Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно–інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Перший (бакалаврський)
(рівень вищої освіти)

Розроблення консалтингової системи для виробництва типу Lean production
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТАКІТ–21–1
ЗЕЛЕНЬКО В.І.
(власне ім'я, прізвище)

Спеціальності 151 Автоматизація, комп`ютерно–інтегровані технології та робототехніка
(код і повна назва спеціальності)

Тип програми освітньо–професійна
(освітньо–професійна або освітньо–наукова)

Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)

Керівник проф. СЕЗОНОВА І.К.
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Зеленько Вадим Ігорович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

"19" червня 2025 р Зеленько В.І.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація, комп`ютерно–інтегровані технології
Тип програми Освітньо–професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«28» квітня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Зеленько Вадиму Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення консалтингової системи для виробництва типу Lean production
Затверджена наказом по університету від 19.05.2025 №390 Ст.
2. Термін подання здобувачем роботи до екзаменаційної комісії 09.06.2025
3. Вихідні дані до роботи
Розробити консалтингову систему для виробництва типу Lean production, яка складатиметься з серверної та клієнтськох частини. Для розробки даного продукту використати мови програмування JavaScript, TypeScript, програмна платформа Node.js, бібліотеки React, Refine та MIU. БД MongoDB

4. Перелік питань, що потрібно опрацювати в роботі
Вступ.
Аналіз предметної області.
Формування вимог до програмної системи.
Архітектура та проектування програмного забезпечення.
Опис розроблених програмних рішень.
Тестування розробленого програмного забезпечення.
Висновки.
Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій _____
Демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) – 10 с. формату А4.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт–ка
1	Аналіз технічного завдання	28.04– 04.05.2025	Виконано
2	Створення специфікації ПЗ	05.05–10.05.2025	Виконано
3	Проектування ПЗ	11.05–17.05.2025	Виконано
4	Розробка ПЗ	18.05–20.05.2025	Виконано
5	Тестування ПЗ	21.05–25.05.2025	Виконано
6	Подання роботи на перевірку Інтернет-сервісом StrikePlagiarism	26.05–31.05.2025	Виконано
7	Оформлення пояснювальної записки	26.05–31.05.2025	Виконано
8	Підготовка презентації та доповіді	01.06–03.06.2025	Виконано
9	Подання роботи на рецензію	04.06–08.06.2025	Виконано
10	Подання роботи на підпис зав. кафедри	09.06–11.06.2025	Виконано
11	Подання кваліфікаційної роботи в ЕК	12.06–16.06.2025	Виконано

Дата видачі завдання 28.04.2025

Здобувач _____ ЗЕЛЕНЬКО В. І.
(підпис) (посада, власне ім'я, прізвище)

Керівник роботи _____ проф. СЕЗОНОВА І. К.
(підпис) (власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 63 с., 22 рис., 26 джерел.

LEAN PRODUCTION, ЦИФРОВІЗАЦІЯ, KANBAN, 5S, TYPESCRIPT, JAVASCRIPT, MONGODB, КОНСАЛТИНГОВА СИСТЕМА, REACT.

Мета роботи – розробити автоматизовану консалтингову систему для покращення ефективності виробничих процесів, яка забезпечує комплексну підтримку аналізу діяльності підприємства за принципами Lean Production. Система має дозволяти виявляти втрати та неефективності, формувати обґрунтовані пропозиції щодо їх усунення, а також забезпечувати інтерактивний вебінтерфейс для зручної взаємодії користувачів з Lean-інструментами. Розробка має сприяти підвищенню продуктивності, зниженню витрат і впровадженню культури постійного вдосконалення на підприємстві.

Об’єкт розробки – процес управління Lean-проєктами на виробничому підприємстві.

Предмет розробки – консалтингова система для виробництва типу Lean production.

В роботі описано процес створення інформаційної системи, яка допомагає підприємствам організовувати покращення на основі Lean-підходів. Реалізовано модулі Lean Canvas, Kanban, 5S, систему ролей користувачів, панель проєктів, шаблони звітів та інструменти аналіз. Система побудована з використанням сучасного вебстека: React, Refine, MUI, Node.js та MongoDB. Присутня аутентифікацію через Google OAuth, управління доступом, генерацію PDF-звітів і зберігання даних.

Ціль роботи була досягнута: розроблена система успішно реалізує функції Lean-аналізу та підтримує покращення виробничих процесів.

ABSTRACT

Explanatory note: 63 pp., 22 fig., 26sources.

LEAN PRODUCTION, DIGITALIZATION, KANBAN, 5S, TYPESCRIPT, JAVASCRIPT, MONGODB, CONSULTING SYSTEM, REACT.

Objective of the Thesis – to develop an automated consulting system aimed at improving the efficiency of production processes by providing comprehensive support for enterprise activity analysis based on Lean Production principles. The system should enable the identification of losses and inefficiencies, generate well-founded recommendations for their elimination, and offer an interactive web interface for convenient user interaction with Lean tools. The development is intended to enhance productivity, reduce costs, and promote a culture of continuous improvement within the enterprise.

Object of the Study – the production system of a Lean-type enterprise.

The thesis describes the development process of an information system that helps enterprises organize improvements based on Lean principles. The system includes modules such as Lean Canvas, Kanban, 5S, user role management, project dashboard, report templates, and analytical tools.

The solution is built using a modern web stack: React, Refine, MUI, Node.js, and MongoDB. It features Google OAuth authentication, access control, PDF report generation, and data storage.

The objective of the thesis was achieved: the developed system successfully implements Lean analysis functions and supports the improvement of production processes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – База даних;

ПЗ – Програмне забезпечення;

ПК – Персоналий комп’ютер;

AI – Artificial Intelligence;

API – Application Programming Interface;

CRUD – створення, читання, оновлення, видалення;

CSRF – Cross-Site Request Forgery;

ERP – Enterprise Resource Planning;

JSON – JavaScript Object Notation;

MUI – Material User Interface;

SQL – Structured Query Language;

UML – Unified Modeling Language;

UI – User Interface;

UX – User Experience;

VSM – Value Stream Mapping;

XSS – Cross-Site Scripting.

ВСТУП

Робота присвячена актуальному питанню для виробничих процесів та підвищенню ефективності управління підприємством при впровадженні принципів Lean Production. Lean-підхід дедалі частіше використовується у різних галузях як стратегія мінімізації втрат, підвищення якості та оптимізації внутрішніх процесів.

У сучасних умовах розвитку промисловості дуже важливо стає створення автоматизованої підтримки прийняття рішень, які допомагають впроваджувати та контролювати різні проєкти.

Об’єкт розробки – процес управління Lean-проєктами на виробничому підприємстві.

Предмет розробки – консалтингова система для виробництва типу Lean production.

Метою роботи є створення комп’ютерної консалтингової системи на основі підходу Lean Production, яка надає підприємствам інструменти для планування, впровадження та аналізу покращень у виробничих процесах. Система реалізує ключові модулі командної взаємодії, включаючи Lean Canvas для стратегічного планування, Kanban для управління завданнями, 5S для оцінки організованості робочого середовища, а також панель управління проєктами й засоби збору аналітичних даних з подальшим формуванням рекомендацій. Розробка спрямована на підтримку культури постійного вдосконалення, підвищення ефективності та конкурентоспроможності підприємства.

Розробка реалізована з використанням сучасного вебстека технологій, React, TypeScript, JavaScript, MongoDB, Node.js, Refine та MUI. Реалізовано аутентифікацію через Google OAuth, керування правами доступу, зберігання даних у хмарі та автоматичну генерацію звітів. Система має зрозумілий та стильний інтерфейс, гнучку архітектуру та можливість масштабування під потреби будь яких підприємств.

Платформа може бути ефективно застосована в організаціях для впровадження Lean-підходів, навчання персоналу, підтримки безперервного вдосконалення та досягнення стратегічних цілей.

Питання впровадження такої технології є надзвичайно важливим у різних сферах життя та діяльності людей. У сучасних умовах конкуренції організації прагнуть зменшити витрати, підвищити якість продукції й ефективність внутрішніх процесів, тому що це головне в будь якій галузі. Саме Lean-підхід дозволяє виявляти та усувати втрати, оптимізувати робочі процеси й забезпечити це ідеально.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести огляд існуючих цифрових рішень у даній сфері;

– дослідити принципи роботи основних інструментів Lean;

– реалізувати логіку взаємодії між користувачами з різними ролям;

– розробити алгоритми роботи програми;

– оформити пояснювальну записку згідно з виммогами ДСТУ 3008:2015 [1], методичних вказівок з підготовки кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти відповідної спеціальності 151 Автоматизація, комп`ютерно–інтегровані технології освітньої програми «Автоматизація, комп`ютерно–інтегровані технології» [2].

Тема кваліфікаційної роботи відповідає Цілі сталого розвитку №8, яка передбачає забезпечення гідної праці для всіх та стимулювання стійкого економічного зростання шляхом створення продуктивної зайнятості й інклюзивного розвитку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз Lean Production

Перед тим як почати розробку будь-якого програмного застоосунку чи системи, дуже важливо розуміти головний її принцип.

Якщо простими словами то, Lean Production – це спеціальна концепція або набір правил для того щоб зменшити витрати та збільшити якість продукту. Перші згадки про це були ще за часів Генрі Форда в 1919 році, але в 1948 Едвард Демінг устаткував це для навчання японців управління якістю.

Комп'ютерна консалтингова система – це програмна платформа, яка має в собі багато шарів, які працюють разом, включає серверну, клієнтську частину. Така система призначена для підтримки рішень у виробничому середовищі, орієнтованому на принципи Lean [3].

Lean production, або «ощадливе виробництво», є філософією управління, яка розвивалася в Японії (насамперед у компанії Toyota) і була зосереджена на максимальному задоволенні потреб замовника при мінімальних витратах ресурсів. У сучасному світі ця модель усе частіше впроваджується не тільки у великих промислових корпораціях, а й на малих і середніх підприємствах, що прагнуть до кращого [4-5].

Програмна система для Lean-консалтингу у виробничому середовищі є комплексною сукупністю програмних модулів, до складу яких входять серверна частина, клієнтський веб інтерфейс.

Клієнтська частина системи повинна передбачати окремі інтерфейси для менеджерів, і звичайних працівників. Це необхідно для того, щоб кожна категорія користувачів мала доступ лише до свого функціоналу й могла легко працювати в межах тільки своїх обов'язків.

Менеджер у такій системі відповідає за перегляд зібраної статистики, виявлення відхилень, генерацію Lean-звітів. Також він виконує контроль за вибором працівників на проекті.

Працівники на виробництві можуть вносити свої пропозиції щодо вдосконалення, повідомляти про помічені проблеми або оцінювати рішення.

Система має забезпечувати кожній ролі доступ до інтуїтивного і зручного інтерфейсу, що дозволяє швидко взаємодіяти з даними, приймати рішення на основі генерованих звітів і створювати повноціннне безперервне вдосконалення.

Для формулювання чіткої мети створення будь якої комп'ютерної системи, та також Lean-консалтингу, насамперед слід ураховувати цільову аудиторію. У контексті виробництва, що прагне впроваджувати принципи ощадливого виробництва, ключовими користувачами такої системи виступають аналітики, керівники готових підрозділів або Lean-координатори, а також лінійний персонал, залучений до того або іншого проєкта.

Таким чином, цільова аудиторія системи охоплює як управлінський склад, так і безпосередніх учасників виробничого процесу. Передбачається, що більшість користувачів будуть мати базовий досвід в роботі з комп'ютером, бо ефективне використання системи базується на розумінні принципів Lean та здатності працювати з базовими елементами інтерфейсу.

Основне призначення консалтингової системи – створити єдину цифрову платформу для підтримки Lean-ініціатив, яка дозволить оптимізувати внутрішні процеси підприємства, зменшити витрати, покращити якість і забезпечити сталий розвиток.

Система повинна забезпечити зручний механізм введення, редагування та перегляду інформації щодо поточних ініціатив, а також спрощувати обробку заявок на нові пропозиції вдосконалення.

Інформаційні потреби системи охоплюють як передачу, так і обробку даних між різними рівнями управління. Наприклад, керівники змін повинні мати змогу бачити поточні ключові показники виробничої лінії, аналітики – отримувати повний зріз даних у вигляді дашбордів, а працівники – швидко заповнювати картки спостережень чи повідомляти про порушення стандартів. Усі ці дії реалізуються через адаптований вебінтерфейс, доступний із комп'ютера або мобільного пристрою.

1.2 Аналіз предметної галузі

У рамках дослідження розглянуто платформу iObeya – яка орієнтована на рішення для управління процесами на підприємстві, відповідно до принципів Lean та Agile (рис. 1.1). На сторінці сайту зроблено акцент саме на цифровізації нарад, карт потоку цінності (VSM) та Kanban-дошок. Такий підхід є актуальним для компаній, що впроваджують Lean-культуру та прагнуть забезпечити єдине інформаційне середовище для комунікації між підрозділами, незалежно від мысця знаходження працівників [6].

