** табл. 2.10**).** Вона відображає стратегічні фази розвитку NOS9, та пов'язує їх із обов'язковими роботами та результатами HADI-експериментів.

Дорожня карта побудована за принципом *progressive delivery* – поступового розширення функціональності з одночасним тестуванням кожного нового модуля.

Таблиця 2.10 – Product Roadmap board

Етап	Тривалість	Основна мета	Ключові метрики
Discovery	0–1 міс.	Визначення користувацьких сегментів, формування Canvas, постановка гіпотез	≥ 3 підтверджені гіпотези
Early MVP	1–2 міс.	Розробка базового функціоналу, розробка плагіна для VS Code, інтеграція з бекендом	UR ≥ 70%, AL ≤ 10 хв, ER ≥ 15%
Alpha	3–6 міс.	Тестування anomaly detection, розширення дашборду	MTTR ≤ 15 хв, DR ≥ 20%, CFR ≤ 5%
Beta / Release	6–12 міс.	Впровадження Freemium, зростання користувацької бази	RR ≥ 60%, CR ≥ 10%, NPS ≥ +30
Scaling / Stability	12-18 міс.	Масштабування та стабілізація	Reliability (SLO) ≥ 99,5 %.

Після формування первинної дорожньої карти продукту (Product Roadmap), що базувалася на обов'язкових роботах, гіпотезах і метриках, команда розробила Product Backlog (продуктовий беклог) – структурований список функціональних вимог, технічних завдань і покращень, необхідних для створення мінімально життєздатного продукту (MVP).

Продуктовий беклог (рис. 2.1) виступає центральним артефактом управління вимогами в Agile, забезпечуючи прозорість процесу розробки, узгодження пріоритетів і поетапне наближення команди до досягнення

стратегічних OKR. На його основі здійснюється планування спринтів, розподіл ресурсів і визначення ключових deliverables кожного етапу.

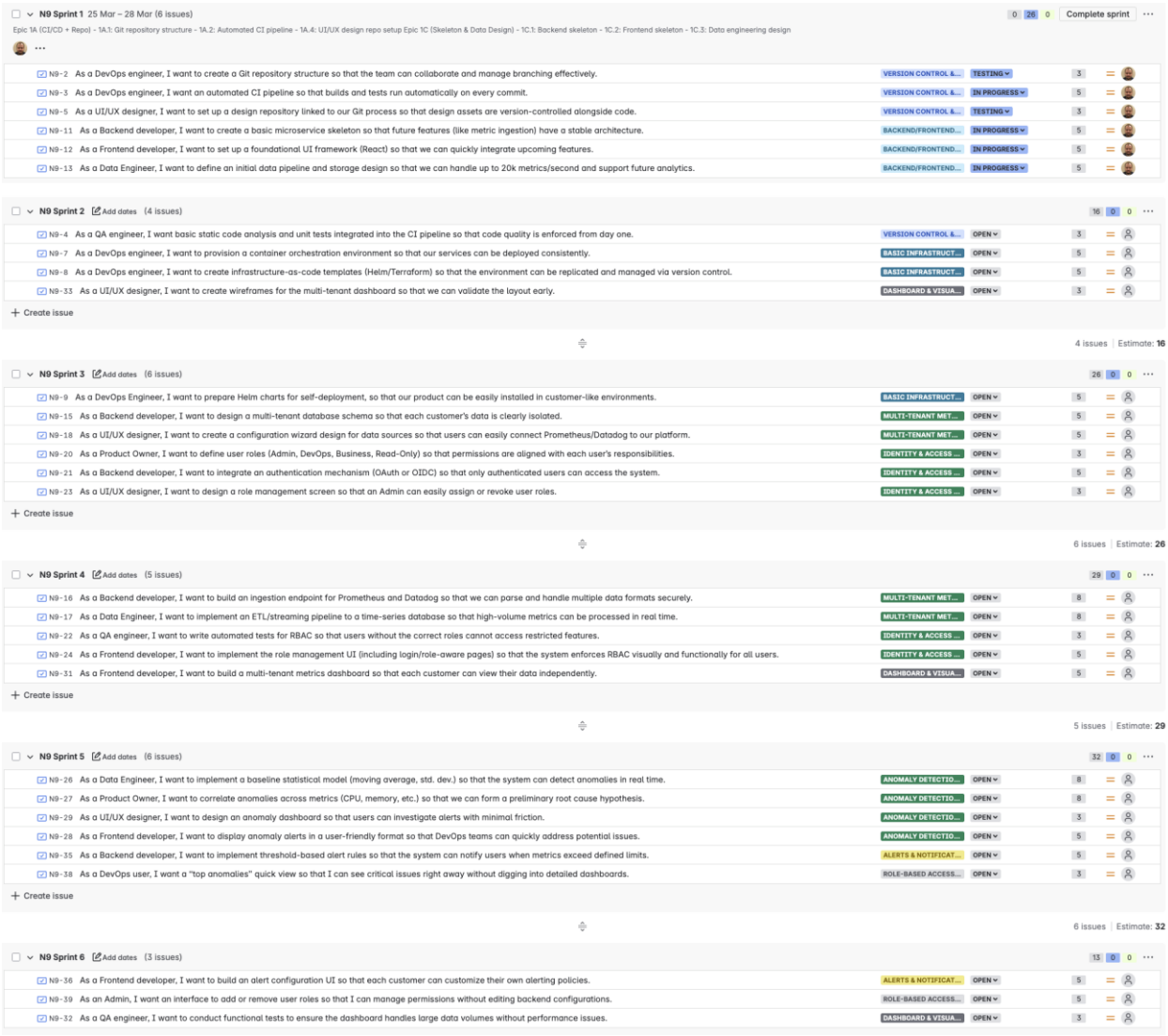

Рисунок 2.1. – Фрагмент продуктового беклогу

Джерело: розроблено автором

Усі вимоги були описані у форматі User Stories, що відображають потреби користувачів у формі *«роль – дія – очікуваний результат»* (*As a [role], I want [action], so that [benefit]*). Такий підхід дозволяє представити функціональність продукту з погляду кінцевої цінності для користувача, а не лише технічної реалізації.

Для кожної User Story визначено метрику успіху (success metric), яка дає змогу оцінити рівень досягнення очікуваного результату та вплив реалізованої функціональності на користувацький досвід або бізнес-ціль. Використання таких

метрик створює основу для data-driven product management, коли рішення про пріоритети ухвалюються на основі вимірюваних показників, отриманих під час тестування MVP та аналізу зворотного зв’язку користувачів.

Також окремі User Stories були сформовані для кожної з персон:

- Карен (рис. 1.2) – менеджер з операційної діяльності підприємства, зацікавлена у моніторингу аномалій та стабільності системи (табл. 2.11);
- Даррен (рис. 1.3) – співзасновник і розробник, відповідальний за технічну реалізацію продукту та інтеграцію сервісів (табл. 2.12);
- Джеймі (рис. 1.4) – провідний інженер, який забезпечує експлуатаційну надійність і ефективність інфраструктури (табл. 2.13).

Таблиця 2.11 – User Story для Карен

№	User Story	Мета	Критерії прийнятності
1	Як менеджер з операційної діяльності, я хочу отримувати повідомлення про виявлення аномалій, щоб уникнути простоїв і зменшити витрати.	Отримання автоматичних повідомлень про аномалії для запобігання збоям у роботі системи	• Повідомлення про аномалії спрацьовують, коли виявляються аномалії в роботі системи (зразки даних та точки виявлення будуть надані). • Повідомлення про аномалії не спрацьовують, коли аномалії в роботі системи не виявляються (зразки даних та точки виявлення будуть надані).
2	Як менеджер з операцій, я хочу відстежувати ключові показники ефективності в режимі реального часу, щоб мати змогу спостерігати за стабільністю системи та приймати обґрунтовані рішення	Моніторинг KPI у реальному часі для прийняття оперативних рішень	• Система відображає всі зміни інфраструктури (наприклад, масштабування, оновлення, розгортання) з часовими мітками. • Система відображає ключові показники продуктивності (наприклад, CPU, використання, час відгуку) до і після змін. • Оновлення даних відбувається кожні 2 секунди або менше в тестовому середовищі.
3	Як менеджер з експлуатації, я хочу інтегрувати інструмент моніторингу з існуючими поточними робочими процесами, щоб уникнути додаткового навантаження на мою команду.	Спрощення інтеграції нового інструменту у наявні процеси	• Інтеграція завершується протягом 4 годин. • Зміни конфігурації системи під час налаштування мінімальні. • Під час процесу інтеграції доступна технічна підтримка.

Таблиця 2.12 – User Story для Джеймі

№	User Story	Мета	Критерії прийнятності
1	Як провідний інженер, я хочу інструмент, вплив якого на продуктивність не перевищує 5% швидкості IDE, щоб я міг зменшити кількість повторюваних завдань і зосередитися на розробці функцій.	Оптимізувати інструмент таким чином, щоб його використання не знижувало продуктивність розробницького середовища	• Показники впливу на продуктивність є контекстно-залежними та відображають реальні сценарії використання • Вплив інструменту на швидкодію IDE не перевищує 5%.
2	Як провідний інженер, я хочу інтегрувати систему в Visual Studio Code, щоб отримувати аналітичні дані та результати безпосередньо у моєму середовищі розробки.	Отримувати аналітичну інформацію та результати аналізу безпосередньо в IDE без додаткових кроків інтеграції	• Інструмент доступний як розширення VS Code на Marketplace; • Аналітичні дані та результати коректно відображаються в останній стабільній версії VS Code
3	Як провідний інженер, я хочу отримувати точну аналітику в режимі реального часу про потенційні проблеми з кодом, щоб мати змогу проактивно вирішувати проблеми та підтримувати високу якість коду.	Забезпечити своєчасне виявлення проблем з кодом і можливість їх оперативного усунення	• Показники оновлюються кожні 2 секунди або менше у тестовому середовищі з набором даних у 200 000 рядків; • Аналітика охоплює час виконання функцій, частоту помилок і час завантаження залежностей; • Точність аналізу перевищує 95% при виявленні витоків пам’яті, функцій із надмірним використанням CPU та проблем із залежностями

Таблиця 2.13 – User Story для Даррена

№	User Story	Мета	Критерії прийнятності
1	Як співзасновник стартапу, я хочу оновити безкоштовну версію продукту без втрати даних, щоб мати змогу продовжувати користуватися інструментом без впливу на клієнтів.	Забезпечити безперервність користування продуктом під час оновлення, уникаючи втрати даних і порушень у роботі клієнтів	• Користувачі можуть оновити пробну версію до платного плану без втрати даних; • Після успішного оновлення система відображає підтверджувальне повідомлення; • Безкоштовна версія недоступна протягом перехідного періоду.
2	Як розробник, я хочу, щоб інструмент плавно інтегрувався в мій робочий процес, щоб я міг	Оптимізувати процес налаштування та інтеграції	• Інтеграція завершується не довше ніж за 15 хвилин; • Наявні чіткі інструкції з налаштування та технічна

	мінімізувати час налаштування та негайно розпочати роботу.	інструменту в середовище розробки	документація; • Після інтеграції продукт готовий до повноцінного використання без додаткової конфігурації.
3	Як розробник, я хочу, щоб система автоматично надавала пропозиції щодо якості коду, щоб я міг вдосконалити своє програмне забезпечення з обмеженими ресурсами.	Підвищити якість програмного коду шляхом автоматизованого аналізу та рекомендацій щодо покращення	• Пропозиції з'являються після кожного виконання коду; • Для кожної пропозиції надаються покрокові рекомендації з реалізації; • Рекомендації відповідають найкращим галузевим практикам (best practices).

