Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ДЛЯ УПРАВЛІННЯ РОЗРОБКОЮ ІТ-ПРОЄКТІВ

Виконав: студент 4 курсу, групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напрямку підготовки, спеціальності)
Гненний І. О.
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. КН
Іванчук Я. В.
(прізвище та ініціали)
«07» 06 2024 р.

Рецензент: д.т.н., проф., зав. каф. КСУ
Ковтун В. В.
(прізвище та ініціали)

«07» 06 2024 р.

Допущено до захисту
Зав. кафедри проф., д.т.н. Яровий А.А.
«07» 06 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«07» 03 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Гненному Іллі Олександровичу

1. Тема роботи: Програмний модуль для управління розробкою IT-проєктів.
Керівник роботи: Іванчук Ярослав Володимирович, д.т.н., проф.
затверджені наказом вищого навчального закладу "11" 03 2024 року №80.
2. Термін подання студентом роботи 27.05.2024 р.
3. Вихідні дані до роботи: персональні дані про користувачів; API для взаємодії веб-додатку та сервера, веб-додаток для управління розробкою IT-проєктів; кількість одночасних користувачів не менш 15 чол.; кількість проектів не менш 5 шт.; операційні системи: Windows, Linux; браузери: Chrome, FireFox.
4. Зміст текстової частини (перелік питань, які потрібно розробити): аналіз предметної області; розробка архітектури програмного модуля; проєктування взаємодії функціональних об'єктів інформаційної системи; розробка алгоритму функціонуання програмного модуля для управління розробкою IT-проєктів; програмна реалізація та тестування програмного модуля для управління розробкою IT-проєктів.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): загальна структурна схема програмного модуля для управління розробкою IT-проєктів; загальна схема взаємодії WEB-додатка із SMTP сервером; типова схема модуля управління проєктами, командами, часом; схема алгоритму функціонування програмного модуля для упраління розробкою IT-проєктів; схема алгоритму реєстрації користувача, створення проєкта у модулі для упраління розробкою IT-проєктів; схема алгоритму створення завдання у модулі для упраління розробкою IT-проєктів; загальний вигляд вікна авторизації, реєстрації, системи ManageD для управління розробкою IT-проєктів; загальний вигляд вікна зі списком проєктів, мітінгів, аналітики затрат часу і виконаних задач.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Іванчук Я. В., д.т.н., проф.. каф. КН		

7. Дата видачі завдання 07.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Приміт-ка
		Початок	Закінчен-ня	
1	Аналіз предметної області, існуючих методів розробок, актуальність впровадження систем для управління розробкою ІТ-проєктів, сучасних систем для управління розробкою ІТ-проєктів	06.05.24	10.05.24	
2	Проєктування програмного модулю для управління розробкою ІТ-проєктів	11.05.24	15.05.24	
3	Програмна реалізація модулю для управління розробкою ІТ-проєктів	16.05.24	29.05.24	
4	Розробка інструкції користувача.	30.05.24	03.06.24	
5	Оформлення пояснювальної записки	03.06.24	04.06.24	
6	Оформлення матеріалів до захисту БКР.	04.06.24	06.06.24	

Студент _____ **Гненний І. О.**
(підпис) *(прізвище та ініціали)*

Керівник роботи _____ **Іванчук Я. В.**
(підпис) *(прізвище та ініціали)*

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 77 сторінок формату А4, на яких є 27 рисунків, список використаних джерел містить 19 найменувань.

Дана бакалаврська робота присвячена розробці модуля управління проєктами з використанням сучасних технологій та методів. Модуль управління проєктами дозволяє користувачам забезпечувати ефективне управління проєктами, командами, ресурсами та часом. У результаті проведених експериментів та тестувань було підтверджено високу функціональність та зручність використання розробленого модуля. Впровадження такого модуля дозволить покращити організацію робочих процесів та підвищити ефективність виконання завдань, що має широкі перспективи застосування в ІТ-компаніях.

Ключові слова: управління ІТ-проєктами, менеджмент, алгоритм, розробка, програмне забезпечення.

ABSTRACT

The bachelor thesis consists of 77 pages of A4 format, on which there are 27 figures, the list of used sources contains 19 items.

This bachelor thesis is devoted to the development of a project management module using modern technologies and methods. The project management module allows users to effectively manage projects, teams, resources and time. As a result of the conducted experiments and testing, the high functionality and ease of use of the developed module were confirmed. The implementation of such a module will improve the organization of work processes and increase the efficiency of task performance, which has wide prospects for application in IT companies.

Keywords: IT project management, management, algorithm, development, software.

ЗМІСТ

ВСТУП

Актуальність дослідження. Актуальність теми дослідження обумовлена постійним зростанням кількості ІТ-проєктів та потребою у вдосконаленні методів їхнього управління. Сучасні підходи до управління ІТ-проєктами, такі як Agile, Scrum, та Kanban, вимагають використання спеціалізованих інструментів, які забезпечують гнучкість, прозорість та ефективність робочих процесів. З огляду на зростаючу складність і масштабність проєктів, традиційні методи управління стають недостатніми для забезпечення належного контролю та координації. Програмні модулі для управління ІТ-проєктами дозволяють автоматизувати багато аспектів управління, від планування і розподілу завдань до відстеження прогресу та управління ресурсами. Додатково, сучасний ІТ-ринок вимагає швидкої адаптації до змінних умов і потреб клієнтів, що робить необхідним застосування інтегрованих рішень для управління проєктами. Використання програмних модулів забезпечує не лише автоматизацію процесів, але й надає аналітичні дані [2-4]. Розробка ефективних програмних модулів для управління ІТ-проєктами є надзвичайно важливою для підвищення продуктивності команд та покращення результатів проєктів. В умовах, коли ринок ІТ-послуг стрімко розвивається, а конкуренція стає жорсткішою, впровадження таких модулів стає ключовим фактором успіху для компаній, що прагнуть залишатися лідерами у своїй галузі [3, 4].

Метою дослідження є підвищення ефективності проєктування інформаційних систем за допомогою програмного модуля для управління розробкою ІТ-проєктів, що дозволить враховувати усі потенційні ризики, бюджет, вимоги та обмеження, а також потреби у масштабуванні результатів проєктування.

Об’єктом дослідження є процеси управління розробкою ІТ-проєктів, зокрема, методи планування, розподілу завдань, контролю за виконанням робіт та управління ресурсами.

Предметом дослідження є алгоритми і програмне забезпечення для програмного модуля системи управління розробкою ІТ-проєктів.

Задачі дослідження:

1. Провести аналіз методів розробок ІТ-проєктів.
2. Провести аналіз актуальності впровадження систем для управління розробкою ІТ-проєктів.
3. Провести аналіз проблем існучих програмних систем управління розробкою ІТ-проєктів.
4. Розробка архітектури і визначення форми взаємодії функціональних об’єктів інформаційної системи.
5. Розробити алгоритми роботи модуля для управління розробкою ІТ-проєктів
6. Провести аналіз мов програмування для на написання модуля для управління розробкою ІТ-проєктів.
7. Виконати програмну реалізацію модуля для управління розробкою ІТ-проєктів.
8. Провести тестування програмного модулю для управління розробкою ІТ-проєктів.

Апробація роботи.

Робота апробувалась на наступних конференціях:

- Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації 2024 року, доповідь «Аналіз перспективи розробки програмного забезпечення для управління розробкою ІТ-проєктів» [1].

Публікації: електронні тези науково-технічної конференції «Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації» [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Методи розробки ІТ-проєктів

Аналіз методів розробки ІТ-проєктів може включати в себе розгляд різних підходів та методологій, які використовуються для керування та виконання проєктів в галузі інформаційних технологій. Ось деякі ключові аспекти для аналізу:

1. Waterfall (каскадний метод) — послідовний метод розробки програмного забезпечення, названий так через діаграму, схожу на водоспад (як показано на рисунку 1.1) [5]. Це традиційна модель розробки, де проєкт розбивається на послідовні етапи: аналіз, проєктування, розробка, тестування та впровадження.

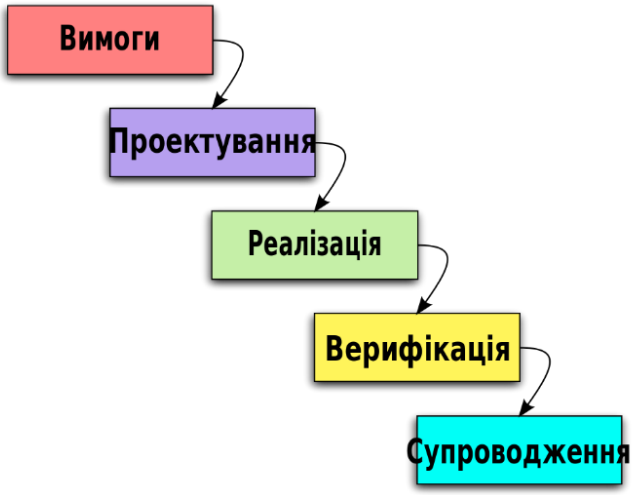

Рисунок 1.1 – Типова схема методу проєктування ІТ-проєктів типу «Водоспадна модель (Waterfall)»

- Переваги: Простота у використанні та розумінні, ясно визначені етапи та вимоги.

- Недоліки: Мало гнучкості, важко вносити зміни на пізніших етапах, ризик не врахування зміни вимог клієнта.

2. Agile (Гнучкий метод) — клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки

еволюціонують через співпрацю між багатофункціональними командами здатними до самоорганізації. Гнучка розробка — засіб для підвищення продуктивності розробників програмного забезпечення [6]. Agile - це набір методологій, які ставлять акцент на ітеративному та інкрементальному підході до розробки.

- Переваги: Гнучкість, здатність швидко реагувати на зміни, більше зосередження на співпраці з клієнтом та вартості продукту.

- Недоліки: Потребує більшого впровадження та підтримки, може бути складним у використанні для деяких команд.

3. Scrum — це кістяк процесу, який складається з набору методів і ролей, які були попередньо визначені. Також він є одним з найпопулярніших фреймворків Agile, який використовує ітеративний підхід та робочі спринти. Основні дійові особи: ScrumMaster, який відповідає за процеси, керує ними та є керівником проєкту; Власник продукту, який представляє інтереси кінцевих користувачів та інших зацікавлених сторін продукту; і команда розробників [7].

Працівники створюють функціональний ріст програмного забезпечення протягом кожного спринту, що триває від п’ятнадцяти до тридцяти днів, або тривалість, яка визначається командою [7].

Кожен спринт починається з набору можливостей, які виконуються. Цей етап, відомий як продуктовий backlog, або документація запитів на виконання робіт, має найвищу пріоритетність за рівнем роботи, яка повинна бути виконана. Запити на виконання робіт, які були визначені під час наради з планування спринту, переміщуються в етап спринту. Власник продукту повідомляє про завдання, які він хоче виконати протягом цієї наради. Після цього команда оцінює свою потенційну продуктивність, щоб завершити необхідні компоненти протягом наступного спринту. Протягом спринту команда виконує певний список завдань. Впродовж цього періоду ніхто не має права змінювати список запитів на виконання робіт; це означає, що вимоги заморожені протягом спринту [7]:

- Переваги: Прозорість процесу, створення потужної командної роботи, швидке впровадження та здатність швидко реагувати на зміни.

- Недоліки: Вимагає великої участі команди, може бути викликом для менш досвідчених команд.

4. Kanban - це методологія, яка акцентується на візуалізації робочого процесу та обмеженні робочих завдань, які можна виконати одночасно. Канбан став ефективним інструментом в управлінні системою виробництва у цілому, і довів себе, як відмінний спосіб пропагування вдосконалень. Проблемні місця виявляються за зменшенням кількості продуктів у циркуляції. Однією з переваг канбан є встановлення верхньої межі на кількість деталей, що очікують опрацювання, й уникнення таким чином перевантаження системи виробництва [8].

- Переваги: Простота в розумінні та використанні, зменшення перевантаження та підвищення продуктивності.

- Недоліки: Може вимагати більшої уваги до деталей, менше структурованої методології порівняно з Scrum.

1.2 Аналіз актуальності впровадження систем управління розробкою IT-проєктів

Впровадження систем управління розробкою IT-проєктів є надзвичайно актуальним з точки зору ефективності, якості та успішності проєктів. Ось ключові аспекти актуальності впровадження таких систем [9]:

1. Керування складністю проєктів: IT-проєкти часто мають велику складність через розмаїття вимог, технологічні стеки та учасників. Системи управління проєктами дозволяють керувати цією складністю та забезпечувати виконання проєктів у встановлені терміни та бюджет.

2. Керування командою та ресурсами: Ефективне використання ресурсів, планування завдань та розподіл робочого часу є ключовими для успіху проєктів. Системи управління розробкою надають інструменти для керування цими аспектами, забезпечуючи оптимальне використання ресурсів та підтримку командного співробітництва.

3. Контроль якості та забезпечення якості: Забезпечення високої якості програмного забезпечення є важливим аспектом будь-якого ІТ-проєкту. Системи управління розробкою допомагають встановлювати процеси контролю якості, автоматизувати тестування та забезпечувати відслідковування помилок для швидкого виправлення.

4. Зменшення ризиків та забезпечення відкритості: Впровадження систем управління розробкою дозволяє керувати ризиками та забезпечувати відкритість та прозорість у проєкті. Це дозволяє уникнути непередбачених проблем та забезпечити вчасну реакцію на зміни у вимогах чи умовах.

5. Підтримка гнучких методологій розробки: Багато сучасних систем управління розробкою підтримують гнучкі методології розробки, такі як Scrum або Kanban, що відповідає вимогам сучасних ІТ-проєктів.

Загалом, впровадження систем управління розробкою ІТ-проєктів є необхідним кроком для забезпечення успішного виконання проєктів у сучасному інформаційному середовищі. Вони допомагають підвищити продуктивність, зменшити ризики та забезпечити високу якість продукту.

1.3 Огляд існуючих програмних систем управління розробкою ІТ-проєктів

Програма для управління розробкою ІТ-проєктів Asana була заснована у 2008 році двома колишніми співробітниками Facebook – Дастіном Московіцем та Джастіном Розенштейном. Головною метою створення Asana було забезпечити ефективну спільну роботу та управління проєктами в організаціях. Назва Asana походить від слова "асана", яке означає позу в йозі, що вказує на спокій та концентрацію [10].

За роки свого існування Asana зазнала значного росту та розвитку. Компанія залучила інвестиції, включаючи відомих інвесторів, таких як Peter Thiel та Mark Zuckerberg. У 2012 році Asana випустила публічну версію свого програмного забезпечення, що стало доступним для широкої громадськості.

Основні аспекти:

1. Концепція продукту: Asana створена з метою полегшення спільної роботи та управління проєктами в командному середовищі. Вона надає засоби для створення, організації та виконання завдань, а також для спілкування та співпраці між учасниками команди.

2. Популярність: Asana стала одним з найбільш популярних інструментів управління проєктами у світі. Вона використовується тисячами компаній та організацій різних розмірів у більш ніж 190 країнах.

3. Інновації та розвиток: Команда Asana постійно вдосконалює продукт, додаючи нові функції та можливості. Вони активно реагують на потреби користувачів та тенденції в управлінні проєктами, щоб забезпечити максимальну корисність свого продукту.

4. Корпоративна культура: Asana славиться також своєю корпоративною культурою, яка акцентується на відкритості, співпраці та розвитку співробітників.

На рисунку 1.2 показаний інтерфейс Asana.

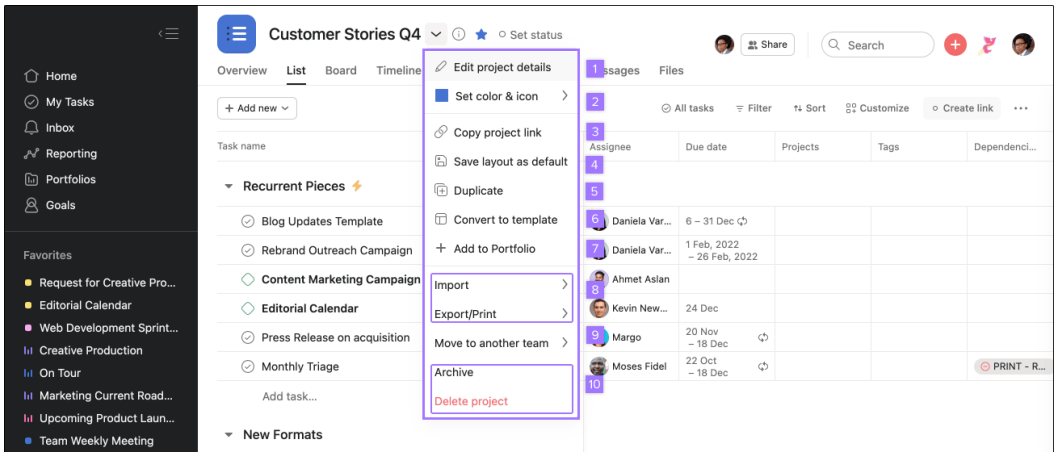

Рисунок 1.2 – Загальний вигляд інтерфейсного вікна програми Asana

Розглянемо детальніше про особливості та можливості Asana:

1. Створення завдань та проєктів: Asana дозволяє створювати завдання та організовувати їх у вигляді проєктів. Завдання можна легко назначати учасникам команди, встановлювати строки, призначати пріоритети та додавати необхідні деталі.

2. Календар та строки: Asana має вбудований календарний перегляд, що дозволяє відстежувати строки завдань та проєктів. Це допомагає уникнути просочення строків та керувати часовими ресурсами ефективніше.

3. Завдання та підзавдання: Asana дозволяє розбивати завдання на більш малі компоненти за допомогою підзавдань. Це спрощує виконання складних завдань та дозволяє більш ефективно керувати їх виконанням.

4. Спільна робота та комунікація: Asana забезпечує можливості спільної роботи та комунікації в межах команди. Користувачі можуть обговорювати завдання, ділитися коментарями та файлами, що сприяє покращенню комунікації та збільшенню продуктивності.

5. Інтеграція з іншими інструментами: Asana підтримує інтеграцію з багатьма іншими сервісами та інструментами, такими як Google Drive, Slack, Dropbox, Microsoft Teams та інші. Це дозволяє забезпечити гладку роботу з улюбленими інструментами вашої команди.

6. Інтуїтивний інтерфейс користувача: Asana відома своїм інтуїтивно зрозумілим інтерфейсом, що дозволяє новим користувачам швидко освоювати інструмент та починати працювати над проєктами.

У цілому, Asana - це потужний інструмент для управління проєктами, який дозволяє командам ефективно організовувати роботу, відстежувати прогрес та досягати поставлених цілей.

Інструмент управління проєктами та відстеження проблем Jira розроблений австралійською компанією Atlassian. Початково випущений у 2002 році як інструмент для відстеження проблем у сфері програмного забезпечення, Jira з часом став широко використовуватися командами Agile розробників для відстеження помилок, історій, епіків та інших завдань [11]. На рисунку 1.3 показаний інтерфейс Jira.

Продукт постійно розвивається, а компанія Atlassian розширила платформу Jira, щоб вона задовольняла потреби різних типів команд. На сьогоднішній день, Jira пропонує ряд внутрішніх продуктів, серед яких:

- Jira Software: Спеціально адаптований для роботи розробників програмного забезпечення. У цьому продукті передбачені проблеми та робочі процеси для планування, розробки та випуску програмного продукту. Також підтримуються популярні методології управління проєктами, такі як "Scrum" та "Канбан".

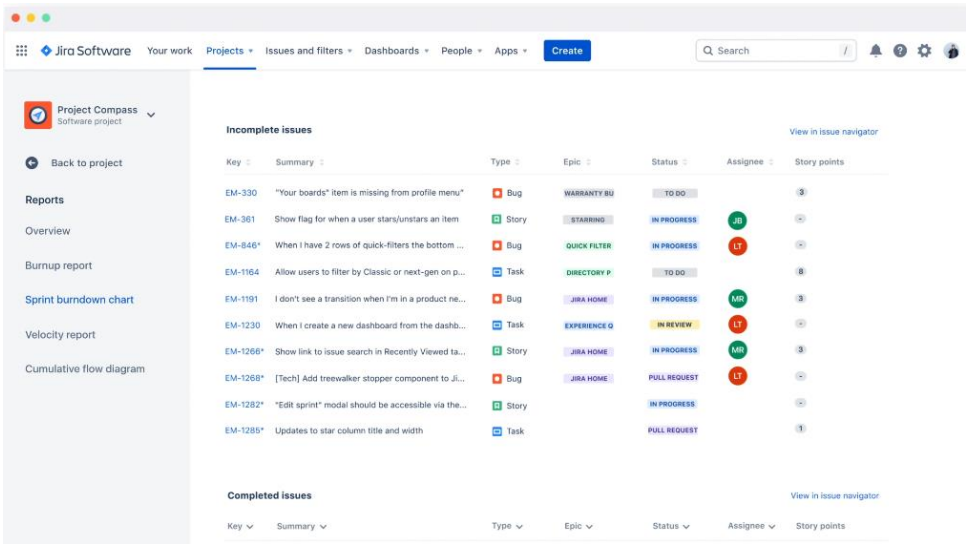

Рисунок 1.3 – Загальний вигляд інтерфейсного вікна програми Jira

- Jira Core: Розроблений як система управління проєктами, що може використовуватися в різних галузях, не обмежених сферою IT.