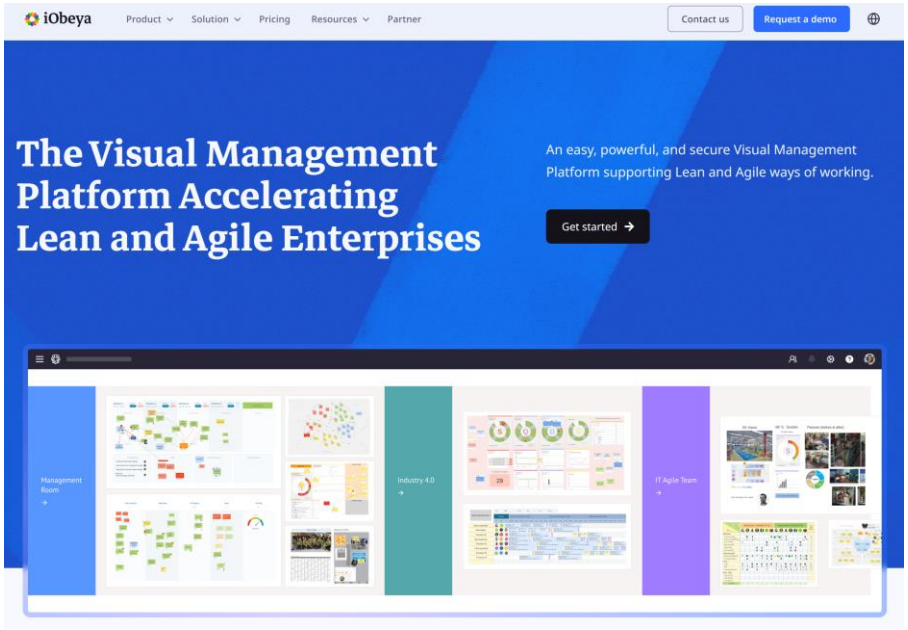

Рисунок 1.1 – Сторінка з веб-сайту https://www.iobeya.com

Навігація сайту інтуїтивна – у шапці доступні розділи з описом функцій, типів рішень для різних команд (виробничі, аналітичні, управлінські), галузевих кейсів, а також можливістю замовлення демонстрації [7]. Інтерфейс платформи iObeya містить гнучкі віртуальні кімнати, де кожна команда працює зі звичними для себе інструментами (рис. 1.2).

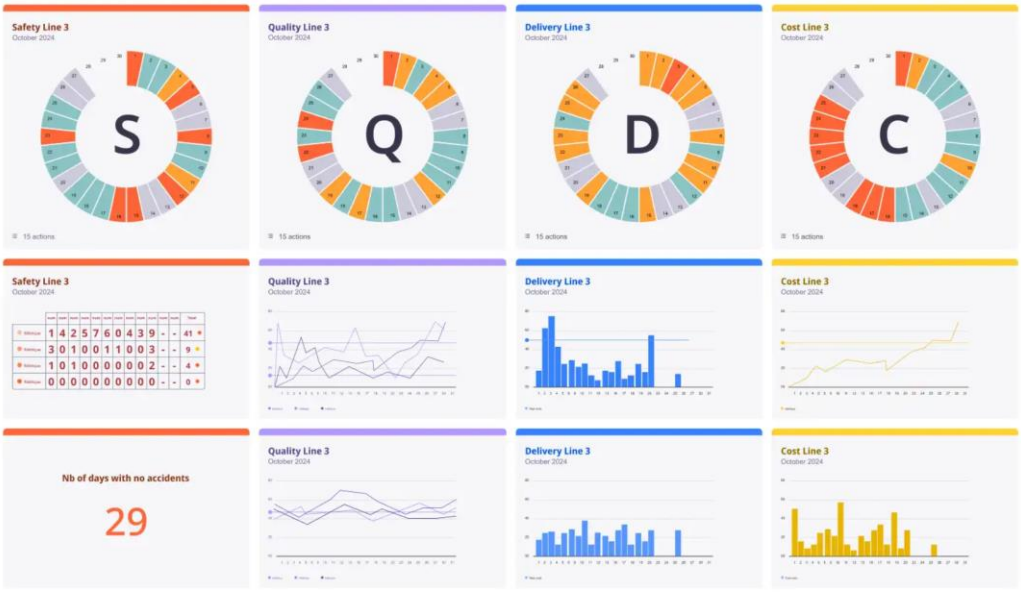

Рисунок 1.2 – Розділ Lean Production

В окремому блоці представлено галерею шаблонів, серед яких доступні 5S, Root Cause Analysis, A3 Thinking, Problem Solving Boards та інші візуальні елементи, які можуть бути кастомізовані під потреби конкретного підприємства (рис. 1.3). Це дозволяє командам проводити зустрічі без використання паперових документів, з можливістю перегляду змін у режимі реального часу, що суттєво підвищує продуктивність обговорень і зменшує бюрократію.

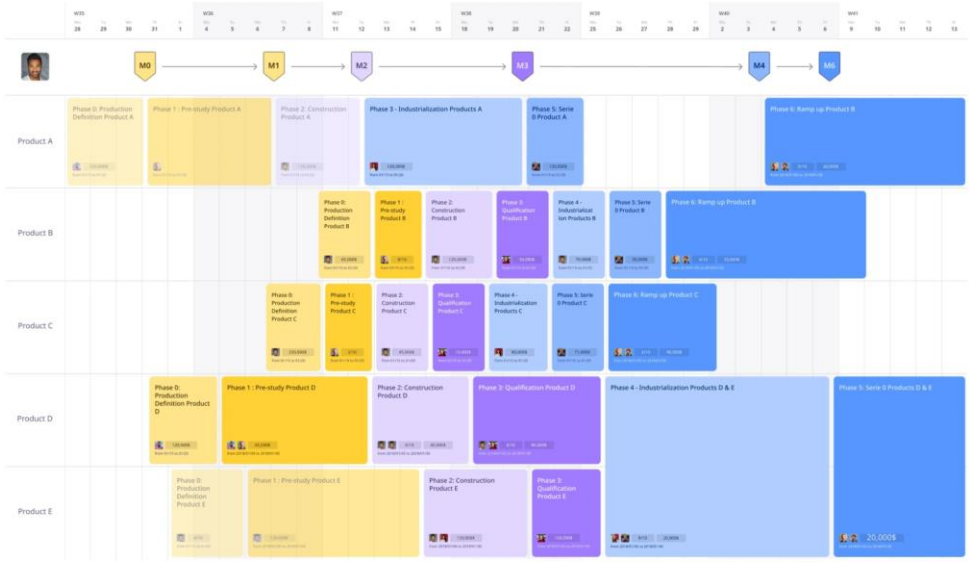

Рисунок 1.3 – Kanban дошка

Переваги цієї системи полягають у тому, що вона:

– забезпечує централізовану платформу для Lean-комунікації між командою;

– підтримує широкий спектр візуальних інструментів, які можна адаптувати під будь-яку виробничу задачу;

– дає змогу автоматично генерувати звіти та вести аналітику.

Серед недоліків слід зазначити:

– складну інтеграцію з внутрішніми ERP-системами, особливо для малих підприємств без IT-відділу;

– закриту архітектуру, яка не дозволяє вільно розширювати функціональність або інтегрувати нестандартні модулі;

– високу вартість ліцензії, що робить систему малодоступною для малого бізнесу.

Після аналізу можна дійти висновку, що для підвищення конкурентоспроможності вітчизняного рішення доцільно поєднати кращі функції подібних платформ, спростити користувацький досвід та передбачити гнучку адаптацію під різні масштаби виробництва.

2 ПЛАНУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Головною метою розробки системи є створення такого програмного забезпечення яке забезпечить підтримку процесів Lean-консалтингу у виробничому середовищі. Система повинна забезпечувати користувачам засоби для створення, ведення, перегляду та аналізу Lean-проєктів, а також надавати повний набір цифрових інструментів для роботи з виробництвом [3-5].

Програмна система повинна виконувати такі завдання для користувачів:

– створення та редагування Lean-проєктів;
– заповнення шаблонів (5S, Kanban, Lean Canvas);
– збереження, перегляд і управління проєктною інформацією;
– додавання учасників;
– призначення ролей;
– перегляд рекомендацій фінального звіту про реалізацію покращення;
– перегляд чеклісту успішного впровадження Lean на підприємстві.

Для адміністраторів (аналітиків або Lean-консультантів) система повинна надавати:

– повний контроль за управлінням користувачами, проєктами, шаблонами та аналітичними модулями;
– інструменти для оцінки ефективності реалізованих заходів;
– можливість формування експортованих звітів у форматах PDF та Excel для подальшого аналізу.

2.2 Нефункціональні вимоги

Якість системи завжди іде поряд з фунціоналом, тому зручніть та ефективна робота повинні бути на вищому рівні.

Продукт має включати:

– швидке оброблення запитів, навіть при одночасній роботі великох кількості користувачів;

– відгук інтерфуйсу повинен бути миттєвим, а операції з БД оптимізованими;

– безпека від несанкціонованого доступу. Захист від атак типу SQL-ін’єкцій, XSS та CSRF є обов’язковим [8];

– архітектура має дозволяти легке додавання нових функцій та модулів безпорушення існуючого функціоналу. Коректна робота при зростанні кількості користувасів і обсягу даних;

– система повинна працювати стабільно без збоєв і забезпечуватизбереження даних навіть у разі апаратних або програмних збоїв;

– інтерфейс має бути інтуїтивно зрозумілим і адаптивним для різнихпристроїв (ПК,мобільні пристрої). Для дизайну використовуються сучасні додатки (Figma, Material UI). Програма повинна працювати коректро на різних операційних системах та веб-браузерах (Chrome, Firefox, Safari) [9-10];

– код має бути структурованим і документованим, для кращої підтримки таобстуговування.

Нефункціональні виоги визначають якість і стабільність роботи системи, безпеку та масштабованість, зручніть та адаптивність. Врахування цих вимог гарантує комфорт у використанні.

2.3 Технології та інструменти реалізації

Клієнтська частина реалізується як веб-застосунок на основі бібліотеки React.js, з використанням Refine Framework та Material UI. Візуальні компоненти створюються у Figma або Paint для створення сучасного дизайну [9, 11, 12, 13, 14].

Серверна частина побудована на Node.js з використанням Express і мови JavaScript. Для зберігання даних використовується MongoDB з ORM Mongoose. Авторизація реалізована через OAuth2 (Google) з використанням JWT-токенів [15].

Загальний стек технологій: React, Refine, TypeScript, JavaScript, MUI, Node.js, Express, MongoDB, Mongoose, OAuth2, JWT, HTML, CSS, JSON [16].

Кожна технологія звязана між собою, і саме це робить даний продукт цілісним та дозволяє використовувати новітні розробки на максимум.

3 ПРОЕКТУВАННЯ ТА АРХІТЕКТУРА ПЗ

3.1 UML проєктування ПЗ

Перед початком розробки будь-яких частин проєкту, обов’язковим є наявність якісних UML діаграм, які показуют як продукт виглядає з середини. Для серверної частини була побудована діаграма розгортання (рис. 3.1), діаграма прецендентів (рис. 3.2), (уявлення взаємодії ролей з системою), діаграма станів (рис. 3.3), (життевий цикл об’єкта).

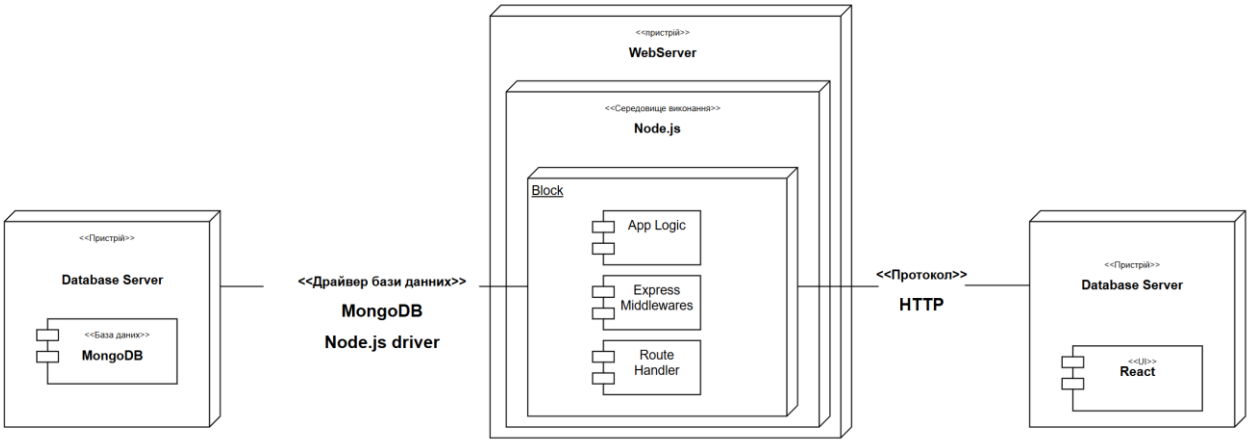

Рисунок 3.1 – Діаграма розгортання

Система складається з трьох компонентів які тісно працюють між собою, це: вебсервер на якому обробляється інформація, база даних, та веб-застосунок, який все поєднує.

База даних це центральне та головне сховище інформації, в якій зберігаються всі необхідні дані. Веб-сервер відповідає за обробку та управління. Веб-застосунок забезпечує інтерфейс.