Для підвищення ефективності планування та прийняття рішень щодо пріоритетів завдань було використано матрицю Ейзенхауера (Eisenhower Matrix) – інструмент стратегічного тайм-менеджменту, що базується на розподілі активностей за двома критеріями: важливість (вплив на досягнення цілей) та терміновість (часова критичність виконання).

Застосування цього методу дозволило команді чітко класифікувати елементи беклогу відповідно до їхньої цінності для продукту та строків реалізації, розділивши їх на чотири категорії:

1. Термінові та важливі – завдання, що мають безпосередній вплив на стабільність MVP і потребують негайного виконання (наприклад, усунення критичних дефектів, інтеграція бекенду з плагіном).
2. Важливі, але не термінові – стратегічні задачі, які визначають довгостроковий розвиток продукту (наприклад, оптимізація CI/CD-процесів або покращення юзабіліті інтерфейсу).
3. Термінові, але не важливі – операційні завдання, що можуть бути делеговані чи автоматизовані (наприклад, налаштування тестового середовища).
4. Не термінові й не важливі – активності, які мають низьку пріоритетність і можуть бути відкладені або виключені з поточного спринту.

Таким чином, використання матриці Ейзенхауера дало змогу оптимізувати планування спринтів, мінімізувати ризик перевантаження команди та забезпечити фокус на завданнях, що створюють найбільшу цінність для користувачів і бізнесу.

Таблиця 2.14 – Приклад використання матриці Ейзенхауера для пріоритизації розробки MVP-плагіна

Категорія	Приклади завдань	Основні метрики	Статус
Важливе і термінове	Розробка MVP-плагіна, налаштування збору базових метрик	UR ≥ 70%, ER ≥ 15%	Виконується
Важливе, але не термінове	Створення дашборду, документації користувача	MTTR ≤ 15 хв, RR ≥ 60%	Планується
Неважливе, але термінове	Автотестування, базове CI/CD	DF ≥ 3, CFR ≤ 5%	Планується
Неважливе і не термінове	Інтеграція з іншими IDE, маркетинг	NPS ≥ +30	Відкладено

2.4 Постановка цілей та організація команди

Для вимірювання прогресу та узгодження діяльності учасників команда застосовує методологію OKR (Objectives and Key Results). Це сучасна система управління цілями, що забезпечує стратегічну узгодженість, прозорість і вимірюваність результатів. Її використання сприяє концентрації зусиль команди на створенні конкретної цінності для користувача та дозволяє оцінювати успіх не лише за виконаними завданнями, а й за досягнутими результатами. Відповідно до дослідження Ю. Семененка, впровадження OKR сприяє підвищенню ефективності управління за рахунок поєднання гнучкості стратегічного підходу та вимірюваності результатів, а також створює умови для синергії з іншими інструментами управління результативністю [7]. У межах проєкту NOS9 основним інструментом цілепокладання виступає саме система OKR.

Перша ціль (Objective 1) полягає у створенні технічно стабільного MVP продукту NOS9, готового до внутрішнього тестування та збору первинного користувацького зворотного зв’язку. Ця ціль визначає стратегічний фокус команди – забезпечення працездатного мінімально життєздатного продукту, який може бути перевірений у реальному середовищі.

Ключові результати (Key Results):

- KR1.1. Забезпечити повну роботу інфраструктури збору даних (метрики, логи, трасування) з обробкою не менше 20 000 метрик/секунду у режимі реального часу.;
- KR1.2. Забезпечити інтеграцію бекенду з плагіном із затримкою обміну даними не більше 2 с;
- KR1.3. Досягти стабільності роботи MVP із впливом на швидкодію IDE (VS Code) не більше 5 %;
- KR1.4. Реалізувати автоматизоване тестування не менш ніж 80 % основних сценаріїв використання;
- KR1.5. Отримати не менше 10 якісних відгуків користувачів за результатами внутрішнього тестування.

Друга ціль (Objective 2) полягає в забезпеченні базової користувацької цінності для трьох ключових сегментів (Карен, Джеймі та Даррена)

Ключові результати (Key Results):

- KR2.1. Забезпечити можливість завершення інтеграції NOS9 з інфраструктурою клієнта за менше ніж 4 години (вимога Карен);
- KR2.2. Забезпечити можливість інтеграції VS Code-плагіну з NOS9 за менше ніж 15 хвилин (вимога Даррена);
- KR2.3. Реалізувати базову панель моніторингу з трьома ключовими групами метрик (CPU, Memory, Response Time) та логуванням інфраструктурних змін (Обов'язковий компонент Core Dashboard MVP);
- KR2.4. Забезпечити роботу плагіна VS Code з відображенням базових аналітичних підказок і підтримкою проєктів до 200 000 рядків коду (вимога Джеймі);
- KR2.5. Реалізувати рольову модель (RBAC) з мінімум трьома ролями користувачів: Manager, Engineer, Developer (обов'язкова частина архітектури MVP);

Застосування OKR у межах розробки продукту дозволяє команді перевести стратегічні наміри в конкретні вимірювані результати, підтримувати спільне бачення мети, а також своєчасно ідентифікувати відхилення від запланованого курсу. Таким чином, система OKR виступає інструментом не лише планування,

а й безперервного вдосконалення, що забезпечує прозорість і підзвітність у процесі розробки продукту.

Отже, визначені OKR формують стратегічний орієнтир діяльності команди та створюють основу для ефективного управління розробкою продукту. Реалізація цих цілей потребує злагодженої взаємодії між учасниками, гнучкої структури відповідальностей і високого рівня комунікаційної узгодженості. Саме тому наступним кроком стає формування кросфункціональної команди, здатної забезпечити повний цикл створення й виведення продукту на ринок.

Розробка продукту NOS9 здійснюється кросфункціональною командою, яка об’єднує інженерну, аналітичну та управлінську експертизу. Такий підхід відповідає принципам Agile Software Development [8], що передбачає автономність членів команди, колективну відповідальність за результат і здатність до швидкої адаптації в умовах турбулентного середовища.

Команда складається із семи висококваліфікованих фахівців, кожен з яких виконує чітко визначену роль у створенні мінімально життєздатного продукту (MVP). Така структура забезпечує комплексне покриття всіх етапів розробки SaaS-рішення, від формування вимог і дизайну до автоматизації, тестування та релізу.

До складу команди входять:

- Product Owner (PO) / Scrum Master (SM) – одна людина, що поєднує дві ролі в Scrum-команді:
  - Як PO, відповідає за формування бачення продукту, пріоритизацію Product Backlog та затвердження результатів кожної ітерації. Виконує роль посередника між бізнес-цілями та технічною реалізацією, забезпечуючи орієнтацію команди на створення користувацької цінності;
  - Як SM, фасилітує процеси Scrum, підтримує продуктивність команди, координує планування, стендапи та ретроспективи. Його роль полягає у створенні сприятливого середовища для командної ефективності та усуненні перешкод у роботі;

З огляду на ранній етап розвитку продукту NOS9 та обмежений розмір команди, зазначена роль PO/SM фактично також виконує функції первинного

технічного керівництва (прото-Engineering Manager). Це зумовлено необхідністю забезпечення наскрізної координації між продуктовою логікою, технічними рішеннями та операційними процесами. У цьому контексті PO/SM бере участь у визначенні архітектурних пріоритетів, контролі технічного боргу, узгодженні DevOps-процесів та організації якісної взаємодії між підкомандами. Хоча роль Engineering Manager формально не виділена, виконання її ключових функцій на ранній фазі розвитку продукту забезпечує цілісність технічного циклу й підтримує стабільність розробки MVP;

- Frontend Developers (2) – відповідають за клієнтську частину системи, зокрема реалізацію плагіна для Visual Studio Code і користувацького інтерфейсу дашборду. Вони забезпечують інтуїтивну взаємодію користувача з продуктом і тісно співпрацюють із бекенд-командою для інтеграції даних;
- UX Designer (0.5) – забезпечує перетворення технічних можливостей продукту на зрозумілі, доступні та ефективні сценарії взаємодії користувачів із системою;
- Backend Developers (2) – розробляють серверну частину NOS9, створюють API для обміну даними, базу даних для зберігання метрик і логів, а також механізми аналітики в реальному часі;
- Automation QA Engineer – розробляє автоматизовані тести, які перевіряють стабільність роботи продукту, інтеграційні сценарії та продуктивність системи, забезпечуючи контроль якості на всіх етапах розробки;
- DevOps Engineer – налаштовує процеси CI/CD, контейнеризацію через Docker і Kubernetes, а також впроваджує інструменти моніторингу (Prometheus, Grafana) для підтримки стабільної інфраструктури розгортання.

Команда функціонує в гібридному форматі, поєднуючи очну та віддалену співпрацю спеціалістів із різних країн. Така модель роботи підвищує гнучкість, але водночас вимагає ефективної координації, прозорих комунікацій та використання спільного цифрового середовища. Робочою мовою команди є англійська, що зумовлено міжнародним складом учасників і глобальною орієнтацією продукту.

Основні інструменти, що застосовуються у щоденній діяльності:

- *Slack* – головний канал оперативної комунікації;
- *Jira* – система планування завдань і ведення беклогу;
- *GitHub* – платформа контролю версій і проведення code review;
- *Figma* – засіб для прототипування інтерфейсів;
- *Miro* – середовище для візуалізації roadmap і фасилітаційних сесій.

Робочий процес побудований навколо коротких двотижневих спринтів, кожен із яких завершується подіями Sprint Review та Retrospective. Під час Review команда демонструє досягнуті результати, а в межах Retrospective аналізує процеси, визначаючи можливості для подальшого вдосконалення. Такий підхід забезпечує стабільний темп розвитку продукту, сприяє підвищенню командної зрілості та реалізує принцип безперервного покращення (*continuous improvement*) – ключовий для Agile-команд.

Комунікація в команді побудована на прозорості та регулярності. Щоденні stand-up meetings проводяться для синхронізації прогресу та виявлення блокерів. Щотижневі planning sessions визначають завдання на наступний спринт, а sprint demos дозволяють представити результати розробки Product Owner'у та всій команді.

Retrospective-сесії допомагають команді аналізувати процеси, виявляти успішні практики та визначати напрями для вдосконалення.

Додатково використовується коротке опитування (*Team Sentiment Survey*), яке фіксує емоційний стан учасників і дозволяє підтримувати здорову командну динаміку.

Команда приділяє значну увагу професійному розвитку учасників, розглядаючи навчання не як окремий процес, а як інтегровану складову командного менеджменту. В основу підходу покладено модель 70:20:10 [9], що передбачає збалансоване поєднання практичного досвіду, соціального навчання та формальної освіти:

- 70% навчання відбувається через практичну діяльність – безпосередню участь у спринтах, релізах та вирішенні реальних завдань продукту;

- 20% – через взаємодію в команді, обмін досвідом, менторинг і парне програмування;
- 10% – через формальне навчання, зокрема онлайн-курси, вебінари, внутрішні тренінги та участь у професійних спільнотах.

