

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Вебсистема управління життєвим циклом
інноваційних ідей та формування команд»**

Виконав студент групи ПЗ-22-4

ВОЗНЮК Андрій Володимирович

Керівник роботи:

МОРОЗОВ Андрій Васильович

Рецензент:

ТАЛАВЕР Олег Володимирович

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ВОЗНІЮК Андрій Володимирович**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: **«Вебсистема управління життєвим циклом інноваційних ідей та формування команд»,**

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **вебсистеми управління життєвим циклом інноваційних ідей та формування команди.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	Морозов А.В.	27.02.2026р.	02.03.2026р.
2	Морозов А.В.	02.03.2026р.	13.03.2026р.
3	Морозов А.В.	13.03.2026р.	22.05.2026р.

РЕФЕРАТ

Випускна робота бакалавра на тему «Вебсистема управління життєвим циклом інноваційних ідей та формування команд». Робота складається зі вступу, основної частини, висновків та переліку використаних джерел. Загальний обсяг звіту становить 89 сторінок, робота містить 31 рисунок, 3 таблиці та 35 джерел літератури.

В межах звіту було здійснено всебічне дослідження предметної області краудсорсингу, віртуальних стартап-інкубаторів та платформ для спільної розробки проєктів. Розглянуто особливості процесів генерації інноваційних ідей, управління життєвим циклом ініціатив та специфіку пошуку і залучення кандидатів для формування команд у цифровому просторі.

Результати проведеного переддипломного дослідження дозволили чітко сформулювати функціональні та технічні специфікації для проєктованої платформи. Програмна реалізація системи базуватиметься на комплексному технологічному стеку (Symfony, React, Redux, NestJS, MySQL, Docker, PHP workers, Mercure). Це дозволить автоматизувати процеси обробки заявок і транзакцій рейтингової системи, досягти максимальної реактивності інтерфейсу, а також інтегрувати великі мовні моделі та інструменти штучного інтелекту (LLM, Gemini API, Ollama, LangChain, LangGraph) для забезпечення роботи AI-асистента стейкхолдера і структурування неформатованих ідей.

Ключові слова: вебсистема управління ідеями, формування команд, краудсорсинг, штучний інтелект, Symfony, React, NestJS, Ollama, LangGraph, архітектура програмного забезпечення, гейміфікація, Docker.

					ІІЗ – 121 – 26 – ІІЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебсистема управління життєвим циклом інноваційних ідей та формування команд.	Літ.	Арк.	Аркушів
Розроб.		А. В. ВОЗНЮК						
Керівник		А.В. МОРОЗОВ					3	89
Рецензент		О.В. ТАЛАВЕР				Житомирська політехніка, група ІІЗ-22-4		
Зав. каф.		Т.А. ВАКАЛЮК						

ABSTRACT

Bachelor's thesis on the topic "Web System for Managing the Lifecycle of Innovative Ideas and Team Formation." The paper consists of an introduction, main body, conclusions and a list of references. The total volume of the report is 89 pages; the paper contains 31 figures, 3 tables, and 35 references.

Within the scope of the report, a comprehensive study of the subject area of crowdsourcing, virtual startup incubators, and platforms for collaborative project development was conducted. The features of innovative idea generation processes, initiative lifecycle management, and the specifics of searching for and attracting candidates for team formation in the digital space were examined.

The results of the conducted pre-diploma research allowed for the clear formulation of functional and technical specifications for the designed platform. The software implementation of the system will be based on a comprehensive technology stack (Symfony, React, Redux, NestJS, MySQL, Docker, PHP workers, Mercure). This will make it possible to automate the processing of applications and rating system transactions, achieve maximum interface reactivity, and integrate large language models and artificial intelligence tools (LLM, Gemini API, Ollama, LangChain, LangGraph) to power a stakeholder AI assistant and structure unformatted ideas.

Keywords: idea management web system, team formation, crowdsourcing, artificial intelligence, Symfony, React, NestJS, Ollama, LangGraph, software architecture, gamification, Docker.

		<i>A.B. БОЗНІОК</i>			ІІЗ – 121 – 26 – ІІЗ	<i>Арк.</i>
		<i>A.B. МОРОЗОВ</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>4</i>

ЗМІСТ

РЕФЕРАТ	4
ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ	8
1.1 Поставлення завдань та їх вирішення	8
1.2. Аналіз аналогів програмного продукту.....	12
1.3. Вибір архітектури системи	20
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	23
Висновки до першого розділу	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБСИСТЕМА	30
2.1 Визначення архітектури інтеграції штучного інтелекту	30
2.2 Розробка інформаційного забезпечення та структури бази даних	33
2.3 Проєктування алгоритмів роботи системи та взаємодії	38
2.4 Серверне забезпечення.....	43
2.5 Реалізація CI/CD та архітектури збереження файлів	48
Висновки до другого розділу.....	52
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ	54
3.1 Опис роботи з програмним додатком (Опис інтерфейсу).....	54
3.2 Адаптивність та доступність додатка.....	62
3.3 Роутинг та маршрутизація	66
3.4 User Stories та User Flows.....	70
3.5 Заходи безпеки бази даних	73
3.6 Тестування додатку	76
Висновки до третього розділу	80
ВИСНОВКИ.....	82
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	84
ДОДАТКИ.....	87

ВСТУП

Актуальність обраного напрямку дослідження пояснюється активним переходом процесів спільної роботи, стартап-інкубації та проєктного менеджменту у цифровий простір. Розробка дієвих механізмів управління інноваційними ідеями та швидкого формування команд є пріоритетним питанням, оскільки успішність будь-якої ініціативи безпосередньо залежить від ефективності взаємодії стейкхолдерів, валідації концепцій та прозорості розподілу ролей.

Зі збільшенням кількості користувачів та інтенсифікацією потоку нових ініціатив, завдання з управління життєвим циклом проєктів, інтеграції систем гейміфікації (економіка балів, досягнення) [4] та структурування інформації набувають особливої складності. Це зумовлює потребу в програмних продуктах із високим рівнем продуктивності, захищеності та системної гнучкості. Орієнтація на готові системи управління проєктами часто не дозволяє повноцінно реалізувати специфічні вимоги стартап-середовища, такі як використання штучного інтелекту для генерації ідей або ведення глобального рейтингу учасників. У цьому контексті проєктування інноваційної вебсистеми, що базується на сучасній розподіленій архітектурі та мікросервісних підходах, виступає актуальним науково-технічним завданням. Така структура дозволяє поєднати цілісність бізнес-логіки з можливістю безшовної інтеграції складних модулів (AI-асистент, інструменти аналітики) та високою адаптивністю до масштабування.

Метою роботи є всебічне дослідження сегмента краудсорсингових платформ та віртуальних стартап-інкубаторів [12], аналіз функціонуючих ринкових рішень та розробка комплексу технічних вимог до програмної вебсистеми управління життєвим циклом інноваційних ідей та формування команд, програмна реалізація якої запланована у межах дипломного проєкту.

1. Для досягнення поставленої мети необхідно вирішити такі завдання:

Дослідити архітектуру ринку та моделі взаємодії користувачів у сфері

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		6

краудсорсингу і тімбілдіngu для визначення пріоритетних функціональних можливостей системи.

2. Розробити логічну структуру взаємодії ключових компонентів платформи: від механізмів обробки неформатованих ідей за допомогою штучного інтелекту до транзакцій рейтингової системи та управління ризиками.
3. Здійснити порівняльну оцінку актуальних технологічних стеків та існуючих платформ для спільної роботи, ідентифікувавши їхні архітектурні обмеження та переваги [10].
4. Аргументувати доцільність застосування обраного комплексу технологій (Symfony, React, NestJS, інструменти AI) як оптимального підходу для створення масштабованого вебсервісу та підготувати технічне завдання на його проєктування.

Об’єктом дослідження є процеси функціонування краудсорсингових систем та архітектура цифрової інфраструктури для забезпечення управління життєвим циклом інноваційних ідей і формування команд в онлайн-середовищі.

Предметом дослідження є сукупність архітектурних шаблонів, технологічних стеків та інженерних методик проєктування масштабованих вебплатформ з інтеграцією інструментів штучного інтелекту та систем гейміфікації.

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Поставлення завдань та їх вирішення

У процесі проходження переддипломної практики основну увагу було зосереджено на розробці архітектурного базису вебсистеми управління життєвим циклом інноваційних ідей та формування команд. Головне завдання полягало у трансформації результатів аналізу ринку краудсорсингу та віртуальних інкубаторів у цілісну технічну модель майбутнього продукту. Цей етап включав не лише проєктування інтерфейсних рішень, а й деталізацію алгоритмів взаємодії між користувачами, інтеграцію систем штучного інтелекту для обробки природної мови (NLP) [1] та створення складної бізнес-логіки управління рейтинговою системою.

Особливості функціонування платформ для спільної роботи диктують специфічні вимоги до побудови користувацької взаємодії. Проєктована система передбачає гнучке розмежування прав доступу та інструментарію на основі рольової моделі: стейкхолдерів (які генерують ідеї та керують проєктами), рядових користувачів (які подають заявки на участь та виконують завдання) та адміністраторів (з доступом до інструменту імперсонації для діагностики та підтримки). Однією з найбільш критичних технічних перепон для існуючих рішень у цій ніші є низька ефективність процесу формування ідей з неструктурованих даних та відсутність прозорості системи мотивації (гейміфікації), що призводить до низького рівня залученості учасників.

Ключовим завданням переддипломної практики визначено проєктування сучасної вебплатформи на базі розподіленої мікросервісної архітектури, що дозволяє забезпечити стабільне функціонування кількох взаємопов'язаних стратегічних компонентів. Першочерговим інженерним напрямом у межах цієї архітектури є розробка та інтеграція інтелектуального AI-асистента стейкхолдера, що вимагає побудови складного пайплайну взаємодії з великими мовними моделями за допомогою інструментів Ollama,

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		8

LangChain та LangGraph. Цей інтелектуальний модуль відповідає за автоматизований аналіз і трансформацію неформатованого тексту або звичайних розмов користувачів у чітко структуровані картки інноваційних ідей.

Для забезпечення прозорості та контролю робочих процесів важливим кроком є створення надійного алгоритму на базі кінцевих автоматів (State Machine), який координує весь життєвий цикл ініціатив. Цей алгоритм дозволяє логічно переводити ідеї через послідовні статуси, починаючи від генерації та публікації, і закінчуючи збором команди, активною фазою розробки та фінальним завершенням проєкту. Зазначений процес безпосередньо пов'язаний із підсистемою менеджменту заявок, яка забезпечує обробку запитів від потенційних кандидатів із можливостями їх затвердження (approve) або відхилення (reject) стейкхолдерами.

Мотиваційну складову платформи та підтримку високого рівня залученості учасників гарантує розробка безпечної транзакційної системи, що відповідає за нарахування та списання рейтингових балів за виконання проєктів. Цей модуль також охоплює управління ризиками маніпуляцій, впровадження наочної системи унікальних досягнень (ачівок) та автоматизоване формування Глобального рейтингу користувачів. Усі ці функціональні елементи потребують визначення чітких архітектурних меж між автономними модулями із використанням технологічного стеку фреймворків Symfony, NestJS, React та Redux. Надійна робота та висока швидкодія системи досягається завдяки налаштуванню асинхронної обробки фонових задач за допомогою PHP workers і Mercure, а захист інформаційного контуру та API-ендпоінтів забезпечується впровадженням суворої системи авторизації на базі JWT-токенів.

Для забезпечення прозорості робочих процесів необхідно реалізувати модуль управління життєвим циклом проєктів. Це передбачає створення надійного алгоритму на базі скінченних автоматів (State Machine) для

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		9

логічного переведення ідей через послідовні статуси: генерація, публікація, збір команди, активна фаза та завершення. Цей процес нерозривно пов'язаний із системою обробки заявок від потенційних кандидатів із можливостями їх затвердження чи відхилення.

Окремим комплексним завданням є реалізація внутрішньої економіки балів та гейміфікації платформи. Розробка безпечної транзакційної системи нарахування та списання рейтингових балів за виконання проєктів, управління ризиками маніпуляцій, а також впровадження системи досягнень та формування Глобального рейтингу учасників слугуватимуть потужним інструментом мотивації.

Усі ці функціональні елементи потребують гарантованої архітектурної масштабованості та безпеки. Відповідно, завдання охоплює визначення чітких меж відповідальності між модулями із використанням технологій Symfony, NestJS, React та Redux, налаштування асинхронної обробки фонових задач за допомогою PHP workers і Mercure, а також впровадження надійної системи авторизації на базі JWT-токенів із суворим захистом API.

Важливим вектором розробки є створення комплексної підсистеми управління користувачами та забезпечення суворого контролю доступу. Для гарантування безпеки програмного інтерфейсу впроваджується надійна система авторизації та реєстрації на базі JWT-токенів. Цей механізм нерозривно пов'язаний із гнучкою рольовою моделлю, яка чітко розмежовує права між рядовими користувачами, менторами, лідерами команд, стейкхолдерами, інвесторами та адміністраторами. Кожен учасник платформи наділяється інструментами для створення та управління власним профілем, де деталізуються професійні навички, стек опанованих технологій та пріоритетні категорії проєктів. Водночас реалізовано блок налаштувань приватності, який дозволяє самостійно керувати видимістю контактних даних та відображенням персонального рейтингу. Для забезпечення ефективного адміністрування та технічної підтримки спроектовано спеціальний інструмент імперсонації, за

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		10

допомогою якого адміністратор може авторизуватися від імені будь-якого користувача для оперативної діагностики проблем.

Центральним елементом бізнес-логіки системи виступає модуль генерації та управління інноваційними ініціативами. Стейкхолдери отримують функціонал для публікації вичерпних карток ідей, у яких фіксуються назва, детальний опис, строгі часові рамки (дедлайни), необхідна кількість учасників та релевантні теги. З метою оптимізації цього етапу впроваджується інтелектуальний AI-асистент, здатний автоматично формувати структуровані проєктні пропозиції на основі неформатованого тексту або звичайного діалогу з ініціатором. Після успішної валідації та остаточного затвердження складу виконавців система забезпечує алгоритмічну трансформацію ініціативи, переводячи ідею у статус активного проєкту для початку безпосередньої спільної роботи екіпажу.

Для ефективної комунікації між авторами ідей та потенційними виконавцями розробляється глобальна пошукова система та структурований каталог ініціатив. Цей функціонал дозволяє користувачам переглядати всі публічно доступні проєкти, застосовуючи багатокритеріальні фільтри та механізми сортування. Знайшовши релевантну пропозицію, фахівці здійснюють менеджмент заявок: подають запити на участь, відстежують їхній поточний статус та мають змогу відкликати власні звернення за потреби. Зі свого боку, стейкхолдери використовують панель обробки кандидатів, де реалізовано процеси схвалення або відхилення отриманих заявок із можливістю обов'язкового зазначення аргументованої причини у разі відмови, що гарантує прозорість процесу формування команд.

Окремим стратегічним напрямом є проєктування внутрішньої економіки поінтів та багаторівневої системи гейміфікації для підтримки високої залученості учасників. Основою цього модуля є алгоритм обчислення та нарахування рейтингових балів за завершення проєктів, який спирається на об'єктивну оцінку внеску кожного виконавця з боку стейкхолдера.

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Додатковим стимулом виступає розроблена система досягнень, яка автоматично генерує та відображає нагороди за специфічну активність та здобутки в процесі роботи. Агрегація цих показників дозволяє формувати відсортовані списки користувачів у межах Глобального рейтингу платформи. Водночас для захисту екосистеми від можливих зловживань впроваджується модуль управління ризиками, який автоматизує процес пропорційного віднімання рейтингових балів у разі дострокового або неуспішного завершення проєкту, враховуючи порушення встановлених дедлайнів та загальну кількість залучених до команди спеціалістів.

1.2. Аналіз аналогів програмного продукту

В умовах глобалізації ринку праці та переходу процесів створення інноваційних проєктів у цифрове середовище, ефективність платформ для краудсорсингу визначається спроможністю системи миттєво реагувати на запити стейкхолдерів та забезпечувати швидкий і точний пошук релевантних спеціалістів для формування команд. Проєктування власного програмного забезпечення в межах дипломної роботи передбачає ретельний аналіз ключових гравців ринку віртуальних інкубаторів та платформ спільної розробки. Це необхідно для ідентифікації актуальних технологічних трендів, а також виявлення типових системних помилок, яких варто уникнути при розробці архітектури майбутнього бізнесу.

У межах порівняльного аналізу було відібрано три типові програмні платформи, кожна з яких демонструє специфічну архітектурну модель та унікальну стратегію побудови процесу генерації ідей та користувацької взаємодії.

Першою розглянутою системою є Devpost – глобальна платформа, орієнтована на проведення технічних змагань (хакатонів) та агрегацію інноваційних програмних рішень. Серед сильних сторін цієї платформи виділяється ефективний механізм тімбілдингу, який надає можливість гнучкого пошуку учасників бази даних за конкретними технічними

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		12

навичками, стеком технологій та зацікавленістю в певних категоріях проєктів. Крім того, система пропонує чітко структурований життєвий цикл ініціативи, що охоплює всі етапи: від реєстрації ідеї до формування команди, публікації прототипу та його оцінювання експертами. Важливим мотиваційним інструментом є розвинена система портфоліо та досягнень, де у деталізованому профілі користувача зберігається історія участі, здобуті нагороди (ачівки) та галерея реалізованих проєктів.

Поряд із цим, аналіз виявив низку суттєвих слабких сторін платформи Devpost. Насамперед це стосується відсутності AI-інструментів для генерації ідей: опис проєктів формується виключно вручну, оскільки платформа не має інтелектуальних помічників для швидкого структурування неформатованого тексту або допомоги стейкхолдерам. Ще одним недоліком є фрагментована економіка балів, яка проявляється у відсутності єдиної безперервної системи рейтингування, що стимулювала б постійну активність користувачів поза межами конкретних змагань. Зрештою, система характеризується обмеженою трансформацією ініціативи. Після формування команди та завершення первинного етапу генерації ідей, користувачі змушені переходити у сторонні сервіси управління (такі як Jira чи Trello) для повноцінного ведення проєкту та менеджменту ризиків, оскільки внутрішній інструментарій Devpost не підтримує глибокого управління проєктами.

		<i>A.B. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>A.B. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		13

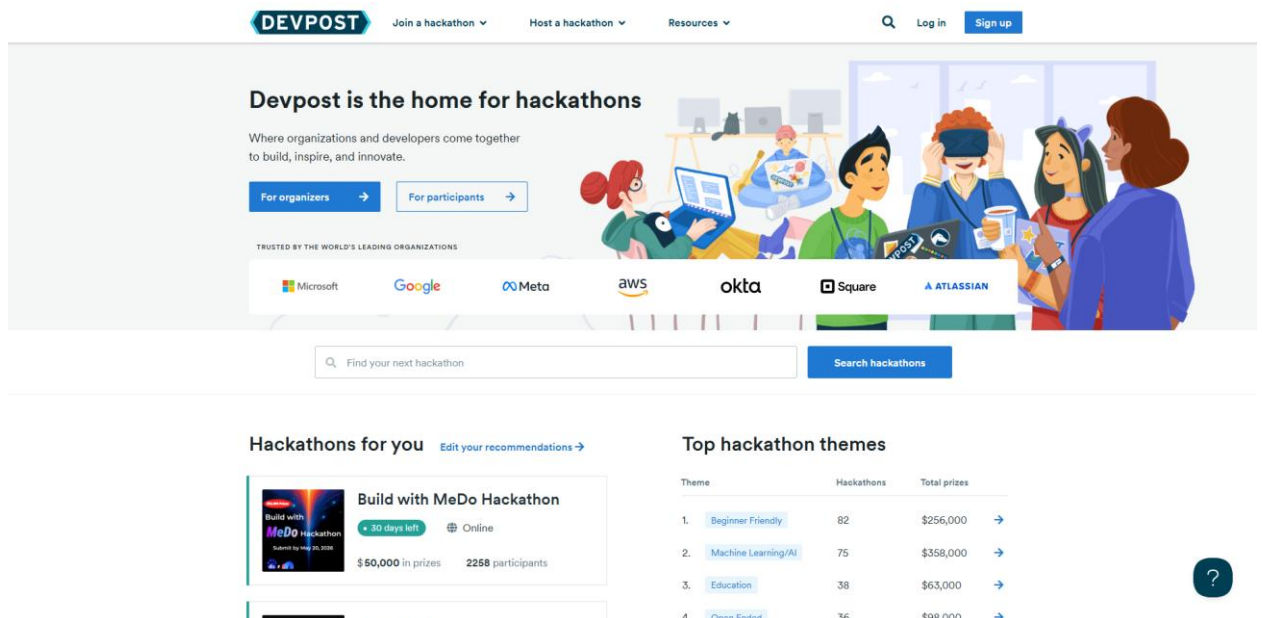


Рис.1.1. – Інтерфейс сайту **Devpost**.

Наступною розглянутою системою є IdeaScale, яка функціонує як одна з провідних корпоративних платформ для управління інноваціями та краудсорсингу. Цей сервіс надає організаціям комплексний інструментарій для збору ініціатив від співробітників чи зовнішньої аудиторії, проведення їх глибокої експертної оцінки та подальшої трансформації найкращих концепцій у реальні проєкти. Важливою перевагою платформи є гнучке управління життєвим циклом ініціативи завдяки наявності кастомізованого конструктора етапів. Такий підхід дозволяє детально налаштовувати логічну воронку проходження ідеї від початкової чернетки до стадії практичного впровадження із застосуванням різноманітних критеріїв затвердження або відхилення експертами. Крім того, система пропонує потужні інструменти валідації, серед яких виділяються вбудовані алгоритми для проведення цільових кампаній з генерації ідей, механізми крауд-голосування та багатовимірної експертної оцінки. Ці засоби ефективно допомагають відфільтрувати найбільш життєздатні концепції з великих масивів неструктурованих даних. Доповнюється цей функціонал розвиненими аналітичними дашбордами для

стейкхолдерів, які дають змогу в режимі реального часу відстежувати окупність інвестицій та загальний рівень залученості аудиторії.

Незважаючи на потужний аналітичний інструментарій, платформа має низку суттєвих обмежень, особливо в контексті спільної розробки. Головним недоліком є надзвичайно слабка орієнтація на тімбілдінг, оскільки архітектура сервісу сфокусована виключно на еволюції самої ідеї. Система принципово не надає інструментів для автоматизованого підбору виконавців за технічними навичками чи формування повноцінної робочої команди для реалізації затвердженої ініціативи. Ще однією відчутною проблемою є ізольована рейтингова система, що проявляється у відсутності наскрізної гейміфікації та єдиного Глобального рейтингу. Користувачі мають можливість заробляти мотиваційні бали лише в межах однієї ізольованої корпоративної кампанії, що критично знижує довготривалу зацікавленість незалежних учасників продовжувати роботу на платформі. Зрештою, архітектура системи надмірно перевантажена специфічним функціоналом для бізнесу, орієнтованим переважно на великий корпоративний сектор. Це робить користувацький інтерфейс об'єктивно занадто складним та створює невиправдано високий поріг входу для швидкої адаптації незалежними стартаперами чи студентськими спільнотами.

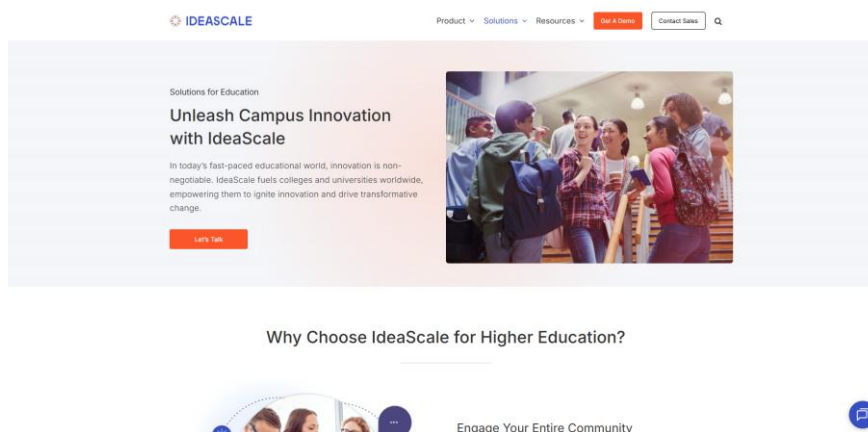


Рис.1.2. – Інтерфейс сайту IdeaScale

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Третьою проаналізованою системою є Brightidea, яка являє собою комплексну екосистему для управління інноваціями. Ця платформа орієнтована насамперед на великі корпорації та організації, що прагнуть перетворити процес генерування ідей на постійний і прогнозований потік бізнес-результатів. Сервіс пропонує набір спеціалізованих модульних додатків, адаптованих для різноманітних сценаріїв використання: від внутрішніх корпоративних скриньок пропозицій до організації глобальних хакатонів та відкритих інноваційних викликів.