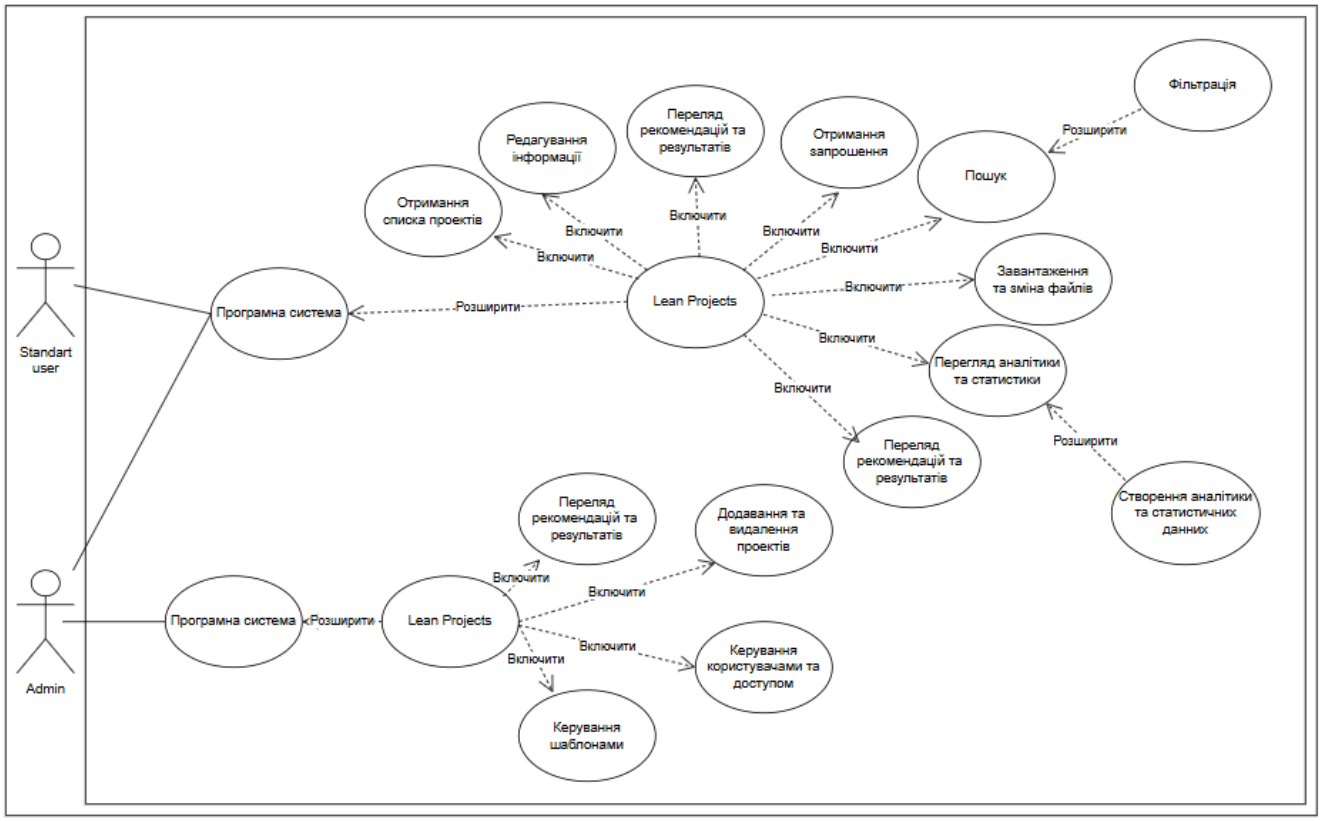

Рисунок 3.2 – Діаграма прецедентів: Standard user та Admin

Standard user може виконувати наступні дії:

– отримувати доступні Lean-проєкти;

– переглядати детальну інформацію про проєкт;

– редагувати власну інформацію у межах проєкту;

– отримувати запрошення до участі в проєктах;

– переглядати рекомендації та результати;

– шукати і фільтрувати дані;

– отримувати рекомендації на основі аналітики.

Admin виконує дії Standard user та також має наступні унікальні дії:

– автентифікація з правами адміністратора;

– додавання нових проєктів або їх видалення;

– редагування існуючих проєктів;

– перегляд рекомендацій і результатів;

– повний контроль над усіма процесами в системі.

– створення аналітики та статистичних даних.

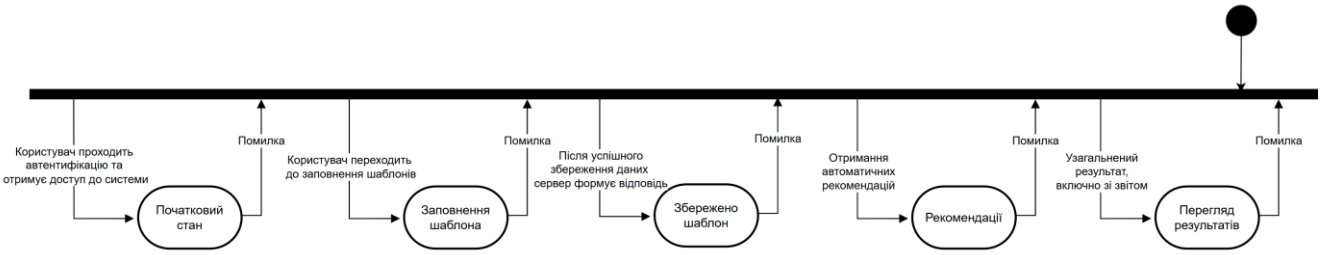

Рисунок 3.3 – Діаграма станів: Standard user

Для кожного типу коростувача побудована діаграма станів. Для Standard user можемо спостерігати: початок роботи, (користувач проходить автентифікацію та отримує доступ до системи), заповнення шаблона (після входу користувач вносить інформацію про проблему, аналіз, поточну ситуацію та можливі покращення), збереження шаблона, (після заповнення зберігається у базі даних, а сервер формує відповідь про успішне збереження), отримання рекомендацій, перегляд результатів. Також хочу додати що будь-який користувач після реєстрації на сервісі має вибір між створенням або додаваням до вже існуючого проєкту. Тільки після цього отримується тип користувача. Це дає змогу розшрити досвід та можливості використання додатку, більш гнучко поділити обов’язки (рис. 3.4).

Для користувачів типу Admin також побудована діаграма. Так як Admin має можливість працювати як і Standard user, було об’єднано деякі функції а саме: редагування та створення проєктів, видалення проєктів, додавання та виключення учасників, (керування складом учасників кожного проєкту).

Хочу звернути увагу що в системі може бути тільки один Admin, це створено для більшої безпеки та уникнути конфліктів при керуванні.

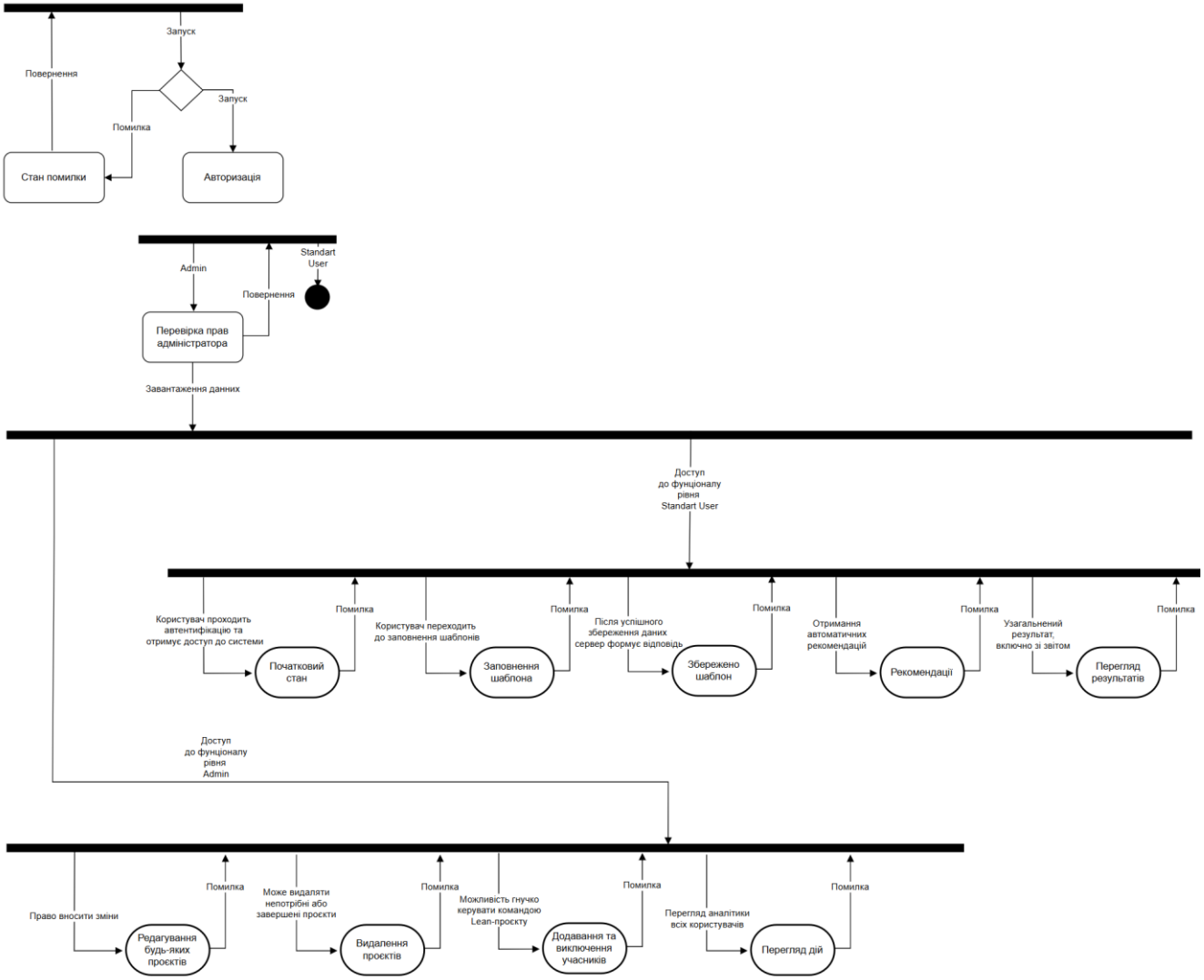

Рисунок 3.4 – Діаграма станів: Admin

Діаграма компонентів показує, які саме компоненти та яким чином взаємодіють із сервером з сервером. Показано, що тільки кілька файлів мають доступ до API, що відповідає одному з основних принципів React – створення незалежних компонентів (рис. 3.5).

Звязок статичних компонентів системи з головними файлами, завдяки цьому зв’язку, підтримується логічна структура (рис. 3.6).

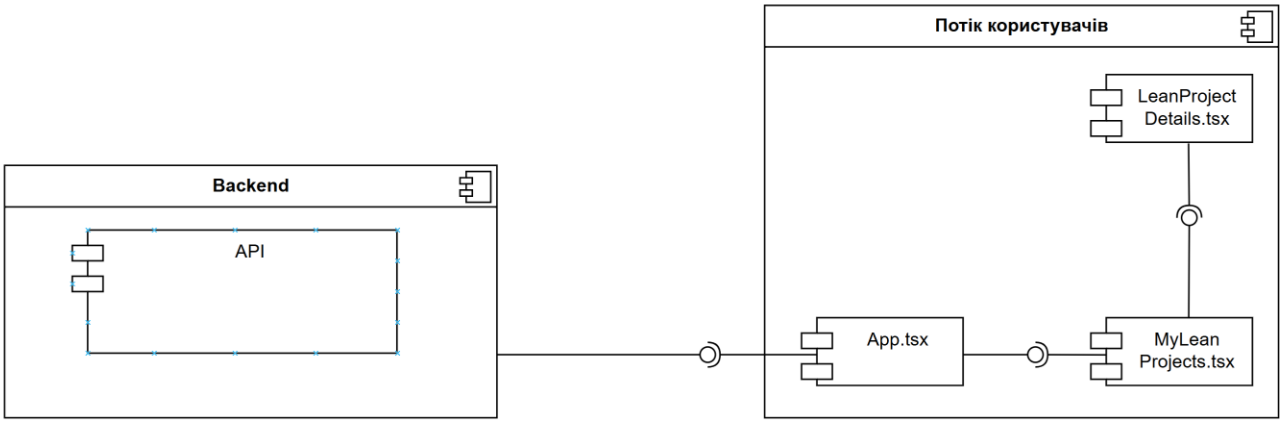

Рисунок 3.5 – Діаграма компонентів

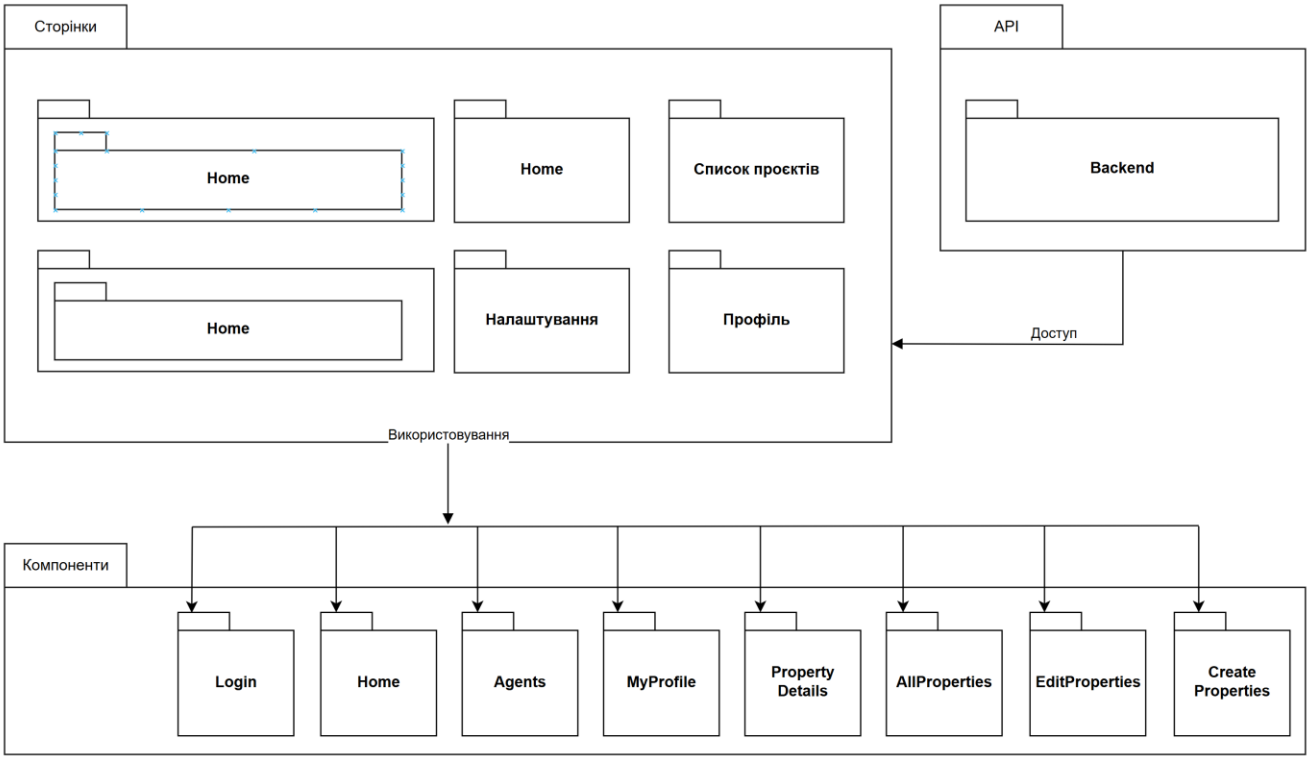

Рисунок 3.6 – Діаграма модулів проєкту

3.2 Архітектура ПЗ

Для проєктування подібних систем був широкий вибір архітектури ПЗ. Взагалі, основних моделей є декілька. Кожен має свої переваги та недоліки, які слід враховувати.