Такий підхід відповідає сучасним тенденціям розвитку талантів у гібридному робочому середовищі, де головним ресурсом стають навички та компетенції, а процес навчання розглядається як форма управління змінами, спрямована на адаптивність і безперервне оновлення знань [10].

Після кожного спринту команда проводить ретроспективу, під час якої оцінює власну ефективність за допомогою інструментів Team Health Check та Team Canvas. На основі результатів визначаються індивідуальні цілі розвитку – наприклад, покращення навичок автоматизації тестування, оптимізація CI/CD-процесів або підвищення якості комунікацій між підрозділами. Цей підхід забезпечує цикл безперервного навчання, у якому розвиток компетенцій поєднується з підвищенням продуктивності, що узгоджується з концепцією формування «skill based» у сучасних організаціях.

2.5 Бюджет проєкту

Використаємо приблизну середню ринкову вартість роботи відповідних спеціалістів, що базується на середніх значеннях для ринку ЄС/віддаленої роботи (remote) згідно наступних ресурсів:

- Levels.fyi – спеціалізується на аналітиці зарплат у технологічних компаніях, особливо для DevOps, Backend, Frontend, QA, PM/PO, UX;
- Glassdoor – одне з найпопулярніших джерел, для усереднених даних;
- Payscale – велика база зарплат за професіями та регіонами. Доречно використовувати для DevOps, QA, SM, PO, UX;
- Indeed Salaries – дані на основі сотень тисяч вакансій. Особливо релевантне джерело по Scrum Master, Product Owner та QA Automation;
- RemoteOK – дані по зарплатах для remote-спеціалістів, включаючи Frontend/Backend, DevOps, QA Automation, Product roles, UX Designer.

Таблиця 2.15 – Середні діапазони зарплат

Позиція	**Місячна ставка, EUR**
Product Owner / Scrum Master	6 000 – 9 000
Frontend Developer (Senior, x2)	5 000 – 7 000 кожен
UX Designer (0.5 ставки)	3 000 – 5 000
Backend Developer (Senior, x2)	8 000 – 11 000 кожен
Automation QA Engineer	4 000 – 6 000
DevOps	8 000 – 11 000

Таким чином, місячний бюджет команди складає ~ 56 000 EUR.

Для повного бюджетування MVP також врахуємо:

- додаткові операційні витрати на хмарну інфраструктуру, сторонні сервіси (GitHub, Atlassian, etc.) та ліцензії - ~ 6 000 EUR;
- Найм персоналу (Talent Acquisition) - ~ 2 000 EUR;
- Юридичні витрати (Legal & Compliance) - ~ 3 000 EUR;
- Витрати на управління та нагляд (Governance / Management overhead) - ~ 3 000 EUR;
- Неявні операційні витрати - ~ 2 000 EUR.

Таким чином, орієнтовний бюджет NOS9 на місяць складає 72 000 EUR.

Орієнтовний бюджет на розробку MVP (6 місяців) становить 432 000 EUR.

2.6 Виконання робіт

З огляду на те, що повний обсяг робіт не може бути визначений наперед через наявність гіпотез та необхідність їх експериментальної перевірки, планування виконується у коротких ітераціях, де результати кожного інкременту впливають на подальшу структуру Product Backlog. Таким чином, розвиток продукту NOS9 відбувається не як реалізація фіксованого плану, а як послідовність перевірених кроків, у межах яких команда адаптує пріоритети, уточнює вимоги та формує наступні елементи змісту робіт. Для забезпечення керованості цього процесу використовуються артефакти Scrum – насамперед

спринтове планування, контроль виконання завдань та прозорість ходу розроблення.

Налаштування Jira-workflow та дошки використані командою, було змінено для забезпечення прозорості, повторюваності та відповідності ролям.

Workflow (рис. 2.2) включає такі стани:

- Open – створено, чекає призначення;
- In Progress – активна робота розробника;
- Testing – тестування розробником або QA Engineer;
- Resolved – перевірено технічно, очікує закриття;
- Closed – завершена робота;
- Reopened – повернення задачі у роботу після виявлення дефектів.

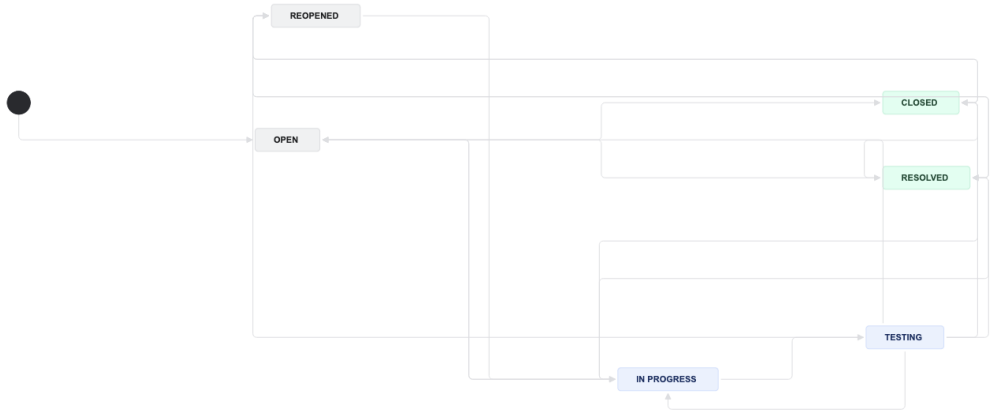

Рисунок 2.2. – Налаштування Jira workflow

Джерело: розроблено автором

Налаштування workflow (рис. 2.3) прив’язане до складу команди NOS9:

- Frontend / Backend Engineers працюють здебільшого у статусах In Progress;
- QA Engineer виконує переходи Testing → Resolved;
- DevOps Engineer відповідає за перевірку інфраструктурних задач та їхній перехід до завершення;
- Product Owner закриває задачі після приймального огляду;

- Scrum Master контролює відповідність руху задач правилам workflow.

Рисунок 2.3. – Кінцевий автомат станів Jira workflow

Джерело: розроблено автором

Таким чином, Jira workflow відображає реальний розподіл відповідальностей і забезпечує структурованість виконання у кросфункціональній команді.

Створення продукту почалося 25 березня 2025р.

Перші два спринти були підготовчими фактичного до процесу розробки та мали коротшу довжину, ніж зазвичай.

Sprint 1 Goal (Ціллю Спринту 1) було створення базової технічної інфраструктури продукту NOS9, необхідної для подальшої розробки MVP. У межах спринту команда зосереджується на формуванні фундаментальних артефактів – репозиторіїв, CI/CD-процесів, початкової архітектури фронтенду та бекенду, а також на проєктуванні первинного конвеєра обробки даних. Результатом спринту має стати повністю готове до роботи технічне середовище, яке забезпечує можливість швидкої інтеграції, масштабування та подальшого розвитку функціональності платформи.

Стан дошки (рис. 2.4) ілюструє синхронну працю команди на початку спринту та рух задач відповідно до чітко визначеного workflow.

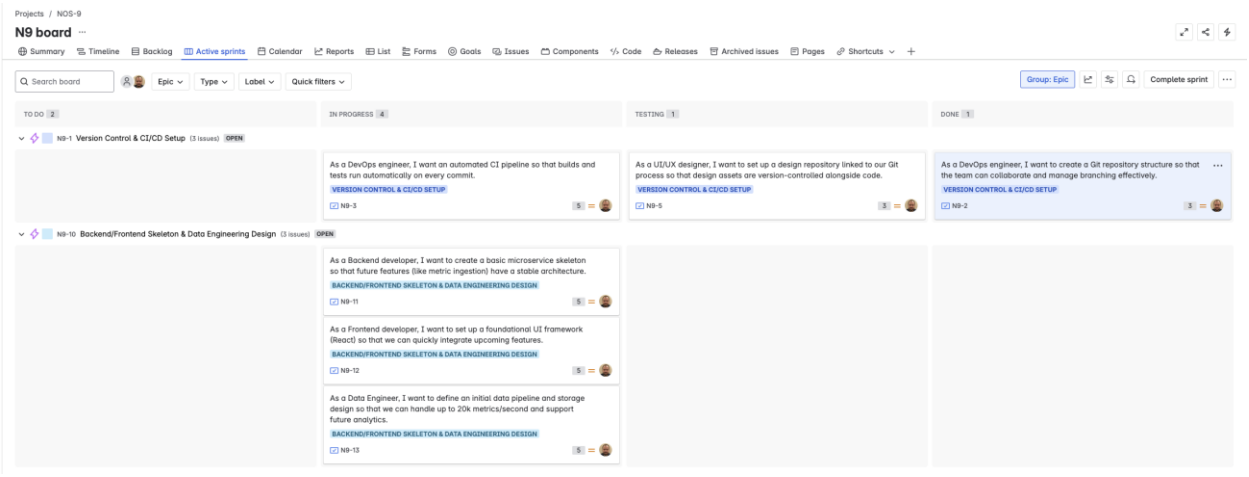

Рисунок 2.4. – Фактичний стан задач у середині спринту 1

Джерело: розроблено автором

Sprint 2 Goal (Ціллю Спринту 2) (рис 2.5) є розширення та зміцнення технічної інфраструктури продукту NOS9 шляхом впровадження механізмів забезпечення якості коду, створення відтворюваного середовища для розгортання сервісів, а також побудови базових компонентів візуальної частини продукту. Результатом спринту має стати стабільне середовище розробки та тестування, у якому сервіси можуть розгортатися послідовно, а якість коду

контролюється автоматизовано. Додатково має бути сформовано початкові wireframes багатотенантного дашборду для подальшої валідації користувацького інтерфейсу.

Стан дошки (рис. 2.3) ілюструє стан розробки ближче до закінчення спринту.

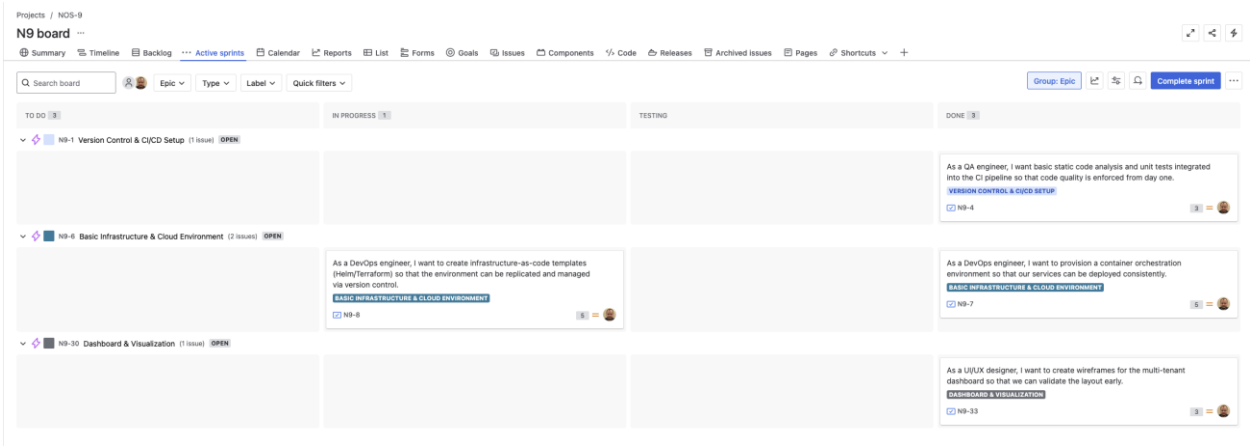

Рисунок 2.5. – Фактичний стан задач ближче до завершення спринту 2. Джерело: розроблено автором

2.7. Оцінка ефективності Scrum-команди

Оцінювання ефективності роботи Scrum-команди здійснювалося на основі ключових метрик Jira, отриманих у межах спринтів розроблення MVP NOS9. Систематичний аналіз цих метрик дозволяє визначити рівень зрілості команди, якість організації процесів і ступінь відповідності фактичної роботи встановленим практикам Scrum. Для інтерпретації результатів використано дані Velocity Chart, Burndown Chart, Sprint Reports та Cumulative Flow Diagram, що відображають структуру виконання задач, стабільність потоків робіт і збалансованість навантаження відповідно до прийнятого workflow.