Серед ключових переваг цієї екосистеми виділяється яскраво виражений модульний підхід, який забезпечує користувачів готовими структурними шаблонами для супроводу різних етапів життєвого циклу ідеї, таких як проведення змагань, лабораторних досліджень чи пітчінгів. Це дає змогу швидко адаптувати функціонал системи під специфічні потреби конкретної організації без необхідності залучення розробників для написання коду. Додатковою сильною стороною є наявність глибокої аналітики та інструментів відстеження впливу, які дозволяють об'єктивно оцінювати фінансову ефективність впроваджених ініціатив, прогнозувати майбутні витрати та наочно візуалізувати прогрес реалізації кожного проєкту за допомогою канбан-дошок. Забезпечення безперервного доступу до цих функцій реалізується через розвинену мобільну інфраструктуру, що дозволяє стейкхолдерам та іншим учасникам процесу оперативно взаємодіяти з ідеями, залишати експертні відгуки та брати участь у голосуваннях у режимі реального часу з будь-яких пристроїв.

Попри значні функціональні можливості, система Brightidea характеризується і певними недоліками, провідним з яких є надмірна складність конфігурації. Через наявність величезної кількості гнучких налаштувань та адміністративних функцій, процес первинного впровадження платформи часто виявляється затяжним і вимагає проведення спеціалізованого навчання для персоналу.

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		16



Рис.1.3. – Інтерфейс сайту Brightidea

Крім того, жорстка орієнтація продукту виключно на корпоративний сегмент зумовлює високу вартість ліцензування та архітектурну надмірність, що робить використання цієї платформи економічно непрактичним і майже недоступним для невеликих стартап-спільнот чи освітніх ініціатив. Критичним функціональним обмеженням також є відсутність глибокої інтеграції з технологіями штучного інтелекту для структурування даних. Незважаючи на загальну потужність аналітичного ядра, система не пропонує користувачам інтерактивних AI-асистентів для динамічного формування карток ідей з неформатованих розмов або для автоматизованого підбору команд на основі аналізу технічної та психологічної сумісності учасників.

Результати проведеного порівняльного дослідження свідчать про дефіцит на ринку програмних продуктів, які б гармонійно поєднували інтелектуальну автоматизацію структурування ідей з наскрізними механізмами гейміфікації та цільового тимбілдингу. Це обґрунтовує доцільність проектування вебсистеми управління життєвим циклом інноваційних ідей на базі розподіленої мікросервісної архітектури з інтеграцією інструментів штучного інтелекту. Таке комплексне рішення дозволить подолати фрагментованість існуючих віртуальних інкубаторів,

		A.B. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		A.B. МОРОЗОВ				17
Змн.	Арк.	№ докум.	Підпис	Дата		

забезпечити швидке формування команд на основі компетенцій та уникнути функціональної перевантаженості, притаманної великим корпоративним аналогам.

Додатковою характерною рисою архітектури Brightidea є можливість розгортання гібридних хмарних середовищ, що дозволяє великим організаціям гнучко керувати локаціями збереження конфіденційних даних та налаштовувати корпоративні протоколи безпеки. Платформа підтримує розширену інтеграцію із зовнішніми корпоративними системами авторизації та каталогами користувачів, спрощуючи централізований менеджмент облікових записів на рівні великих організаційних структур. Окрім цього, система володіє вбудованим модулем патентного аналізу, який допомагає компаніям автоматично перевіряти генеровані ідеї на предмет унікальності та відповідності чинному законодавству у сфері інтелектуальної власності. Розробники платформи також приділяють значну увагу інструментам внутрішнього фінансування, надаючи стейкхолдерам можливість розподіляти віртуальні бюджети між перспективними командами безпосередньо всередині робочого простору екосистеми.

У наведеній таблиці представлено порівняльний аналіз популярних існуючих платформ управління інноваціями, таких як Devpost, IdeaScale та Brightidea, за їхніми ключовими функціональними характеристиками. Зазначене порівняння розкриває специфіку їхнього основного фокусу, підходів до управління каталогами проєктів, а також наявність інструментів формування команд і механізмів гейміфікації. Результати цього аналізу дозволяють об'єктивно оцінити можливості та обмеження наявних на ринку рішень для подальшого обґрунтування необхідності розробки унікального функціоналу власної вебсистеми.

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Порівняння аналогів

Характеристика	Devpost	IdeaScale	Brightidea
Основний фокус	Організація хакатонів та публічних змагань розробників	Корпоративний краудсорсинг та багатоетапна валідація ідей	Екосистемне управління інноваціями в Enterprise-секторі
Управління каталогом	Структурована подача проєктів із чіткими дедлайнами для журі	Гнучкий конструктор воронки (стадій) проходження ідеї	Модульні шаблони (Apps) для різних типів інноваційних викликів
Інструменти підбору	Гнучкий пошук учасників за технічними навичками та стеком	Обмежені (акцент на колективному обговоренні, а не на виконавцях)	Відсутні цільові інструменти тимбілдингу (пріоритет на ідеї)
Система мотивації та гейміфікація	Система досягнень (ачівки) та публічне портфоліо	Локальні рейтинги в межах однієї кампанії, крауд-голосування	Відстеження фінансового впливу (ROI), без глобального рейтингу
Система лояльності	Глобальне портфоліо та "ачівки"	Відсутня наскрізна система (лише локальна)	Відсутня (оцінюється лише ROI ідеї)
Профіль покупця	Розширений (історія хакатонів, стек, навички)	Базовий (орієнтація на обговорення)	Базовий корпоративний профіль
Персоналізація	Обмежена (рекомендації хакатонів за тегами)	Відсутня	Мінімальна (за корпоративними інтересами)
Швидкодія інтерфейсу	Висока (оптимізовано для великих навантажень)	Середня (через складні дашборди)	Середня (залежить від конфігурації)

Результати порівняльного аналізу демонструють, що наявні на ринку рішення не повною мірою задовольняють специфічні запити цільової

аудиторії та динамічні вимоги сучасного стартап-середовища і краудсорсингу. Велика частина конкурентних платформ або фокусується виключно на процесах експертної валідації ідей без надання дієвих інструментів для цільового формування команд на основі компетенцій, або має громіздку корпоративну структуру, яка перешкоджає швидкому впровадженню інтелектуальних систем структурування контенту (AI-асистентів) та наскрізних механізмів гейміфікації для довгострокової мотивації учасників.

1.3. Вибір архітектури системи

На етапі проектування вебсистеми управління життєвим циклом інноваційних ідей та формування команд критично важливим є вибір фундаментальної структури системи. Архітектурний підхід безпосередньо впливає на те, наскільки ефективно сервіс оброблятиме пікові навантаження під час масових публікацій проєктів (наприклад, у період проведення хакатонів чи дедлайнів), як швидко відбуватиметься обробка неструктурованого тексту інтегрованими AI-моделями та наскільки стабільно працюватимуть модулі комунікації і транзакцій рейтингової системи. Спираючись на досвід провідних гравців ринку краудсорсингу та враховуючи необхідність поступового масштабування бізнес-логіки, було проведено порівняльний аналіз двох найбільш розповсюджених методологій: класичного моноліту та мікросервісної побудови [2]. Традиційна монолітна архітектура розглядає платформу для спільної роботи як нерозривний програмний блок. У такій моделі візуальний каталог ідей, функціонал управління заявками, система підрахунку економіки балів та модуль взаємодії зі штучним інтелектом існують у межах єдиної кодової бази та спільного процесу.

На противагу традиційним підходам, мікросервісна (або розподілена) архітектура розглядає платформу як конгломерат автономних програмних одиниць, що взаємодіють через мережу. У такій моделі функціонал розбивається на окремі сервіси: наприклад, основний сервіс бізнес-логіки та профілів, ізольований модуль чатів та сповіщень (на базі NestJS/Mercure),

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		20

сервіс взаємодії з великими мовними моделями (Ollama/LangChain) та окрема система управління глобальним рейтингом. Головною перевагою є практично необмежена горизонтальна масштабованість. Це дозволяє нарощувати потужність лише для тих частин системи, які відчувають найбільший тиск — наприклад, виділяти додаткові обчислювальні ресурси для AI-модуля під час активної генерації ідей або масштабувати сервіс обробки сокетів під час формування команд, не зачіпаючи при цьому логіку адміністрування. Окрім того, кожен сервіс може розроблятися на найбільш підходящому для нього технологічному стеку, поєднуючи PHP для бізнес-логіки, Node.js для чатів та спеціалізовані інструменти для ШІ. Також є декілька недоліків, наприклад перехід на повністю ізольовані мікросервіси для проєкту середнього масштабу створює значне інфраструктурне навантаження. Процеси оркестрації (Docker), гарантування узгодженості даних між різними базами (наприклад, щоб інформація про прийняття кандидата в команду миттєво синхронізувалася з модулем чатів та перераховувала рейтинг) та моніторинг мережевих затримок вимагають значних ресурсів. Для реалізації дипломного проєкту використання "чистих" мікросервісів часто є надмірним, оскільки витрати на підтримку інфраструктури можуть перевищити вигоду.

Саме тому оптимальним рішенням виступає гібридний підхід — модульний моноліт із виділеними міросервісами як ядро системи з винесенням найбільш навантажених чи специфічних завдань (AI-генерація, real-time чати, фонові обробка) в окремі контейнеризовані сервіси та воркери. Аналізуючи специфіку проєкту, для реалізації вебсистеми управління життєвим циклом ідей було обрано саме таку гібридну архітектуру: “Модульний моноліт із виділеними мікросервісами” (рис. 1.4). Цей підхід став оптимальним компромісом, оскільки він дозволяє уникнути хаотичності класичного моноліту та надмірної інфраструктурної складності чистих мікросервісів.

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		21

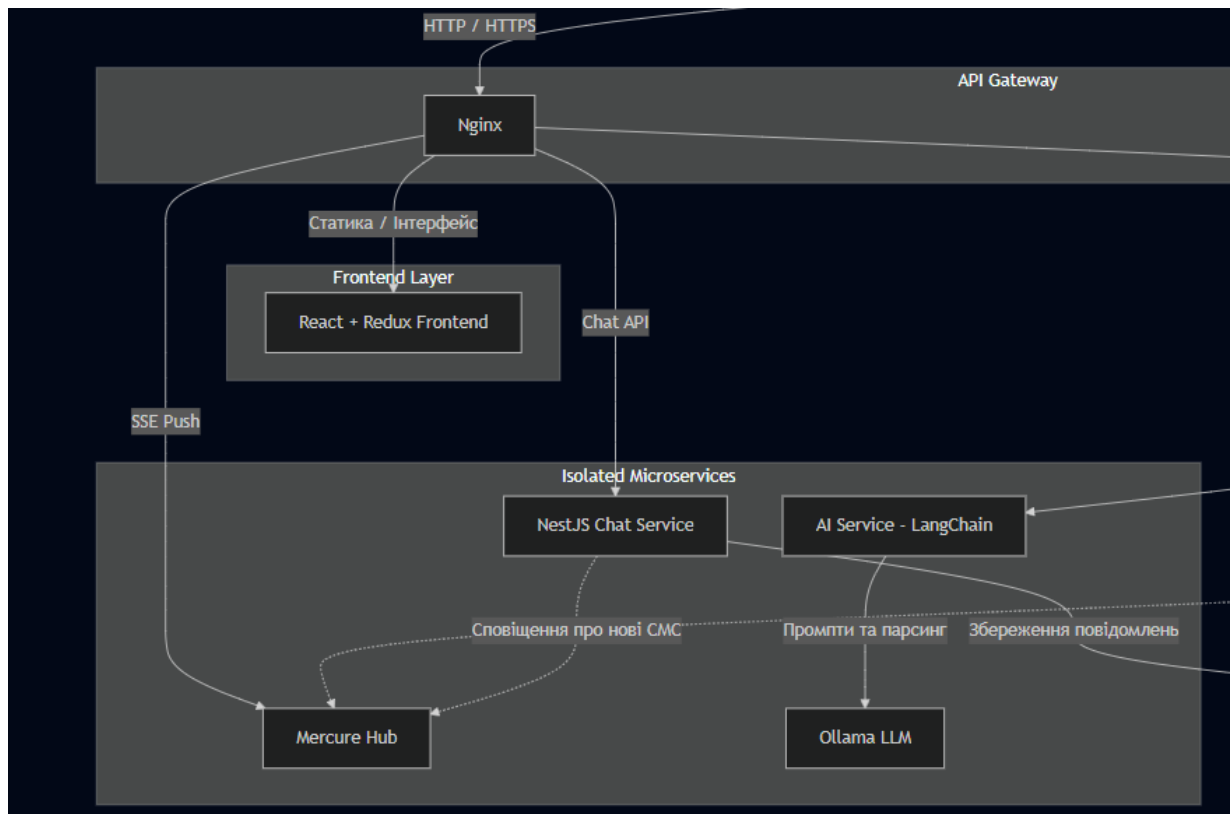


Рис. 1.4. – Вигляд модульно-монолітної архітектури з виділеними мікросервісами

В межах такої архітектури основна бізнес-логіка розробляється як єдиний програмний комплекс, проте з жорстким розмежуванням на автономні модулі (наприклад, модуль «Каталог ідей», модуль «Рейтингова система та економіка балів», модуль «Менеджмент заявок»). Кожен блок має власну зону відповідальності та взаємодіє з іншими через чітко визначені інтерфейси [32], що мінімізує їхню взаємозалежність, тоді як найбільш ресурсоємні завдання (AI-асистент, real-time чати) винесено в ізольовані контейнери.

Побудова сучасної краудсорсингової платформи ґрунтується на клієнт-серверній моделі [7], яка є перевіреним стандартом для створення масштабованих SaaS-рішень. Функціонально система розділена на дві ключові зони відповідальності. Клієнтська сторона працює на стороні користувача у веббраузері, забезпечуючи інтерактивну взаємодію з візуальним каталогом проєктів, динамічне відображення Глобального рейтингу, зручну подачу заявок на участь в командах та миттєве отримання push-сповіщень про зміну

статусів. Серверна сторона розміщується на серверних потужностях і відповідає за «мозок» системи: валідацію переходів життєвого циклу ідеї, оркестрацію AI-моделей для структурування тексту, безпеку авторизації та рольової моделі, безперервний підрахунок транзакцій рейтингу та надійну взаємодію з базою даних MySQL.

Використання гібридної архітектури в межах цієї моделі дозволяє досягти високої стабільності бізнес-процесів. Завдяки чіткій ізоляції функціональних вузлів та асинхронній обробці задач, система зберігає цілісність і високу швидкість відгуку навіть при масових зверненнях до AI-генератора або під час пікової активності користувачів у періоди дедлайнів формування команд.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Формування технологічного стеку для проєктуваної вебсистеми управління життєвим циклом інноваційних ідей ґрунтується на гострій необхідності гармонійного поєднання надійної бізнес-логіки обробки складних транзакцій, високої реактивності користувацького інтерфейсу під час активної командної взаємодії, а також безперешкодної можливості інтеграції сучасних, вимогливих до обчислювальних ресурсів інструментів штучного інтелекту.

У якості фундаментального інструмента для розробки бекенду та формування стійкого ядра системи за моделлю модульного моноліту було обрано екосистему PHP у поєднанні з потужним фреймворком Symfony [25]. Такий вибір безпосередньо обґрунтовується винятковою гнучкістю його архітектури, наявністю розгалуженої системи перевірених часом компонентів та вбудованим інструментарієм, що ідеально підходить для побудови складних і масштабованих SaaS-рішень. Важливою архітектурною перевагою використання Symfony є можливість налаштування пулу PHP workers, що дозволяє системі максимально ефективно та асинхронно обробляти

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		23

ресурсомісткі фонові задачі. До таких процесів належать регулярний перерахунок Глобального рейтингу учасників платформи та оперативна відправка транзакційних листів через протокол Gmail SMTP, що відбувається абсолютно непомітно для користувача та повністю виключає блокування основного потоку виконання програми. Поряд із цим, для забезпечення безперебійної та швидкої комунікації всередині робочих екіпажів, інфраструктура передбачає розгортання ізольованого мікросервісу на базі NestJS, який бере на себе виключну відповідальність за обробку командних чатів. Завдяки глибокій асинхронній подієво-орієнтованій природі середовища Node.js, цей сучасний фреймворк виступає найкращим технологічним рішенням для обслуговування високонавантажених модулів комунікації в режимі реального часу [20].

Для побудови високоінтерактивного та ергономічного клієнтського інтерфейсу (Frontend) застосовується популярна компонентна бібліотека React [26]. Її використання в тандемі з інструментом Redux дозволяє повноцінно та якісно реалізувати архітектурну концепцію Single Page Application (SPA), чим гарантується надзвичайно висока швидкість відгуку браузера при навігаційних переходах між глобальним каталогом інноваційних ідей та сторінками особистого профілю учасника. Впровадження Redux бере на себе критично важливу функцію детермінованого глобального управління станами додатку, що є абсолютно необхідним для безпомилкової та швидкої синхронізації динамічних даних користувача (таких як баланс рейтингових балів чи оновлені статуси поданих заявок) одночасно у багатьох незалежних компонентах екрану. За загальну реактивність комунікації між клієнтським додатком та сервером відповідає інтегрований брокер повідомлень Mercure Hub, що працює на базі стандарту Server-Sent Events (SSE). Застосування цієї технології забезпечує можливість миттєво доставляти push-сповіщення — наприклад, про успішне схвалення заявки стейкхолдером — без створення

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		24

надмірного навантаження на мережу, усуваючи застарілу потребу постійного опитування сервера клієнтом.

Фундаментом для надійного управління структурованою інформацією слугує реляційна система керування базами даних MySQL 8.0, яка на програмному рівні гарантує повну стабільність та відповідність суворим вимогам ACID-транзакцій. Ця характеристика є безальтернативною умовою для коректного та безпечного функціонування внутрішньої економіки балів: навіть при одночасному паралельному нарахуванні або списанні рейтингу великій кількості користувачів, дані завжди залишатимуться цілісними та захищеними від аномалій конкурентного доступу. Водночас для оптимізації роботи бази даних та зняття з неї невластивого навантаження, збереження значних обсягів неструктурованого медіаконтенту (зокрема, об'ємних додатків до проєктів, технічної документації та аватарів профілів) делеговано спеціалізованому об'єктному сховищу AWS S3, що гарантує високу доступність, безпеку та географічну масштабованість статичних файлів платформи.

Інтеграція штучного інтелекту реалізована через зв'язку Ollama/Gemini та LangChain/LangGraph. Цей спеціалізований AI-мікросервіс відповідає за структурування неформатованих ідей користувачів у чіткі проєктні вимоги (json-формат), використовуючи гібридний підхід до інтеграції ШІ (хмарний Gemini API для продуктового середовища та локальні моделі Ollama для ізольованих On-Premise розгортань). Використання локального середовища Ollama (з моделями типу Llama-3 або Mistral) дозволяє розгортати систему в повністю закритих корпоративних контурах без ризику витоку конфіденційних даних у зовнішні хмари, забезпечуючи нульову вартість інференсу при наявності відповідного апаратного забезпечення. Водночас для публічного релізу платформи (Production) інтегровано API нейромережі Gemini від Google [22]. Це забезпечує кращу масштабованість системи, високу швидкість генерації відповідей за рахунок потужностей хмарного провайдера

та доступ до ширшого контекстного вікна без необхідності утримання дорогої локальної інфраструктури з потужними графічними прискорювачами (GPU).

Завдяки такій двовекторній архітектурі, абстрагований за допомогою LangChain інтерфейс дозволяє гнучко перемикатися між локальним та хмарним інференсом через конфігураційні змінні оточення (ENV), не змінюючи при цьому базовий код та логіку графів управління станами системи. Безпека користувачів, окрім стандартної авторизації через Google OAuth, базується на використанні JWT-токенів... (далі ваш текст про JWT та RBAC без змін).

Безпека користувачів, окрім стандартної авторизації через Google OAuth, базується на використанні JWT-токенів для захисту API-ендпоінтів та розмежуванні прав доступу на основі Role-Based Access Control (RBAC).

Технічні вимоги для експлуатації платформи залежать від обраної стратегії розгортання ШІ-модуля. Для базового хмарного використання (з Gemini API) вимоги до обладнання є мінімальними. Однак, для розгортання системи в режимі підвищеної конфіденційності (On-Premise), де передбачено контейнеризацію Docker та запуск локальних LLM (Ollama).

Для успішного розгортання та експлуатації платформи вимоги до серверної інфраструктури охоплюють комплекс апаратних та програмних рішень. Обчислювальний базис системи має спиратися на 8-ядерний центральний процесор, хоча для забезпечення ефективної паралельної обробки фонових задач та стабільної роботи воркерів рекомендується використовувати CPU з 12 ядрами.

Особливу роль відіграє графічний прискорювач флагманського рівня з максимальним обсягом відеопам'яті, необхідний для апаратного прискорення інференсу великих мовних моделей у середовищі Ollama. У промислових масштабах доцільно застосовувати високопродуктивні серверні рішення рівня NVIDIA H100 або A100 із 80 гігабайтами VRAM, тоді як для локального тестування чи менш навантажених контурів підійдуть топові споживчі

відеокарти на зразок NVIDIA RTX 4090 із 24 гігабайтами пам'яті, що гарантує миттєве опрацювання множинних AI-запитів без затримок.

Мінімально необхідний обсяг оперативної пам'яті становить 16 гігабайт, проте для безперебійного функціонування мовних моделей та швидкої підтримки In-Memory структур реляційної бази даних оптимальним вибором буде виділення 64 гігабайт RAM. Дискowa підсистема має бути представлена швидкісним твердотільним накопичувачем ємністю від 1 терабайта, оскільки значна частина цього простору використовуватиметься під збереження великих Docker-образів та ваг мовних моделей.

Надійне підключення до мережі вимагає пропускної здатності від 1000 мегабіт на секунду, а також обов'язкового впровадження протоколів шифрування SSL і TLS для повного захисту WebSocket та Server-Sent Events з'єднань. У якості програмного середовища для запуску платформи використовується Linux-сумісна операційна система, переважно дистрибутиви Ubuntu або Debian, із попередньо встановленим інструментарієм контейнеризації Docker Engine та Docker Compose.

Комплексна комбінація вищезазначених технологій формує не просто базове робоче середовище, а створює надзвичайно потужний та адаптивний інфраструктурний фундамент для довгострокового масштабування проєкту. Такий підхід дозволяє гармонійно поєднати виконання суворої, транзакційно безпечної бізнес-логіки бекенду з передовими інструментами автоматизації та інноваційними алгоритмами тімбілдингу. Завдяки чіткому розмежуванню зон відповідальності між модульним монолітом на базі Symfony, високопродуктивними комунікаційними мікросервісами NestJS розроблена вебсистема отримує високий рівень відмовостійкості і легко піддається подальшій функціональній модернізації без ризику порушення цілісності ядра.

Крім того, обрана архітектурна парадигма формує оптимальні умови для безшовної інтеграції складних та ресурсомістких рішень. Завдяки використанню асинхронних черг та об'єктних сховищ, інтенсивна робота

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		27

пайплайнів обробки природної мови жодним чином не впливає на стабільність критичних бізнес-процесів платформи, зокрема на управління життєвим циклом проєктів. Зрештою, ретельно спроектований технологічний стек виступає надійним гарантом збереження високої продуктивності системи навіть за умов стрімкого збільшення кількості активних користувачів та експоненційного зростання обсягів оброблюваної інформації, надаючи учасникам максимально стабільний цифровий простір.

Висновки до першого розділу

У межах першого розділу було сформовано концептуальний фундамент для розробки сучасної вебсистеми управління життєвим циклом інноваційних ідей та формування команд. Визначено ключові функціональні блоки системи, спрямовані на автоматизацію процесів краудсорсингу, валідацію ідей стейкхолдерами та створення прозорого середовища для взаємодії розробників на основі інтегрованої системи гейміфікації (економіки балів).

Проведено детальне дослідження ринкових аналогів (платформ для віртуальних інкубаторів та проведення хакатонів), у ході якого проаналізовано їхні технологічні підходи до управління проєктними воронками, механізмами цільового тімбілдингу та мотивації учасників. На основі виявлених переваг і недоліків існуючих рішень було обґрунтовано необхідність розробки власної системи, яка усуває проблему фрагментованості процесу та забезпечує глибшу інтеграцію інструментів штучного інтелекту для автоматизованого структурування неформатованих ідей.

Виконано порівняльний аналіз архітектурних стратегій, за результатами якого для реалізації вебсистеми було обрано гібридний підхід — модульний моноліт із виділеними мікросервісами. Такий вибір дозволяє поєднати цілісність реляційних даних і надійність транзакцій, характерні для монолітів, із високим ступенем ізоляції найбільш навантажених процесів (AI-генерація, real-time чати), що створює необхідні умови для відмовостійкості та безперебійного масштабування проєкту.

