МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Інформаційна технологія підтримки оперативного планування роботи проектного менеджера

Здобувача (ки) групи ІТ.м-23 Райка Дмитра Івановича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Д.І. Райко
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к.т.н., доц. Федотова Н.А. _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Райко Дмитру Івановичу
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Інформаційна технологія підтримки оперативного планування роботи проектного менеджера

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «___» ___грудня___ 2023 р.

3 Вхідні дані до кваліфікаційної роботи завдання проектного менеджера, проекти для розподілення

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі, методи дослідження, проектування інформаційної технології підтримки оперативного планування роботи проектного менеджера, розробка інформаційної технології

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність, мета та задачі, задачі дослідження, огляд існуючих аналогів, результати проведеного аналізу аналогів, функціональні вимоги, засоби реалізації, структурно-функціональне моделювання, моделювання варіантів використання, архітектура інформаційної технології, схема реалізованої бази даних засобами PostrgeSQL, демонстрація роботи програмного продукту, апробація, висновок.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Підготовка специфікації	20.09.2023	
2	Розробка інформаційної технології підтримки оперативного планування роботи проектного менеджера	20.11.2023	
3	Тестування	16.11.2023	
4	Впровадження в дію	01.12.2023	
5	Налаштування правильної роботи інформаційної технології підтримки оперативного планування роботи проектного менеджера	01.12.2023	

Магістрант _____ Дмитро РАЙКО

Керівник роботи _____ к.т.н., доц. Наталія ФЕДОТОВА

АНОТАЦІЯ

Темою кваліфікаційної роботи магістра є «Інформаційна технологія підтримки оперативного планування роботи проектного менеджера».

Пояснювальна записка складається зі вступу, 4 розділів, висновку, списку використаних джерел із 37 найменувань, 2 додатків. Загальний обсяг роботи – 92 сторінки, у тому числі 55 сторінок основного тексту, 3 сторінки списку використаних джерел, 34 сторінки додатків.

Кваліфікаційна робота магістра присвячена створенню інформаційної технології підтримки оперативного планування роботи проектного менеджера. Під час роботи було проведено аналіз предметної області, визначено мету та актуальність розробки. Було обрано відповідні технології для її розробки, встановлено цілі проекту, визначено основні завдання та описано використані методи дослідження. Робота включає процес проектування системи на різних етапах, враховуючи бізнес-потреби та можливі сценарії застосування програмного продукту. Також була здійснена програмна реалізація бази даних.

В результаті виконання даного дослідження була створена інформаційна технологія підтримки оперативного планування роботи проектного менеджера. Використання даної системи сприятиме покращенню процесу управління проектами та співробітниками, завдяки автоматизації процесу розподілу завдань поміж персоналу.

Використання запропонованої інформаційної системи сприятиме ефективності роботи як проектного менеджера, так і всієї команди, допомагаючи раціоналізувати робочий навантаження співробітників, що, у свою чергу, призведе до зниження стресового рівня в команді.

Ключові слова: AUTOMATION, ASSIGNMENT, KANBAN, PROJECT MANAGEMENT, TASK MANAGEMENT, TIME MANAGEMENT.

ЗМІСТ

ВСТУП

Актуальність. В сучасному бізнес-середовищі, де конкуренція посилюється і ринкові умови змінюються зі швидкістю світла, роль проектного менеджера стає вкрай важливою. Інформаційні технології вже давно стали невід'ємною частиною більшості організацій і галузей, і вони продовжують швидко розвиватися.

Сучасні проекти стають все складнішими і вимагають більше ресурсів, зусиль та координації. Інформаційні технології допомагають управляти цією складною ієрархією ефективно та з мінімальними ризиками, не втрачаючи сучасність та модернізм. Сучасний підхід до управління проектами вимагає більшої оперативності та швидкості, оскільки банально «Час - це гроші.».

Нові технології дають змогу отримувати дані в реальному часі дані про виконання завдань, що дозволяє проектному менеджеру швидко реагувати на зміни та вирішувати проблеми. Враховуючи ситуацію у світі та події останніх років не можна упустити той факт, що онлайн системи допомогли втриматися бізнесу, як під час тотального локдауну, так і під час війни. Саме подібні сервіси згуртовують та об’єднують людей з різних країн світу в одну єдину система, яка працює як одне ціле, та дозволяють ефективно управляти командами на відстані та забезпечують спільний доступ до необхідних ресурсів та інформації. Отже, інформаційна технологія підтримки оперативного планування роботи проектного менеджера стає необхідним інструментом для досягнення успіху в управлінні проектами в сучасному бізнес-середовищі.

Тема дослідження. Інформаційна технологія підтримки оперативного планування роботи проектного менеджера.

Об’єкт дослідження. Процес розподілу завдань серед персоналу IT компанії, враховуючи їх професію, кількість вільного часу та досвіду.

Предмет дослідження. Інформаційна технологія підтримки оперативного планування роботи проектного менеджера.

Мета. Розробка інформаційної технології підтримки оперативного планування роботи проектного менеджера.

Для досягнення мети проекту необхідно виконати наступні задачі:

– виконати аналіз проблемної області, провести огляд сучасних публікацій, визначити актуальність та цільову аудиторію використання даної технології;

– виконати аналіз аналогів інформаційних систем;

– провести аналіз та визначити технологій розробки даної інформаційної системи;

– спроектувати моделі та структури інформаційної системи;

– втілити структуру інформаційної технології;

– реалізувати функціонал;

– провести тестування програмного продукту.

Практична цінність. Оптимізація процесу оперативного планування роботи проектного менеджера, ефективний розподіл завдань серед команди IT компанії, враховуючи ключові параметри кожного з співробітників, за допомогою штучного інтелекту для аналізу робочих навантажень та оптимального розподілу завдань. Впровадження даної інформаційної технології сприятиме підвищенню продуктивності роботи проектного менеджера та команди в цілому, а також допоможе уникнути перевантаження співробітників, і в результаті знизити рівень стресу в колективі.

Гіпотеза дослідження. Впровадження інформаційної технології для операційного планування роботи проектного менеджера може значно підвищити ефективність управління проектами, забезпечуючи точніше прогнозування та оптимізацію використання ресурсів. Дослідження використання цієї технології може виявити оптимальні методи впровадження та підтримки оперативного планування, що сприятиме підвищенню продуктивності та досягненню успішних результатів у проектах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Інформаційні технології стали основним інструментом ведення бізнесу з економічного аспекту та з точки зору менеджменту. На даний момент неможливо уявити наше життя без нових застосунків, які можуть полегшувати повсякденні речі та процеси. Через це найпершим етапом переддипломної практики став огляд джерел і публікацій, в результаті чого стало можливим виділити актуальні потреби бізнесу та суспільства.

В сучасному проектному менеджменті інформаційні технології відіграють ключову роль, забезпечуючи ефективно управління, планування, контроль та аналіз проектів. Використання IT інструментів спрощує процеси і підвищує продуктивність команд, а також дозволяє проектним менеджерам своєчасно реагувати на зміни та виклики [1,2].

Саме від проектного керівника у сфері розробки IT-продуктів залежить досягнення успіху усього проекту. Тому інтеграція новітніх технологій та розробка передових методологій, які спрямовані на збільшення продуктивності проектного менеджера, мають значний вплив на успіх, високу якість виконання проекту, а також на ефективне управління проектами.

В роботі Кожемякіна В.А. вказано, що під час роботи над проектом проектний менеджер забезпечує не лише контроль над виконанням роботи та дотримання дедлайнів виконання завдань, а й повинен приймати рішення у розподіленні задач, виставленні пріоритетів на виконання, подолання викликів та проблем, які виникають під час роботи над проектом [3].

Всі проекти мають щось спільне, а саме стандартну структуру: кожен проект проходить через три основні етапи – старт, середина та фініш, що і являє собою життєвий цикл проекту.

Життєвий цикл проекту – це набір послідовних етапів, які проходять усі проекти від моменту їх ініціалізації до завершення, не беручи в увагу їх особливості. Метою життєвого циклу є створення зрозумілої та легкої у використанні структури для реалізації та управління проектами. Основними фазами життєвого циклу проекту складають: ініціація, планування, виконання, контролювання, та завершення [4]. В рамках цього циклу, різні методології розробки, такі як Agile, вносять свої особливості у кожен з цих етапів, пристосовуючи їх до конкретних потреб проекту та команди.

Agile Software Development є орієнтованою на гнучкість та швидку адаптацію до змін методологією, з акцентом на людському факторі та постійному залученні замовника у процес розвитку. Agile відрізняється своїм ітеративним підходом, де команда працює в коротких циклах, що дозволяє ефективно реагувати на зміни та забезпечувати високу якість кінцевого продукту. Такий підхід дозволяє мінімізувати ризики та забезпечує гнучкість у виконанні проектів, допомагаючи команді та замовникам досягати більш високих результатів [5,6]. Враховуючи ефективність та гнучкість Agile методології, саме її планується застосувати в майбутньому проекті.

Ефективний вибір члена команди для конкретного завдання може істотно знизити складності проекту. У статі запропоновано двофазний підхід до розподілу завдань, що включає ідентифікацію ключових факторів та залежностей, а також кількісний метод визначення найбільш підходящих членів команди для конкретних завдань, враховуючи їх здібності та компетенції. Врахування різних аспектів, таких як технічні навички, досвід та середовище, робить цей метод ефективним для розподілу завдань [7].

Ручне призначення завдань є обмеженим, та неефективним у великих проектах з розробки програмного забезпечення, так як вимагає багато часу та має схильність до людського фактору, що означає допущення помилок при розподілі завдань між співробітниками [8].

Тож, для автоматизації розподілу завдань необхідно застосувати штучний інтелект(ШІ). Моделі ШІ, такі як оптимізація за допомогою алгоритму

мурашиних колоній, відзначаються здатністю зменшувати складність прийняття рішень через ітеративні мережі. Також, ефективними є алгоритми баєсовських мереж, відомі своєю здатністю обробляти великі набори даних із невизначеними та суб'єктивними судженнями. Ці моделі інтегрують різні рівні причинно-наслідкових зв'язків, використовуючи стохастичні моделі для прогнозування майбутніх умов [9].

1.2 Аналіз інформаційних систем

Реалії IT-бізнесу такі, що важко уявити навіть звичайні процеси без інформаційних рішень. Такі важливі речі, як планування проектами та керування персоналом потребують онлайн сервіси, які можуть спростити та прискорити відповідні дії з сторони управління компанії, тобто проектного менеджера, керівника або ж власника компанії.

Такі інформаційні рішення-помічники, допомагають спрощували планування робіт розповсюджені та популярні серед усіх IT-компаній, оскільки без них поставлені процеси неможливо вже представити.

Задля порівняння було відібрано три інформаційні системи планування робіт. А саме інформаційна система «Trello», «Monday.com» та «Microsoft Project».

«Trello» – це багато платформна система, яка була розроблена компанією Trello Enterprise та має пряме призначення для управління проектами (рис. 1.1). Система має широкий функціонал, сучасний дизайн та можливість інтеграції інших сервісів від Atlassian [10].

Серед основних функцій додатку можна виділити такі:

– додавання, редагування та видалення завдань;
– є можливість додавати різні статуси для проектів (дошок);

– відображення завдань у вигляді Kanban-дошок, що повноцінно сумісне з методологією розробки Agile;

– можливість переглядати список своїх завдань у вигляді таблиці чи календаря;

– управління учасниками робочого простору та їх привілеями.

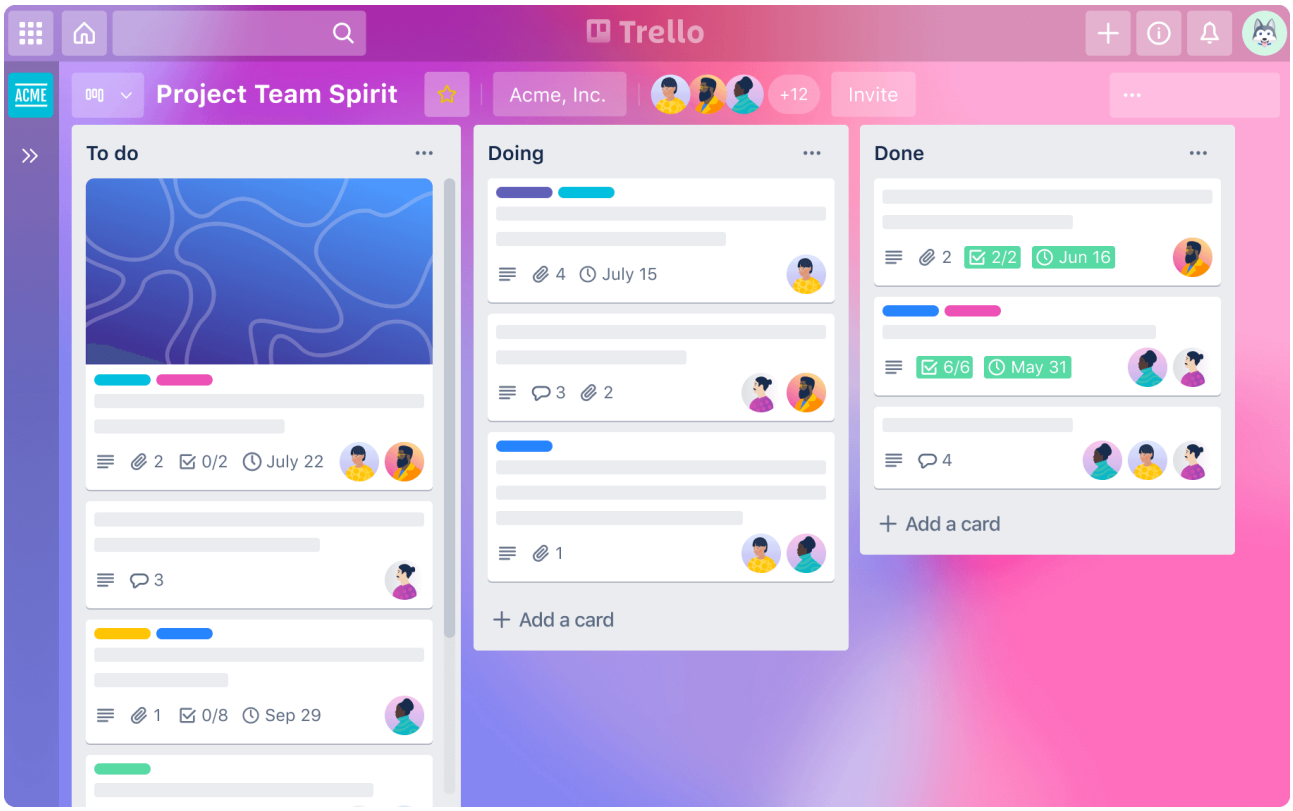


Рисунок 1.1 – Головна сторінка проекту в системі «Trello»

Джерело: [11]

З погляду дизайну ця система є сучасною та інтуїтивно зрозумілою та надає всі можливості для менеджменту проектів та підтримки завдань без сильного ознайомлення вивчення, тобто приступити до роботи можна в короткий час після реєстрації.

Після дослідження даного web-рішення, можна зробити висновок, що сервіс справді має широкий набір основних функцій, але не забезпечує автоматизованого розподілу задач між співробітниками команди.

Наступна web-орієнтована платформа для управління проектами та командної роботи «Monday.com» призначена для підвищення продуктивності, спрощення планування проектів, підвищення прозорості робочих процесів і покращення комунікації всередині команд(рис. 1.2) [12].

Серед функцій web-платформи можна виділити:

– візуалізація проектів для ефективного планування;

– можливість призначати завдання, встановлювати терміни та відстежувати прогрес;

– інтеграція інструментів для обміну повідомленнями, коментарями до завдань і спільної роботи;

– засоби для створення звітів і аналізу даних для кращого розуміння продуктивності та ефективності проектів;

– створення налаштованих правил, які автоматично запускають дії на основі певних умов.

Рисунок 1.2 – Функціональні можливості платформи «Monday.com»

Джерело: [13]

Серед переваг даної веб-платформи можна виокремити сучасний вигляд, гнучкість налаштування, інтуїтивно зрозумілий інтерфейс, а також автоматизація робочих процесів. Але, функціонал автоматизації не забезпечує застосування штучного інтелекту, а просто прописує правила та встановлює тригери.

Microsoft Project – це система управління проектами, яка надає користувачам широкий набір можливостей для планування, координації та відстеження процесу розробки проекту [14]. Головне вікно Microsoft Project зображене на рисунку 1.3.

Основними функціональними можливостями Microsoft Project є:

– створення детальних розкладів проекту;
– управління ресурсами проекту;
– можливість встановлювати бюджети та витрати в рамках проекту;
– аналіз даних проекту;
– управління ризиками проекту.

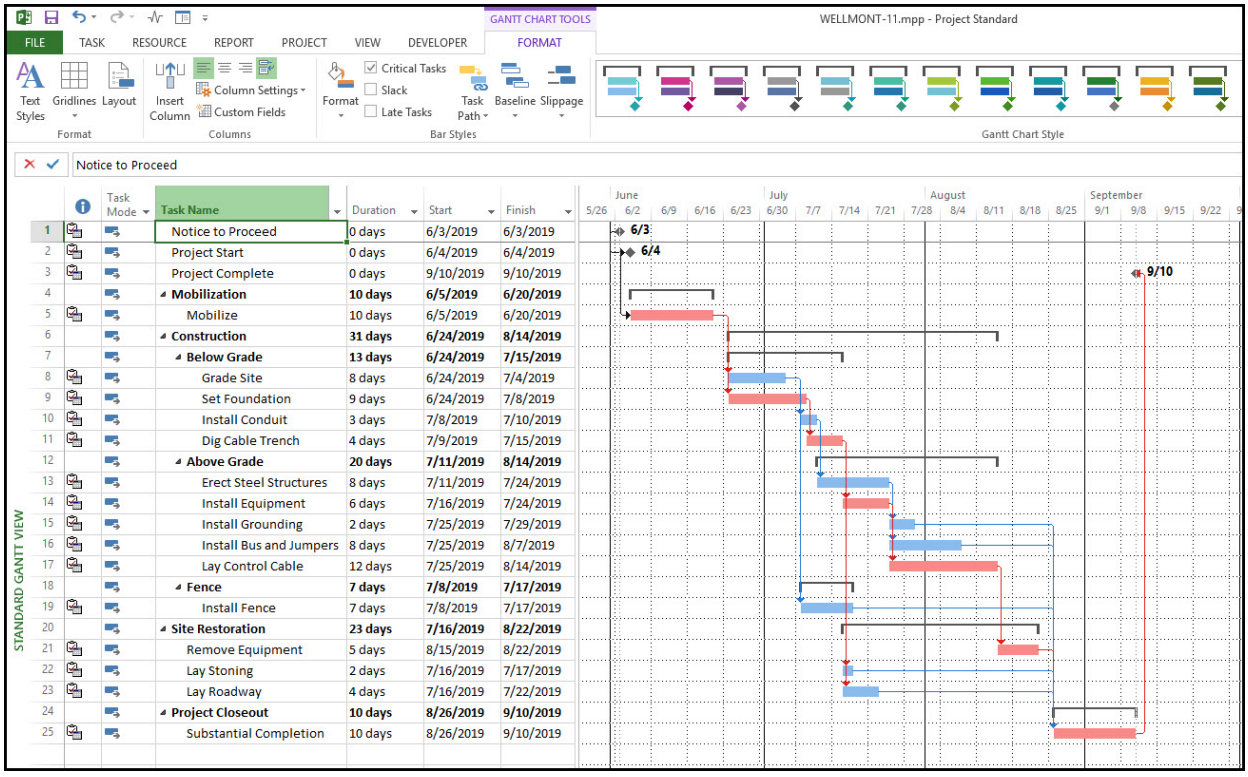


Рисунок 1.3 – Головний екран Microsoft Project

Джерело: [15]

Дизайн додатку відповідає передовим тенденціям, є лаконічним та інтуїтивно-зрозумілим.

Серед переваг даного програмного продукту є гнучкість у візуалізації даних, ефективне управління ресурсами, та розширені можливості створення детальних звітів, проте Microsoft Project не забезпечує автоматизації розподілу ресурсів та завдань.