Можна виділити наступні:

– клієнт-сервер;

– компонентна архітекура;

– проблемно-орієнтований дизайн;

– багатошарова архітектура;

– трирівнева архітектура.

В даному проєкті було обрано саме трирівневу архітектуру (рис. 3.7). Для мене, головною перевагою стало наявність зрозумілої і перевіреної структури.

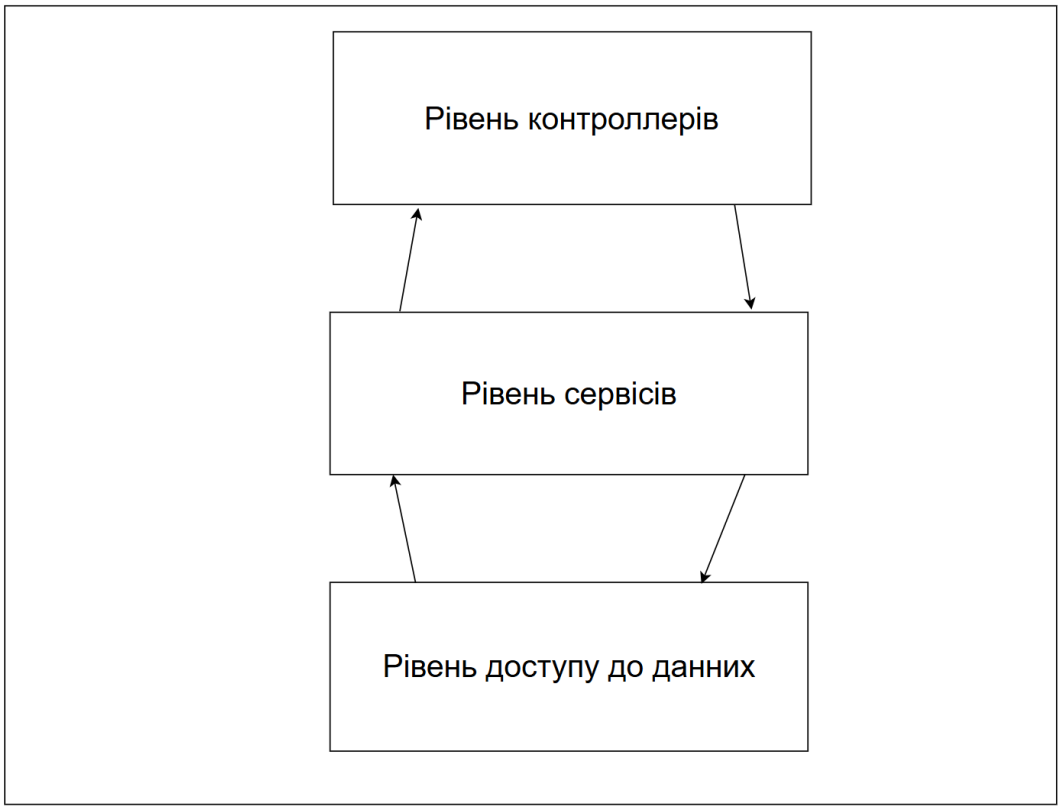

Рисунок 3.7 – Архітектура програмного забезпечення

Загальна струкура:

1. Рівень контролерів.

Цей рівень відповідає за обробку вхідних запитів вд користувача. Контролери приймають запити, визначають, куди їх направляти і головне взаємодіють фронтендом. Вони також відповідають за маршрутизацію, формування відповіді та виклик відповіднох методів сервісного рівня;

2. Рівень сервісів.

Тут реалізована бізнес-логіка програми. Сервіси отримують дані від контролерів і виконують з ними всі необхідні дії наприклад перевірку, розрахунки, взаємодію з базою даних. Тут зосереджена основна логіка, яка не залежить від бази даних чи інтерфейсу;

3. Рівень доступу до даних:

На цьому рівні зберігаються модулі, які безпосредньо працюють з даними. У випадку використання MongoDB, цей рівень включає визначення схем і модулей (бібліотека mongoose) [17-18]. Тут задається структура документів, а також методи, що реалізують операції створення, читання, оновлення та видалення інформації (рис. 3.8).

Уся система побудована відповідно до принципів розподілення відповідальностей (Separation of Concerns), що дозволяє ефективно керувати складністю проєкту. Кожен рівень має чітко визначену зону, що покращує тестування, підтримку та розширюваність системи.

Для наочного представлення архітектури проєкту була створена графічна схема. Діаграмма відображає логічну організацію файлової системи, включаючи основні модулі клієнтської та серверної частини, контроллери, маршрути та головні компоненти інтерфейсу.

На схемі чітко видно ієрархію папок та файлів. Саме це дозволяє легко орієнтуватися, змінювати або додавати новий функціонал до проєкту.

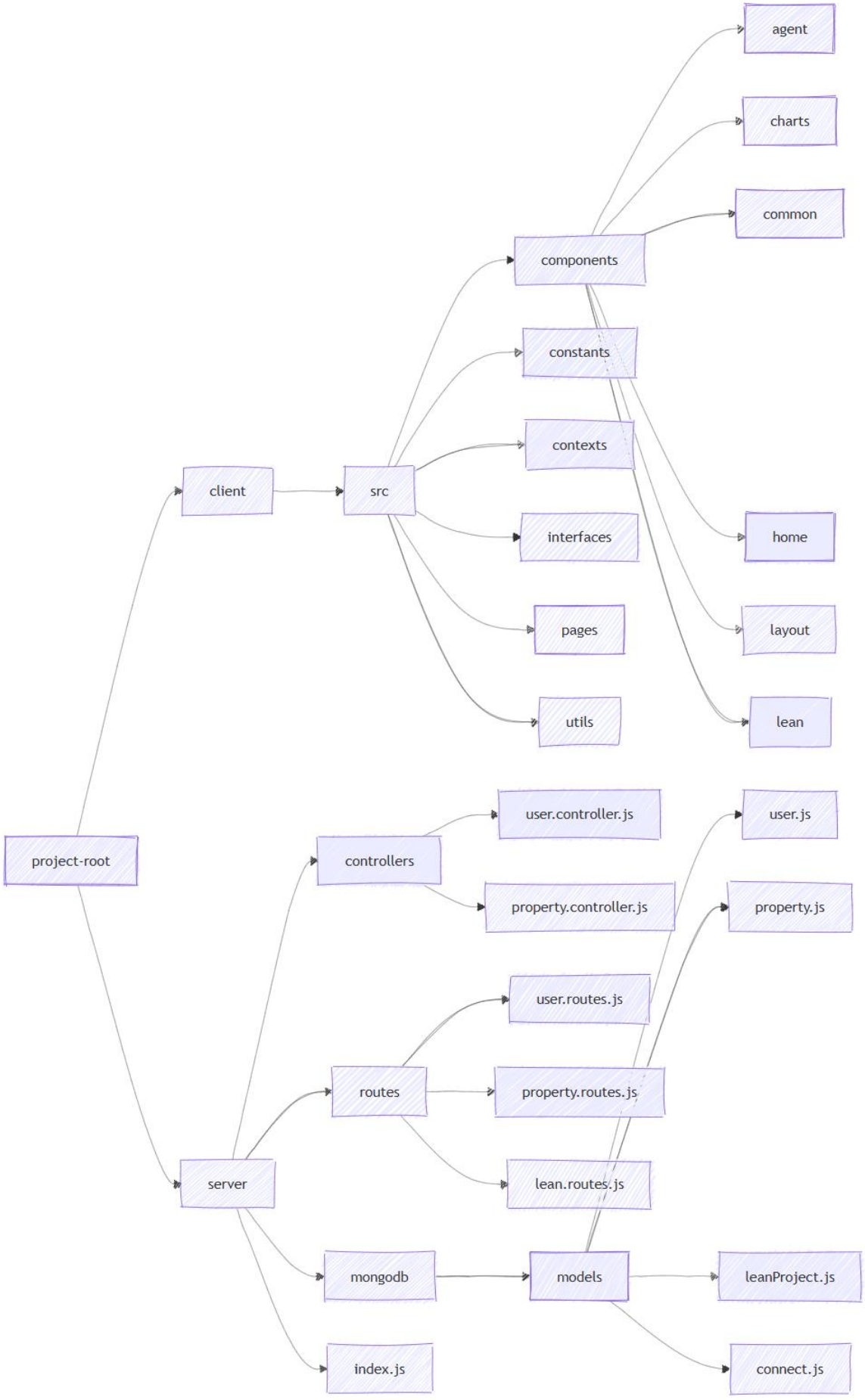

Рисунок 3.8 – Структура програмного забезпечення

Організація проекту та створення зрозумілої структури папок та файлів збільшує продуктивніть праці. Існує головне розділення на клієнтську та серверну частину. Клієнтська частина включає статичні файли, шаблони, сторінки та компоненти. Серверна в свою чергу має головний файл та 3 директорії: controllers, routes, mongodb. Вони відповідають вибраній архітектурі.

3.3 Зберігання данних

Для надійного зберігання данних та в майбутньому легкого доступу створено спеціальну схему для демонстрації (рис. 3.9).

У структурі бази даних реалізовано 6 таблиць, кожна з яких виконує чітко визначену функцію в межах Lean-платформи.

Таблиця «user» є центральною, оскільки саме користувач бере участь у створенні проєктів, надсилає та отримує повідомлення в чаті, а також може бути доданий до проєктів як учасник.

Таблиця «lean_project» містить основну інформацію про Lean-проєкти, включає список учасників і здатна генерувати повідомлення або сповіщення, пов’язані з перебігом проєкту.

Таблиця «participant» моделює зв’язок користувачів із конкретними Lean-проєктами, що дозволяє визначати належність учасників до певної ініціативи.

Таблиця «message» використовується для реалізації внутрішньої комунікації між користувачами в межах платформи, забезпечуючи зручний обмін текстовими повідомленнями.

Таблиця «notification» слугує для передачі важливих подій і системних сповіщень, що стосуються змін у Lean-проєктах або участі користувачів.

Таблиця «property» використовується для зберігання загальних матеріалів, пов’язаних із Lean-методологією, зокрема прикладів, інструкцій, шаблонів, а також цінових оцінок або методичних рекомендацій.

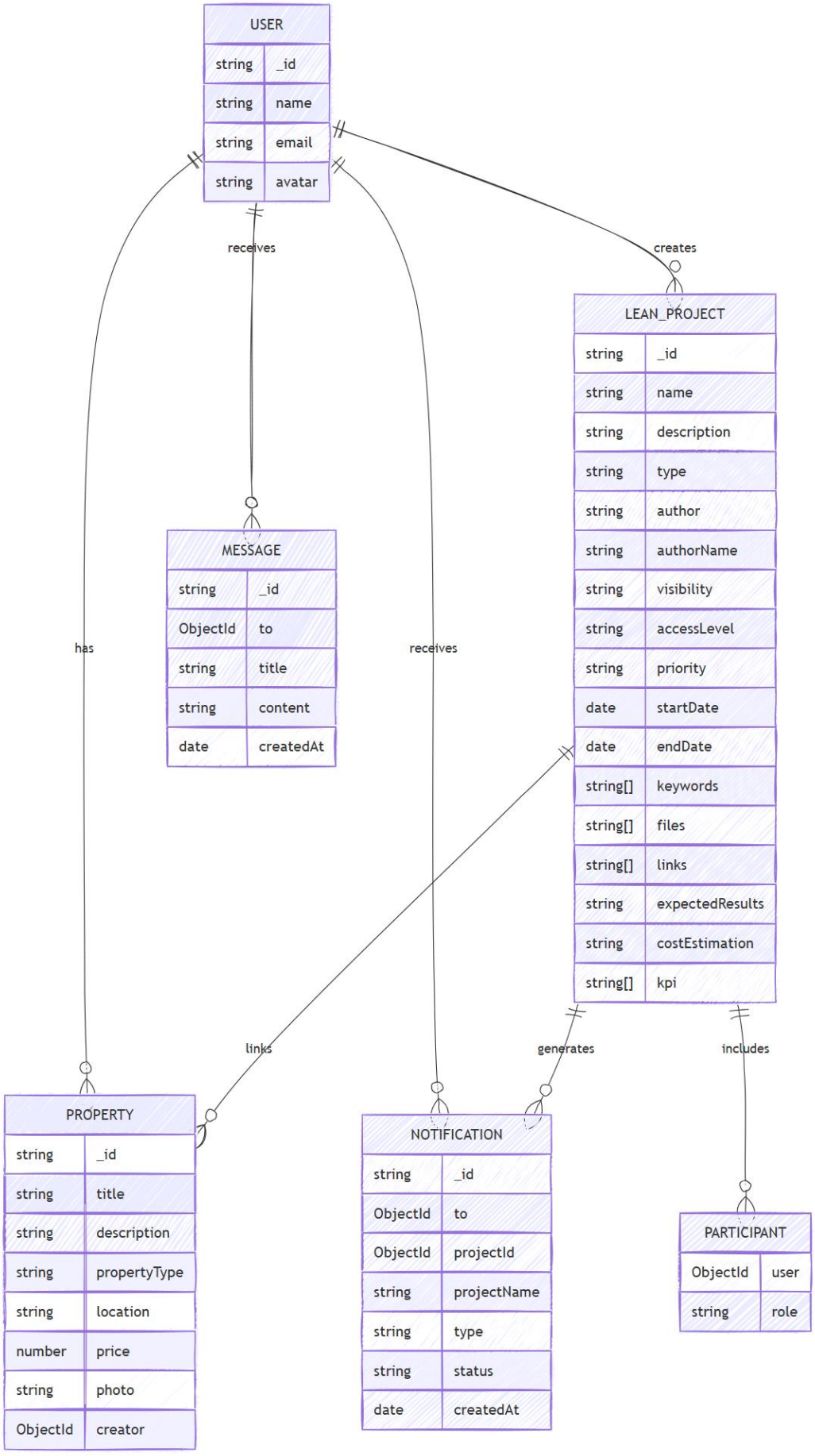

Рисунок 3.9 – Структура бази даних

3.4 Алгоритми та методи

У процесі розробки комп’ютерної консалтингової системи для Lean-виробництва було реалізовано кілька важливих механізмів, які дозволяють ефективно працювати з Lean-проєктами. Один із основних алгоритмів – створення нового проєкту, де користувач автоматично стає його адміністратором. Це зроблено дуже просто: при створенні документа в базі додається масив participants, у якому автор відразу стає admin. Ось так це виглядає в коді:

```
const newProject = new LeanProject({
    ...req.body,
    participants: [
  { user: req.body.author, role: "admin" }
  ],
})
```

Після цього проєкт зберігається через метод .save() і одразу стає доступним для перегляду та редагування. Це дозволяє уникнути додаткових кроків з боку користувача – все відбувається автоматично.