Розглянемо базові критерії оцінювання на прикладі перших двох спринтів.

Аналіз Velocity Chart (діаграми швидкості) показує, що команда завершила весь обсяг задач у межах обох спринтів (рис. 2.6). У першому спринті продуктивність була максимальною завдяки зосередженню на стабільних

foundational-задачах (створення CI/CD, архітектурних скелетів, первинного data pipeline).

У другому спринті швидкість знизилася, що пояснюється збільшенням частки інфраструктурних задач та робіт, пов'язаних із автоматизацією тестування та IaC. Така динаміка відповідає типовому циклу адаптації команди на ранніх етапах роботи.

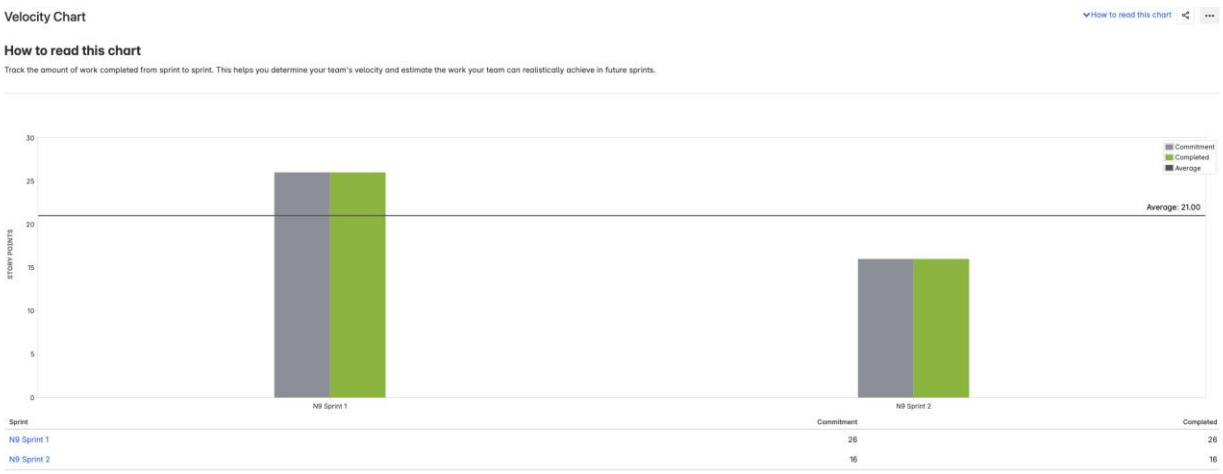

Рисунок 2.6. – Velocity Chart за перші два спринти

Джерело: розроблено автором

Burndown Charts (діаграми вигорання задач) демонструють рівномірний рух задач протягом спринтів.

У першому спринті діаграма показує (рис. 2.7), що більша частина задач виконувалася рівномірно протягом спринту. Були відсутні значні «стрибки» у завершенні задач – це свідчить про стабільний потік робіт. Також не було додано жодних додаткових задач у Sprint Backlog, а всі заплановані елементи були завершені. Таким чином, перший спринт продемонстрував чітку відповідність між планом та результатом.

Burndown Chart (діаграма вигорання задач) для другого спринту відображає іншу динаміку (рис. 2.8) – на початку спринту виконання йшло повільніше, ніж у першому спринті, що характерно для інфраструктурних задач (кластери, IaC, автоматизація тестування). Проте ближче до завершення спринту команда встигла виконати повний обсяг робіт, що відображено у різкому падінні

лінії burndown. Також гарним показником є відсутні задачі, що було перенесено до наступного спринту.

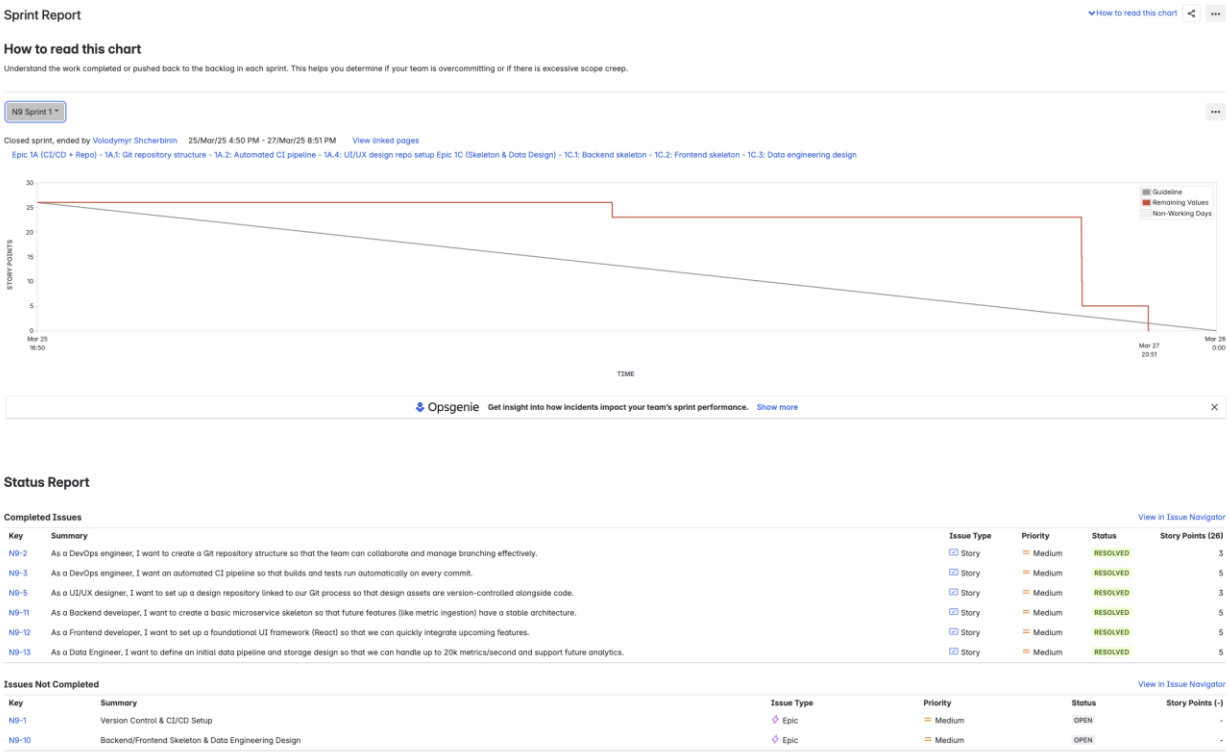

Рисунок 2.7. – Sprint 1 Report

Джерело: розроблено автором

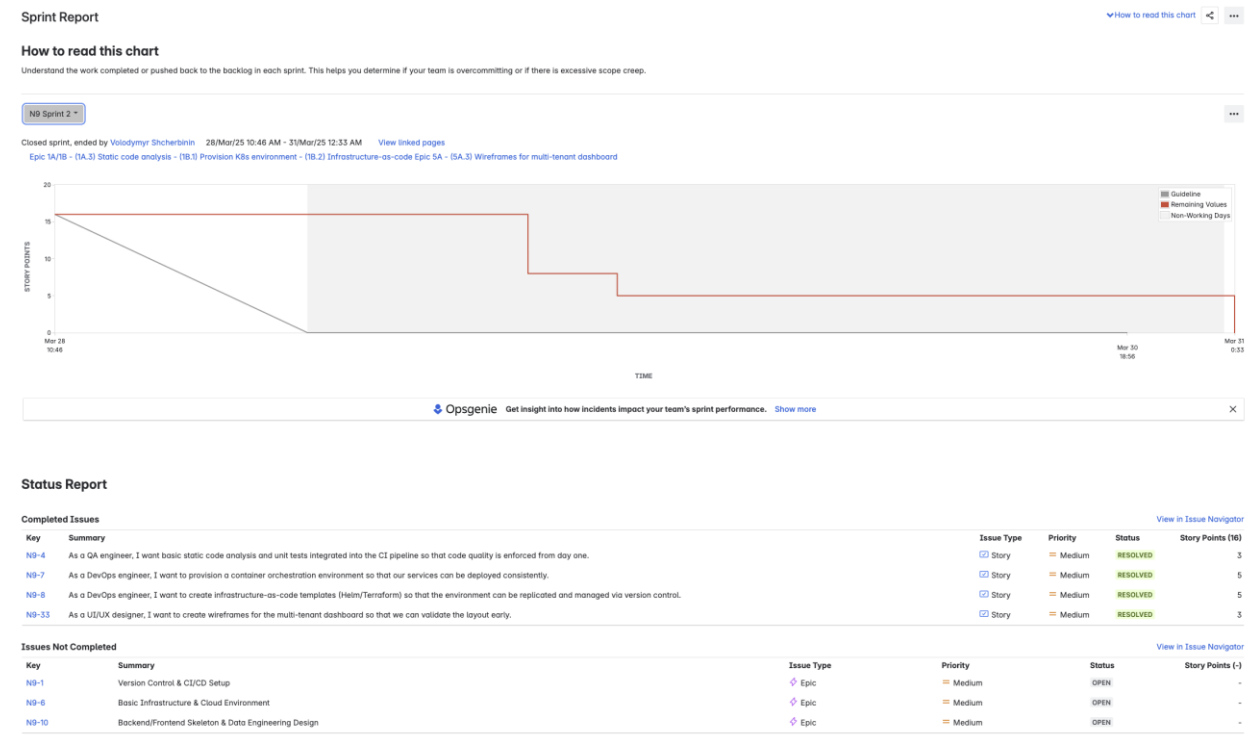

Рисунок 2.8. – Sprint 2 Report.

Джерело: розроблено автором

Таким чином, другий спринт досяг мети, а різниця між фактичним і ідеальним burndown є характерною для спринтів з високою технічною складністю.

Інша діаграма – Cumulative Flow Diagram (CFD, Кумулятивна діаграма потоку) дає змогу оцінити стабільність потоку робіт та наявність потенційних вузьких місць.

Аналіз діаграми (рис 2.9) показує:

- широка зона «To Do» на початку відповідає природному формуванню обсягу спринтів на інфраструктурному етапі;
- зона «In Progress» зростає плавно, без різких стрибків, що свідчить про відсутність значного перевантаження команди;
- зона «Testing» лишається відносно тонкою, що демонструє ефективну роботу QA Engineer та загальний швидкий цикл тестування, разом з правильним застосуванням Definition of Done;
- розширення зони «Done» стабільне, що підтверджує відсутність вузьких місць на стадії завершення задач.