Результатом здійснення порівняння аналогів є таблиця 1.1.

Таблиця 1.1 – Порівняльна таблиця аналогів інформаційних систем

Характеристика	**Trello**	**Monday.com**	**MS Project**
Сучасний дизайн	+	+	+
Зручний UI/UX	+	-	+
Управління проектами	+	+	+
Управління плануванням задач проекту	+	+	+
Автоматизований розподіл задач	-	-	-
Можливість формування звіту	-	+	+
Візуалізація проектів різними способами	-	+	+

Джерело: побудовано автором

Проаналізувавши інформацію з таблиці 1.1 можна визначити функціональні можливості, які необхідно буде втілити під час розробки, а також недоліки, яких слід уникнути. Отже, інформаційній технології необхідно мати наступні риси: інтуїтивно-зрозумілий сучасний інтерфейс, забезпечувати управління проектами, планування задач проекту, а також автоматично розподіляти задачі поміж виконавців.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою кваліфікаційної роботи магістра є розробка інформаційної технології підтримки оперативного планування роботи проектного менеджера. Розроблений програмний продукт буде корисний для ІТ компаній, а саме для проектних менеджерів, для ефективного керування проектами, для команди розробників, для кращого розуміння своїх ролей і обов'язків. Також, ця технологія може бути також корисною для фрілансерів або малих команд, які шукають ефективні інструменти для самоорганізації та планування своєї роботи. Втілення розроблюваної системи та подальше її використання в ІТ компанії дозволить автоматизувати процес розподілення задач поміж командою, та покращить ефективність роботи над розробкою ПЗ та процеси в середині команди розробки.

Розроблена інформаційна технологія повинна задовольняти наступним вимогам до функціональності:

– створення, редагування та видалення проектів;

– створення, редагування та видалення завдань;

– проведення автоматичного розподілу завдань між членами команди;

– візуалізація проектів у вигляді Kanban дошки, календаря та звичайного списку [16].

Розроблена інформаційна технологія має також бути зі зручним користувацьким інтерфейсом з оптимальним застосуванням візуальних ефектів. Результати розподілу мають бути відображені з високою швидкістю на сторінці.

Для виконання мети цього проекту необхідно втілити такі завдання:

– провести аналіз існуючих аналогів інформаційних технологій;

– здійснити детальний аналіз та визначення технологій для розробки програмного продукту;

– розробити структуру та модель інформаційної технології;

– реалізувати структуру інформаційної системи;

– реалізувати та інтегрувати функціональні можливості інформаційної технології підтримки оперативного планування роботи проектного менеджера;

– провести тестування інформаційної технології.

Під час встановлення мети проекту було здійснено планування проектних робіт, деталі якого представлені у Додатку А.

2.2 Методи дослідження

Для вирішення основної задачі системи, а саме автоматичного розподілу завдань серед членів команди, було проведено дослідження різних методів штучного інтелекту та обробки даних.

Першим методом було розглянуто Neural Networks, суть якого полягає в тому, щоб глибоке навчання та нейронні мережі використовуються для ідентифікації складних шаблонів та зв’язків між даними. Даний підхід може бути використаний для аналізу попередньої продуктивності та рекомендацій щодо найкращих кандидатів для виконання відповідного завдання [17].

Методи Random Forest та Gradient Boosting використовуються для поєднання декількох моделей прогнозування для підвищення точності під час призначення виконавців. Ці алгоритми базуються на тому, що кожне дерево рішень навчається на випадковій підмножині даних і в результаті використовується для прийняття рішень. В контексті підтримки оперативного планування дані методи можуть використовуватися для призначення виконавців, оцінюючи їхні навички та досвід на основі історії [18,19].

Наступним методом було розглянуто обробку природної мови (NLP). Цей метод відноситься до контексту штучного інтелекту. Він займається вирішенням завдань, пов'язаних з аналізом (розумінням) та синтезом (генерацією) природної

мови. Ключовими завданнями, які є важливими при розробці інформаційних технологій, є розпізнавання та ідентифікація завдань, пошук інформації, видобування даних, а найголовніше - встановлення зв’язків та визначення відносин між об’єктами в тексті, наприклад, визначення хто і з чим працює [20].

Алгоритм Кластеризації – це метод, який дозволяє групувати учасників проекту за їхніми навичками та досвідом. Тобто спростити вибір кандидатів до завдань, оскільки вони можуть бути розподілені в групи за подібністю [21].

Обираючи між цими методами, я обрав NLP, бо цей метод має вагомі переваги. Перш за все, цей підхід відзначається аналізом і розумінням текстової інформації, що дозволяє ідентифікувати завдання, встановлювати зв’язки між об’єктами та визначити залежності між різними наборами даних. Крім того, Обробка природної мови може спростити взаємодію з командою та вдосконалювати процес прийняття рішень.

2.3 Вибір технологій

Після того, як був проведений аналіз актуальності роботи та визначений тип ПЗ, треба визначитись з технологією реалізації даної інформаційної технології.

Для програмної реалізації інформаційної технології було обрано наступні технології: бібліотеку React.js для створення інтерактивного інтерфейсу [22], бібліотеку styled-component – для надання web-сторінці адаптивності [23], Express.js – програмний каркас розробки серверної частини вебзастосунків для Node.js [24,25], Docker – як основа для розгортання програмного продукту [26], мову Python – для аналізу даних та розгортання штучного інтелекту [27], та об’єктно-орієнтовану базу даних PostgreSQL для організації та збереження даних [28].

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

Після детального дослідження предметної області, уточнення актуальності та цілей розробки, а також розробки плану дій для реалізації проекту, наступним кроком є розробка зазначеної інформаційної технології. В процесі цього були розроблені схеми в нотації IDEF0, що докладно відображають послідовність процесів, задіяних у автоматизації розподілу задач між співробітниками.

3.1 Діаграми нотації IDEF0

Під час створення функціональної моделі розроблюваної інформаційної системи, використовуються різні техніки структурного та об’єктно-орієнтованого аналізу. Серед них, одним із ключових методів є IDEF0, який застосовується для ефективного моделювання.

Діаграма нотації IDEF0 є інструментом функціонального моделювання, який використовується для візуалізації робочих процесів та систем. Ця нотація зосереджується на функціональних аспектах системи або процесу, забезпечуючи чітке уявлення про те, як функції взаємодіють між собою та з зовнішнім середовищем [29].

Основні елементи діаграми IDEF0 включають:

– механізми;
– вхідні дані;
– вихідні дані;
– управління.

Діаграми IDEF0 часто використовуються для аналізу існуючих процесів при виявленні можливостей для їх оптимізації, а також для розробки нових

систем чи процесів, оскільки вони допомагають визначити ключові функції та взаємозв'язки між ними.

Процес «Розподілення задач поміж співробітниками» має наступні дані:

– вхідні дані: інформація про співробітника, інформація про навички співробітника, інформація про завдання, інформація про проект.

– вихідні дані: назначені на виконання завдань співробітники.

– управління: обмеження ресурсів та бюджету, життєвий цикл розробки ПП, політика компанії.

– механізми: інформаційна технологія, апаратне забезпечення, користувач, алгоритми розподілу.

На рисунку 3.1 зображена контекстна діаграма IDEF0 інформаційної технології з розподілення задач проекту між співробітниками компанії.

Рисунок 3.1 – Контекстна діаграма IDEF0

Джерело: побудовано автором

Для деталізації процесів інформаційної технології було виконано декомпозицію першого рівня для процесу розподілу задач проекту поміж співробітниками(рис. 3.2).

Дані для діаграми наступні:

– вхідні дані: вимоги до проекту, специфіка проекту, інформація про співробітника, інформація про навички співробітника, інформація про завдання, інформація про проект..

– вихідні дані: назначені на виконання завдань співробітники.

– управління: обмеження ресурсів та бюджету, життєвий цикл розробки ПП, політика компанії.

– механізми: інформаційна технологія, апаратне забезпечення, користувач, алгоритми розподілу.

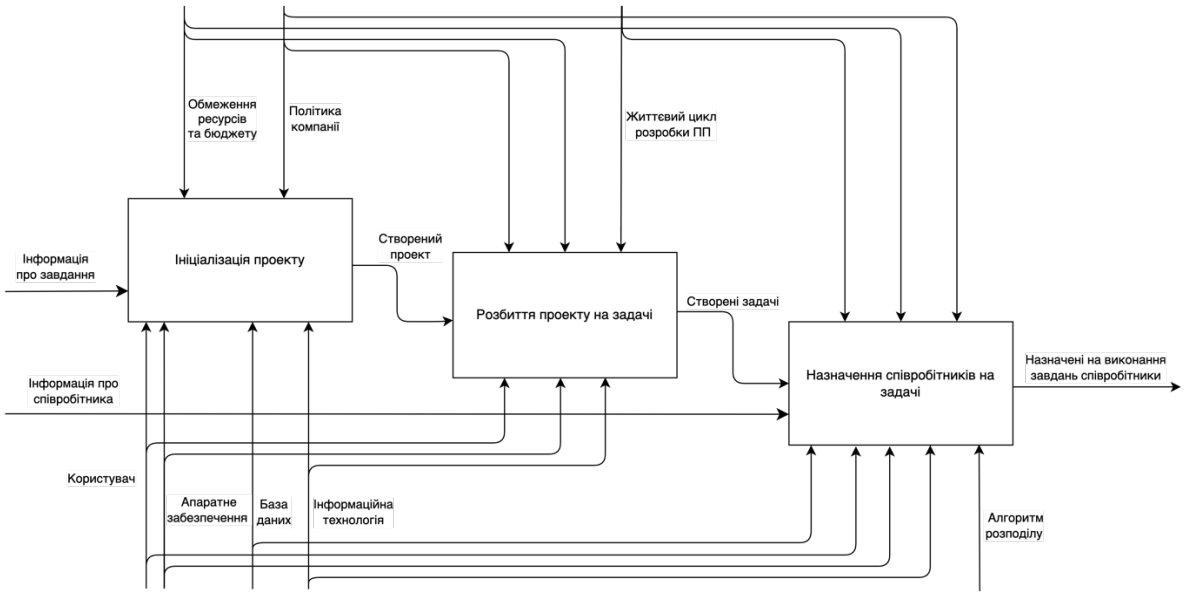


Рисунок 3.2 – Діаграма декомпозиції першого рівня

Джерело: побудовано автором

Для деталізації процесу призначення співробітників на певні задачі було виконано декомпозицію діаграми першого рівня(рис. 3.3).

Дані для діаграми наступні:

– вхідні дані: вимоги до проекту, специфіка проекту.

– вихідні дані: назначені на виконання завдань співробітники.

– управління: обмеження ресурсів та бюджету, життєвий цикл розробки ПП, політика компанії.

– механізми: інформаційна технологія, апаратне забезпечення, користувач, алгоритми розподілу.

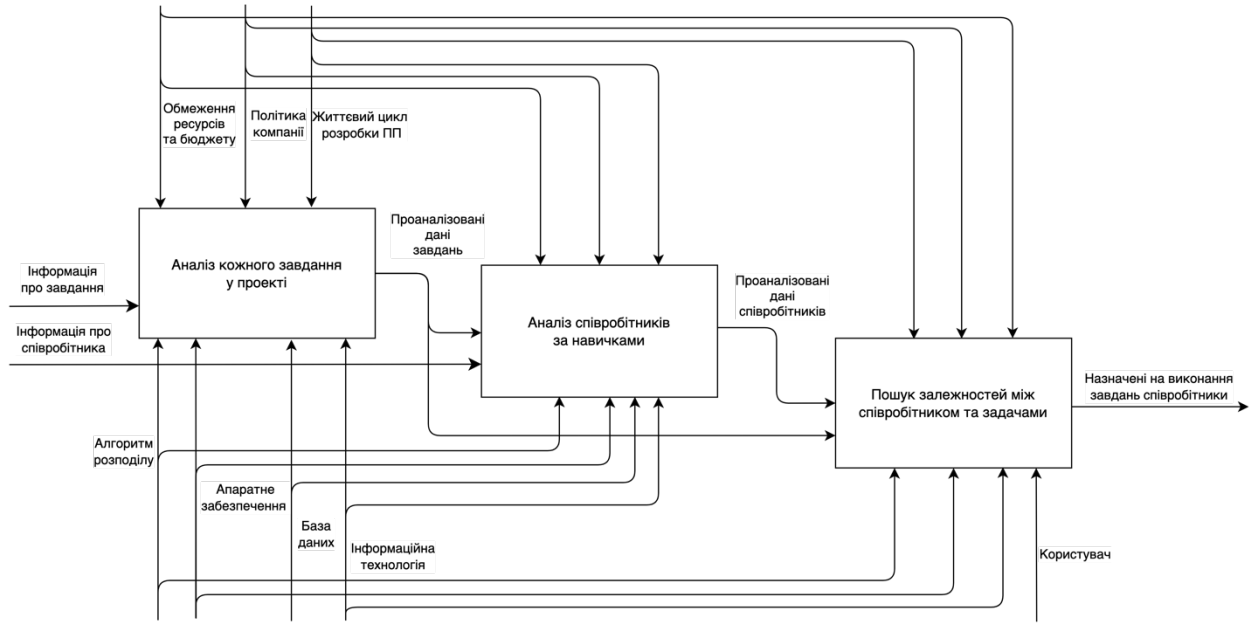


Рисунок 3.3 – Діаграма декомпозиції процесу розподілення завдань між користувачами

Джерело: побудовано автором

3.2 Use Case діаграма

Після завершення етапу моделювання важливо розробити діаграму варіантів використання(Use Case diagram). Діаграма варіантів використання є важливим інструментом в області проектування ПП, особливо в контексті об’єктно-орієнтованого аналізу та проектування. Use Case діаграма допомагає візуалізувати функціональні вимоги системи і відносини між різними користувачами(акторами) та системою [30]. Основними компонентами діаграми варіантів використання є актори, відносини, системні межі, та самі варіанти використання. Діаграми Use Case посилюють розуміння функціоналу системи, допомагають визначити користувацькі вимоги, сприяють спілкуванню між розробниками та зацікавленими сторонами.

Для інформаційної технології підтримки оперативного планування роботи проектного менеджера виділяють наступні use case-и:

– додавання та видалення співробітників компанії;
– створення, редагування та перегляд опису проекту;
– редагування учасників проекту;
– розбиття проекту на підзавдання;
– створення, редагування та перегляд завдань.

Акторами діаграми варіантів використання є власник компанії, проектний менеджер та працівник. Діаграма використання інформаційної технології підтримки оперативного планування роботи проектного менеджера зображена на рисунку 3.4.

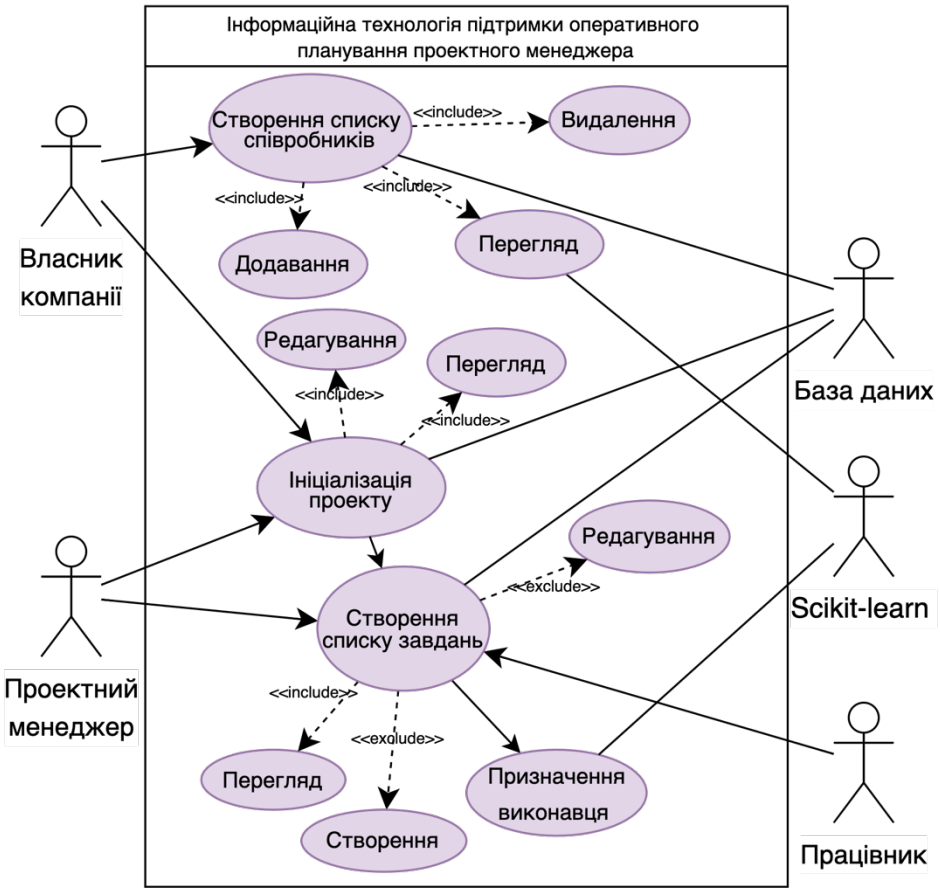


Рисунок 3.4 – Use Case діаграма

Джерело: побудовано автором

3.3 Проектування моделі бази даних

Моделювання бази даних – це ключовий етап у створені системи зберігання та обробки інформації. Цей процес включає визначення та організацію даних, а також встановлення взаємозв’язків між ними. Завдяки ретельному виконання даної фази розробки, можна забезпечити високу ефективність і широкий функціонал в інформаційній технології. Усі логічні моделі баз даних складаються з трьох основних компонентів: сутностей, відносин та атрибутів.[31]

Сутність це базовий об’єкт, який представляє набір речей, або ж понять, які безпосередньо пов’язані предметною областю. Зв’язок між різними сутностями встановлюється на основі залежності одного стеку даних від інших. Атрибути, в свою чергу, представляють властивості та характеристики, розширюючи опис сутності.

В процесі моделювання було виділено наступні сутності:

– користувач (User);
– токени авторизації (Token);
– проєкт (Project);
– роль, або ж посада користувача (Role);
– член проєкту (Staff);
– завдання (Task);
– статус (Status);
– документ (Document).

Структура бази даних представлена в таблиці 3.1.