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		28

На завершення розділу детально обґрунтовано вибір сучасного комплексного технологічного стеку на базі Symfony, React, NestJS, MySQL та інструментів ШІ (Ollama, LangChain), який забезпечує максимальну реактивність інтерфейсу та безпеку API. Водночас, зважаючи на високу вартість апаратного забезпечення для локального інференсу, для продуктової (Production) версії системи визначено доцільним впровадження альтернативного хмарного рішення на базі API від провідних провайдерів (наприклад, Gemini), що дозволить знизити інфраструктурні витрати при збереженні високої якості генерації.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБСИСТЕМА

2.1 Визначення архітектури інтеграції штучного інтелекту

Проектування сучасної системи управління життєвим циклом інноваційних ідей вимагає побудови гнучкої та масштабованої архітектури, здатної витримувати високі навантаження та забезпечувати безперебійну роботу інтелектуальних модулів. В основу розроблюваного програмного комплексу покладено модульний моноліт із виділеними мікросервісами, який передбачає суворе логічне та технологічне розділення основного бекенду і спеціалізованого модуля штучного інтелекту.

Таблиця 2.1

Технологічний стек системи та призначення компонентів

Найменування компонента	Призначення в системі
Symfony та NestJS	Реалізація основної бізнес-логіки, управління базою даних та транзакціями рейтингової системи.
Python	Забезпечення роботи ізольованого мікросервісу для обробки даних та взаємодії з мовними моделями.
Gemini API	Виконання функцій інтелектуального асистента для аналізу текстів та генерації контенту.
LangChain	Конструювання системних підказок та управління контекстом розмови з ШІ.
LangGraph	Моделювання складних багатоетапних процесів обробки даних у вигляді орієнтованих графів.

Основна бізнес-логіка, управління базою даних та транзакціями рейтингової системи реалізуються засобами фреймворків Symfony та NestJS [29], тоді як уся складна взаємодія з великими мовними моделями [3] винесена

в окремий ізольований мікросервіс [33], розроблений мовою програмування Python. Вибір даного технологічного стеку обґрунтовується наявністю в екосистемі Python найбільш розвиненої та актуальної інфраструктури для роботи з AI-агентами.

Варто зазначити, що архітектура Python-мікросервісу спроектована за принципом абстракції, що дозволяє легко перемикаати провайдерів мовних моделей. Хоча на етапі прототипування та тестування (як описано у Розділі 1) використовувалися локальні моделі Ollama для забезпечення незалежності від зовнішніх сервісів, у фінальному продуктовому середовищі (Production) реалізовано інтеграцію з хмарною неймережею Gemini. Таке рішення зумовлене необхідністю оптимізації інфраструктурних витрат (відсутність потреби в дорогих GPU) та вищою швидкістю обробки запитів. Доступ до моделі забезпечується через відповідний програмний інтерфейс (API Gemini). У межах Python-мікросервісу розгорнуто фреймворк LangChain, який виконує роль сполучної ланки між вхідними даними від користувача та мовною моделлю. Застосування LangChain дозволяє стандартизувати процес конструювання системних підказок (промптів), ефективно управляти контекстом тривалих розмов та реалізовувати механізми виклику функцій (Function Calling), що перетворює модель із простого генератора тексту на автономного агента [6] (рис. 2.1).

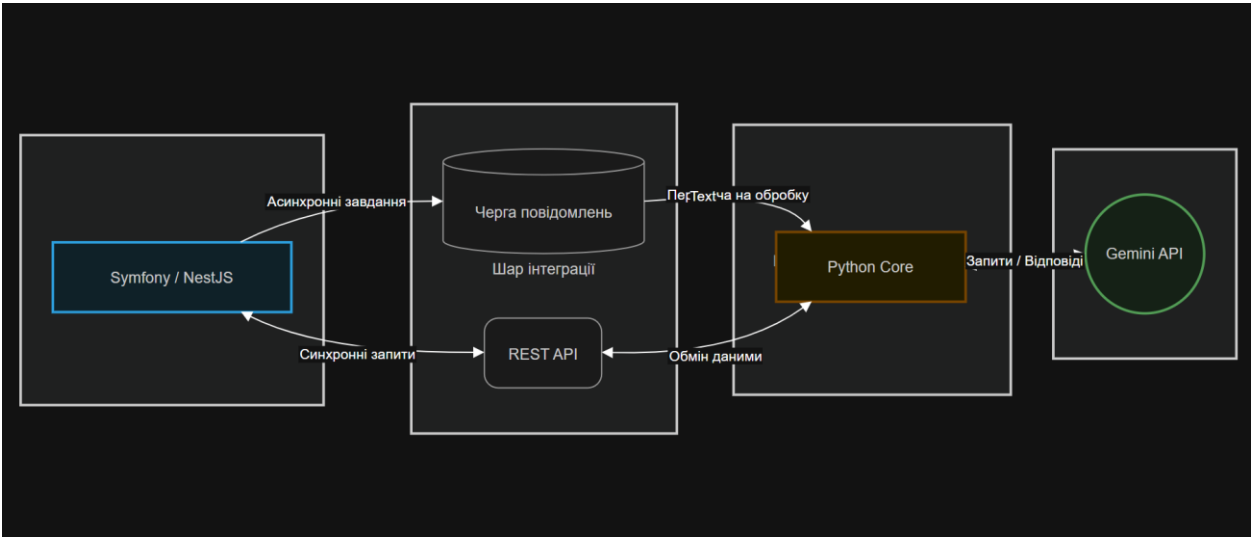


Рис. 2.1 – Схема взаємодії мікросервісів та модуля штучного інтелекту

Оскільки процес формування структурованої картки ідеї з неформатованого тексту є багатоетапним завданням, лінійної обробки запитів виявилось недостатньо. Для подолання цього архітектурного обмеження до Python-мікросервісу було інтегровано бібліотеку LangGraph, яка забезпечує моделювання складних процесів у вигляді орієнтованих циклічних графів [15]. Логіка роботи AI-асистента розбивається на окремі стани: первинний збір інформації, перевірка повноти даних, генерація опису та фінальне затвердження. Система здатна динамічно переміщуватися між цими вузлами графа, циклічно уточнюючи незрозумілі деталі у стейкхолдера, що максимально імітує роботу реального бізнес-аналітика. У результаті такого циклічного опрацювання генерується стандартизований JSON-об'єкт, який Python-мікросервіс повертає основному бекенду для збереження у базі даних.

Для забезпечення високої надійності роботи інтелектуальних алгоритмів до архітектури системи підключено хмарну платформу LangSmith. Цей інструмент інтегровано безпосередньо у код Python-модуля для здійснення безперервного моніторингу, глибокого трасування (tracing) та аналізу кожного кроку виконання графа в LangGraph. Завдяки LangSmith розробник отримує можливість детально аналізувати витрати токенів під час звернень до API Gemini, відстежувати тривалість виконання окремих вузлів та швидко локалізувати логічні помилки у складних ланцюжках взаємодії. Це дозволяє безпечно тестувати нові версії системних підказок на основі історичних даних, гарантуючи високу якість формування карток проєктів у продуктовому середовищі платформи.

Окрім внутрішньої логіки обробки даних, важливою складовою архітектури є механізм комунікації між Python-мікросервісом та основним бекендом платформи. Зважаючи на те, що генерація відповідей великими мовними моделями потребує певного часу, взаємодія між компонентами реалізована на базі асинхронного підходу. Основний сервер ініціює завдання з формування картки ідеї та продовжує обробку інших користувацьких запитів

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		32

без блокування потоку виконання. Після успішного проходження всіх стадій графа в LangGraph, мікросервіс автоматично надсилає готовий структурований результат через систему зворотних викликів (webhooks) або брокер повідомлень, що гарантує цілісність даних та високу чуйність користувацького інтерфейсу.

Для забезпечення незалежного розгортання та ефективного використання апаратних ресурсів Python-мікросервіс функціонує у повністю ізольованому контейнеризованому середовищі на базі технології Docker. Це дозволяє гнучко та горизонтально масштабувати виключно інтелектуальний модуль системи у періоди пікових навантажень, не впливаючи на роботу основного веб-сервера. Додатково на рівні програмного коду імплементовано надійні механізми відмовостійкості, які включають автоматичну обробку мережових виключень та стратегію повторних запитів з експоненційною затримкою у разі тимчасового перевищення квот API Gemini. Завдяки такому підходу гарантується стабільна робота AI-асистента та зводиться до мінімуму ризик переривання процесу збору вимог через нестабільність з'єднання із зовнішніми хмарними сервісами.

2.2 Розробка інформаційного забезпечення та структури бази даних

Проектування інформаційного забезпечення веборієнтованої системи управління життєвим циклом інноваційних ідей та формування команд базується на створенні цілісної, нормалізованої та продуктивної реляційної моделі даних. Враховуючи розподілену архітектуру майбутнього програмного комплексу, де бізнес-логіка розподілена між фреймворками Symfony та NestJS, до структури збереження інформації висуваються підвищені вимоги щодо цілісності, швидкодії та уникнення колізій ідентифікаторів. Як система керування базами даних обрана MySQL версії 8.0 [21] з використанням системи збереження InnoDB, що дозволяє повноцінно реалізувати механізми транзакційності, підтримку зовнішніх ключів та високу швидкість обробки складних запитів. Усі текстові поля оптимізовані під кодування utf8mb4 та

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ІЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		33

відповідне зіставлення `utf8mb4_unicode_ci` для забезпечення коректного збереження мультязичного контенту та спеціальних символів. Спроектowana інформаційна база відзначається значним масштабом та високим рівнем деталізації бізнес-процесів предметної області, про що свідчить наявність 57 взаємопов'язаних таблиць. З огляду на такий великий обсяг даних і складність архітектури реляційних зв'язків, повну фізичну модель бази даних у вигляді діаграми винесено в додаток А.

Однією з ключових архітектурних особливостей розробленої бази даних є використання універсальних унікальних ідентифікаторів UUID у вигляді двійкового формату `binary(16)` для більшості стратегічних сутностей платформи. Такий підхід мінімізує ризики колізій під час генерації унікальних ключів на стороні різних мікросервісів або асинхронних PHP workers, а також приховує реальну кількість записів у системі від кінцевого користувача, підвищуючи рівень безпеки API. Для об'єктів, що вимагають строгої послідовності або де обсяг даних є критичним для оптимізації індексів, наприклад, для завдань або файлових метаданих, застосовано традиційні автоінкрементні цілі числа. Системна цілісність моделі додатково підтримується використанням концепції м'якого видалення за допомогою поля визначення часу вилучення запису, що дозволяє зберігати історію транзакцій та аналітичні дані навіть після формального видалення об'єктів користувачами.

На сервісі Symfony реалізовано API, доступ до якого здійснюється через зручний інтерфейс Swagger, приклад, якого зображено на (рис. 2.2)

		<i>A.B. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>A.B. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		34

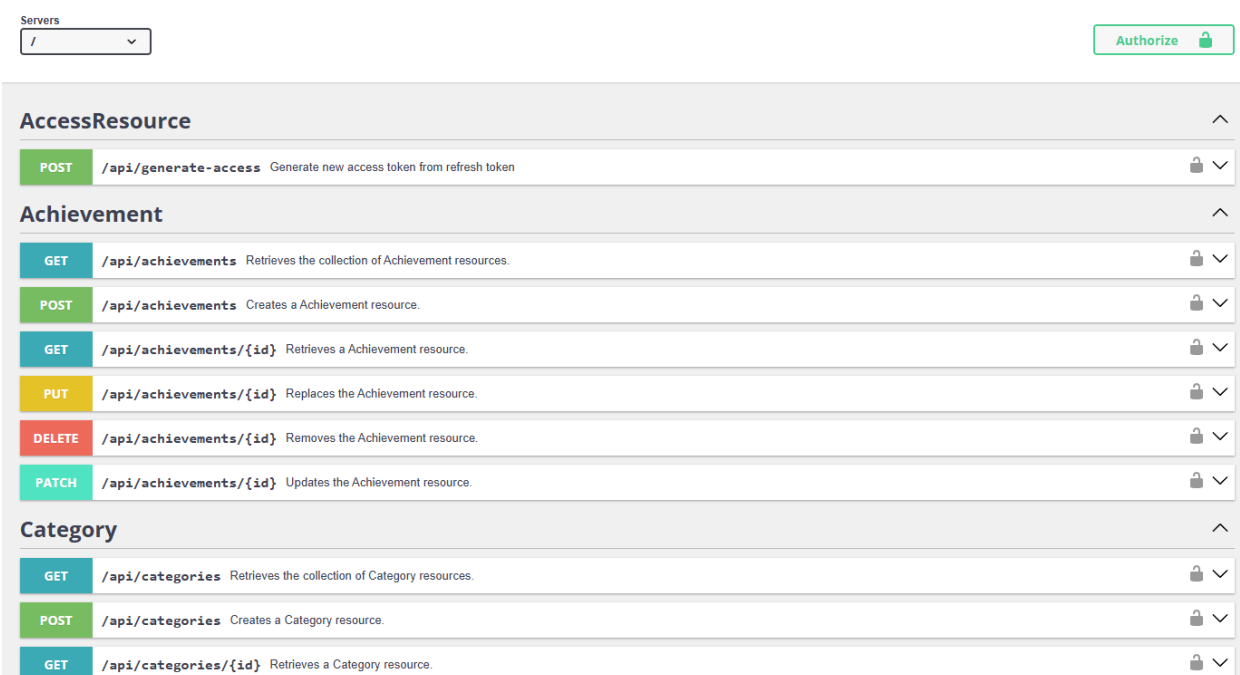


Рис. 2.2 – Доступ до API через Swagger

Центральне місце в інформаційній структурі займає сутність ідеї, представлена таблицею для збереження базових та конфігураційних параметрів інноваційних ініціатив. Вона фіксує унікальний ідентифікатор, назву проєкту, детальний текстовий опис та додаткові аналітичні блоки, такі як каталізатори та якірні прив'язки, що необхідні ІІІ агенту Gemini для подальшої класифікації. Для збереження гнучкості відображення текстового контенту передбачені спеціальні поля форматів опису, які визначають спосіб рендерингу даних на фронтенді. Логіка управління життєвим циклом ідеї інтегрована безпосередньо у структуру через систему зовнішніх ключів, що пов'язують кожну ініціативу зі статусами, рівнями складності та дефолтними параметрами для майбутніх завдань. Часові межі активності ідеї контролюються полями дат початку та завершення кампанії, а системні поля створення та оновлення забезпечують можливість повного аудиту дій стейкхолдерів.

Процес декомпозиції ініціатив на конкретні кроки реалізації забезпечується сутністю завдань, яка має тісний реляційний зв'язок із

таблицею ідей. Таблиця завдань використовує автоінкрементний первинний ключ та містить зовнішні ключі для прив'язки до конкретної ідеї, статусу, пріоритету та типу. У структурі чітко розмежовані ролі користувачів: фіксується ідентифікатор автора завдання та ідентифікатор безпосереднього виконавця, що дозволяє будувати динамічні робочі процеси в межах сформованої команди. Для реалізації гнучких методологій розробки в кожному завданні передбачено поле для збереження оцінки складності у відносних одиницях, а також посилання на спринт. Текстові атрибути завдання включають назву та розгорнутий опис із підтримкою динамічного форматування. Зв'язки типу багато до багатьох із сутностями тегів та вкладених файлів винесені в окремі проміжні таблиці, що забезпечує гнучкість фільтрації та структурування завдань у межах інтерфейсу користувача.

Для організації гнучкого планування та контролю прогресу виконання проєктів у системі спроектовано сутність спринтів. Кожен запис у відповідній таблиці жорстко прив'язаний до батьківської ідеї та містить унікальний ідентифікатор, назву спринту та текстове поле для фіксації мети поточної ітерації. Часові рамки ітерації обмежуються полями дати початку та завершення спринту, а поточний стан контролюється текстовим полем статусу. Для забезпечення роботи аналітичного модуля, що будує діаграми згоряння задач, розроблено додаткову сутність знімків стану спринту, яка щоденно фіксує кількість залишкових та виконаних оціночних одиниць, а також загальну кількість закритих завдань, що дозволяє стейкхолдерам оцінювати продуктивність команди в реальному часі.

Мотиваційна складова платформи та ведення глобального рейтингу користувачів спираються на сутність досягнень, взаємодія з якою фіксується у таблиці користувацьких досягнень. Ця проміжна сутність виконує роль сполучної ланки між профілем розробника та глобальним каталогом нагород системи. Окрім зовнішніх ключів користувача та досягнення, таблиця містить атрибут поточного статусу, що дозволяє відстежувати як повністю

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ІЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		36

розблоковані нагороди, так і ті, що перебувають у процесі виконання умов. Кожен факт присвоєння досягнення супроводжується фіксацією точного часу та автоматично ініціює перерахунок рейтингових балів у таблиці зв'язку користувача та ідеї, де акумулюються бали за активну участь у конкретних проєктах.

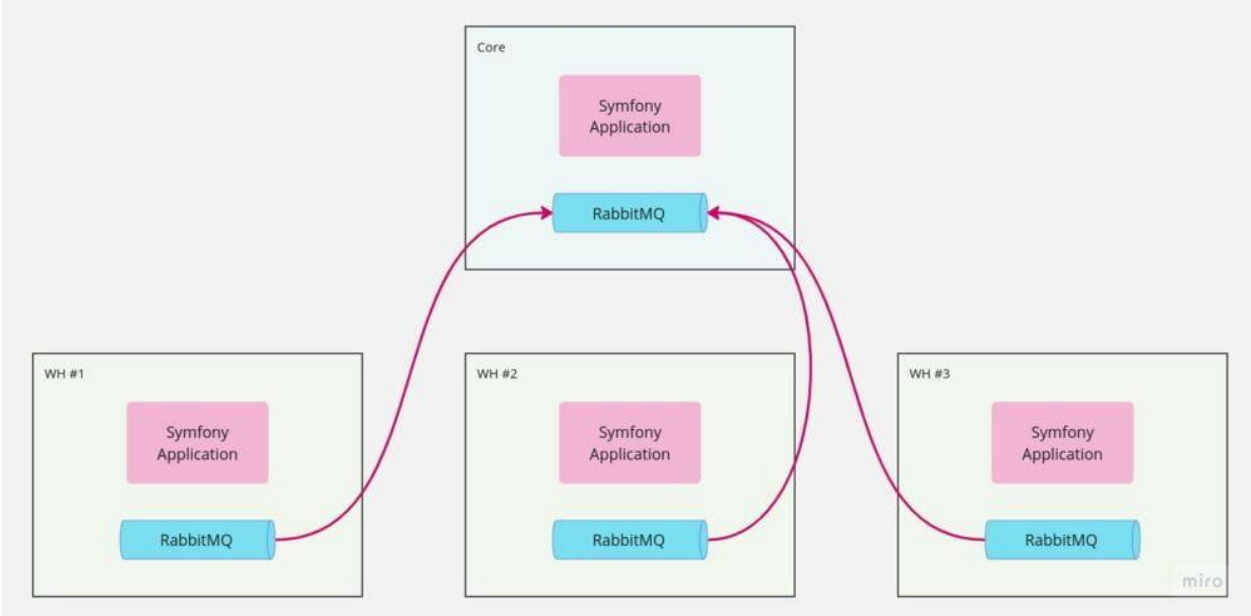


Рис. 2.3 – Робота Symfony messenger

Реляційна цілісність бази даних забезпечується каскадними обмеженнями на рівні зовнішніх ключів, які детально прописані для кожної залежної таблиці. При видаленні або оновленні батьківських сутностей система використовує стратегії каскадного видалення залежних метаданих або встановлення значень у стан null для збереження історичної цінності даних, наприклад, при видаленні профілю користувача його коментарі та створені завдання залишаються в системі з анонімізованим автором. Описана структура інформаційного забезпечення повністю покриває функціональні вимоги щодо ведення краудсорсингових кампаній, дозволяє асинхронно обробляти події через черги повідомлень Symfony Messenger (рис 2.3) і створює надійну основу для реактивної взаємодії користувачів у реальному часі.

2.3 Проєктування алгоритмів роботи системи та взаємодії

Успішність функціонування веборієнтованої системи управління життєвим циклом інноваційних ідей та формування команд безпосередньо залежить від точного моделювання алгоритмічного забезпечення, яке керує поведінкою бізнес-сутностей, координує асинхронні процеси та організує роботу інтелектуального AI-агента. У межах проєктування архітектури було розроблено комплекс взаємопов'язаних алгоритмів, що охоплюють життєвий цикл ініціатив, логіку графа станів мовної моделі, процеси комплектування проєктних команд та транзакційний облік елементів гейміфікації. Усі алгоритмічні рішення розроблені з урахуванням мікросервісної взаємодії між фреймворками Symfony, NestJS та модулем обробки даних на мові Python, забезпечуючи високу швидкість реакції інтерфейсу та цілісність даних на рівні реляційної СУБД.

Ключовим елементом системи виступає алгоритм управління життєвим циклом інноваційної ідеї, реалізований на базі концепції кінцевих автоматів (State Machine) за допомогою компонента Symfony Workflow. Даний алгоритм забезпечує суворий контроль за переходами сутності між логічними статусами, запобігаючи несанкціонованим змінам станів та автоматично ініціюючи супутні системні події. На початковому етапі, коли стейкхолдер взаємодіє з інтелектуальним асистентом, ідея перебуває у стані чернетки. Алгоритм обробки вхідних даних передбачає, що сутність не може бути опублікована у загальному каталозі до моменту, поки ІІІ агент не сформує повний пакет обов'язкових метаданих, що фіксуються в інформаційній структурі. Після успішної валідації та присвоєння категорії ідея переходить у статус публікації, що робить її доступною для всього краудсорсингового середовища та запускає процеси відкритого обговорення, коментування та збору реакцій користувачів.

Наступний крок алгоритму життєвого циклу активує фазу формування команди, під час якої рядові користувачі платформи отримують можливість

		А.В. ВОЗНІЮК			ІІЗ – 121 – 26 – ІІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		38

подавати заявки на участь у проєкті. Алгоритм обробки заявок контролює зміну станів сутності запитів на приєднання, перевіряючи наявність вільних рольових місць, визначених стейкхолдером для конкретної ініціативи. При переході ідеї у статус активної розробки алгоритм автоматично блокує приймання нових заявок, ініціює створення першого планового спринту та активує модуль моніторингу завдань. Завершення життєвого циклу проєкту передбачає переведення ідеї у фінальний статус із подальшим запуском підсистеми розподілу рейтингових балів та фіксації досягнень, що забезпечує логічне закриття всіх пов'язаних завдань і спринтів.

Окрему складність становить алгоритм функціонування інтелектуального асистента, спроектований у межах Python-мікросервісу з використанням бібліотеки LangGraph. Логіка взаємодії з великою мовною моделлю Gemini побудована не як лінійний скрипт, а як циклічний керований граф, де кожен вузол відповідає за певну аналітичну підзадачу, а ребра визначають умови переходу на основі аналізу відповідей ШІ. Перший вузол графа відповідає за ініціалізацію сесії та привітання користувача. При отриманні неформатованого тексту ідеї керування передається до вузла первинного аналізу, де Gemini здійснює пошук ключових сутностей, таких як назва, технологічний стек, очікувані результати та потенційні ризики. Якщо модель ідентифікує брак критично важливої інформації, алгоритм через умовне ребро повертає процес на етап генерації уточнювальних запитань, формуючи діалогове вікно зі стейкхолдером.

Після того як усі необхідні параметри зібрано, активується вузол структурування, завданням якого є трансформація розмовного тексту у стандартизований об'єкт формату JSON. Алгоритм промпт-інжинірингу на цьому етапі суворо обмежує вихідні дані моделі, вимагаючи відповідності заздалегідь визначеній схемі даних, яка містить поля назви, детального опису, категорій та масиву необхідних проєктних ролей. Готовий об'єкт проходить фінальний вузол валідації, де перевіряється коректність структури документа.

За умови успішного проходження перевірки Python-мікросервіс через REST-інтерфейс передає сформовані дані на основний бекенд системи, де асинхронні PHP workers здійснюють фізичний запис у таблиці бази даних, зв'язуючи нову ідею з профілем її творця. Весь ланцюжок викликів логується у системі LangSmith, що дозволяє відстежувати часові характеристики реакції штучного інтелекту та оптимізувати витрати токенів.