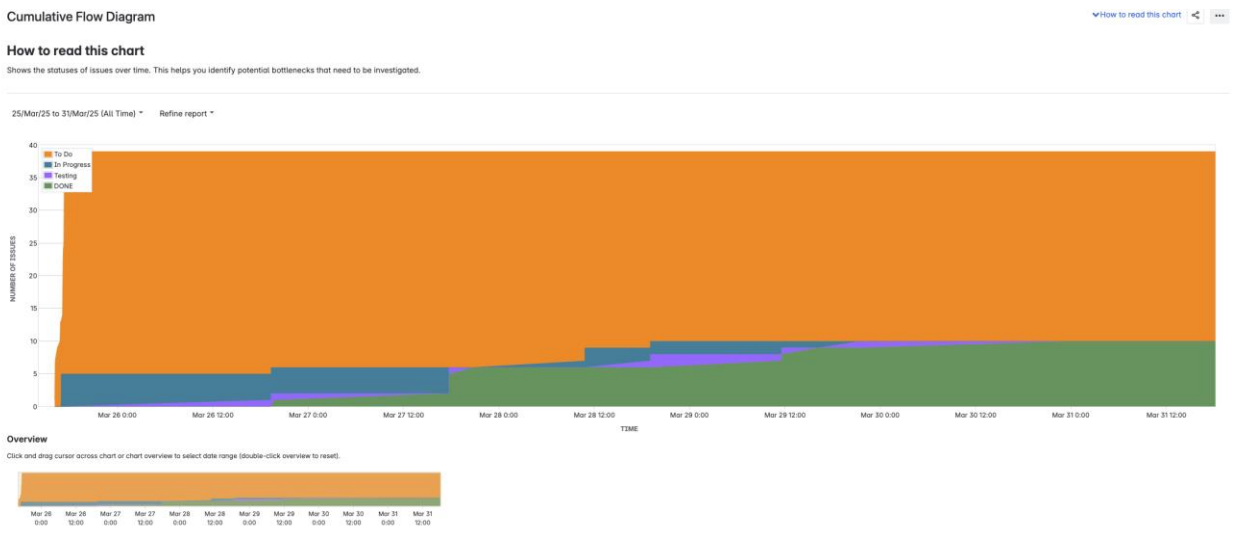

Рисунок 2.9. – Cumulative Flow Diagram (Кумулятивна діаграма потоку)

Джерело: розроблено автором

Таким чином, CFD демонструє збалансоване завантаження команди, відсутність критичних вузьких місць та правильну організацію потоків роботи відповідно до Jira-workflow.

Зведений аналіз наведених метрик дозволяє зробити такі узагальнені висновки:

- команда NOS9 продемонструвала високу стабільність у виконанні інкрементів MVP, завершивши всі задачі двох початкових спринтів без перенесення;
- velocity команди стабілізується, що свідчить про налагодженість процесів та коректну оцінку задач;
- діаграми Burndown показують відсутність значних відхилень, що характерно для зрілих Scrum-процесів навіть на ранньому етапі;
- cumulative flow diagram підтверджує відсутність перевантаження команди та збалансований рух задач між статусами;
- команда працює як повноцінна кросфункціональна одиниця, що дозволяє ефективно реалізувати Hypothesis-Driven Development в майбутніх ітераціях та розвивати MVP у складному технічному домені.

Висновки до розділу 2

У другому розділі було обґрунтовано підхід до гнучкого управління створенням платформи NOS9 та продемонстровано, як методологічні принципи Agile інтегруються з реальною практикою роботи продуктово-інженерної команди.

Обґрунтовано доцільність використання гібридного підходу, що поєднує Scrum та Lean Startup. Це дозволило адаптувати розвиток NOS9 до умов високої технологічної невизначеності, притаманної домену спостережності, та знижувати ризики помилкових продуктових рішень.

Здійснено декомпозицію змісту робіт на обов’язкові (архітектурно неминучі) та такі, що реалізуються через HADI-цикли. Було сформовано сценарії використання (Use Cases), які відображають ключові сценарії взаємодії персон NOS9 з системою.

На основі Product Vision Board та Lean Canvas побудовано Product Roadmap, яка відображає етапи розвитку NOS9 від Discovery до масштабування і стабілізації, а також пов’язує ці етапи з ключовими метриками. Далі дорожню карту деталізовано через Product Backlog сформований у форматі користувацьких історій із чітко визначеними критеріями прийнятності та метриками успіху.

Показано, як стратегічні цілі розвитку продукту трансформуються в операційно вимірювану систему за допомогою методології OKR. Сформовані OKR для MVP NOS9 охоплюють як технічну стабільність платформи, так і створення базової користувацької цінності для ключових сегментів.

Описано структуру кросфункціональної команди NOS9, до складу якої входять Product Owner / Scrum Master, Frontend та Backend розробники, UX Designer, Automation QA Engineer та DevOps Engineer. Показано, як ця структура підтримує повний цикл створення SaaS-рішення – від формування вимог і UX-дизайну до автоматизації, тестування та релізу. Окрема увага приділена інструментам комунікації й співпраці (Slack, Jira, GitHub, Figma, Miro), а також практикам безперервного навчання та розвитку компетенцій команди.

Здійснено орієнтовну оцінку бюджету проєкту на основі середніх ринкових ставок фахівців високого рівня та додаткових операційних витрат (інфраструктура, сервіси, рекрутинг, юридичний супровід, управлінські витрати). Отриманий прогнозований бюджет MVP NOS9 за шестимісячний період демонструє економічну масштабність завдання та дозволяє оцінити фінансові передумови реалізації продукту.

На прикладі перших двох спринтів проаналізовано організацію виконання робіт та налаштування Jira-workflow. Показано, що workflow виступає кінцевим автоматом станів, який відображає реальний розподіл відповідальностей між членами команди. Sprint Goals для перших двох спринтів фокусувалися на побудові базової технічної інфраструктури, автоматизації процесів, створенні відтворюваного середовища розгортання та первинних UX-артефактів. Аналіз Velocity Chart, Burndown Charts та Cumulative Flow Diagram засвідчив стабільну

продуктивність команди, відсутність перенесених задач, збалансоване навантаження та відсутність критичних вузьких місць у процесі.

Узагальнюючи, можна зробити висновок, що у розділі 2 продемонстровано цілісну модель гнучкого управління створенням продукту NOS9, яка поєднує методологічні засади Agile та Lean Startup із реальними артефактами продуктового й інженерного циклу. Така модель забезпечує прозору трансформацію стратегічного бачення у конкретні інкременти, дозволяє керувати невизначеністю через HADI-експерименти та підтримує послідовний рух до створення життєздатного MVP платформи спостережності NOS9.

РОЗДІЛ 3
ЛІДЕРСТВО, УПРАВЛІННЯ ВЗАЄМОДІЄЮ ТА КОМУНІКАЦІЯМИ В AGILE-СЕРЕДОВИЩІ

3.1. Передумови вдосконалення організації роботи команди NOS9

Створення платформи NOS9 відбувалося у контексті високої технологічної складності, що поєднувало розподілені дані, обробку метрик у режимі реального часу та вимоги до мінімальної затримки роботи системи. Характер домену спостережності (observability) визначив специфічні умови діяльності команди, які посилювали вплив факторів невизначеності, підвищеної когнітивної складності та необхідності швидкої адаптації до змін [4]. Водночас організаційна структура команди NOS9 перебувала на ранній стадії становлення, а окремі ролі поєднували в собі декілька векторів відповідальності.

Подальший розвиток платформи NOS9 вимагав формалізації принципів взаємодії між учасниками команди [11]. На ранніх етапах значна частина комунікацій здійснювалася у неформальному режимі, що було природним у середовищі стартапу. Однак збільшення кількості компонентів, розширення інтеграцій, зростання обсягу кодової бази та ускладнення експериментальних циклів HADI поступово створили потребу у систематизованих механізмах координації. Саме тому команда зіткнулася з необхідністю підсилення процедур управління, що включають структуровані зустрічі, синхронізацію пріоритетів та регулярну валідацію технічних рішень.

Особливої ваги набуло питання розподілу відповідальності між ролями. Складаючи вісім осіб, команда NOS9 працювала у форматі, у якому одна людина виконувала функції Product Owner, Scrum Master та фактично виконувала обов’язки первинного технічного керівника (Engineering Manager). Така модель була виправданою на етапі MVP, але вона створювала ризики надмірної концентрації рішень в одній ролі. Це впливало на продуктивність команди, адже кожна зміна вимагала участі цієї ключової фігури, а питання, пов'язані з UX, DevOps, Backend або Frontend, потребували додаткової координації.

Окрім внутрішніх організаційних факторів, відчутним став вплив зовнішнього контексту. Команда працювала у гібридному віддаленому форматі, що було пов’язано з різними часовими поясами та культурними відмінностями [12]. У таких умовах виникала потреба в удосконаленні асинхронних комунікацій, запровадженні чітких правил оновлення інформації та узгодженні очікувань щодо відповідей у Slack і Jira. Ці аспекти мали безпосередній вплив на швидкість ухвалення рішень і якість командної синхронізації.

Поступове накопичення технічного боргу стало ще одним ключовим фактором, який підштовхнув команду до необхідності зміцнення комунікаційних практик. Через швидку реалізацію функціональності в рамках HADI-експериментів та постійний розвиток VS Code-плагіна, дашборду й серверної системи, з’явилася потреба у регулярних обговореннях архітектурних рішень, формуванні RFC-документів та створенні прозорих критеріїв визначення пріоритетів.

Після аналізу вищевказаних чинників було прийнято рішення, що для досягнення стабільного розвитку платформи NOS9 команді необхідні покращені інструменти управління взаємодією, підтримка технічного лідерства та удосконалення процесів, які дозволять забезпечити гнучкість, узгодженість та високу якість роботи в умовах технологічної складності.

3.2. Інженерний менеджмент у продуктових IT-командах та його значення для розвитку продукту

Інженерний менеджмент у продуктових технологічних компаніях виступає ключовим механізмом забезпечення стабільної, передбачуваної та якісної роботи інженерних команд. У контексті створення платформи NOS9 значення інженерного менеджменту зростає через високий ступінь технологічної складності, різноманітність компонентів системи та поєднання декількох парадигм розроблення – Scrum, Lean Startup, DevOps та HADI-експериментування [5, 11, 18]. Така багатовекторність створює потребу в системних підходах до організації технічної роботи, які забезпечують

узгодженість командних дій, якість архітектурних рішень та низький рівень ризиків при швидкому масштабуванні функціональності.

3.2.1. Архітектурне лідерство та системне технічне мислення

Архітектура NOS9 включає потоки телеметрії, модулі обробки метрик у режимі реального часу, аналітичний шар, механізми зберігання даних, інтеграційні конвеєри та взаємодію з клієнтськими інструментами (зокрема VS Code-плагіном). Ця складність потребує не лише технічної компетентності, а й здатності забезпечувати цілісність архітектурної еволюції [14]. У ранній фазі розвитку продукту окремі рішення приймалися індивідуально розробниками відповідних компонентів, що є типовим для MVP-етапу. Однак із розширенням функціональності й збільшенням залежностей між підсистемами постала потреба в системному технічному лідерстві.