Таблиця 3.1 – Порівняльна таблиця аналогів інформаційних систем

Таблиця	**Властивість**	**Тип**	**Ключі**	**Обмеження**	**Призначення**
User	id	serial	PK	NOT NULL	Унікальний ідентифікатор запису
	email	varchar(255)		NOT NULL, Unique	Пошта, за якою буде відбуватися авторизація та надаватися доступ до акаунта

Продовження таблиці 3.1

Таблиця	Властивість	Тип	Ключі	Обмеження	Призначення
	password	varchar(255)		NOT NULL	Пароль
	name	varchar(255)		NOT NULL	Ім’я користувача
	surname	varchar(255)		NOT NULL	Прізвище користувача
	type	varchar(255)			Тип користувача
	actiovationLink	varchar(255)			Посилання для активації акаунта (hash)
	isActivated	boolean			Індикатор, який відповідає за те, чи активований акаунт
	Image	varchar(255)			Посилання за яким зберігається фото профіля
	roleId	integer	FK		Роль (посада) користувача
	createdAt	timestamp with time zone		NOT NULL	Дата та час створення запису
	updatedAt	timestamp with time zone		NOT NULL	Дата та час оновлення запису

Продовження таблиці 3.1

Таблиця	Властивість	Тип	Ключі	Обмеження	Призначення
	notes	text			Короткий опис про користувача (Про себе)
	skills	text			Навички користувача
	keywords	text			Ключові слова для відповідного користувача
	isRestricted	boolean			Ідентифікатор, який визначає чи видалений користувач
Token	id	serial		NOT NULL	Унікальний ідентифікатор запису
	token	text		NOT NULL	Хешоване значення, яке містить дані користувача
	createdAt	timestamp with time zone		NOT NULL	Дата та час створення запису
	updatedAt	timestamp with time zone		NOT NULL	Дата та час оновлення запису
	userId	integer	FK		Ідентифікатор користувача

Продовження таблиці 3.1

Таблиця	Властивість	Тип	Ключі	Обмеження	Призначення
Project	id	serial	PK	NOT NULL	Унікальний ідентифікатор запису
	name	varchar(255)		NOT NULL	Назва проєкту
	description	Name			Опис проєкту
	createdAt	timestamp with time zone		NOT NULL	Дата та час створення запису
	updatedAt	timestamp with time zone		NOT NULL	Дата та час оновлення запису
	attachaments	integer[]			Масив приєднаних додатків до проєкту
Role	id	serial		NOT NULL	Унікальний ідентифікатор запису
	name	varchar(255)		NOT NULL	Назва ролі (посади)
	roles	varchar(255)[]			Масив дозволів до функціоналу системи
	createdAt	timestamp with time zone		NOT NULL	Дата та час створення запису

Продовження таблиці 3.1

Таблиця	Властивість	Тип	Ключі	Обмеження	Призначення
	updatedAt	timestamp with time zone		NOT NULL	Дата та час оновлення запису
	keywords	text			Ключові слова та фрази, які відносяться до відповідної посади
Staff	userId	integer	FK	NOT NULL	Ідентифікатор користувача
	projectId	integer	FK	NOT NULL	Ідентифікатор проєкту, до якого відноситься користувач
	id	serial	PK	NOT NULL	Унікальний ідентифікатор запису
Task	id	serial	PK	NOT NULL	Унікальний ідентифікатор запису
	title	varchar(255)		NOT NULL	Назва завдання
	description	text			Опис завдання
	parentTaskId	integer	FK		Ідентифікатор батьківського завдання
	type	varchar(255)		NOT NULL	Тип завдання

Продовження таблиці 3.1

Таблиця	Властивість	Тип	Ключі	Обмеження	Призначення
	createdAt	timestamp with time zone		NOT NULL	Дата та час створення запису
	updatedAt	timestamp with time zone		NOT NULL	Дата та час оновлення запису
	assignId	integer	FK		Ідентифікатор члена команди, який закріплений за завданням
	statusId	integer	FK		Ідентифікатор статусу
	attachments	integer[]			Масив приєднаних додатків до завдання
	fromDate	timestamp without time zone			Дата та час створення початку виконання завдання
	dueDate	timestamp without time zone			Дата та час створення закінчення виконання завдання

Продовження таблиці 3.1

Таблиця	Властивість	Тип	Ключі	Обмеження	Призначення
	projectId	integer			Ідентифікатор проєкту, до якого відноситься завдання
	createdBy	integer			Ідентифікатор члена команди, який створив запис
Status	id	serial	PK	NOT NULL	Унікальний ідентифікатор запису
	name	varchar(255)		NOT NULL	Ім’я статусу
	createdAt	timestamp with time zone		NOT NULL	Дата та час створення запису
	updatedAt	timestamp with time zone			Дата та час оновлення запису
	color	varchar(20)			Колір, який використовуєтьс я для легкої ідентифікації статусу
	description	varchar(255)			Опис статусу

Продовження таблиці 3.1

Таблиця	Властивість	Тип	Ключі	Обмеження	Призначення
	ordering	integer			Значення, яке відповідає за порядок відображення статусів
Document	id	serial	PK	NOT NULL	Унікальний ідентифікатор запису
	type	varchar(255)		NOT NULL	Тип документа
	title	varchar(255)		NOT NULL	Назва документу
	link	varchar(255)		NOT NULL	Посилання, за яким можна отримати доступ до документу
	createdAt	timestamp with time zone		NOT NULL	Дата та час створення запису
	updatedAt	timestamp with time zone		NOT NULL	Дата та час оновлення запису

Джерело: побудовано автором

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПІДТРИМКИ ОПЕРАТИВНОГО ПЛАНУВАННЯ РОБОТИ ПРОЕКТНОГО МЕНЕДЖЕРА

4.1 Архітектура інформаційної технології

Процес розробки програмного продукту включає в себе проектування архітектури. Інформаційна технологія була створена на мікросервісній архітектурі, що складається з клієнтської та серверної частини додатку. В даному випадку за front-end відповідає бібліотека React.js [32]. Back-end в свою чергу має складнішу архітектуру, оскільки вона об’єднує основний сервер Node.js та сервіс, побудований на мові програмування Python.

Архітектура інформаційної технології зображена на рисунку 4.1

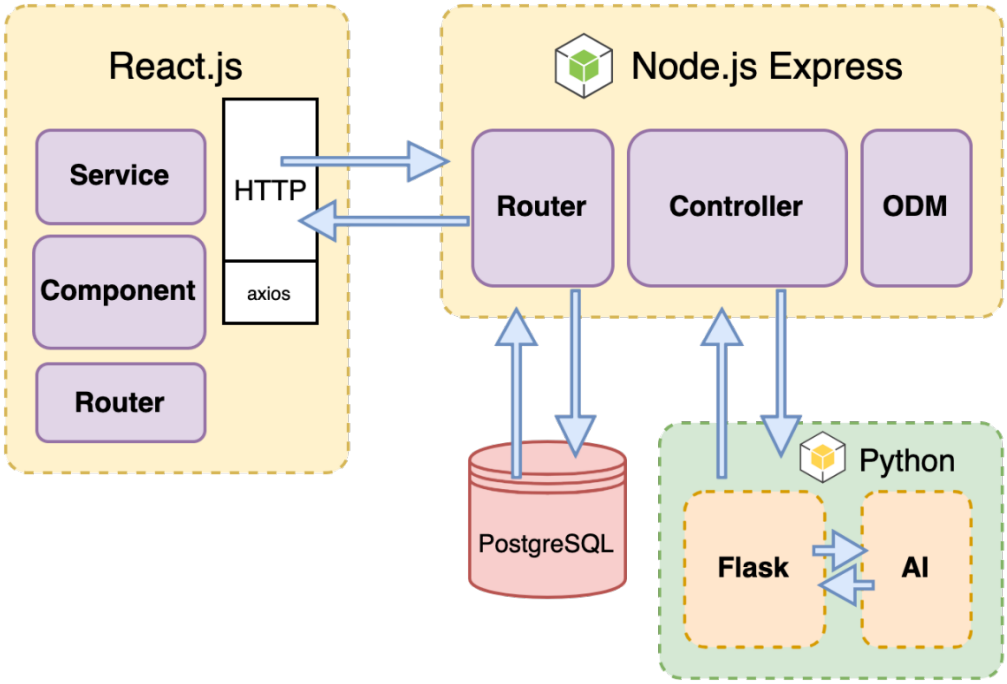


Рисунок 4.1 – Архітектура IT

Джерело: побудовано автором

4.2 Реалізована база даних

Було реалізовано базу даних згідно попереднього проектування її структури. На рисунку 4.2 представлена логічна модель даних для інформаційної технології підтримки оперативного планування роботи проектного менеджера.

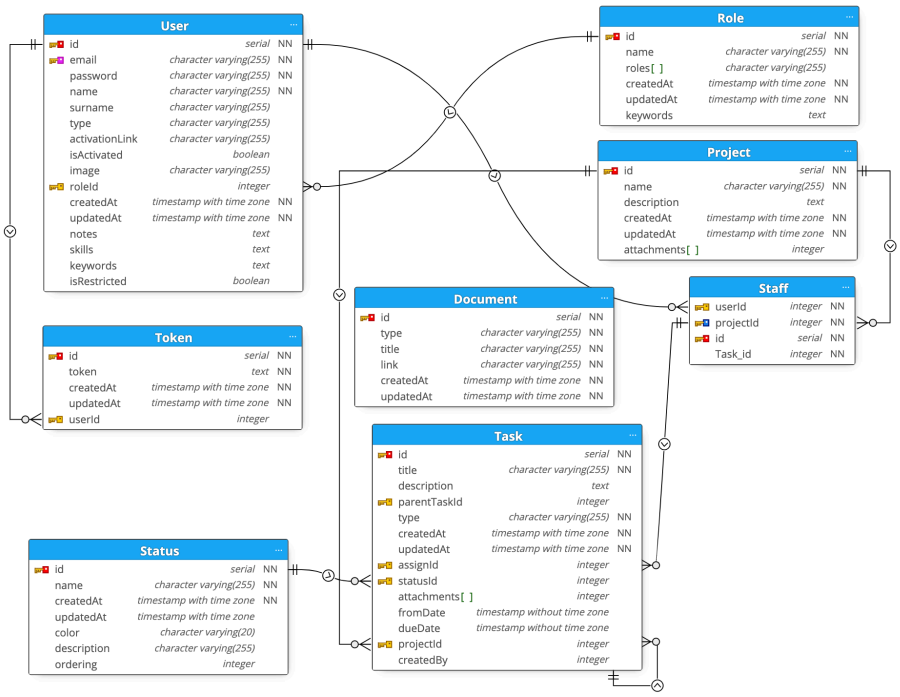


Рисунок 4.2 – Реалізована модель бази даних засобами PostgreSQL

Джерело: побудовано автором (знімок з екрану)

4.3 Реалізація інформаційної технології

Даний етап життєвого циклу розробки програмного продукту було вирішено розпочати з створення бази даних та серверної частини застосунку. Оскільки було обрано технології PostgreSQL для база даних та Node.js для створення back-end частини IT, то спочатку було створено docker-compose файл,

який містить усі потрібні налаштування бази даних та конфігурації для правильної роботи мейнфрейму.

На рисунку 4.3 зображено вміст файлу docker-compose.yml.

Рисунок 4.3 – Вміст файлу docker-compose.yml

Джерело: побудовано автором (знімок з екрану)

Наступним кроком було розроблено початкову базу даних, яка складалась з декількох основних таблиць. На рисунку 4.4 можна переглянути частину коду, що містила першу міграцію

```
await queryInterface.createTable( tableName: 'User', attributes: {
  id: {
    type: Sequelize.INTEGER,
    allowNull: false,
    primaryKey: true,
    autoIncrement: true,
  },
  email: { type: Sequelize.STRING, allowNull: false, unique: true },
  password: { type: Sequelize.STRING, allowNull: false },
  name: { type: Sequelize.STRING, allowNull: false },
  surname: { type: Sequelize.STRING },
  type: { type: Sequelize.STRING },
  activationLink: { type: Sequelize.STRING },
  isActivated: { type: Sequelize.BOOLEAN, defaultValue: false },
  image: { type: Sequelize.STRING },
  createdAt: { type: Sequelize.DATE, allowNull: false },
  updatedAt: { type: Sequelize.DATE, allowNull: false },
});
```

Рисунок 4.4 – Перша міграція до БД

Джерело: побудовано автором (знімок з екрану)

Для правильної взаємодії між сервером та базою даних було вирішено використовувати доволі популярну бібліотеку Sequelize, яка включає в себе міграції, моделі та імітує основні запити до бази даних. Модель – це структурований файл, який описує усі властивості та залежності однієї сутності. Іншими словами, це таблиця бази даних, яка описана кодом [33]. Приклад такої моделі зображено на рисунку 4.5.

```
module.exports = (sequelize) => {
  const User = sequelize.define(
    'User',
    {
      email: {
        type: DataTypes.STRING,
        allowNull: false,
        set(v) {
          this.setDataValue('email', v.toLowerCase());
        },
        unique: {
          args: true,
          msg: 'Oops. Looks like you already have an accoun
        },
        validate: {...},
      },
      password: {
        type: DataTypes.STRING,
        allowNull: false,
      },
      name: {
        type: DataTypes.STRING,
        allowNull: false,
        validate: {
          notEmpty: true,
        },
      },
      surname: DataTypes.STRING,
      type: {
        type: DataTypes.STRING,
      },
```

Рисунок 4.5 – Файл моделі таблиці User

Джерело: побудовано автором (знімок з екрану)

На даному зображені можна побачити частину файлу з моделлю користувача (User) в базі даних.

Наступним етапом є розробка основної логіки сервера та імплементації функціоналу для обробки специфічних запитів до відповідних ендпойнтів.

Як приклад далі було розглянуто логіку відповідей на всі запити user напрямку. На рисунку 4.6 можна побачити основні маршрути різних запитів.

```
const { Router } = require('express');
const {
  authMiddleware,
  asyncMiddleware,
} = require('../middlewares');
const { initialState, updateUser } = require('../controllers/user');

const router = Router();

router.use(authMiddleware);

router.get(
   path: '/initial-state',
  asyncMiddleware( fn: async (req, res) => initialState(req, res)),
);
router.put(
   path: '/:userId',
  asyncMiddleware( fn: async (req, res) => updateUser(req, res)),
);
router.put(
   path: '/:userId',
  asyncMiddleware( fn: async (req, res) => updateUser(req, res)),
);

module.exports = router;
```

Рисунок 4.6 – Файл для user маршруту

Джерело: побудовано автором (знімок з екрану)

На рисунку 4.7 зображено контролери для вищевказаних запитів. Можна побачити основну логіку та запити до бази даних через відповідні моделі.

```
const { Sequelize } = require('sequelize');
const {
  models: {
    Projects, User, Role, Task, Status,
  },
} = require('../models');

module.exports = {
  initialState: async (req, res) => {
    const { id: userId, currentProject } = req.session;

    const projects = await Projects.findAll({
      attributes: ['id', 'name', 'description', 'attachments'],
      raw: true,
    });

    let currPrjId = currentProject;
    if (!currPrjId && projects[0]) {
      currPrjId = projects[0].id;
    }

    const account = await User.findOne({
      where: { id: userId },
      attributes: ['id', 'roleId', 'name', 'surname', 'image', 'isActivated', 'email', 'type', 'not
      raw: true,
    });

    const itemStaff = await User.findAll({
      attributes: ['id', 'roleId', 'name', 'surname', 'image', 'isActivated', 'email', 'type', 'not
      raw: true,
      order: ['name', 'surname'],
      where: {
        isRestricted: false,
      },
    });

    const currentStaff = currPrjId ? await User.findAll({
      attributes: ['id', 'roleId', 'name', 'surname', 'image', 'isActivated', 'email', 'type', 'not
        [Sequelize.col( col: 'Projects.Staff.id'), 'staffId'],
        [Sequelize.col( col: 'Projects.id'), 'projectId'],
```

Рисунок 4.7 – User контролер

Джерело: побудовано автором (знімок з екрану)

Далі було розроблено front-end частину застосунку. Як було вказано раніше, було обрано бібліотеку React.js. За допомогою неї можна якісно та професіонально побудувати web-системи різного роду складності. Для полегшення та вдосконалення структури коду було використано бібліотеку styled-component, яка дозволяє писати стилі до компонентів прямо в javascript файлах, або ж там, де вони повинні бути розміщені.

На рисунку 4.8 зображено частину коду, яка відповідає за одну з сторінок інформаційної технології.

```
class KanbanPage extends Component {
  onDragOver = (ev) => {
    ev.preventDefault();
  };

  onDragStart = (ev, name) => {
    ev.dataTransfer.setData('id', name);
  };

  onDrop = (ev, statusId) => {
    const taskId = ev.dataTransfer.getData('id');

    this.props.saveTask(taskId, statusId);
  };

  render() {
    return (
      <RootWrapper>
        {this.renderStatusAreas()}
      </RootWrapper>
    );
  }

  renderStatusAreas = () => {
    const { status } = this.props;

    return status.map((s) => {
      const statusId = s.get('id');
      return (
        <StyledArea
          key={`status-area-${statusId}`}
          onDragOver={(e) => this.onDragOver(e)}
          onDrop={(e) => this.onDrop(e, s.get('id'))}
        >
          <StatusWidget primary statusId={statusId} />
          {this.renderStatusTasks(statusId)}
        </StyledArea>
      );
    });
  };
```

Рисунок 4.8 – Частина коду сторінки Kanban

Джерело: побудовано автором (знімок з екрану)

React, як сучасний фреймворк має свої привілеї архітектури і взаємодії інтерфейсу з користувачем. Одним з них є інтерактивність на сайті з зчитуванням нових даних без перезавантаження сторінок. Це досягається тим, що при рендерингу головний процес не взаємодіє з DOM-деревом сайту, він має віртуальну копію, і спочатку змінює її. Для того, щоб технологія підтримувала

мультисторінкову структуру, потрібно використовувати спеціальний розділ бібліотеки, а саме react-router-dom [34].

На рисунку 4.9 можна переглянути розподіл маршрутів між основними сторінками інформаційної технології.

```
import { KanbanPage } from '../pages/kanban';
import { ProjectEditPage, ProjectPage, ProjectSettingsPage } from '../pages/project';
import { TaskEditPage, TaskPage } from '../pages/task';
import { UserEditPage, UserPage } from '../pages/user';
import { CompanySettingsPage } from '../pages/company';

const Routers = () => {
  const isLoading = useSelector( selector: ({ ui }) => ui.get('isFetching'));

  if (isLoading) {
    return <div> Loading ...</div>;
  }

  return (
    <Routes>
      <Route path="/" element={<PrivateRoute> <Wrapper /></PrivateRoute>}>
        <Route index element={<WorkspacePage/> } />
        <Route path={ROUTE_WORKSPACE} element={<WorkspacePage/>} />
        <Route path={ROUTE_KANBAN} element={<KanbanPage />} />

        <Route path={ROUTE_COMPANY_SETTINGS} element={<CompanySettingsPage />} />

        <Route path={ROUTE_PROJECT} element={<ProjectPage />} />
        <Route path={ROUTE_PROJECT_EDIT} element={<ProjectEditPage />} />
        <Route path={ROUTE_PROJECT_SETTINGS} element={<ProjectSettingsPage />} />

        <Route path={ROUTE_USER} element={<UserPage />} />
        <Route path={ROUTE_USER_EDIT} element={<UserEditPage />} />

        <Route path={ROUTE_TASK} element={<TaskPage />} />
        <Route path={ROUTE_TASK_EDIT} element={<TaskEditPage />} />
      </Route>
      <Route path="/" element={<PublicLayout><Wrapper /></PublicLayout>}>
        <Route path={ROUTE_SIGN_UP} element={<SignUpPage />} />
        <Route path={ROUTE_SIGN_IN} element={<SignInPage />} />
      </Route>
    </Routes>
  );
};
```

Рисунок 4.9 – Частина коду розподілення маршрутів додатку

Джерело: побудовано автором (знімок з екрану)

Наступним кроком було реалізовано автоматичний розподіл відповідальних за завдання. Цей етап втілений на основі мікросервісної архітектури. Тобто це окремий функціонал, який відповідає за визначений

складний функціонал та вирішує окрему визначену задачі, але в той же час є частиною однієї великої системи.

Задля покращення продуктивності було використано мову програмування python для даного сервісу. Спочатку був створений невеликий сервер за допомогою бібліотеки Flask. (Рис. 4.10)

```python
@app.route('/assign', methods=['POST'])
def assign_tasks():
    data_string = request.data.decode('utf-8')

    intermediate_data = json.loads(data_string)

    if isinstance(intermediate_data, str):
        data_json = json.loads(intermediate_data)
    else:
        data_json = intermediate_data

    users_df = pd.DataFrame(data_json['userList'])
    tasks_df = pd.DataFrame(data_json['taskList'])

    users_df['roleKeywords'].fillna('', inplace=True)

    vectorizer = TfidfVectorizer()

    combined_text = pd.concat([users_df['roleKeywords'], tasks_df['text']])
    combined_vectors = vectorizer.fit_transform(combined_text)

    user_vectors = combined_vectors[:len(users_df)]
    task_vectors = combined_vectors[len(users_df):]
    similarity_matrix = cosine_similarity(user_vectors, task_vectors)


    assigned_tasks_df = assign_tasks_fn(similarity_matrix, users_df, tasks_df)

    return jsonify(series_to_list(assigned_tasks_df))

if __name__ == "__main__":
    app.run(debug=True)
```

Рисунок 4.10 – Частина коду автоматичного розподілення користувачів

Джерело: побудовано автором (знімок з екрану)

Далі було створено сам алгоритм розподілу, який включає в себе аналіз тексту опису та назви завдання і підбір співробітника, який найбільш підходить відповідно до посади та навичок користувача.