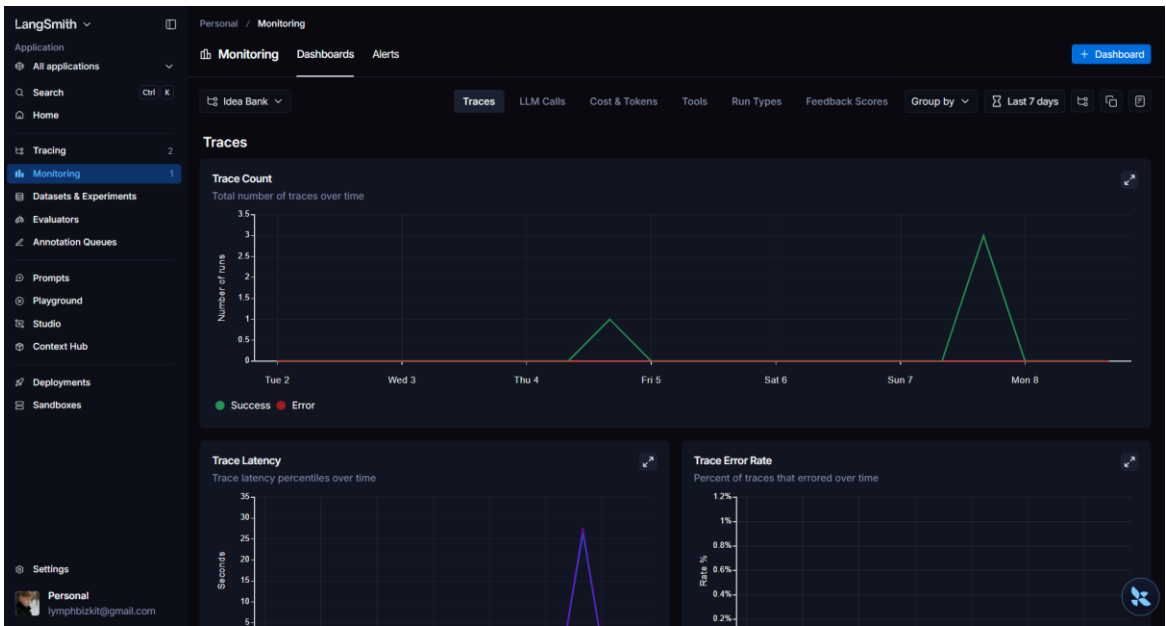


Рис. 2.4 – Система трекінгу LLM-моделей

Алгоритм динамічного формування команд базується на аналізі сутностей заявок та зіставленні профілів компетенцій розробників із вимогами проєкту. Коли користувач подає запит на участь, алгоритм зчитує масив його підтверджених навичок та порівнює їх із переліком необхідних технологій, які були згенеровані моделлю Gemini для цієї ідеї. Пріоритетність розгляду кандидатів визначається на основі їхнього поточного місця у Глобальному рейтингу учасників, що вираховується через сукупність накопичених балів за попередні успішно виконані завдання. Стейкхолдер проєкту отримує структурований дашборд, де кандидати ранжуються за коефіцієнтом релевантності, який враховує збіг навичок, досвід у схожих категоріях проєктів та рівень активності на платформі. При прийнятті позитивного

рішення алгоритм автоматично створює запис у таблиці зв'язку користувачів та ідей, закріплюючи за учасником обрану проєктну роль.

Важливою складовою інженерного забезпечення системи є транзакційний алгоритм гейміфікації та обліку рейтингових балів, який відповідає за мотивацію користувачів і підтримання академічної доброчесності середовища. Будь-яка дія, що передбачає зміну балансу балів користувача — наприклад, успішне закриття завдання в межах спринту або розблокування нового досягнення — обробляється через механізм ізольованих баз даних із рівнем ізоляції транзакцій `Serializable`. Це повністю виключає можливість виникнення стану гонитви (`race condition`) при одночасному нарахуванні балів декількома PHP workers. Алгоритм зчитує правила досягнення з конфігураційного поля таблиці ачівок, перевіряє виконання встановлених умов за допомогою аналізу логів завдань, і у випадку успіху створює новий запис у таблиці користувацьких досягнень. Паралельно запускається подія в чергу повідомлень, яка через сервіс `Mercure` та `WebSockets` миттєво оновлює дашборд користувача на фронтенді, забезпечуючи максимальну реактивність та інтерактивність вебсистеми.

У межах розробленої вебсистеми управління життєвим циклом інноваційних ідей та формування команд для автентифікації [23] та авторизації користувачів застосовується механізм безпечного передавання даних у форматі `JSON Web Token` (рис. 2.5). Після успішного підтвердження облікових даних розробника або стейкхолдера сервер формує токен, який містить закодовану інформацію про профіль, зокрема унікальний ідентифікатор та рівень доступу. Цей токен криптографічно підписується за допомогою секретного ключа, що гарантує цілісність та достовірність даних, запобігаючи несанкціонованому втручанню. Застосування такого підходу забезпечує надійний та масштабований метод керування доступом у розподіленій мікросервісній архітектурі платформи [5], де важлива швидкість взаємодії та гнучке управління правами.

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Для генерації токена на стороні серверної частини системи, реалізованої засобами обраних фреймворків Symfony та NestJS, використовуються відповідні модулі безпеки, які формують структуру на основі об'єкта користувача. Процес включає створення інформаційного навантаження, що містить ідентифікатор учасника, електронну пошту, визначену роль у системі та встановлений термін дії сесії. Сформований об'єкт кодується у форматі текстового представлення, після чого підписується за допомогою обраного криптографічного алгоритму. Підпис гарантує, що дані не були змінені сторонньою особою після створення. У структурі згенерованого токена умовно виділяються три складові: заголовок з інформацією про алгоритм, безпосереднє інформаційне навантаження та верифікаційний підпис.

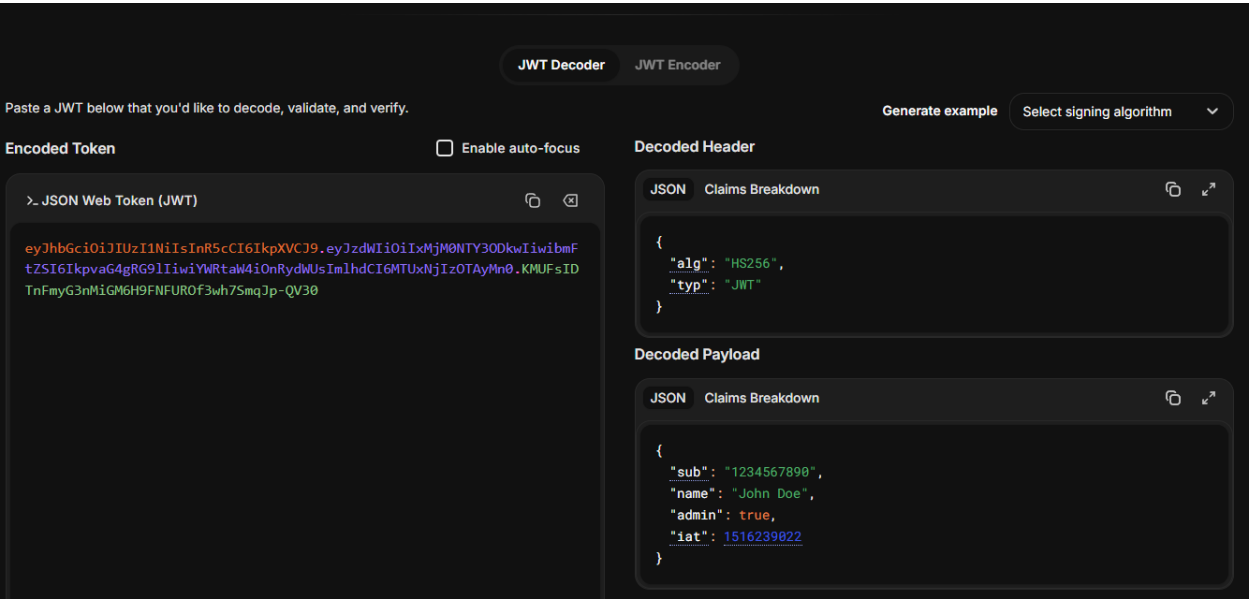


Рис. 2.5 – Розшифрований вміст JWT

На стороні клієнтського вебдодатка після успішної авторизації через відповідний програмний інтерфейс отриманий токен зберігається у локальному сховищі браузера. Вибір локального сховища зумовлений необхідністю збереження сесії користувача навіть після перезавантаження сторінки або тимчасового закриття вкладки із краудсорсинговою платформою. Під час кожного наступного звернення до захищених маршрутів системи, наприклад, при створенні нової ідеї чи подачі заявки на вступ до проєктної команди, клієнтський додаток автоматично додає збережений токен у

заголовок авторизації кожного мережевого запиту. Цей процес реалізується через глобальну конфігурацію HTTP-клієнта, що дозволяє централізовано додавати необхідні дані без дублювання логіки в окремих компонентах інтерфейсу.

На рівні бекенду функціонує механізм проміжного програмного забезпечення, який перехоплює всі вхідні запити та перевіряє наявність і коректність переданого токена. Якщо підпис визнається дійсним, а термін дії не минув, ідентифікаційні дані витягуються та стають доступними в контролерах для подальшої обробки бізнес-логіки. На основі цих даних система приймає рішення про надання доступу до конкретних ресурсів згідно з проєктними ролями. У випадку відсутності або недійсності токена запит автоматично відхиляється з поверненням відповідного статусу помилки доступу. Завдяки такому рішенню архітектура системи залишається безстейтовою, оскільки сервер не зберігає інформації про сесію між запитами, що є критично важливим для забезпечення високої продуктивності та гнучкого масштабування розробленої системи.

2.4 Серверне забезпечення

Для забезпечення повного контролю над апаратними ресурсами, гнучкості масштабування та підвищення загального рівня безпеки програмного комплексу, розгортання цільового середовища було реалізовано на базі власного виділеного сервера. З метою надійної ізоляції процесів застосовано технологію апаратної віртуалізації на рівні ядра операційної системи Linux за допомогою гіпервізора KVM [24] (рис. 2.6).

Адміністрування гіпервізора та комплексний моніторинг стану системи здійснюється за допомогою графічної вебконсолі управління Red Hat Web Console, що надає централізований інтерфейс для зручного контролю системних ресурсів. Для функціонування розробленої системи управління інноваційними ідеями було створено ізольовану віртуальну машину з емульованою архітектурою. Апаратна конфігурація віртуального середовища

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		43

передбачає виділення 48 гігабайтів оперативної пам'яті та двох віртуальних процесорних ядер у режимі наскрізної передачі функцій центрального процесора.

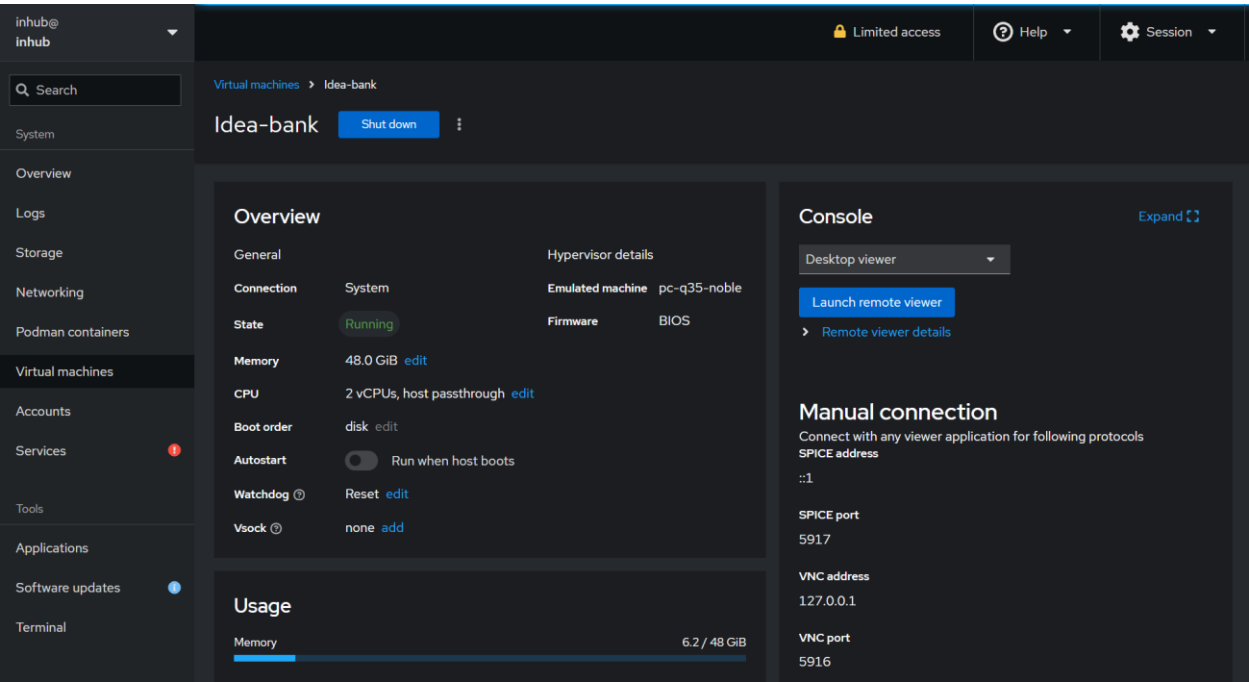


Рис. 2.6 – Інтерфейс Red Hat Web Console

Це дозволяє гостьовій операційній системі максимально ефективно використовувати апаратні інструкції фізичного хоста. Дискова підсистема реалізована за допомогою віртуального накопичувача об'ємом 30 гігабайтів у розширюваному форматі qcow2, який підключено через шину обміну даними virtio для забезпечення високої швидкості дискових операцій.

Мережева взаємодія віртуальної машини організована через оптимізований мережевий інтерфейс типу virtio, підключений до стандартної віртуальної мережі гіпервізора з використанням механізму трансляції мережевих адрес. Гостьовій системі динамічно призначається внутрішня маршрутизована адреса локальної підмережі, що надійно ізолює її від прямого доступу ззовні. Для організації безпечного доступу до вебдодатка з глобальної мережі Інтернет на рівні брандмауера фізичної хост-системи налаштовано спеціальні правила переадресації портів, які автоматично перенаправляють

весь вхідний цільовий трафік на відповідну внутрішню IP-адресу віртуальної машини.

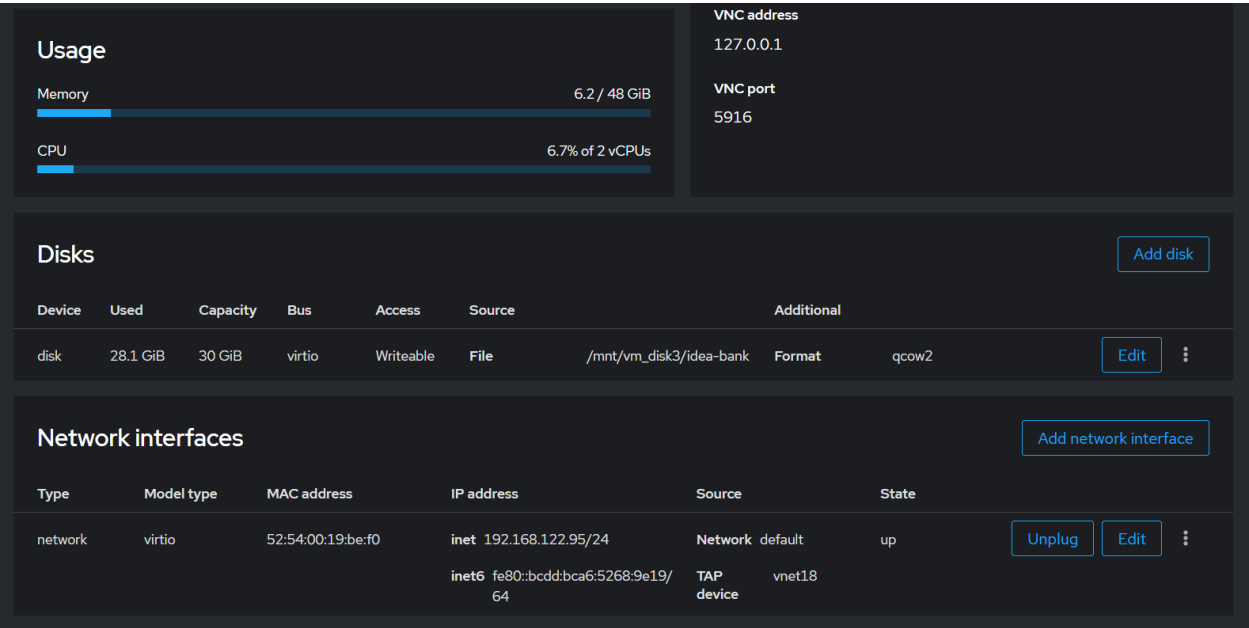


Рис. 2.7 – Мережевий інтерфейс VM та використання диска.

Для ефективного розподілу запитів між різними мікросервісами розробленої системи всередині віртуального середовища розгорнуто зворотний проксі-сервер. Його основним завданням є термінація захищених з'єднань, обробка вхідних запитів та їх подальша внутрішня маршрутизація до відповідних сервісів основного бекенду або ізольованого модуля штучного інтелекту. Також хочу зазначити, що сервер не зберігає файлів, які були завантажені користувачами прямо на хості. Саме тому розмір дискового простору віртуальної машини не є великим, адже всі файли зберігаються в хмарному сховищі файлів AWS S3. Водночас кількість оперативної пам'яті потрібна для запуску локальної LLM Ollama.

Окрему увагу під час конфігурації локальної серверної інфраструктури було приділено питанням кібербезпеки та захищеного управління доступом. Віддалене управління фізичним сервером та віртуальною машиною здійснюється виключно за протоколом безпечного оболонкового з'єднання з використанням криптографічних ключів, тоді як традиційну автентифікацію за паролем було повністю відключено на рівні конфігурації системної служби.

Для безпечного зберігання, генерації та управління життєвим циклом приватних ключів доступу, а також інших критичних конфігураційних секретів інфраструктури, було використано систему управління паролями та секретами Кеерер. Застосування цього інструментарію забезпечує надійне шифрування конфіденційних даних стандартом високого рівня та дозволяє безпечно інтегрувати необхідні ключі у змінні середовища під час роботи конвеєра автоматизованого розгортання, ефективно унеможливаючи компрометацію доступів через системи контролю версій.

Для забезпечення захищеної та контрольованої мережевої взаємодії між глобальною мережею та віртуальною машиною на рівні фізичного хоста гіпервізора KVM було налаштовано механізм трансляції мережевих адрес. Гостьовій операційній системі призначається ізольована статична IP-адреса з внутрішнього діапазону віртуальної мережі, що унеможливорює пряме несанкціоноване звернення до неї із зовнішнього середовища. Маршрутизація трафіку реалізується за допомогою підсистеми пакетного фільтрування операційної системи хоста, де визначені суворі правила переспрямування портів. Завдяки цьому всі мережеві запити, що надходять на публічну адресу фізичного сервера, проходять етап інспекції та автоматично транслуються на відповідні порти внутрішньої віртуальної машини, забезпечуючи прозору, але захищену доставку мережевих пакетів.

Організація віддаленого адміністрування віртуального середовища вимагає особливого підходу до безпеки, тому доступ через протокол безпечної оболонки налаштовано з використанням прокидання портів з рівня хоста. Замість прямого доступу використовується правило переадресації, за яким підключення до спеціально виділеного нестандартного порту фізичного сервера автоматично спрямовується на системний порт служби управління всередині віртуальної машини. Автентифікація за паролем повністю відключена на обох рівнях інфраструктури на користь обміну криптографічними ключами. Для безпечного управління цими ключами

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		46

застосовується система Кеерер (Рис. 2.8), яка дозволяє автоматизовано інтегрувати облікові дані безпосередньо в термінальні сесії або конвеєри розгортання, повністю виключаючи необхідність зберігання приватних ключів у відкритому вигляді на локальних робочих станціях [13].

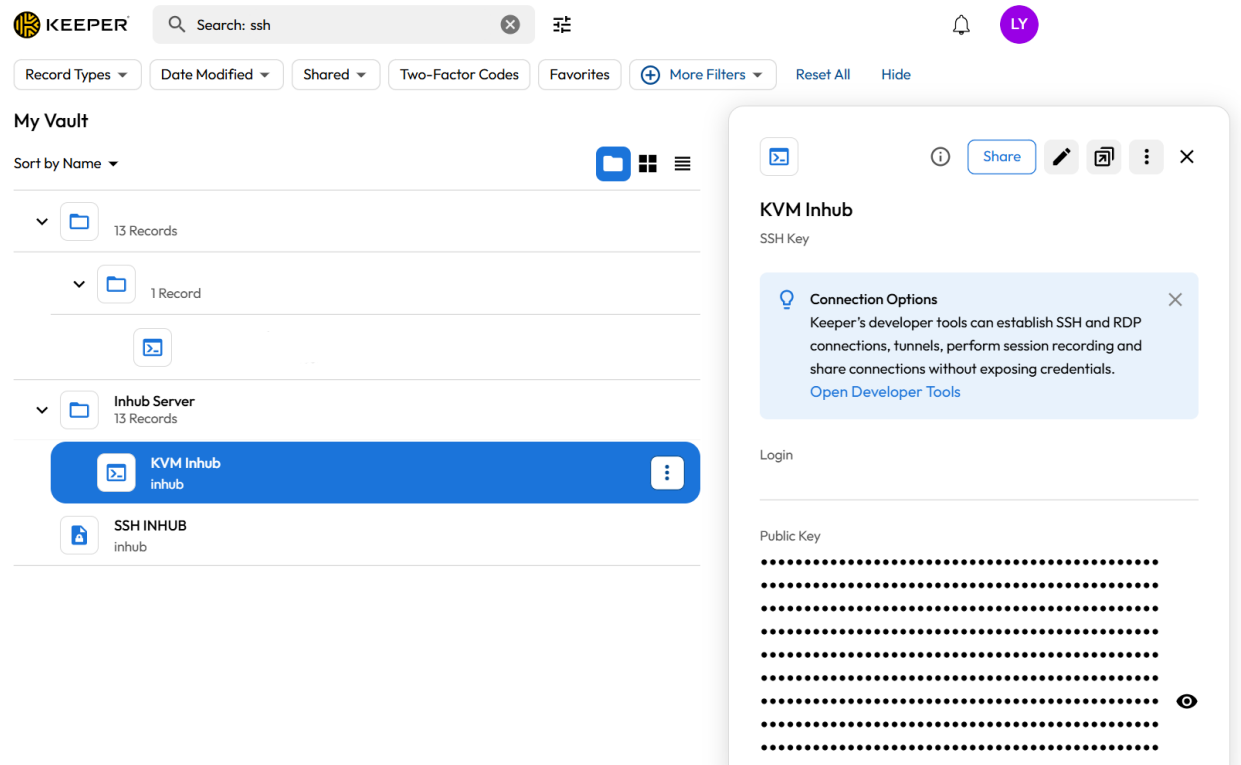


Рис. 2.8 – Інтерфейс Кеерер та збереження SSH ключів.

Досить цікавим фактом, є те що Кеерер доступний у всіх можливих дистрибутивах лінукс, як термінальний менеджер паролів. Також має інтерфейс в десктоп додатках для всіх операційних систем.

Для ефективного розподілу вхідного вебтрафіку та додаткового захисту вебдодатка на рівні самого гіпервізора розгорнуто та налаштовано зворотний проксі-сервер. Він виконує роль єдиної точки входу для всіх зовнішніх запитів до системи, здійснюючи розшифрування захищених з'єднань за допомогою сертифікатів безпеки безпосередньо на фізичному сервері. Після дешифрування та первинної перевірки запитів проксі-сервер перенаправляє цільовий трафік у віртуальну мережу до внутрішньої адреси гостьової системи. Таке архітектурне рішення не лише суттєво знижує обчислювальне навантаження на віртуальну машину, звільняючи ресурси для роботи

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		47

інтелектуальних модулів системи, але й приховує внутрішню топологію мережі від потенційних зловмисників, створюючи додатковий надійний буферний шар безпеки [3].

2.5 Реалізація CI/CD та архітектури збереження файлів

Сучасний підхід до розробки веборієнтованих систем управління життєвим циклом інноваційних ідей потребує повної автоматизації процесів валідації, тестування та доставки програмного забезпечення на цільові сервери. Для забезпечення стабільної роботи розроблюваної системи, мінімізації людського фактора та прискорення релізного циклу було спроектовано та впроваджено конвеєр безперервної інтеграції та розгортання [9].