- Jira Service Desk: Це додатковий модуль, розроблений для IT-команд, що дозволяє менеджерам колл-центрів, агентам служби підтримки та іншим спеціалістам керувати квитками, інцидентами та змінами.

Технічно Jira написана на Java і розгортається як стандартний файл Java WAR у контейнері Servlet Java, такому як Tomcat. Користувачі взаємодіють з Jira через веб-браузер, використовуючи веб-інтерфейс. Вона підтримує виклики віддалених процедур через REST, а раніше також підтримувала SOAP і XML-RPC, але ці протоколи були відмінені у версії 7.0.

Щоб обробляти веб-запити, надіслані користувачами, Jira використовує OpenSymphony WebWork 1. Цей фреймворк MVC дозволяє обробляти кожен запит за допомогою дії WebWork, що зазвичай використовує інші об'єкти, такі як класи утиліти та менеджера для виконання завдання. Що стосується шару View, то Jira використовує JSP, тобто JavaServer Pages, для генерації більшості HTML, що подається користувачеві як відповідь на їх веб-запити.

Програма для управління розробкою IT-проєктів Trello була розробленою командою Fog Creek Software (згодом перейменовано на Trello, Inc.), в якій брав участь Джоел Спольськи, співзасновник Stack Overflow. Перша версія Trello була запущена у вересні 2011 року. У 2017 році Trello було придбано компанією Atlassian, яка спеціалізується на розробці програмного забезпечення для управління проєктами та спільної роботи [12]. На рисунку 1.4 показаний інтерфейс Trello.

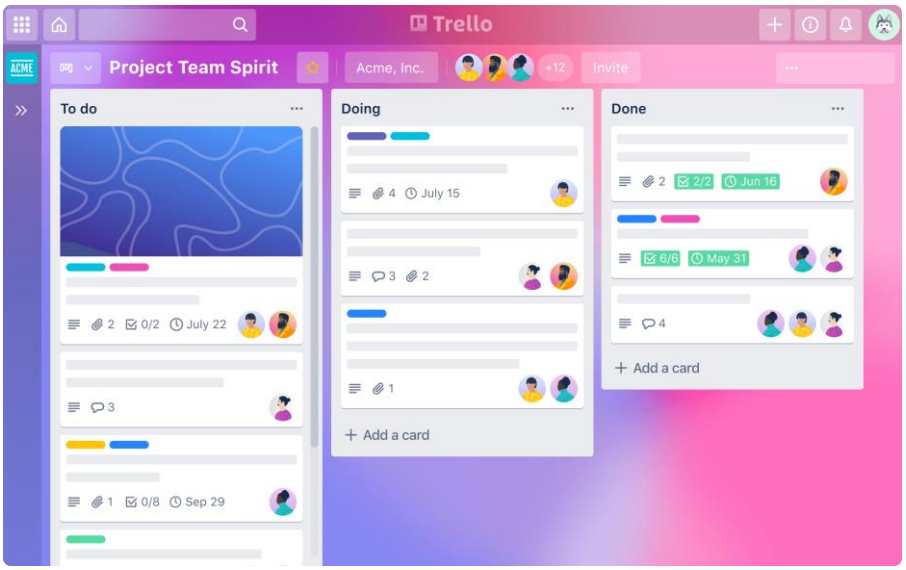

Рисунок 1.4 – Загальний вигляд інтерфейсного вікна програми Trello

Основні аспекти:

1. Концепція продукту: Trello базується на концепції карточок на дошці. Кожна дошка в Trello представляє собою проєкт або задачу, а карточки - це завдання або ідеї, які можна переміщувати між різними списками на дошці. Це дозволяє користувачам візуально організовувати свої завдання та відстежувати їх прогрес.

2. Списки та карточки: Trello дозволяє користувачам створювати необмежену кількість списків на дошці та додавати до них карточки. Карточки можуть містити текстовий опис, додаткові файли, дедлайни, чеклисти тощо.

3. Поділ на команди та проєкти: Trello дозволяє користувачам створювати команди для спільної роботи над проєктами. В рамках кожної команди можна створювати власні дошки та назначати користувачів на різні ролі.

4. Інтеграція з іншими сервісами: Trello підтримує інтеграцію з багатьма іншими інструментами, такими як Google Drive, Slack, Dropbox, GitHub та інші. Це дозволяє забезпечити гладку роботу з улюбленими інструментами користувачів.

5. Мобільний додаток: Trello має мобільний додаток для iOS та Android, що дозволяє користувачам отримувати доступ до своїх дошок та картичок з будь-якого пристрою та в будь-який час.

6. Безкоштовний та платний варіанти: Trello пропонує безкоштовний тарифний план з базовими функціями, а також платні тарифні плани з розширеними можливостями, такими як більше привілеїв для команд, інтеграція з більшою кількістю сервісів тощо.

Розглянемо детальніше про особливості та можливості Trello:

1. Дошки, списки та карточки: Основна структура Trello - це дошки, які можуть містити різні списки, а кожен список містить карточки. Карточки можуть містити текстовий опис, прикріплені файли, дедлайни, чек-листи, мітки тощо. Користувачі можуть переміщувати карточки між списками, щоб відстежувати прогрес та організовувати завдання.

2. Коментарі та обговорення: Кожна карточка має функцію коментарів, що дозволяє користувачам обговорювати завдання, ділитися ідеями та надавати зворотний зв'язок. Це спрощує комунікацію в межах команди та допомагає вирішувати завдання спільними зусиллями.

3. Чек-листи: Trello дозволяє додавати до карточок чек-листи, які допомагають у відстеженні проміжних кроків у виконанні завдання. Користувачі можуть позначати виконані елементи чек-листа, щоб відобразити прогрес.

4. Мітки та категорії: Trello надає можливість додавати мітки до карточок для категоризації та відстеження різних аспектів завдань. Мітки можуть мати різні кольори і назви, що дозволяє створювати власні системи класифікації.

5. Дедлайни та сповіщення: Користувачі можуть встановлювати дедлайни для карточок та списків, щоб відстежувати терміни виконання завдань. Trello автоматично надсилає сповіщення про наближення дедлайнів, щоб користувачі могли вчасно реагувати на потреби проєкту.

6. Шаблони та регулярні завдання: Trello дозволяє створювати шаблони дошок та карточок, що спрощує процес організації проєктів та завдань. Крім того, користувачі можуть налаштовувати регулярні повторювані завдання для автоматизації рутинних процесів.

7. Інтеграція з іншими сервісами: Trello підтримує інтеграцію з багатьма популярними сервісами, такими як Google Drive, Dropbox, Slack, GitHub, Jira тощо. Це дозволяє користувачам легко обмінюватися інформацією та інтегрувати Trello з іншими інструментами, які вони вже використовують.

Загалом, Trello є простим у використанні, але потужним інструментом для управління проєктами та спільної роботи команд. Його візуальний підхід до організації завдань робить його популярним серед широкого кола користувачів.

1.4 Висновок до розділу 1

У першому розділі розглянуто методи розробки IT-проєктів, такі як Waterfall, Agile, Scrum та Kanban. Проаналізовано актуальність впровадження таких систем, визначено їх основні принципи, переваги та недоліки. Окрім того, досліджено сучасні програмні продукти для управління IT-проєктами, такі як Asana, Jira та Trello. Проаналізовано їх функціональність, зручність використання, а також переваги та недоліки. та розглянуто сучасні програмні продукти для управління IT-проєктами, такі як Asana, Jira, Trello, а також їх переваги і недоліки.

2 РОЗРОБКА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ УПРАВЛІННЯ РОЗРОБКОЮ ІТ-ПРОЄКТІВ

2.1 Розробка архітектури програмного модуля

При розробці структури програмного забезпечення для організації роботи управління розробкою ІТ-проєктів доцільно застосувати модульне програмування. Це підхід, орієнтований на зменшення складності програмних продуктів та можливість повторного використання окремих рішень в інших проєктах. Модульна структура дозволяє розділити функціональність програми на незалежні модулі, кожен з яких містить все необхідне для виконання конкретної задачі [13].

Модульне програмування дозволяє зменшити обсяг коду, зробити його більш зрозумілим, прискорити процес написання та тестування програм, а також знизити витрати на їх експлуатацію.

Якісне модульне програмування характеризується організацією програми з максимальною кількістю незалежних модулів і мінімальною кількістю зв'язків між ними [13].

Застосування модульного підходу особливо актуальне в контексті управління розробкою ІТ-проєктів, де існує потреба у гнучкості та можливості швидкого внесення змін без порушення роботи всього програмного комплексу. Кожен модуль може бути розроблений, протестований та вдосконалений окремо, що знижує ризик виникнення помилок і спрощує процес підтримки та оновлення системи [13].

Наприклад, модулі управління проєктами, командами, ресурсами, часом та безпекою можуть бути розроблені незалежно один від одного, що забезпечує можливість їхнього паралельного розвитку та тестування.

У випадку потреби додавання нових функцій або модифікації існуючих, зміни вносяться тільки в окремий модуль, мінімально впливаючи на інші частини системи.

Модульне програмування також сприяє підвищенню якості програмного забезпечення завдяки можливості багаторазового використання перевірених та

надійних модулів у різних проєктах. Це забезпечує не лише економію часу та ресурсів на розробку, але й підвищує стабільність та надійність програмних продуктів.

Для ефективного розв'язання задачі управління розробкою IT-проєктів слід розробити модульну структуру програмного забезпечення, що включає наступні основні модулі:

1. Модуль управління проєктами (Project Management Module):

- Створення та управління проєктами: можливість створення нових проєктів та редагування існуючих.

- Управління завданнями та підзавданнями: створення, призначення, редагування та видалення завдань і підзавдань.

- Планування етапів та спринтів: підтримка методологій Scrum та Kanban для організації робочого процесу.

2. Модуль управління командами (Team Management Module):

- Управління користувачами та ролями: створення профілів користувачів, призначення ролей (розробник, менеджер проєкту тощо).

- Комунікація та співпраця: внутрішні обговорення завдань, спільні дошки.

3. Модуль управління ресурсами (Resource Management Module):

- Управління людськими ресурсами: розподіл завдань між членами команди, відстеження їх завантаженості.

4. Модуль управління часом (Time Management Module):

- Трекінг часу: відстеження часу, витраченого на завдання.

- Календарі та розклади: планування робочих зустрічей, дедлайнів та інших подій.

5. Модуль безпеки та доступу (Security and Access Control Module):

- Управління доступом: контроль доступу до проєктів, завдань та модулів.

6. Інтерфейс користувача.

Загальна структура програмного модуля для для управління розробкою IT-проєктів представлено на рисунку 2.1.

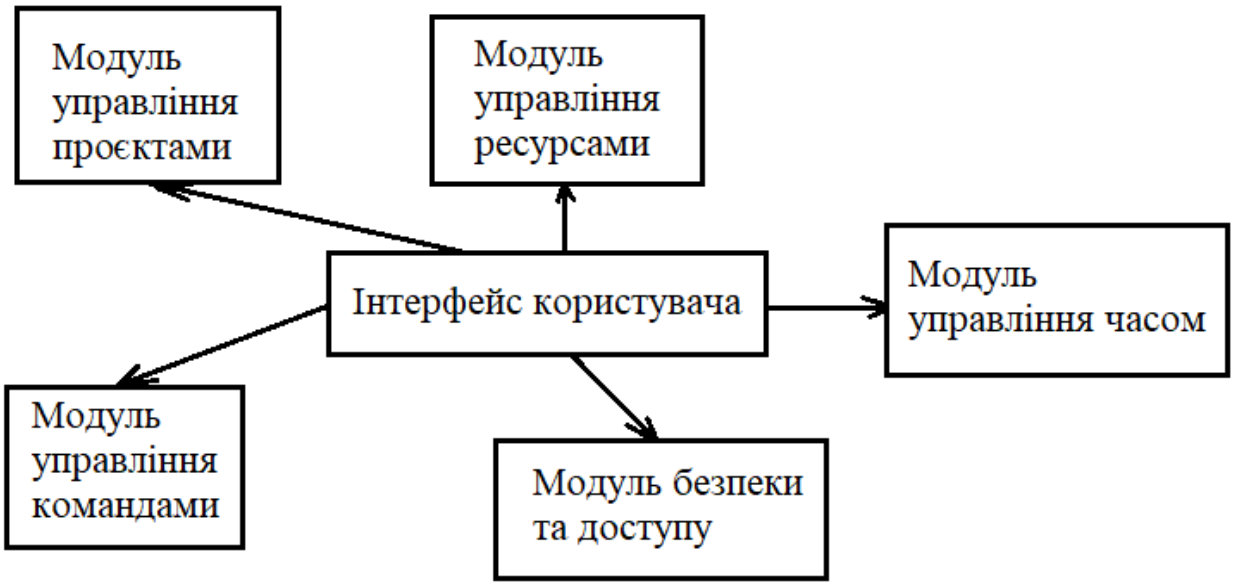

Рисунок 2.1 - Загальна структурна схема програмного модуля для управління розробкою IT-проєктів

Ці модулі забезпечують комплексне управління всіма аспектами розробки IT-проєктів, дозволяючи ефективно планувати, виконувати та контролювати свої проєкти.

Для забезпечення додаткового сповіщення для юзерів, ми будемо додатково використовувати SMTP сервер. Було розглянуто інші варіанти, такі як Firebase Cloud Messaging і Apple Push Notification Service, але цей вибір виявився найкращим через кілька ключових причин.

По-перше, SMTP (Simple Mail Transfer Protocol) є стандартним протоколом для відправлення електронної пошти в Інтернеті. Він підтримується більшістю поштових серверів і клієнтів, що забезпечує його широке використання та сумісність. Це означає, що наші користувачі можуть отримувати сповіщення на будь-яку поштову скриньку без необхідності додаткових налаштувань чи встановлення спеціального програмного забезпечення [14].

По-друге, SMTP сервери можуть бути налаштовані з високим рівнем безпеки, включаючи підтримку шифрування TLS (Transport Layer Security). Це гарантує, що сповіщення, які ми надсилаємо користувачам, будуть захищені від несанкціонованого

доступу під час передачі через Інтернет. Таким чином, ми можемо забезпечити конфіденційність і цілісність даних [14].

По-третє, використання SMTP серверів дозволяє легко інтегрувати функцію відправки сповіщень в існуючі IT-проєкти. Багато фреймворків і мов програмування мають вбудовану підтримку для роботи з SMTP, що спрощує процес розробки та впровадження. Це дозволяє швидко налаштувати і масштабувати систему сповіщень відповідно до потреб нашого проєкту [14].

Крім того, SMTP сервери забезпечують високу надійність доставки повідомлень. Вони мають вбудовані механізми для повторної спроби відправки повідомлень у разі тимчасових збоїв і можуть зберігати повідомлення в черзі до успішної доставки. Це мінімізує ризик втрати важливих сповіщень і підвищує рівень задоволеності користувачів.

Враховуючи всі ці фактори, вибір SMTP сервера для сповіщень є оптимальним рішенням для забезпечення надійної, безпечної і зручної системи сповіщення користувачів для модуля управління розробкою IT-проєктів.

Компоненти:

1. Користувач (User): ініціює процес відправки листа.
2. WEB-додаток (WEB Application): приймає запит від користувача і готує лист.
3. SMTP Сервер (SMTP Server): приймає лист від WEB-додатка і передає його одержувачу.

Послідовність дій:

1. Користувач надсилає запит на відправку листа до Web-додатка.
2. WEB-додаток створює лист (формує повідомлення).
3. WEB-додаток підключається до SMTP серверу.
4. WEB-додаток автентифікується на SMTP сервері.
5. WEB-додаток надсилає лист на SMTP сервер.
6. SMTP сервер передає лист одержувачу.
7. SMTP сервер повертає відповідь про успішне відправлення або помилку. Web-додаток повідомляє користувача про результат.

На рисунку 2.2 зображена UML діаграма послідовностей користувача із SMTP сервером.

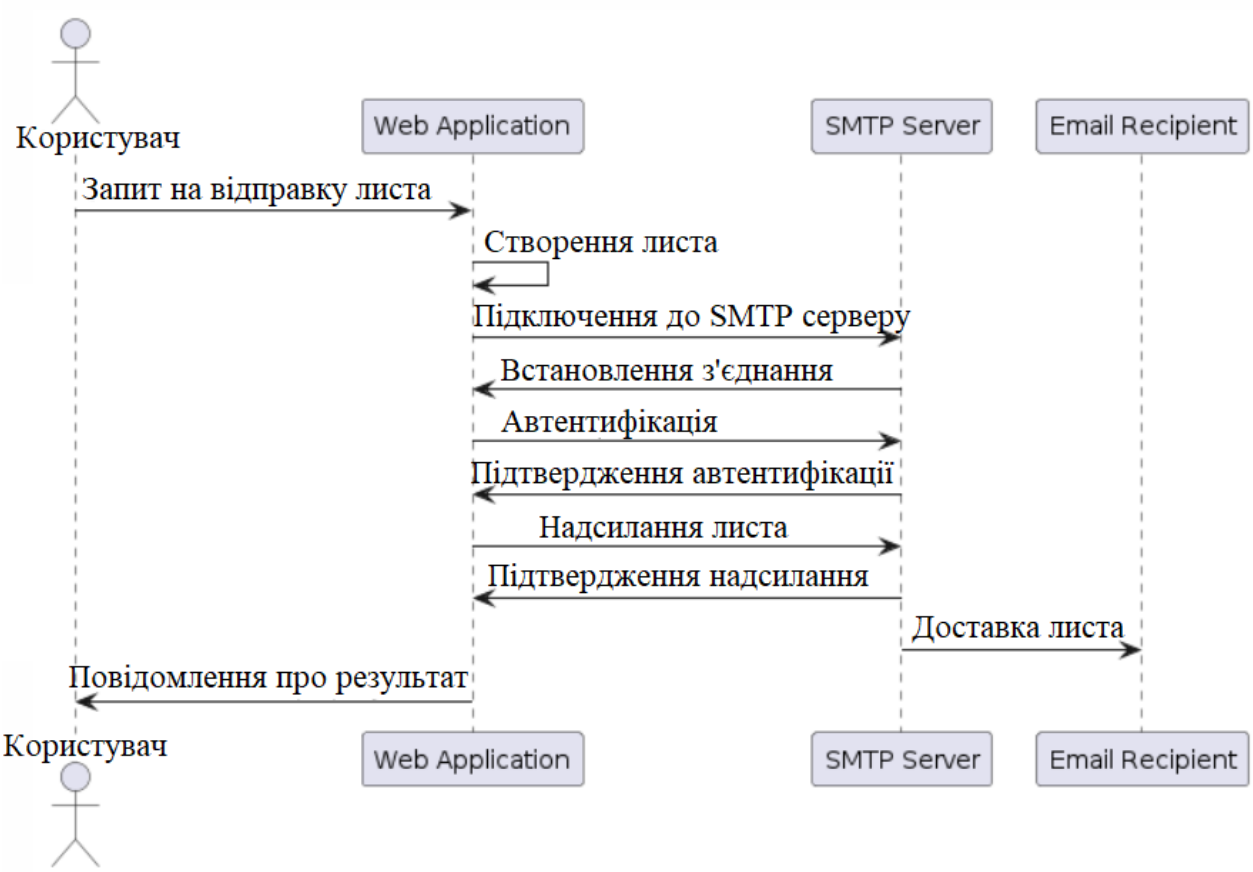

Рисунок 2.2 – Типова схема взаємодії WEB-додатка із SMTP сервером

2.2 Розробка форми взаємодії функціональних об’єктів інформаційної системи

Взаємодія з програмним забезпеченням здійснюється через веб-інтерфейс користувача. Спочатку активується модуль захисту, який вимагає введення логіна та пароля для підтвердження доступу до даних користувача в системі. Якщо користувача немає в системі потрібно зареєструватися з певною роллю, показано на рисунку 2.3, розроблено у Figma.

