МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук,

доцент _____ Є. О. Давиденко

«___» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК ВІДСЛІДКОВУВАННЯ ТА ПІДТРИМКИ ПРОДУКТИВНОСТІ ІТ-ФАХІВЦІВ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.22011009

Здобувачка

_____ Є. В. Костюк

підпис

«__» _____ 2024 р.

Керівник ст. викладачка

_____ С. Ю. Боровльова

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент

_____ А. О. Алексєєва

підпис

«__» _____ 2024 р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри Давиденко Є. О. _____

« _24_ » ____січня____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи _409_ факультету комп'ютерних наук_____

__Костюк Єлизаветі Валеріївні_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Вебзастосунок відслідковування та підтримки продуктивності роботи ІТ-фахівців_____
_____

Затверджена наказом по ЧНУ від « 22 » ___грудня___ 2024 р. № _____269____

2. Строк представлення кваліфікаційної роботи «__» _____ 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є функціонуючий вебзастосунок відслідковування та підтримки продуктивності роботи ІТ фахівців_____
_____

4. Перелік питань, що підлягають розробці

_Аналіз сучасних методологій та інструментів для підтримки та відслідковування продуктивності у роботі, розробка зручного для користувачів інтерфейсу, моделювання та проєктування системи, розробка ПЗ, тестування роботи ПЗ та здійснення аналізу результатів розробки_____
_____

5. Перелік графічних матеріалів

Презентація

6. Завдання до спеціальної частини

Дослідження питань охорони праці, які безпосередньо пов'язані з діяльністю розробника програмного забезпечення

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О.	Кафедра екології Медичного інституту ЧНУ ім. Петра Могили	Спеціальна частина з охорони праці

Керівник роботи ст. викладачка Боровльова Світлана Юріївна
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання Костюк Єлизавета Валеріївна
(прізвище, ім'я, по батькові)

_____
(підпис)

Дата видачі завдання «_24_» ____січня_____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок відслідковування та підтримки продуктивності роботи ІТ-фахівців_____
_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	23.01.2024	24.01.2024	Виконано
2.	Огляд літератури за темою роботи	25.01.2024	27.01.2024	Виконано
3.	Складання календарного плану КРБ	28.01.2024	28.01.2024	Виконано
4.	Аналіз предметної області	29.01.2024	03.02.2024	Виконано
5.	Розробка проєктних рішень	04.02.2024	06.02.2024	Виконано
6.	Моделювання та конструювання ПЗ	07.02.2024	12.02.2024	Виконано
7.	Кодування, тестування розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	13.02.2024	15.04.2024	Виконано
8.	Розробка спеціальної частини з охорони праці	16.04.2024	20.04.2024	Виконано
9.	Оформлення КРБ та презентації	21.04.2024	01.04.2024	Виконано
10.	Відгук керівника КРБ	28.05.2024	28.05.2024	Виконано
11.	Попередній захист	03.06.2024	03.06.2024	Виконано
12.	Рецензування	15.06.2024	18.06.2024	Виконано
13.	Захист кваліфікаційної роботи			

Розробила студентка Костюк Єлизавета Валеріївна _____
(прізвище, ім'я, по батькові) *(підпис)*

« _28_ » __січня___ 2024 р.

Керівник роботи ст. викладачка Боровльова С. Ю. _____
(посада, прізвище, ім'я, по батькові) *(підпис)*

« _28_ » __січня___ 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок відслідковування та підтримки продуктивності роботи ІТ-фахівців»

Студентка 409 гр.: Костюк Єлизавета Валеріївна

Керівник: ст. викладачка кафедри ІПЗ, Боровльова С. Ю.

Кваліфікаційна робота присвячена розробці програмного забезпечення для автоматизації процесу відслідковування та підтримки продуктивності роботи ІТ-фахівців зі збереженням ефективності та попередженням вигорання. Актуальність полягає в тому, що в теперішній час існує велика проблема збереження продуктивності та концентрації, тому важливо мати інструменти для запобігання відволікань та планування.

Об’єкт роботи: процеси управління продуктивністю роботи ІТ-фахівців.

Предмет роботи: набір інструментів та підходів до відслідковування та підтримки продуктивності ІТ-фахівців.

Мета роботи полягає у створенні вебзастосунку, який дозволить відслідковувати та підтримувати продуктивність роботи ІТ-фахівців.

Кваліфікаційна робота бакалавра складається зі вступу, чотирьох розділів, висновку та переліку джерел посилання.

У вступі наводиться актуальність теми, а також визначаються мета та завдання для реалізації програмного застосунку.

У першому розділі аналізуються існуючі програмні рішення для підтримки та відслідковування продуктивності, а також формується специфікація вимог до вебзастосунку, що розробляється.

У другому розділі описується моделювання об’єкту та предмету дослідження, а також функціональні та інформаційні моделі програмного забезпечення.

У третьому розділі описується процес розробки архітектури, моделювання та проєктування програмного забезпечення, що включає вибір мов програмування, технологій та бібліотек, а також UML-діаграми та інтерфейс користувача з вербальним описом.

У четвертому розділі демонструється процес роботи з кодуванням та тестуванням вебзастосунку відслідковування та підтримки продуктивності роботи ІТ-фахівців.

У висновках аналізуються отримані результати.

Кваліфікаційна робота бакалавра викладена на 71 сторінку, вона містить 4 розділи, 32 ілюстрації, 14 таблиць, 20 джерел в переліку посилань.

Ключові слова: вебзастосунок, розробка на Django, JavaScript, відслідковування та підтримка продуктивності, автоматизація процесів ведення задач, комунікація в команді, попередження вигорання програмістів.

ABSTRACT

of the Bachelor's Thesis

"Web application for monitoring and supporting the productivity of IT specialists"

Student: Kostiuk Yelyzaveta Valeriivna

Supervisor: Senior Lecturer of the Department of Software Engineering

Borovlova S. Y.

The qualification work is devoted to the development of software for automating the process of monitoring and supporting the productivity of IT specialists while maintaining efficiency and preventing burnout. The relevance is that nowadays there is a big problem of staying productive and focused, so you should know about the tools to deal with it.

The object of work: the process of managing the work productivity of IT specialists.

The subject of work: a set of tools and approaches to monitor and support the productivity of IT specialists.

Objective: to create a web application that allows to monitor and support the productivity of IT specialists.

The bachelor's qualification work consists of an introduction, four sections, a conclusion and a list of reference sources.

The introduction states the relevance of the topic, and defines the purpose and tasks for implementing the software application.

The first section analyzes the existing software solutions for support and performance monitoring, and also forms the specification of requirements for the web application under development.

The second section describes the modeling of the object and the subject of research, as well as the functional and information models of the software.

The third chapter describes the process of architecture development, modeling and software design, including the choice of programming languages, technologies and libraries, as well as UML diagrams and a user interface with verbal description.

The fourth chapter demonstrates the process of coding and testing the web application for monitoring and supporting the productivity of IT professionals.

The conclusions analyze obtained results.

The qualification work of the bachelor is presented on 71 pages, it contains 4 sections, 32 illustrations, 14 tables, 20 sources in the list of references.

Keywords: web application, development on Django, JavaScript, monitoring and support of productivity, automation of task management processes, team communication, prevention of programmer burnout.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних
БЕМ – блок, елемент, модифікатор
КРБ – кваліфікаційна робота бакалавра
ПЗ – програмне забезпечення
СКБД – система керування базами даних
ЦА – цільова аудиторія

API – application programming interface
MVT – model-view-template
CSS – cascading style sheets
CRUD – create, read, update, delete
HTML – hypertext preprocessor
JS – JavaScript
SQL – Structured Query Language
UI – User Interface
UML – Unified Modeling Language

ВСТУП

Останні десятиліття супроводжуються стрімким розвитком інформаційних технологій, що зумовлює зростання числа ІТ-фахівців, проте усталений концепт необхідності встигнути все й одразу призводить до неприємних наслідків, серед яких найбільш популярним є вигорання [1]. Йому передують рішення планувати свою роботу — складати списки задач та графіки роботи, делегувати обов’язки та експериментувати з підходами до тайм-менеджменту. Проте, важливо саме правильно підійти до питання підтримки та відслідковування продуктивності, оскільки помилкові рішення можуть призвести навіть до погіршення поточної ситуації [2].

Дослідження теми дозволить виокремити найоптимальніші для конкретних осіб підходи до організації робочих процесів, що мають практичний потенціал для підвищення продуктивності та забезпечення стабільності діяльності підприємства чи самостійного розвитку фахівця.

Актуальність теми полягає у тому, що ІТ-фахівцеві необхідно розуміти свої можливості та ресурси для організації продуктивного навчання та роботи у власній області. В контексті постійних змін технологій та бізнес-вимог важливо забезпечити ефективність та запобігти вигоранню, що трапляється доволі часто при неправильному плануванні робочих процесів та відпочинку [3]4. Окрім цього, продуктивність роботи окремих спеціалістів напряму впливає на результативність та конкурентоспроможність компаній, в яких вони працюють.