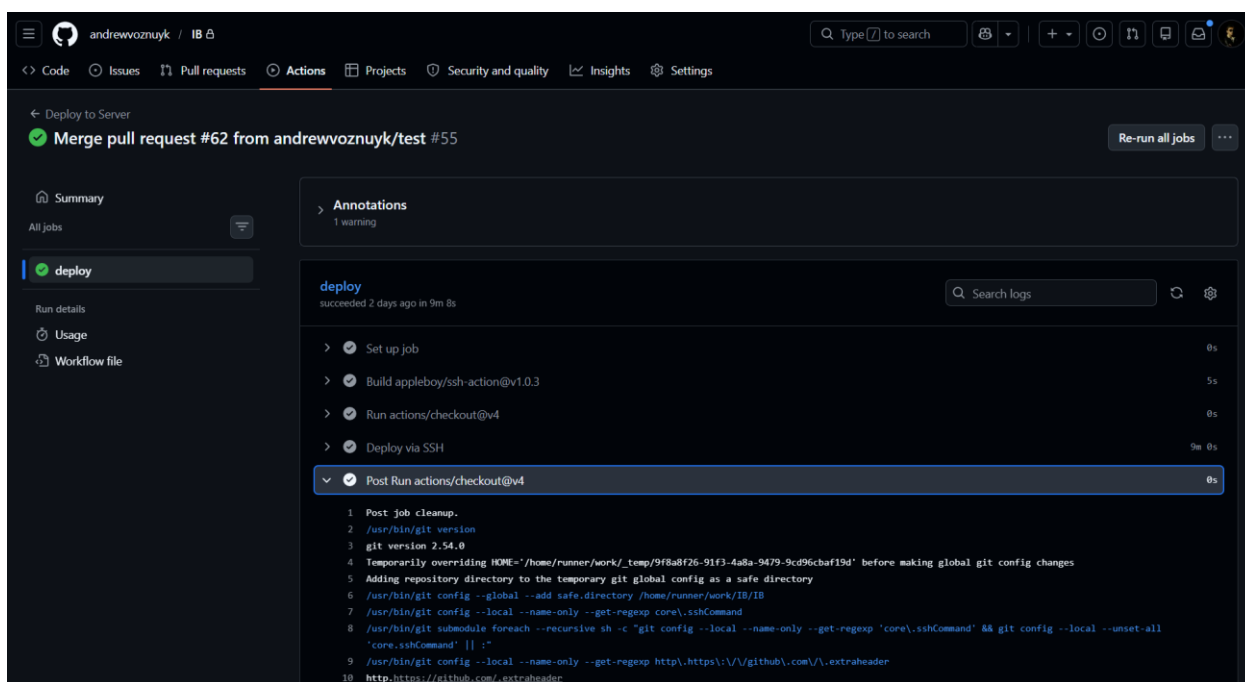


Рис. 2.9 – Процес роботи GitHub Actions.

Основними завданнями розробленого конвеєра є автоматична перевірка відповідності вихідного коду прийнятим стандартам оформлення, проведення статичного аналізу коду та запуск модульних та інтеграційних тестів при кожній спробі внесення змін до головної гілки репозиторію. Також у системі передбачено контейнеризацію додатку за допомогою технології Docker [27] для забезпечення ідентичності середовищ розробки та промислової

		A.B. ВОЗНІЮК			ІПЗ – 121 – 26 – ІПЗ	Арк.
		A.B. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		48

експлуатації і автоматизоване оновлення сервісів на сервері без зупинки обслуговування кінцевих користувачів. Конвеєр складається з кількох послідовних етапів, які виконуються ізольовано в контейнеризованому середовищі віртуальних виконавців. Послідовність виконання етапів автоматизованого розгортання програмного комплексу проілюстровано на (рис. 2.10). Архітектурний конфігураційний файл автоматизації розгортання, реалізований на базі інструментарію GitHub Actions [17], наведено в додатку В.

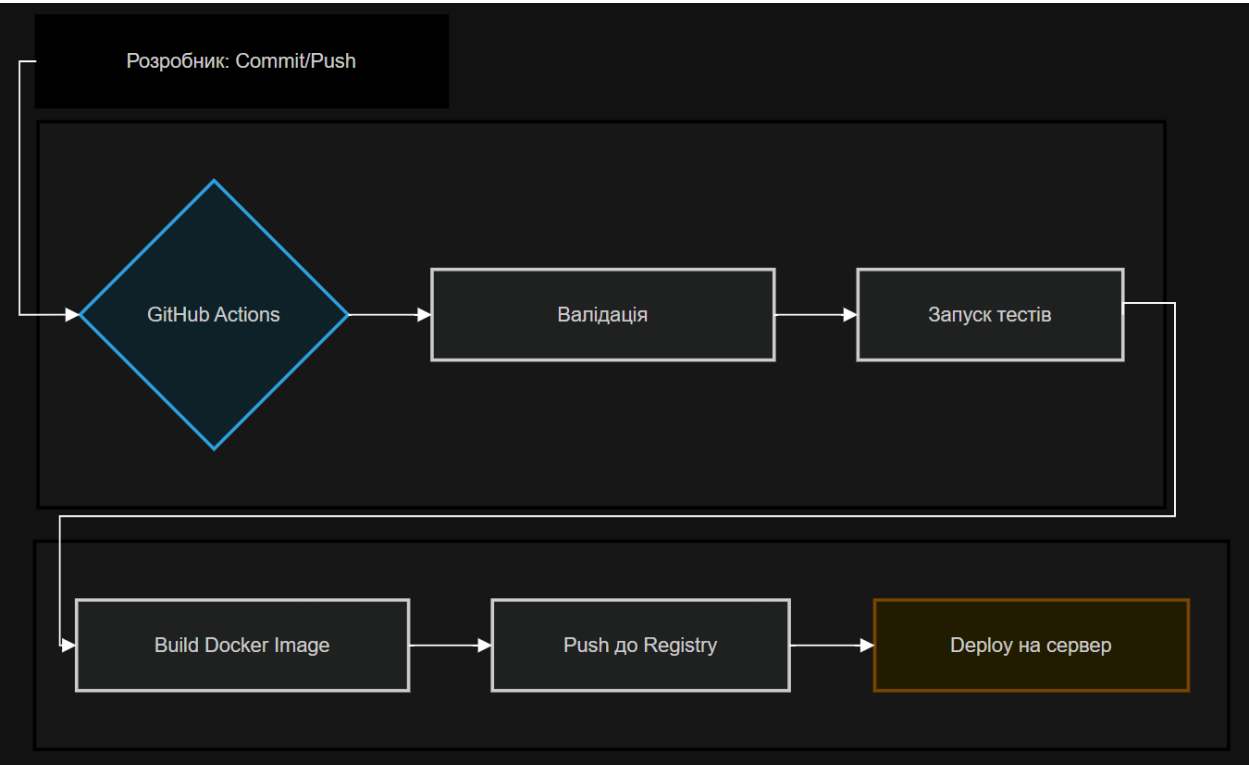


Рис. 2.10 – Схема процесу безперервної інтеграції та розгортання (CI/CD)

На краудсорсингових платформах та в системах управління проєктами постійно генерується значна кількість різноманітних статичних та динамічних файлів, до яких належать графічні матеріали презентацій інноваційних ідей, технічна документація, аватари профілів розробників, а також прикріплені файли у процесі спільної роботи сформованих команд. Збереження таких файлів безпосередньо на локальній файловій системі сервера додатків має низку критичних недоліків.

У разі горизонтального масштабування виникає потреба в постійній складній синхронізації локальних сховищ, а вихід з ладу диска сервера може призвести до безповоротної втрати проєктної документації, якщо заздалегідь не налаштовано багаторівневі системи резервного копіювання. Крім того, процеси читання та віддачі великих файлів клієнтам займають потоки вебсервера, що призводить до загального зниження швидкості обробки основних запитів до системи. Для усунення цих архітектурних обмежень у розробленій системі управління життєвим циклом інноваційних ідей було реалізовано інтеграцію з хмарним об'єктним сховищем від компанії Amazon [14].

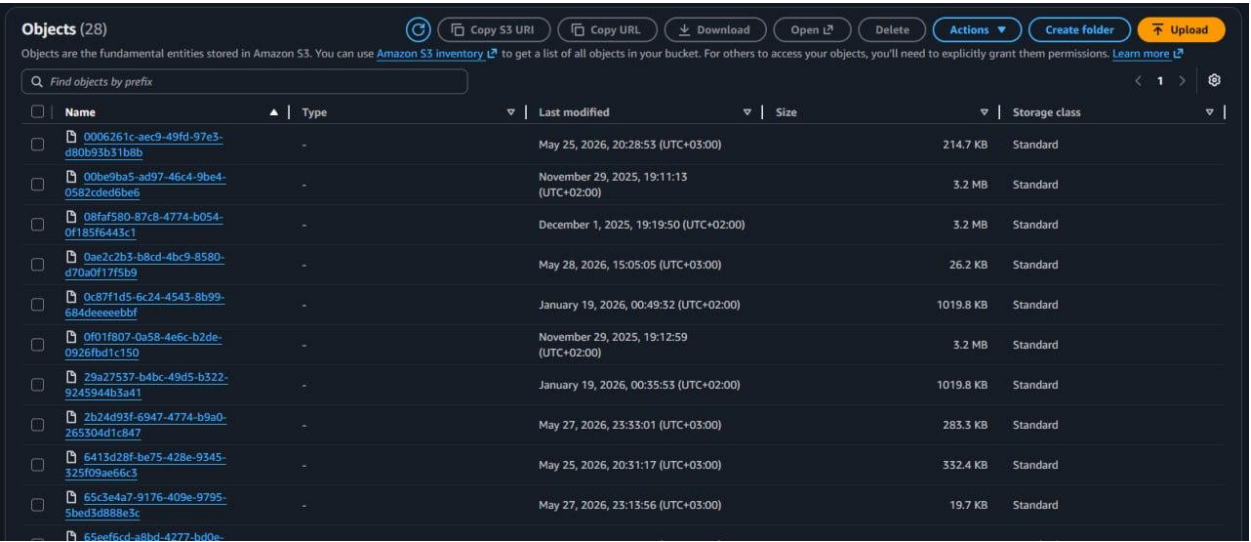


Рис. 2.11 – Місце і формат розташування файлів у AWS S3

Взаємодія програмного комплексу з хмарним сховищем побудована на принципі найменших привілеїв, що гарантує необхідний рівень безпеки даних. У консолі управління доступом було створено окремого службового користувача, для якого визначено чітку політику з правами доступу виключно до конкретного виділеного сховища інформаційної системи. Політики доступу, реалізовані у форматі обміну даними, наведено в додатку С.

В інформаційній системі реалізовано дві різні архітектурні моделі доступу до завантажених файлів залежно від їх призначення та встановленого рівня конфіденційності.

Перша модель застосовується для публічних активів, таких як аватари профілів розробників, обкладинки проєктів та загальні графічні матеріали презентацій інноваційних ідей, які зберігаються у публічно доступній директорії. База даних зберігає лише пряме посилання на об'єкт, завдяки чому клієнтський додаток завантажує зображення безпосередньо з мережі доставки контенту, повністю розвантажуючи основний бекенд системи.

Друга модель розроблена для конфіденційних документів, до яких належать закрита технічна документація, детальні бізнес-плани ініціатив, об'єкти інтелектуальної власності та внутрішні робочі матеріали сформованих команд. Для цих об'єктів програмно заблоковано будь-який публічний доступ з глобальної мережі. У випадку, коли авторизованому учаснику команди або стейкхолдеру проєкту необхідно переглянути документ, серверна частина додатка генерує тимчасове підписане посилання. Зазначене посилання містить унікальний криптографічний підпис та має суворо обмежений термін дії, після закінчення якого воно автоматично стає недійсним, що ефективно унеможлиблює несанкціонований витік комерційної та проєктної інформації.

Інтеграція інтелектуального асистента на базі великих мовних моделей для автоматизованого структурування описів інноваційних ідей вимагає впровадження спеціалізованих інструментів для моніторингу та аналізу роботи штучного інтелекту. Для вирішення цього завдання в архітектуру системи було інтегровано платформу LangSmith, яка забезпечує повний контроль над процесом взаємодії між серверною частиною додатка та мовною моделлю Gemini.

Використання цього сервісу дозволяє здійснювати безперервне трасування викликів, зберігаючи детальну історію всіх надісланих запитів від творців ідей та отриманих відповідей від ШІ агента. Це є критично важливим етапом для виявлення неочікуваної поведінки системи, налагодження складних ланцюжків обробки текстових даних при генерації карток проєктів

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		51

та підвищення загальної надійності інтелектуальних функцій розробленого програмного комплексу.

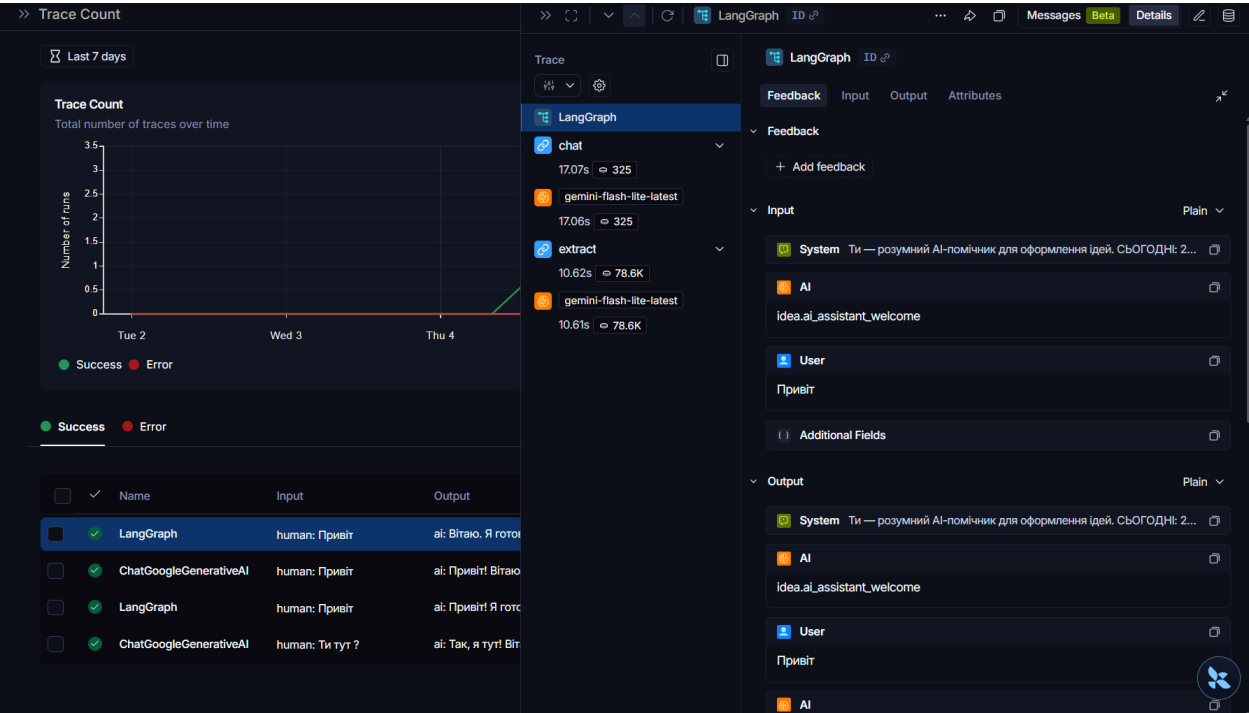


Рис. 2.12 – Безперервне трасування викликів у LLM

Окрім збереження історії взаємодії, сервіс LangSmith надає розширені можливості для збору та візуалізації телеметричних даних, що є необхідною умовою для оптимізації ресурсних витрат підприємства. Платформа автоматично агрегує детальну статистику про кількість використаних токенів під час кожної транзакції, загальний час затримки при генерації відповіді та відсоток успішних виконань системних запитів. Аналіз зібраних аналітичних метрик дає змогу оцінювати економічну ефективність використання мовних моделей, виявляти неефективні або надто об'ємні запити та своєчасно коригувати алгоритми формування системних промптів.

Висновки до другого розділу

У другому розділі спроектовано мікросервісну архітектуру веборієнтованої системи управління інноваційними ідеями, що дозволило ефективно розділити основну бізнес-логіку на базі фреймворків Symfony та NestJS та ресурсоємний модуль штучного інтелекту, реалізований мовою Python. Завдяки інтеграції великої мовної моделі Gemini та використанню

інструментів LangChain і LangGraph розроблено алгоритм циклічної взаємодії інтелектуального асистента з користувачем, що забезпечило автоматизований збір, аналіз і структурування неформатованих текстових даних у стандартизовані картки проєктів.

Розроблено оптимізовану структуру реляційної бази даних MySQL, яка гарантує безпеку, цілісність та швидкодію завдяки використанню універсальних унікальних ідентифікаторів, механізмів м'якого видалення та каскадного контролю цілісності. Побудовано комплекс логічних алгоритмів, що координують життєвий цикл ідей за допомогою кінцевих автоматів, забезпечують динамічне комплектування проєктних команд на основі аналізу компетенцій кандидатів та гарантують транзакційну безпеку при нарахуванні балів у системі гейміфікації.

Для забезпечення стабільності й масштабованості програмного комплексу спроектовано надійні інфраструктурні рішення. Впроваджено конвеєр безперервної інтеграції та розгортання з використанням технології контейнеризації Docker для автоматизованого оновлення сервісів. Процеси збереження файлів оптимізовано через інтеграцію з хмарним сховищем Amazon S3 із використанням механізму генерації тимчасових підписаних посилань для доступу до конфіденційних документів. Також налаштовано систему безперервної телеметрії роботи інтелектуальних модулів через платформу LangSmith для аналізу та подальшої оптимізації витрат обчислювальних ресурсів. Запропоновані архітектурні рішення комплексно вирішують поставлені в роботі інженерні завдання.

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		53

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ

3.1 Опис роботи з програмним додатком (Опис інтерфейсу)

Головна сторінка каталогу інноваційних ідей є центральним інформаційним вузлом інтерфейсу розробленої платформи, де користувачі мають змогу переглядати, шукати та взаємодіяти із запропонованими ініціативами. Структурно вікно додатка логічно розділене на декілька функціональних зон: верхнє глобальне навігаційне меню, основну робочу область відображення контенту та інтегровану бічну панель інтелектуального асистента. Навігаційна панель забезпечує швидкий перехід між ключовими підсистемами комплексу, такими як дашборд, події, завдання, рейтингова таблиця лідерів та спільнота, а також містить елементи управління профілем користувача та налаштуваннями локалізації.

В основній робочій області представлено масив поточних ініціатив у вигляді інтерактивних карток проєктів. Кожна картка компактно відображає найважливіші метадані ідеї, серед яких назва, тематична категорія, короткий текстовий опис суті пропозиції, ідентифікаційні дані автора та час публікації матеріалу. Інтерфейс карток також виводить актуальні соціальні метрики, зокрема лічильники вподобань та коментарів, що дозволяє візуально оцінювати рівень зацікавленості спільноти певним проєктом.

Для забезпечення зручної навігації та швидкого пошуку релевантної інформації у верхній частині робочої області реалізовано панель управління контентом. Вона включає випадаючі списки для фільтрації ідей за відповідними категоріями та інструменти сортування записів за критерієм новизни. Поруч розміщено акцентовану кнопку створення нової ініціативи, яка ініціює процес додавання об'єкта в систему. Зовнішній вигляд сторінки каталогу ідей та панелі інтелектуального помічника наведено на (рис. 3.1).

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		54

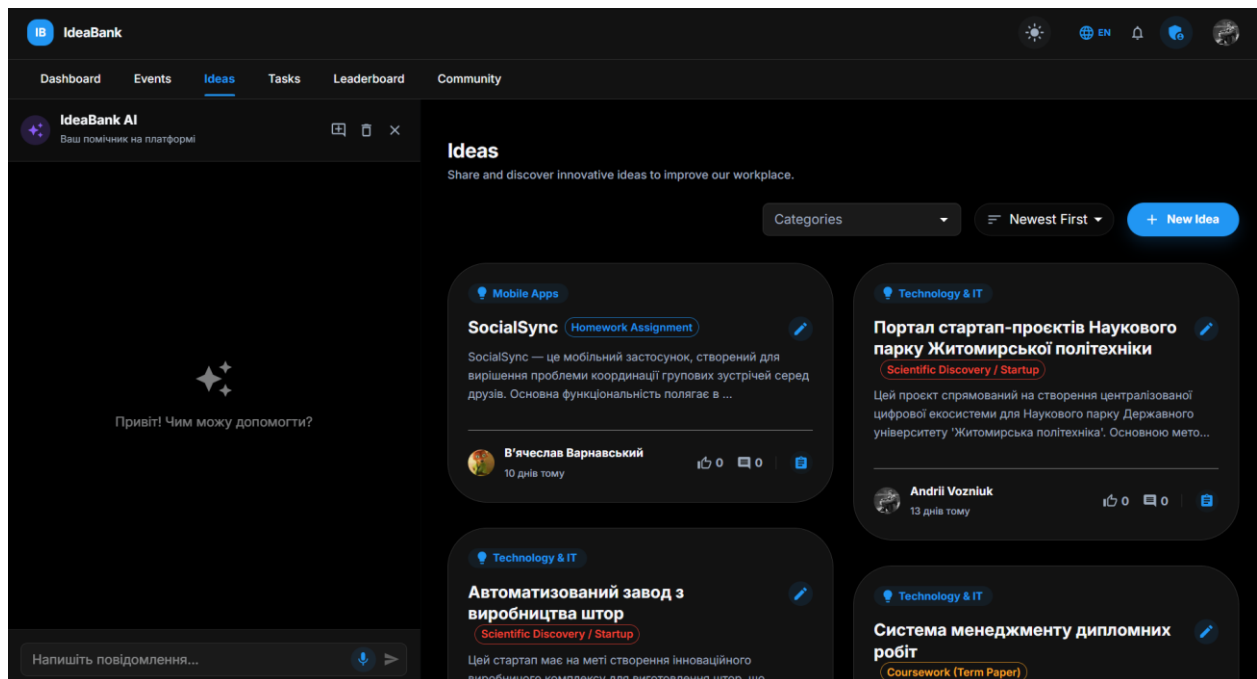


Рис. 3.1 – Інтерфейс сторінки каталогу інноваційних ідей та ШІ-асистента.

Ключовою особливістю розробленого користувацького інтерфейсу є наскрізна інтеграція інтелектуального помічника на базі великих мовних моделей, панель якого закріплена з лівого боку екрана і є доступною на кожній сторінці вебсистеми. ШІ-асистент функціонує у форматі інтерактивного діалогового чату, підтримуючи як текстове, так і голосове введення інформації. До його основних можливостей належить ведення природного діалогу з користувачем, надання розгорнутих консультацій щодо функціоналу платформи та принципів управління проектами. Важливою практичною функцією асистента є здатність безпосередньо допомагати стейкхолдерам в автоматизованому заповненні складних форм системи, шляхом збору неструктурованих даних через діалог та їх подальшої трансформації у стандартизовані параметри карток проектів. Такий архітектурний підхід до побудови інтерфейсу суттєво знижує когнітивне навантаження на користувача та підвищує загальну ефективність взаємодії з програмним комплексом.

Сторінка створення нової інноваційної ідеї являє собою комплексний інтерфейс, що містить розширену форму з великою кількістю спеціалізованих полів. Для оптимізації процесу заповнення цих даних та зменшення когнітивного навантаження на користувача, до інтерфейсу безпосередньо

		A.B. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		A.B. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		55

інтегровано панель інтелектуального помічника. Взаємодія розпочинається з того, що користувач вводить неструктурований текстовий або голосовий запит у вікно чату, описуючи загальне бачення свого проєкту. Наприклад, формулюється базовий запит про бажання створити платформу для колаборації викладачів та студентів певного навчального закладу. Процес ініціалізації діалогу та формування запиту до інтелектуального помічника наведено на (рис. 3.2).

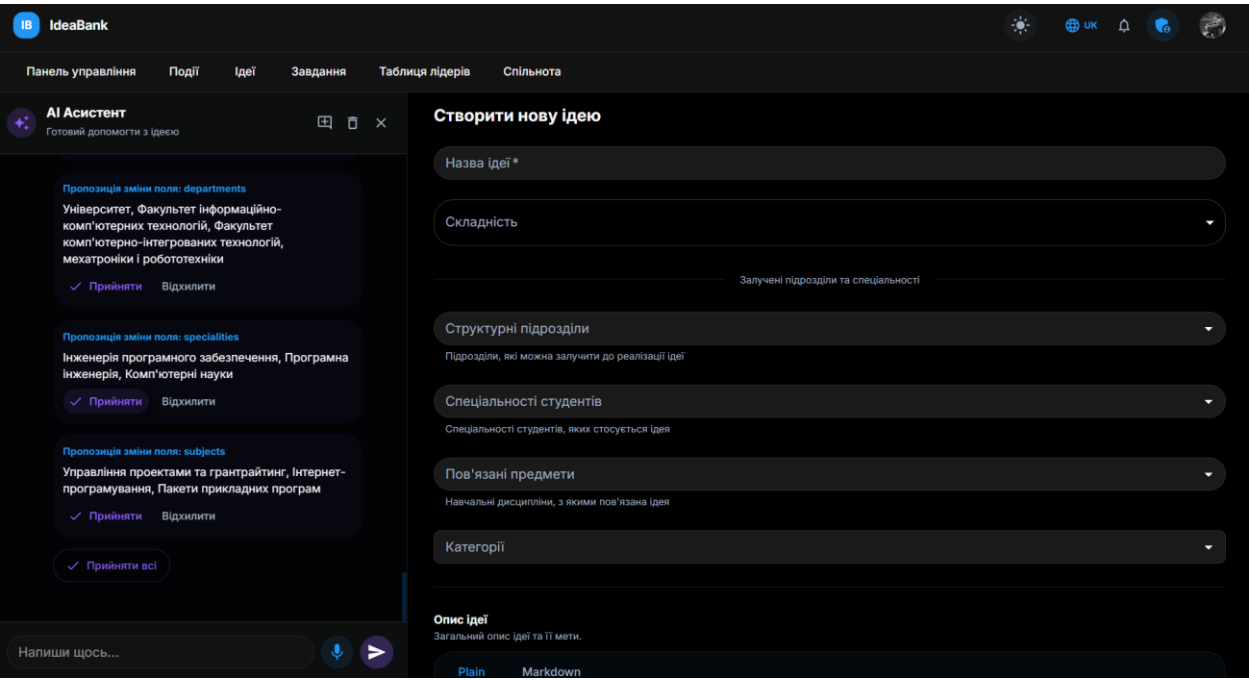


Рис. 3.2 – Ініціалізація діалогу з ШІ-асистентом для створення проєкту.