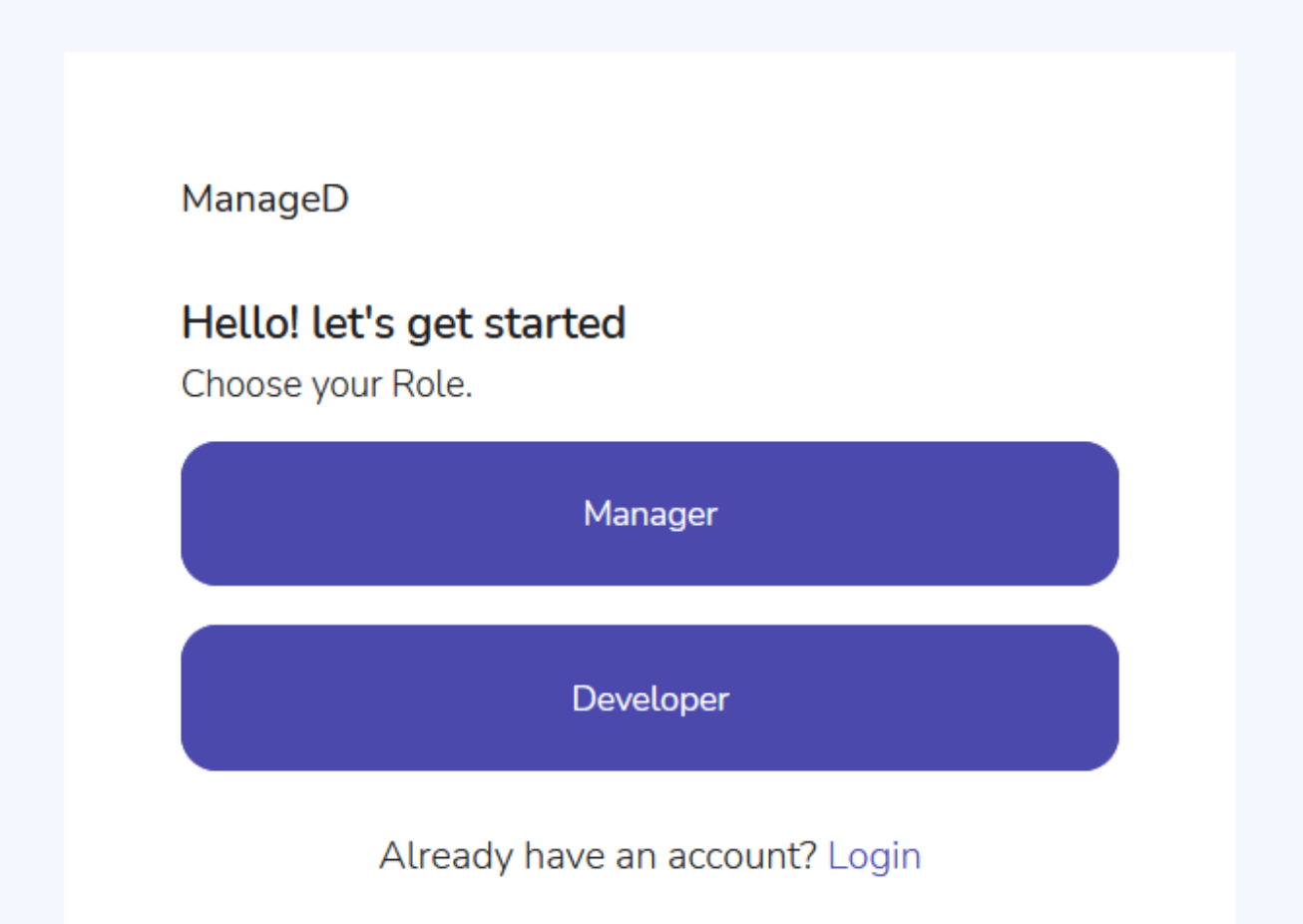

Рисунок 2.3 – Візуальна частина реєстрації під певною роллю у програмному модулі для управління розробкою IT-проєктів

Потім активується модуль користувацького інтерфейсу, який відповідає за взаємодію з користувачем та подання інформації у зручній формі, показано на рисунку 2.4, розроблено у Figma. Цей модуль включає розробку меню, кнопок, полів введення, списків та інших графічних елементів, що дозволяють користувачеві вибирати опції, вводити дані та виконувати дії. Модуль користувацького інтерфейсу забезпечує зручну та інтуїтивно зрозумілу навігацію, дозволяючи користувачеві легко взаємодіяти з іншими модулями програми.

У модулі управління проєктами який показаний на рисунку 2.5 користувач має можливість легко створювати нові проєкти в системі шляхом заповнення форми з основними даними про проєкт, такими як назва, опис, дата початку і закінчення,

бюджет і т. д. Після створення проєкту користувач може переглянути його загальну інформацію на головній панелі та перейти до детальної сторінки проєкту.

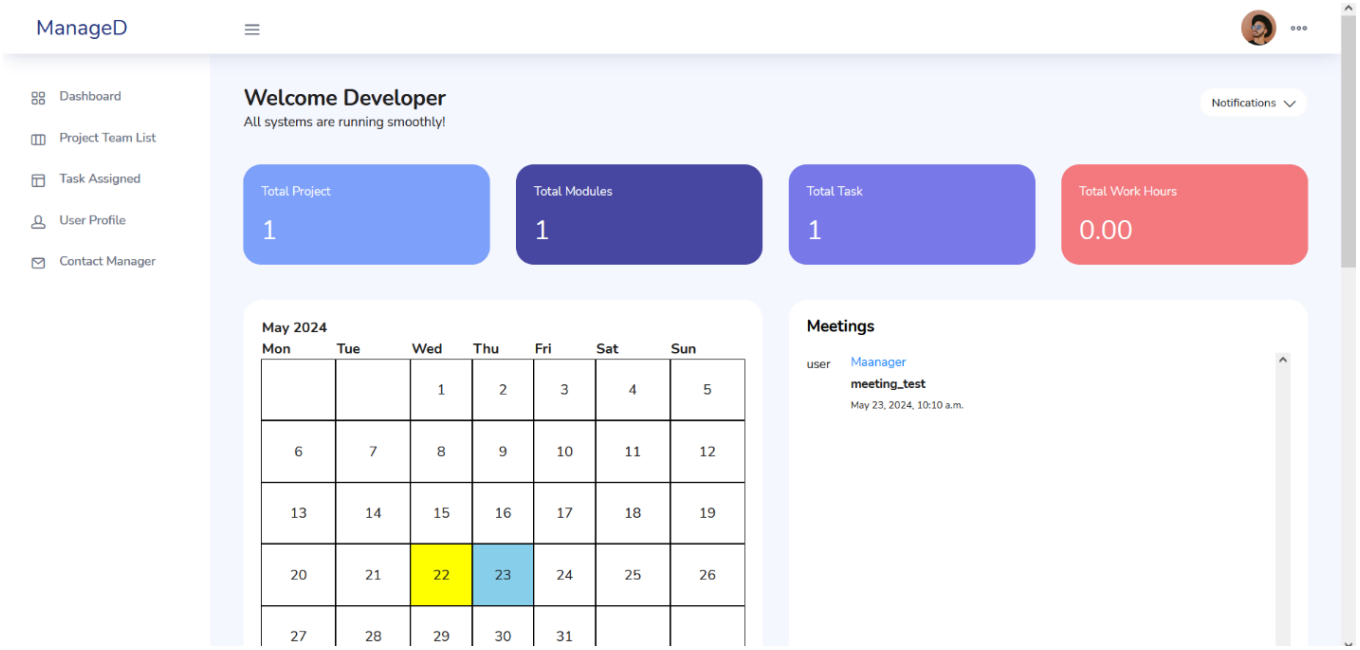

Рисунок 2.4 – Показ користувацького інтерфейсу під роллю developer у програмному модулі для управління розробкою IT-проєктів

На сторінці проєкту користувач має можливість переглядати список завдань, які потрібно виконати в рамках проєкту. Він може створювати нові завдання, надавати їм опис, встановлювати пріоритети та дедлайни. Крім того, користувач може призначати завдання іншим учасникам команди та відстежувати прогрес виконання кожного завдання.

Щодо планування етапів та спринтів, користувач може створювати нові спринти або етапи розробки, вказуючи їхню тривалість та перелік завдань, які повинні бути виконані протягом цього періоду. Це дозволяє користувачам організувати робочий процес у формі коротких ітерацій або спринтів, що полегшує відстеження прогресу і досягнення мети.

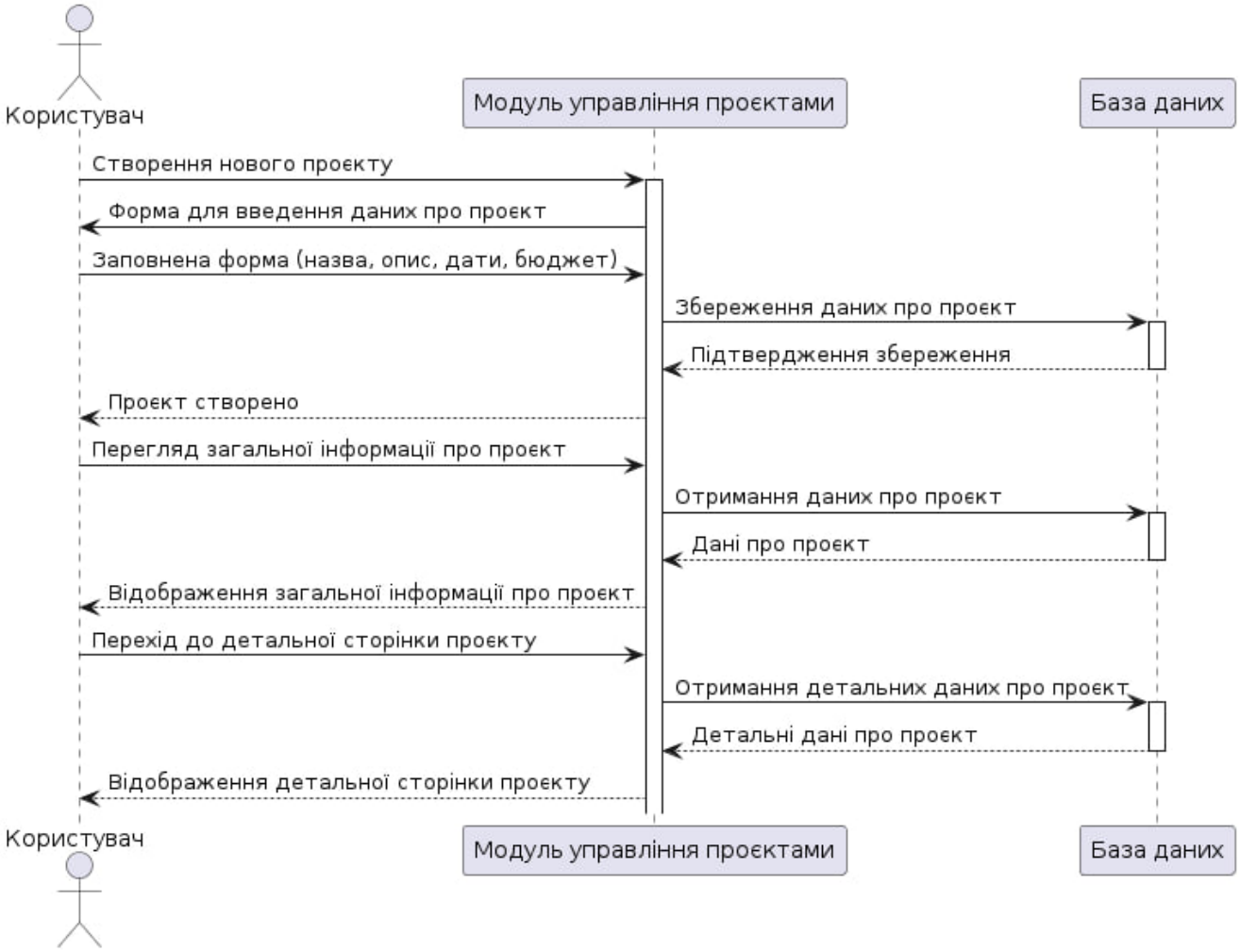

Рисунок 2.5 – Типова схема модуля управління проєктами

Модуль управління командами який показаний на рисунку 2.6 створений для полегшення співпраці та координації між учасниками проєкту. Він надає користувачам можливість додавати нових учасників з відповідними правами доступу(ролі). Крім того, користувачі можуть спілкуватися та обмінюватися інформацією. Модуль також забезпечує доступ до спільних ресурсів, таких як файли та документи, що дозволяє команді спільно працювати над проєктом.

Крім того, користувачі можуть спільно планувати робочі зустрічі та події за допомогою календаря, що допомагає забезпечити організованість та вчасність виконання завдань.

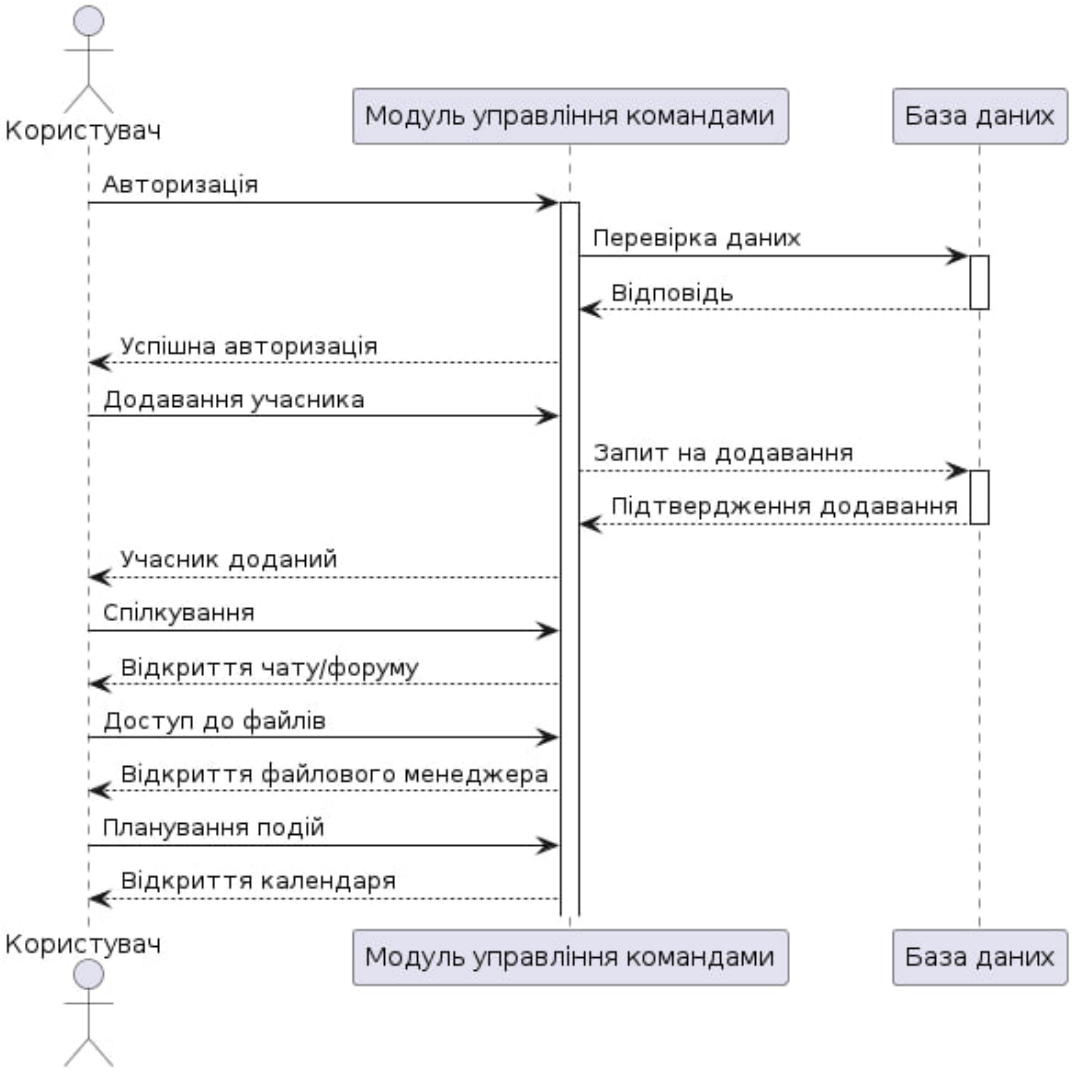

Рисунок 2.6 – Типова схема модуля управління командами

Модуль управління ресурсами який показаний на рисунку 2.7 спрощує робочий процес команди, дозволяючи користувачам ефективно розподіляти та використовувати ресурси. Користувачі можуть легко призначати завдання іншим учасникам команди та відстежувати їх прогрес. Крім того, модуль допомагає користувачам відстежувати використання ресурсів, таких як час та матеріальні ресурси, для забезпечення оптимального використання. Команда може планувати відпустки та відпочинок учасників команди, забезпечуючи безперервність робочого процесу. У цілому, модуль управління ресурсами сприяє кращій координації та ефективності робочих процесів, допомагаючи команді досягати своїх цілей.

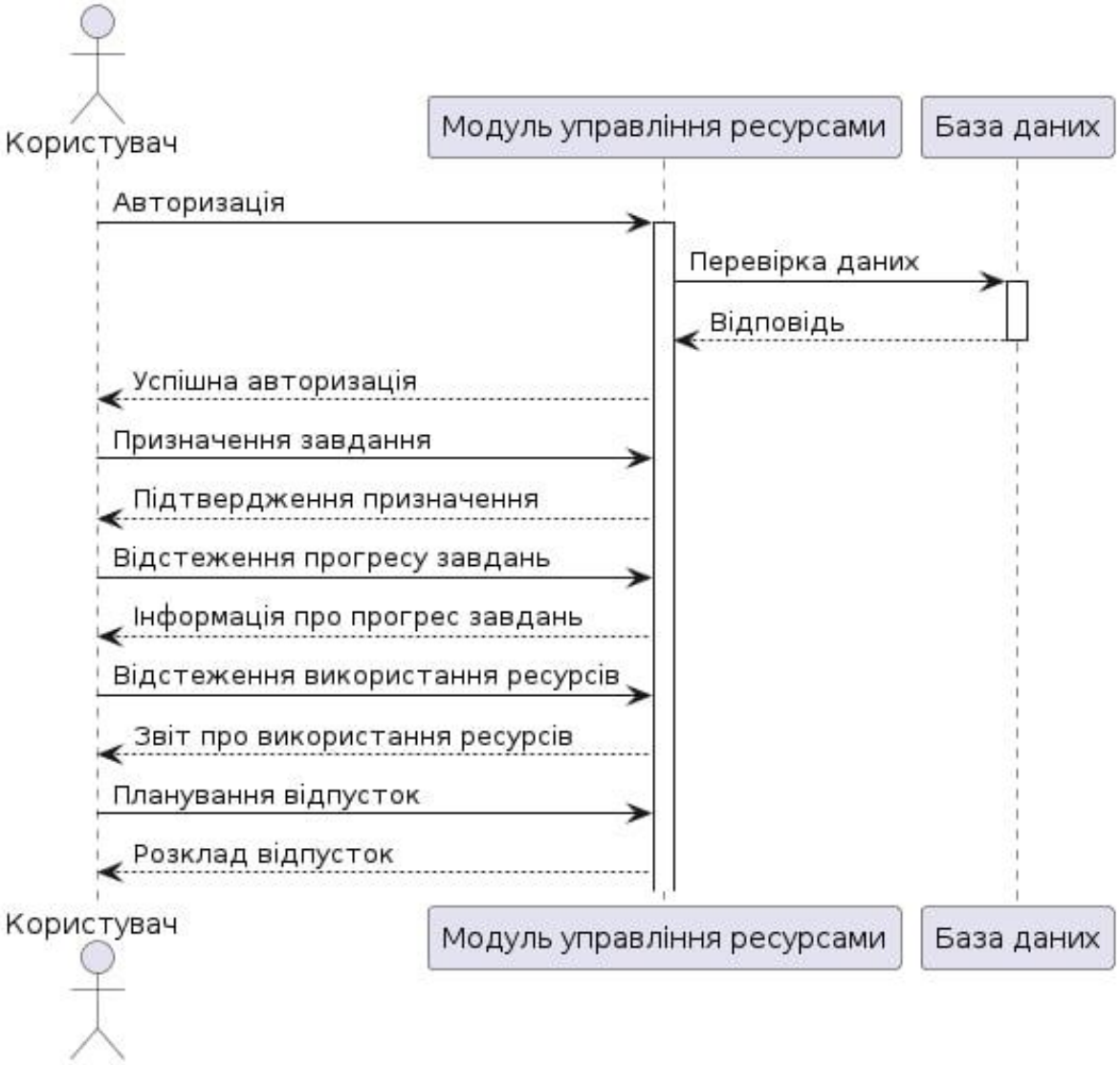

Рисунок 2.7 – Типова схема модуля управління ресурсами

Модуль управління часом який показаний на рисунку 2.8 спрощує керування часом учасників команди, забезпечуючи їм зручний інструментарій для трекінгу та планування робочих годин. Користувачі можуть легко відстежувати витрачений на завдання час, встановлювати дедлайни та керувати календарем подій. Це дозволяє їм краще організовувати свій робочий графік і вчасно виконувати завдання. Модуль також надає можливість аналізувати та аудитувати використання часу, допомагаючи користувачам виявляти ефективність своєї робочої діяльності та робити відповідні корективи.

Рисунок 2.8 – Типова схема модуля управління часом

2.3 Розробка алгоритму функціонування програмного модулю для управління розробкою IT-проєктів

Розроблений алгоритм функціонування програмного модуля для управління розробкою IT-проєктів, який показаний на рисунку 2.9.

Розроблений алгоритм для управління розробкою IT-проєктів може бути розподілений на наступні кроки:

1. Початок:

- Алгоритм розпочинається з виконання.