4.4 Демонстрація роботи інформаційної технології

Перша сторінка сайту, яку потрібно пройти користувачу – це авторизація. На ній розташована одна форма з двома полями для вводу: електронна пошта та пароль. Форма має усі перевірки на правильність введених та обов'язкових для заповнення даних. На сторінці присутня можливість перейти на сторінку створення нового акаунта. (Рис. 4.11)

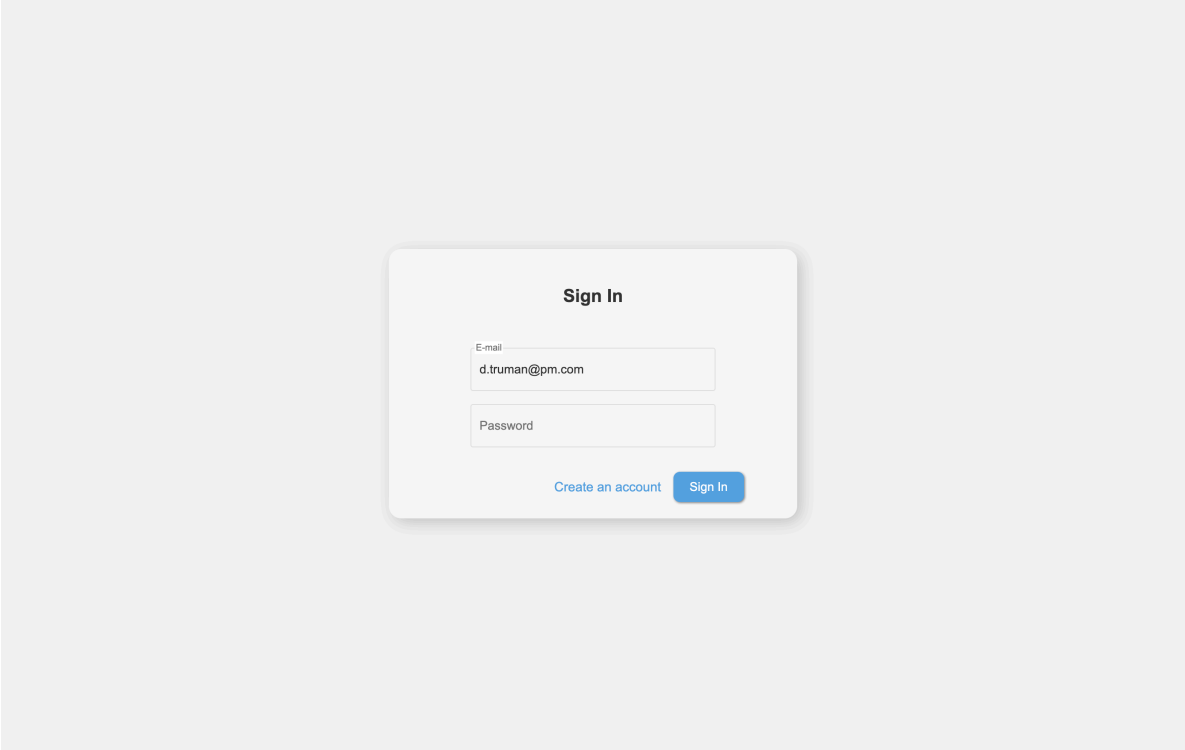


Рисунок 4.11 – Сторінка авторизації

Джерело: побудовано автором (знімок з екрану)

Сторінка реєстрації має схожий стиль, відрізняється кількістю перерахованих полів у формі. (Рис. 4.12)

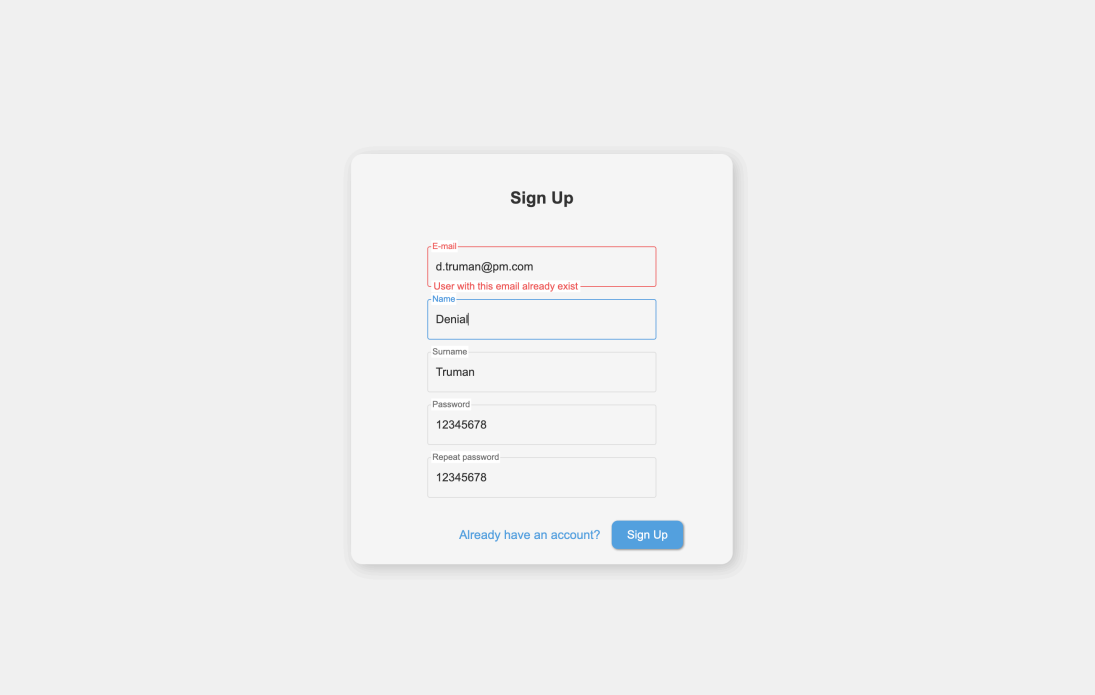


Рисунок 4.12 – Сторінка реєстрації

Джерело: побудовано автором (знімок з екрану)

Після авторизації користувача система запитає за яким проектом він хоче працювати. Після вибору відбудеться автоматична переадресація на головну сторінку системи, тобто Workspace. (Рис. 4.13)

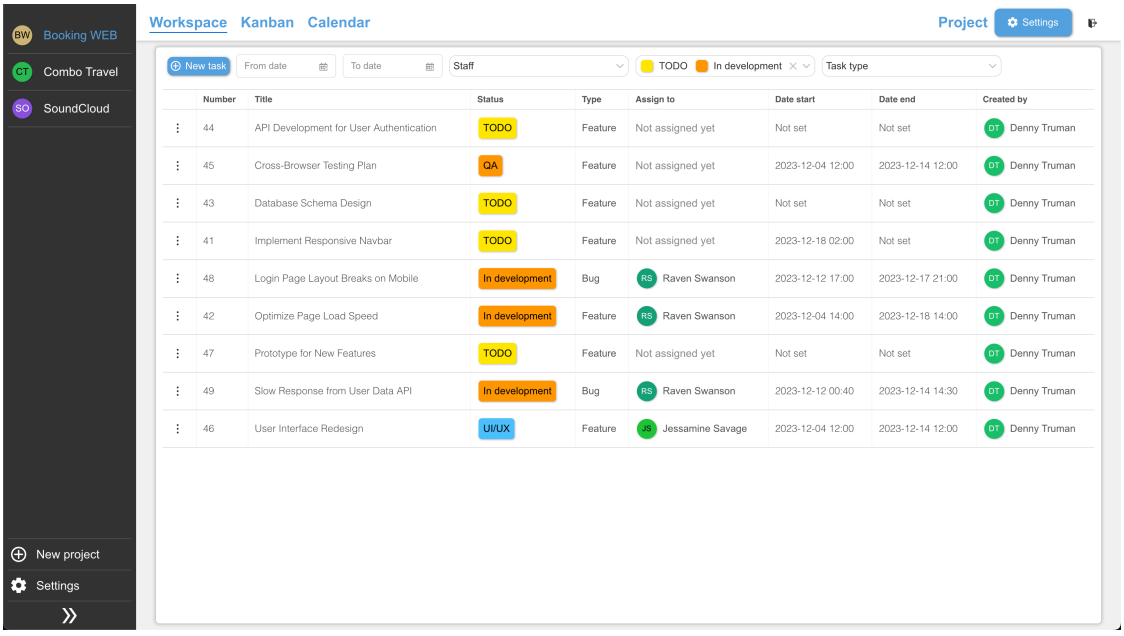


Рисунок 4.13 – Сторінка Workspace

Джерело: побудовано автором (знімок з екрану)

На даній сторінці можна переглянути основні дані завдань, відфільтрувати їх за ключовими параметрами.

Загалом сайт має логічний та легкий для сприйняття дизайн. Сайдбар забезпечує швидке перемикання між проектами, тоді як хедер надає доступ до головних розділів сайту.

Сторінка Kanban оптимізує управління проектом, надаючи візуально чітку структуру, яка дозволяє користувачам без зусиль оцінювати обсяг завдань, а також встановлювати пріоритети та планувати порядок виконання робіт. (Рис 4.14)

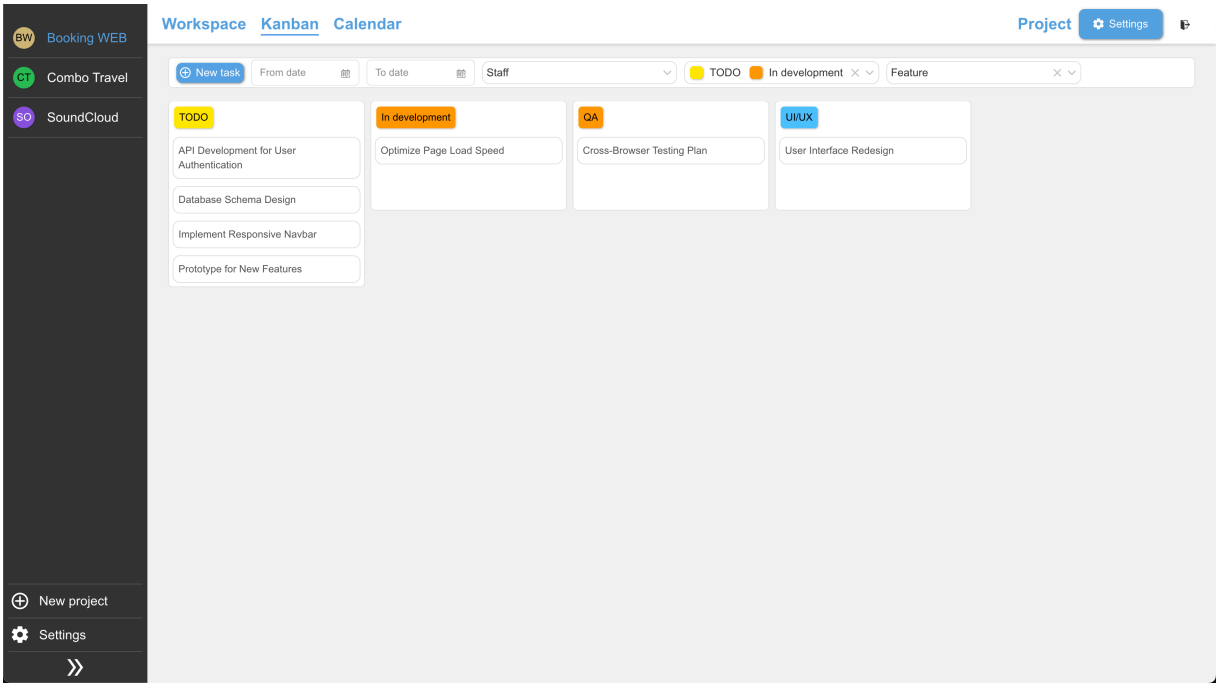


Рисунок 4.14 – Сторінка Kanban

Джерело: побудовано автором (знімок з екрану)

Ще одним з найпопулярнішим видом перегляду є календар, через що було розроблено відповідну сторінку з відповідним функціоналом. Він включає повноекранний календар з можливість навігації між датами та зміною виду відображення. Дану сторінку можна переглянути на рисунку 4.15

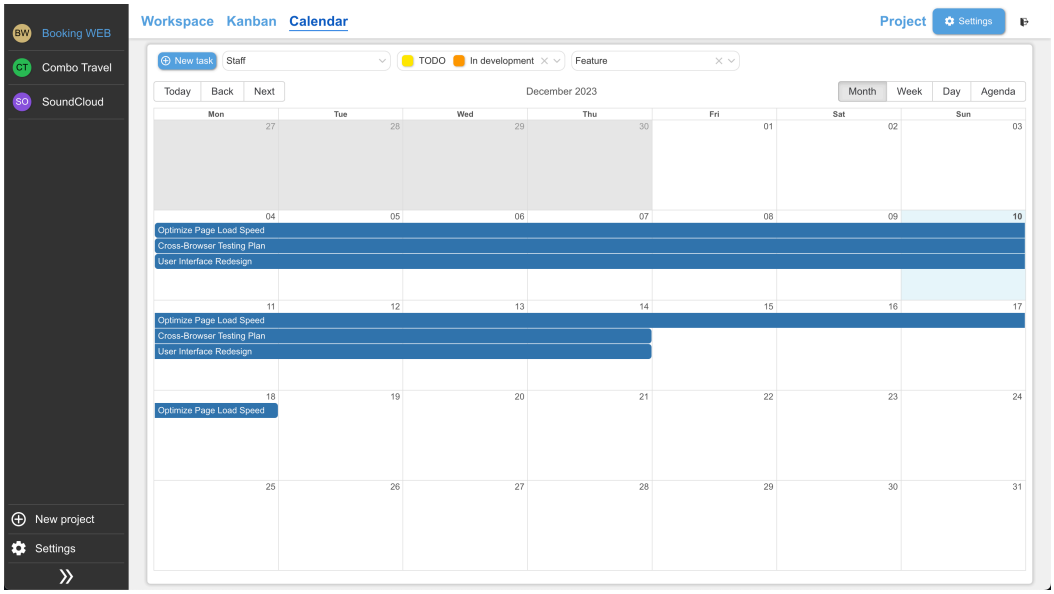


Рисунок 4.15 – Сторінка Kanban

Джерело: побудовано автором (знімок з екрану)

Натиснувши на будь-яке завдання користувача одразу переадресує на сторінку перегляду. На ній можна переглянути таку інформацію: тип, дата початку та дата кінця, статус, проєкт, до якого задача прикріплена або ж хто відповідальний за неї. Приклад такої сторінки можна переглянути на рисунку 4.16.

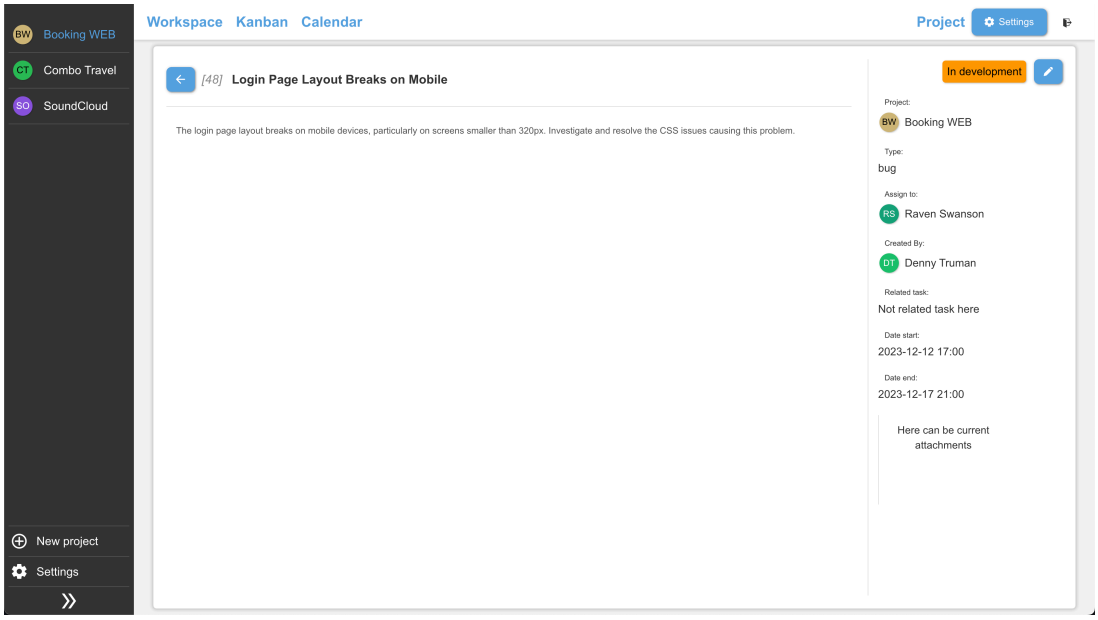


Рисунок 4.16 – Сторінка перегляду завдання

Джерело: побудовано автором (знімок з екрану)

Для того, щоб редагувати завдання, перш за все, потрібно мати привілеї, наступним кроком потрібно натиснути на іконку редагування, яка знаходить на сторінці перегляду. Відкрита сторінка має схожий вигляд, тільки будуть присутні поля для зміни відповідних полів запису. (Рис. 4.17)

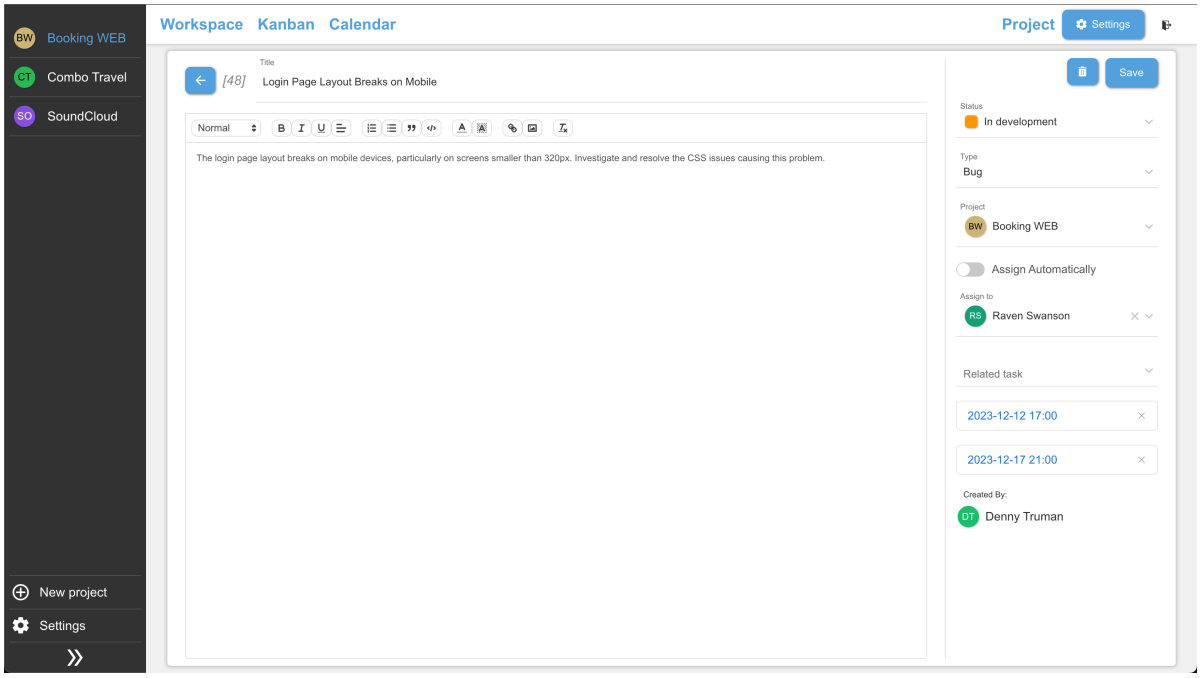


Рисунок 4.17 – Сторінка редагування завдання

Джерело: побудовано автором (знімок з екрану)

Щоб редагувати інформацію користувача, потрібно мати відповідні привілеї. В системі було передбачено редагування особистих даних для кожного співробітника окремо. Таку сторінку можна переглянути на рисунку 4.18.