Об’єктом роботи є процеси управління продуктивністю роботи ІТ-фахівців.

Предметом є набір інструментів та підходів для відслідковування та підтримки продуктивності ІТ-фахівців.

Метою роботи є створення вебзастосунку, який дозволить відслідковувати та підтримувати продуктивність роботи ІТ-фахівців.

Для досягнення цієї мети виокремлено ряд наступних **завдань**, які необхідно розв’язати:

– проаналізувати сучасні методології та інструменти для підтримки та відслідковування продуктивності у роботі;

– розробити зручний інтерфейс, що передбачатиме користувацькі побажання та не відволікатиме їх від роботи;

– спроєктувати систему та реалізувати її у вигляді вебзастосунку;

– додати інструменти попередження вигорання для реалізованих процесів у вебзастосунку;

– протестувати працездатність та ефективність вебзастосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Виділення часу для аналітичної складової при розробці проєкту грає велику роль для забезпечення коректності системи, її підтримці та розширюваності. На цьому етапі не лише розглядається сама задача, але й описується ряд структурних та функціональних особливостей об’єкту дослідження, а також формується специфікація вимог до відповідного програмного забезпечення.

Правильний початок забезпечує гарний фундамент для роботи над проєктом та гарантує наявність необхідних даних, які знадобляться при подальшому дослідженні та розробці.

1.1 Аналоги програмного забезпечення підтримки продуктивності

Трьома популярними застосунками-аналогами, які можна виділити серед інших, є ProofHub, Trello та Asana. Всі вони безпосередньо пов’язані з відслідковуванням та підтримкою власної продуктивності, тому варто проаналізувати їх сильні та слабкі сторони. В табл. 1.1 наведено характеристику першого сервісу – ProofHub.

Таблиця 1.1 – Характеристика ProofHub

Характеристика	**Інформація**
Розробник	Sandeep Kashyap
Архітектура	Client-server
Мова реалізації	Мова програмування – php, БД – MySQL, вебсервер – nginx. Використовується ряд JS-бібліотек, серед яких Preact, LazySizes та jQuery. CMS – WordPress, з CDN можна виокремити Cloudflare, jsDelivr та cdnjs.

Продовження таблиці 1.1

Характеристика	**Інформація**
Перелік функцій та характеристик	– створення персональних завдань та можливість делегування різним учасникам команди; – модуль тайм-трекінгу для зручного керування робочим часом через таймшити, таймери, оцінку часу та фільтри; – вбудовані звіти про витрачений час, статус виконання завдань, використання ресурсів; – дошки Kanban, що легко масштабуються та налаштовуються; – створення шаблонів для швидкого запуску та перегляду проєктів; – інтеграції з популярними хмарними сховищами (Google Drive, Box, Dropbox, Onedrive); – різні формати відображення даних: таблиці, дошки, діаграми Ганта, календар та логи активності.
Джерело інформації	www.proofhub.com

Окрім основних характеристик для ProofHub можна виділити ряд переваг та недоліків, які наведено в табл. 1.2.

Таблиця 1.2 – Переваги та недоліки ProofHub

Переваги	**Недоліки**
Повний контроль над командами та проєктами	Відволікаючі повідомлення.
Простий для розуміння інструмент, який не потребує великої кількості часу для навчання користування функціоналом.	Отримання відповіді від служби підтримки клієнтів може зайняти багато часу.
Індивідуальні робочі процеси та призначення завдань, гарантуючи, що кожен знає, що робити.	Ціни можуть бути доволі високими, особливо для малого бізнесу та стартапів.
Гнучкі інструменти проєктування.	UI/UX потребує вдосконалення.

Інтерфейс вебзастосунку ProofHub наведено на рис. 1.1.

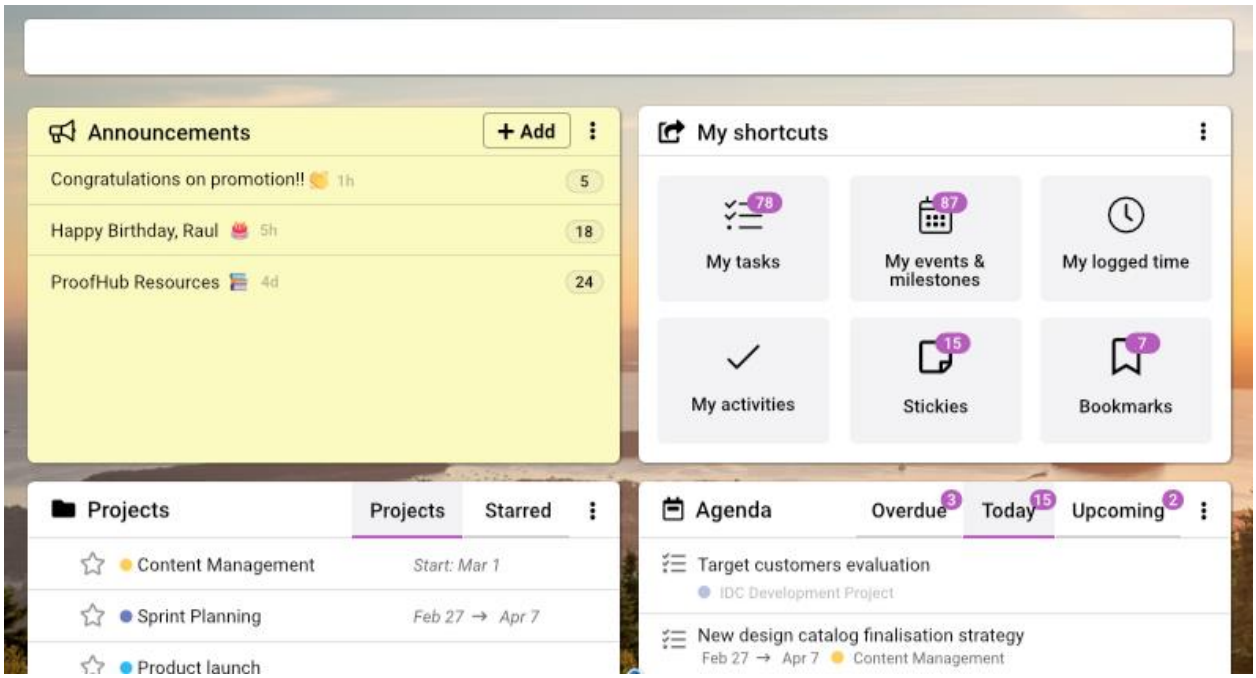

Рисунок 1.1 – Інтерфейс ProofHub

Наступним аналогом є популярний вебзастосунок Trello, характеристики якого додано в табл. 1.3.

Таблиця 1.3 – Характеристика Trello

Характеристика	**Інформація**
Розробник	Fog Creek Software
Архітектура	Client-server
Мова реалізації	У якості мови програмування використовуються Node.js та Backbone.js, frontend-частина – React; у якості бази даних використовується MongoDB
Перелік функцій та характеристик	– дошки – робочий простір, що може бути створений для різних проєктів або завдань; – створення списків для групування карток за різними етапами чи категоріями; – картки як окремі завдання чи елементи проєкту; – можливість додавати коментарі та вести обговорення в картках; – функціонал прикріплення файлів для зберігання додаткових матеріалів або ресурсів, що пов’язані з завданням; – створення чек-листів; – задання термінів виконання завдань з відповідними повідомленнями; – додання учасників у проєкт з налаштуванням прав доступу.
Джерело інформації	trello.com

Після виокремлення основної інформації слід перейти до переваг та недоліків, які наведено в табл. 1.4.

Таблиця 1.4 – Переваги та недоліки Trello

Переваги	Недоліки
Простий та інтуїтивний інтерфейс.	Обмежена функціональність у безкоштовній версії, може бути занадто простим для складних проєктів.
Безкоштовний план, доступний для користувачів.	Відсутність деталізованих засобів аналізу та звітності.
Гнучкість у налаштуванні завдань та проектів.	Відсутність вбудованих інструментів для відстеження часу, автоматизації.

Як і було зазначено, інтерфейс є доволі зрозумілим для застосунку подібної тематики. Його відображено на рис. 1.2.