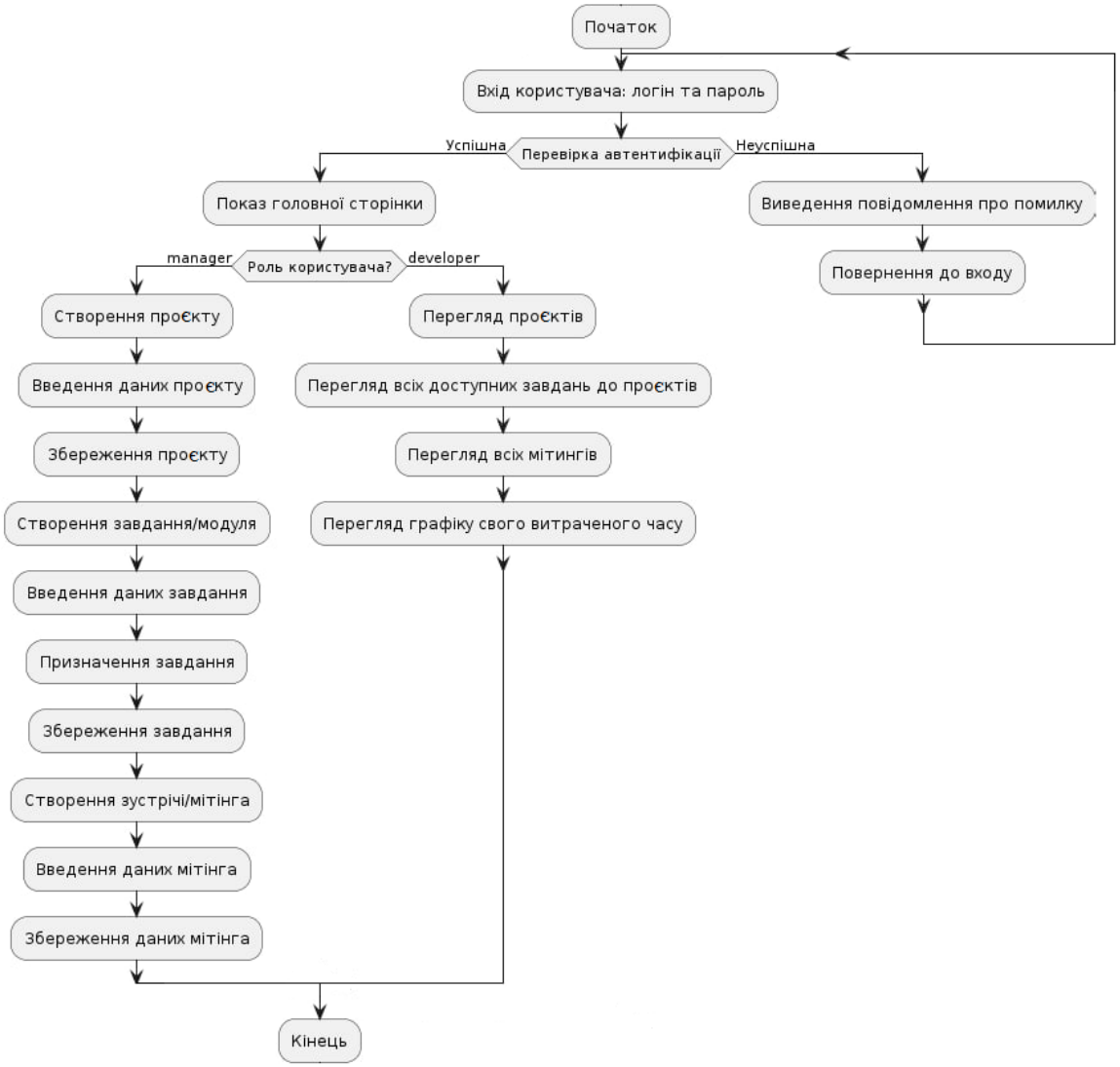

Рисунок 2.9 – Схема алгоритму функціонування програмного модуля для управління розробкою IT-проєктів

2. Вхід користувача: логін та пароль:

- Користувач вводить свої облікові дані (логін та пароль) для доступу до системи. На сторінці входу відображається форма з якою може взаємодіяти користувач.

3. Перевірка автентифікації:

- Система перевіряє автентичність введених даних.

- Якщо автентифікація успішна, користувач переходить до наступного кроку і система надсилає лист на пошту.

- Якщо автентифікація неуспішна, система виводить повідомлення про помилку, і користувач повертається до кроку введення логіну та паролю.

4. Показ головної сторінки:

- Після успішної автентифікації користувач бачить головну сторінку системи, бачить різні графіки на інформацію від інших користувачів в залежності від ролі.

5. Роль користувача:

- Система визначає роль користувача: менеджер (manager) або розробник (developer).

Для менеджера:

6. Створення проєкту:

- Менеджер має можливість створити новий проєкт.

7. Введення даних проєкту:

- Менеджер вводить необхідні дані для створення проєкту. Користувач має можливість ввести назву, опис, статус проєкту, технології які використовуються, скільки потрібно часу, початок виконання та закінчення проєкту і також прикріпити файли.

8. Збереження проєкту:

- Система зберігає введені дані про проєкт в базу даних.

9. Створення завдання/модуля:

- Менеджер може створити завдання або модуль для проєкту.

10. Введення даних завдання:

- Менеджер вводить дані для нового завдання. Користувач має можливість ввести назву, опис, статус завдання, пріорітет, скільки потрібно часу, початок виконання та закінчення.

11. Призначення завдання:

- Менеджер призначає завдання відповідним членам команди.

12. Збереження завдання:

- Система зберігає введені дані про завдання в базу даних.

13. Створення зустрічі/мітінга:

- Менеджер може створити зустріч або мітінг.

14. Введення даних мітінга:

- Менеджер вводить дані про зустріч. Користувач має можливість ввести назву, опис мітінга, прикріпити файли, вибрати певних користувач для кого цей мітінг і залишити дату.

15. Збереження даних мітінга:

- Система зберігає введені дані про мітінг в базу даних.

Для розробника:

16. Перегляд проєктів:

- Розробник може переглядати список доступних проєктів, які зробив користувач з роллю менеджера і призначив для нього.

17. Перегляд всіх доступних завдань до проєктів:

- Розробник переглядає всі завдання, призначені на нього, або доступні для виконання, які зробив користувач з роллю менеджера і призначив для нього.

18. Перегляд всіх мітінгів:

- Розробник може переглядати всі мітінги, в яких він бере участь.

19. Перегляд графіку свого витраченого часу:

- Розробник може переглядати графік часу, витраченого на виконання завдань.

Користувач взаємодіє з системою управління IT-проєктами, починаючи з автентифікації і закінчуючи управлінням проєктами, завданнями, мітінгами та відстеженням часу. Різні ролі користувачів (менеджер і розробник) мають різні можливості та функції в системі, що дозволяє ефективно організовувати та контролювати процес розробки проєктів.

Розроблений алгоритм реєстрації користувача у модулі для управління IT-проєктів, який показаний на рисуноку 2.10.

Пояснення блок-схеми:

1. Користувач вибирає роль:

- Перший крок полягає у виборі ролі, за яку користувач хоче реєструватися (менеджер або девелопер).

Рисунок 2.10 – Схема алгоритму реєстрації користувача у модулі для управління IT-проєктів

2. Введення даних:

- Користувач вводить необхідні дані, такі як юзернейм, email, стать, дата народження, зарплата і пароль.

3. Перевірка даних:

- Система перевіряє, чи всі введені дані коректні.

4. Реєстрація користувача:

- Якщо дані введені коректно, користувач зареєстрований в системі, і відправляється вітальний лист на електронну пошту, а користувач отримує повідомлення про успішну реєстрацію.

- Якщо дані введені некоректно, користувач отримує повідомлення про необхідність введення коректних даних і повертається до введення даних.

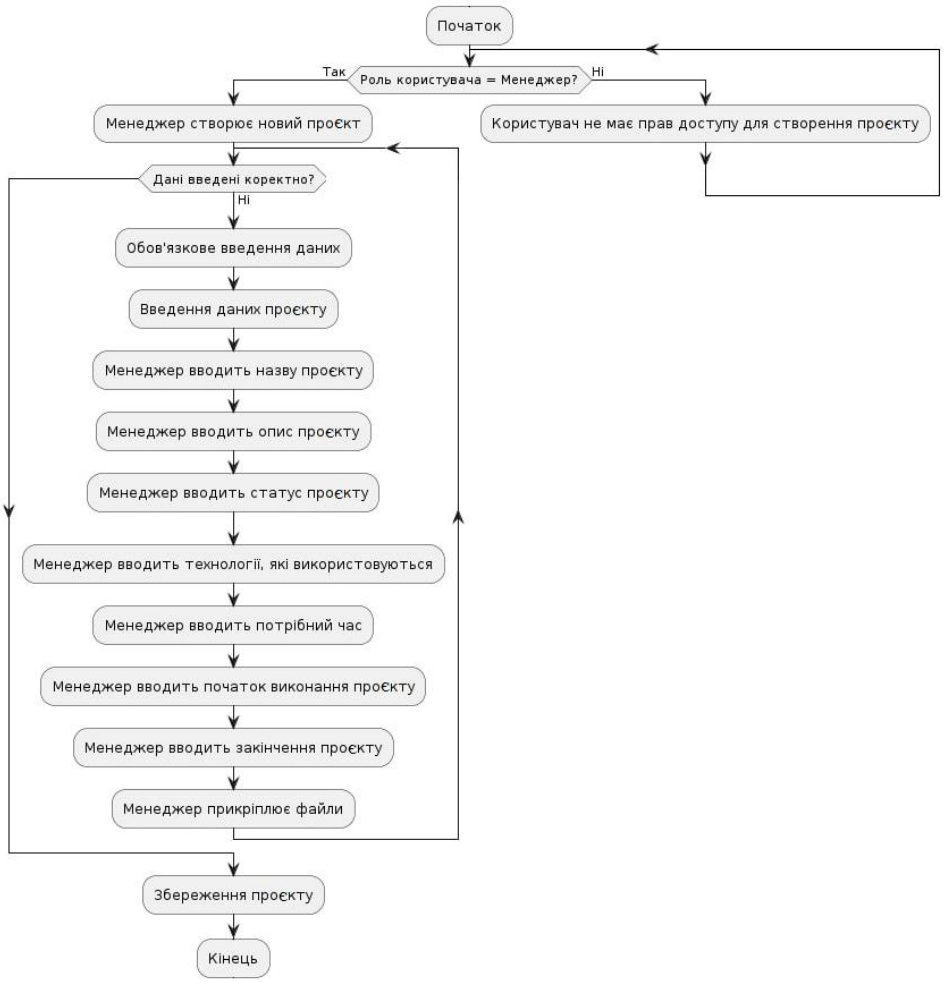

Рисунок 2.11 – Схема алгоритму створення проєкта у модулі для управління розробкою ІТ-проєктів

На рисунку 2.11 показано алгоритм створення проєкта у модулі для управління розробкою ІТ-проєктів.

Опис блок-схеми процесу створення нового проєкту менеджером:

Початковий етап:

1. Перевірка ролі користувача:

- Блок-схема починається з перевірки ролі користувача. Якщо користувач є менеджером, то процес продовжується.

- Якщо користувач не є менеджером, з'являється повідомлення "Користувач не має прав доступу для створення проєкту" і процес завершується.

Створення проєкту:

2. Менеджер створює новий проєкт:

- Якщо користувач є менеджером, він може ініціювати створення нового проєкту.

Введення даних проєкту:

3. Перевірка введених даних:

- Система перевіряє, чи всі необхідні дані введені коректно.

- Якщо дані введені некоректно, користувачеві потрібно обов'язково ввести дані заново.

4. Обов’язкове введення даних:

- Якщо дані не введені коректно, система повертається до введення даних проєкту.

5. Введення даних проєкту:

- Менеджер вводить необхідні дані для створення проєкту, такі як:

- Назва проєкту: Введення назви проєкту.

- Опис проєкту: Введення опису проєкту.

- Статус проєкту: Введення статусу проєкту (наприклад, новий, в процесі, завершений).

- Технології, які використовуються: Введення списку технологій, які будуть використовуватися в проєкті.

- Потрібний час: Введення потрібного часу для виконання проєкту.

- Початок виконання проєкту: Введення дати початку проєкту.

- Закінчення проєкту: Введення дати завершення проєкту.

- Прикріплення файлів: Прикріплення файлів, що стосуються проєкту.

6. Збереження проєкту:

- Після введення всіх необхідних даних і їх перевірки на коректність, проєкт зберігається в системі.

7. Завершення процесу:

- Процес завершується після успішного збереження проєкту.

Розроблений алгоритм створення завдання у модулі для управління розробкою ІТ-проєктів, який показаний на рисунку 2.12.

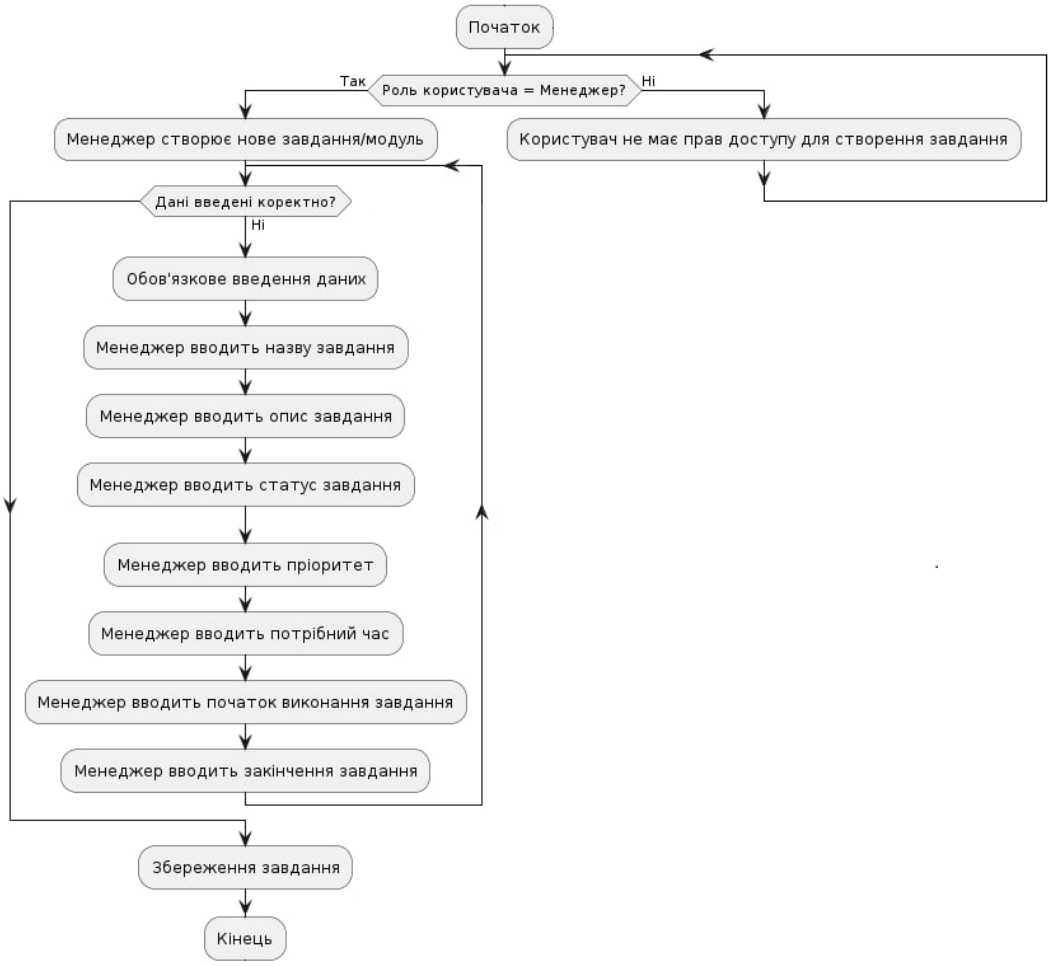

Рисунок 2.12 – Схема алгоритму створення завдання у модулі для управління розробкою IT-проєктів

Опис блок-схеми:

1. Початок: Процес починається з перевірки ролі користувача.

2. Роль користувача = Менеджер?: Якщо користувач є менеджером, процес продовжується.

- Якщо користувач не є менеджером, процес завершується з повідомленням "Користувач не має прав доступу для створення завдання".

3. Менеджер створює нове завдання/модуль: Менеджер ініціює створення нового завдання або модуля.

4. Введення даних завдання: Менеджер вводить всі необхідні дані для створення завдання. Це включає введення назви, опису, статусу, пріоритету, потрібного часу, дати початку та закінчення завдання.

5. Дані введені коректно?: Система перевіряє, чи всі дані введені коректно.

- Якщо дані не введені коректно, менеджеру потрібно обов'язково ввести дані заново.

- Якщо дані введені коректно, процес продовжується.

6. Збереження завдання: Завдання зберігається в системі.

7. Завершення процесу: Процес завершено.

Ця блок-схема описує всі основні кроки процесу створення нового завдання або модуля менеджером, забезпечуючи коректність і повноту введених даних перед збереженням завдання в системі.

2.4 Висновок розділу 2

У даному розділі було детально розглянуто архітектуру програмного модуля для управління розробкою IT-проєктів. Основна увага приділялась створенню та аналізу структури програмного модуля, що включає кілька ключових компонентів, таких як модулі управління проєктами, командами, ресурсами та часом.

Побудовані UML-діаграми взаємодії функціональних компонентів програмного модуля дозволили наочно відобразити процеси, що відбуваються в системі. Зокрема, було детально проаналізовано сценарії створення, редагування та управління проєктами і завданнями, а також управління командами та ресурсами.

Розроблений алгоритм функціонування програмного модуля для управління розробкою IT-проєктів охоплює всі основні етапи взаємодії користувача із системою, починаючи від автентифікації та закінчуючи управлінням завданнями та мітингами. Було проаналізовано логіку роботи системи для різних ролей користувачів, таких як менеджери та розробники, що дозволяє забезпечити ефективний та гнучкий підхід до управління проєктами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ УПРАЛІННЯ РОЗРОБКОЮ ІТ-ПРОЄКТІВ

3.1 Обґрунтування вибору мови програмування

При розробці програмного модуля для управління розробкою ІТ-проєктів, важливо ретельно обрати мову програмування, яка забезпечить оптимальну продуктивність, масштабованість, зручність розробки та підтримки. У цьому розділі ми розглянемо кілька популярних мов програмування, порівняємо їхні переваги та недоліки, а також обґрунтуємо вибір конкретної мови для даного проєкту.

Переваги мови програмування Python [15]:

1. Легкість у вивченні та використанні: Python має простий і зрозумілий синтаксис, що робить його ідеальним для швидкої розробки та прототипування.

2. Велика стандартна бібліотека: Python надає багатий набір модулів та пакетів для різних завдань, включаючи роботу з базами даних, мережевими протоколами та веб-розробку.

3. Популярність та спільнота: Велика спільнота розробників та безліч доступних ресурсів, документації та прикладів коду.

4. Фреймворки для веб-розробки: Django та Flask є популярними фреймворками, які дозволяють швидко створювати потужні веб-додатки.

Недоліки мови програмування Python:

1. Продуктивність: Python є інтерпретованою мовою, тому може бути повільнішим у виконанні порівняно з компільованими мовами, такими як C++ або Java.

2. Обмежена багатопоточність: Через Global Interpreter Lock (GIL) Python має обмеження у виконанні багатопотокових задач.

Переваги мови програмування Java [16]:

1. Продуктивність: Java є компільованою мовою, що забезпечує високу продуктивність виконання.

2. Масштабованість: Java широко використовується у великих корпоративних додатках, що потребують високої надійності та масштабованості.

3. Портативність: Java програми можуть працювати на будь-якій платформі, що підтримує JVM (Java Virtual Machine).

Недоліки мови програмування Java:

1. Складність: Java має більш складний синтаксис і круту криву навчання порівняно з Python або JavaScript.

2. Великий обсяг коду: Java вимагає більше коду для реалізації тих самих функцій, що і в Python або JavaScript.

Після ретельного аналізу двох провідних фронтенд технологій, React і Angular, ми дійшли до висновку, що React є найбільш підходящим вибором для нашого проєкту. React відомий своєю легкістю у вивченні, простим синтаксисом та високою продуктивністю завдяки віртуальному DOM. Він також забезпечує компонентний підхід, що сприяє повторному використанню коду та модульності. Гнучкість React дозволяє інтегрувати його з різними бібліотеками та інструментами, що робить його ідеальним для проєктів, які вимагають швидкого прототипування та легкого масштабування. Крім того, велика спільнота розробників та багата екосистема додаткових бібліотек, таких як Redux і React Router, полегшують розробку та підтримку проєкту [17-18].

Зважаючи на особливості проєкту управління розробкою IT-проєктів, а також враховуючи переваги та недоліки різних мов програмування, найбільш оптимальним вибором є Python.

При виборі бази даних для програмного модуля управління розробкою IT-проєктів ми зважили на кілька ключових факторів, включаючи надійність, продуктивність, масштабованість, підтримку фіч. Після ретельного аналізу обрано PostgreSQL як базу даних для цього проєкту. Ось детальне обґрунтування цього вибору [19].

Переваги PostgreSQL:

1. Надійність і стабільність:

- PostgreSQL відома своєю надійністю та стабільністю. Вона використовується в багатьох великих організаціях та має хорошу репутацію як безпечна і стабільна база даних.

2. Підтримка розширених фіч:

- PostgreSQL пропонує широкий набір функціональних можливостей, включаючи підтримку складних запитів, транзакцій, збережених процедур, тригерів та розширень. Це робить її ідеальною для складних додатків з великою кількістю бізнес-логіки.

3. Сумісність з Django:

- Django, обраний нами фреймворк для бекенду, має відмінну підтримку PostgreSQL. Django ORM (Object-Relational Mapping) забезпечує простий і зручний спосіб взаємодії з базою даних, що полегшує розробку і підтримку.

4. Масштабованість і продуктивність:

- PostgreSQL може ефективно працювати з великими обсягами даних і високими навантаженнями. Вона підтримує різні методи масштабування, включаючи горизонтальне та вертикальне масштабування, що дозволяє адаптуватися до зростаючих вимог проєкту.

5. Розширюваність:

- PostgreSQL має модульну архітектуру, що дозволяє додавати нові функції через розширення. Це забезпечує гнучкість і можливість адаптації бази даних до специфічних потреб проєкту.