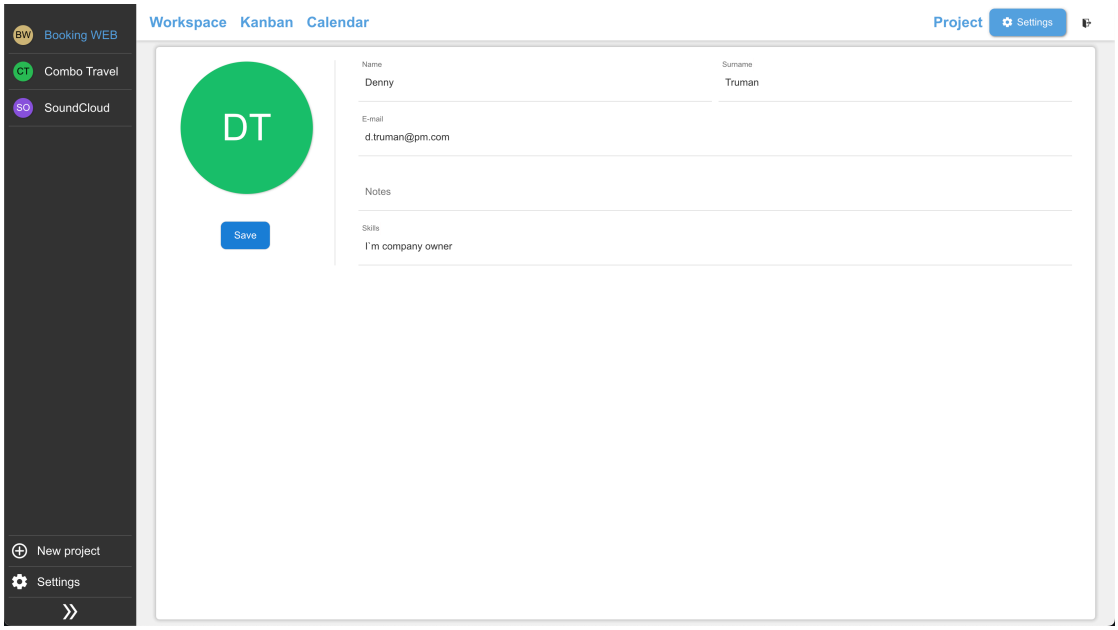


Рисунок 4.18 – Сторінка редагування інформації користувача

Джерело: побудовано автором (знімок з екрану)

Система розрахована на одну компанію, але на багато проєктів. Звичайно ж кожен проект може мати як різних так і спільних учасників. Задля зручності керування персоналом було створено відповідну сторінку, яка зображена на рисунку 4.19.

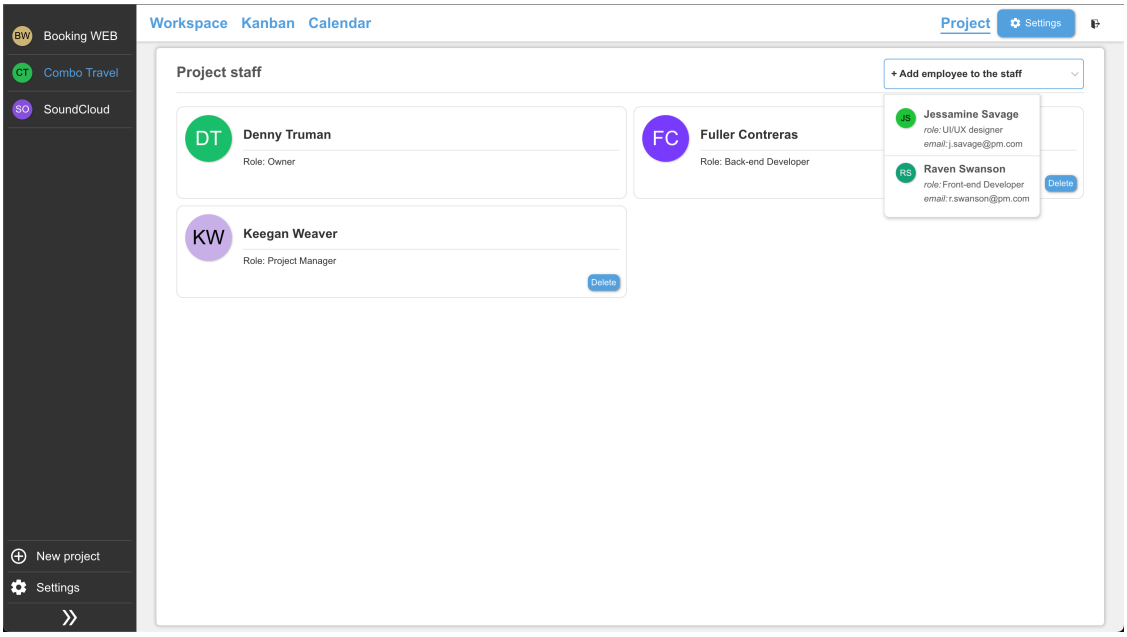


Рисунок 4.19 – Сторінка редагування співробітників проекту

Джерело: побудовано автором (знімок з екрану)

Редагування персоналу компанії відбувається на сторінці «Settings» (рис. 4.20). На цій сторінці можна додавати, та видаляти персонал з компанії. Віконце видалення персоналу зображено на рисунку 4.21. Після видалення можна відновити користувача в компанію, так як він просто відображається у полі «Restricted employees»(рис. 4.22).

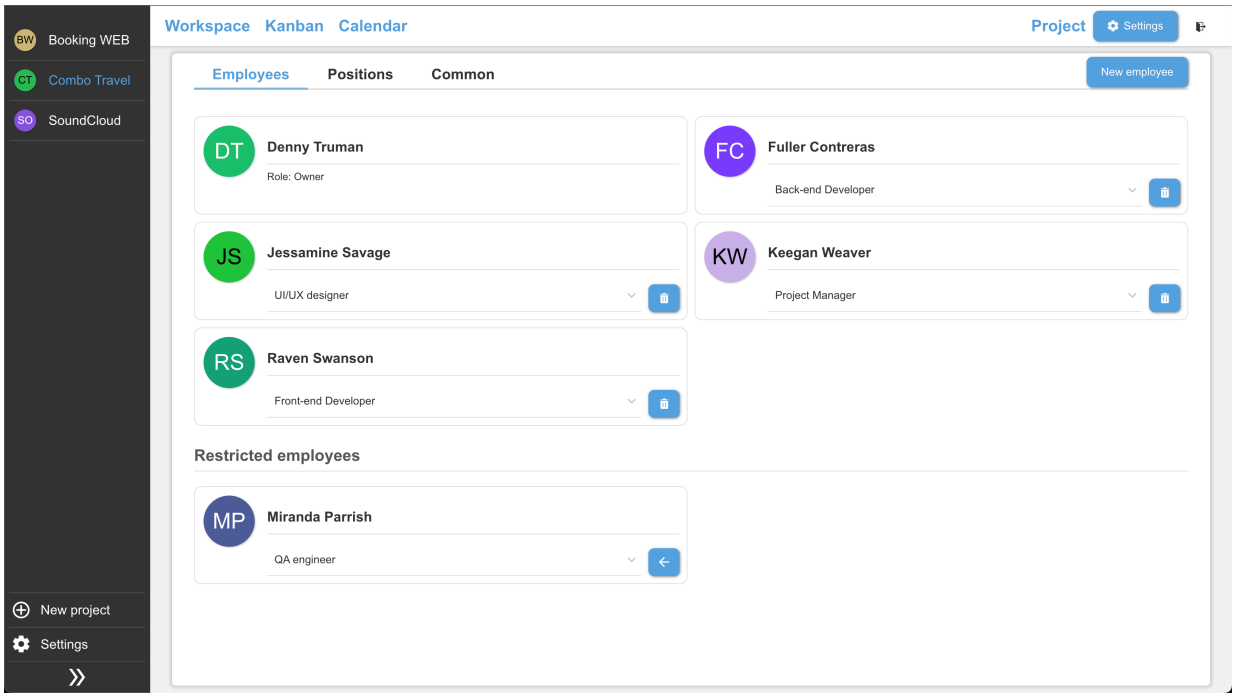


Рисунок 4.20 – Сторінка редагування співробітників компанії

Джерело: побудовано автором (знімок з екрану)

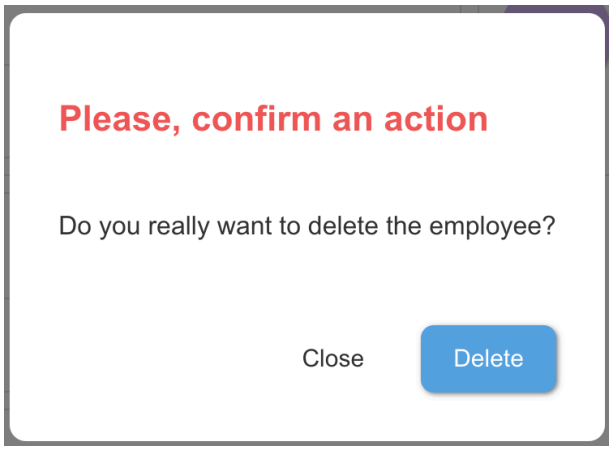


Рисунок 4.21 – Модальне вікно підтвердження видалення співробітника з компанії

Джерело: побудовано автором (знімок з екрану)

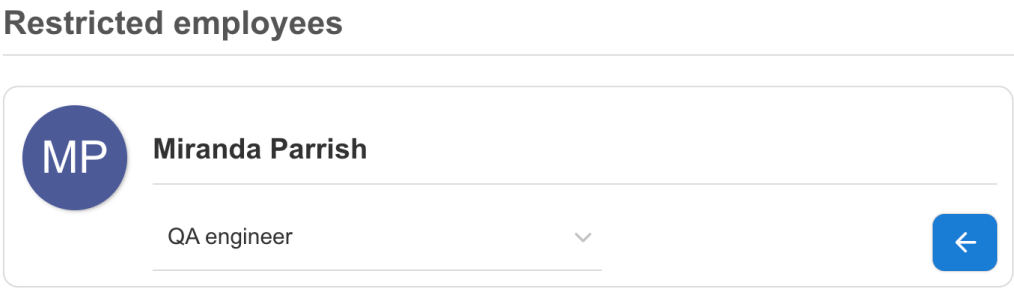


Рисунок 4.22 – Блок з відстороненими співробітниками

Джерело: побудовано автором (знімок з екрану)

Зазвичай співробітники IT-компанії мають гнучкі можливості в командах та можуть мати різні привілеї в залежності від посади. Через це було розроблено відповідну сторінку налаштувань, яка в зручному вигляді дає змогу надати відповідні привілеї користувачів у системі. Дана сторінка зображена на рисунку 4.23.

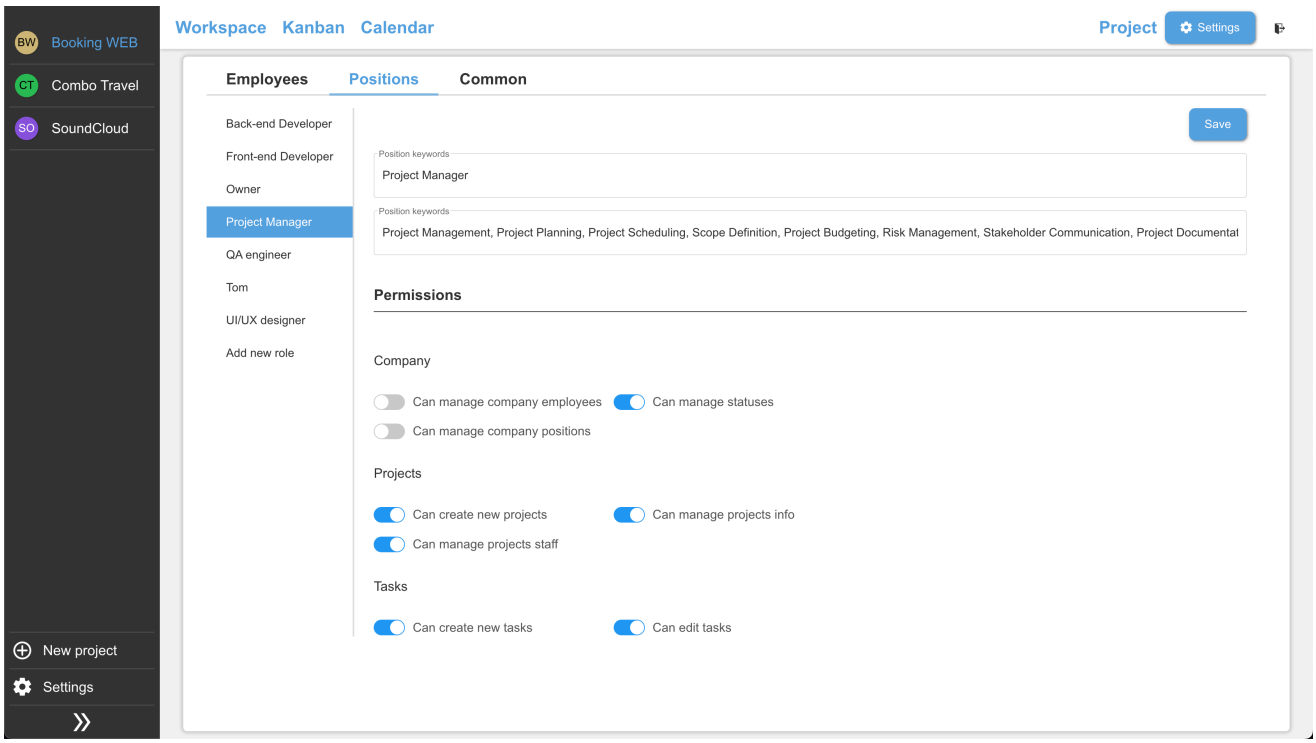


Рисунок 4.23 – Сторінка редагування привілеїв користувача

Джерело: побудовано автором (знімок з екрану)

Задля забезпечення кастомізації під різні потреби проекту, було додано можливість додавати власні статуси до завдань. Сторінка з створення статусу зображена на рисунку 4.23, модальне вікно містить ім’я, опис та колір статусу.

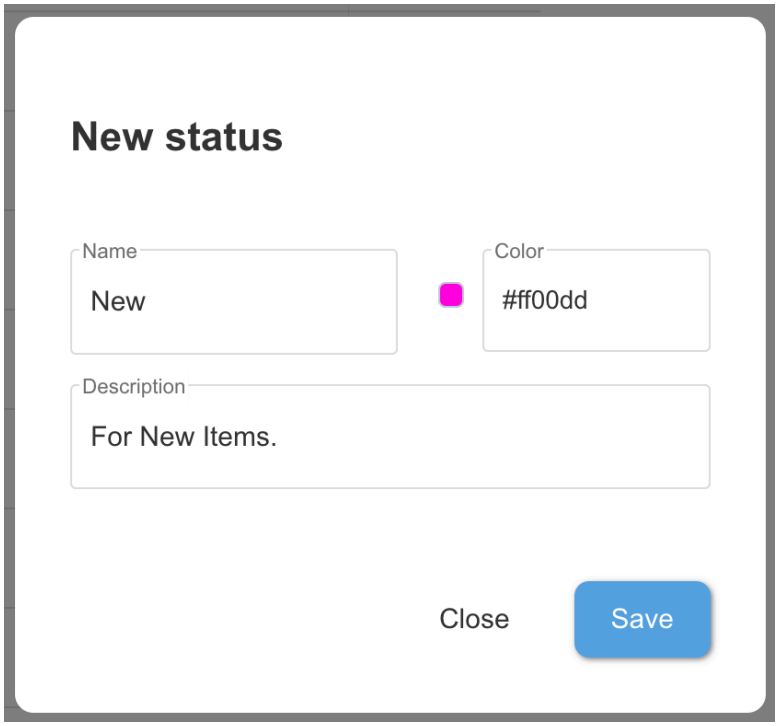


Рисунок 4.23 – Модальне вікно створення статусу

Джерело: побудовано автором (знімок з екрану)

Після додання нового статусу він успішно відображається на сторінці з статусами. Для них є можливість редагування та видалення. Сторінка зі списком статусів завдань ображена на рисунку 4.24.

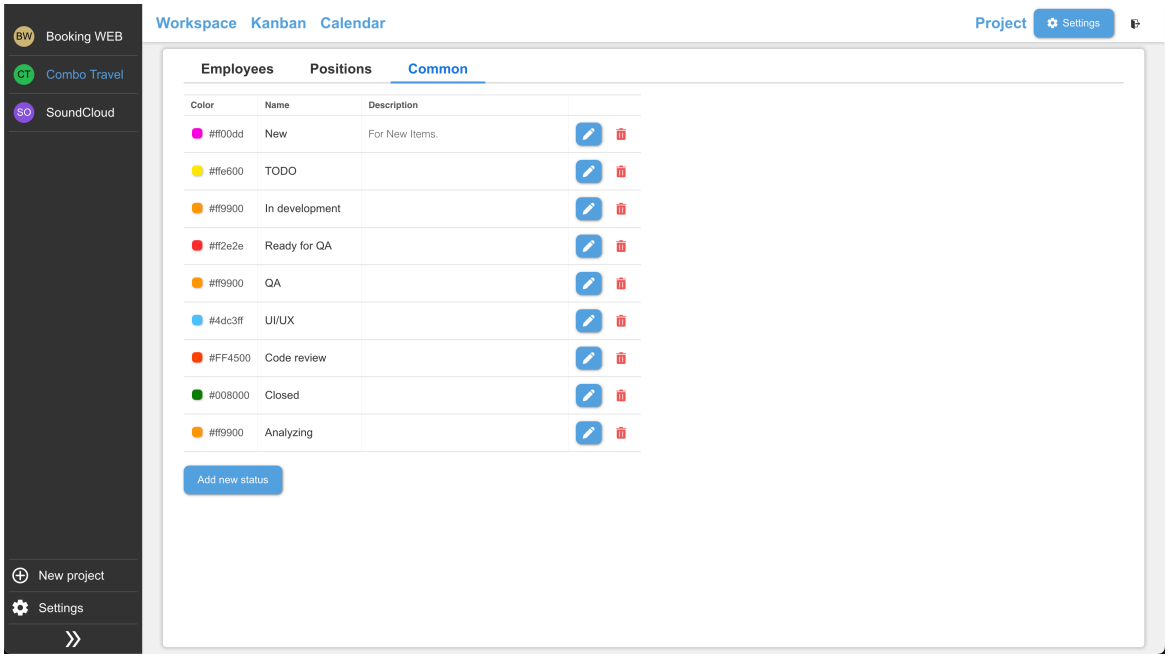


Рисунок 4.24 – Сторінка з статусами завдань

Джерело: побудовано автором (знімок з екрану)

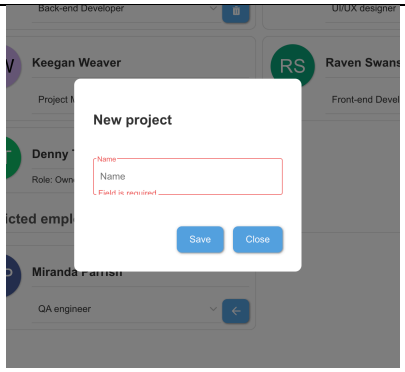
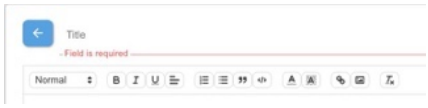
4.5 Тестування інформаційної технології

Для забезпечення надійної роботи реалізованого функціоналу інформаційної технології важливо провести функціональне тестування. Цей процес передбачає верифікацію вхідних даних форм створення, редагування та видалення даних про проект, завдання, користувача, статуси і т.д. Також треба протестувати навігацію по додатку, а також основний функціонал інформаційної технології.