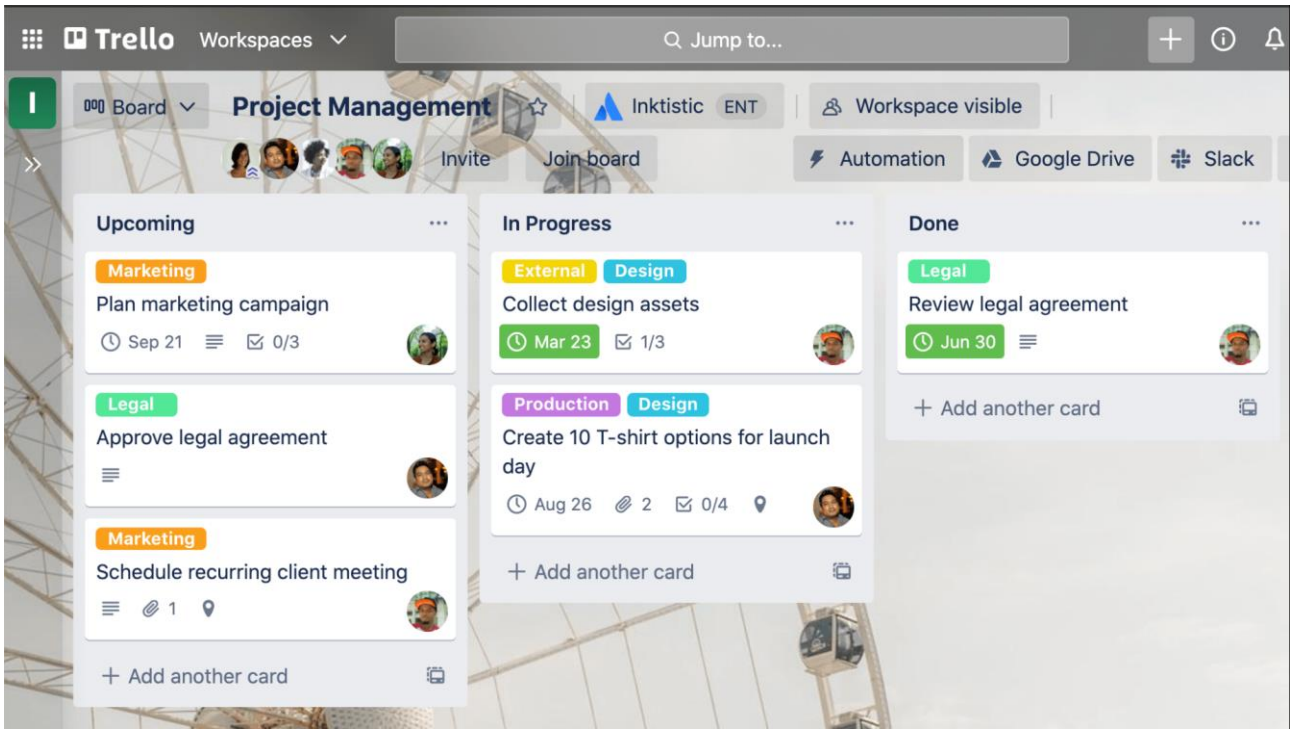

Рисунок 1.2 – Інтерфейс Trello

Останнім аналогом є Asana, характеристики якого вказані в табл. 1.5.

Таблиця 1.5 – Характеристика Asana

Характеристика	**Інформація**
Розробники	Dustin Moskovitz, Justin Rosenstein
Архітектура	Client-server
Мова реалізації	Мова програмування – JS з Node.js, веб-фреймворк та веб-сервер – Next.js; фронтенд-частина – React
Перелік функцій та характеристик	– створення завдань для себе та інших членів команди, вказуючи всю необхідну інформацію; розбиття на підзавдання та налаштування завдань; – створення проєктних дошок з багатим функціоналом для кооперації; – відображення завдань та подій у вигляді календаря, що допомагає легко визначати терміни та графік робіт; – можливість встановлювати терміни виконання завдань та призначати їх користувачам; – фільтри та розширений пошук; – генерація звітів та аналітики щодо ходу проектів та продуктивності команди; – застосунки та інтеграції з багатьма сервісами по типу Google Drive та Dropbox.
Джерело інформації	asana.com

Asana також має ряд переваг та недоліків, які виокремлено в табл. 1.6.

Таблиця 1.6 – Переваги та недоліки Asana

Переваги	**Недоліки**
Простий та інтуїтивний інтерфейс.	Багато функцій доступні тільки в платних версіях, що може обмежувати малі команди чи організації
Персоналізація вигляду завдань – канбан-дошки, списки, графіки та таблиці.	Велика кількість функцій, яка може заплутати початківців.
Зручна система коментарів та обговорень для комунікації в команді.	Відсутність вбудованої функції відстеження часу.

Інтерфейс застосунку можна побачити на рис. 1.3.

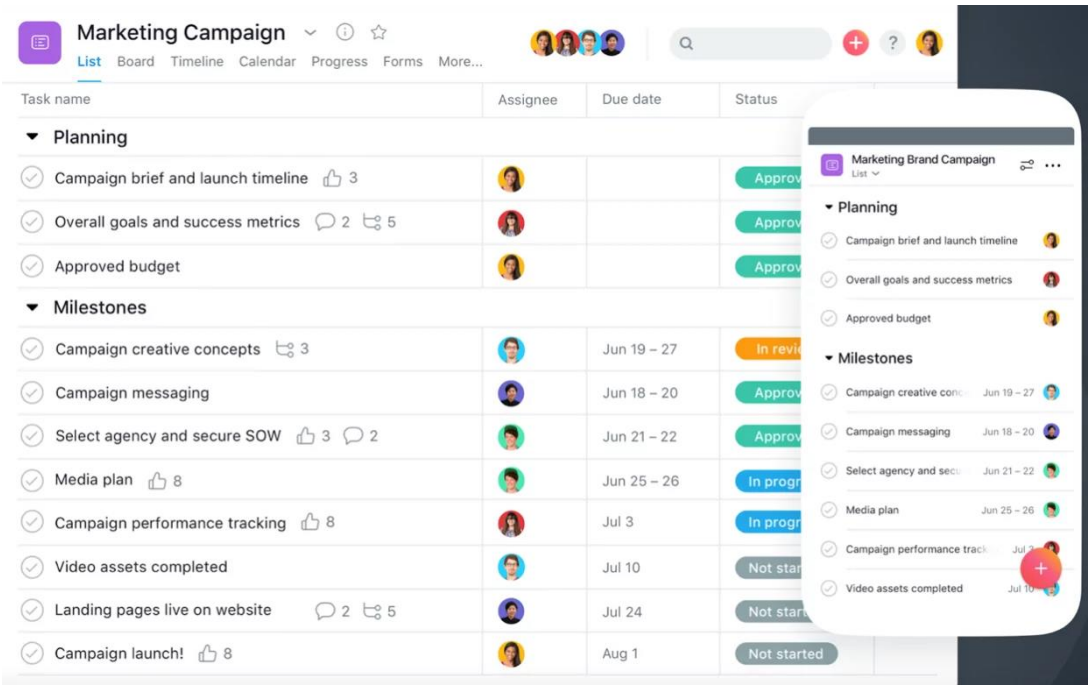

Рисунок 1.3 – Інтерфейс Asana

Порівнявши всі три проєкти, можна побачити ряд аналогічних функцій, які вони надають, а також виділити унікальні та проаналізувати, чи приносять вони якусь користь та чи сприяють гарному користувацькому досвіду.

1.2 Специфікація вимог програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням застосунку є автоматизація процесу відслідковування та підтримки продуктивності ІТ-фахівців.

Погодження, що ухвалені в програмній документації

Для створення програмного забезпечення використовується ОС Windows 11, мова програмування Python та її вебфреймворк Django.

Погоджено, що для розробки API додатково використовуватиметься Django Rest Framework, а для фронтенд-частини – JavaScript та бібліотека jQuery.

Межі проєкту ПЗ

Крайньою датою завершення роботи над ПЗ є 05.06.2024.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Сфера застосування цього ПЗ може бути будь-якою – від використання для налагодження робочих процесів до введення дисципліни у повсякденне життя.

Цільовою аудиторією проєкту є ІТ-працівники, проте функціонал застосунку не обмежується для використання лише ними.

Характеристики користувачів

Користувач повинен мати працюючий пристрій (смартфон, планшет, ноутбук чи ПК) та доступ до мережі Інтернет.

Загальна структура і склад системи

Система складається з вебсерверу, БД, фронтенд-частини, бекенд-частини та окремого API.

Загальні обмеження

Система обмежена для користувачів без доступу до мережі Інтернет.

ФУНКЦІЇ СИСТЕМИ

Функція створення задачі.

Функція передбачає створення задачі з рядом параметрів – назвою, описом, терміном виконання, пріоритетом та поточної дошки в Kanban, після чого задача з'являється на окремій сторінці.

Вхідна інформація: назва, опис, пріоритет (низький, середній, високий), кінцевий термін виконання. Вихідна інформація – створена задача.

Функціональними вимогами є авторизований користувач та наявність таблиці задач в БД.

Функція запуску таймеру Pomodoro.

Функція передбачає вибір однієї зі створених робочих сесій (або створення за відсутності), вказання завдань на відповідний часовий проміжок та запуск таймеру за натисканням на кнопку. Таймер автоматично змінює «етапи» сесії з роботи на відпочинок та відображає поточний статус.