Отримавши вхідний текст, клієнтський додаток ініціює складний процес обробки даних через шар інтеграції. Перед тим як передати запит до великої мовної моделі, ізольований мікросервіс звертається до основного прикладного програмного інтерфейсу системи для отримання актуальних системних довідників. Програмний комплекс завантажує поточні списки доступних структурних підрозділів, спеціальностей, навчальних дисциплін та загальних категорій проєктів. Зібрані масиви даних динамічно вбудовуються у системну підказку для нейромережі. Таке збагачення контексту є архітектурно необхідним кроком, оскільки воно гарантує, що штучний інтелект не буде генерувати неіснуючі назви факультетів чи предметів, а здійснюватиме точне

алгоритмічне зіставлення намірів користувача з наявною реляційною базою даних платформи.

На наступному етапі мовна модель аналізує семантику користувацького запиту та використовує механізми структурування вихідних даних для формування комплексної відповіді. Штучний інтелект самостійно генерує релевантну назву проєкту, розгорнутий опис ідеї, формулює проблематику в полі анкера та здійснює багатокритеріальну класифікацію. Під час класифікації алгоритм підбирає масиви ідентифікаторів для відповідних багатовибіркових списків форми, визначаючи дотичність ідеї до конкретних інженерних чи гуманітарних спеціальностей. Результат обробки повертається на клієнтську частину вебдодатка у вигляді чітко структурованого об'єкта даних.

Інтерфейс вебсистеми спроектовано з дотриманням принципу збереження контролю за оператором [11], тому згенеровані дані не перезаписують вміст форми миттєво. Натомість у бічній панелі інтелектуального асистента генерується динамічний список інтерактивних пропозицій щодо зміни конкретних полів. Користувач візуально оцінює окремі інформаційні блоки з пропозиціями для структурних підрозділів, спеціальностей студентів, пов'язаних предметів та дат завершення. Кожен такий блок містить елементи керування для індивідуального прийняття або відхилення запропонованих значень, а також передбачено глобальну функцію для синхронного застосування всіх згенерованих даних. Інтерфейс взаємодії з динамічними пропозиціями від мовної моделі представлено на (рис. 3.3).

		<i>A.B. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>A.B. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Рис. 3.3 – Часткове заповнення полів форми ІІІ агентом.

Після підтвердження пропозицій користувачем, клієнтська логіка інтерфейсу автоматично заповнює відповідні компоненти у формі. Багатовибіркові списки миттєво оновлюються, відображаючи додані графічні теги обраних факультетів, спеціальностей та категорій складності.

Рис. 3.4 – Автоматизоване заповнення конфігураційних полів та багатовибіркових списків.

Великі текстові поля розширеного опису та анкера заповнюються згенерованим контентом із дотриманням необхідного форматування. Цей оптимізований потік взаємодії дозволяє за лічені секунди перетворити коротку абстрактну думку на повноцінно оформлену і структуровану картку проєкту, повністю підготовлену до збереження у базі даних та подальшої публікації. Результат автоматизованого заповнення конфігураційних полів та розширених текстових блоків форми продемонстровано на (рис. 3.4) та (рис. 3.5) відповідно.

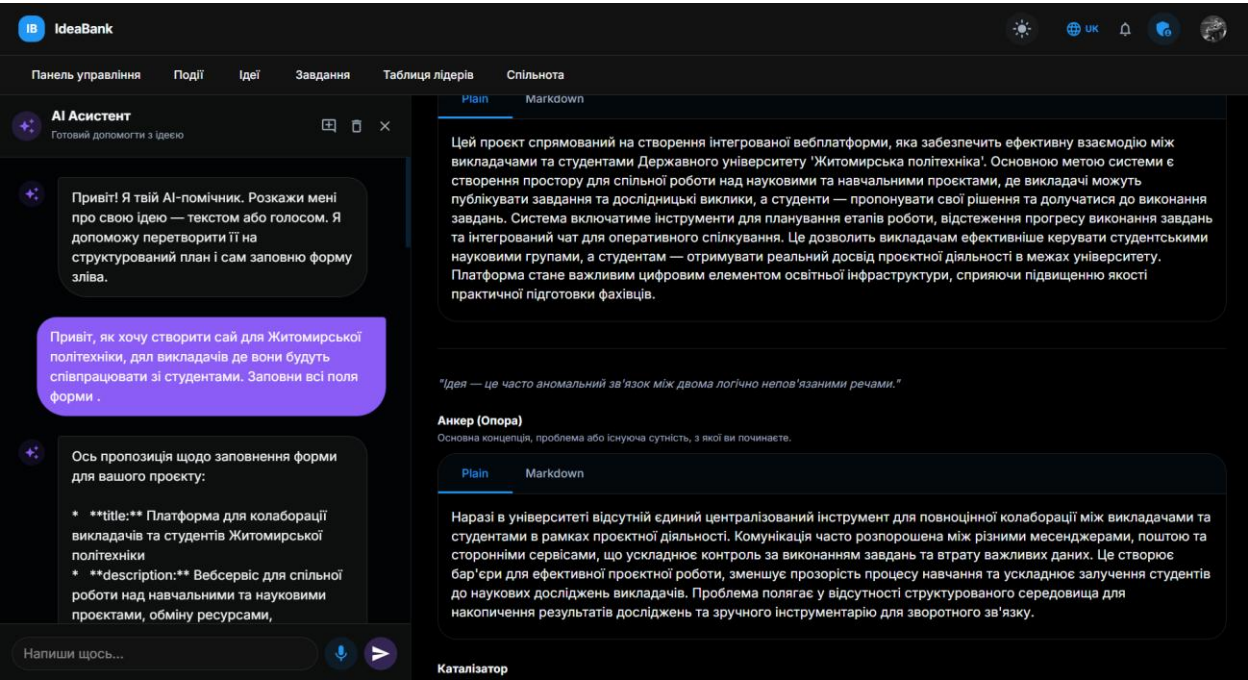


Рис. 3.5 – Автоматизоване заповнення розширених текстових блоків опису проєкту.

Автоматизоване заповнення розширених текстових блоків опису проєкту. Сторінка детального перегляду інноваційної ідеї є основним робочим простором для управління проєктом та взаємодії між стейкхолдерами. Інтерфейс логічно поділено на центральну інформаційну область та бічну панель інструментів управління командою. У верхній частині робочої області розташовано навігаційну панель із кнопками швидкого доступу до ключових підсистем проєкту. Серед них наявні елементи переходу до інтегрованих чатів для оперативного спілкування, модулів управління завданнями та спринтами

для реалізації гнучких методологій розробки, а також інструменти редагування та видалення ініціативи, доступні автору або адміністратору системи. Окрім цього, передбачено механізм соціальної взаємодії через систему вподобань, що дозволяє користувачам висловлювати підтримку проєкту. Загальний вигляд верхньої частини сторінки ідеї наведено на (рис. 3.6).

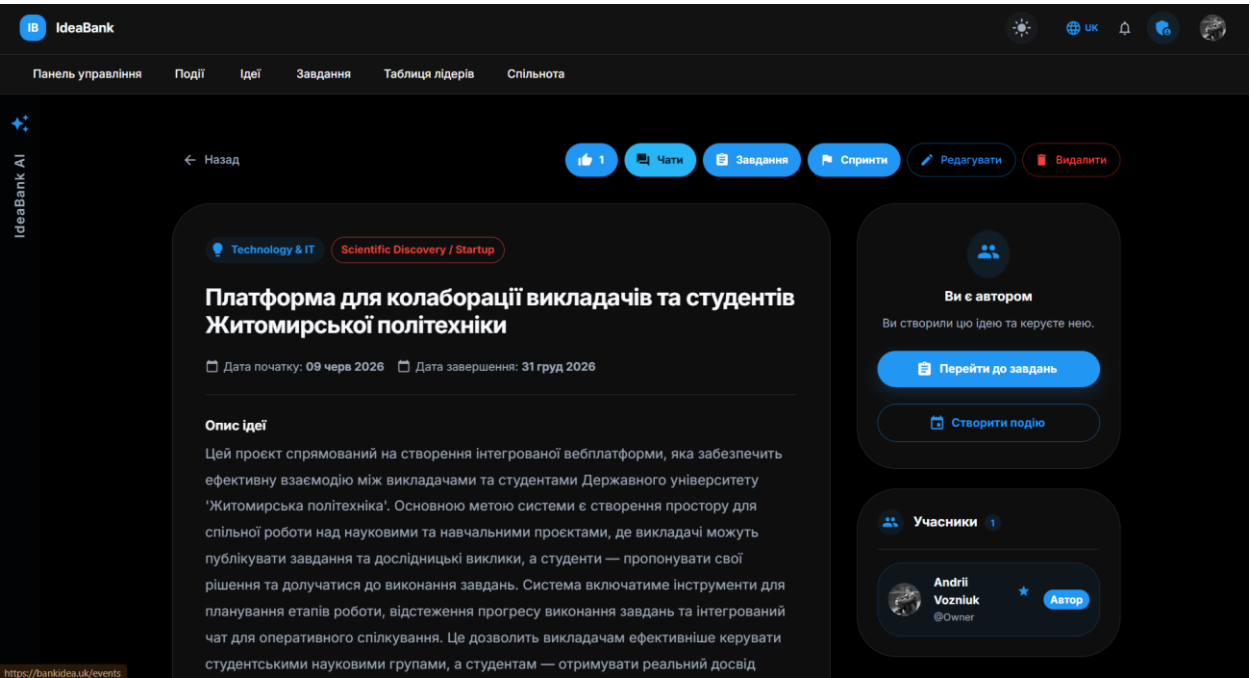


Рис. 3.6 – Інтерфейс верхньої частини сторінки детального перегляду інноваційної ідеї.

Центральний блок сторінки містить вичерпну інформацію про ініціативу, яка була згенерована та заповнена на попередньому етапі за допомогою інтелектуального помічника. Тут відображаються класифікаційні графічні теги, які визначають приналежність до галузі та категорії складності проєкту, розгорнута назва, а також чітко зафіксовані часові рамки активності, що включають дати початку та планового завершення. Нижче розміщено розгорнутий текстовий опис, який деталізує основну мету, проблематику та запропоноване рішення. Права бічна панель виконує роль центру управління складом проєктної команди. Для автора ідеї тут виводиться спеціальний інформаційний блок із швидкими діями щодо переходу до управління

завданнями чи до створення нових подій. Нижче розташовано модуль учасників, де відображається поточний склад команди із зазначенням ролей, зокрема виділенням власника та інших залучених фахівців.

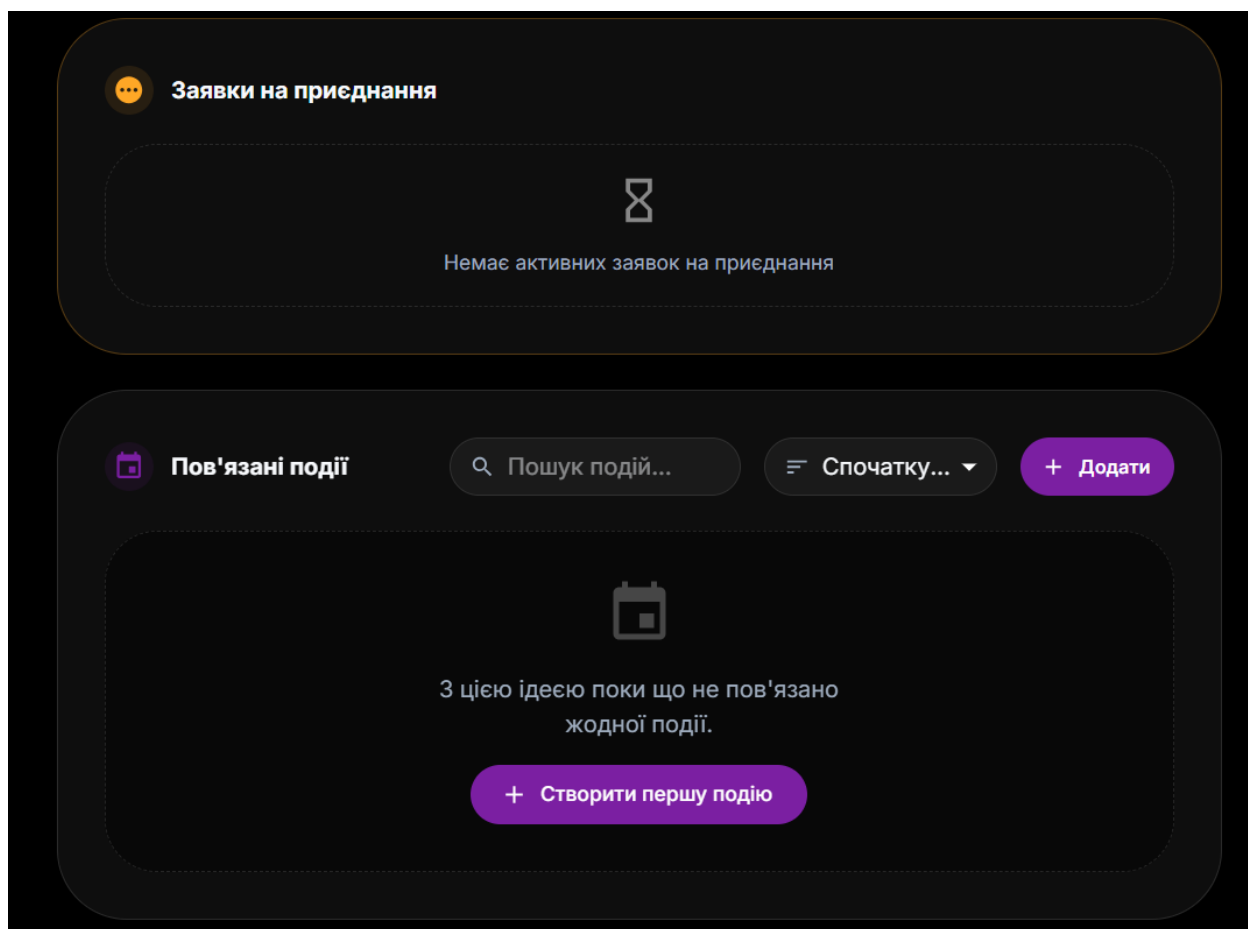


Рис. 3.7 – Інтерфейс управління заявками на приєднання та пов'язаними подіями проєкту.

Нижня частина сторінки детального перегляду присвячена процесам масштабування команди та організації заходів у межах проєкту. Окремий інтерфейсний блок призначений для управління заявками на приєднання, де автор або відповідальні особи можуть розглядати запити від інших користувачів платформи, які бажають долучитися до реалізації ініціативи. Під ним розташовано модуль пов'язаних подій, який дозволяє планувати зустрічі, презентації чи спільні обговорення. Цей блок оснащений внутрішньою системою пошуку, інструментами багатокритеріального сортування та елементами керування для створення нових записів, що забезпечує зручну

організацію робочого процесу без необхідності переходу до інших глобальних розділів вебдодатка. Структуру нижньої частини сторінки проєкту продемонстровано на (рис. 3.7).

3.2 Адаптивність та доступність додатка

З метою забезпечення максимальної доступності та зручності використання програмного комплексу на різноманітних пристроях, користувацький інтерфейс вебсистеми спроектовано з використанням принципів адаптивного вебдизайну. При відображенні на мобільних пристроях з меншою діагоналлю екрана архітектура сторінок автоматично трансформується для оптимізації робочого простору та покращення користувацького досвіду. Глобальне навігаційне меню, яке у десктопній версії розташовувалося у верхній частині екрана, у мобільному поданні переноситься на нижню панель у вигляді закріпленого блоку з іконками. Такий підхід гарантує швидкий та ергономічний доступ до ключових розділів платформи однією рукою, що є сучасним стандартом ергономіки для мобільних вебдодатків.

Адаптація інтерфейсу безпосередньо стосується складних форм введення даних, зокрема сторінки створення нової інноваційної ідеї. Усі елементи управління, такі як текстові поля, випадні списки багатовибіркового пошуку та конфігураційні перемикачі, автоматично масштабуються на всю доступну ширину екрана мобільного пристрою. Це забезпечує комфортну взаємодію з елементами інтерфейсу за допомогою сенсорного управління та запобігає виникненню горизонтального прокручування. Оскільки одночасне паралельне відображення розширеної форми та бічної панелі інтелектуального помічника на вузьких екранах фізично неможливе, доступ до ІІІ-асистента реалізовано через спеціальну виділену кнопку у верхній частині робочої області. Натискання на цю кнопку викликає помічника у форматі перекриваючого екрана. Візуальне представлення адаптованої форми створення ідеї для мобільних пристроїв наведено на (рис. 3.8).

		А.В. ВОЗНІЮК			ІІЗ – 121 – 26 – ІІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		62

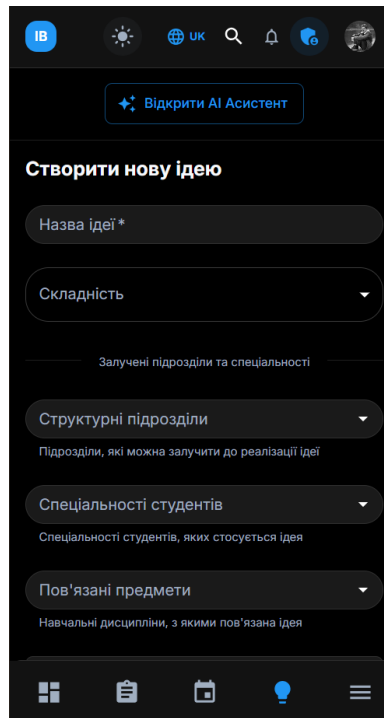


Рис. 3.8 – Адаптивний інтерфейс форми створення ідеї з нижньою навігаційною панеллю.

Робоча область самого інтелектуального помічника у мобільному поданні також повністю оптимізована для зручного ведення діалогу в умовах обмеженого простору. Інтерфейс чату розгортається на весь екран пристрою, відображаючи повідомлення користувача та розгорнуті згенеровані відповіді мовної моделі у вигляді компактних інформаційних блоків, які легко читаються. Панель введення тексту, оснащена кнопками для надсилання повідомлень та активації мікрофона для голосового вводу, надійно зафіксована в нижній частині екрана. Це дозволяє стейкхолдеру безперешкодно взаємодіяти з асистентом, зберігаючи при цьому доступ до всього функціоналу інтелектуального аналізу та автоматизованого заповнення форм, який доступний у повноцінній десктопній версії системи. Інтерфейс мобільної версії вікна взаємодії з ШІ-асистентом зображено на (рис. 3.9).

		A.B. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		A.B. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		63

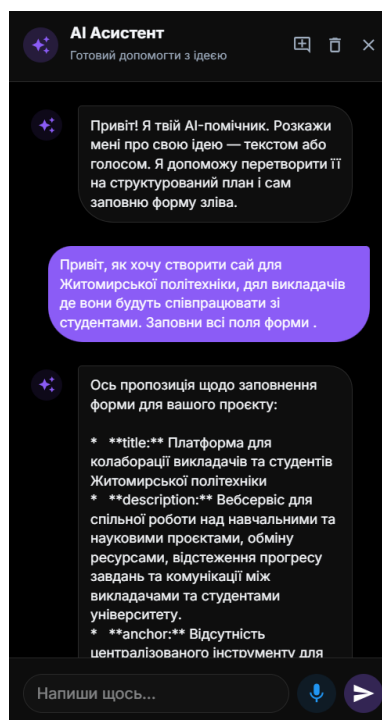


Рис. 3.9 – Адаптивний інтерфейс чату інтелектуального помічника на мобільному пристрої.

Організація взаємодії з масивами даних в умовах обмеженого екранного простору мобільних пристроїв вимагає застосування спеціалізованих інтерфейсних рішень. У мобільній версії вебсистеми управління інноваційними ідеями механізм фільтрації та сортування контенту реалізовано за допомогою плаваючої панелі управління (Floating Action Button). На головній сторінці каталогу ідей ця панель закріплена в нижній частині робочої області екрана, безпосередньо над глобальним навігаційним меню. Таке розташування забезпечує швидкий доступ до інструментів пошуку без необхідності прокручування сторінки до самого верху. Панель містить акцентовану кнопку виклику фільтрів, а також окрему функціональну клавішу із символікою зірок, яка призначена для швидкого виклику інтелектуального помічника у форматі перекриваючого діалогового вікна. Візуальне відображення сторінки каталогу з плаваючою панеллю управління наведено на рис. 3.10.

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		64

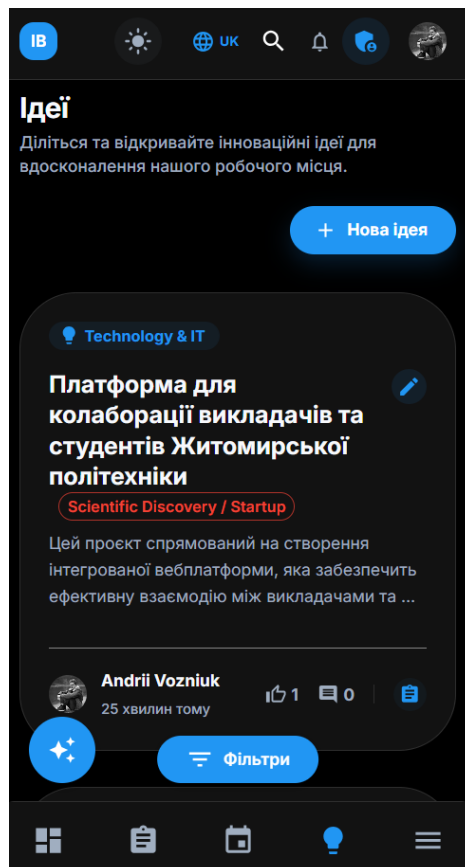


Рис. 3.10 – Інтерфейс каталогу ідей із плаваючою панеллю фільтрації та виклику III.

При активації кнопки фільтрів система розгортає модальне вікно, яке перекриває нижню частину екрана (Bottom Sheet), фокусуючи увагу користувача виключно на параметрах пошуку. Внутрішня структура цього вікна містить оптимізовані для сенсорного керування випадаючі списки. Користувач має можливість обрати специфічну категорію проєктів для відображення, а також налаштувати критерії сортування, наприклад, розташувати ініціативи у хронологічному порядку від найновіших до найстаріших. Після вибору необхідних параметрів застосування фільтрів здійснюється натисканням масивної кнопки підтвердження, яка розташована в самому низу модального вікна, що робить її легкодоступною для натискання великим пальцем. Такий підхід мінімізує кількість необхідних дій для знаходження релевантної інформації та підвищує загальну ергономічність

		A.B. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		A.B. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		65

мобільного додатка. Інтерфейс модального вікна налаштування фільтрів та сортування продемонстровано на (рис. 3.11).