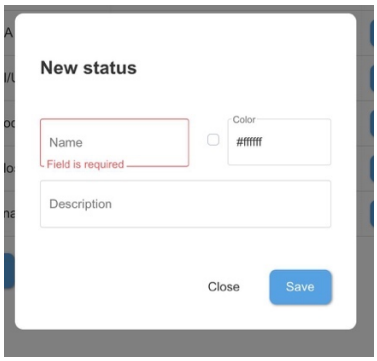
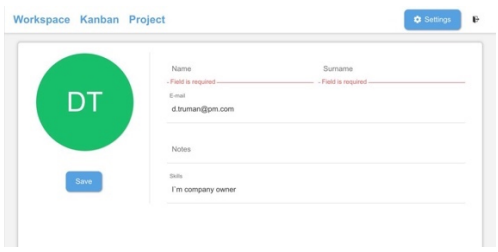
Спочатку необхідно перевірити роботу посилань у головному меню, щоб відкривались правильні сторінки. Під час перевірки переходу на різноманітні сторінки – відображались відповідні дані. Проекти, завдання, користувачі, ролі, статуси створюються, редагуються та видаляються успішно. Також їх інформація відображається коректно на сторінках «Kanban», «Project», «Workspace», «Task».

Тестування функціональності форм в інформаційній технології було здійснено використовуючи метод «чорного ящика», що передбачає перевірку працездатності системи без втручання у її внутрішню структуру та код[35]. Деталі та результати тестування були узагальнені у вигляді таблиці 4.1.

Таблиця 4.1 – Тестування форм на валідні та невалідні дані

№	Назва тест кейсу	Очікуваний результат	Фактичний результат	0/1
1	Перевірка форми створення проекту введенням коректних даних.	Проект успішно створений.	Проект створений коректно і відображається у боковому меню.	1
2	Перевірка форми створення проекту введенням невалідних даних.	Відображається текст повідомлення з помилкою.		1
3	Перевірка форми створення завдання проекту введенням коректних даних.	Завдання успішно створено.	Завдання відображається на Канбан сторінці та на головній сторінці проекту.	1
4	Перевірка форми створення завдання проекту введенням не коректних даних.	Відображається текст повідомлення з помилкою.		1

Продовження таблиці 4.1

№	Назва тест кейсу	Очікуваний результат	Фактичний результат	0/1
5	Перевірка форми створення статусу введенням валідних даних.	Статус успішно створено.	Новий статус відображається під час створення та редагування завдань.	1
6	Перевірка форми створення статусу не валідними даними.	Статус не створено, повідомлення про помилку підсвічується червоним кольором.		1
7	Перевірка форми редагування профілю користувача валідними даними.	Дані користувача успішно збережно.	Оновлені дані користувача коректно відображаються на сторінці користувача та під час призначення завдань.	1
8	Перевірка форми редагування профілю користувача невалідними даними.	Дані користувача не оновлено, повідомлення про помилку підсвічується червоним кольором.		1

Джерело: побудовано автором

Внаслідок виконаного тестування, серйозних помилок чи недоліків не було знайдено. Весь функціонал працює належним чином.

ВИСНОВОК

Під час виконання кваліфікаційної роботи магістра було здійснено проектування інформаційної технології підтримки оперативного планування роботи проектного менеджера. Проведено аналіз існуючих інформаційних технологій, виконано дослідження визначених функціональних вимог, а саме:

– створення, редагування та видалення проектів;

– створення, редагування та видалення завдань;

– проведення автоматичного розподілу завдань між членами команди;

– фільтрація/сортування за атрибутами проекту та користувачів;

– візуалізація проектів у вигляді Kanban дошки, календаря та звичайного списку.

В процесі дослідження предметної області була виявлена необхідність у розробці інформаційної технології підтримки оперативного планування роботи проектного менеджера, оскільки наразі не існує розроблених аналогів цієї системи. Під час аналізу існуючих аналогів програмного продукту були виявлені ключові переваги, на яких потрібно зосередитися в ході проектування, а також виявлено недоліки, що вимагають подальшого удосконалення при реалізації проекту

У наступному етапі розробки дипломного проекту було здійснено встановлення цілей проекту, визначено як функціональні, так і нефункціональні вимоги до системи, а також сформульовано завдання для її реалізації. Крім того, було виконано планування робочих процесів, що допомогло визначити терміни виконання проекту.

Далі у процесі розробки відбулось детальне проектування робочих процесів та сценаріїв використання у майбутній інформаційної технології. Це було здійснено за допомогою створення IDEF0 діаграми, які сприяли декомпозиції загальних процесів на більш деталізовані процеси, а також за допомогою Use Case діаграми, яка проілюструвала різноманітні сценарії використання системи. Такий підхід дозволив точно окреслити функціонал

інформаційної системи і розбити її на менші компоненти для зручності в реалізації.

Дана інформаційна технологія була розроблена з використанням Express.js та Flask у якості серверної частини додатку, бібліотеку React.js для створення інтерфейсу, бібліотеку scikit-learn для реалізації методу автоматичного розподілу задач серед співробітників.

Після виконання етапу розробки було проведено тестування інформаційної технології методом black-box.

Створювана інформаційна система буде користуватись значним попитом серед компаній у сфері інформаційних технологій, які зацікавлені у якісній оптимізації процесів планування та розподілу задач серед виконавців, аналізуючи їх пріоритет а також навички членів команди.

Отже, створена інформаційна технологія підтримки оперативного планування роботи проектного менеджера дозволить оптимізувати процес планування задач проектним менеджером, забезпечивши автоматизацію розподілу задач між членами команди розробки, а також покращить продуктивність роботи проектного менеджера, впливатиме на позитивний настрій в колективі, завдяки відсутності перенавантаження співробітників .

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ricardianto P. et al. The Role of Information Technology in Business Agility: Systematic Literature Review GENERAL MANAGEMENT The Role of Information Technology in Business Agility: Systematic Literature Review.

2. Wu X., Klein G., Jiang J.J. On the Road to Digital Transformation: A Literature Review of IT Program Management // Project Management Journal. SAGE Publications Inc., 2023. Vol. 54, № 4. P. 409–427.

3. Кожемякін В А. Аналіз можливостей застосування генеративного ШІ у сфері управління ІТ проектами. 2023. 1–79 р.

4. Возниця А С. Програмна система task-менеджер для керування ІТ проектами. Київ: НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ, 2021. 1–103 р.

5. Singh M., Chauhan N., Popli R. Task Allocation in Distributed Agile Software Development Environment Using Unsupervised Learning // Journal of Engineering Research (Kuwait). 2022. Vol. 10.

6. Hinderks A. et al. A Methodology for Integrating User Experience Methods and Techniques into Agile Software Development.

7. Aslam W., Ijaz F. A Quantitative Framework for Task Allocation in Distributed Agile Software Development // IEEE Access. 2018. Vol. 6.

8. Oliveira P. et al. Issue Auto-Assignment in Software Projects with Machine Learning Techniques // Proceedings - 2021 IEEE/ACM 8th International Workshop on Software Engineering Research and Industrial Practice, SER and IP 2021. 2021.

9. Barenkamp M., Rebstadt J., Thomas O. Applications of AI in classical software engineering // AI Perspectives. 2020. Vol. 2, № 1.

10. Kaur A. App Review: Trello // J Hosp Librariansh. Routledge, 2018. Vol. 18, № 1. P. 95–101.

11. Що таке Trello: можливості, приклади використання тощо | Trello [Electronic resource]. URL: https://trello.com/uk/tour (accessed: 04.12.2023).

12. Introduction to monday.com – Support [Electronic resource]. URL: https://support.monday.com/hc/en-us/articles/115005310945-Introduction-to-monday-com- (accessed: 22.11.2023).

13. Avis monday.com : test de l'outil idéal pour les startups et entrepreneurs [Electronic resource]. URL: https://www.presse-citron.net/monday-com-outil-saas-startup-avis-test/ (accessed: 04.12.2023).

14. Програмне забезпечення для керування проектами | Microsoft Project [Electronic resource]. URL: https://www.microsoft.com/uk-ua/microsoft-365/project/project-management-software?rtc=1 (accessed: 22.11.2023).

15. Displaying The Task Path In Microsoft Project [Electronic resource]. URL: https://tensix.com/displaying-the-task-path-in-microsoft-project/ (accessed: 04.12.2023).

16. Alaidaros H., Omar M., Romli R. The state of the art of agile kanban method: challenges and opportunities // Independent Journal of Management & Production. Independent Journal of Management and Production, 2021. Vol. 12, № 8. P. 2535–2550.

17. Schmidhuber J. Deep learning in neural networks: An overview // Neural Networks. Pergamon, 2015. Vol. 61. P. 85–117.

18. González-Recio O., Jiménez-Montero J.A., Alenda R. The gradient boosting algorithm and random boosting for genome-assisted evaluation in large data sets // J Dairy Sci. Elsevier, 2013. Vol. 96, № 1. P. 614–624.

19. Belgiu M., Drăgu L. Random forest in remote sensing: A review of applications and future directions // ISPRS Journal of Photogrammetry and Remote Sensing. Elsevier, 2016. Vol. 114. P. 24–31.

20. Khurana D. et al. Natural language processing: state of the art, current trends and challenges // Multimed Tools Appl. Springer, 2023. Vol. 82, № 3. P. 3713–3744.

21. Rodriguez M.Z. et al. Clustering algorithms: A comparative approach // PLoS One. Public Library of Science, 2019. Vol. 14, № 1. P. e0210236.

22. React – JavaScript-бібліотека для створення користувацьких інтерфейсів [Electronic resource]. URL: https://uk.legacy.reactjs.org/ (accessed: 19.11.2023).

23. styled-components [Electronic resource]. URL: https://styled-components.com/ (accessed: 19.11.2023).

24. Get Programming with Node.js - Jonathan Wexler - Google книги [Electronic resource]. URL: https://books.google.com.ua/books?hl=uk&lr=&id=TTozEAAAQBAJ&oi=fnd&pg=PT10&dq=about+express.js&ots=cHpyZ7kWym&sig=7P7PouW9gdAEobf5rbbeBLxCoSc&redir_esc=y#v=onepage&q=about%20express.js&f=false (accessed: 22.11.2023).

25. Express - Node.js web application framework [Electronic resource]. URL: https://expressjs.com/ (accessed: 22.11.2023).

26. Rad B.B., Bhatti H.J., Ahmadi M. An Introduction to Docker and Analysis of its Performance // IJCSNS International Journal of Computer Science and Network Security. 2017. Vol. 17, № 3.

27. Welcome to Python.org [Electronic resource]. URL: https://www.python.org/ (accessed: 19.11.2023).

28. PostgreSQL: The world’s most advanced open source database [Electronic resource]. URL: https://www.postgresql.org/ (accessed: 19.11.2023).

29. Presley A., Liles D.H. The Use of IDEF0 for the Design and Specification of Methodologies. 1998.

30. UML Use Case Diagram Tutorial | Lucidchart [Electronic resource]. URL: https://www.lucidchart.com/pages/uml-use-case-diagram (accessed: 22.11.2023).

31. Посібник Н. Оксана Мулеса Яна Варга ІНФОРМАЦІЙНІ СИСТЕМИ ТА РЕЛЯЦІЙНІ БАЗИ ДАНИХ.

32. Liu G. et al. Microservices: Architecture, container, and challenges // Proceedings - Companion of the 2020 IEEE 20th International Conference on

Software Quality, Reliability, and Security, QRS-C 2020. Institute of Electrical and Electronics Engineers Inc., 2020. P. 629–635.

33. Sequelize | Feature-rich ORM for modern TypeScript & JavaScript [Electronic resource]. URL: https://sequelize.org/ (accessed: 06.12.2023).

34. Nguyen H. Single-page application and front-end testing methods : built with React and React Router, tested with Jest and Cypress. 2022.

35. What is Black Box Testing? - Check Point Software [Electronic resource]. URL: https://www.checkpoint.com/cyber-hub/cyber-security/what-is-penetration-testing/what-is-black-box-testing/ (accessed: 09.12.2023).

36. Siami-Irdemoosa E., Dindarloo S.R., Sharifzadeh M. Work breakdown structure (WBS) development for underground construction // Autom Constr. Elsevier, 2015. Vol. 58. P. 85–94.

37. Діаграма Ганта (Gantt Chart) [Electronic resource]. URL: https://www.maxzosim.com/diaghrama-ganta/ (accessed: 10.12.2023).

ДОДАТОК А

А1. Планування робіт

На сучасному етапі розвитку ІТ-галузі, інформаційні технології стають невід’ємною частиною оперативного управління та планування в компаніях. Особливо важливою вона є для проектних менеджерів, які прагнуть оптимізувати робочі процеси та підвищити ефективність команд. Автоматизація рутинних задач, особливо тих, що вимагають значного часу та мануальної праці, відкриває нові можливості для зосередження уваги на більш складних та стратегічних аспектах управління проектами.

Інформаційна технологія є фундаментальним інструментом для підтримки оперативного планування в роботі проектного менеджера, відіграючи вирішальну роль в успішній реалізації проектів.

А2. Деталізація мети методом SMART.

Чітке визначення цілей на етапі концептуального проектування є ключовим фактором для забезпечення ефективного та високоякісного виконання проекту. Використання методу SMART для деталізації цілей проекту дозволяє структурувати та ясно визначити необхідні параметри, що сприяє кращому планування та виконанню. Результати цього процесу відображено у таблиці А.1

Таблиця А.1 – Деталізація мети проекту методом SMART

Specific(Конкретна)	Розробити та імплементувати інформаційну технологію підтримки оперативного планування роботи проектного менеджера, задля оптимізації роботи проектного менеджера.
Measurable(Вимірювана)	Результатом створення дипломного проекту є інформаційна технологія підтримки оперативного планування роботи проектного менеджера.
Achievable(Досяжна)	Для досягнення мети проекту необхідні знання бібліотек React.js, Styled-components, мов програмування JavaScript, Python, бази даних PostgreSQL, фреймворку Express.js та навичок створення документації.
Relevant(Реалістична)	Розроблена інформаційна технологія дозволить автоматизувати процес назначення співробітників на виконання задач, покращить ефективність планування, звільняє час проектного менеджера для зосередження на більш складних аспектах проекту.
Time-framed(Обмежена у часі)	Термін досягнення мети проекту визначено з замовником і дорівнює 3 місяці.

Джерело: побудовано автором

А3. Планування змісту робіт.

Work Breakdown Structure(WBS) – це інструмент розподілу проекту на керовані компоненти, що допомагає керівникам проектів і командам у визначенні і структуруванні обсягу робіт. Мета розробки WBS полягає у створенні чіткої, детальної та ієрархічної структури. WBS структурує роботу у

формі ієрархії, де на верхньому рівні цієї ієрархії розташовується кінцевий продукт, а кожен наступний, більш нижній рівень, представляє собою більш деталізоване визначення робочих задач. Процес декомпозиції завдань триває до того моменту, поки вони не досягнуть розміру, який є оптимальним для ефективного управління та контролю, але водночас достатньо великого, щоб мати практичне значення. На рисунку А.1 представлено WBS для проекту розробки інформаційної технології підтримки оперативного планування роботи проектного менеджера [36].

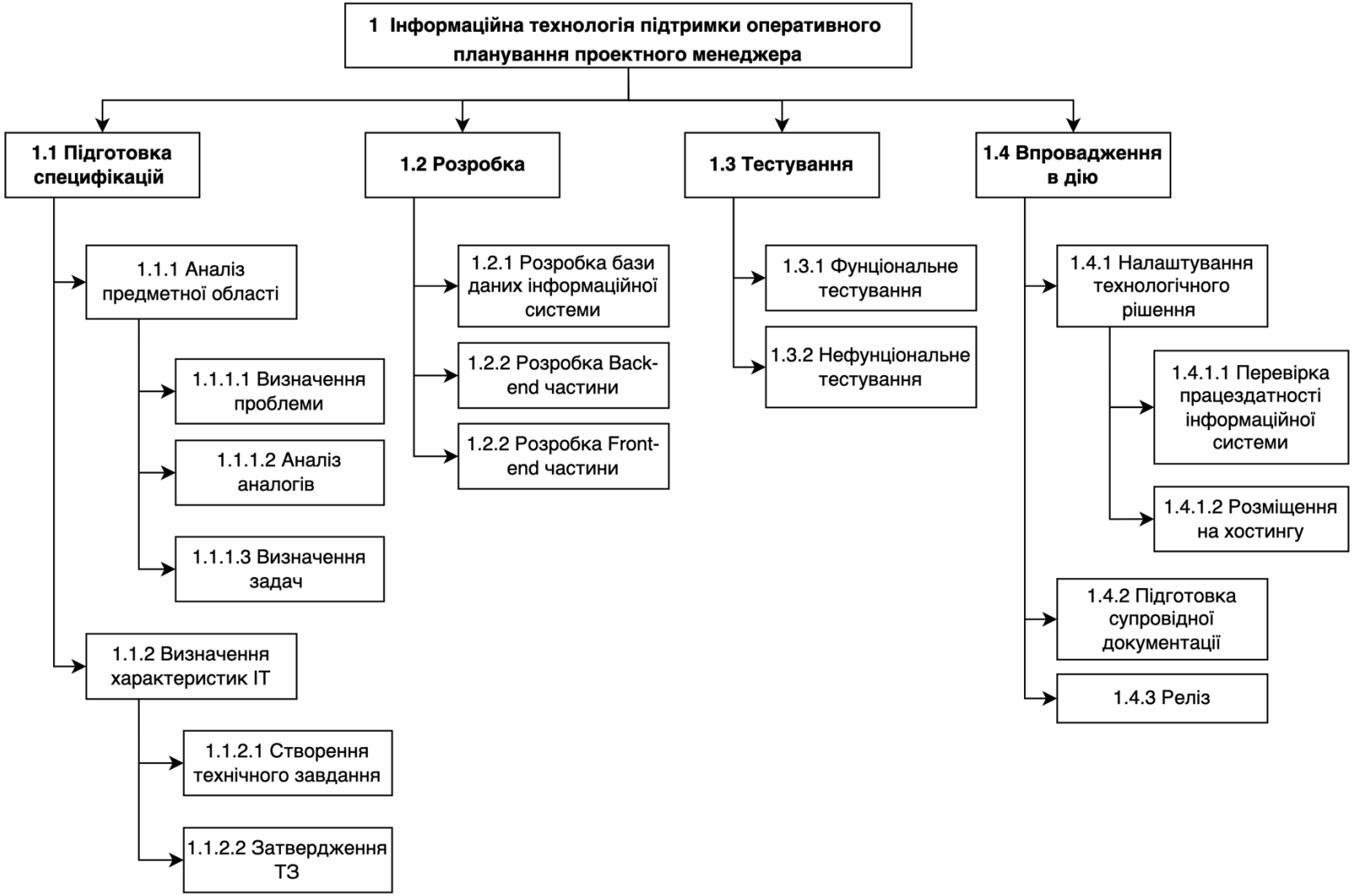


Рисунок А.1 – WBS-структура робіт проєкту

Джерело: побудовано автором

А4. Планування структури виконавців.

Розробка організаційної структури виконавців (OBS) є наступним кроком після декомпозиції процесів за допомогою WBS. OBS або Organization Breakdown Structure – це структура, яка застосовується для демонстрації ієрархії команди імплементації проекту, і з’ясовує як організована команда та розподіл ресурсів у відповідності з метою виконання проекту. На рисунку А.2 зображена організаційна структура планування робіт проекту. Дані про учасників проекту подано у таблиці А.2.

Таблиця А.2 – Виконавці проекту

Роль	**Ім’я**	**Проектна роль**
Розробник	Райко Д.І.	Розробка frontend-у та backend-у інформаційної технології.
Тестувальник	Райко Д.І.	Проведення тестування програмного продукту.
Проектувальник	Райко Д.І.	Проектування структури програмного продукту та бази даних.
Керівник проекту	Федотова Н.А.	Створення завдання на розробку проекту, рецензія імплементованого програмний продукт.
Менеджер	Райко Д.І.	Контроль над дотриманням дедлайну, розподіл ресурсів та задач на розробку проекту. Проведення аналізу та збору інформації.