Також було реалізовано функціонал додавання інших учасників у вже створений проєкт. Щоб уникнути дублювання, спочатку перевіряється, чи не був користувач уже доданий раніше:

```
const exists = project.participants.find(
    (p) => p.user.toString() === userId
)
if (exists) return res.status(400).json({ message: "Користувач вже доданий" })
```

Якщо такого учасника немає, то він додається, а потім передається в тілі запиту. Зміни зберігаються:

```
project.participants.push({ user: userId, role: role || "user" })
await project.save()
```

Крім того, у системі реалізовано запит на отримання усіх проєктів певного користувача. Він виконує фільтрацію за полем author:

```
const projects = await LeanProject.find({ author: req.params.userId })
```

Окрема увага була приділена структурі самого об’єкта Lean-проєкту. У моделі LeanProject реалізовано шаблони для методик 5S, Lean Canvas, Kanban та інших. Наприклад, структура шаблону 5S виглядає так:

```
leanTemplates: {
  fiveS: {
    sort: [{ object, status, reason }],
    setInOrder: [{ object, location, labeling }],
    shine: [{ zone, responsible, frequency }],
    ...
  }
}
```

Це дозволяє зберігати деталізовану інформацію по кожному з кроків методу 5S прямо в одному документі проєкту, і потім виводити її в інтерфейсі.

Шаблон Kanban також інтегрований у модель. Там можна створювати задачі з різними статусами:

```
kanban: [
  {
    title: String,
    status: { type: String, enum: [] },
    assignedTo: String,
    dueDate: Date,  },]
```

Усе це працює на базі MongoDB через Mongoose. Дані зберігаються як вкладені об’єкти, і можуть бути легко змінені або відфільтровані.

Також важливою частиною стала робота з користувачами. Вони зберігаються окремо, але мають звязок з проєктами. Наприклад, ось як виглядає модель користувача:

```
const UserSchema = new mongoose.Schema({
  name: String,
  email: String,
```

```
  avatar: String,
  allProperties: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Property' }]
})
```

Через ендпоінт /users/search реалізований пошук за email із підтримкою регулярних виразів:

```
const users = await User.find({
        email: { $regex: query, $options: "i" },
}).limit(5)
```

Таким чином, якщо потрібно додати когось у проєкт, адміністратор просто вводить частину email, і система пропонує відповідні профілі. Це зроблено для зручності командної роботи та пришвидшення роботи.

Усе це реалізовано на Node.js з використанням Express, CORS, body-parser, і все зібрано у REST API. Система працює швидко, надійно і з урахуванням специфіки Lean-підходу, де важливо не витрачати час дарма – ні в реальному виробництві, ні в інтерфейсі [19-20].

3.5 Створення UI / UX системи

UI і UX – це 2 найголовніші та ключові частини цього проєкту. Саме від них залежить настрій, який відчуває користвач. Якісний та привабливий дизайн – запорука будь-якого продукту. Працівника проєкту повинен зустрічати спокійний та виважений дизайн. Я обрав синій та зелений кольори для невеликих елементів, які треба внести до уваги [10]. Також, однією з фунцій, стала можливість зміни теми персоналізації. Було обрано світлу та темну колюрову палітру.

На рисунку 3.10 можна спостерігати початкову сторінку. На ній міститься вікно з Lean-проєктами. Зліва видніється сайдбар з можливими функціями та швидким переходом до інших розділів.

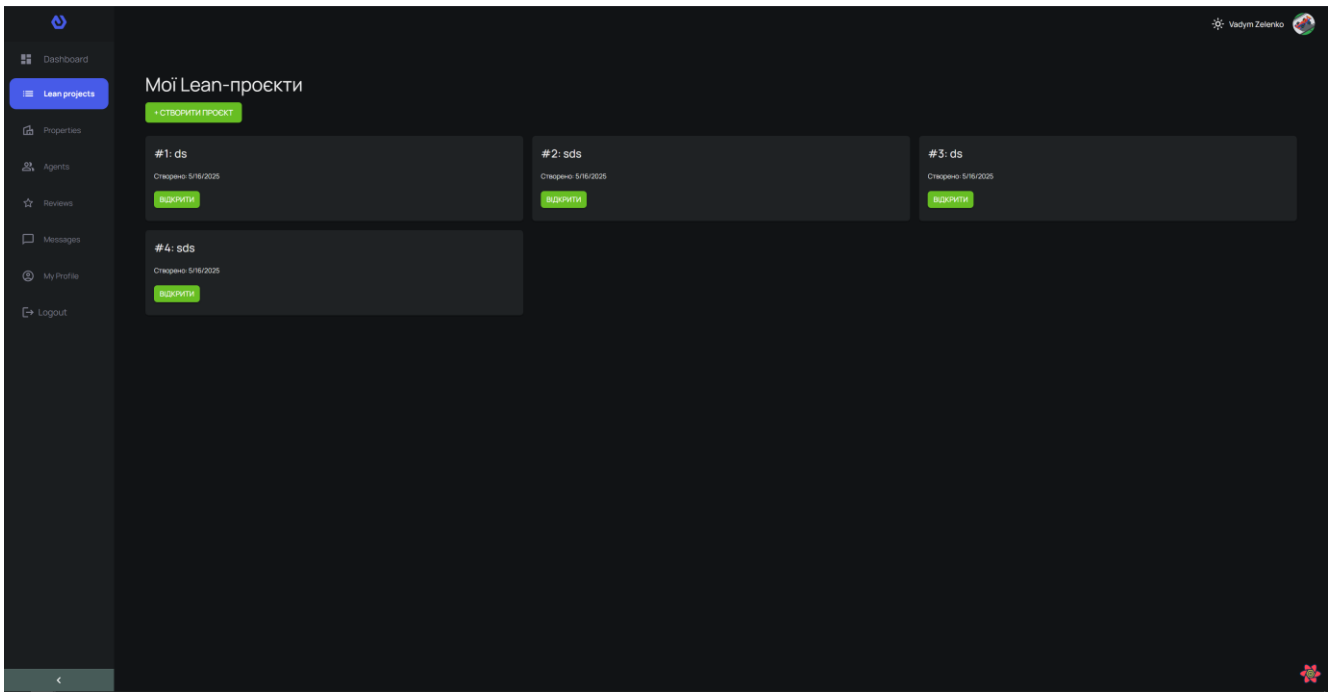

Рисунок 3.10 – Початкова сторінка програми

Інтерфейс модуля KanbanBoard, який реалізує один із ключових принципів Lean Production – візуалізацію потоку робіт. Kanban-дошка поділена на п’ять основних колонок: Backlog, To Do, In Progress, Review та Done, що відповідає стандартним етапам життєвого циклу завдання (рис. 3.11).

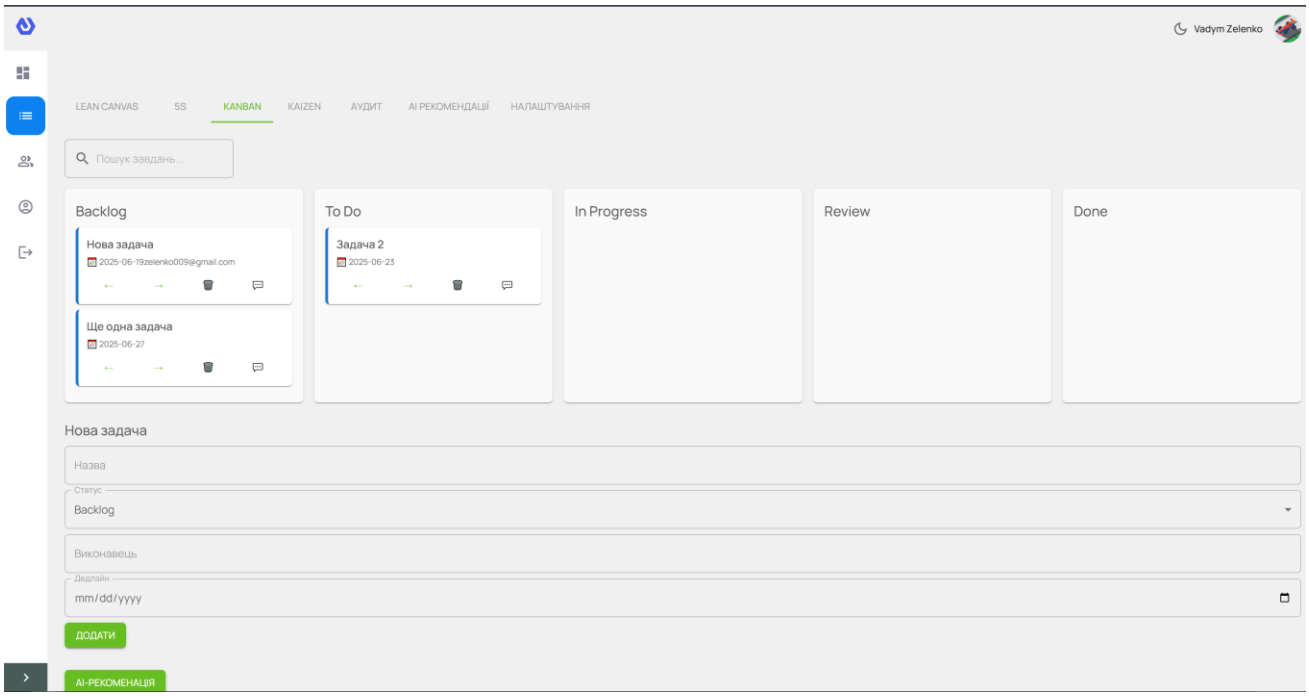

Рисунок 3.11 – Сторінка Kanban

Користувач має можливість додавати нові задачі за допомогою форми, розміщеної нижче: тут вводиться назва задачі, вибирається її статус (тобто колонка, у якій вона з’явиться), виконавець і дедлайн. Цікавим компонентом є те що в кожної задачі є пункт «КОМЕНТАРІ». Це дозволяє легко робити відгуки, що добре сказується на команді. Також наяві кнопки для швидкого перекидування задач на інші колонки. Кнопка «ДОДАТИ» створює нову задачу, а «ЗБЕРЕГТИ ЗМІНИ» дозволяє оновити вже наявні. Всі дані та інформація записується в JSON формат, що в подальшому використовується для аналізу.

Такий підхід дозволяє прозоро керувати завданнями, зменшувати перевантаження виконавців, виявляти вузькі місця у процесі та підтримувати команду.

Один з принципів який використовують для комплексного аналізу робочого місця це методологія 5S (рис. 3.12). Для заповнення доступно 5 форм, і кожна містить 3 колони. Із-за того що форми мають вільне заповнення, (без чітких норм запису данних), це дає можливіть підлаштовуватись під будь-який проєкт, колектив та робоче місце (рис. 3.13).