Вхідна інформація: робоча сесія, список завдань. Вихідна інформація – запущений таймер, оновлена історія використання таймеру Pomodoro.

Функціональною вимогою є авторизований користувач, до якого прив’язана існуюча робоча сесія.

Функція створення проєкту.

Функція передбачає створення проєкту для спільної роботи з іншими користувачами після зазначення ряду даних.

Вхідна інформація: назва, опис. Вихідна інформація: повідомлення про успішне створення проєкту та переадресація на його сторінку.

Функціональною вимогою є авторизований користувач без обмеження на кількість активних проєктів або з їх малою кількістю.

Функція запрошення користувача до проєкту.

Функція передбачає надсилання запрошення до проєкту вказаному користувачу, при прийнятті якого той отримає доступ на відповідну сторінку. У разі відхилення відправник має можливість запросити користувача ще раз.

Вхідна інформація: ім’я користувача у системі. Вихідна інформація: повідомлення про успішну операцію та системне повідомлення користувачу, що отримує запрошення.

Функціональною вимогою є авторизований користувач з активним контактом – користувачем, якому надсилається запрошення.

Функція надсилання повідомлень.

Функція передбачає надсилання повідомлення конкретному користувачу або надсилання відповіді на будь-яке отримане повідомлення.

Вхідна інформація: ім’я користувача, тема повідомлення та текст. Вихідна інформація: повідомлення про успішне надсилання та переадресація на відповідну сторінку, з боку отримувача – нове непрочитане повідомлення.

Функціональною вимогою є авторизований користувач з доступом до надсилання повідомлень іншому користувачу. Відсутність доступу означає блокування відправника отримувачем або ж відповідні обмеження у налаштуваннях профілю.

Функція створення нотаток.

Функція передбачає створення персоналізованих нотаток з можливістю переміщення між різними теками за допомогою Drag & Drop. Персоналізація передбачає можливе задання кольору, назви та вибору теки.

Вхідна інформація: назва нотатки, текст та колір у форматі HEX. Вихідна інформація: повідомлення про успішно створену нотатку та переадресація на відповідну сторінку.

Функціональною вимогою є авторизований користувач.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації

Джерелом вхідної інформації системи є сам користувач та ряд даних за замовчуванням, що можуть обиратися або ж при виконанні певних умов, або ж користувачем. Він задає дані вручну на відповідних сторінках (інформація проєкту, задачі, нотатки), змінює їх та видаляє.

Вимоги до способів організації, збереження та ведення інформації

Дані організовуються та зберігаються в БД PostgreSQL, який взаємодіє з застосунком безпосередньо через RESTful API. БД є окремим Docker-контейнером, який, як і контейнер застосунку, прив’язаний до мережі з драйвером bridge.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Користувач не обмежується вибором пристрою, якщо він має можливість зайти в будь-який сучасний браузер. Для ноутбука чи ПК рекомендується обрати ОС з графічним інтерфейсом (Windows, macOS, деякі дистрибутиви Linux) та мати принаймні 2 ГБ оперативної пам’яті для забезпечення оптимальної продуктивності. Окрім цього, рекомендується мати відеокарту з підтримкою WebGL для коректного відображення вебсторінок та мультимедійного вмісту.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Система має архітектуру клієнт-сервер, що передбачає наявність ряду клієнтів, що використовують дані та функціонал з серверу. Клієнтом є браузер, що використовує вебсервер (бекенд-частину).

Системне програмне забезпечення

Фронтенд-частина ПЗ розробляється за допомогою JavaScript, jQuery та препроцесору Stylus, а для бекенд-частини – Django (вебфреймворк), Django

Rest Framework та PostgreSQL. Передача інформації з сервера на клієнт і навпаки відбувається за допомогою RESTful API, а дані зберігаються в БД PostgreSQL.

Мережне програмне забезпечення

ПЗ створено на ОС Windows 11 з використанням IDE Visual Studio Code, а також з використанням віртуальної машини для розгортання Docker-контейнерів.

Програмне забезпечення ведення інформаційної бази

Система проводить CRUD операції з PostgreSQL-базою застосунку.

Мова і технологія розробки ПЗ

Основні мови програмування, що використовуються у проєкті – Python та JavaScript з їх фреймворками (бібліотеками) Django, Django Rest Framework та jQuery відповідно. Окрім цього, використовується препроцесор стилів Stylus.

Для розгортання ПЗ застосовується Docker та docker-compose, в якому зазначено два контейнери – самого застосунку та бази даних, які здійснюють комунікацію через мережу з драйвером типу bridge.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс користувача має бути побудований на усіх вимогах UI/UX дизайну для досягнення найбільш коректного результату. Вебсайт не може мати відволікаючий інтерфейс, щоб користувачі могли фокусувати увагу лише на власних задачах.

Мокапи для розробки створено за допомогою онлайн-сервісу Figma, який надає ряд інструментів для реалізації user-friendly середовища, а шаблон має два основні елементи – панель навігації та панель з основним функціоналом сторінки. Інтерфейс є адаптивним, що означає зручність для користувачів мобільних пристроїв.

Апаратний інтерфейс

У ролі апаратного інтерфейсу виступає пристрій користувача – смартфон чи ПК, за допомогою якого він взаємодіє з вебзастосунком через браузер.

Програмний інтерфейс

Django є багатофункціональним вебфреймворком мови програмування Python, що дозволяє створювати будь-які вебсайти та широко використовується з Django Rest Framework – RESTful API-фреймворком. Django підтримує MVT-архітектуру та використовує рушій шаблонів Jinja для представлення змінних python безпосередньо на сторінках. JQuery є популярною бібліотекою JavaScript, що покращує деякий базовий функціонал та спрощує операції з DOM. Stylus є препроцесором для генерації css-файлів, що має можливість використання ряду операцій та змінних для більш гнучкої розробки дизайну вебзастосунку.

Комунікаційний протокол

Для обміну даними в мережі Інтернет використовується стек протоколів TCP/IP, а для комунікації між клієнтом та сервером – HTTP та HTTPS.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

ПЗ є доступним для усіх користувачів, які мають будь-який придатний апаратний інтерфейс, працюючий браузер та підключення до мережі Інтернет.

Супроводжуваність

Система є гнучкою та розширюваною, тому її легко діагностувати та модифікувати, при цьому забезпечуючи стабільність та мінімізацію впливу неочікуваних результатів. ПЗ має покриття тестами, що дає можливість реагувати на небажану поведінку існуючого функціоналу при певних змінах.

Переносимість

ПЗ може працювати на будь-якій ОС з графічним інтерфейсом.

Продуктивність

ПЗ не використовує значні ресурси пристрою, а швидкість його роботи залежить від швидкості підключення до мережі Інтернет.

Надійність

Система не припинятиме працювати через внутрішні дефекти, оскільки вона здатна підтримувати свій рівень функціональності за їх наявності за допомогою перехоплення помилок та їх логування. Окрім цього, система здатна до відновлення та повернення в будь-яку з попередніх робочих версій.

Безпека

Автентифікація в застосунку виконується за допомогою пошти користувача та паролю, який зберігається в БД у хешованому вигляді. Окрім цього, в БД є таблиця з інформацією про активні сеанси користувачів, що дозволяє відстежувати їх стан за допомогою серіалізованої інформації, унікального ідентифікатору сеансу та його дати й часу закінчення.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи бакалавра проаналізовано предметну область та виділено декілька аналогів до вебзастосунку відслідковування та підтримки продуктивності ІТ-фахівців – ProofHub, Trello та Asana. При цьому досліджено їх основні характеристики, функціональні особливості, переваги та недоліки, які враховуватимуться при проєктуванні власної системи.

Окрім цього, додано специфікацію вимог до програмного забезпечення, що розробляється, де зазначено призначення та межі проєкту, загальний опис, функціональність, ряд вимог (до інформаційного, технічного, програмного забезпечень та до зовнішніх інтерфейсів) і властивості. Ця специфікація допомагає окреслити суть проєкту та мати розуміння про його масштаб, а також є орієнтиром при плануванні процесу розробки.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Без виокремлення важливих проєктних рішень, тобто моделювання ПЗ, складно уявити його подальшу розробку, оскільки це є доповненням специфікації вимог та створенням деталізованого алгоритму дій. Слід визначити етапи створення проєкту, що розробляється, та описати його майбутню структуру, а також змоделювати ієрархічне представлення об’єктів через IDEF0.

Окрім цього, потрібно створити сценарії варіантів використання системи для кращого розуміння очікувань користувачів та масштабу роботи.

2.1 Етапи створення проєкту

Розробка ПЗ полягає не тільки в процесі кодування – вона передбачає багато кроків як перед, так і після цього. Наявність конкретного плану з певною послідовністю дій сприяє продуктивній роботі над застосунком, що означає використання правильних підходів та мінімізації кількості ситуацій, при яких розробник не знає, що робити далі [4]. Кожен з кроків розділяється на задачі меншого об’єму, а при бажанні вказується пріоритет – таким чином в першу чергу можна зайнятися найважливішими завданнями. Окрім цього, від подібного планування залежить тривалість розробки проєкту, оскільки з’являється розуміння про складність задач та можливі витрати часу [5].