Джерело: побудовано автором

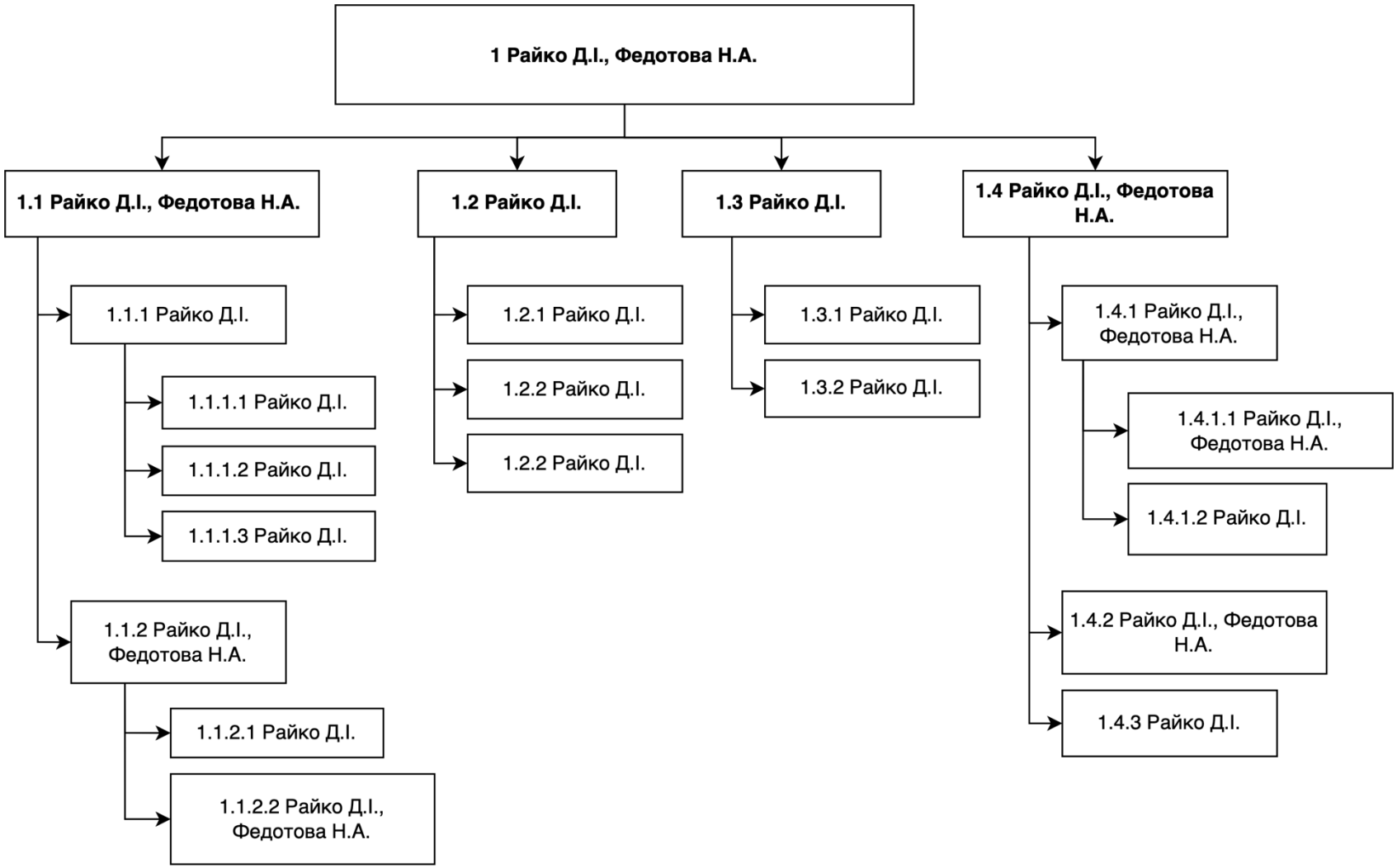


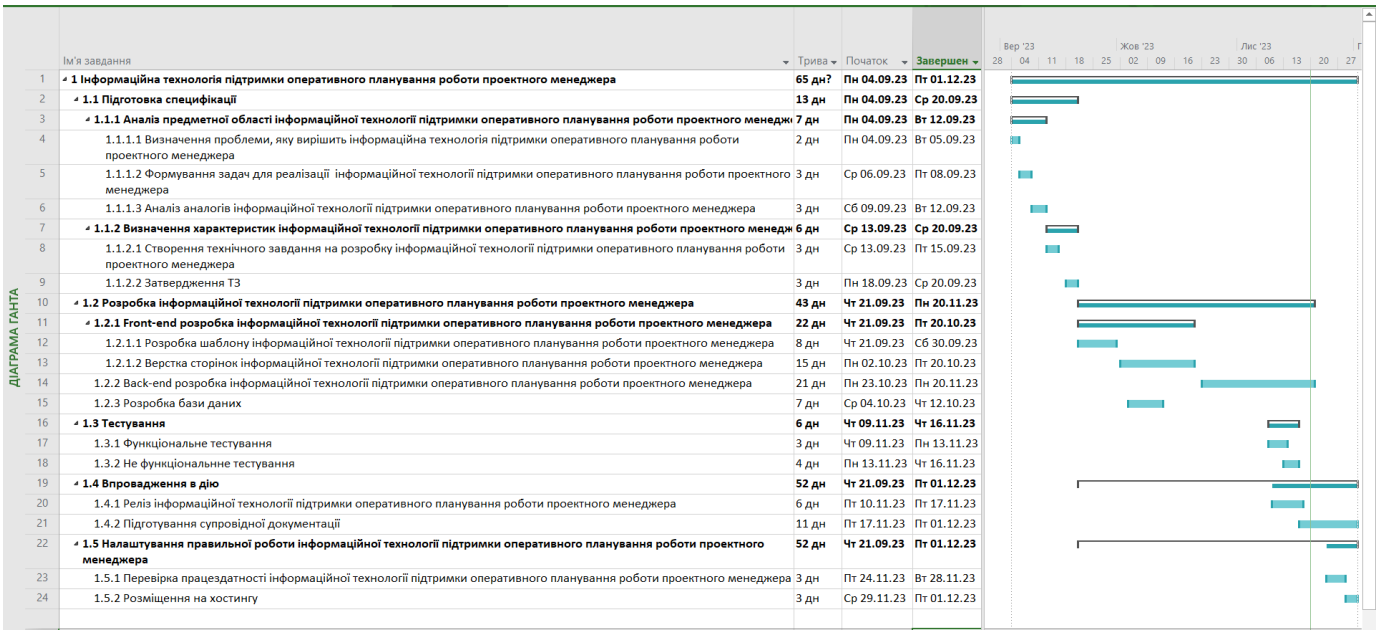
Рисунок А.2 – OBS-структура робіт проєкту

Джерело: побудовано автором

А5. Діаграма Ганта.

Розробка календарного плану робіт є ключовою складовою управління проектом. Цей план представляє собою діаграму з розподілом часових рамок для кожного завдання, що сприяє точному визначенню загальної тривалості проекту з урахуванням доступних ресурсів [37].

Календарний графік проекту представлено на рисунку А.3.



	Ім'я завдання	Трива	Початок	Завершен
1	**1 Інформаційна технологія підтримки оперативного планування роботи проектного менеджера**	**65 дн?**	**Пн 04.09.23**	**Пт 01.12.23**
2	**1.1 Підготовка специфікації**	**13 дн**	**Пн 04.09.23**	**Ср 20.09.23**
3	**1.1.1 Аналіз предметної області інформаційної технології підтримки оперативного планування роботи проектного менедж**	**7 дн**	**Пн 04.09.23**	**Вт 12.09.23**
4	1.1.1.1 Визначення проблеми, яку вирішить інформаційна технологія підтримки оперативного планування роботи проектного менеджера	2 дн	Пн 04.09.23	Вт 05.09.23
5	1.1.1.2 Формування задач для реалізації інформаційної технології підтримки оперативного планування роботи проектного менеджера	3 дн	Ср 06.09.23	Пт 08.09.23
6	1.1.1.3 Аналіз аналогів інформаційної технології підтримки оперативного планування роботи проектного менеджера	3 дн	Сб 09.09.23	Вт 12.09.23
7	**1.1.2 Визначення характеристик інформаційної технології підтримки оперативного планування роботи проектного менедж**	**6 дн**	**Ср 13.09.23**	**Ср 20.09.23**
8	1.1.2.1 Створення технічного завдання на розробку інформаційної технології підтримки оперативного планування роботи проектного менеджера	3 дн	Ср 13.09.23	Пт 15.09.23
9	1.1.2.2 Затвердження ТЗ	3 дн	Пн 18.09.23	Ср 20.09.23
10	**1.2 Розробка інформаційної технології підтримки оперативного планування роботи проектного менеджера**	**43 дн**	**Чт 21.09.23**	**Пн 20.11.23**
11	**1.2.1 Front-end розробка інформаційної технології підтримки оперативного планування роботи проектного менеджера**	**22 дн**	**Чт 21.09.23**	**Пт 20.10.23**
12	1.2.1.1 Розробка шаблону інформаційної технології підтримки оперативного планування роботи проектного менеджера	8 дн	Чт 21.09.23	Сб 30.09.23
13	1.2.1.2 Верстка сторінок інформаційної технології підтримки оперативного планування роботи проектного менеджера	15 дн	Пн 02.10.23	Пт 20.10.23
14	1.2.2 Back-end розробка інформаційної технології підтримки оперативного планування роботи проектного менеджера	21 дн	Пн 23.10.23	Пн 20.11.23
15	1.2.3 Розробка бази даних	7 дн	Ср 04.10.23	Чт 12.10.23
16	**1.3 Тестування**	**6 дн**	**Чт 09.11.23**	**Чт 16.11.23**
17	1.3.1 Функціональне тестування	3 дн	Чт 09.11.23	Пн 13.11.23
18	1.3.2 Не функціональне тестування	4 дн	Пн 13.11.23	Чт 16.11.23
19	**1.4 Впровадження в дію**	**52 дн**	**Чт 21.09.23**	**Пт 01.12.23**
20	1.4.1 Реліз інформаційної технології підтримки оперативного планування роботи проектного менеджера	6 дн	Пт 10.11.23	Пт 17.11.23
21	1.4.2 Підготування супровідної документації	11 дн	Пт 17.11.23	Пт 01.12.23
22	**1.5 Налаштування правильної роботи інформаційної технології підтримки оперативного планування роботи проектного менеджера**	**52 дн**	**Чт 21.09.23**	**Пт 01.12.23**
23	1.5.1 Перевірка працездатності інформаційної технології підтримки оперативного планування роботи проектного менеджера	3 дн	Пт 24.11.23	Вт 28.11.23
24	1.5.2 Розміщення на хостингу	3 дн	Ср 29.11.23	Пт 01.12.23

Рисунок А.3 – Календарний графік проекту

Джерело: побудовано автором

А6. Управління ризиками проекту.

На етапі планування проектних робіт особлива увага приділяється аналізу та управлінню ризиками. Цей процес включає кілька кроків: ідентифікацію ризиків, оцінку їх впливу та ймовірності, розробку стратегій їх зменшення або усунення, моніторинг та контроль за застосуванням цих стратегій. У таблиці А.3 було перелічено ризики даного проекту. Оцінки ризиків надано у таблиці А.4 Таблиця А.5 показує шкалу ризиків за типом, величиною впливу та ймовірністю.

Таблиця А.3 – Ризики проекту.

№ ризику	Назва (опис) ризику
1	Відсутність кваліфікованих знань
2	Технічні збої у обладнанні
3	Проблеми з інтернетом
4	Проблема з електропостачанням
5	Відсутність достатнього фінансування проекту
6	Захворювання одного з членів команди
7	Недостатня якість вихідного коду
8	Поява альтернативного продукту
9	Недостатній контроль якості
10	Зміна вимог замовника

Джерело: побудовано автором

Таблиця А.4 – Результати визначення ймовірності, впливу та рангу ризику проекту.

№ ризику	Назва (опис) ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
1	Відсутність кваліфікованих знань	0,1	0,2	0,02
2	Технічні збої у обладнанні	0,3	0,2	0,06
3	Проблеми з інтернетом	0,5	0,05	0,025
4	Проблема з електропостачанням	0,3	0,2	0,06

Продовження таблиці А.4

№ ризику	Назва (опис) ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
5	Відсутність достатнього фінансування проекту	0,5	0,2	0,1
6	Захворювання одного з членів команди	0,1	0,4	0,04
7	Недостатня якість вихідного коду	0,7	0,2	0,14
8	Поява альтернативного продукту	0,5	0,2	0.1
9	Недостатній контроль якості	0,3	0,2	0,06
10	Зміна вимог замовника	0,1	0,8	0,08

Джерело: побудовано автором

Таблиця А.5 – Шкала оцінювання ризика типом, ймовірністю та величиною впливу.

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Джерело: побудовано автором

Для зменшення негативного впливу потенційних загроз на проект, критично важливо розробити детальний план реагування на ризики. Цей план має включати аналіз та оцінку ефективності запропонованих стратегій зниження

ризиків, враховуючи можливі наслідки для проекту. Така оцінка базується на критеріях, визначених у таблиці А.5. В результаті створення плану реагування була сформована матриця, що відображає ймовірність і вплив різних ризиків, що було описано у таблиці А.6. У цій матриці використані кольори для класифікації ризиків: зелений для прийнятних, жовтий для тих, що можна виправдати, та червоний для недопустимих ризиків.

Таблиця А.6 – Матриця ймовірності та впливу

Ймовірність виникнення ризику	**Вплив ризику**				
	0,05	**0,1**	**0,2**	**0,4**	**0,8**
0,9	0,045	0,09	0,18	0,3	0,72
0,7	0,035	0,07	0,14	0,28	0,56
0,5	0,025	0,05	0,1 **R3, R4**	0,2	0,4
0,3	0,015	0,03 **R1**	0,06	0,12	0,24
0,1	0,005	0,01	0,02	0,04	0,08 **R2**

Джерело: побудовано автором

В таблиці А.7 описано класифікацію ризиків за їх рівнем, відповідно до їх індексних значень. Детальний опис ризиків та стратегій їх управління описано у таблиці А.8.

Таблиця А.7 – Шкала оцінювання за рівнем ризику.

№	**Назва**	**Межі**	**Ризики, які входять(номера)**
1	Прийнятні	$0,005 \leq R \leq 0,05$	1, 3, 6, 9
2	Виправдані	$0,05 \leq R \leq 0,14$	2, 4, 5, 7, 8, 10
3	Недопустимі	$0,14 \leq R \leq 0,72$	

Джерело: побудовано автором

Таблиця А.8 – Ризики та стратегії реагування

ID	**Статус ризику**	**Опис ризику**	**Ймовірність виникнення**	**Вплив ризику**	**Ранг ризику**	**План А**	**Тип стратегії реагування**	**План Б**
RS-1	Відкритий	Відсутність кваліфікованих знань	Низька	Середній	0,02	1. Ознайомитися з потрібними технологіями (пришвидшені курси і тд.). 2. Обрати більш доступну технологію, якою можна замінити потреби.	Зменшення	Найняти розробника, який буде мати досвід та знання з відповідного стеку технологій
RS-2	Відкритий	Технічні збої у обладнанні	Низька	Середній	0,06	1. Регулярне обслуговування обладнання. 2. Мати резервні копії інформації та розроблюваної системи.	Зменшення	Мати запасне обладнання.

Продовження таблиці А.8

RS-3	Відкритий	Проблеми з інтернетом	Середня	Низький	0,025	Мати декілька джерел інтернету	Прийняття	Знайти найближче легкодоступне місце з інтернетом (кафе, коворкінг)
RS-4	Відкритий	Проблема з електропостачанням	Низька	Середній	0,06	Мати альтернативне джерело електроенергії.	Прийняття	Знайти найближче місце з можливість підключитися до електроенергії.
RS-5	Відкритий	Відсутність достатнього фінансування проекту	Середня	Середній	0,1	Підписання контракту з замовником щодо своєчасного та повноцінного фінансування послуг	Зменшення	Мати резервні кошти для перекриття недостатнього фінансування

Продовження таблиці А.8

RS-6	Відкритий	Захворювання одного з членів команди	Низька	Високий	0,04	Замінити розробника іншою людиною на короткий термін	Прийняття	Пошук підрядників або аутсорсинг певних задач
RS-7	Відкритий	Недостатня якість вихідного коду	Висока	Середній	0,14	Забезпечення увімкнення автоматичного збереження даних, створення локальних копій.	Зменшення	Створення локальних та хмарних копій.
RS-8	Новий	Поява альтернативного продукту	Середня	Середній	0,1	Уникання витоку інформації з кола розробників задля збереження унікальності продукту.	Прийняття	

Продовження таблиці А.8

RS-9	Відкритий	Недостатній контроль якості	Низька	Середній	0,06	Впровадження системи управління якістю та регулярні перевірки	Зменшення	Залучення зовнішніх аудиторів для незалежної оцінки якості
RS-10	Відкритий	Зміна вимог замовника	Низька	Високий	0,08	Регулярне узгодження вимог з замовником	Прийняття	Гнучке управління проектом, що дозволяє адаптуватися до змін

Джерело: побудовано автором

ДОДАТОК Б

Б1. Лістинг код модуля для автоматичного розподілення виконавця

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
import json

app = Flask(__name__)
CORS(app, resources={r"/*": {"origins":
"http://localhost:22443"}})

def series_to_list(series):
    return list(series.reset_index().to_dict(orient='records'))

def assign_tasks_fn(similarity_matrix, users_df, tasks_df):
    task_assignments = []
    assigned_tasks = set()

    for task_index in range(similarity_matrix.shape[1]):
        best_user_index = -1
        best_similarity = 0

        for user_index in range(similarity_matrix.shape[0]):
            if users_df.iloc[user_index]['roleKeywords'].strip()
== '':
                continue

            if similarity_matrix[user_index][task_index] >
best_similarity:
                best_user_index = user_index
                best_similarity =
similarity_matrix[user_index][task_index]

        if best_user_index != -1:
            assigned_tasks.add(task_index)
            task_assignments.append({
                "userId":
users_df.iloc[best_user_index]['userId'],
                "taskId": tasks_df.iloc[task_index]['id'],
            })

    for task_index in range(len(tasks_df)):
        if task_index not in assigned_tasks:
            task_assignments.append({
```

```
                "userId": None,
                "taskId": tasks_df.iloc[task_index]['id'],
            })

    return pd.DataFrame(task_assignments)

@app.route('/assign', methods=['POST'])
def assign_tasks():
    data_string = request.data.decode('utf-8')

    intermediate_data = json.loads(data_string)

    if isinstance(intermediate_data, str):
        data_json = json.loads(intermediate_data)
    else:
        data_json = intermediate_data

    users_df = pd.DataFrame(data_json['userList'])
    tasks_df = pd.DataFrame(data_json['taskList'])

    users_df['roleKeywords'].fillna('', inplace=True)

    vectorizer = TfidfVectorizer()

    combined_text = pd.concat([users_df['roleKeywords'],
tasks_df['text']])
    combined_vectors = vectorizer.fit_transform(combined_text)

    user_vectors = combined_vectors[:len(users_df)]
    task_vectors = combined_vectors[len(users_df):]
    similarity_matrix = cosine_similarity(user_vectors,
task_vectors)


    assigned_tasks_df = assign_tasks_fn(similarity_matrix,
users_df, tasks_df)

    return jsonify(series_to_list(assigned_tasks_df))

if __name__ == "__main__":
    app.run(debug=True)
```

Б2. Лістинг файлу controller/user.js

```
const { Sequelize } = require('sequelize');
const {
  models: {
    Projects, User, Role, Task, Status,
```