Архітектурне лідерство в цьому контексті включає:

- формування принципів модульності та інтерфейсної узгодженості компонентів;
- забезпечення відповідності технічних рішень довгостроковим стратегічним цілям;
- регулярні архітектурні рев'ю (Architecture Review) для запобігання розходженню компонентів [15];
- використання RFC (Request for Comments) для узгодження архітектурних та технічних рішень [14]. На відміну від ADR (Architecture Decision Records), що фіксує лише кінцеве рішення, RFC дає можливість організувати повноцінне обговорення пропозиції, зберегти всі варіанти та аргументи, а також забезпечити залученість усіх підкоманд;
- превентивний контроль технічних ризиків, пов'язаних зі змінами у схемах даних, навантаженням або API [16].

Відсутність формалізованої ролі технічного керівника (Engineering Manager) у команді NOS9 частково компенсується тим, що PO/SM здійснює базову координацію, однак це не замінює необхідності у технічному лідерстві. У складних системах без такої функції зростає ризик технічної фрагментації –

ситуації, коли рішення підкоманд перестають узгоджуватися між собою, а архітектура поступово втрачає цілісність. Тому на пост-MVP етапі архітектурне лідерство має бути підсилене окремою роллю або чітко визначеним процесом.

3.2.2. Управління технічним боргом на основі моделі міграцій

Одним із ключових викликів для NOS9 є природне накопичення технічного боргу [17], яке відбувається внаслідок:

- пришвидшеної розробки функцій MVP;
- великої кількості HADI-експериментів;
- еволюції IDE-плагіна;
- регулярних змін у потокових pipeline'ах;
- розширення Data Storage та API.

У традиційних командах технічний борг зазвичай усувається за принципом "коли з'явиться час". Проте підхід Вілла Ларсона [16] пропонує системну модель міграції. Міграція – це спланований, контрольований і чітко структурований процес зміни технічної системи. На відміну від рефакторингу, який часто має невизначений обсяг, міграції передбачають:

- визначення цілісного стану системи, до якого потрібно перейти;
- декомпозицію цілі на етапи, що можуть виконуватися паралельно;
- мінімізацію ризиків шляхом поступових змін;
- централізований технічний нагляд за виконанням;
- синхронізацію між командами, що працюють над різними частинами системи.

У застосуванні до NOS9 міграції можуть включати:

- перехід від монолітних сценаріїв збору даних до модульних pipeline-компонентів;
- уніфікацію протоколів передачі телеметрії;
- реорганізацію структури даних у сховищі для підвищення продуктивності;
- поетапне впровадження більш складних механізмів виявлення аномалій;

- міграцію частини функціональності ide-плагіна на окремі модулі, що зменшують навантаження на ide.

Завдяки застосуванню міграцій технічні зміни не блокують роботу команди, а навпаки – стають передбачуваними та керованими. Це важливо як для стабільності системи, так і для збереження продуктивності інженерів, адже неконтрольований технічний борг спричиняє приховане зростання складності, що з часом знижує швидкість постачання інкрементів.

3.2.3. DevOps-культура як інфраструктурна основа

DevOps у NOS9 виконує роль не лише практики автоматизації розгортання [18], а й стратегічної складової архітектури продукту. Оскільки NOS9 є системою спостережності, команда використовує власні інструменти для моніторингу продуктивності, стабільності та реагування на інциденти. Це створює унікальну ситуацію: DevOps-процеси стають частиною дослідницького циклу продукту.

DevOps-підходи NOS9 включають:

- CI/CD-пайплайни, що забезпечують швидке тестування та розгортання змін;
- автоматизовані перевірки продуктивності та регресій;
- інтеграцію з Prometheus і Grafana для діагностики;
- використання контейнеризації для уніфікації середовищ;
- регулярний аналіз Change Failure Rate (CFR) та Lead Time for Changes (LTC) [11].

DevOps у NOS9 прямо підтримує HADI-цикли, адже швидке розгортання та вимірювання результатів експериментів є критично важливими для перевірки продуктових гіпотез. У цьому контексті DevOps перестає бути окремою функцією і перетворюється на платформу операційного мислення [18].

3.2.4. Психологічна безпека, довіра та інженерна культура

У сучасних технологічних командах психологічна безпека є одним із найважливіших факторів продуктивності [12, 19, 20]. Для NOS9 це особливо актуально через:

- високий ступінь когнітивного навантаження;
- необхідність ухвалювати технічні рішення в умовах невизначеності;
- роботу з потенційно інцидентними сценаріями;
- віддалений формат взаємодії.

Психологічна безпека дозволяє:

- відкрито обговорювати помилки;
- ставити запитання без страху критики;
- пропонувати альтернативні технічні рішення;
- повідомляти про ризики на ранніх етапах;
- зберігати високий моральний стан команди.

Формування інженерної культури NOS9 включає:

- прозорість у прогнозах та оцінках;
- регулярні синхронізації;
- чіткі канали повідомлення про технічні ризики;
- механізми командної підтримки (Team Sentiment Survey);
- обмін знаннями через воркшопи та дизайн-сесії.

Завдяки цьому команда здатна ефективно працювати навіть в умовах інтенсивного темпу релізів і високих вимог до стабільності системи.

Також в умовах розробки високотехнологічних проєктів важливим чинником є формування комфортного середовища діяльності всіх членів команди. Мушинський О.Ю. пропонує будувати таке середовище базуючись на тріаді «безпека–довіра–благополуччя».

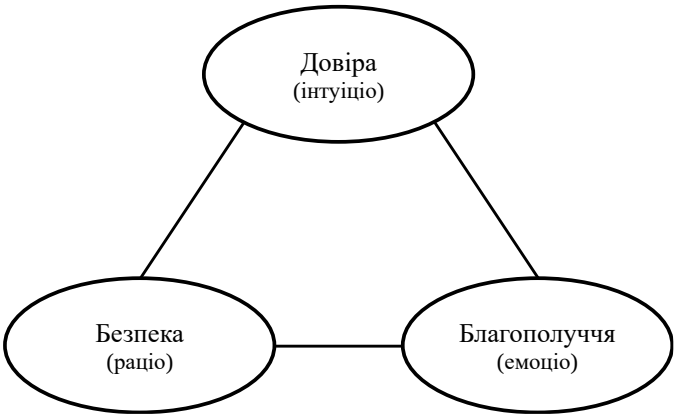

Рисунок 3.1 – Модель системної тріади ключових факторів для побудови комфортного середовища команди

Джерело: [21]

3.2.5. Управління залежностями та когнітивним навантаженням у команді

Одним із ключових аспектів інженерного менеджменту у складних технологічних продуктах є управління залежностями між компонентами та зменшення когнітивного навантаження на інженерів [22]. Продукт складається з низки підсистем, що функціонують як у слабко зв'язаному, так і у високо взаємозалежному режимах. До них належать: модуль збору телеметрії, конвеєри обробки даних, серверна аналітична система, дашборд, а також плагін для VS Code IDE. Розвиток цих компонентів потребує постійної координації між командами Frontend, Backend і DevOps.

Залежності між компонентами створюють низку викликів:

- необхідність точного узгодження API та форматів даних;
- синхронізацію розгортання нових версій;
- ризики інцидентів у разі несумісних змін;
- збільшення часу на прийняття рішень через складність контекстів.

Когнітивне навантаження зростає пропорційно кількості систем, за які відповідає інженер [16]. У NOS9 це особливо відчутно через необхідність роботи з потоковими даними, а також складність VS Code-плагіна, який має забезпечувати стабільність і водночас низький вплив на продуктивність IDE.

Для зменшення когнітивного навантаження доцільними є такі інженерні практики:

1. Формалізація інтерфейсів - документація API та форматів даних через OpenAPI [23], RFC та внутрішні протоколи дозволяє мінімізувати непорозуміння між компонентами.

2. Використання контрактного тестування (Contract Testing) [11] дозволяє перевіряти сумісність компонентів без необхідності піднімати всю систему.

3. Введення технічних менторів у межах підкоманд зменшить тиск на окремих розробників, забезпечуючи обмін знаннями.

4. Перехід до «компонентної» відповідальності. Це означає, що кожен інженер працює в межах чітко визначеного простору, що знижує когнітивне навантаження та підвищує автономність.

3.2.6. Значення інженерного менеджменту для подальшої еволюції

Інженерний менеджмент формує основу для стійкого розвитку NOS9. У складних системах успіх залежить не лише від технічних рішень, а й від здатності команди підтримувати узгодженість, уникати надмірної складності та створювати передбачуваний процес розробки. Саме інженерний менеджмент забезпечує баланс між швидкістю інновацій і стабільністю системи.

У контексті NOS9 це означає:

- підтримку структурованої архітектурної еволюції;
- впровадження міграцій для зменшення технічного боргу;
- розвиток DevOps-культури;
- формування психологічної безпеки;
- підтримку процесів навчання та командного розвитку.

Таким чином, інженерний менеджмент є фундаментом для переходу продукту від MVP до масштабованої платформи.

3.3. Управлінські рекомендації щодо підвищення ефективності роботи команди

Управлінські рекомендації для команди ґрунтуються на аналізі особливостей роботи, визначених у попередніх підрозділах, а також на сучасних підходах інженерного менеджменту, Agile-практик і практичних принципів розвитку високопродуктивних технічних команд. Метою є формування системи рішень, які дозволять команді підвищити узгодженість роботи, зберегти стійкість процесів у міру масштабування продукту та підсилити здатність до інноваційного розвитку.

3.3.1. Оптимізація організації ролей та підсилення технічного лідерства

Команда працювала в умовах, коли функції Product Owner та Scrum Master виконувала одна особа. Такий підхід є типовим для ранніх фаз стартапу, проте зі

збільшенням складності продукту та кількості взаємозалежних компонентів зростає потреба у розподілі цих ролей. Концентрація занадто широких обов'язків в одній позиції створює ризики:

- зниження якості продуктового аналізу;
- перевантаження продуктового менеджера операційними задачами;
- нестачі уваги до системного технічного розвитку;
- уповільнення прийняття рішень через значний обсяг координаційної роботи.

Тому обґрунтованою рекомендацією є поетапне розділення PO/SM на окремі ролі:

- Product Owner – зосереджений на формуванні продуктової логіки, взаємодії зі стейкголдерами та управлінні Product Backlog.
- Scrum Master – відповідальний за дотримання Scrum-процесів, фасилітацію зустрічей та підтримку командної динаміки.

Паралельно рекомендовано формалізувати роль Engineering Manager або Technical Lead [16], відповідального за наступні аспекти:

- технічну узгодженість системи;
- контроль технічного боргу;
- організацію архітектурних сесій;
- наставництво інженерів;
- технічну частину HADI-експериментів;
- стандартизацію DevOps-практик.

Введення такої ролі дозволяє зменшити навантаження на розробників, підвищити якість технічних рішень і забезпечити стабільність архітектури під час масштабування.

3.3.2. Удосконалення комунікаційних підходів

У віддаленій команді, що працює у різних часових поясах, якість комунікацій є критично важливим чинником [12]. У NOS9 комунікації відбувалися переважно асинхронно, однак збільшення складності потребує

впровадження структурованих правил, які забезпечать прозорість та ефективність взаємодії.

Рекомендації включають:

1. Формалізацію Team Agreements, що мають охоплювати:

a. правила відповіді у Slack;

b. способи ескалації питань;

c. очікування щодо асинхронних оновлень;

d. формати проведення мітингів.