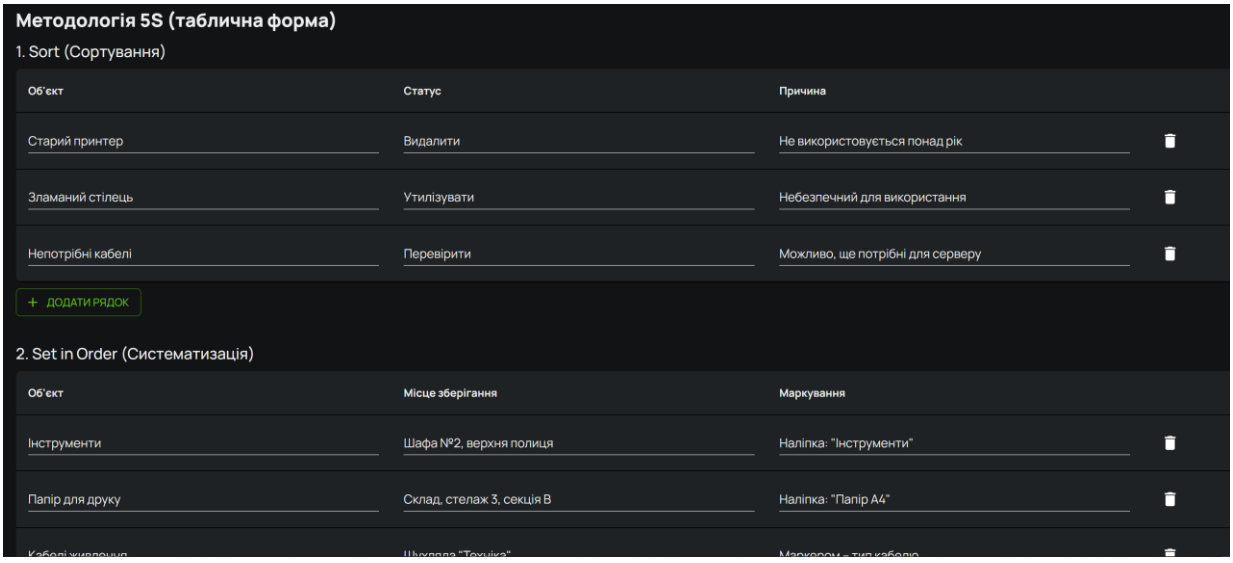

Рисунок 3.12 – Розділ 5S: Sort, Set in Order

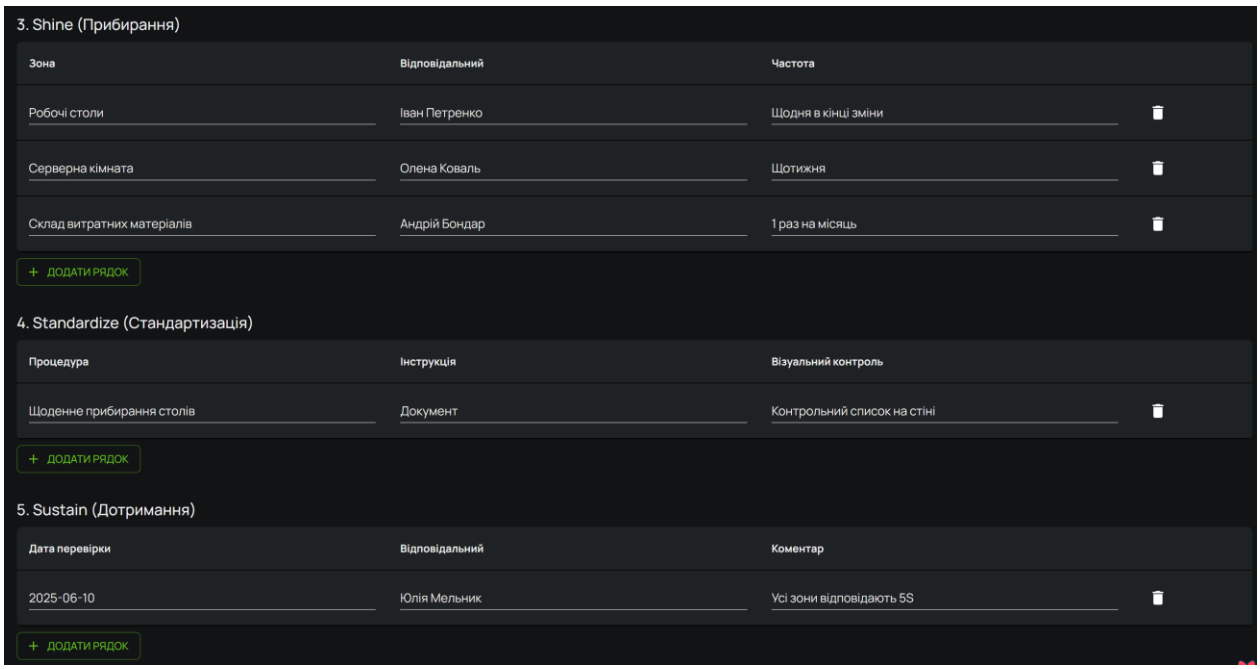

Рисунок 3.13 – Розділ 5S: Shine, Standardize, Sustain

Наступний розділ це Lean-аудит. Він є, без сумнівів, важливою частиною виробничих та логістичних, (в залежності від проєкту), процесів, і його головна мета це виявлення втрат, щоб в подальшому вдосконалюватись (рис. 3.14).

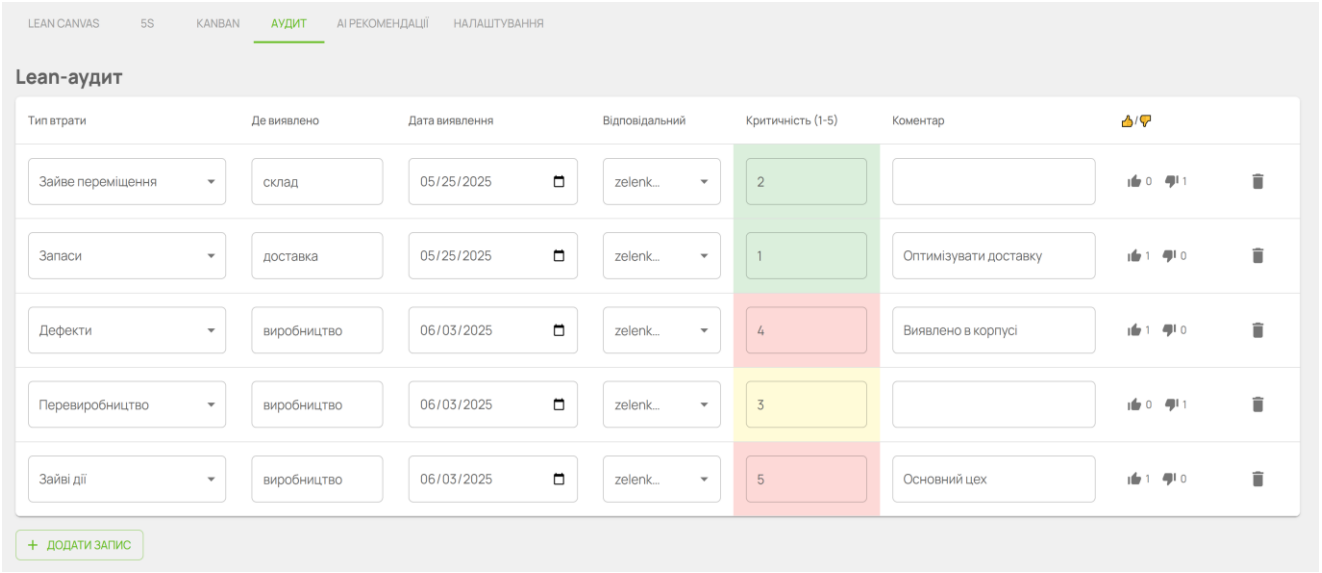

Рисунок 3.14 – Розділ Lean-аудиту

Форма дозволяє зазначити тип втрати, місця виявлення, дату, відповідального працівника і обовязково рівень критичності (від 1 до 5).

Передбачено ще кілька функій, а саме: коментар до ситуації та можливісь голосування. Це дає розуміння про колектив та внутрішню політику. Також ці дані використовуються для генерації точніших звітів. Дизайн виконаний з акцентом на зручність та швидкого вивченя інтерфейсу. Поле критичності має логіку підсвітки трьома кольорами, це привертає увагу.

Кожен проєкт унікальний. Кожен містить свої задачі, розвязки та ідеї. І щоб генерувати точні рекомендації було додано автоматичний аналіз всієї інформації штучним інтелектом. Це дозволяє швидко отримати свіжу аналітику по вибраному проєкту, охарактеризувати ключові проблеми та надати підсказки [21-23].

Даний розділ містить в собі текстовий блок, кнопку генерації та кнопки створення звітів в форматах PDF, MD, TXT та EXCEL. Алгоритм підвищує ефективність та зменшує робоче навантаження в співробітників (рис. 3.15).

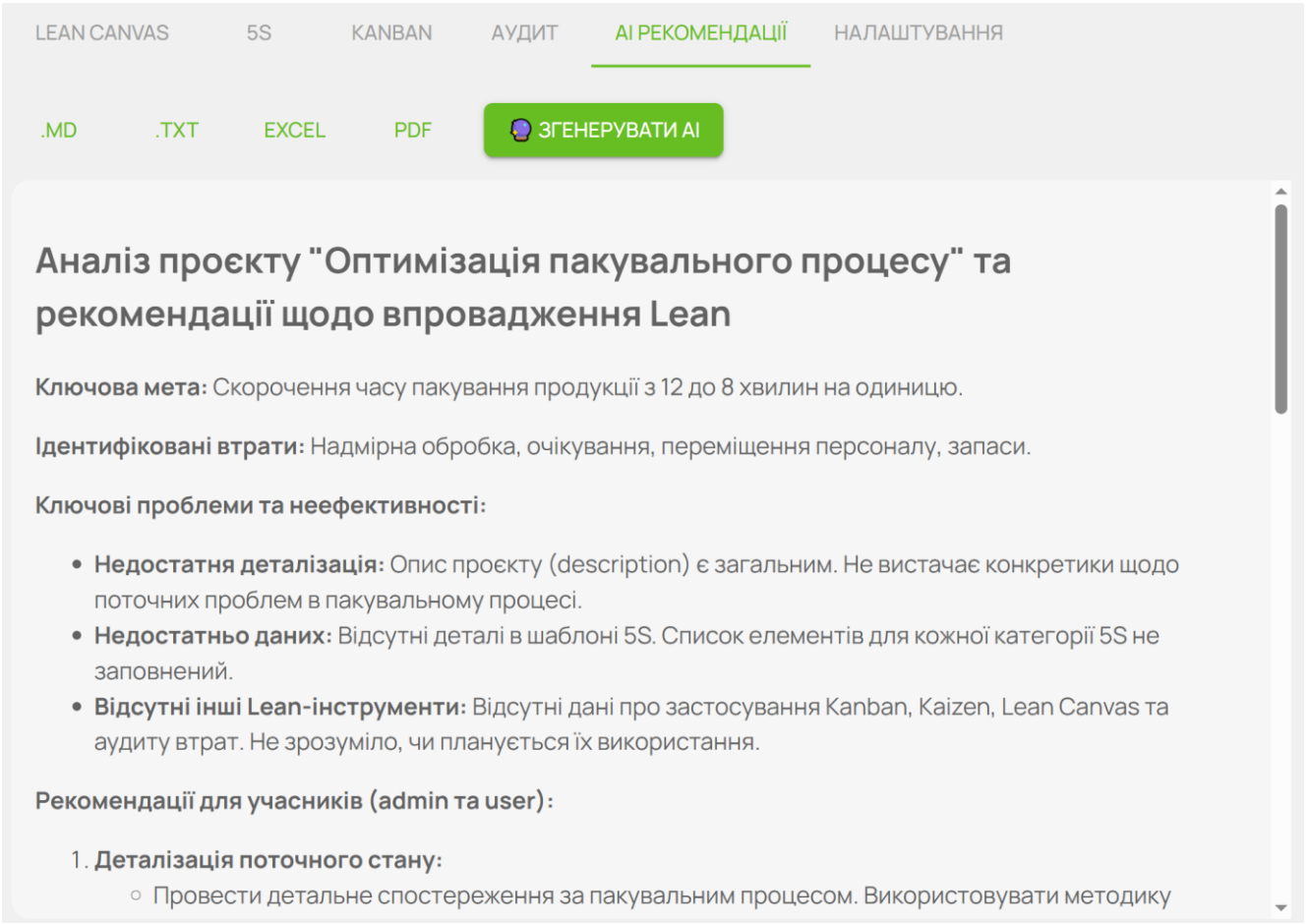

Рисунок 3.15 – Розділ AI-рекомендацій

Останній розділ це налаштуваня. Тут реалізована логіка пошуку, додавання та видалення користувача з проєкту. Наявна кнопка видалення проєкту, яка працює з подвійним підверженням, щоб уникнути випадкового видалення (рис. 3.16).

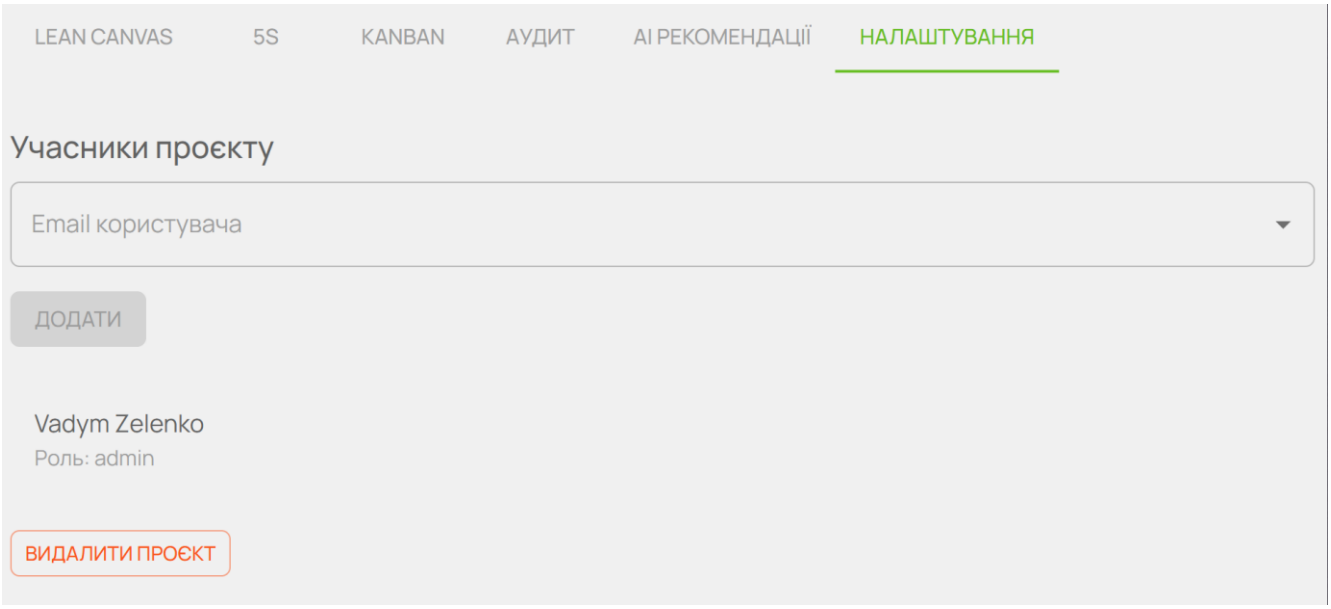

Рисунок 3.16 – Розділ налаштувань

Інтерфейс сторінки зі списком агентів для пошуку користувачів (рис. 3.17). Основна частина екрана містить перелік агентів із зазначенням імені, електронної пошти та зображення профілю.