```
  },
} = require('../models');

module.exports = {
  addUser: async (req, res) => {
    const data = req.body;

    const newPermission = await Role.create({
      ...data,
    }, { raw: true });

    res.status(200).json({
      new: newPermission.dataValues,
    });
  },

  initialState: async (req, res) => {
    const { id: userId, currentProject } = req.session;

    const projects = await Projects.findAll({
      attributes: ['id', 'name', 'description', 'attachments'],
      raw: true,
    });

    let currPrjId = currentProject;
    if (!currPrjId && projects[0]) {
      currPrjId = projects[0].id;
    }

    const account = await User.findOne({
      where: { id: userId },
      attributes: ['id', 'roleId', 'name', 'surname', 'image',
'isActivated', 'email', 'type', 'notes', 'skills',
'isRestricted'],
      raw: true,
    });

    const itemStaff = await User.findAll({
      attributes: ['id', 'roleId', 'name', 'surname', 'image',
'isActivated', 'email', 'type', 'notes', 'skills',
'isRestricted'],
      raw: true,


      order: ['name', 'surname'],
      where: {
        isRestricted: false,
      },
    });

    const currentStaff = currPrjId ? await User.findAll({
```

```
      attributes: ['id', 'roleId', 'name', 'surname', 'image',
'isActivated', 'email', 'type', 'notes', 'skills', 'isRestricted',
        [Sequelize.col('Projects.Staff.id'), 'staffId'],
        [Sequelize.col('Projects.id'), 'projectId'],
      ],
      raw: true,
      include: {
        model: Projects,
        attributes: ['id'],
        where: {
          id: currPrjId,
        },
      },
      where: {
        isRestricted: false,
      },
      order: ['name', 'surname'],
    }) : [];

    const restricted = await User.findAll({
      attributes: ['id', 'roleId', 'name', 'surname', 'image',
'isActivated', 'email', 'type', 'notes', 'skills',
'isRestricted'],
      raw: true,
      where: {
        isRestricted: true,
      },
      order: ['name', 'surname'],
    });



    const roles = await Role.findAll({
      attributes: ['id', 'name', 'roles', 'keywords'],
      raw: true,
      order: ['name'],
    });

    const tasks = await Task.findAll({
      raw: true,
      where: { projectId: currPrjId },
      order: ['title', 'fromDate'],
    });


    const statuses = await Status.findAll({
      attributes: ['id', 'name', 'description', 'color',
'ordering'],
      order: ['ordering', 'name'],
      raw: true,
    });
```

```
    const currentRole = await Role.findOne({
      where: { id: account.roleId },
      raw: true,
    });

    res.status(200).json({
      projects: {
        items: projects,
        current: currPrjId,
      },
      staff: {
        items: itemStaff,
        currentItems: currentStaff,
        restrictedItems: restricted,
      },
      roles: {
        items: roles || [],
        self: currentRole || {},
      },
      account: {
        ...account,
        roles: currentRole.roles || [],
      },
      tasks: {
        items: [...tasks],
      },
      status: {
        items: [...statuses],
      },
    });
  },
  updateUser: async (req, res) => {
    const { userId } = req.params;

    await User.update({ ...req.body }, { where: { id: userId } });

    const userInfo = await User.findOne({
      where: { id: userId },
      attributes: ['id', 'roleId', 'name', 'surname', 'image',
'isActivated', 'email', 'type', 'notes', 'skills',
'isRestricted'],
      raw: true,
    });

    res.send({ user: userInfo });
  },
};
```

Б3. Лістинг файлу controller/task.js

```
const { Op } = require('sequelize');

const fetch = (...args) => import('node-fetch').then(({ default:
fetch }) => fetch(...args));

const {
  models: {
    Task, User, Staff, Role, Status,
  },
} = require('../models');
const { convertDescriptionToText } =
require('../helpers/tasksHelper');

const autoAssignTask = async (projectId, task) => {
  const staff = await Staff.findAll({
    attributes: ['userId', 'id'],
    where: { projectId },
    raw: true,
  });

  const usersPromises = staff.map(async ({ userId, id }) => {
    const userData = await User.findOne({
      where: { id: userId },
      attributes: [['id', 'userId'], 'roleId', 'isRestricted'],
      raw: true,
    });

    return {
      ...userData,
      staffId: id,
    };
  });

  const users = await Promise.all(usersPromises);

  const userListPromises = users.map(async (u) => {
    const userRole = await Role.findOne({
      where: { id: u.roleId },
      attributes: [['name', 'roleName'], ['keywords',
'roleKeywords']],
      raw: true,
    });
    return { userId: u.staffId, ...userRole, isRestricted:
u.isRestricted };
  });
  const userList = await Promise.all(userListPromises);
```

```
  let request;
  let answer = {};
  try {
    request = await fetch(
      'https://dmytroraiko.pythonanywhere.com/assign',
      {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'same-origin',
        body: JSON.stringify(
          {
            userList: userList.filter((user) =>
!user.isRestricted),
            taskList: [task],
          },
        ),
      },
    );
    answer = JSON.parse(await request.text());
  } catch (e) {
    console.log(e);
  }

  return answer[0] || {};
};

module.exports = {
  openedTasks: async (req, res) => {
    const { currentProject: projectId } = req.session;

    const tasks = await Task.findAll({
      raw: true,
      where: { projectId, statusId: 1 },
      attributes: ['description', 'title', 'id', 'statusId',
'type'],
    });

    const taskList = tasks.map((task) => {
      const { description, ...taskParams } = task;
      const text = convertDescriptionToText(description);
      return { ...taskParams, text };
    });

    const staff = await Staff.findAll({
      attributes: ['userId'],
      where: { projectId },
      raw: true,
    });
    const staffArr = staff.map((s) => s.userId);
```

```
    const users = await User.findAll({
      where: { id: { [Op.in]: staffArr } },
      attributes: [['id', 'userId'], 'roleId'],
      raw: true,
    });

    const userListPromises = users.map(async (u) => {
      const userRole = await Role.findOne({
        where: { id: u.roleId },
        attributes: [['name', 'roleName'], ['keywords',
'roleKeywords']],
        raw: true,
      });
      return { userId: u.userId, ...userRole };
    });
    const userList = await Promise.all(userListPromises);

    res.status(200).json({ taskList, userList });
  },
  add: async (req, res) => {
    const {
      title,
      description,
      parentTaskId,
      type,
      assignId,
      statusId,
      fromDate,
      dueDate,
      createdBy,
      projectId,
      attachments = [],
      autoAssign,
    } = req.body;

    let autoAssignedId = null;

    if (autoAssign) {
      const { userId } = await autoAssignTask(projectId, {
        title, id: -1, statusId, type, text: description,
      });
      autoAssignedId = userId;
    }

    const newTask = await Task.create({
      title,
      description,
      parentTaskId,
      type,
      assignId: assignId || autoAssignedId,
```

```
      statusId,
      fromDate: fromDate || null,
      dueDate: dueDate || null,
      attachments,
      projectId,
      createdBy,
    }, { raw: true });

    res.status(200).json({
      new: newTask.dataValues,
    });
  },
  update: async (req, res) => {
    const { taskId } = req.params;
    let { assignId } = req.body;
    const {
      id,
      updatedAt,
      createdAt,
      autoAssign,
      ...data
    } = req.body;

    if (autoAssign) {
      const { userId } = await autoAssignTask(data.projectId, {
        title: data.title, id: -1, statusId: data.statusId, type:
data.type, text: data.description,
      });
      assignId = userId;
    }

    await Task.update({
      ...data,
      dueDate: data.dueDate || null,
      fromDate: data.fromDate || null,
      assignId,
    }, {
      where: { id: taskId },
    });

    const task = await Task.findOne({ where: { id: taskId }, raw:
true });

    res.status(200)
      .json({
        task: {
          taskId,
          ...task,
        },
      });
```

```
  },
  deleteController: async (req, res) => {
    const { taskId } = req.params;

    await Task.destroy({ where: { id: taskId } });

    res.status(200).json({ taskId });
  },

  createStatus: async (req, res) => {
    const data = req.body;

    console.log(req.body);
    const newStatus = await Status.create({
      ...data,
    }, { raw: true });

    res.status(200).json({ new: newStatus.dataValues });
  },

  updateStatus: async (req, res) => {
    const { statusId } = req.params;
    const data = req.body;

    await Status.update({
      ...data,
    }, {
      where: { id: statusId },
    });

    const updatedEntity = await Status.findOne({
      attributes: ['name', 'description', 'color', 'ordering',
'id'],
      where: { id: statusId },
      raw: true,
    });

    res.status(200).json({ status: updatedEntity });
  },

  deleteStatus: async (req, res) => {
    const { statusId } = req.params;

    await Status.destroy({ where: { id: statusId } });
    // delete

    res.status(200).json({ deletedEntityId: statusId });
  },
};
```

Б4. Лістинг файлу pages/WorkspacePage.js

```
import React, { useMemo } from 'react';
import { useSelector } from 'react-redux';
import { useNavigate } from 'react-router-dom';
import moment from 'moment';

import { Table } from '../../table';
import { Button } from '../../ui';
import { ROUTE_NEW_TASK, ROUTE_TASK_EDIT } from
'../../routes/constants';
import { absoluteRoute, filterEntity } from '../../../helpers';
import { DATE_TIME_FORMAT } from '../../../common/date-time';
import { UserCardWidget, StatusWidget } from '../../widgets';
import { PageFilter } from '../../filter';
import { TASK_TYPES } from '../../../common/data';

const columns = [
  {
    title: null,
    id: 'id',
    width: 30,
    menu: true,
  },
  {
    title: 'Number',
    id: 'id',
    width: 50,
  },
  {
    title: 'Title',
    id: 'title',
    width: 300,
    style: {
      minWidth: 300,
    },
  },
  {
    title: 'Status',
    id: 'statusId',
    width: 70,
    component: (id) => <StatusWidget primary statusId={id} />,
  },
  {
    title: 'Type',
    id: 'type',
    width: 70,
    component: (type) => (
      <div style={{ whiteSpace: 'nowrap' }}>
```

```
        {TASK_TYPES[type]}
      </div>
    ),
  },
  {
    title: 'Assign to',
    id: 'assignId',
    width: 100,
    component: (id) => <UserCardWidget userId={id} />,
  },
  {
    title: 'Date start',
    id: 'fromDate',
    width: 140,
    style: { whiteSpace: 'nowrap' },
    component: (value) => (value ?
moment(value).format(DATE_TIME_FORMAT) : 'Not set'),
  },
  {
    title: 'Date end',
    id: 'dueDate',
    width: 140,
    style: { whiteSpace: 'nowrap' },
    component: (value) => (value ?
moment(value).format(DATE_TIME_FORMAT) : 'Not set'),
  },
  {
    title: 'Created by',
    id: 'createdBy',
    width: 100,
    component: (id) => <UserCardWidget userId={id} />,
  },
];

const WorkspacePage = () => {
  const tasks = useSelector(({ tasks: t }) => t.get('items'));
  const navigate = useNavigate();
  const filter = useSelector(({ ui }) => ui.get('filter'));

  const noDataAction = useMemo(() => (
    <Button
      onClick={() =>
navigate(absoluteRoute(ROUTE_TASK_EDIT.replace(':taskId', 'new-
task')))}
      leftIcon="add_rounded"
      text="Add task"
    />
  ), []);

  const filteredTasks = useMemo(() => {
```

```
    const dateStart = filter.get('dateStart');
    const dateEnd = filter.get('dateEnd');
    const f1 = filterEntity(tasks, 'assignId',
filter.get('staffFilter'));
    const f2 = filterEntity(f1, 'type',
filter.get('taskTypeFilter'));
    let f3 = f2;
    if (dateStart) {
      f3 = f3.filter((t) => t.get('fromDate') && +new
Date(t.get('fromDate')) >= dateStart);
    }
    let f4 = f3;
    if (dateEnd) {
      f4 = f4.filter((t) => t.get('dueDate') && +new
Date(t.get('dueDate')) <= dateEnd);
    }
    return filterEntity(f4, 'statusId',
filter.get('statusFilter'));
  }, [filter, tasks]);

  return (
    <>
      <PageFilter
        innerPage
        btns={
          <Button
            leftIcon="add_rounded"
            text="New task"
            onClick={() =>
navigate(absoluteRoute(ROUTE_NEW_TASK))}
            btnProps={{ style: { padding: '3px 5px', minWidth: 70
} }}
          />
        }
      />
      <Table
        columns={columns}
        data={filteredTasks}
        noDataAction={noDataAction}
        tableProps={{
          style: {
            minWidth: '100%',
          },
        }}
      />
    </>
  );
};

export default WorkspacePage;
```

Б5. Лістинг файлу сторінки редагування

```
import React, {
  useCallback, useEffect, useMemo, useState,
} from 'react';
import { Map } from 'immutable';
import styled from 'styled-components';
import ImmutablePropTypes from 'react-immutable-proptypes';
import PropTypes from 'prop-types';
import { useDispatch } from 'react-redux';
import moment from 'moment';
import { useNavigate } from 'react-router-dom';

import { FlexWrapper, H3 } from '../../../../styled-components';
import {
  Button, DateTimePicker, IconButton, Input, QuillField, Checkbox,
} from '../../../ui';
import Form from '../../../form/Form';
import {
  StatusSelect, ProjectSelect, StaffSelect, TaskSelect,
} from '../../../select';
import { addTask, editTask, removeTask } from
'../../../../actions/tasks';
import { DATE_TIME_FORMAT } from '../../../../common/date-time';
import DetailBlock from '../task/Detail';
import { UserCardWidget } from '../../../widgets';
import TaskTypeSelect from '../../../select/TaskTypeSelect';
import { ConfirmPopup } from '../../../modals';
import { absoluteRoute } from '../../../../helpers';
import { ROUTE_TASK_EDIT } from '../../../routes/constants';

const TaskEditPage = ({
  taskId, errors, initFormData, taskData,
}) => {
  const [confirmDeleting, setConfirmDeleting] = useState(false);

  const [formData, setFormData] = useState(initFormData);
  const dispatch = useDispatch();
  const navigate = useNavigate();

  useEffect(() => {
    const assignId = taskData?.get('assignId');
    if (formData.autoAssign && assignId) {
      setFormData((state) => ({ ...state, autoAssign: false,
assignId }));
    }
  }, [taskData]);

  const onSubmit = useCallback(() => {
```

```
  if (taskId) {
    dispatch(editTask(taskId, formData));
  } else {
    dispatch(addTask(formData))
      .then(({ new: data }) => navigate(absoluteRoute(
        ROUTE_TASK_EDIT.replace(':taskId', data.id),
      )));
  }
}, [formData, taskId]);

const deleteFn = useCallback(() => {
  dispatch(removeTask(taskId));
}, [taskId]);

const onChange = useCallback((field, value) => {
  const data = { ...formData };
  data[field] = value;
  setFormData(data);
}, [formData]);

const onAttachmentChange = (id) => {
  const data = { ...formData };
  const attchsArr = data.attachments || [];
  attchsArr.push(id);
  data.attachments = attchsArr;
  setFormData(data);
};

const onAutoAssignChange = () => {
  const data = { ...formData };
  data.autoAssign = !data.autoAssign;
  setFormData(data);
};

const actions = useMemo(() => (
  <ActionWrapper>
    {taskId ? (
      <IconButton
        onClick={() => setConfirmDeleting(true)}
        iconName="trash"
      />
    ) : null }
    <Button onClick={onSubmit}>
      {taskId ? 'Save' : 'Create'}
    </Button>
  </ActionWrapper>
), [onSubmit, taskId]);

const datetimeOnChange = useCallback((field, value) => {
  const date = moment(value).format(DATE_TIME_FORMAT);
```

```
    onChange(field, date);
  }, [onChange]);

  return (
    <Form
      initial={initFormData}
      onSubmit={onSubmit}
      errors={errors}
      withActions={false}
      wrapperStyles={{ padding: '0 15px' }}
    >
      <RootWrapper>
        <VerticalWrapper>
          <TitleWrapper>
            <IconButton iconName="back" onClick={() => navigate(-
1)} />
            {taskId ? (
              <IdText>[{taskId}]</IdText>
            ) : null}
            <Input
              borders={['bottom']}
              fullWidth
              label="Title"
              placeholder="Title"
              error={errors.get('title')}
              value={formData.title}
              onChange={(e) => onChange('title', e.target.value)}
            />
          </TitleWrapper>
          <QuillField
            placeholder="Task details"
            onChange={onChange}
            name="description"
            value={formData.description}
            initialValue={initFormData.description}
            attachmentChange={onAttachmentChange}
          />
        </VerticalWrapper>
        <VerticalWrapper
          style={{
            gap: 20,
            minWidth: 300,
            maxWidth: 300,
            paddingLeft: 15,
            borderLeftWidth: 1,
          }}
        >
          {actions}

          <StatusSelect
```

```
  name="statusId"
  error={errors.get('statusId') || ''}
  value={formData.statusId}
  setValue={(value) => onChange('statusId', value)}
/>

<TaskTypeSelect
  name="type"
  error={errors.get('type') || ''}
  value={formData.type}
  setValue={(value) => onChange('type', value)}
/>

<ProjectSelect
  name="projectId"
  error={errors.get('projectId') || ''}
  value={formData.projectId}
  setValue={(value) => onChange('projectId', value)}
/>

<Checkbox
  label="Assign Automatically"
  value={formData.autoAssign}
  onChange={onAutoAssignChange}
/>

{!formData.autoAssign && (
  <StaffSelect
    name="assignId"
    error={errors.get('assignId') || ''}
    value={formData.assignId}
    setValue={(value) => onChange('assignId', value)}
    showClearBtn
  />)}

<TaskSelect
  label="Related task"
  name="parentTaskId"
  placeholder="Related task"
  error={errors.get('parentTaskId') || ''}
  value={formData.parentTaskId}
  setValue={(value) => onChange('parentTaskId', value)}
  withoutItems={[taskId]}
  showClearBtn
/>

<DateTimePicker
  size="lg"
  // error={errors.get('fromDate') || ''}
  label="Date start"
```

```
            placeholder="Date start"
            value={formData.fromDate}
            setValue={(value) => datetimeOnChange('fromDate',
value)}
            maxDate={moment(formData.dueDate) || ''}
            fullWidth
          />
          <DateTimePicker
            size="lg"
            // error={errors.get('dueDate') || ''}
            label="Date end"
            placeholder="Date end"
            value={formData.dueDate}
            setValue={(value) => datetimeOnChange('dueDate',
value)}
            minDate={moment(formData.fromDate) || ''}
            fullWidth
          />

          {taskId ? (
            <DetailBlock
              label="Created By:"
              value={<UserCardWidget userId={formData.createdBy}
/>}
            />) : null }

          {formData.attachments.length ? 'Attachments' : null}
        </VerticalWrapper>
      </RootWrapper>
      {taskId ? (
        <ConfirmPopup
          onClose={() => setConfirmDeleting(false)}
          cbOnConfirm={deleteFn}
          message={`Do you really want to delete the task
(#${taskId})?`}
          isShown={confirmDeleting}
          isWarning
        />
      ) : null}
    </Form>
  );
};

TaskEditPage.propTypes = {
  initFormData: PropTypes.object.isRequired,
  errors: ImmutablePropTypes.map,
  taskData: ImmutablePropTypes.map,
  taskId: PropTypes.string,
};
```

```
TaskEditPage.defaultProps = {
  errors: new Map(),
  taskData: new Map(),
  taskId: null,
};

export default TaskEditPage;

const RootWrapper = styled(FlexWrapper)`
  justify-content: space-between;
  column-gap: 25px;
  width: 100%;
  height: 100%;
`;

const VerticalWrapper = styled(FlexWrapper)`
  flex-direction: column;
  gap: 15px;
  height: 100%;
  width: 100%;
  border-width: 0;
  border-style: solid;
  border-color: ${({ theme }) => theme.stroke.light};
`;

const ActionWrapper = styled(FlexWrapper)`
  justify-content: flex-end;
  flex-wrap: wrap;
  gap: 10px;
  width: 100%;
`;

const TitleWrapper = styled(FlexWrapper)`
  align-items: center;
  gap: 10px;
  width: 100%;
`;

const IdText = styled(H3)`
  font-style: italic;
  padding: 0;
  margin: 0;
  margin-right: 5px;
  font-weight: 100;
  color: ${({ theme }) => theme.color.additionalLight}
`;
```