Саме етапи, в яких полягає реалізація застосунку, можуть варіюватися в залежності від проєкту та команди розробників, проте для створення вебзастосунку відслідковування та підтримки продуктивності ІТ-фахівців використовуватимуться наступні кроки:

– збір інформації та планування концепції майбутнього продукту, написання списку задач з їх пріоритетом та складністю;

– проєктування та моделювання системи;

– розробка інтерфейсу користувача;
– вибір технологій для розробки;
– написання коду;
– розгортання та тестування застосунку;
– реліз першої версії продукту.

Перший крок передбачає не тільки виокремлення мети проєкту та переліку очікуваного функціоналу, але й формулювання конкретних цілей, визначення актуальності продукту та окреслення проблеми, що він вирішуватиме. Варто зазначити, що розроблені на початковому етапі задачі не є кінцевими і будуть змінюватися та доповнюватися впродовж життєвого циклу проєкту.

На етапах проєктування та моделювання велику роль грають діаграми (класів, активностей, розгортання, станів і подібні), оскільки вони графічно відображають необхідну для розробників інформацію.

Окрім цього, потрібно вирішити низку важливих для ПЗ питань, щоб мати краще уявлення про застосунок і зібрати більше інформації [6]. Питання, що підлягають аналізу:

– проблеми, які вирішує ПЗ;
– переваги серед аналогів;
– перелік основного функціоналу системи;
– технології та інструменти для розробки ПЗ;
– ролі, права та обмеження користувачів застосунку;
– цільова аудиторія проєкту [7];
– очікуваний результат розробки.

Етап розробки інтерфейсу зазвичай передбачає використання спеціалізованої програми, серед яких для створення mock-ups використано Figma. Після вибору технологій розробляються back-end та front-end частини

проєкту, які тестуються та у випадку відсутності помилок можуть бути розгорнуті у продакшн-середовищі.

2.2 Створення сценаріїв Use Case

Створення варіантів сценаріїв використання допомагає при розробці функціоналу застосунку, оскільки в них описано не тільки очікуваний результат після певного алгоритму дій, але й необхідні умови, перешкоди та розширення [8]. В таких сценаріях описуються дії головного актора, тому всі ефекти від виконання дій мають бути однаковими для користувачів подібного типу [9].

Нижче описано три варіанти сценаріїв використання для окремих випадків – короткий usecase для створення робочого графіку, поверхневий – для запуску таймера за системою Pomodoro, та повний (табл. 2.1) для опису дії запрошення користувача у проєкт.

Короткий сценарій передбачає невеликий опис, поверхневий доповнюється альтернативними варіантами розвитку подій, а повний супроводжується рядом характеристик, що максимально деталізують сценарій та окреслюють усі можливі аспекти.

Короткий usecase – «Створення графіку роботи».

Користувач успішно авторизується у системі, використовуючи свої дані, та переходить на сторінку редагування графіків роботи через відповідну кнопку, що знаходиться на головній сторінці. Користувач натискає на кнопку «Створити новий графік» та отримує форму для введення назви та періоду, після чого редагує значення налаштувань приватності. Користувач прикріплює створений Google-календар, що інтегрується з системою, або створює його самостійно з наявним у системі інструментарієм. Користувач натискає на кнопку «Створити» та підтверджує створення нового графіку роботи, після чого перенаправляється на сторінку його редагування.

Поверхневий usecase – «Запуск таймера за системою Pomodoro».

Головний сценарій: Користувач успішно авторизується у системі, використовуючи свої дані, та переходить на сторінку «Pomodoro Timer» за допомогою однойменної кнопки в головному меню. Користувач натискає на один зі створених раніше робочих сесій та бачить спливаюче вікно з детальною інформацією про таймер – назву, опис, час для роботи, час для відпочинку, кількість повторів, дату та час останнього запуску та посилання для переходу до аналітики з використанням цього таймеру. Користувач натискає на кнопку «Почати сесію» та потрапляє на сторінку зі спрощеним інтерфейсом, запущеним таймером, чергою кіл роботи та відпочинку, а також полем для вводу можливих задач на робочу сесію. Користувач продовжує чи зупиняє робочу сесію.

Альтернативні сценарії:

– користувач не увійшов у систему, тому він буде переадресований на сторінку авторизації;

– користувач вже має запущений таймер і не може запустити новий, при цьому він отримає відповідне повідомлення з помилкою;

– користувач не може запустити таймер, тому що не створював сесії;

– у системі стався технічний збій та дані створених робочих сесій не можуть бути отримані з бази даних.

Таблиця 2.1 – Повний usecase «Запрошення користувача до команди»

Назва	**Характеристика**
Use Case Name	Запросити користувача до команди
Scope	Система відслідковування та підтримки продуктивності
Level	Мета користувача
Primary Actor	Користувач (розробник)

Продовження таблиці 2.1

Stakeholders and interests	– користувач – засновник створюваного проєкту, та майбутній член команди – запрошений;
Preconditions	– користувач з достатніми у проєкті правами авторизований у системі; – користувач, який приєднується до команди, авторизований у системі.
Success guarantee	– користувачі не є заблокованими; – проєкт не є заблокованим чи замороженим.
Main Success Scenario	– користувач, що має достатньо прав у проєкті, авторизується у системі; – користувач переходить на сторінку «Мої проєкти» через однойменну кнопку в меню; – користувач обирає необхідний проєкт та переходить на сторінку його редагування; – користувач натискає на кнопку «Запросити розробника» та отримує спливаюче вікно – поле для вводу імені чи електронної адреси, що мають бути в системі; – користувач-адміністратор проєкту вводить електронну адресу іншого користувача; – користувач-адміністратор проєкту обирає права, які запрошений користувач матиме у разі приєднання, а також вказує його роль; – користувач-адміністратор проєкту натискає на кнопку «Відправити запрошення»;

Продовження таблиці 2.1

Назва	**Характеристика**
Main Success Scenario	– система надсилає запрошеному користувачу внутрішнє повідомлення та лист на електронну пошту, в якому вказується певна інформація про користувача-адміністратора проєкту та його опис, а також посилання на підтвердження запрошення; – користувач переходить за посиланням та бачить більш детальну інформацію про проєкт; – користувач підтверджує приєднання до проєкту натисканням на кнопку «Приєднатися»; – система перенаправляє користувача на сторінку проєкту.
Extensions	а) користувач не може авторизуватися у системі; – користувач переходить на сторінку авторизації та вводить свої дані; – користувач натискає на кнопку «Увійти»; – система повідомляє користувачу про некоректно введену електронну адресу та/або пароль; – користувач розпочинає процес авторизації з початку; б) користувач не має достатньо прав у проєкті: – користувач переходить на сторінку проєкту; – користувач не може натиснути на кнопку «Запросити розробника»;

Продовження таблиці 2.1

Назва	**Характеристика**
Extensions	– система за допомогою спливаючого вікна повідомляє користувачу, що він не має достатньо прав для здійснення такої операції; в) користувач-адміністратор проєкту у поле пошуку вводить електронну адресу, яка не зареєстрована у системі: користувач натискає на кнопку «Запросити розробника» та отримує спливаюче вікно – поле для вводу імені чи електронної адреси, що мають бути в системі; – користувач вводить електронну адресу, обирає необхідні права та роль; – користувач натискає на кнопку «Відправити запрошення»; – система повідомляє користувача-адміністратора проєкту, що користувача з такою електронною адресою чи іменем не знайдено у системі; – користувач-адміністратор проєкту проходить процес пошуку користувача з початку;

Продовження таблиці 2.1

Назва	Характеристика
Extensions	г) користувач, якого запросили, не перейшов за посиланням для підтвердження протягом 12 годин: – користувач отримує внутрішнє повідомлення у системі та лист на електронну адресу з посиланням для підтвердження запрошення; – користувач ігнорує посилання як мінімум 12 годин; – посилання для підтвердження приєднання стає недійсним; – користувач-адміністратор проєкту отримує повідомлення про те, що запрошений користувач не підтвердив приєднання до проєкту; д) користувач відхилив запрошення до проєкту: – користувач отримує внутрішнє повідомлення у системі та лист на електронну адресу з посиланням для підтвердження запрошення; – користувач переходить за посиланням та бачить більш детальну інформацію про проєкт; – користувач натискає на кнопку «Відхилити запрошення; – користувач-адміністратор проєкту отримує повідомлення про те, що запрошений користувач відхилив це запрошення.