6. Активна спільнота та підтримка:

- PostgreSQL має велику та активну спільноту розробників, що забезпечує регулярні оновлення, виправлення помилок та підтримку нових функцій. Це гарантує, що база даних буде актуальною і безпечною протягом тривалого часу.

Стек технологій для програмного модуля управління IT-проєктами:

- Backend: Для розробки серверної частини додатку обрано використовувати Python Django.

- Database: Як базу даних обрано PostgreSQL. Це потужна, надійна та масштабована реляційна база даних, яка добре інтегрується з Django.

- Frontend: Для створення інтерактивного інтерфейсу користувача обрано React.js. React.js - це бібліотека JavaScript, що дозволяє створювати динамічні та чуйні інтерфейси користувача.

- API: Для забезпечення взаємодії між фронтендом та бекендом ми будемо використовувати Django REST Framework.

3.2 Програмна реалізація системного модуля

Загалом було розроблено ряд методів для підтримки функціоналу модуля системи управління IT-проєктами, такі як Projects, Teams, Resources, Time Management, що відповідають за створення та управління проєктами, командами, ресурсами, а також облік часу, витраченого на завдання.

Пропоную більш детально розглянути модуль управління проєктами, який був розроблений саме для ефективного управління проєктами в IT-середовищі.

Метод «create_project» відповідає за створення нового проєкту:

```
from django.shortcuts import get_object_or_404
from django.contrib.auth.models import User
from .models import Project
from django.http import JsonResponse
def create_project(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        description = request.POST.get('description')
        owner_id = request.POST.get('owner_id')
        owner = get_object_or_404(User, id=owner_id)

        project = Project.objects.create(
            name=name,
            description=description,
            owner=owner
```

```
        )
        return JsonResponse({'id': project.id, 'name': project.name})
    return JsonResponse({'error': 'Invalid request'}, status=400).
```

Цей метод обробляє запити на створення нового проєкту. Він перевіряє, чи метод запиту є POST, отримує дані з запиту, створює новий об'єкт «Project» та зберігає його в базі даних. Після успішного створення проєкту повертається JSON-відповідь з деталями створеного проєкту.

Метод «get_projects» відповідає за представлення списку проєктів:

```
from django.http import JsonResponse
from .models import Project

def get_projects(request):
    projects = Project.objects.all().values('id', 'name', 'description', 'owner__username')
    return JsonResponse(list(projects), safe=False).
```

Цей метод отримує всі проєкти з бази даних і повертає їх у форматі JSON для відображення у вигляді списку.

Пропоную більш детально розглянути модуль управління командами.

Метод «add_user_to_team» відповідає за додавання користувача до команди:

```
from django.shortcuts import get_object_or_404
from django.contrib.auth.models import User
from .models import Team
from django.http import JsonResponse

def add_user_to_team(request, team_id):
    if request.method == 'POST':
        user_id = request.POST.get('user_id')
        user = get_object_or_404(User, id=user_id)
        team = get_object_or_404(Team, id=team_id)
        team.members.add(user)
        return JsonResponse({'message': 'User added to team successfully'})
```

```
    return JsonResponse({'error': 'Invalid request'}, status=400).
```

Цей метод обробляє запити на додавання користувача до команди. Він перевіряє метод запиту, отримує користувача та команду за їх ID, додає користувача до членів команди і повертає JSON-відповідь з повідомленням про успішне додавання.

Пропоную більш детально розглянути модуль управління ресурсами.

Метод «create_resource» відповідає за створення нового ресурсу:

```
from django.shortcuts import get_object_or_404
from .models import Resource, Project
from django.http import JsonResponse

def create_resource(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        type = request.POST.get('type')
        quantity = request.POST.get('quantity')
        project_id = request.POST.get('project_id')
        project = get_object_or_404(Project, id=project_id)

        resource = Resource.objects.create(
            name=name,
            type=type,
            quantity=quantity,
            project=project
        )
        return JsonResponse({'id': resource.id, 'name': resource.name})
    return JsonResponse({'error': 'Invalid request'}, status=400).
```

Цей метод обробляє запити на створення нового ресурсу. Він перевіряє метод запиту, отримує дані з запиту, створює новий об'єкт «Resource» та зберігає його в базі даних, а потім повертає JSON-відповідь з деталями створеного ресурсу.

Пропоную більш детально розглянути код реалізації нового завдання.

Метод «create_task» відповідає за створення нового завдання:

```
from django.shortcuts import get_object_or_404
from .models import Task, Project
from django.http import JsonResponse

def create_task(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        description = request.POST.get('description')
        due_date = request.POST.get('due_date')
        project_id = request.POST.get('project_id')
        project = get_object_or_404(Project, id=project_id)

        task = Task.objects.create(
            name=name,
            description=description,
            due_date=due_date,
            project=project
        )
        return JsonResponse({'id': task.id, 'name': task.name})
    return JsonResponse({'error': 'Invalid request'}, status=400).
```

Цей метод обробляє запити на створення нового завдання. Він отримує дані з POST-запиту, створює новий об'єкт «Task» та зберігає його в базі даних. Після успішного створення завдання повертається JSON-відповідь з деталями завдання.

Пропоную більш детально розглянути модуль управління часом.

Метод «log_time» відповідає за реєстрацію часу, витраченого на завдання:

```
from django.shortcuts import get_object_or_404
from .models import TimeEntry, Task
from django.http import JsonResponse
```

```
def log_time(request):
    if request.method == 'POST':
        task_id = request.POST.get('task_id')
        user_id = request.POST.get('user_id')
        start_time = request.POST.get('start_time')
        end_time = request.POST.get('end_time')

        task = get_object_or_404(Task, id=task_id)
        user = get_object_or_404(User, id=user_id)

        time_entry = TimeEntry.objects.create(
            task=task,
            user=user,
            start_time=start_time,
            end_time=end_time
        )
        return JsonResponse({'id': time_entry.id, 'task': task.name, 'user': user.username})
    return JsonResponse({'error': 'Invalid request'}, status=400).
```

Цей метод обробляє запити на реєстрацію часу, витраченого на завдання. Він отримує дані з POST-запиту, створює новий об'єкт «TimeEntry» та зберігає його в базі даних, а потім повертає JSON-відповідь з деталями запису часу.

3.3 Тестування програмного модуля для управління IT-проєктами

При тестуванні програмного модуля управління IT-проєктами необхідно було перевірити правильність введення, збереження та відображення даних, а також взаємодію з іншими компонентами системи. Зокрема, варто перевірити коректність

роботи сторінок авторизації, редагування проєктної інформації, а також роботу кожного з блоків системи.

Тестування почнемо з того, що відкриємо редактор коду Visual Studio Code та запустимо програму за допомогою команди «python manage.py runserver». Далі нам потрібно запустити сервер PostgreSQL, щоб програма мала доступ до нашої бази даних. Відкривши посилання (localhost) http://127.0.0.1:8000/, бачимо вікно авторизації, як показано на рисунку 3.1.

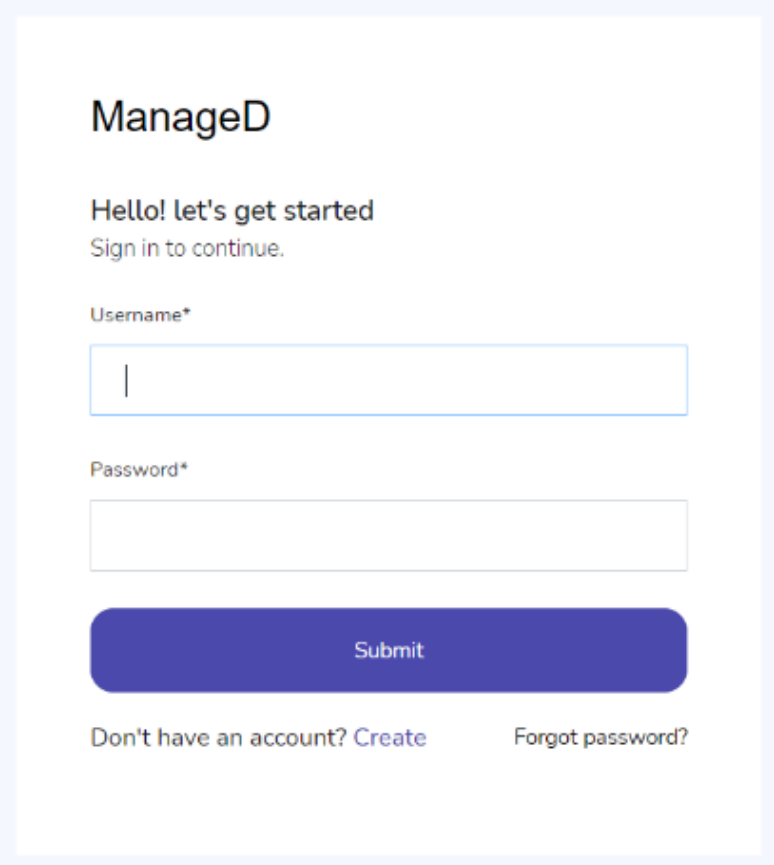

Рисунок 3.1 – Загальний вигляд вікна авторизації

Після введення логіну та паролю ми бачимо вже повноцінну систему, яку можна розглянути більш детально на рисунку 3.2.

На рисунку 3.3 відображається візуальне представлення списку проєктів, які ми можемо редагувати на свій смак, додаючи, видаляючи та оновлюючи інформацію про кожен окремий проєкт.

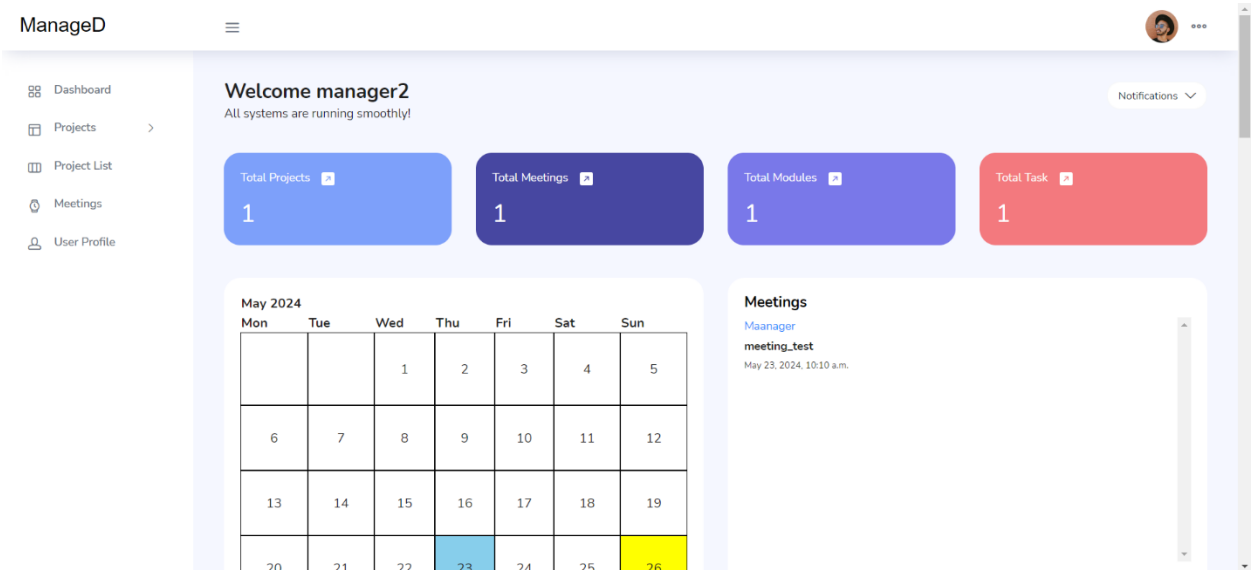

Рисунок 3.2 – Загальний вигляд системи ManageD для управління розробкою IT-проєктів

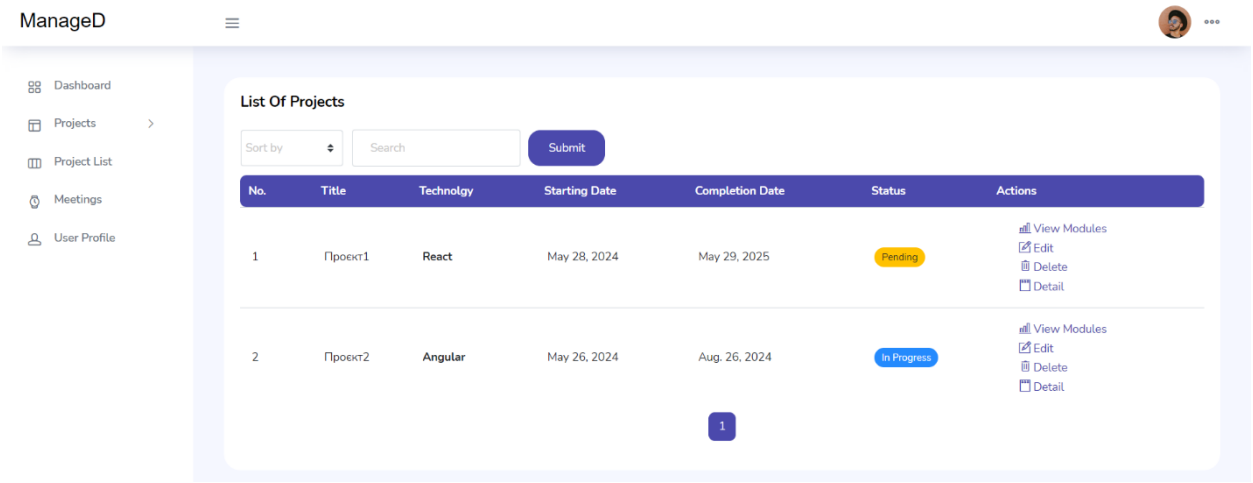

Рисунок 3.3 – Загальний вигляд інтерфейсного вікна зі списком проєктів

На рисунку 3.4 показано візуальне представлення створення команди, якою можна управляти, додаючи нових учасників та відразу призначити їй проєкт.

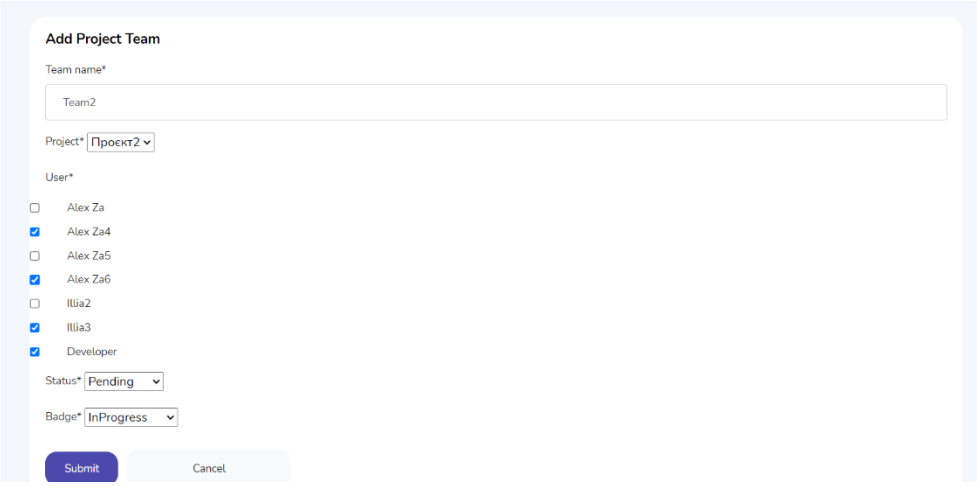

Рисунок 3.4 – Загальний вигляд вікна із створення команди

На рисунку 3.5 відображається вікно для створення проєкту, відразу ми можемо прикріпити потрібні нам файли, поставити дедлайни та запланувати затрати на час.

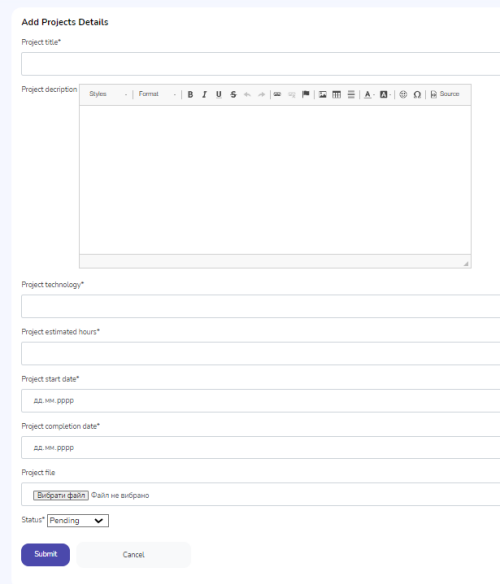

Рисунок 3.5 – Загальний вигляд вікна створення проєкта

На рисунку 3.6 відображається вікно для створення модуля задач для проєкту.

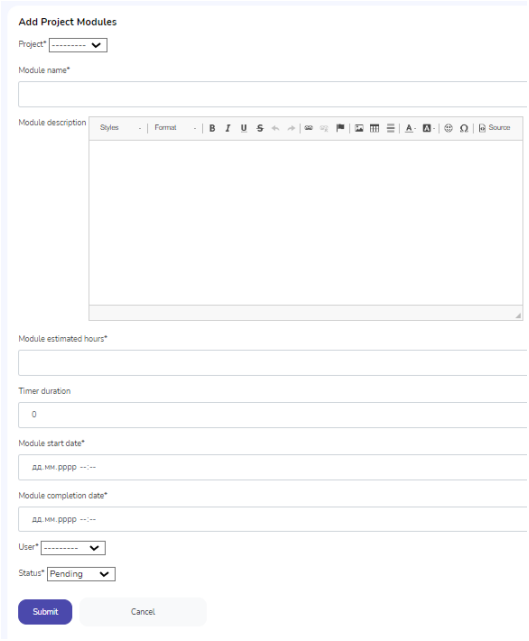

Рисунок 3.6 – Загальний вигляд вікна створення модуля

На рисунку 3.7 відображається вікно для створення задачі для модуля.

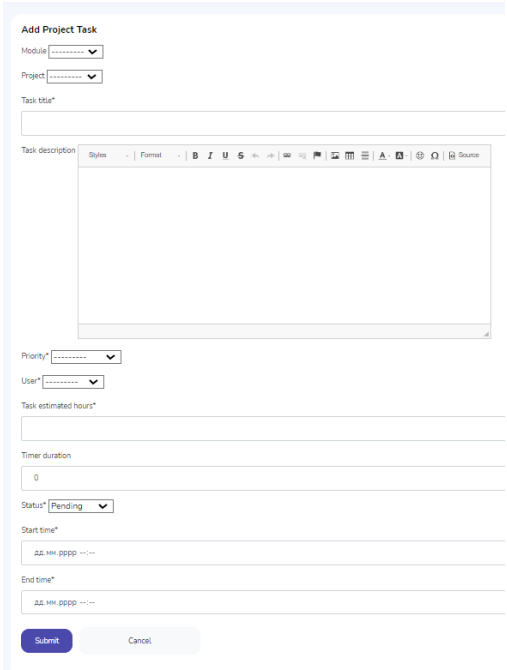

Рисунок 3.7 – Загальний вигляд вікна створення задачі

На рисунку 3.8 відображається візуальне представлення списку мітінгів, які ми можемо редагувати, додаючи, видаляючи та оновлюючи інформацію про кожен окремий мітінг.

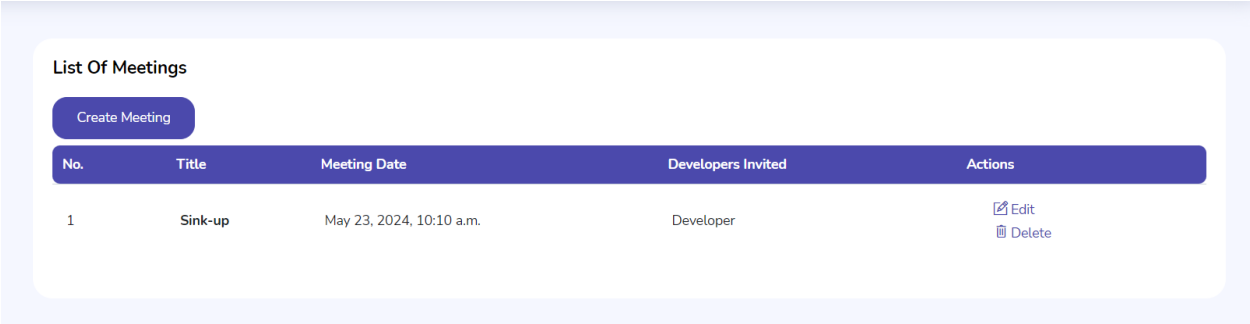

Рисунок 3.8 – Загальний вигляд вікна зі списком мітінгів

На рисунку 3.9 відображається візуальне загальне представлення інформації профіля, які ми можемо редагувати, додаючи, видаляючи та оновлюючи інформацію.