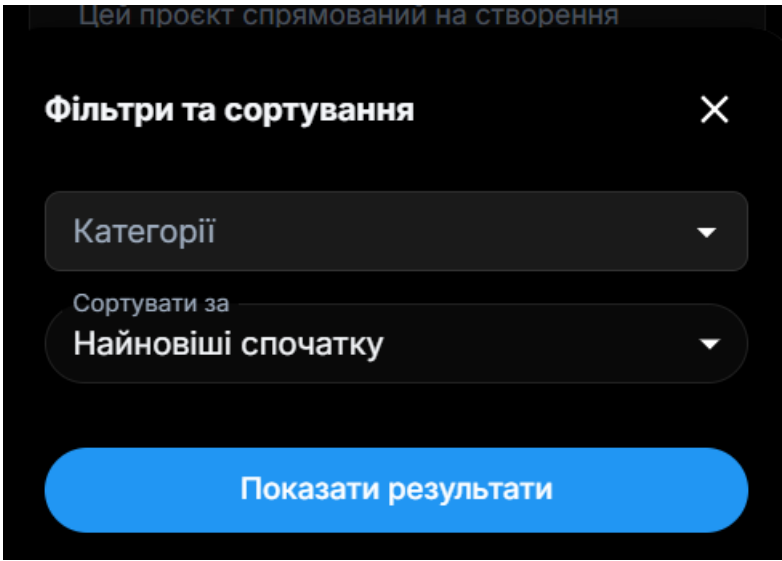


Рис. 3.11 – Модальне вікно налаштування параметрів фільтрації та сортування ідей.

3.3 Роутинг та маршрутизація

Для забезпечення ефективної взаємодії між клієнтською та серверною частинами вебсистеми управління інноваційними ідеями розроблено структуровану схему маршрутизації. У таблиці 3.1 наведено перелік маршрутів прикладного програмного інтерфейсу та відповідних шляхів клієнтського додатка, які відповідають за управління життєвим циклом інноваційних ідей. Ця таблиця описує логіку обробки мережевих запитів для отримання загального каталогу ініціатив, перегляду детальної інформації про конкретний проєкт, а також процесів автоматизованого створення, редагування та видалення записів у базі даних платформи.

Таблиця 3.1

Маршрутизація сторінки ідей на API та на клієнтській частині

HTTP метод	Маршрут API	Маршрут React	Дія
GET	/api/ideas	/ideas	Отримання переліку ідей

GET	/api/ideas/:ideaId	/ideas/:id	Отримання деталей конкретної ідеї
POST	/api/ideas	-	Створення нової ідеї (з форми)
PUT	/api/ideas	-	Редагування наявної ідеї
DELETE	/api/ideas/:ideaId	-	Видалення ідеї з системи

Взаємодія клієнтського вебдодатка із серверною частиною системи здійснюється через прикладний програмний інтерфейс, побудований за принципами REST-архітектури [8]. Обмін інформацією між фронтендом та бекендом відбувається виключно у форматі структурованих даних JSON, що вимагає встановлення відповідних HTTP-заголовків для кожного мережевого запиту. Зокрема, клієнтський додаток під час ініціалізації з'єднання повинен обов'язково передавати заголовки визначення типу контенту та формату прийняття даних зі значенням application/json. Це гарантує коректну серіалізацію та десеріалізацію даних як на етапі відправлення параметрів нової інноваційної ідеї, так і під час отримання масивів інформації для відображення у каталозі проєктів. Для виконання операцій із захищеними ресурсами платформи, до яких належать створення ініціатив, управління складом команди та редагування власного профілю, прикладний програмний інтерфейс вимагає обов'язкової ідентифікації клієнта. Робота з авторизованими запитами передбачає додавання до них криптографічного токена доступу, який попередньо отримується під час успішного проходження процедури входу до системи. Цей токен передається у стандартному заголовку авторизації з використанням схеми Bearer. На рівні клієнтської логіки налаштовано механізм автоматичного перехоплення запитів, який динамічно додає необхідний авторизаційний заголовок до кожного вихідного HTTP-запиту,

забезпечуючи безперебійний доступ до приватних маршрутів прикладного інтерфейсу.

Для забезпечення централізованого керування краудсорсинговою платформою та підтримки актуальності її інформаційного наповнення розроблено спеціалізовану панель адміністратора. У верхній частині інтерфейсу розташовано аналітичний дашборд, який у режимі реального часу відображає ключові кількісні метрики активності системи: загальну кількість зареєстрованих користувачів, активні події, подані інноваційні ідеї та наявні класифікаційні категорії. Таке візуальне представлення дозволяє відповідальній особі здійснювати швидкий моніторинг загального стану програмного комплексу та оцінювати рівень залученості стейкхолдерів у процес генерації ідей без необхідності формування додаткових аналітичних звітів.

Основний робочий простір панелі адміністратора організовано у вигляді модульної сітки (рис. 3.12), що забезпечує інтуїтивно зрозумілий доступ до глибокої конфігурації платформи. Серед ключових елементів реалізовано модуль управління системними словниками, за допомогою якого адміністратор має змогу централізовано наповнювати та редагувати довідники тегів, категорій, навичок та освітніх програм, що є критично важливим для коректної роботи алгоритмів ШІ-асистента та механізмів фільтрації. Окрім цього, інтерфейс надає доступ до гнучкого налаштування параметрів внутрішньої системи управління завданнями, конфігурації поведінки вбудованих чатів та безпосереднього адміністрування облікових записів. Такий підхід гарантує високий рівень гнучкості системи, дозволяючи адаптувати її під змінні вимоги процесу формування команд без необхідності втручання у вихідний програмний код бекенду.

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		68

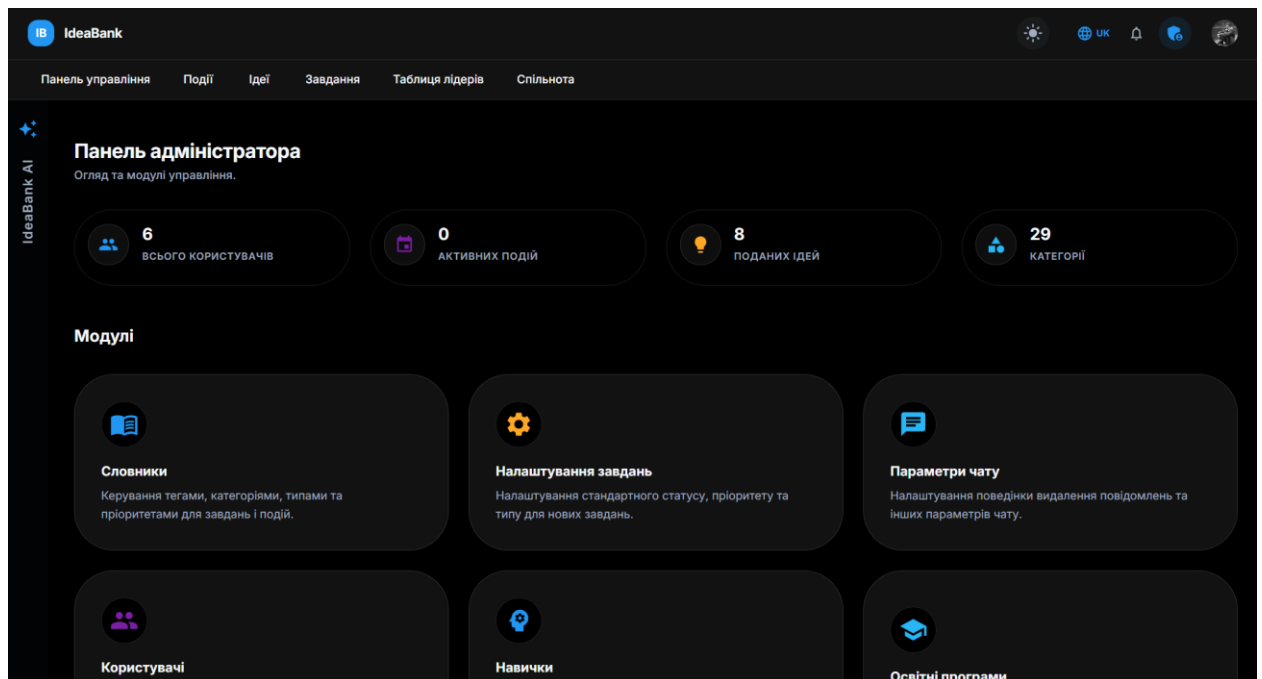


Рис. 3.12 – Адмінпанель для заповнення довідників порталу.

Організація роботи з прикладним програмним інтерфейсом базується на строгому використанні семантики стандартних HTTP-методів залежно від типу виконуваної операції. Для отримання даних, наприклад, завантаження списку доступних факультетів або перегляду деталей конкретного завдання, застосовується метод GET, при якому всі необхідні параметри фільтрації та пагінації передаються безпосередньо у рядку запиту. Створення нових сутностей, таких як публікація нової ідеї або подання заявки на приєднання до команди, виконується за допомогою методу POST, де основне інформаційне навантаження розміщується в тілі запиту. Для модифікації існуючих записів використовується метод PUT, а для безпечного видалення даних із системи — метод DELETE. Такий підхід забезпечує передбачуваність поведінки системи та дозволяє ефективно кешувати відповіді на рівні мережевої інфраструктури.

Важливою складовою роботи з API є механізм обробки відповідей та стандартизованих кодів стану серверних помилок. У разі успішного виконання операції сервер повертає статус успіху разом із запитаним об'єктом даних. Якщо ж під час валідації вхідних даних виявляються невідповідності, наприклад, при спробі створити проєкт із порожньою назвою, або неправильно

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		69

вказаною категорією, прикладний інтерфейс генерує відповідь із кодом помилки клієнтського запиту та масивом деталізованих повідомлень для кожного некоректного поля. При спробі несанкціонованого доступу повертаються коди відмови в авторизації, які клієнтський додаток перехоплює для автоматичного перенаправлення користувача на сторінку входу. Системний підхід до обробки помилок дозволяє інтерфейсу своєчасно інформувати стейкхолдерів про статус виконання їхніх дій та підтримувати високий рівень стабільності роботи вебсистеми.

3.4 User Stories та User Flows

У підрозділі розглядаються основні сценарії взаємодії користувачів із розробленою вебсистемою управління життєвим циклом інноваційних ідей та формування команд. Аналіз поведінкових патернів охоплює ключові етапи роботи з платформою, починаючи від базової реєстрації та налаштування безпеки з використанням JWT-токенів, до складних процесів управління профілями, генерації проєктних ініціатив та координації спільних завдань. Детальне моделювання цих процесів дозволяє глибоко зрозуміти, яким чином учасники різних рольових груп взаємодіють із функціональними модулями системи для досягнення своїх професійних чи академічних цілей. Описані потоки роботи надають чітке уявлення про логічну послідовність дій користувачів у різноманітних сценаріях експлуатації програмного комплексу.

Моделювання вимог через історії користувачів розкриває базові потреби кожної з визначених рольових груп платформи, до яких належать звичайні користувачі, ментори, лідери, стейкхолдери, інвестори та адміністратори. З позиції звичайного користувача формується потреба у перегляді глобального каталогу ідей із можливістю гнучкого фільтрування та сортування, а також у подачі заявок на участь у цікавих ініціативах із подальшим відстеженням їхнього статусу. Стейкхолдер, у свою чергу, висловлює потребу у зручних інструментах для генерації та публікації нових ідей із зазначенням необхідної кількості учасників та дедлайнів, де ключову роль відіграє взаємодія з

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		70

інтелектуальним AI-асистентом для автоматизованого структурування неформатованого тексту. Для адміністративного персоналу критично важливою є наявність інструменту імперсонації, який дозволяє тимчасово переглядати систему від імені будь-якого авторизованого користувача для оперативної діагностики та вирішення технічних проблем. Також невід’ємною ідеєю, була голосова взаємодія користувача з ШІ агентом (рис. 3.13)

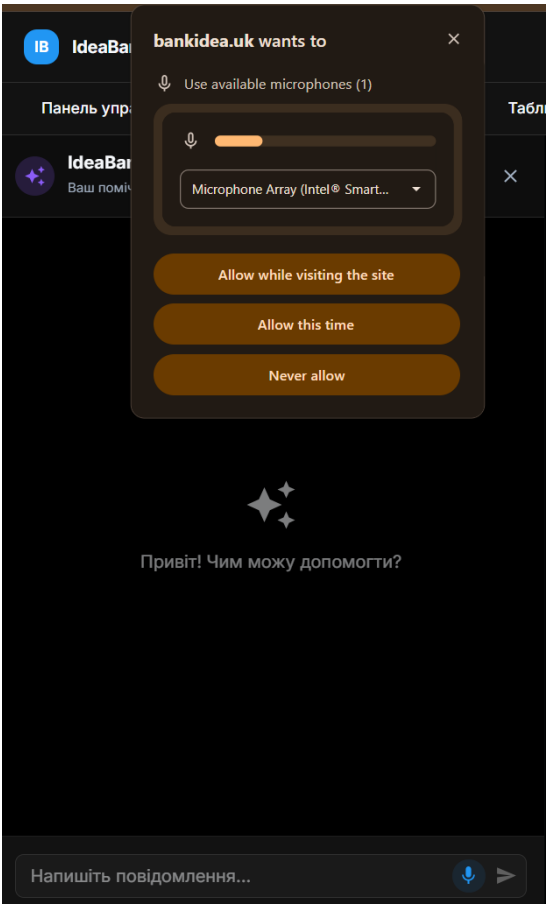


Рис. 3.13 – Голосова взаємодія користувача з ШІ агентом.

Окремий блок історій користувачів сфокусовано на питаннях персоналізації, приватності та мотиваційних механізмах платформи. Кожен учасник системи має потребу у гнучкому управлінні власним профілем, де він може зазначати свої професійні навички, відомі технології та пріоритетні категорії проєктів, одночасно контролюючи налаштування приватності для приховування контактних даних від сторонніх осіб. З точки зору гейміфікації та економіки балів, користувачі зацікавлені в отриманні об'єктивної оцінки своєї роботи, нарахуванні рейтингових балів за успішно завершені проєкти,

здобутті унікальних досягнень та просуванні у Глобальному рейтингу. Водночас для стейкхолдерів передбачено механізми управління ризиками, що реалізуються через можливість дострокового завершення проблемних проєктів із пропорційним відніманням рейтингових балів у відповідальних осіб залежно від порушених дедлайнів.

Послідовність виконання конкретних операцій у системі формалізується через розробку потоків взаємодії. Процес інтеграції нового учасника розпочинається на сторінці реєстрації, де після введення базових даних система генерує та надсилає лист для підтвердження електронної пошти. Після успішної верифікації користувач проходить автентифікацію, отримує захищений токен доступу та автоматично перенаправляється до головного каталогу ідей. Для ініціації нового проєкту стейкхолдер відкриває відповідну форму та розпочинає діалог із ШІ-асистентом, який структурує введену інформацію та формує готову картку ідеї для публікації. Після появи ідеї в загальному каталозі інші користувачі можуть ознайомитися з її деталями та надіслати запит на приєднання до команди розробників.

На фінальному етапі потоку взаємодії стейкхолдер здійснює обробку отриманих заявок від кандидатів, приймаючи рішення про схвалення або відхилення запитів із можливістю вказати аргументовану причину відмови. Після укомплектування команди ініціатива офіційно трансформується у статус активного проєкту, що відкриває доступ до інструментів спільної роботи та відстеження прогресу. Завершення життєвого циклу проєкту супроводжується запуском алгоритмів економіки балів: система автоматично аналізує внесок кожного учасника, розподіляє рейтингові бали, оновлює позиції у Глобальному рейтингу та присвоює відповідні досягнення за виявлену активність. Графічне представлення описаних кроків користувача, що демонструють логічні кроки при реєстрації, управлінні ідеями, формуванні команд та нарахуванні рейтингу, наведено на (рис. 3.14), що дозволяє наочно оцінити комплексну архітектуру взаємодії в межах розробленої вебсистеми.

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ІЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		72

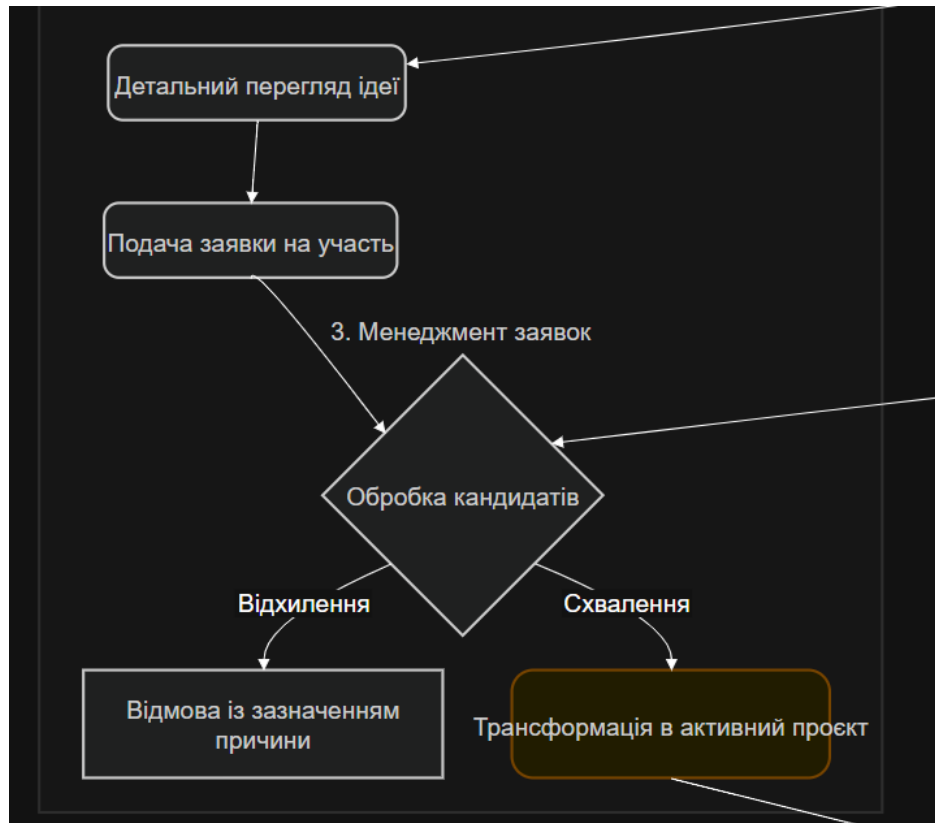


Рис. 3.14 – Графічне представлення описаних кроків користувача.

3.5 Заходи безпеки бази даних

Забезпечення конфіденційності, цілісності та доступності інформації у вебсистемі управління життєвим циклом інноваційних ідей вимагає впровадження комплексного підходу до захисту бази даних та управління конфігураційними секретами. Основою безпеки доступу до реляційної системи керування базами даних [28] є суворе розділення параметрів для локального середовища розробки та промислової експлуатації. Усі критичні дані, такі як адреси хостів, імена користувачів та паролі доступу до бази даних, передаються в додаток виключно у вигляді змінних середовища. Під час локальної розробки ці змінні зберігаються у спеціальних конфігураційних файлах формату .env. Для унеможливлення випадкової компрометації цих даних через репозиторії систем контролю версій, файли з локальними секретами суворо ігноруються алгоритмами відстеження змін, що гарантує відсутність конфіденційної інформації у вихідному коді програмного комплексу. Формування промислових конфігурацій відбувається за іншим

принципом і виключає зберігання секретів у відкритому вигляді на файловій системі сервера. Для безпечної доставки конфігураційних параметрів у продуктове середовище під час автоматизованого розгортання застосовується підсистема керування секретами на рівні конвеєра безперервної інтеграції. Використання механізму GitHub Secrets (рис. 3.15) дозволяє безпечно передавати облікові дані бази даних у процеси збірки та доставки без їхнього фізичного збереження на проміжних логічних вузлах.

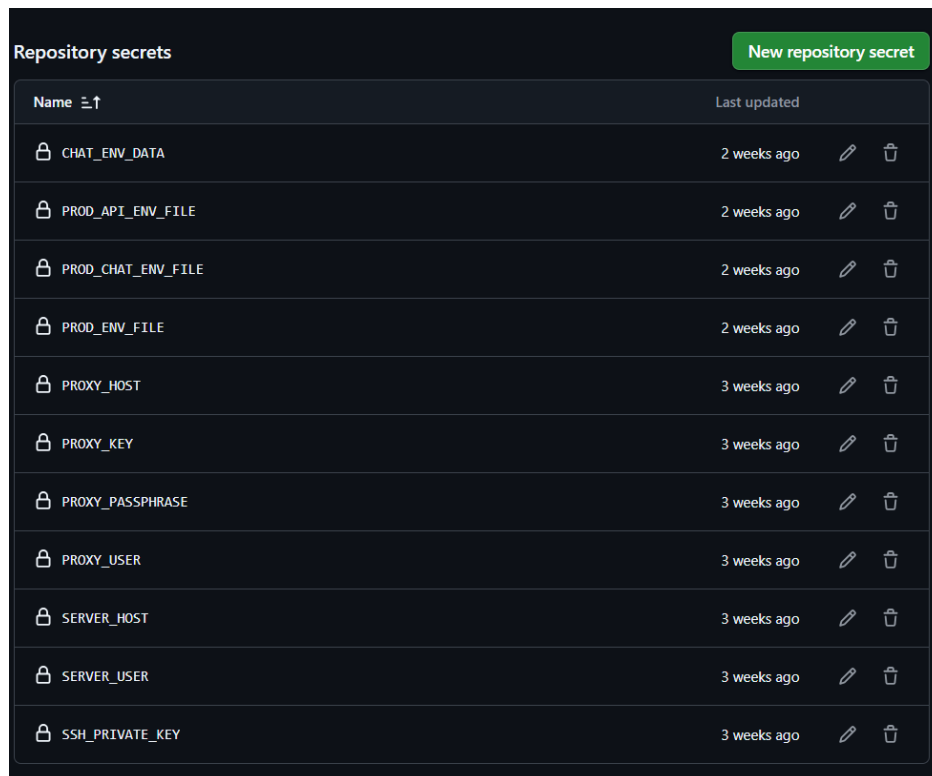


Рис. 3.15 – Github Secrets у репозиторії.

Першоджерелом для генерації та довгострокового зберігання цих критичних параметрів доступу виступає менеджер паролів та секретів Кеерг. Він забезпечує надійне криптографічне шифрування облікових записів бази даних та дозволяє адміністратору централізовано керувати доступами. Секрети з цієї системи безпечно інтегруються у конвеєр розгортання, дозволяючи динамічно ін'єктувати паролі безпосередньо у змінні середовища віртуальної машини лише в момент фактичного запуску або оновлення сервісів, що зводить до мінімуму ризику витоку інформації через людський фактор. Окрім безпеки облікових даних, захист інформаційного середовища

реалізується на рівні мережевої архітектури шляхом суворої ізоляції мікросервісів за допомогою технології [30] контейнеризації. Контейнер із системою керування базами даних розгортається у спеціально створеній ізольованій віртуальній мережі середовища Docker. Важливою архітектурною особливістю спроектованої інфраструктури є повна відсутність публікації (відкриття) портів бази даних на рівень операційної системи хоста або в глобальну мережу Інтернет. Доступ до порту бази даних дозволено виключно внутрішнім контейнерам основного бекенду, які перебувають у тій самій спільній віртуальній мережі. Такий підхід гарантує, що навіть у випадку теоретичної компрометації зовнішнього брандмауера або проксі-сервера, пряме мережеве звернення до бази даних ззовні буде фізично неможливим, оскільки вона надійно ізольована на рівні віртуального комутатора контейнерної інфраструктури.

Додатковим рівнем безпеки архітектури даних є реалізація принципу найменших привілеїв на рівні самої системи керування базами даних. Замість використання єдиного глобального суперкористувача для всіх потреб платформи, для кожного ізольованого мікросервісу створюється окремий логічний користувач із суворо обмеженими правами доступу. Такий підхід гарантує, що у випадку теоретичної компрометації одного з програмних компонентів, потенційний зломисник не зможе отримати доступ до даних інших підсистем або внести критичні зміни до глобальної структури баз даних. Кожен обліковий запис налаштовується виключно на взаємодію з конкретною логічною базою або схемою, яка належить відповідному мікросервісу, тим самим формуючи міцні та непроникні кордони безпеки безпосередньо на рівні сховища інформації.

Зокрема, для основного бекенд-сервісу, який відповідає за управління користувачами та життєвим циклом проєктів, створено спеціалізованого користувача з правами на виконання виключно базових операцій маніпулювання даними (читання, запис, оновлення та видалення) у межах

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ПЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				75
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

цільових таблиць. Аналогічна модель ізоляції застосовується для мікросервісу гейміфікації та економіки балів, де обліковий запис бази даних має доступ лише до таблиць фінансових транзакцій і глобального рейтингу, без фізичної можливості переглядати конфіденційні дані користувачів з основного ядра. Водночас для інтелектуального AI-асистента виділено окремий обліковий запис, який оптимізовано переважно для інтенсивного читання історичних даних та запису нових згенерованих карток ідей, що додатково захищає систему від несанкціонованої модифікації критичної бізнес-логіки з боку інтеграційних пайплайнів обробки природної мови.