Продовження таблиці 2.1

Назва	**Характеристика**
Special Requirements	– користувач має бажання та погоджується доєднатися до проєкту, в який його запрошено; – користувач має стабільне Інтернет-з'єднання, щоб побачити повідомлення про запрошення у проєкт одразу ж; – користувач має доступ до електронної пошти, прив'язаної до профілю; – підтвердження про приєднання до проєкту має надійти максимум за 12 годин.
Technology and Data Variations List	– користувач-адміністратор проєкту може використати один із декількох способів запрошення – одноразове посилання, напряму через профіль користувача чи за допомогою його імені або електронної адреси; – ім'я користувача у системі та електронна пошта унікальні, тому при пошуку повної адреси або імені в результаті буде лише один точний користувач; – система може пропонувати ім'я користувачів, що вже запропоновані в системі, при початку введення після трьох символів; – через 12 годин попереднє запрошення стає недійсним і користувач-адміністратор проєкту може повторити його надсилання за необхідності.

Кінець таблиці 2.1

Назва	Характеристика
Frequency of Occurrence	Система може працювати безперервно
Miscellaneous	– необхідність в обмеженнях для приєднання до окремих проєктів; – отримання користувачем листів на електронну адресу на основі дій інших користувачів після приєднання до відповідного проєкту; – права, якими володіє користувач, що тільки приєднався до проєкту; – можливий workflow для виходу та повернення до проєкту.

Окрім цього, можна виокремити можливість створення Use Case діаграми, в якій можна описати можливі дії користувачів (акторів) та їх взаємозв’язок. Звичайний користувацький досвід передбачає наступні ролі:

– адміністратор системи (відповідає за конфігурацію та управління системою, має права доступу до всіх функцій, відповідає за безпеку та обслуговування);

– користувач (IT-спеціаліст, використовує систему для відслідковування завдань та проблем у роботі, отримує відповідні сповіщення від системи).

За наявності проєкту ролі у ньому відрізняються та налаштовуються окремо від тих, що вже призначені у вебзастосунку:

– менеджер проекту (відслідковує статуси завдань та створює нові, встановлює виконавців та визначає терміни, спостерігає за прогресом та відповідає за workflow);

– розробник (отримує завдання в системі та змінює їх статус в процесі виконання з наданням певних звітів, вносить корегування чи коментарі до поставлених задач, має зручний інструмент комунікації з менеджером та іншими учасниками проекту).

Відповідну діаграму сценаріїв використання зображено на рис. 2.1. Актори позначаються перед системою, що представлена прямокутником з назвою застосунку, та пов’язані між собою. Основний користувач може виконувати ряд дій (функцій), які можуть або включати ще одну дію (include, що передбачає автоматичне виконання певної дії внаслідок попередньої), або бути розширеними (extend, що передбачає додаткову, тобто необов’язкову дію після виконання основної).

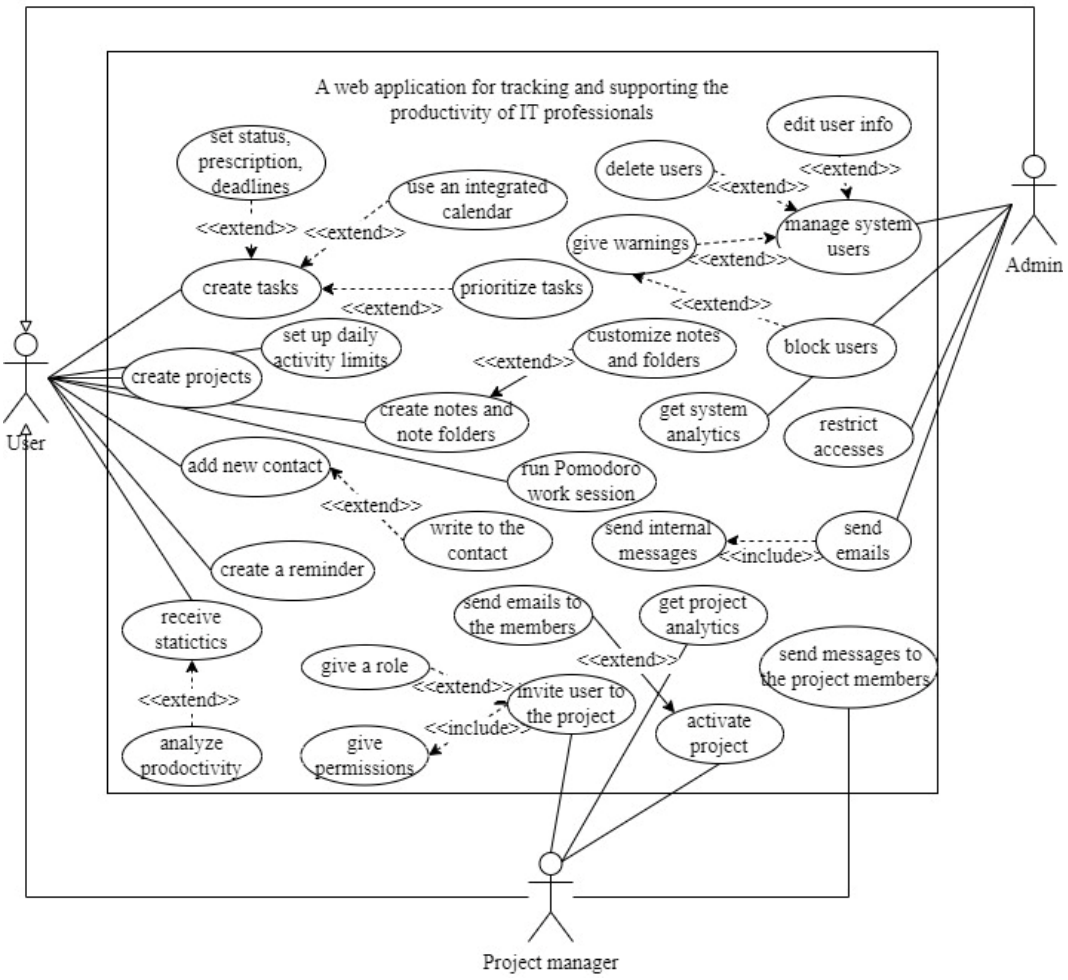

Рисунок 2.2 – Діаграма Use Case

Використання подібних сценаріїв допомагає розробникам зрозуміти різні шляхи, якими середньостатистичний користувач може піти з певною метою. Самі користувачі можуть бути пов’язані між собою – наприклад, поділяти роль чи мати відповідні привілеї для виконання специфічних дій.

Окрім цього, визначаються альтернативні сценарії, тому контролювати можливі помилки та неочікувані результати стає простіше, як і розширювати існуючі елементи, просто додаючи нову дію чи актора.

2.3 Розробка за допомогою шаблону MVT

Хоча більшість застосунків будується на основі архітектурного шаблону MVC (Model, View, Controller) [10], MVT є не менш використовуваним рішенням проєктування архітектури системи, при якому роль контролеру грає View, а саме представлення замінюється на шаблон (template) [11]. Моделі в такому випадку грають ролі інтерфейсів зі збереженими даними з певною логічною структурою (класи з атрибутами та методами), а представлення використовують ці дані для отримання у шаблонах. На рис. 2.2 зображено принцип роботи такого архітектурного рішення.

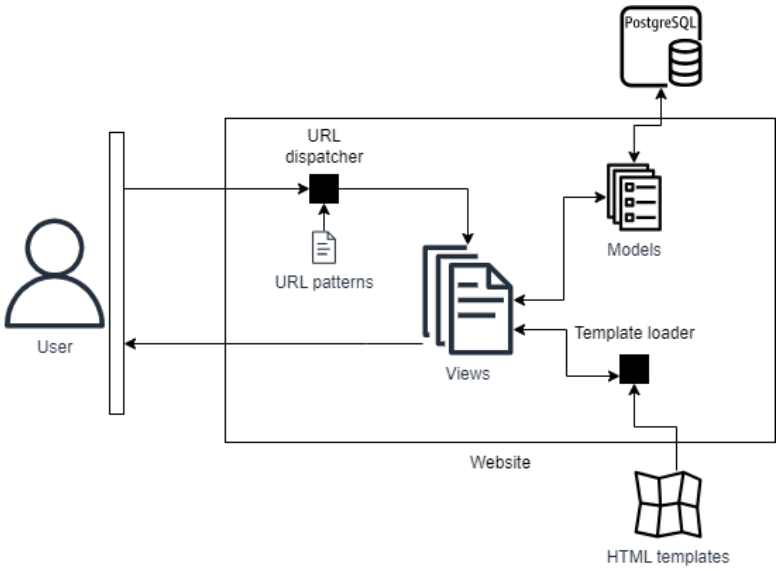

Рисунок 2.2 – Структура MVT

Користувач надсилає запит на сервер і застосунок обробляє посилання, порівнюючи його з існуючими варіантами (архітектура Django), після чого або ж переходить до представлення, або ж показує користувачу помилку у разі відсутності відповідного патерну. В представленні збираються дані, що будуть використовуватися у шаблоні (у тому числі й об’єкти з БД), та підтягується сам шаблон, повертаючи відповідь у вигляді окремої сторінки [12].