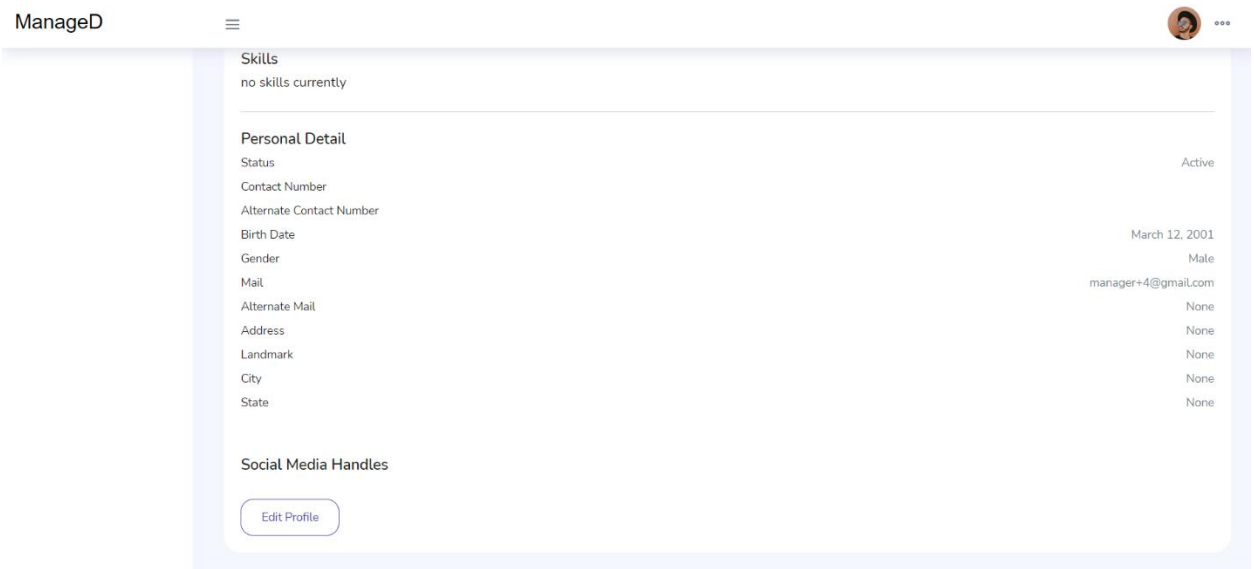

Рисунок 3.9 – Загальний вигляд вікна для редагування особистої інформації

На рисунку 3.10 відображається візуальне представлення всіх модулів та задач, які можуть бути в трьох статусах: очікується, в прогресі і виконано.

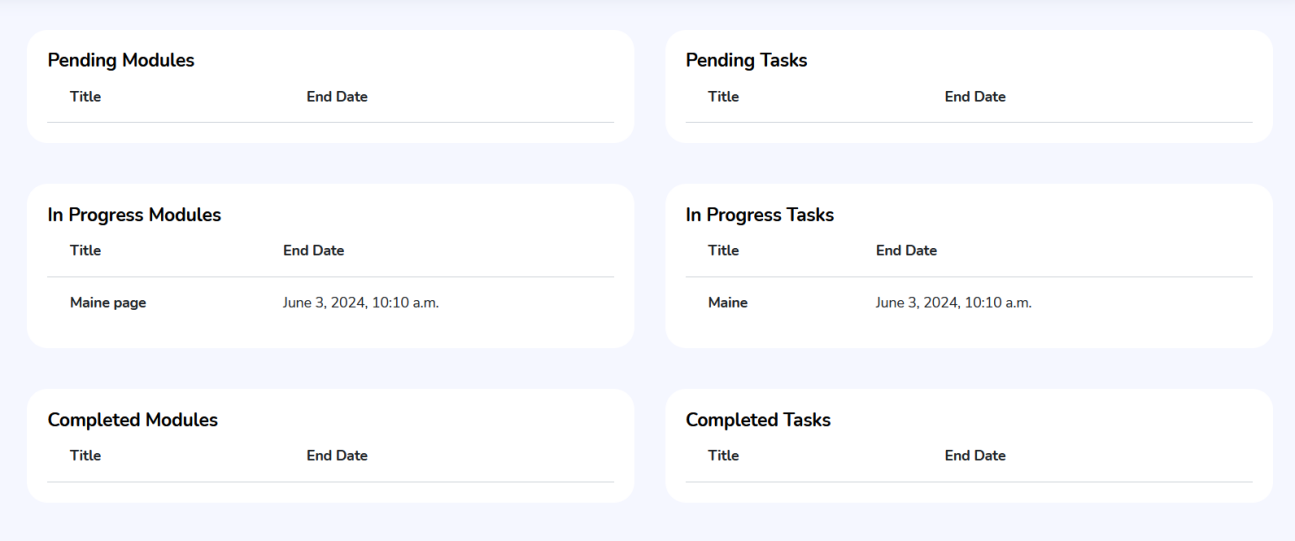

Рисунок 3.10 – Загальний вигляд вікна усіх модулів і задач

На рисунку 3.11 відображається аналітичне представлення витрачення часу і виконаних задач.

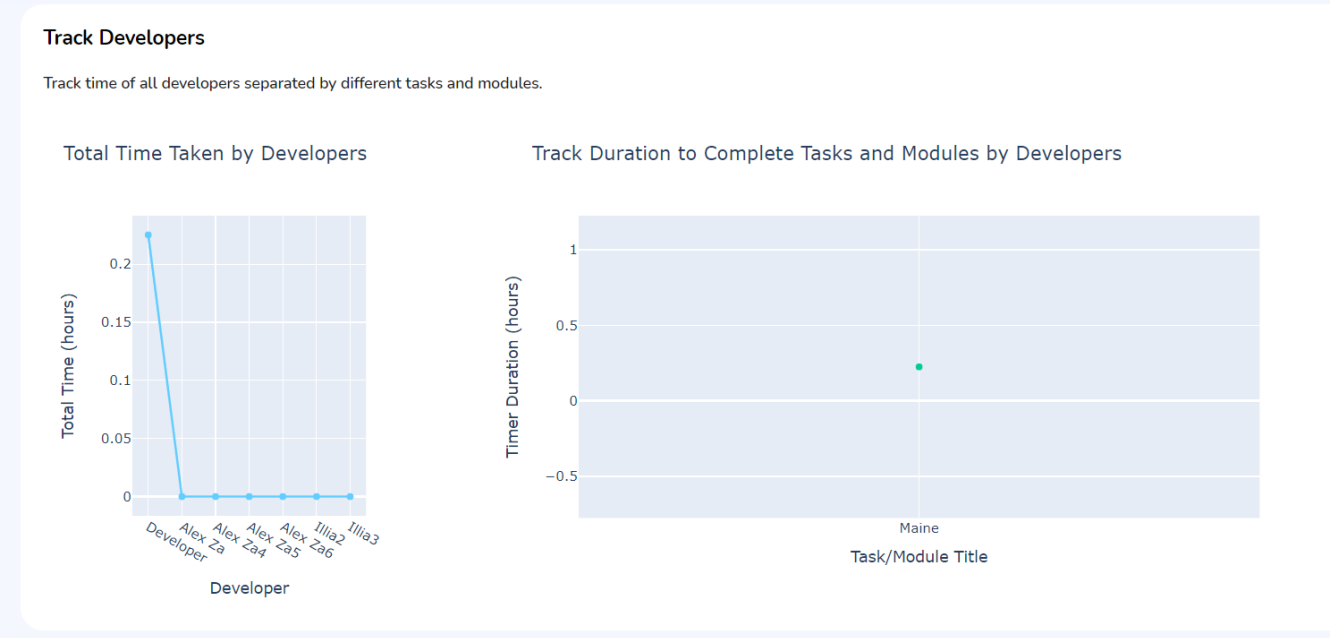

Рисунок 3.11 – Загальний вигляд вікна для аналітики затрат часу і виконаних задач

У таблиці 3.1 порівняно з іншими сучасними системами.

Таблиця 3.1 – Порівняльна характеристика систем

Характеристика	Jira	Asana	Trello	ManageD
1	2	3	4	5
Масштабування функціоналу	-	-	-	+
Зрозумілий інтерфейс	+/-	+	+	+
Ціна	+/-	-	+	+
Перегляд аналітики по проєктах	+	+	-	+
Зміна статусів для завдань	+	+	+	+
Можливість відстеження часу, витраченого на завдання	+	+	-	+

Програмний модуль втілює методологію Agile для IT-розробок. Agile методологія фокусується на гнучкому, ітеративному підході до розробки програмного забезпечення, що дозволяє командам швидко реагувати на зміни та забезпечувати високоякісний продукт через регулярні ітерації (тривалістю від одного до чотирьох тижнів, включає в себе всі етапи розробки: планування, дизайн, розробку, тестування та реліз) та постійний зворотний зв'язок. Така система дозволяє ефективно управляти проєктами, командами, ресурсами та часом, забезпечуючи прозорість та контроль на всіх етапах розробки.

При проєктування системи Vivada, використовуючи різні системи для управління розробкою IT-проєктів, такі як Trello і ManageD, ми порівняли затрати часу і грошей на цю розробку, показано у таблиці 3.2.

Таблиця 3.2 – Порівняльна характеристика затрат часу і грошей

Назва	Час (год)	Гроші(USD)
Trello	960	19200
ManageD	720	14400

3.4 Висновок до розділу 3

У даному розділі було проаналізовано популярні мови програмування та визначено мови для розробки програмного модуля управління IT-проєктами. Використані технології виявилися ефективними та допомогли забезпечити роботу модуля управління IT-проєктами. Вони надали потрібний рівень функціональності, зручність розробки та надійність. Використання Python Django, PostgreSQL, React у поєднанні дозволило створити програмний модуль, який може ефективно використовуватися для управління проєктами, командами, ресурсами та часом.

Обраними програмними засобами було реалізовано програмний модуль системи управління IT-проєктами. Було протестовано та налагоджено роботу програмного модуля системи управління IT-проектами. Модуль управління IT-проєктами є ефективним інструментом для планування, організації, відстеження та керування різними аспектами IT-проєктів. Використання сучасних веб-технологій дозволяє досягти високої продуктивності, автоматизувати процеси та забезпечити гнучкість для управління проєктами будь-якої складності.

ВИСНОВКИ

Поставлені перед бакалаврською дипломною роботою задачі виконано в повному обсязі, а саме:

- проаналізовано методи та засоби управління розробкою ІТ-проєктами;
- проаналізовано актуальність впровадження систем для управління розробкою ІТ-проєктів та аналіз проблем існуючих програмних систем;
- розроблено архітектуру та визначено форми взаємодії функціональних об’єктів інформаційної системи;
- розроблено алгоритми роботи модуля для управління розробкою ІТ-проєктів;
- обґрунтовано вибір технології для розробки програмного модуля управління ІТ-проєктами. Розроблено структурну схему програмного модуля та алгоритм його роботи;
- проаналізовано мови програмування та визначено мови для розробки комплексного програмного модуля управління ІТ-проєктами. Обраними програмними засобами було розроблено програмний модуль управління розробкою ІТ-проєктів;
- проведено тестування розробленого програмного модуля управління ІТ-проєктами. Модуль управління проєктами має високу ефективність, зручність використання, а також розширюваність та адаптивність;
- за допомогою цього модуля ми могли скоротити час розробки програми Vivada на 240 годин та зекономити 4800 дол.

Мета роботи - підвищення ефективності управління ІТ-проєктами шляхом розробки програмного модуля, який здатний ефективно організовувати та відстежувати всі аспекти проєкту. Цей модуль дозволяє покращити планування, контроль та виконання проєктів, забезпечуючи ефективне управління командами, ресурсами та часом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гненний І. О., Іванчук Я. В. "Аналіз перспективи розробки програмного забезпечення для управління розробкою IT-проєктів" в Матеріалах конференції «LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)», Вінниця, 2023. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20969.

2. Особливості системи управління проєктами в IT-компаніях [Електронний ресурс]: Режим доступу: http://www.agrosvit.info/index.php?op=1&z=3205&i=14.

3. Нові підходи в управлінні IT [Електронний ресурс]: Режим доступу: https://eba.com.ua/novi-pidhody-v-upravlinni-abo-yak-biznes-tsili-povyazani-z-protsesamy/.

4. Актуальність та ефективність управління в IT-проєктах: [Електронний ресурс]: Режим доступу: https://economyandsociety.in.ua/index.php/journal/article/view/3342.

5. Водоспадна модель [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D0%BE%D0%B4%D0%BE%D1%81%D0%BF%D0%B0%D0%B4%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C.

6. Гнучка розробка програмного забезпечення [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/%D0%93%D0%BD%D1%83%D1%87%D0%BA%D0%B0_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F.

7. Скрам [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B0%D0%BC.

8. Канбан [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B1%D0%B0%D0%BD.

9. Основи управління [Електронний ресурс]: Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/7c313e5c-5477-4be2-9806-d32e9eace0c3/content.

10. Що таке Асана [Електронний ресурс]: Режим доступу: https://cloudfresh.com/ua/cloud-blog/shho-take-asana-project-management-yak-sprostyty-vash-robochyj-protses/.

11. Jira [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Jira.

12. Trello [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Trello.

13. Модульне програмування [Електронний ресурс]: Режим доступу: https://it-rating.ua/modulne-programuvannya-v-veb-rozrobtsi-perevagi-vikoristannya-moduliv-ta-komponentiv-dlya-pidtrimki-kodu.

14. SMTP [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/SMTP.

15. Переваги і недоліки мови Python [Електронний ресурс]: Режим доступу: https://blog.ithillel.ua/articles/perevagi-i-nedoliki-movi-python.

16. Переваги мови Java [Електронний ресурс]: Режим доступу: https://np.pl.ua/2021/03/perevahy-movy-prohramuvannia-java/.

17. Порівняння React, Angular і Vue [Електронний ресурс]: Режим доступу: https://dou.ua/forums/topic/39933/.

18. React чи Angular [Електронний ресурс]: Режим доступу: https://wezom.com.ua/ua/blog/react-vs-angular.

19. Що таке PostgreSQL? [Електронний ресурс]: Режим доступу: https://www.guru99.com/uk/introduction-postgresql.html.

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмний модуль для управління розробкою IT-проєктів

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95,54% Схожість 4,46%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Гненний І.О.

Керівник роботи Іванчук Я.В.

ДОДАТОК Б

Інструкція користувача

1. Відкриваємо редактор коду Visual Studio Code.

2. Запускаємо сервер бази даних PostgreSQL, щоб програма мала доступ до нашої бази даних.

3. Запускаємо програму за допомогою команди python manage.py runserver.

4. Відкриваємо посилання localhost http://127.0.0.1:8000/ та користуємось програмою.

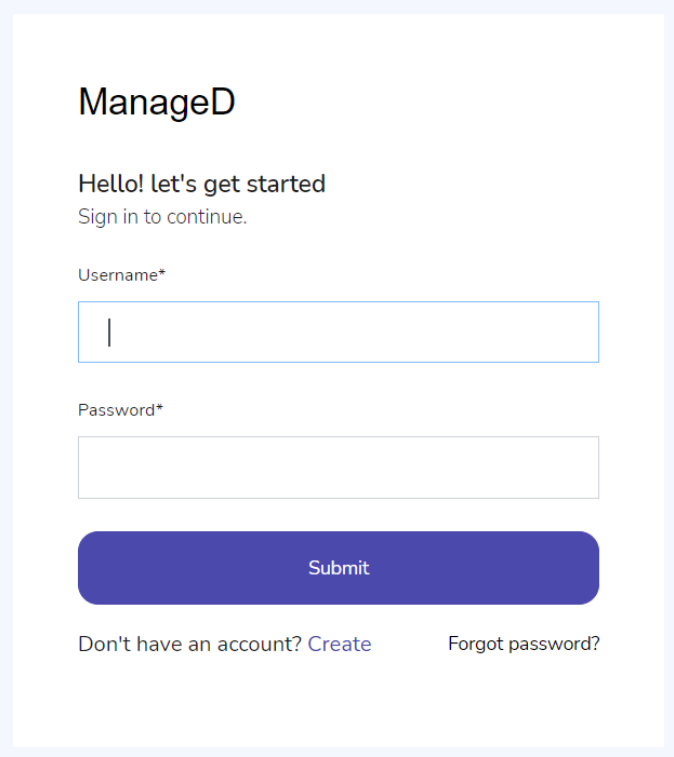

Рисунок Б.1 – Загальний вигляд вікна авторизації

5. Переглядаємо проєкти, додаємо редагуємо видаляємо за необхідності.

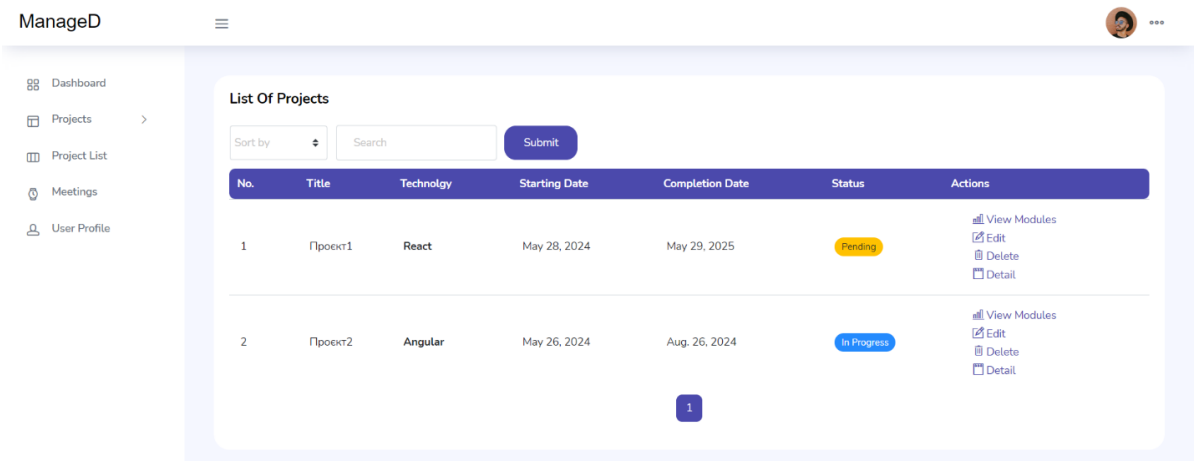

Рисунок Б.2 – Загальний вигляд програмного модуля

6. Після виконання модулів та завдань проєкта отримайте дешборд з усією необхідною аналітикою.