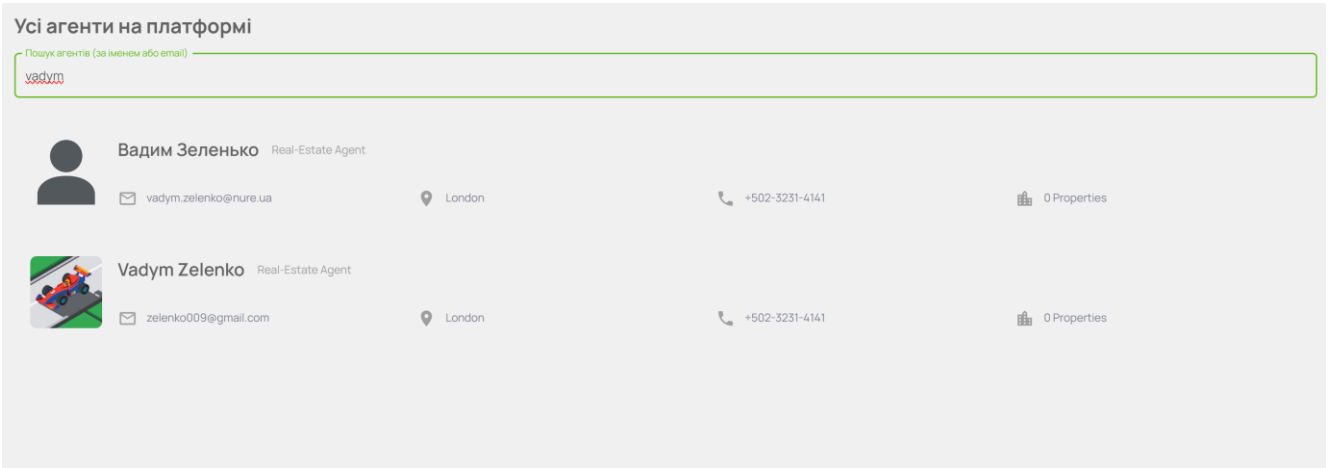

Рисунок 3.17 – Роздід агентів

Кожному користувачу автоматично створюється та надається профіль. В ньому є головна інформація про працівника. Ще є пункт запрошень, куди приходять сповіщення для додавання в проєкт. Запрошення можно прийняти та відхилити. Також наявна функция видалення аккаунту з системи (рис. 3.18).

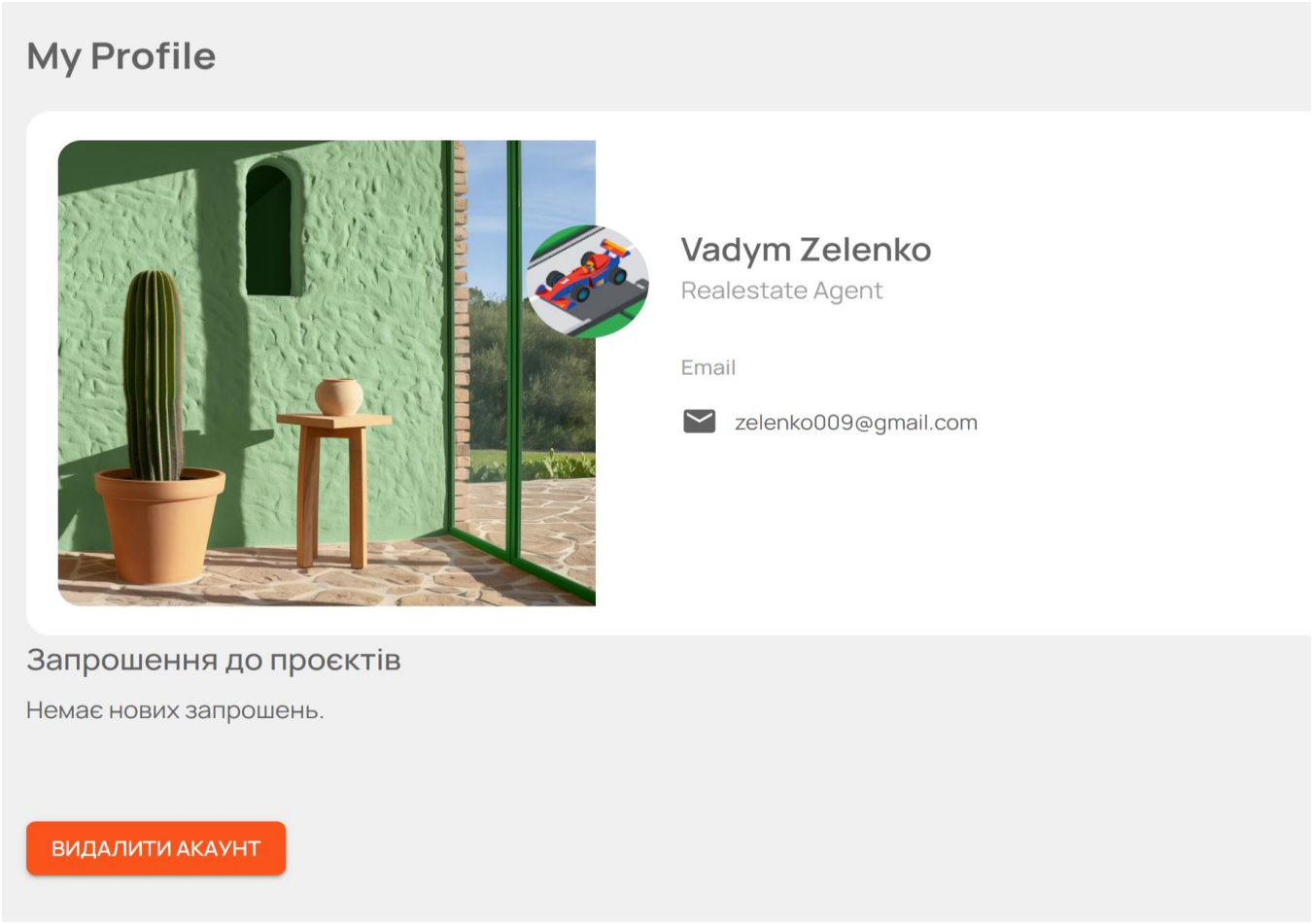

Рисунок 3.18 – Профіль користувача

Варто згадати про тестових агентів (рис. 3.19). Для перевірки інтерфейсу, сторінок користувачів та роботи бази даних, я створив кілька сотень агентів, які мають аналогічні властивості, як і справжні профілі. Це було зробленно за домопогою самостійно написаного скрипту, який використовує цикл та пришвидшує процес.

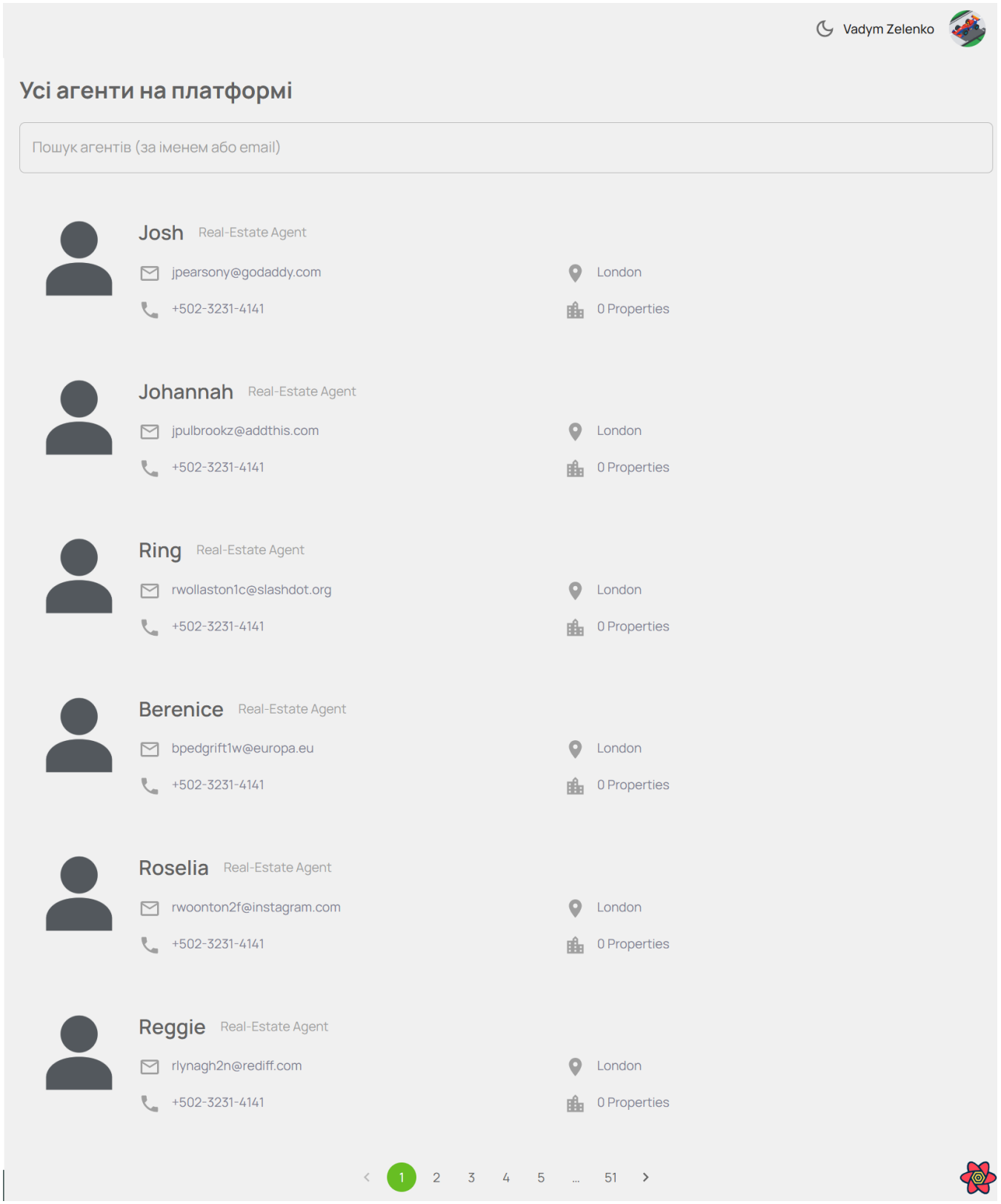

Рисунок 3.19 – Тестові агенти

4 ОПИС ПРИЙНЯТИХ РІШЕНЬ

4.1 Опис серверної і клієнтської частин проекту

Щоб отримати якісну роботу або проєкт, потрібно приймати правильні та продумані рішення. Рішення повинні обиратися та прийматися так щоб безпека, ефективність та зручніть об’єднувались в одне ціле.

Проєкт має дві головні частини – серверна та клієнтська. В якості сервернох логіки було обрано Node.js – асинхронну платформу, що на мою думку, ідеально підходить для побудови високонавантажених веб-додатків. Однією з переваг це здатність обчислювати багато запитів одночасно та не блокувати при цьому основний потік. Саме це дає нам продуктивність та прискорення.

Сховище реалізовано за допомогою нереляційнох бази MongoDB, яка зберігає в форматі JSON. Навколо цього типу файлів побудовано не тільки зберігання, але і звязок з штучним інтелектом. База прекрасно співпрацює з JavaScript та дає хороші показники швидкості, гнучкості та головне безпеки. Звязок створюється через спеціальний ключ, захищений криптографічним протоколом.

```
mongoose.set('strictQuery', true);
mongoose.connect(url)
```

В проєкті реалізовано багату структуру збереження. Саме вона записується в директорії models. Для кожного розділу чи задачі є свій файл. Наприклад в файлі leanProjects.js включає учасників з ролями, принципи Lean. Для Kanban реалізовану таку модель:

```
kanban: [
  {
 title: String,
status: { type: String, enum: ["backlog", "todo", "in_progress", "review", "done"]},
   assignedTo: String,
   dueDate: Date,
```

```
}]
```

Файл notification.js має поля to, projectId, type, status та виглядає так:

```
{
  type: { type: String, enum: ["invite"] },
  status: { type: String, enum: ["pending", "accepted", "declined"] }
}
```

Сервер має також головний файл index.js, де прописується маршрути та ендпоінти:

```
app.use('/api/kaizen', kaizenRouter)
app.use('/api/v1/lean', leanRouter)
app.use('/api/v1/users', userRouter)
```

Використання транзакції Mongoose це запорука забезпечення цілісности при створенні повязаних між собою об’єктів:

```
const session = await mongoose.startSession();
session.startTransaction();
const user = await User.findOne({ email }).session(session);
const newProperty = await Property.create({ ... });
user.allProperties.push(newProperty._id);
await user.save({ session });
await session.commitTransaction();
```

Перевагаю є те що об’єкт не буде створено частково, що виключає «висячі» записи.