Такий підхід дає змогу розділити обов’язки у коді та гарно його організувати, відокремлюючи логіку від представлення та забезпечуючи гнучкість і масштабованість застосунку. Окрім цього, організація коду та файлів дає можливість виділити окремі компоненти ПЗ, елементи в яких можна логічно згрупувати, що можна побачити на рис. 2.3. Такі частини як задачі, повідомлення, проєкти та нотатки мають власні класи, та поведінку об’єктів, тому створювати усі моделі в одному файлі було б поганим архітектурним рішенням.

Також створено окремі компоненти – директорії для статичних елементів, медіа-файлів, html-шаблонів та стилів, які описано в налаштуваннях (settings) для зручної організації та спрощеного доступу.

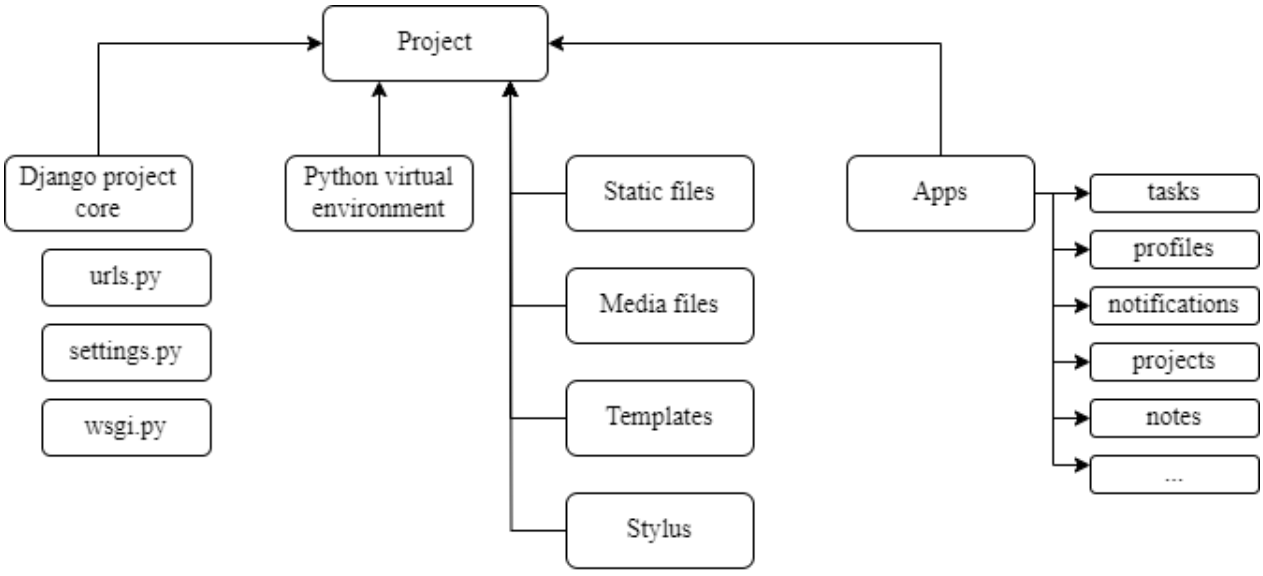

Рисунок 2.3 – Структура системи

Архітектура застосунку не лише сприяє комфортній розробці, гнучкості та масштабованості проєкту, але й попереджує помилки при взаємодії окремих компонентів між собою. Таке ПЗ простіше обслуговувати, а новим розробникам у випадку приєднанні до команди стане зрозуміліше не тільки орієнтуватися у великих проєктах, а й організовувати код самостійно.

2.4 Розробка функціональних моделей

При проєктуванні системи важливо приділити достатню увагу бізнес-процесам, що складатимуть основну частину функціональності. Для цього варто використовувати методологію функціонального моделювання IDEF0, яка передбачає представлення певних процесів у графічній формі з певною ієрархією, яка деталізує та полегшує розуміння предметної області. З допомогою цієї нотації можна спрощувати складні технічні системи та взаємодію їх компонентів, даючи змогу розробникам рухатися з нижніх рівнів ієрархії.

На рис. 2.4 відображено верхній рівень діаграми функціональної моделі запрошення користувача до проєкту нотації IDEF0.

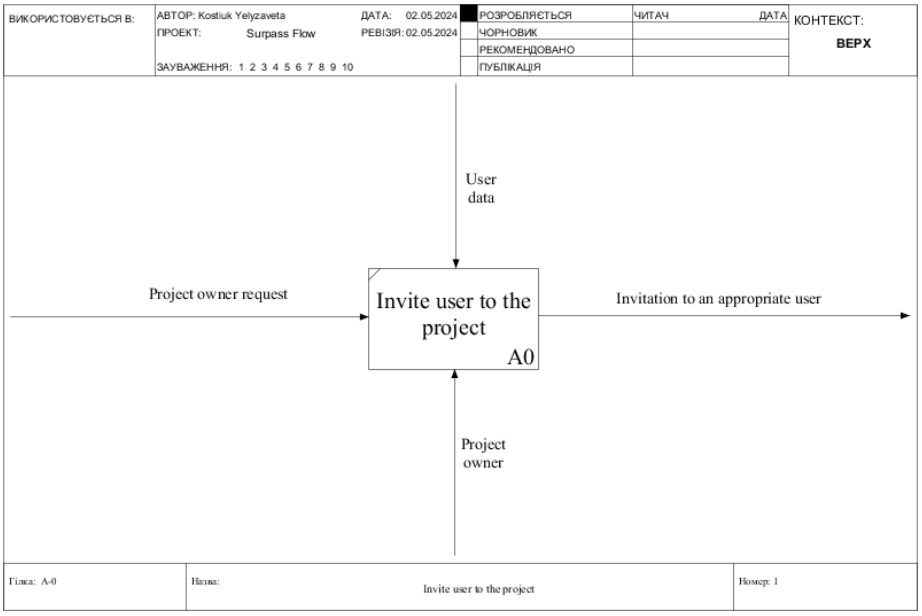

Рисунок 2.4 – Верхній рівень IDEF0

На цьому зображенні можна побачити чотири типи стрілок – вхід (input), яка зображується зліва від компоненту та позначає інформацію, яка обробляється в бізнес-процесі; управління (control), яке входить у верхню частину компоненту та напряму впливає на процес і позначає правила та стандарти; вихід (output) виходить з правої частини компоненту в напрямку до кінця діаграми та позначає результат виконання бізнес-процесу, а механізм (mechanism) означає ресурси чи виконавця процесу, відображаючись на діаграмі стрілкою з нижньої сторони компоненту.

Кожна модель може мати декілька рівнів, саме вони і є окремими діаграмами з власними описами. На рис. 2.5 зображено більш деталізовану модель другого рівня, яка описує функцію всередині об’єднаного процесу запрошення користувача до проєкту.

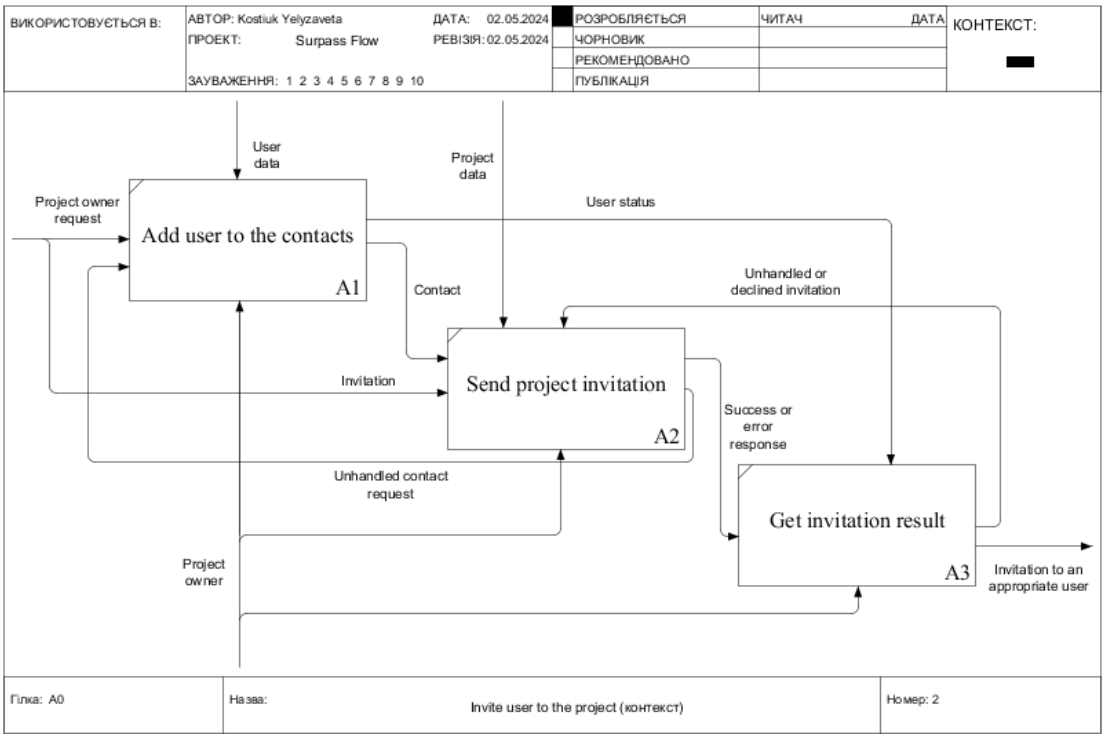

Рисунок 2.5 – Другий рівень IDEF0

На зображенні можна побачити, що після переходу з верхнього рівня виокремлено три основні функції, які виконуються при запрошенні користувача.