Окрім експлуатаційних облікових записів, архітектура безпеки бази даних передбачає наявність спеціалізованих технічних користувачів для виконання адміністративних та аналітичних завдань. Для процесів автоматизованого розгортання та застосування міграцій виділено окремого користувача, який володіє розширеними правами на зміну структури таблиць (DDL-операції), проте ці облікові дані використовуються виключно конвеєром безперервної інтеграції на короткотривалому етапі оновлення системи і ніколи не передаються в робоче середовище додатків. Також для потреб резервного копіювання та побудови зведеної аналітики адміністраторами платформи налаштовано користувача з правами виключно на читання інформації. Такий багаторівневий розподіл ролей нівелює ризики деструктивного впливу програмних помилок на цілісність інформації та забезпечує повну відповідність системи сучасним стандартам безпеки розробки.

3.6 Тестування додатку

Для забезпечення високої надійності та стабільності роботи розробленої вебсистеми управління життєвим циклом інноваційних ідей було впроваджено практику автоматизованого модульного тестування серверної [19] частини. Реалізація тестів здійснюється на рівні фреймворку Symfony за допомогою спеціалізованого пакета інтеграції з середовищем PHPUnit. Основною метою модульного тестування є ізольована перевірка коректності виконання

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		76

ключових алгоритмів бізнес-логіки системи без необхідності взаємодії з реальною базою даних чи зовнішніми мережевими сервісами. У межах розробки краудсорсингової платформи було створено набір тестів, які автоматично перевіряють критичні процеси, зокрема логіку валідації вхідних даних при генерації нової картки ідеї, коректність нарахування рейтингових балів у модулі економіки балів та дотримання строгих правил рольової моделі при обробці заявок кандидатів на участь у команді.

Процес написання модульних тестів базується на активному використанні механізмів створення імітаційних об'єктів для безпечної підміни складних архітектурних залежностей. Наприклад, під час тестування сервісу трансформації неформатованого тексту в структурований проєкт, реальні звернення до програмного інтерфейсу великих мовних моделей повністю замінюються макетами з ізольованими тестовими даними. Це гарантує високу швидкість виконання тестового набору та виключає вплив сторонніх факторів на результати перевірки коду. Розроблений пакет модульних тестів Symfony органічно інтегровано в налаштований конвеєр безперервної інтеграції GitHub Actions. Завдяки цьому при кожній спробі внесення змін до вихідного коду система автоматично ініціює запуск усього тестового набору в ізольованому контейнері, ефективно запобігаючи потраплянню регресійних помилок до промислового середовища експлуатації.

Окрім ізольованого тестування окремих компонентів бізнес-логіки, важливою складовою перевірки працездатності програмного комплексу є тестування системи в умовах, максимально наближених до промислової експлуатації. Для забезпечення цього процесу на рівні серверної частини платформи було створено спеціалізовані класи фікстур, які відповідають за автоматизоване генерування та завантаження структурованих масивів імітаційних даних безпосередньо у реляційну базу даних. Цей інструментарій дозволяє програмним шляхом масово створювати репрезентативні тестові облікові записи з різними системними ролями, генерувати сотні карток

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				77
Змн.	Арк.	№ докум.	Підпис	Дата		

інноваційних ідей із прив'язаними завданнями та заявками, а також первинно заповнювати всі необхідні системні довідники категорій і факультетів. Такий архітектурний підхід повністю усуває необхідність ручного введення інформації для перевірки працездатності модулів платформи під час локальної розробки або після розгортання оновлень.

Наявність великого обсягу синтетичної тестової інформації є критично необхідною для комплексного оцінювання стабільності поведінки програмного забезпечення та коректності відтворення користувацького інтерфейсу. Завантаження згенерованих фікстур дає змогу розробникам ефективно тестувати продуктивність виконання запитів до бази даних, перевіряти точність алгоритмів багатокритеріального пошуку, фільтрації та механізмів пагінації у глобальному каталозі ідей під час імітації навантаження. Разом з тим, наповнена імітаційними даними система дозволяє достовірно перевірити правильність математичних розрахунків економіки балів, логіку просування користувачів у рейтингу та формування зведеної аналітичної статистики в панелі управління адміністратора. Процес управління фікстурами реалізовано через консольні команди оболонки, що робить розгортання повнофункціонального тестового середовища швидким та повністю автоматизованим етапом життєвого циклу розробки.

Невідомою частиною загальної стратегії контролю якості є проведення інтеграційного та наскрізного (End-to-End) тестування, яке дозволяє об'єктивно оцінити коректність взаємодії всіх незалежних підсистем. Зважаючи на розподілену архітектуру платформи з чітким відокремленням серверного ядра від клієнтського інтерфейсу, принципово важливим стало підтвердження безперебійного виконання повноцінних користувацьких сценаріїв у реальних умовах.

Завдяки використанню інструментів автоматизації браузерних перевірок здійснюється валідація ключових маршрутів: починаючи від етапу реєстрації та поетапної комунікації з інтелектуальним асистентом для

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ІЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		78

структурування концепту, і закінчуючи процесами затвердження учасників команди та динамічним оновленням рейтингової статистики. Окремий фокус у межах цього процесу спрямовано на перевірку стабільності з'єднання з пайплайнами обробки природної мови, що дозволяє виміряти здатність системи коректно реагувати на складні запити в умовах можливих мережових затримок. Застосування такого багаторівневого підходу до тестування створює міцне підґрунтя для безпечної експлуатації вебсистеми та ефективно мінімізує ризики виникнення критичних збоїв під час реального використання.

Окрему увагу в процесі забезпечення якості було приділено функціональному тестуванню програмних інтерфейсів (API). Для цього використовується спеціалізований базовий клас `ApiTestCase` від інструментарію `API Platform`, який дозволяє імітувати реальні HTTP-запити до серверної частини без розгортання повноцінного вебсервера. Тести цього рівня виконують запити типів GET, POST або PUT до кінцевих точок системи (зокрема, для отримання історії змін завдань, створення ідей чи фільтрації коментарів) та здійснюють комплексну валідацію отриманих відповідей у форматі JSON-LD / Hydra. Автоматично перевіряється відповідність HTTP-статусів очікуваним значенням, структура і типи полів у тілі відповіді, а також коректність роботи механізмів пагінації та вбудованих фільтрів на рівні колекцій ресурсу. Важливим аспектом функціонального тестування вебсистеми є верифікація політик безпеки та моделі розмежування доступу. Перевірка захищеності ендпоінтів реалізується через програмну генерацію автентифікаційних токенів за допомогою сервісу `JWT (lexik_jwt_authentication)`. Під час виконання тестів динамічно створюються імітаційні профілі користувачів із різними рівнями привілеїв (наприклад, авторизовані учасники проєкту та сторонні користувачі), після чого згенерований Bearer-токен додається в HTTP-заголовки авторизованого клієнта. Це дозволяє надійно протестувати правила доступу (Voters / Security правила): тести верифікують, що авторизовані члени команди мають повний

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ІЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		79

доступ до історії логування чи управління ідеєю, тоді як запити від сторонніх осіб повертають порожні вибірки або блокуються на рівні системи контролю доступу. Технічна інфраструктура середовища тестування побудована на засадах повної ізоляції даних та повторюваності результатів. Для запуску тестового набору через утиліту PHPUnit використовується окрема конфігурація середовища (APP_ENV=test) та виділена тестова база даних, що унеможливорює пошкодження чи змішування робочих даних процесу розробки. Ініціалізація компонентів перед кожним запуском забезпечується через спеціалізований скрипт завантаження (bootstrap.php), який динамічно підключає автозавантажувач класів, зчитує конфігураційні параметри та ініціює ядро застосунку (Kernel). Весь життєвий цикл виконання тестів — від налаштування схеми бази даних до запуску ізольованих перевірок та очищення контейнера залежностей після завершення методів — виконується всередині Docker-контейнерів, що гарантує стабільність тестового оточення як на локальній машині розробника, так і в конвеєрі безперервної інтеграції.

Висновки до третього розділу

У розділі розглянуто архітектуру інтерфейсу, питання кібербезпеки та порядок взаємодії користувачів із розробленою вебсистемою управління життєвим циклом інноваційних ідей та формування команд. Програмний комплекс реалізовано за моделлю клієнт-сервер, де клієнтська та серверна частини мають чіткий розподіл обов'язків та взаємодіють між собою через REST API. Основну увагу приділено опису маршрутизації, побудованої за ресурсним принципом із підтримкою типових HTTP-методів, що забезпечує зрозумілу і масштабовану логіку обробки запитів для управління проєктними ініціативами, управлінням заявками кандидатів та користувацькими профілями.

На рівні клієнтського додатка використано бібліотеку React [16], яка дозволяє реалізувати компонентну архітектуру інтерфейсу з логічним поділом на ізольовані функціональні блоки з використанням суворої типізації [35]. Компоненти відповідають окремим підсистемам платформи: глобальному

		А.В. ВОЗНІЮК			ІПЗ – 121 – 26 – ПЗ	Арк.
		А.В. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		80

каталогу ідей, формам генерації контенту, панелі адміністратора та інтерактивному простору комунікації з ІІІ-асистентом. Уся навігація здійснюється за допомогою інструментарію React Router, що синхронізується зі структурою бекенду, дозволяючи зберігати єдиний підхід до управління маршрутами та спрощувати підтримку коду при подальшому масштабуванні краудсорсингової системи.

Окремо розглянуто та впроваджено комплексні заходи безпеки, що включають механізм автентифікації користувачів за допомогою алгоритмів JWT. Цей підхід дозволяє здійснювати безстейтову перевірку рівнів доступу стейкхолдерів та звичайних учасників без необхідності зберігати сесії на сервері. Крім того, деталізовано архітектурні рішення щодо ізоляції бази даних на рівні віртуальної мережі контейнерів та безпечного управління конфігураційними секретами через системи Кеерер і механізми конвеєрів безперервної інтеграції, що мінімізує ризики компрометації конфіденційної інформації.

Для підтвердження високої надійності та стабільності роботи бізнес-логіки системи у розділі описано процеси автоматизованого тестування. Впровадження модульних тестів на рівні серверного фреймворку у поєднанні з використанням класів фікстур для масової генерації імітаційних даних дозволило ефективно перевірити працездатність алгоритмів економіки балів, рольової моделі та багатокритеріального пошуку. Це забезпечило комплексну підготовку платформи до умов експлуатації з реальними навантаженнями та гарантувало відсутність регресійних помилок.

Загалом, побудований користувацький інтерфейс у поєднанні з правильно структурованою маршрутизацією, захищеним бекендом та сучасним механізмом автентифікації формує стабільну платформу для спільної роботи та реалізації стартапів. Інтерфейс адаптований під різні типи пристроїв завдяки застосуванню принципів адаптивного вебдизайну, що робить роботу зі складними формами та інтелектуальним помічником

		<i>А.В. ВОЗНІЮК</i>			ІІІ – 121 – 26 – ІІІ	Арк.
		<i>А.В. МОРОЗОВ</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		81

зручною навіть на мобільних екранах. Обрана архітектура [31] забезпечує зрозумілий порядок використання функціоналу, гнучкість розширення можливостей системи та цілком відповідає сучасним вимогам до високонавантажених платформ управління проєктами.

ВИСНОВКИ

В межах дипломної роботи було успішно спроектовано та реалізовано повнофункціональну вебсистему управління життєвим циклом інноваційних ідей та формування команд. Програмний комплекс побудовано на основі сучасного стеку технологій, зокрема фреймворків Symfony та NestJS для розробки серверної частини, а також бібліотеки React для побудови клієнтського інтерфейсу. Завдяки застосуванню принципів REST-архітектури забезпечено чітке розмежування між фронтендом та бекендом, що суттєво підвищує масштабованість, загальну продуктивність і зручність подальшої підтримки краудсорсингової платформи.

З боку серверної частини реалізовано ефективну взаємодію з реляційною базою даних із дотриманням принципів чистої архітектури [34] та об'єктно-реляційного відображення [18]. Це надало змогу оптимізувати процеси створення, збереження та оновлення складних інформаційних сутностей, таких як розгорнуті картки проєктів, заявки кандидатів та системні довідники адміністратора. Розподіл бізнес-логіки на контролери та ізольовані сервіси, а також впровадження надійних механізмів безпеки, включно з JWT-автентифікацією, управлінням секретами через систему Keeper та мережевою ізоляцією за допомогою технології контейнеризації Docker, гарантують високий рівень захищеності конфіденційних даних стейкхолдерів.

Користувацький інтерфейс забезпечує швидко, динамічну та інтуїтивно зрозумілу взаємодію учасників із програмним комплексом на різних типах пристроїв. Під час розробки були успішно імплементовані всі ключові функціональні вимоги: розширене управління профілями та налаштуваннями приватності, багатокритеріальний пошук у глобальному каталозі ініціатив,

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ІЗ	Арк.
		<i>А.В. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		82

процеси подачі та обробки заявок на участь, а також алгоритми економіки балів та глобального рейтингу. Вагомим інноваційним досягненням стала наскрізна інтеграція інтелектуального ШІ-асистента, який функціонує на базі великих мовних моделей та здатен через діалог здійснювати автоматизоване структурування неформатованого тексту в готові проєктні пропозиції, значно знижуючи когнітивне навантаження на авторів.

Для забезпечення безкомпромісної стабільності та надійності роботи системи було застосовано комплексний підхід до тестування та розгортання. Використання модульних тестів у поєднанні зі спеціалізованими класами фікстур дозволило автоматизовано перевіряти критичні алгоритми рольової моделі та управління ризиками на репрезентативних масивах даних. Налаштування конвєсів безперервної інтеграції та доставки, а також розгортання цільового ізольованого середовища на базі технології апаратної віртуалізації KVM підтверджують високу технологічну зрілість проєкту. У підсумку, результати виконання роботи доводять доцільність обраних архітектурних рішень та дозволяють розглядати розроблену платформу як готовий продукт для впровадження в реальне середовище з метою стимулювання та реалізації інноваційних стартап-ініціатив.

		<i>A.B. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ІЗ	Арк.
		<i>A.B. МОРОЗОВ</i>				
Змн.	Арк.	№ докум.	Підпис	Дата		83

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Attention Is All You Need / A. Vaswani та ін. NeurIPS Foundation. 2017. Т. 30. С. 5998–6000. URL: <https://doi.org/10.48550/arXiv.1706.03762>.
2. Microservices: yesterday, today, and tomorrow. / N. Dragoni та ін. Present and Ulterior Software Engineering. 2017. С. 195–216. URL: https://doi.org/10.1007/978-3-319-67425-4_12.
3. Language Models are Few-Shot Learners. / T. Brown та ін. Advances in Neural Information Processing Systems. 2020. Т. 33. С. 1877–1901. URL: <https://doi.org/10.48550/arXiv.2005.14165>.
4. Gamified crowdsourcing: Conceptualization, literature review, and future agenda. / B. Morschheuser та ін. International Journal of Human-Computer Studies. 2017. Т. 106. С. 26–43. URL: <https://doi.org/10.1016/j.ijhcs.2017.04.005>.
5. Overcoming Security Challenges in Microservice Architectures. / T. Yarygina, A. H. Bagge. 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). 2018. С. 11–20. URL: <https://doi.org/10.1109/SOSE.2018.00011>.
6. The Rise and Potential of Large Language Model Based Agents: A Survey. / Z. Xi та ін. Frontiers of Computer Science. 2023. Т. 18, № 1. С. 1–35. URL: <https://doi.org/10.48550/arXiv.2309.07864>.
7. Principled design of the modern Web architecture. / R. T. Fielding, R. N. Taylor. ACM Transactions on Internet Technology. 2002. Т. 2, № 2. С. 115–150. URL: <https://doi.org/10.1145/514183.514185>.
8. Restful web services vs. "big" web services: making the right architectural decision. / C. Pautasso, O. Zimmermann, F. Leymann. Proceedings of the 17th International Conference on World Wide Web. 2008. С. 805–814. URL: <https://doi.org/10.1145/1367497.1367606>.

9. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. / A. Balalaie, A. Heydarnoori, P. Jamshidi. IEEE Software. 2016. T. 33, № 3. C. 42–52. URL: <https://doi.org/10.1109/MS.2016.64>.
10. Software architecture: a travelogue. / D. Garlan. Proceedings of the Future of Software Engineering. 2014. C. 29–39. URL: <https://doi.org/10.1145/2593882.2593886>.
11. Guidelines for Human-AI Interaction. / S. Amershi та ін. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 2019. C. 1–13. URL: <https://doi.org/10.1145/3290605.3300233>.
12. Two's company, three's a crowd: a case study of crowdsourcing software development. / K. J. Stol, B. Fitzgerald. Proceedings of the 36th International Conference on Software Engineering. 2014. C. 187–198. URL: <https://doi.org/10.1145/2568225.2568249>.
13. Docker Inc. Docker Overview. Docker Documentation. 2023. URL: <https://docs.docker.com/get-started/overview/>
14. Amazon Web Services. Amazon S3 Security Best Practices. AWS Documentation. 2023. URL: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-best-practices.html>
15. LangChain. LangGraph: Multi-Actor, Stateful LLM Applications. LangChain Blog. 2024. URL: <https://blog.langchain.dev/langgraph/>
16. Meta. React: The Library for Web and Native User Interfaces. React Documentation. 2023. URL: <https://react.dev/>
17. GitHub. Using GitHub Actions for CI/CD. GitHub Docs. 2023. URL: <https://docs.github.com/en/actions>
18. Microsoft. Entity Framework Core Documentation. 2023. URL: <https://learn.microsoft.com/en-us/ef/core/>
19. Symfony SAS. Symfony 7.0 Documentation. 2024. URL: <https://symfony.com/doc/current/index.html>

		A.B. БОЗНІОК			ІІЗ – 121 – 26 – ІІЗ	Арк.
		A.B. МОРОЗОВ				85
Змн.	Арк.	№ докум.	Підпис	Дата		

- 20.NestJS. NestJS Progressive Framework. 2024. URL: <https://docs.nestjs.com/>
- 21.MySQL. MySQL 8.0 Reference Manual. 2024. URL: <https://dev.mysql.com/doc/refman/8.0/en/>
- 22.Google. Gemini API Developer Documentation. 2024. URL: <https://ai.google.dev/docs>
- 23.Auth0. The JWT Handbook. 2023. URL: <https://auth0.com/resources/ebooks/jwt-handbook>
- 24.Red Hat. Managing Virtualization with KVM. 2023. URL: <https://docs.redhat.com/>
- 25.Potencier F. Symfony 5: The Fast Track. Paris : Symfony SAS, 2021. 360 p.
- 26.Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. Sebastopol : O'Reilly Media, 2020. 338 p.
- 27.Poulton N. Docker Deep Dive. London : Leanpub, 2021. 302 p.
- 28.DuBois P. MySQL. 5th ed. Boston : Addison-Wesley Professional, 2013. 1248 p.
- 29.Tunio M. A. Nest.js: A Progressive Node.js Framework. Birmingham: Packt Publishing, 2023. 412 p.
- 30.Newman S. Building Microservices. 2nd ed. Sebastopol : O'Reilly Media, 2021. 542 p.
- 31.Fowler M. Refactoring: Improving the Design of Existing Code. 2nd ed. Boston : Addison-Wesley, 2018. 448 p.
- 32.Richardson C. Microservices Patterns. Shelter Island : Manning Publications, 2018. 520 p.
- 33.Grinberg M. Flask Web Development. 2nd ed. Sebastopol : O'Reilly Media, 2018. 430 p.
- 34.Martin R. C. Clean Architecture: A Craftsman's Guide. Boston : Prentice Hall, 2017. 432 p.
- 35.Vanderkam D. Effective TypeScript. Sebastopol : O'Reilly Media, 2019. 312 p.

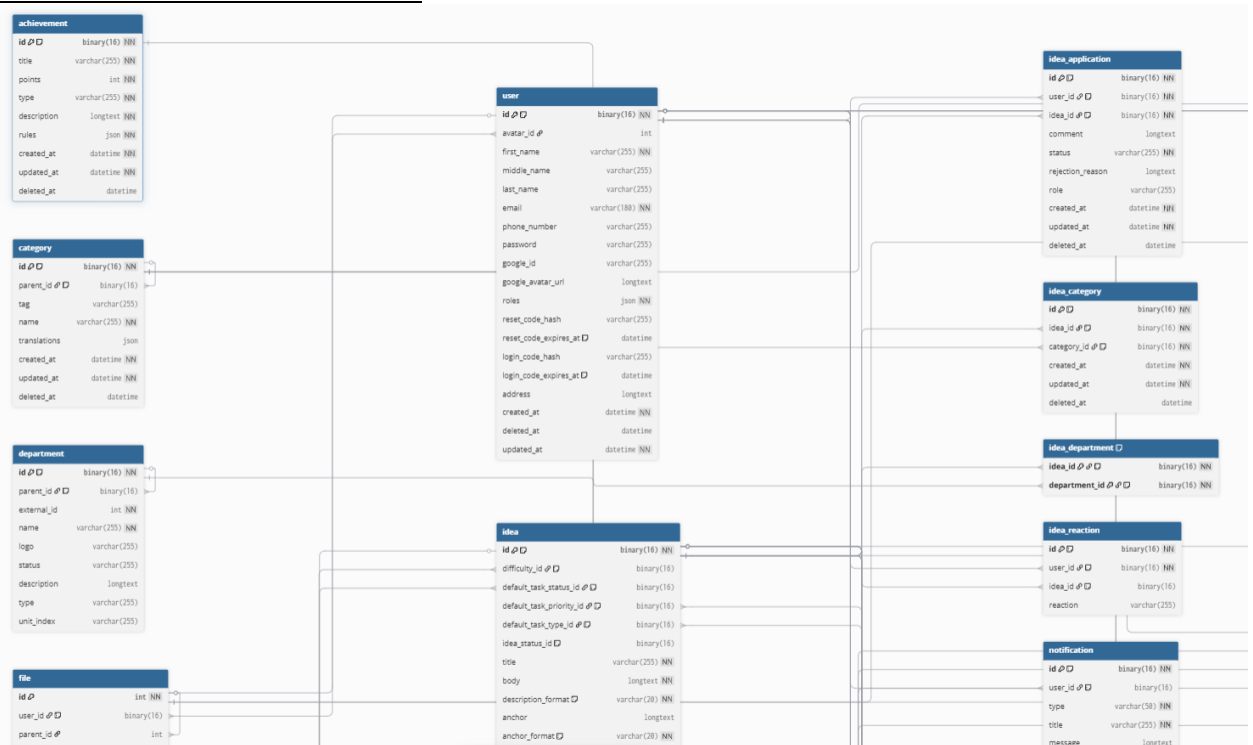
		A.B. БОЗНІОК			ІІІЗ – 121 – 26 – ІІЗ	Арк.
		A.B. МОРОЗОВ				
Змн.	Арк.	№ докум.	Підпис	Дата		86

ДОДАТКИ

		<i>А.В. ВОЗНІЮК</i>			ІПЗ – 121 – 26 – ІЗ	<i>Арк.</i>
		<i>А.В. МОРОЗОВ</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		87

Додаток А

Діаграма бази даних: <https://dbdiagram.io/d/Idea-Bank-6a271baa25fc5bf036bd7f2b>



Додаток Б

name: Deploy to Server

on:

push:
branches:
- main

jobs:

deploy:
runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- name: Deploy via SSH
uses: appleboy/ssh-action@v1.0.3

env:

ENV_DATA: \${ secrets.PROD_ENV_FILE }
API_ENV_DATA: \${ secrets.PROD_API_ENV_FILE }
CHAT_ENV_DATA: \${ secrets.PROD_CHAT_ENV_FILE }

with:

host: \${ secrets.SERVER_HOST }
username: \${ secrets.SERVER_USER }
key: \${ secrets.SSH_PRIVATE_KEY }

		A.B. BOZHIOK			ІІЗ – 121 – 26 – ІІЗ	Арк.
		A.B. МОРОЗОВ				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

passphrase: ${ secrets.SSH_PASSPHRASE }}
proxy_host: ${ secrets.PROXY_HOST }}
proxy_username: ${ secrets.PROXY_USER }}
proxy_key: ${ secrets.PROXY_KEY }}
proxy_passphrase: ${ secrets.PROXY_PASSPHRASE }}
script_stop: true
command_timeout: 120m
envs: ENV_DATA,API_ENV_DATA,CHAT_ENV_DATA
script: |
  cd /home/ideabank/IB
  git fetch origin main
  git reset --hard origin/main
  echo "$ENV_DATA" > .env
  echo "$API_ENV_DATA" > api/.env
  echo "$CHAT_ENV_DATA" > chat-service/.env
  docker compose -f docker-compose.prod.yml build
  docker compose -f docker-compose.prod.yml up -d
  docker image prune -f

```

Додаток В

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IdeaBankS3Access",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::idea-bank-document-storage",
        "arn:aws:s3:::idea-bank-document-storage/*"
      ]
    }
  ]
}

```

		A.B. ВОЗНІЮК			ІІЗ – 121 – 26 – ІІЗ	Арк.
		A.B. МОРОЗОВ				89
Змн.	Арк.	№ докум.	Підпис	Дата		