Реалізація логіки голосування цікава тим, що потрібно врахувати не тільки один голос але і не дозволити поставити протележну реакцію. Користувач не може поставити лайк і дизлайк одночасно:

```
if (type === "like") {
  if (idea.likes.includes(userId)) removeLike();
  else addLike(); removeDislike();
}
```

Реалізація системи запрошень є чи не найважливішою. По-перше користувач повинен вирішити, додаватися в проєкт чи відхилити пропозицію. Створено для зменшення спаму:

```
router.put('/:id/accept', async (req, res) => {
  const notif = await Notification.findById(req.params.id);
  if (!notif) return res.status(404).json({ message: 'Не знайдено' });
  const project = await LeanProject.findById(notif.projectId);
  if (!project) {
      await Notification.findByIdAndDelete(notif._id);
   return res.status(404).json({ message: "Запрошення видалено." });
  }
  notif.status = 'accepted';
  await notif.save();
  const alreadyInProject = project.participants.some(
   (p) => p.user.toString() === notif.to.toString()
  );
  if (!alreadyInProject) {
   project.participants.push({ user: notif.to, role: 'user' });
   await project.save();
  }
  res.json({ message: 'Успішно приєднано до проєкту' });
});
```

По-друге потрібно автоматично видаляти повідомлення після вибору, не тільки з UI, але і з бази даних:

```
router.put('/:id/decline', async (req, res) => {
  const notif = await Notification.findById(req.params.id);
  if (!notif) return res.status(404).json({ message: 'Не знайдено' });
  notif.status = 'declined';
  await notif.save();
  res.json({ message: 'Запрошення відхилено' });
```

```
});
```

При розробці продуктів які будуть використовувати в команді, завжди потрібно проводити аналіз та перевірку тестовими значенями. Я створив сотні фейкових профілів для цього, але щоб очистити тестових користувачів і неавторизованих профілів було написано:

```
await User.deleteMany({
  $or: [
    { avatar: { $exists: false } },
    { name: /bot/i },
    { email: /fake|bot|test/i }
  ]
});
```

Для легкого знаходження користувача із заданим userId використовується цей блок коду:

```
projects.forEach(proj => {
  proj.participants.forEach(p => {
    if (p.user._id.toString() !== userId) agents.push(p.user);
  });
});
```

Це дозволяє швидко шукати по унікальному імені та в подальшому працювати зі знайденим профілем.

В проєкт було інтегровано генеративну модель штучного інтелекту Gemini-2.0-flash, розроблену компанією Google. Основна мета є автоматизований аналіз данних, для формування точних рекомендацій. Взаємодія з штучним інтелектом реалізована через офіційну бібліотеку @google/genai, це дозволяє напряму підключатися з Node.js. Щоб це відбулося легко та безпечно, при використанні, на сервері створюється окремий маршрут /api/v1/ai/gemeni, і вбудована логіка парсить дані з бази та створює окремий JSON файл [21]. В файлі створюються автоматичні запити, додаються дані та очищуються порожні бланки.

```
const prompt = formatPrompt(cleanedData);
```

```
const result = await ai.models.generateContent({
  model: "gemini-2.0-flash",
  contents: [{ parts: [{ text: prompt }] }]
});
```

Отримана відповідь прохообробку та записується на місце застарілих рекомендацій. Це дозволяє робити дві справи одночасно, видалення неподрібних даних та збереження нових.

Щоб підключатися до віддалених серверів потрібних сервісів, потрибно мати спеціальний ключ доступу – API_KEY. Ці ключі записуються в окремий файл .env та дозволяють використовувати їх ізольовано. Ось кілька з них:

– MONGODB_URL – ключ бази данних;
– GEMINI_API – ключ до сервісу штучного інтелекту.

Якщо говорити про клієнтську частину, то там реалізовано не менше цікавих рішень, але більшість була описана в розділі UI. Код в цій частині написаний на TypeScript. TypeScript – це продовження JavaScript, але з статичними типами даних. Є дуже зручним в пошуку помилок та їх випралення. Зазвичай його обирають для великих проєктів де важлива довгострокова підтримка [24].

Клієнська частина побудована на сучасній бібліотеці React. Вона дає створити масштабований, зручний та швидкий інтерфейс. Інтерфейс ділиться на спеціальні компоненти, з яких потім складається весь додаток. Одним з головних принципів React є принцип розділення відповідальностей, наприклад, в компоненті login не може бути логіки про компоненту kanban. Дана бібліотека працює по принципу Single Page Application, це значить що сторінка не перезавантажується, а тільки дозавантажує необхідні компоненти.

Для легкої побудови додатку яких тісно працює з даними було обрано фреймворк Refine. Він має вбудовані функції та методи, які зменшують кількість коду та пришвидшують розробку. Призначений для будь-якого типу CRUD систем, наприклад адмін-панелі або внутрішні інструменти, такі як консалтинг.

Також було прийнято рішення уникати глобальних залежностей між модулями, натомість використовувати контекст Refine, useList, useShow,

usePermissions та інші спеціалізовані хуки. Це забезпечило гнучке керування доступом до ресурсів, дозволило ефективно реалізувати систему ролей та прав доступу без дублювання логіки. Завдяки цьому підхід до розробки залишається масштабованим, а кожен функціональний блок – легко тестується та супроводжується.

Варто згадати за адаптивність інтерфейсу на мобільні пристрої, що дає нові можливості у використанні.

Однією з цікавих функцій є транзакційне створення проєкту та прив'язка автора. Це гарантує, що проєкт і користувач будуть звязані і один одного не загублять при виході з системи:

```
const session = await mongoose.startSession();
session.startTransaction();
const newProject = await LeanProject.create([{ ...projectData }], { session
});
user.projects.push(newProject[0]._id);
await user.save({ session });
await session.commitTransaction(
```

4.2 Підхід до тестування ПЗ

Зважаючи на компактність проєкту, складність деяких функцій та використання фреймворку Refine, я прийняв рішення використовувати комплексний підхід. Проводилось декілька типів тестування – модульне, інтеграційне, UI/UX тестування, ручне тестування та з допомогою тесувальної системи Refine.

Для перевірки основних на ключових серверних функцій (валідація форм, логіка ролей та інші), були написані тести на Node.js. Була вибрана бібліотека Jest. Вона дає змогу перевірити кожну функцію окремо.

Інтеграційне тестування це перевірка взаємодії між frontend та backend частинами. Перевірялось авторизацію через Google, створення проєктів, управління учасниками, голосування за ідеї (лайк і дизлайк).

UI/UX тестування це ручна перевірка адаптивності інтерфейсу, зручності, роботу в різних браузерах. Перевірка анімацій та форм для заповнення.

Ручне тестування є особливо ефективним в випадках, коли треба достідити дивні і неочевидні баги або вразливості, які можуь виникати під час праці різних елементів системи.

Тестування за допомогою Refine у проєкті передбачає те, що можна охопити підготовку середовища, формування чітких сценаріїв і збір метрик в Devtools, для забезпечення повного циклу перевірки. Для додатку такого рівня цього тестування цілком достатнью [12].

4.3 Тестування ПЗ

Локальна та глобольна перевірка фунцій проєкту є чи не найважливішим в розробці. Саме це ми називаємо тестуванням. Воно допомагає виявити потенційні помилки та вчасно їх усунути. Зважаючи систему та процес який вона виконує, було застосовано кілька підходів.

Тестування модульним методом виявило проблеми з валідацією форм, а саме коли поле title порожне, дані не зберігало до бази та сторінка не працювала. Помилки виникли з логікою голосування, при натисканні на кнопку статус already voted повертався двічі і при перезавантаженні записувалось два голоси. Це дало чіткі інструкції, де саме шукати помилку.

Підчас ручного тестування було знайдені цікаві помилки звязані з відключенням інтернету при роботі, а саме показ вбудованого меню JavaScript, яке не давало користватися браузером.

Система Refine Devtools дала найбільший список неточностей. Перевірені стани loading, error, success та робота хуків useList, useCreate, useUpdate, useDelete не давали точних відповідей та не повертало успішних відповідей. Також

перевірено accessControlProvider (рольовий доступ) і при canAccess(“/admin”) повертав true для користувачів Standard user. Ця критична помилка давала права адміністратора всім в проєкті [21].

Інтеграційним тестванням перевірено більшість API-запитів та особливо логіку ролей через токен. Усі маршрути перевірялись як з правильними так і з хибними даними.

4.4 Охорона праці

Забезпечення безпечних умов праці є пріорітетом для будь якої компанії. Це стосується не лише виробництв, але й офісів, де основним робочим інструментом є комп’ютер. Під час цього можуть виникнути деякі загрози для здоров’я, тому дотримання правил є обов’язковим.

Основні чинники ризику:

– перегрів повітря через виділення тепла від комп’ютерного обладнання;

– надмірна візуальна концентрація, що провокує втомлюваність очей;

– імовірність ураження статичною електрикою під час дотику до техніки;

– недостатній рівень освітлення робочого місця;

– вплив електромагнітного випромінювання, яке створюють пристрої;

– яскравість і контраст екрана, що шкодить зору при тривалому контакті.

Організація та правильне облаштування простору – запорука мінімізації ризиків.

Основні рекомендації:

– регулярне використання антистатичних засобів та зволожувачів повітря;

– всі дроти повинні бути надійно закріплені, не створюючи перешкод;

– вилка та розетка мають бути без пошкоджень;

– під час ремонтів в офіс краще уникати займистих матеріалів, віддаючи перевагу вогнетривким;

– приміщення має бети добре провітрюване, з можливістю охолодження в спеку;

– робоча зона повинна бути вільною від предметів які заважають користованню технікою.

Перед вмиканням ПК працівник має переконатись:

– у відсутності оголених проводів;

– у справності обладнання;

– у відсутності пошкоджень корпусів та несправних індкаторів;

– на системному блоці немає предметів які блокують вентиляцію;

– увімкнення техніки має здійснюватись коли справні і заземлені розетки;

– не можна вмикати ПК у вологому середовищі або з мокрими руками;

– під час чищення пристрою він обов’язково має бути вимкнений;

– заборонено знімати корпус пристрою під час його роботи;

– категорично не можна вживати їжу або палити за комп’ютером.

Дотримання правильної постави та організації тіла під час роботи знижує навантаження:

– відстань до монітора – щонайменше 50 см;

– клавіатура розташовується на відстані 20–30 см від краю столу;

– спина злегка спирається на спинку стільця, руки розслаблені;

– категорично не можна вживати їжу або палити за комп’ютером;

– лікті знаходяться на столі або підлокітниках, а не висять у повітрі;

– ноги стоять рівно, не схрещені й не підгорнуті;

– щогодини необхідно вставати та розминати м’язи.

Дотримання норм охорони праці – це не лише формальність, а справжній вклад у здоров’я та продуктивність працівників. Комплексний підхід до безпеки офісної роботи дає змогу попередити травми, зробити комфорт і створити хороше середовище для кожного [25-26].

ВИСНОВКИ

У процесі виконнання кваліфікаційної роботи було побудовано програмний продукт який має детально продуману і протестовану серверну частину та стильну і адаптивну клієнтську частину.

Серверна частина використовує Node.js з використанням JavaScript, це забезпечує надійний фундамент для подальшого масштабування. Налаштовано та підключено базу данних MongoDB.

Клієнтська частина побудована в тандемі бібліотеки React та фреймворку Refine. Це ідельно підходить для побудови консалтингових систем. Стилізація не займає останнє місце. Використовувалась бібліотека Material UI, яка прекрасно вбудовується в даний проєкт.

Проєкт дозволяє не просто робити нотатки, а використовувати штучний інтелект для створення ідеальних рекомендацій. Було створено основні принципи Lean, а саме Kanban, 5S, Lean-аудит [15]. Досліджено приклади реалізації подібних платформ, таких як iObeya, що дало змогу визначити те що необхідно для створення власної системи підтримки Lean проєктів.

Результатом стала розроблений прототип комп’ютерної консалтингової системи яка реалізує базові функції керування Lean проєктами включаючи візуалізацію процесів формування звітів розподіл ролей що можуть бути використані в реальних умовах виробничих підприємств.

Ціль роботи була досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП «УкрНДНЦ». 2016. 30 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно–інтегровані технології» освітньої програми «Автоматизація та комп'ютерно–інтегровані технології» / Упоряд.: І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2022. 55 с.

3. Ohno T. Toyota Production System: Beyond Large-Scale Production. – Productivity Press, 1988.

4. Rother M., Shook J. Learning to See. – Lean Enterprise Institute, 2023.

5. React Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://legacy.reactjs.org/docs/getting-started.html.

6. Lean Enterprise Institute. [Електронний ресурс] / – Режим доступу: www / URL: https://www.lean.org.

7. UX Design Lean UX. [Електронний ресурс] / – Режим доступу: www / URL: https://uxdesign.cc/tagged/lean.

8. Gemini API. [Електронний ресурс] / – Режим доступу: www / URL: https://ai.google.dev.

9. Jest Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://jestjs.io/.

10. Interaction Design Lean UX Principles. [Електронний ресурс] / – Режим доступу: www / URL: https://www.interaction-design.org/literature/article/lean-ux-getting-out-of-the-deliverables-business.

11. TypeScript Handbook. [Електронний ресурс] / – Режим доступу: www / URL: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html.

12. Express.js Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://expressjs.com/.

13. Figma. [Електронний ресурс] / – Режим доступу: www / URL: https://www.figma.com.

14. iObeya Platform. [Електронний ресурс] / – Режим доступу: www / URL: https://www.iobeya.com.

15. OWASP Security Cheat Sheet Series. [Електронний ресурс] / – Режим доступу: www / URL: https://cheatsheetseries.owasp.org/.

16. Google OAuth 2.0 Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://developers.google.com/identity.

17. Mongoose Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://mongoosejs.com.

18. Documentation JWT. [Електронний ресурс] / – Режим доступу: www / URL: https://jwt.io/introduction.

19. Multer Middleware. [Електронний ресурс] / – Режим доступу: www / URL: https://expressjs.com/en/resources/middleware/multer.html.

20. MongoDB Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://www.mongodb.com/docs/.

21. NPM package @google/generative-ai. [Електронний ресурс] / – Режим доступу: www / URL: https://www.npmjs.com/package/@google/generative-ai (дата звернення: 11.06.2025).

22. Material UI Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://mui.com/material-ui/getting-started/overview/.

23. Refine Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://refine.dev/docs.

24. Refine Devtools Documentation. [Електронний ресурс] / – Режим доступу: www / URL: https://refine.dev/docs/devtools/introduction.

25. Гриненко А. В., Швидкий А. М. Охорона праці в IT-сфері: сучасні ризики та засоби їх мінімізації. – 2021. – №2. – С. 45–52.

26. Комплекс навчально-методичного забезпечення навчальної дисципліни «Безпека праці в індустрії ІТ-технологій» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [http://catalogue.nure.ua/knmz] / ХНУРЕ; розроб.: Т. Є. Стиценко, Г. В. Пронюк, Н. М. Сердюк. – Харків, 2017. – 122 с.