2. Впровадження структурованих технічних синхронізацій - регулярні Architecture Sync, API Review [15] та HADI Sync [10];

3. Використання централізованої системи знань, яка має містити RFC, технічні протоколи, описи архітектури, актуальну інформацію щодо експериментів;

4. Визначення прозорих каналів комунікації. Наприклад: Slack – поточні питання, Jira – постановка завдань, GitHub – технічні рев'ю, Notion або GitBook – документація.

Ці заходи дозволяють мінімізувати втрати інформації, покращити узгодженість та зменшити операційні ризики.

3.3.3. Розвиток управління експериментами (HADI) та аналітичних циклів

HADI-цикли є основою продуктового дослідження, однак їх ефективність значною мірою залежить від того, наскільки системно команда працює з гіпотезами. Для підвищення ефективності експериментів доцільно:

1. Запровадити регулярний HADI Review, де команда аналізує результати виконаних гіпотез, валідність отриманих даних та вплив на продуктову дорожню карту;

2. Стандартизувати формат постановки гіпотез, який має включати чітке формулювання припущення, визначення метрик Impact, Confidence, Ease; опис дій, план збору даних, очікувані інсайти;

3. Вести централізований каталог гіпотез, що дозволить уникнути дублювання експериментів та формувати знання команди.

4. Інтегрувати результати експериментів у Roadmap, що підвищить прозорість прийняття рішень і забезпечить логічну послідовність розвитку продукту.

3.3.4. Перехід до Kanban як еволюція Scrum-процесів

Хоча Scrum був ефективним на ранніх етапах розвитку NOS9, подальше масштабування та збільшення технічної складності роблять Kanban [13] природним наступним кроком. Kanban забезпечує:

- безперервний потік роботи;
- зниження навантаження на планувальні зустрічі;
- більшу гнучкість у реакції на технічні ризики;
- кращу інтеграцію HADI-експериментів,
- оптимізацію Cycle Time та Lead Time.

Для переходу до Kanban доцільно:

- ввести WIP-ліміти для кожної колонки дошки;
- використовувати Cumulative Flow Diagram для виявлення вузьких місць;
- зберегти Retrospective як інструменти зворотного зв’язку;
- зосередити увагу не на плануванні ітерацій, а на безперервній пріоритизації.

Таке поєднання дозволить зберегти структурованість роботи, характерну для Scrum, але водночас отримати переваги потокової моделі.

3.3.5. Рекомендації для масштабування команди NOS9

У процесі переходу від MVP до повноцінної SaaS-платформи команда неминуче зіштовхнеться з необхідністю масштабування. Це стосується як кількісного зростання команди, так і структурної складності самої системи. Для забезпечення стабільності розвитку продукту доцільно впровадити низку

управлінських підходів, спрямованих на підтримку ефективності під час масштабування.

1. Перехід до моделі Product Squads [22].

У міру зростання продукту окремі функціональні напрями потребуватимуть автономності. Доцільно формувати невеликі кросфункціональні підкоманди (Squads) із власними зонами відповідальності:

- Data Processing Squad – ядро обробки метрик і логів;
- IDE Experience Squad – розвиток IDE-плагінів;
- Observability Dashboard Squad – інтерфейси, візуалізація, аналітика.

Це знижує навантаження на ключових технічних фахівців та підвищує швидкість прийняття рішень.

2. Стандартизація DevOps-процесів [18] перед масштабуванням.

Щоб уникнути хаотичного зростання інфраструктури, необхідно:

- уніфікувати пайплайни CI/CD;
- стандартизувати інфраструктурні шаблони;
- визначити SLO/SLA для кожного сервісу.

Такі стандарти дозволять уникнути інцидентів, пов’язаних із різними підходами до розгортання.

3. Введення інженерних стандартів якості.

Перед масштабуванням важливо, щоб команда працювала в єдиному технічному стилі:

- код-рев’ю за чіткими правилами;
- формалізація RFC-процесу для ухвалення технічних рішень дозволить уникнути фрагментації архітектури, підсилити комунікації між підкомандами та створити інституційну пам’ять продукту;
- автоматизовані тести як частина Definition of Done.

4. Розвиток онбордингу для нових членів команди [12].

Оскільки нові учасники з’являтимуться частіше, необхідно:

- створити єдиний портал документації;
- окремі гайди для IDE-плагіна, бекенду та DevOps;
- чеклісти для налаштування робочого середовища;

- програму менторства.

Ефективний онбординг значно знижує когнітивне навантаження на команду та прискорює адаптацію новачків.

3.3.6. Інтеграція психологічної безпеки та інженерної культури

У процесі масштабування важливо не лише оптимізувати технічні процеси, але й підтримати інженерну культуру. Психологічна безпека є ключовою умовою продуктивності технічних команд, що підтверджено дослідженнями [12, 13, 19, 22].

У контексті NOS9 психологічна безпека проявляється в таких практиках:

1. Прозорість у комунікаціях. Команда має мати можливість відкрито говорити про проблеми, складнощі та помилки. Це дозволяє своєчасно виявляти ризики та уникати накопичення технічного боргу.
2. Нормалізація експериментів та помилок. Оскільки NOS9 активно використовує HADI-підхід, важливо створити культуру, у якій невдачі експериментів сприймаються як джерело знань, а не як помилка команди.
3. Регулярні командні ретроспективи. Вони дозволяють виявляти напруження всередині команди, обговорювати процеси та приймати рішення щодо покращення взаємодії.
4. Використання інструментів для вимірювання емоційного стану команди. Team Sentiment Survey [19] дає можливість фіксувати психологічний стан учасників та реагувати на проблеми проактивно.
5. Підсилення менторства та обміну знаннями. У складних системах знання часто розподілені між окремими людьми. Менторство допомагає зменшити ризики «вузьких місць» та підвищити командну автономність.

3.3.7. Загальний підсумок рекомендацій

Запропоновані управлінські рішення дозволяють побудувати основу для подальшого масштабування продукту, забезпечують високу якість комунікацій та сприяють розвитку інженерної культури. У поєднанні з інженерним

менеджментом вони формують цілісну модель організації роботи, здатну забезпечити стабільний розвиток продукту в умовах росту та збільшення вимог.

Висновки до розділу 3

У третьому розділі здійснено комплексне дослідження лідерства, організації взаємодії та комунікацій у команді, що розробляє платформу NOS9. Результати аналізу засвідчили, що ефективність роботи над технологічно складним продуктом визначається не лише вибором методології чи інженерних практик, а передусім здатністю команди адаптуватися до змін, підтримувати психологічну безпеку, забезпечувати архітектурну узгодженість і дотримуватися дисципліни процесів.

Розроблено комплекс управлінських рекомендацій, спрямованих на підвищення ефективності команди. Їх реалізація створює підґрунтя для переходу від MVP до повноцінної масштабованої платформи. Запропоновані заходи охоплюють:

- розділення ролей Product Owner і Scrum Master та формалізацію Engineering Manager;
- удосконалення комунікацій через Team Agreements, структуровані технічні синхронізації та посилену документацію;
- розвиток HADI-циклів як інструменту продуктового дослідження;
- перехід до потокової моделі Kanban та впровадження WIP-лімітів;
- підготовку до масштабування через створення Product Squads, стандартизацію DevOps та розвиток онбордингу;
- інституціалізацію психологічної безпеки та інженерної культури.

Узагальнюючи результати розділу, можна стверджувати, що ефективне управління командою продукту базується на взаємопов’язаних компонентах: структурованих процесах, відповідній організації ролей, гнучких методологіях, технічному лідерстві та культурі довіри. Сукупність досліджених підходів створює надійну основу для подальшого розвитку продукту, зменшення технічних ризиків, прискорення інноваційного циклу та підвищення стійкості команди в умовах зростання складності.

ВИСНОВКИ

У кваліфікаційній роботі було впроваджено функції гнучкого управління для створення та інтеграції платформи моніторингу «NOS9», на підвищення стабільності програмних систем та зниження витрат, пов’язаних з пізнім виявленням інцидентів. На основі аналізу бізнес-контексту, потреб користувачів та специфіки ринку спостережності сформовано обґрунтовану концепцію продукту, яка передбачає мінімально інвазивну інтеграцію, комплексність збору даних, підтримку Freemium-моделі та надання інженерам інструментів для виявлення проблем на ранніх етапах розробки.

У роботі визначено ключові умови, які роблять процес створення NOS9 високовимогливим до організації управління: технологічна невизначеність, складність даних, необхідність роботи з різнорідними середовищами та залежність якості рішення від швидкості зворотного зв’язку від користувачів. Це обумовило доцільність застосування гнучких методів, що дозволяють поєднувати короткі інкременти із перевіркою продуктових гіпотез, забезпечувати адаптивність до змін та зосереджувати зусилля команди на створенні перевірюваної цінності.

У ході виконання роботи було проаналізовано сучасний стан, ключові проблеми та найкращі практики моніторингу й виявлення аномалій у провідних IT-компаніях. Це дозволило сформувати обґрунтовані вимоги до архітектури та функціональності платформи спостережності, а також уточнити контекст її використання в реальних умовах розробки та експлуатації програмних систем.

На основі отриманих результатів було запропоновано та обґрунтовано рішення платформи «NOS9», спроєктованої як SaaS-рішення з використанням гнучких методів управління розробкою на основі підходу Scrum. Окрему увагу приділено побудові процесів, які забезпечують поєднання високої швидкості інкрементальної розробки з вимогами до надійності та масштабованості системи спостережності.

Також було описано особливості гнучкого управління створенням мінімально життєздатного продукту (MVP) платформи «NOS9» та здійснено її експериментальну апробацію на реальних кейсах у середовищі IT-компаній. Це

дало змогу перевірити життєздатність продуктового підходу, коректність сформульованих гіпотез та здатність платформи інтегруватися в існуючі процеси команд розробки.

Отримані результати підтверджують, що застосування гнучкого управління в умовах високої складності сприяє підвищенню керованості процесу створення інноваційних програмних продуктів та забезпечує зменшення ризиків, пов’язаних із невизначеністю. Запропоновані управлінські рекомендації можуть бути адаптовані для інших продуктів у сфері спостережності та DevOps, а також використовуватися IT-компаніями як методична основа для організації процесів розробки SaaS-рішень. Платформа NOS9, відповідно до представленої концепції, має потенціал для подальшого розвитку, масштабування та впровадження у практичну діяльність організацій, що прагнуть покращити якість своїх програмних систем.

СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ

1. Ponemon Institute. Cost of Data Center Outages. Report 2016. URL: https://planetaklimata.com.ua/instr/Liebert_Hiross/Cost_of_Data_Center_Outages_2016_Eng.pdf.

2. Atlassian. Cost of Downtime. Incident Management KPIs. URL: https://www.atlassian.com/incident-management/kpis/cost-of-downtime.

3. Osterwalder A. The Business Model Ontology: A Proposition in a Design Science Approach. PhD Thesis. Université de Lausanne, Faculté des Hautes Études Commerciales (HEC Lausanne), 2004. URL: https://iris.unil.ch/entities/publication/2946a1c2-6d59-4440-99e7-f768ca26b8d3

4. Snowden, D., Boone, M. A Leader's Framework for Decision Making. Harvard Business Review, 2007.

5. Schwaber, K., Sutherland, J. The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. 2020. URL: https://scrumguides.org.