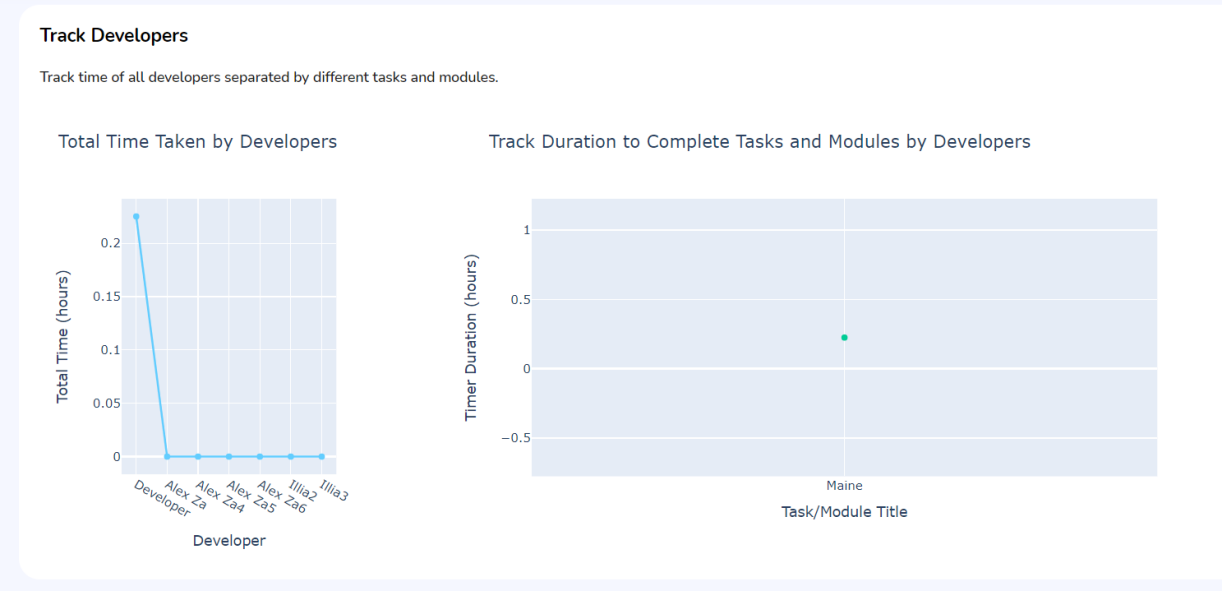

Рисунок Б.3 – Загальний вигляд вікна аналітики

ДОДАТОК В

Лістинг програми

```
from django.contrib import admin
from .models import *
from django.utils.html import format_html

class ProjectAdmin(admin.ModelAdmin):
    save_on_top = True
    list_display=('project_title','project_technology','project_completion_date','status')
    list_display_links=('project_title','project_technology')
    search_fields=['project_title','project_technology','project_completion_date','status']
    list_filter=('status','project_start_date','project_completion_date','project_technology')

    def status_color(self, obj):
        if obj.status == 'Pending':
            return format_html('<span style="color: orange;">{}</span>', obj.status)
        elif obj.status == 'Completed':
            return format_html('<span style="color: green;">{}</span>', obj.status)
        elif obj.status == 'Cancelled':
            return format_html('<span style="color: red;">{}</span>', obj.status)
        elif obj.status == 'In Progress':
            return format_html('<span style="color: blue;">{}</span>', obj.status)
        else:
            return obj.status

    status_color.short_description = 'Status'

class Project_TeamAdmin(admin.ModelAdmin):
    save_on_top = True
    list_display=('team_name','project','status')
    list_display_links=('team_name','project')
    search_fields=['team_name','status']
    list_filter=('status','project')
```

```
class Project_ModuleAdmin(admin.ModelAdmin):
    save_on_top = True
    list_display=('module_name','project','module_completion_date','status')
    list_display_links=('module_name','project')
    search_fields=['module_name','module_estimated_hours','status','module_start_date','module_completion_date']
    list_filter=('status','module_estimated_hours','module_completion_date','module_start_date')

class Project_TaskAdmin(admin.ModelAdmin):
    save_on_top = True
    list_display=('task_title','module','project','user','priority','status','start_time','end_time')
    list_display_links=('task_title','module','project')
    search_fields=['task_title','task_description','priority','status','start_time','end_time']
    list_filter=('priority','status','module','project','start_time','end_time')

class User_TaskAdmin(admin.ModelAdmin):
    save_on_top = True
    list_display=('task','user','status')
    list_filter=('user_totalutil_minutes',)

class Developer_SubmitAdmin(admin.ModelAdmin):
    save_on_top = True
    list_display=('submit_title','submit_developer_name','submit_submit_date','status')
    list_display_links=('submit_title','submit_developer_name')
    search_fields=['submit_title','submit_developer_name','status','submit_file','submit_screenshots','submit_submit_date']
    list_filter=('submit_submit_date','status')

admin.site.register(Project,ProjectAdmin)
admin.site.register(Project_Team,Project_TeamAdmin)
admin.site.register(Project_Module,Project_ModuleAdmin)
admin.site.register(Project_Task,Project_TaskAdmin)
admin.site.register(User_Task,User_TaskAdmin)
admin.site.register(Developer_Submit,Developer_SubmitAdmin)
admin.site.register(TaskTimer)

from django.apps import AppConfig
```

```
class ProjectConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'project'


import django.forms as form
from django import forms
from .models import *
from user.models import User
from django.forms import DateTimeInput,DateInput
from crispy_forms.helper import FormHelper
from crispy_forms.layout import Submit
from django.forms import CheckboxSelectMultiple
from crispy_forms.bootstrap import PrependedText
from crispy_forms.layout import Submit, Layout
from django.core.mail import send_mail
from django.template.loader import render_to_string
from django.utils.html import strip_tags
from django.core.mail import EmailMessage
from django.template.defaultfilters import linebreaks


class AddProjectsForm(form.ModelForm):


    project_start_date = forms.DateField(widget=DateInput(attrs={'type': 'date'}))
    project_completion_date = forms.DateField(widget=DateInput(attrs={'type': 'date'}))
    helper = FormHelper()
    helper.form_method = 'POST'
    helper.add_input(Submit('submit', 'Submit'))


    class Meta:
        model = Project
        fields = '__all__'


class ProjectModulesForm(form.ModelForm):
```

```
    user = form.ModelChoiceField(queryset=User.objects.filter(is_developer=True))
    module_start_date = forms.DateTimeField(widget=DateTimeInput(attrs={'type': 'datetime-local'}))
    module_completion_date = forms.DateTimeField(widget=DateTimeInput(attrs={'type': 'datetime-local'}))
    helper = FormHelper()
    helper.form_method = 'POST'
    helper.add_input(Submit('submit', 'Submit'))
    class Meta:
        model = Project_Module
        fields = '__all__'

class ProjectTaskForm(form.ModelForm):

    start_time = forms.DateTimeField(widget=DateTimeInput(attrs={'type': 'datetime-local'}))
    end_time = forms.DateTimeField(widget=DateTimeInput(attrs={'type': 'datetime-local'}))
    helper = FormHelper()
    helper.form_method = 'POST'
    helper.add_input(Submit('submit', 'Submit'))

    user = form.ModelChoiceField(queryset=User.objects.filter(is_developer=True))
    class Meta:
        model = Project_Task
        fields = '__all__'

class ProjectTeamForm(form.ModelForm):
    user = forms.ModelMultipleChoiceField(
        queryset=User.objects.filter(is_developer=True),
        widget=CheckboxSelectMultiple,
    )
    helper = FormHelper()
    helper.form_method = 'POST'
    helper.add_input(Submit('submit', 'Submit'))
    helper.layout = Layout(
        PrependedText('user', 'user'),
    )
```

```
    class Meta:
        model = Project_Team
        fields ='__all__'

class UserTaskForm(form.ModelForm):
    class Meta:
        model = User_Task
        fields = '__all__'



class DeveloperSubmitForm(forms.ModelForm):

    submit_submit_date = forms.DateTimeField(widget=DateTimeInput(attrs={'type': 'datetime-local'}))
    helper = FormHelper()
    helper.form_method = 'POST'
    helper.add_input(Submit('submit', 'Submit'))

    submit_developer_email = forms.EmailField(label='Developer Email', required=True)



    class Meta:
        model = Developer_Submit
        fields = '__all__'
        widgets = {
            'submit_manager_name': forms.TextInput(attrs={'class': 'form-control', 'editable': True}),
        }

    def __init__(self, user, *args, **kwargs):
        super(DeveloperSubmitForm, self).__init__(*args, **kwargs)
        self.fields['submit_developer_name'].initial = user.username
        self.fields['submit_developer_email'].initial = user.email
        manager = User.objects.filter(is_manager=True).first()
        self.fields['submit_manager_name'].initial = manager.username
        self.fields['submit_submit_date'].initial = timezone.now()
```

```
    def send_email_to_manager(self):
        subject = 'Developer Submission'
        message = render_to_string('project/developer_submission_email.html', {'form': self})
        plain_message = strip_tags(message)
        manager = User.objects.filter(is_manager=True).first()
        to_email = manager.email
        sender_email = self.cleaned_data.get('submit_developer_email')

        # Apply linebreaks filter to the message content
        message_with_linebreaks = linebreaks(plain_message)

        # Create EmailMessage object
        email = EmailMessage(subject, message_with_linebreaks, sender_email, [to_email])
        email.content_subtype = 'html'  # Set content type as HTML

        # Attach submit_file if present
        submit_file = self.cleaned_data.get('submit_file')
        if submit_file:
            file_name = submit_file.name
            file_content = submit_file.read()
            email.attach(file_name, file_content)

        # Attach submit_screenshot if present
        submit_screenshot = self.cleaned_data.get('submit_screenshots')
        if submit_screenshot:
            screenshot_name = submit_screenshot.name
            screenshot_content = submit_screenshot.read()
            email.attach(screenshot_name, screenshot_content)

        # Send the email
        email.send(fail_silently=False)

#Project Models
from datetime import datetime, timezone
from django.conf import settings
```

```
from django.db import models
from django.urls import reverse
from user.models import User
from django.contrib import messages
from django.db.models.signals import post_save
from django.dispatch import receiver
from ckeditor.fields import RichTextField
from django.core.mail import send_mail
from django.utils import timezone
from datetime import timedelta

# Status Class Don't consider this class as it is not used anywhere
status_choice = (("Completed","Completed"),
                 ("Pending","Pending"),
                 ("Cancelled","Cancelled"))
class Status(models.Model):
    status_name = models.CharField(choices=status_choice,max_length=100)

    class Meta:
        db_table='status'

    def __str__(self):
        return self.status_name



# Project Class
status_choice = (("Completed","Completed"),
                 ("Pending","Pending"),
                 ("In Progress","In Progress"),
                 ("Cancelled","Cancelled"))
class Project(models.Model):
    project_title = models.CharField(max_length=100)
    project_decription = RichTextField(null=True,blank=True)
    project_technology = models.CharField(max_length=100)
    project_estimated_hours = models.IntegerField()
```

```
    project_start_date = models.DateField()
    project_completion_date = models.DateField()
    project_file = models.FileField(upload_to='project_files/',null=True,blank=True)
    status = models.CharField(choices=status_choice,max_length=100, default='Pending')

    class Meta:
        db_table='project'

    def __str__ (self):
        return self.project_title



# Project Module Class
status_choice = (("Completed","Completed"),
                ("Pending","Pending"),
                ("In Progress","In Progress"),
                ("Cancelled","Cancelled"))
class Project_Module(models.Model):
    project = models.ForeignKey(Project,on_delete=models.CASCADE)
    module_name = models.CharField(max_length=100)
    module_description = RichTextField(null=True,blank=True)
    module_estimated_hours = models.IntegerField()
    timer_duration = models.DurationField(blank=True, null=True, default=0)
    module_start_date = models.DateTimeField()
    module_completion_date = models.DateTimeField()
    user= models.ForeignKey(User,on_delete=models.CASCADE,default=True)
    status = models.CharField(choices=status_choice,max_length=100,default='Pending')

    class Meta:
        db_table='project_module'

    def __str__(self):
        return self.module_name
```

```
# Project Task Class
status_choice = (("Completed","Completed"),
                 ("Pending","Pending"),
                 ("In Progress","In Progress"),
                 ("Cancelled","Cancelled"))
priorityChoice=(
  ('High','High Priority'),
  ('Less','Less Priority')
)
class Project_Task(models.Model):
  module = models.ForeignKey(Project_Module,on_delete=models.CASCADE,null=True,blank=True)
  project = models.ForeignKey(Project,on_delete=models.CASCADE,null=True,blank=True)
  task_title = models.CharField(max_length=100)
  task_description = RichTextField(null=True,blank=True)
  priority = models.CharField(choices=priorityChoice,max_length=30)
  user= models.ForeignKey(User,on_delete=models.CASCADE,null=True,blank=True)
  task_estimated_hours = models.IntegerField()
  timer_duration = models.DurationField(blank=True, null=True,default=0)
  status = models.CharField(choices=status_choice,max_length=100, default='Pending')
  start_time = models.DateTimeField(null=True, blank=True)
  end_time = models.DateTimeField(null=True, blank=True)


  def save(self, *args, **kwargs):
      if self.status == 'Pending':
          delta = self.end_time - timezone.now()
          if delta.days == 3:
              subject = 'Reminder: Task pending for module completion'
              message = f'Hello {self.user.username},\n\nYou have a pending task "{self.task_title}" for the module "{self.module.module_name}". The task is scheduled to end in 3 days ({self.module.module_completion_date}). Please complete the task before the module ends.\n\nPriority: {self.priority}\n\nThank you.'
              send_mail(subject, message, from_email=settings.EMAIL_HOST_USER, recipient_list=[self.user.email], fail_silently=False)
      elif self.status == 'In Progress':
          delta = self.end_time - timezone.now()
          if delta.days == 3:
              subject = 'Reminder: Task in progress for module completion'
```

```
            message = f'Hello {self.user.username},\n\nYou have a task "{self.task_title}" in progress for the module "{self.module.module_name}". The module is scheduled to end in 3 days ({self.module.module_completion_date}). Please complete the task before the module ends.\n\nPriority: {self.priority}\n\nThank you.'

            send_mail(subject, message, from_email=settings.EMAIL_HOST_USER, recipient_list=[self.user.email], fail_silently=False)

        super().save(*args, **kwargs)


    class Meta:
        db_table='project_task'


    def __str__(self):
        return self.task_title



# Badge Class


badgeChoice=(
    ('InProgress','InProgress'),
    ('QuickFinisher','QuickFinisher'),
    ('LazyLoader','LazyLoader'),
    ('SilentUser','SilentUser')
)
class Badge(models.Model):
    badge = models.CharField(choices=badgeChoice,max_length=25)
    class Meta:
        db_table='badge'


    def __str__(self):
        return self.badge



# Project Team Class
status_choice = (("Completed","Completed"),
                 ("Pending","Pending"),
                 ("Cancelled","Cancelled"))
badgeChoice=(
```

```
    ('IN','InProgress'),
    ('QF','QuickFinisher'),
    ('LL','LazyLoader'),
    ('SU','SilentUser')
)
class Project_Team(models.Model):
    team_name = models.CharField(max_length=100)
    project = models.ForeignKey(Project,on_delete=models.CASCADE)
    user= models.ManyToManyField(User)
    status = models.CharField(choices=status_choice,max_length=100)
    badge = models.CharField(choices=badgeChoice,max_length=25)

    class Meta:
        db_table = 'project_team'

    def __str__(self):
        return self.team_name



# User Task Class
status_choice = (("Completed","Completed"),
                ("Pending","Pending"),
                ("Cancelled","Cancelled"))
class User_Task(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE)
    task = models.ForeignKey(Project_Task,on_delete=models.CASCADE)
    status = models.CharField(choices=status_choice,max_length=100)
    user_totalutil_minutes = models.IntegerField()

    class Meta:
        db_table='user_task'

    def __str__(self):
        return self.user_id
```

```
# Task Badge Class


class Task_Badge(models.Model):
    badge = models.ForeignKey(Badge,on_delete=models.CASCADE)
    task = models.ForeignKey(Project_Task,on_delete=models.CASCADE)


    class Meta:
        db_table='task_badge'


    def __str__(self):
        return self.badge


# Developer Submission Class
status_choice = (("Completed","Completed"),
                 ("In Progress","In Progress"),
                 ("Pending","Pending"),
                 ("Cancelled","Cancelled"))
class Developer_Submit(models.Model):
    submit_title = models.CharField(max_length=200)
    submit_description = RichTextField(null=True,blank=True)
    code_snippets = RichTextField(null=True,blank=True)
    submit_screenshots = models.ImageField(upload_to='developer_task_screenshots/', blank=True, null=True)
    submit_file = models.FileField(upload_to='developer_task_files/',blank=True,null=True)
    status = models.CharField(choices=status_choice,max_length=50, default='In Progress')
    submit_time_spent = models.DurationField(blank=True, null=True)
    submit_submit_date = models.DateTimeField(auto_now_add=True)
    submit_developer_name = models.CharField(max_length=100)
    submit_developer_email = models.EmailField(null=True,blank=True)
    submit_manager_name = models.CharField(max_length=100)
    comments = RichTextField(null=True,blank=True)
    submit_url = models.URLField(blank=True, null=True)


    class Meta:
        db_table='developer_submit'
```

```
    def __str__(self):
        return self.submit_title

class TaskTimer(models.Model):
    task = models.ForeignKey(Project_Task, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    start_time = models.DateTimeField(null=True, blank=True)
    end_time = models.DateTimeField(null=True, blank=True)
    is_paused = models.BooleanField(default=False)
    pause_time = models.DateTimeField(null=True, blank=True)

    class Meta:
        db_table = 'TaskTimer'

    def __str__(self):
        return self.start_time

class TaskTime(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    task = models.ForeignKey(Project_Task, on_delete=models.CASCADE)
    start_time = models.DateTimeField()
    end_time = models.DateTimeField()
    duration = models.DurationField()

    class Meta:
        db_table = 'TaskTime'

    def save(self, *args, **kwargs):
        if self.start_time is not None and self.end_time is not None:
            self.duration = self.end_time - self.start_time
        super(TaskTime, self).save(*args, **kwargs)
```

ДОДАТОК Г

Графічна частина

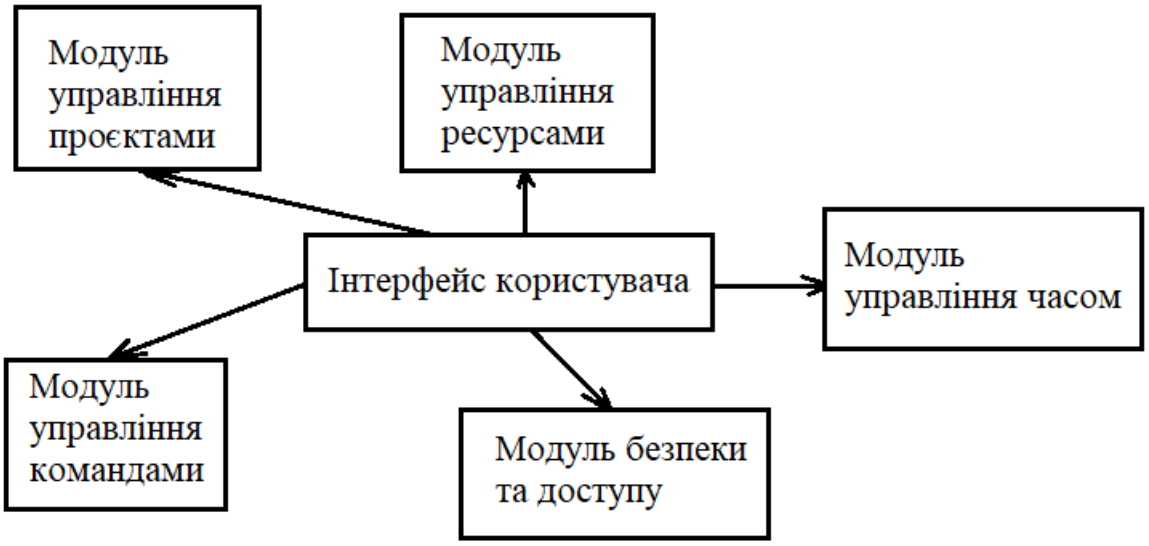

Рисунок Г.1 - Загальна структурна схема програмного модуля для управління розробкою IT-проєктів

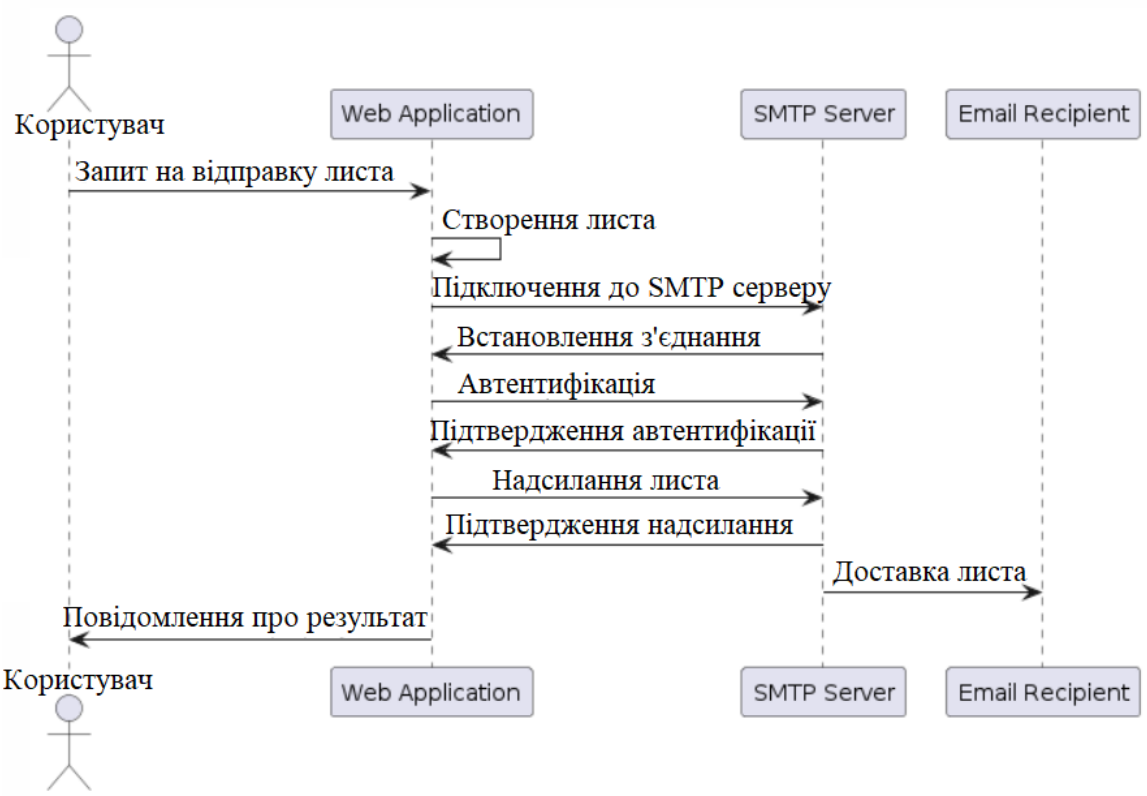

Рисунок Г.2 - Типова схема взаємодії WEB-додатка із SMTP сервером

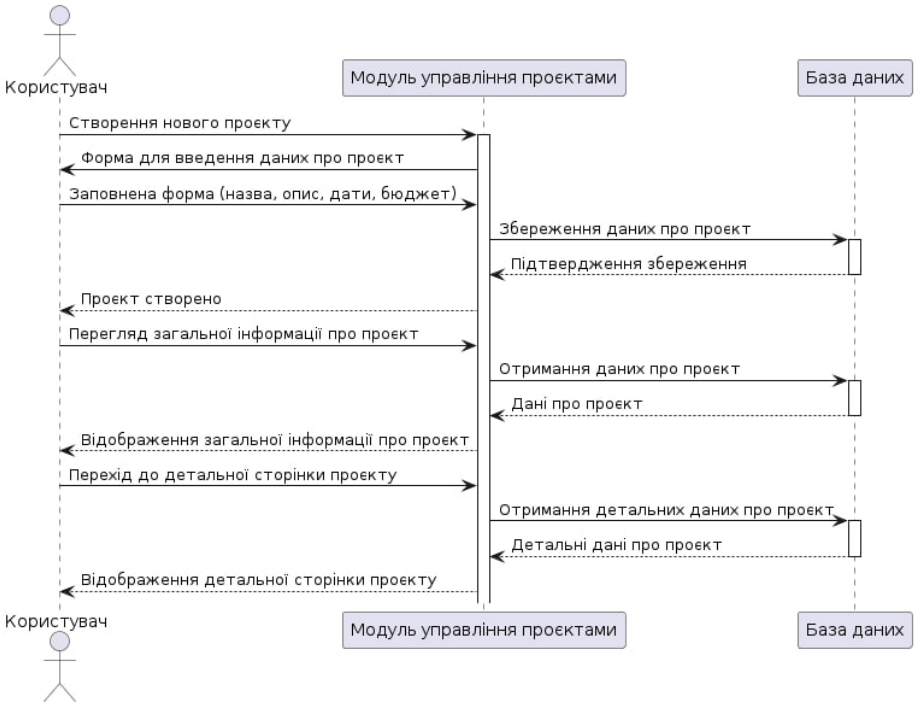

Рисунок Г.3 – Типова схема модуля управління проєктами

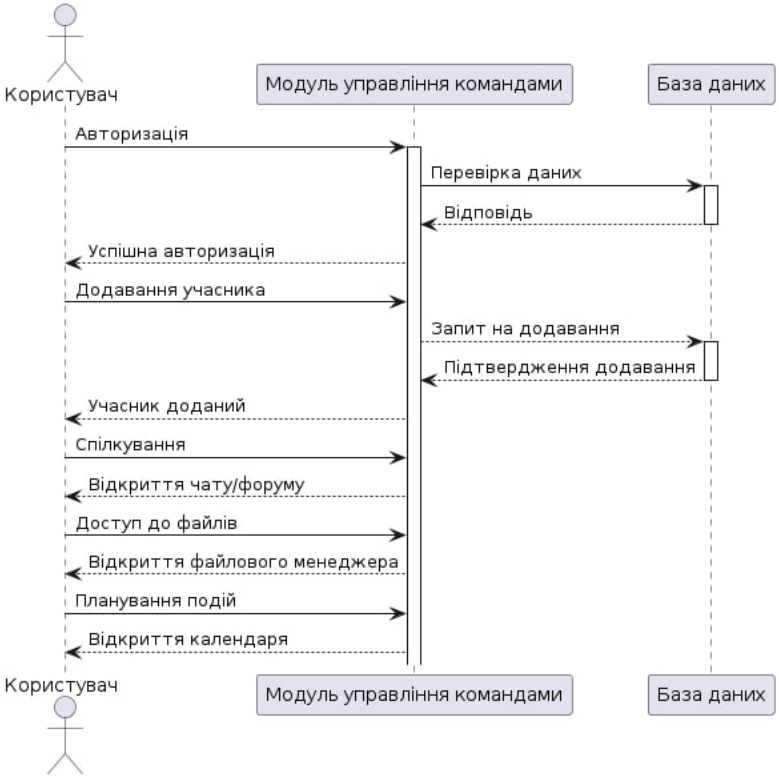

Рисунок Г.4 – Типова схема модуля управління командами

Рисунок Г.5 – Типова схема модуля управління часом

Рисунок Г.6 – Схема алгоритму реєстрації користувача у модулі для управління ІТ-проєктів