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

7. Семененко Ю. Роль KPI та OKR в ефективності діяльності компанії. Herald of Khmelnytskyi National University. Economic Sciences, 2023, № 324(6), с. 227–235. DOI: https://doi.org/10.31891/2307-5740-2023-324-6-37.

8. Agile Alliance. What is Agile? URL: https://agilealliance.org/agile101/.

9. Clardy A. 70-20-10 and the Dominance of Informal Learning: A Fact in Search of Evidence. Human Resource Development Review, 2018, 17(2), pp. 153–178. DOI: https://doi.org/10.1177/1534484318759399.

10. Мушинський О. Ю. Стратегія розвитку талантів у гібридному робочому середовищі. Інтелектуальні інформаційні системи в управлінні проєктами та програмами: матеріали Міжнар. наук.-практ. конф. (Харків–Коблево, 15–20 верес. 2025 р.). Харків: ХНУРЕ, 2025. С. 221–224. URI: https://dspace.krok.edu.ua/handle/krok/7970.

11. Forsgren N., Humble J., Kim G. Accelerate: The Science of Lean Software and DevOps. IT Revolution, 2018.

12. DeMarco T., Lister T. Peopleware: Productive Projects and Teams. 3rd ed. Addison-Wesley, 2013.

13. Anderson D. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010.

14. Ford N., Parsons R., Kua P. Building Evolutionary Architectures: Support Constant Change. O'Reilly Media, 2017.

15. Microsoft Engineering Playbook. Patterns and practices for high-performing engineering teams. Microsoft Corporation, 2024. URL: https://microsoft.github.io/code-with-engineering-playbook/.

16. Larson W. An Elegant Puzzle: Systems of Engineering Management. Stripe Press, 2019.

17. Cunningham W. The WyCash Portfolio Management System // OOPSLA '92 Experience Report, 1992.

18. Kim G., Humble J., Debois P., Willis J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2016.

19. Google. Project Aristotle: Understanding Team Effectiveness. 2016. URL: https://rework.withgoogle.com/intl/en/guides/understanding-team-effectiveness.

20. Edmondson A. The Fearless Organization: Creating Psychological Safety in the Workplace for Learning, Innovation, and Growth. Wiley, 2018.

21. Мушинський О. Ю. Розвиток лідерства в управлінні проєктними командами. Вчені записки Університету «КРОК», 2024, № 4(76), с. 165–173. DOI: https://doi.org/10.31732/2663-2209-2024-76-165-173.

22. Skelton M., Pais M. Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution Press, 2019.

23. OpenAPI Initiative. OpenAPI Specification, Version 3.1. 2021. URL: https://www.openapis.org/.

ДОДАТКИ

Додаток 1. Термінологічний словник кваліфікаційної роботи

Гнучкі методології, підходи та фреймворки:

1. Agile practices (гнучкі практики) – методи і принципи адаптивного управління розробкою продукту, які забезпечують швидке реагування на зміни та інкрементальне створення цінності.
2. Agile Software Development (гнучка розробка програмного забезпечення) – сукупність процесів, що забезпечують короткі цикли постачання функціональності та постійний зворотний зв'язок від користувачів.
3. Scrum framework (фреймворк Scrum) – ітеративний підхід до розробки, що включає визначені ролі, події та артефакти для інкрементального постачання продукту.
4. Scrum Team (Scrum-команда) – кросфункціональна група фахівців, що створює продукт у межах Scrum.
5. Product Owner (власник продукту) – роль, відповідальна за максимізацію цінності продукту, пріоритизацію беклогу та взаємодію зі стейкголдерами.
6. Scrum Master (майстер Scrum) – фасилітатор, який забезпечує дотримання правил Scrum, підтримує продуктивність команди та усуває перешкоди.
7. Sprint (спринт) – фіксований у часі період (у NOS9 – 2 тижні), протягом якого команда створює завершений інкремент продукту.
8. Increment (інкремент продукту) – завершена частина функціональності, що потенційно може бути доставлена користувачам.
9. Kanban (канбан) – метод управління потоками завдань на основі візуалізації, обмеження незавершеної роботи (WIP) та оптимізації часу виконання.
10. Lean Startup (ощадливий запуск продукту) – метод створення продуктів в умовах невизначеності на основі швидкої перевірки гіпотез та MVP.

11. Hypothesis-Driven Development (розроблення, кероване гіпотезами) – підхід, що передбачає тестування гіпотез перед масштабуванням рішень.
12. HADI-cycle (гіпотеза – дія – дані – інсайт) – ітераційний цикл перевірки гіпотез, застосований у NOS9 для розвитку продукту.
13. Cynefin Framework (рамка Кінефін) – модель класифікації проблемних середовищ; продукт NOS9 належить до домену Complex.

Продуктові артефакти та стратегічні інструменти:

1. Product Vision Board (дошка бачення продукту) – стратегічний інструмент, який визначає сегменти користувачів, проблеми, ціннісну пропозицію та бізнес-результати продукту.
2. Business Model Canvas (канва бізнес-моделі) – структурна модель дев'яти блоків, що визначає логіку створення та доставки цінності продукту.
3. Lean Product Canvas (ощадлива канва продукту) – модифікація Business Model Canvas, орієнтована на ранню валідацію продуктового рішення.
4. Persona Canvas (канва персони) – структурований опис архетипу користувача з його потребами, бар'єрами, контекстами та очікуваннями.
5. Jobs-to-be-Done (JTBD, робота, яку треба виконати) – підхід до визначення потреб користувача через очікуваний результат
6. Customer Journey (клієнтська подорож) – послідовність взаємодій користувача із продуктом, що визначає ключові дотики та бар'єри.
7. Product Roadmap (дорожня карта продукту) – високорівневий план розвитку продукту, що містить фази, цілі та ключові результати.
8. Product Backlog (продуктовий беклог) – пріоритизований список вимог, функціональності та технічних робіт продукту.
9. User Story (користувацька історія) – опис потреби користувача у формі «як… хочу… щоб…».
10. Acceptance Criteria (критерії прийнятності) – набір умов, виконання яких підтверджує завершеність User Story.
11. Success Metrics (метрики успіху) – кількісні показники, що використовуються для перевірки досягнення очікуваного результату.

12. OKR (Objectives and Key Results) – система постановки вимірюваних цілей, застосована для планування розвитку NOS9.

Observability, телеметрія та домен аномалій:

1. Observability (спостережуваність) – здатність системи надавати достатню інформацію про свій внутрішній стан через телеметричні дані.
2. Telemetry (телеметрія) – метрики, логи й трасування, що збираються із системи.
3. Metrics (метрики) – числові показники стану системи (CPU, Memory, Latency тощо).
4. Logs (логи) – структуровані або неструктуровані текстові записи подій у системі.
5. Tracing (трасування) – механізм відслідковування шляху виконання запитів у розподілених системах.
6. Anomaly detection (виявлення аномалій) – алгоритмічні процедури ідентифікації нетипової поведінки системи.
7. Telemetry pipeline (телеметричний конвеєр) – механізми збору, обробки та передачі телеметричних даних у NOS9.
8. Real-time processing (обробка в режимі реального часу) – здатність NOS9 обробляти телеметрію із мінімальною затримкою.
9. OpenTelemetry – відкритий стандарт збору й експорту телеметрії, використаний у NOS9.
10. Prometheus – система моніторингу та збору метрик, що інтегрується з NOS9.
11. APM (Application Performance Monitoring) – інструменти моніторингу продуктивності застосунків, які виступають конкурентними альтернативами NOS9.

Інженерні практики, архітектура та DevOps:

1. Architecture Review (архітектурний огляд) – процес оцінювання архітектурних рішень для забезпечення їх узгодженості та стабільності.

2. RFC (Request for Comments) – документ, що ініціює колективне обговорення технічного рішення.
3. ADR (Architecture Decision Record) – запис архітектурного рішення; у NOS9 замінений процесом RFC.
4. Technical debt (технічний борг) – накопичення неідеальних технічних рішень, що знижують гнучкість системи.
5. Migration model (модель міграцій) – структурований метод усунення технічного боргу на основі поетапних змін.
6. CI/CD (Continuous Integration / Continuous Delivery) – автоматизовані процеси інтеграції, тестування та доставки змін у продукт.
7. Docker / Kubernetes – інструменти контейнеризації та оркестрації, які використовуються у NOS9.
8. VS Code extension (розширення Visual Studio Code) – клієнтська частина NOS9, що забезпечує аналітичні підказки в IDE.
9. API (Application Programming Interface) – інтерфейси для взаємодії компонентів NOS9.
10. RBAC (Role-Based Access Control) – механізм керування доступами користувачів у системі.
11. Contract Testing (контрактне тестування) – метод перевірки сумісності компонентів через визначені API-контракти.
12. Pipeline Reliability (надійність конвеєра) – стабільність CI/CD та потоків обробки даних.

Командна взаємодія, комунікації та інженерна культура

1. Team Agreements (командні домовленості) – правило взаємодії, які регулюють комунікацію, відповідальність і очікування учасників.
2. Team Canvas (командна канва) – інструмент визначення ролей, цінностей і принципів співпраці.
3. Team Health Check – регулярне оцінювання командного стану за ключовими параметрами (комунікація, навантаження, довіра).
4. Psychological safety (психологічна безпека) – атмосфера довіри, що дозволяє відкрито обговорювати помилки й пропозиції.

5. Cognitive load (когнітивне навантаження) – кількість інформації, яку інженер повинен утримувати у робочому контексті.
6. Mentorship (менторство) – процес передачі знань у команді з метою прискорення навчання та зниження ризику помилок.
7. Asynchronous communication (асинхронна комунікація) – модель взаємодії у віддалених командах без вимоги миттєвої відповіді.
8. Stand-up meeting (щоденна зустріч) – коротка синхронізація команди щодо прогресу та перешкод.

Метрики, інструменти контролю процесів та ефективності

1. Velocity (швидкість команди) – кількість робочих одиниць (story points), завершених за спринт.
2. Burndown Chart (діаграма згорання) – графік виконання завдань у межах спринту.
3. Cumulative Flow Diagram (кумулятивна діаграма потоку) – інструмент аналізу стабільності роботи та виявлення вузьких місць.
4. Cycle Time (час циклу) – час від початку роботи над завданням до його завершення.
5. Lead Time (час постачання) – час від появи завдання у беклозі до надання результату.
6. MTTD (Mean Time to Detect) – середній час виявлення інциденту.
7. MTTR (Mean Time to Recover) – середній час відновлення після інциденту.
8. Change Failure Rate (частка змін, що спричинили помилки) – ключова метрика DevOps.
9. Deployment Frequency (частота розгортання) – кількість релізів за одиницю часу.

Комерційні та організаційні поняття:

1. SaaS (Software as a Service) – модель хмарного програмного забезпечення, що передбачає доступ до продукту через підписку.
2. Freemium model (фріміум-модель) – модель, у якій базовий функціонал доступний безкоштовно, а розширений – у платному тарифі.

3. Customer Segmentation (сегментація клієнтів) – поділ користувачів на групи за потребами та JTBD.
4. Value Proposition (ціннісна пропозиція) – користь, яку продукт надає цільовому сегменту.
5. North Star Metric (північна зірка продукту) – ключова метрика стратегічної цінності, у NOS9 – скорочення виробничих інцидентів.