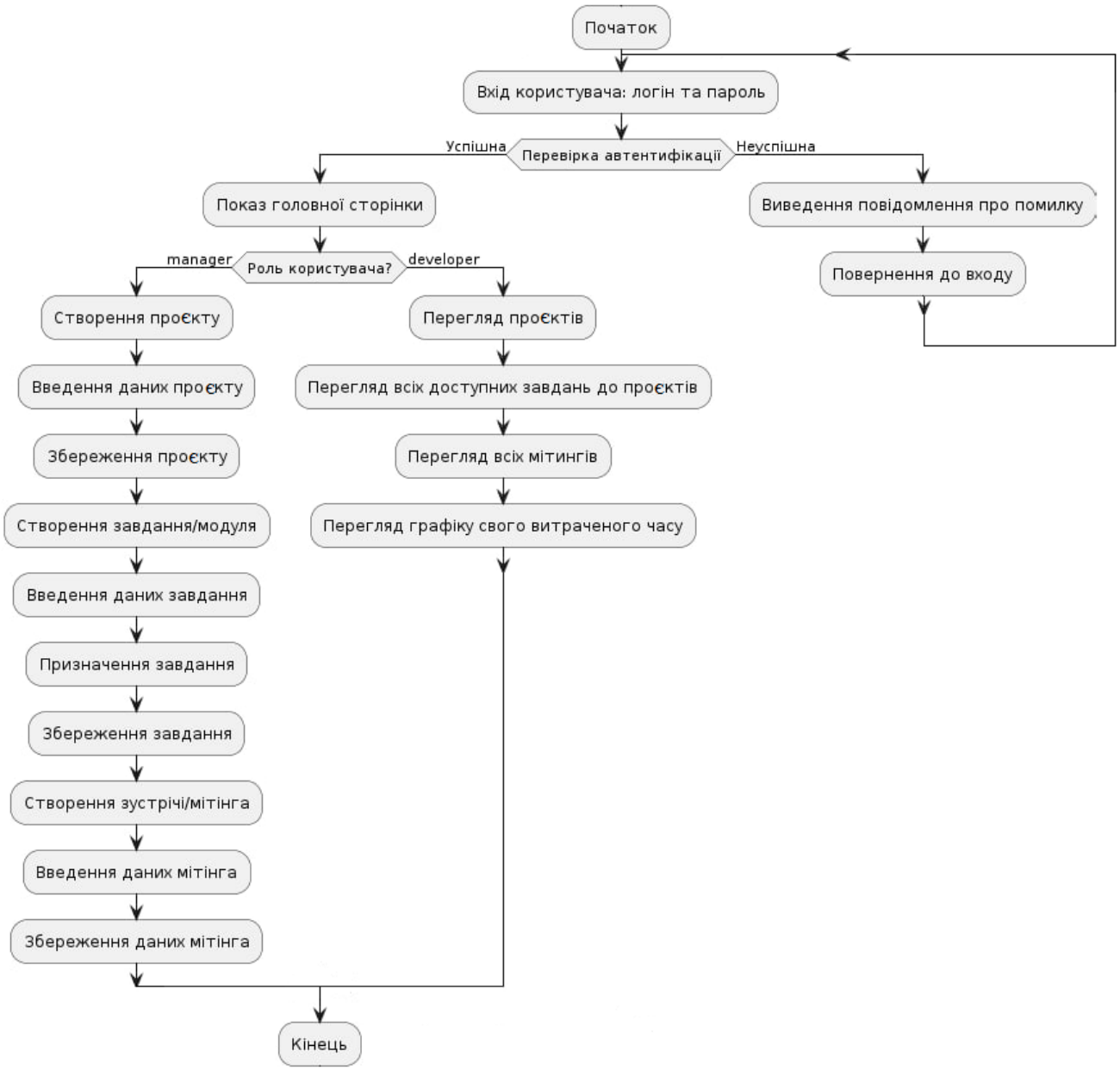

Рисунок Г.7 – Схема алгоритму функціонування програмного модуля для управління розробкою ІТ-проєктів

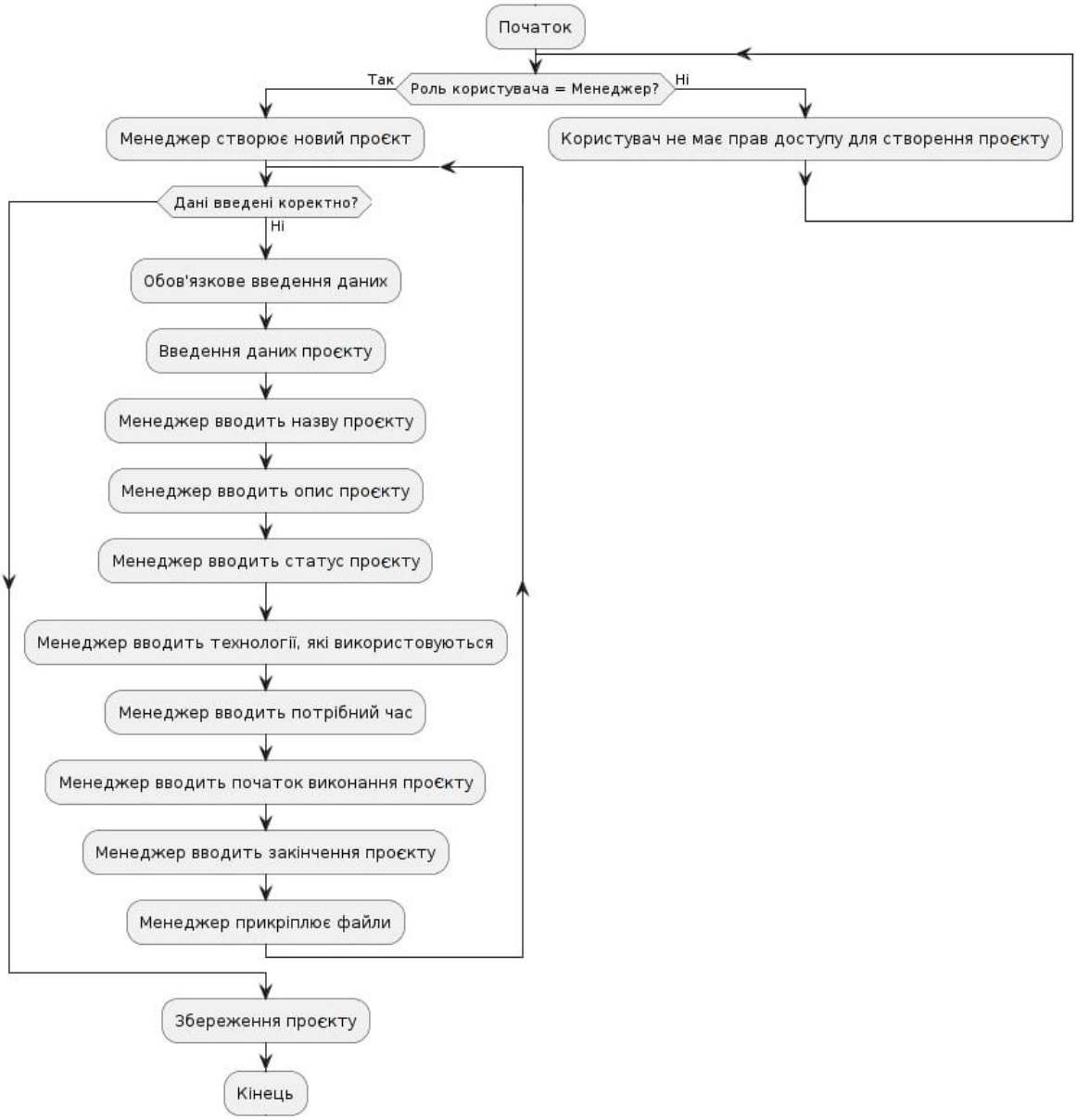

Рисунок Г.8 – Схема алгоритму створення проєкта у модулі для управління розробкою ІТ-проєктів

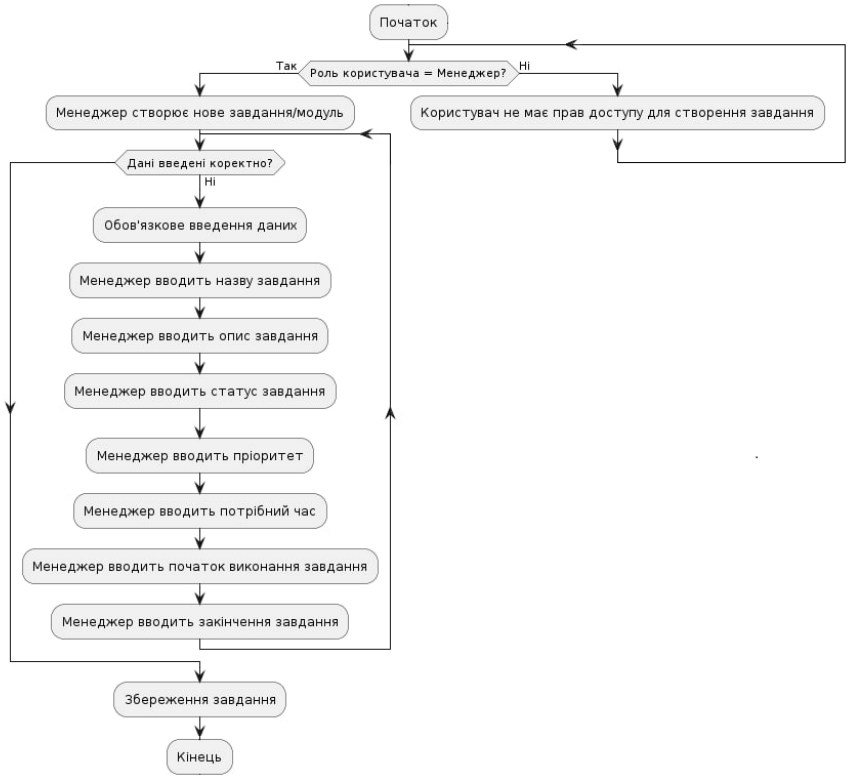

Рисунок Г.9 – Схема алгоритму створення завдання у модулі для управління розробкою IT-проєктів

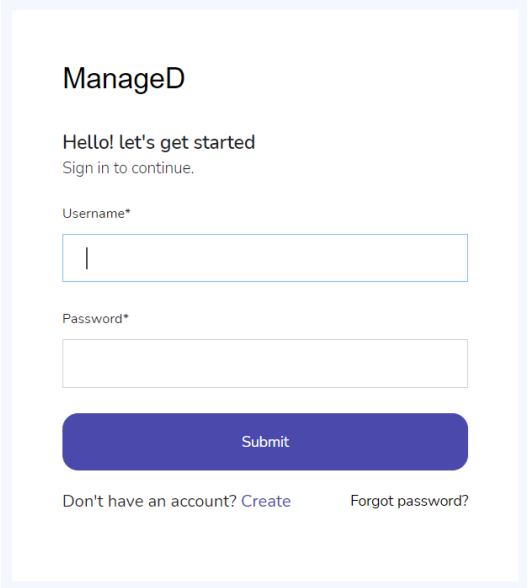

Рисунок Г.10 – Загальний вигляд вікна авторизації

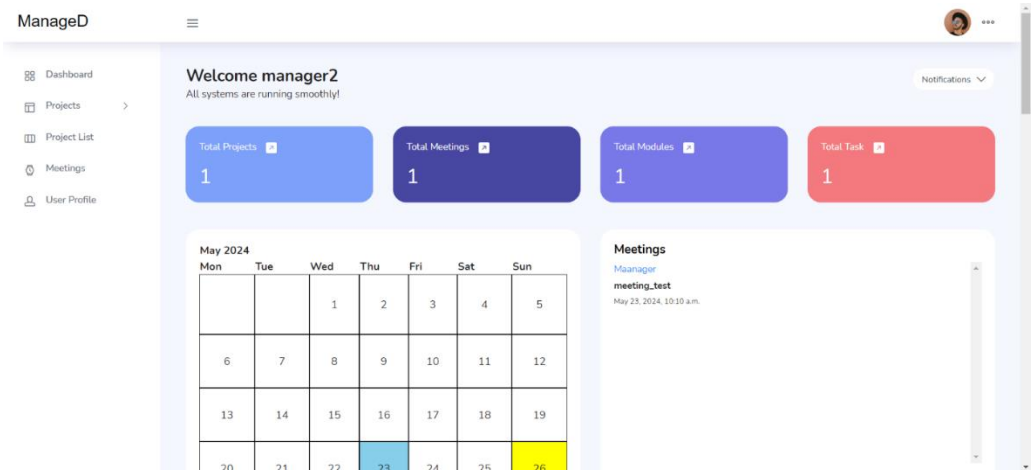

Рисунок Г.11 – Загальний вигляд системи ManageD для управління розробкою IT-проєктів

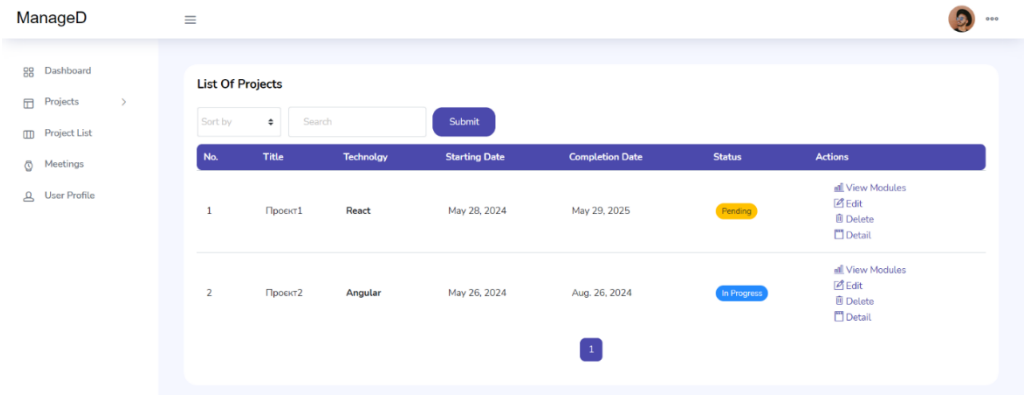

Рисунок Г.12 – Загальний вигляд вікна зі списком проєктів

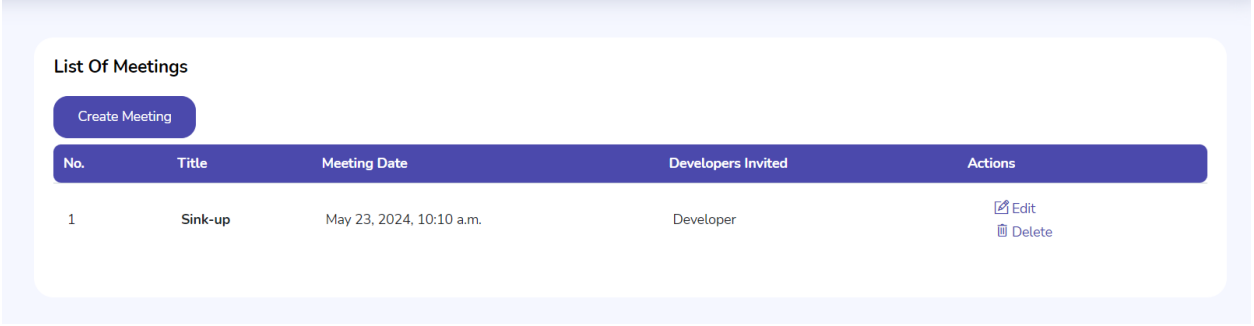

Рисунок Г.13 – Загальний вигляд вікна зі списком мітінгів

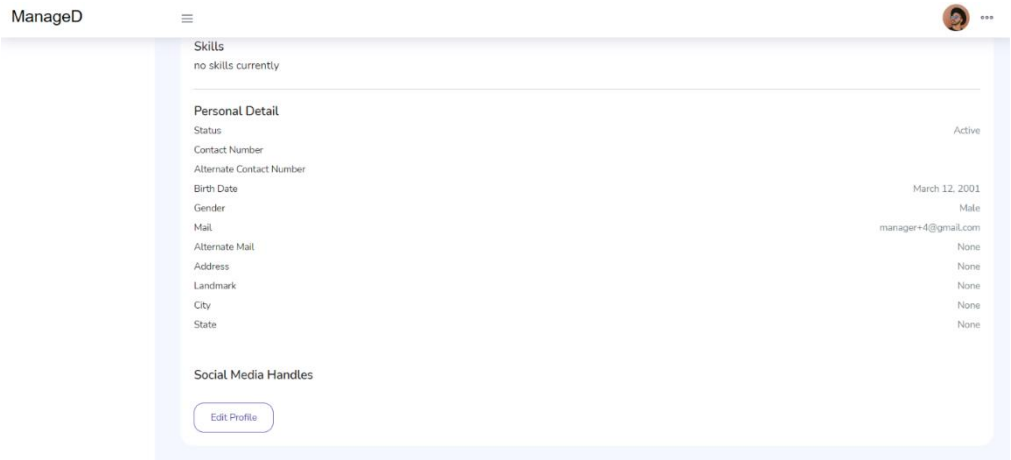

Рисунок Г.14 – Загальний вигляд вікна для редагування особистої інформації

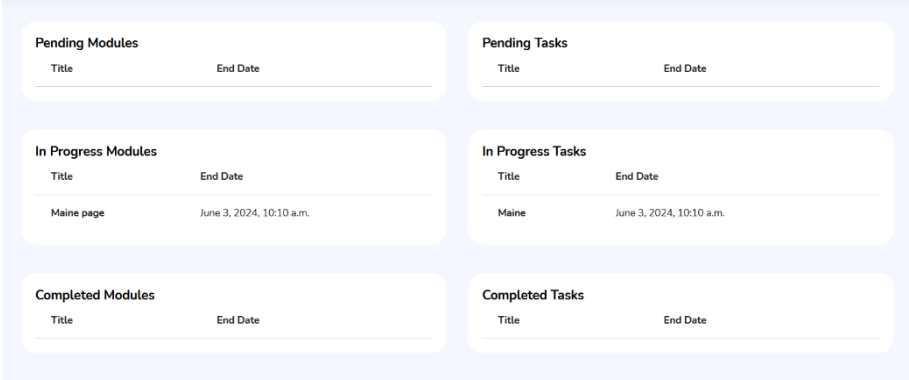

Рисунок Г.15 – Загальний вигляд вікна усіх модулів і задач

Рисунок Г.16 – Загальний вигляд вікна для аналітики затрат часу і виконаних задач