Моделювання в методології IDEF0 дозволило зібрати інформацію та проаналізувати ПЗ, що розробляється, і подати його у вигляді графічної моделі для подальшого використання. Окрім цього, процес моделювання допомагає краще зрозуміти та провести декомпозицію функціоналу, що є ключовим елементом для розробки системи. Методологія IDEF0 сприяє чіткому визначенню та документуванню всіх процесів і їх взаємозв'язків, що дозволяє створити цілісну картину функціонування системи.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи бакалавра змодельовано систему, що розробляється, для зручності при подальшій розробці. Описано етапи створення проєкту та розроблено ряд сценаріїв використання, серед яких короткий варіант, поверхневий та повний, які описували алгоритми створення графіку роботи, запуску таймера Pomodoro та запрошення користувача у проєкт відповідно.

Окрім цього, описано структуру застосунку з огляду на розробку за допомогою структури MVT, а також створено діаграми в нотації IDEF0 для опису основних процесів та представлення об’єктів. Застосування IDEF0 призводить до підвищення ефективності розробки ПЗ, зменшення витрат часу і ресурсів, а також покращення якості кінцевого продукту.

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА АРХІТЕКТУРИ ПРОЄКТУ

Після етапу моделювання та планування процесу розробки проєкту слід приділити значну увагу програмним аспектам та розробити ряд основних UML діаграм, які спроєктують кожен компонент системи. Це потрібно не лише для спрощення роботи, але й для попередження можливих пропусків чи навіть помилок при створенні важливих підсистем.

До того ж, розробляти вебзастосунок без попередньо підготованого інтерфейсу складніше і може призвести до постійних редагувань вже створеного варіанту, оскільки враховуються не всі особливості. Після цього потрібно завершити етап проєктування вибором та оглядом стеку технологій, що будуть використовуватися при подальшому кодуванні системи.

3.1 Створення UML діаграм

UML (Unified Modeling Language) є стандартизованою мовою для розробки діаграм, що активно використовується командами розробників для організації матеріалів під час роботи над проєктами. Побудову таких діаграм варто почати з діаграми розгортання – цей тип відображає структуру середовища, за допомогою якого система буде розгортатися, тобто його фізичне обладнання.

На рис. 3.1 відображено діаграму розгортання вебзастосунку.

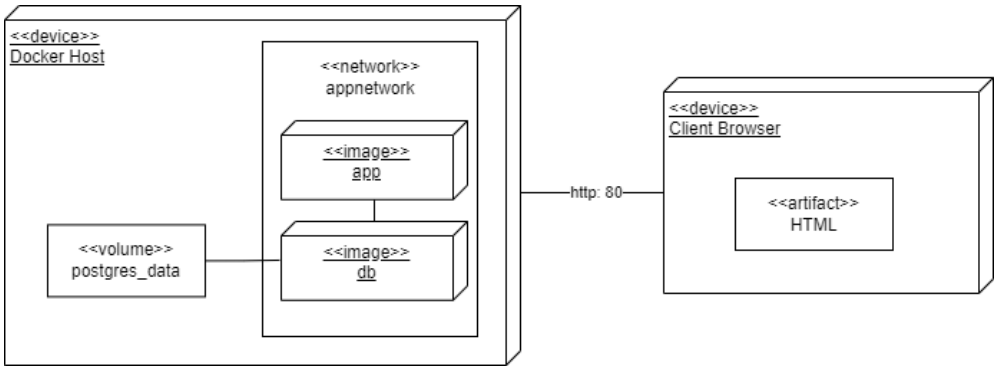

Рисунок 3.1 – Діаграма розгортання

Застосунок розроблятиметься та підійматиметься за допомогою Docker-контейнерів, що надає ряд переваг, серед яких:

– просте масштабування застосунку;
– швидке тестування та розгортання;
– великий простір для персоналізованих налаштувань;
– повна ізольованість від операційної системи;
– оптимізація використання ресурсів.

Відповідно, контейнер app відповідає і за back-end, і за front-end частину, db – за базу даних, при цьому використовуючи volume postgres_data для зберігання даних. Для відображення майбутніх підсистем з відокремленою логікою розроблено діаграму пакетів (рис. 3.2), що включає пакети класів з їх залежностями один від одного.

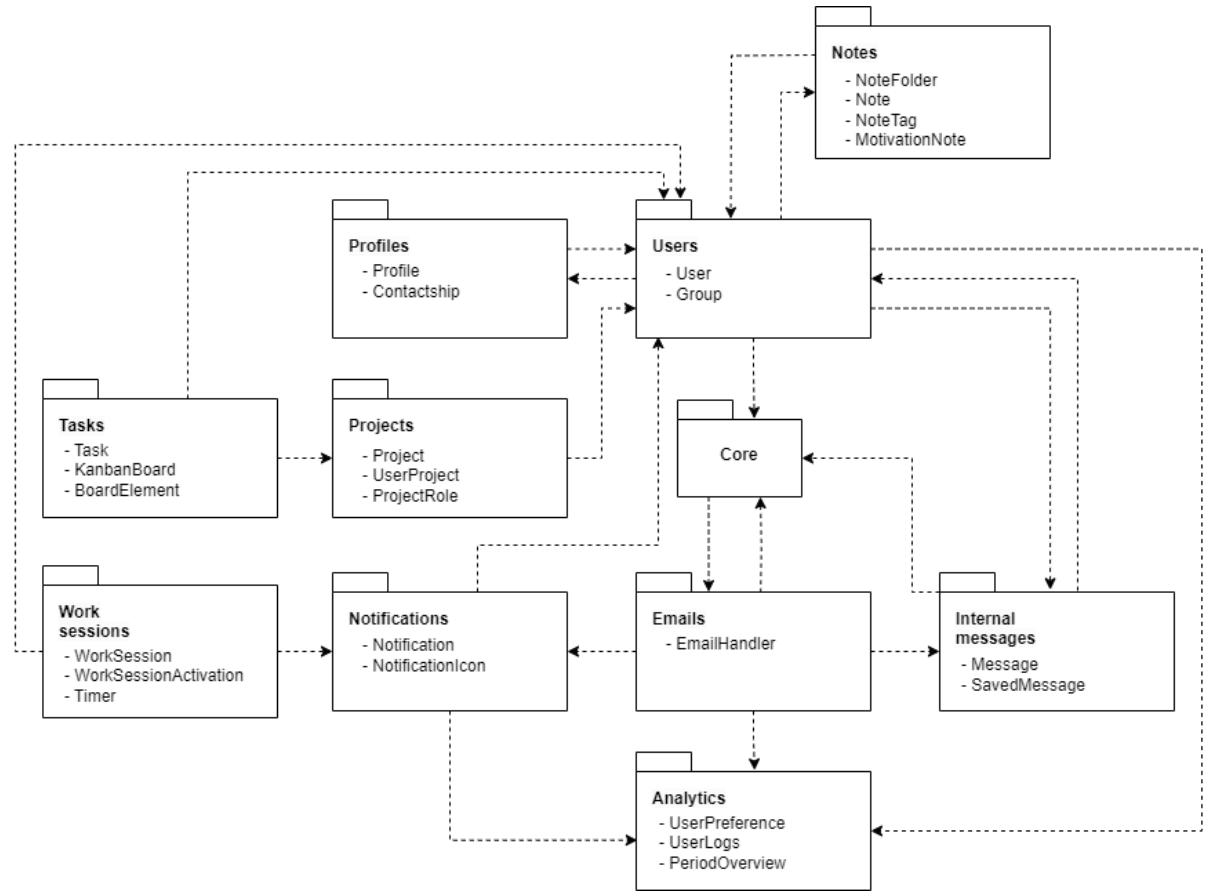

Рисунок 3.2 – Діаграма пакетів

Залежні класи відображають те, що зміни в одному вплинуть на інший – саме за допомогою цієї діаграми можна з легкістю відслідковувати та попереджувати небажані результати та помилки при імпорті одного функціоналу до іншого. Всього додано 11 окремих підсистем з чітко окресленою логікою.

Наступним кроком є діаграма класів (рис. 3.3), що показує класи, які можна побачити в описаних пакетах.

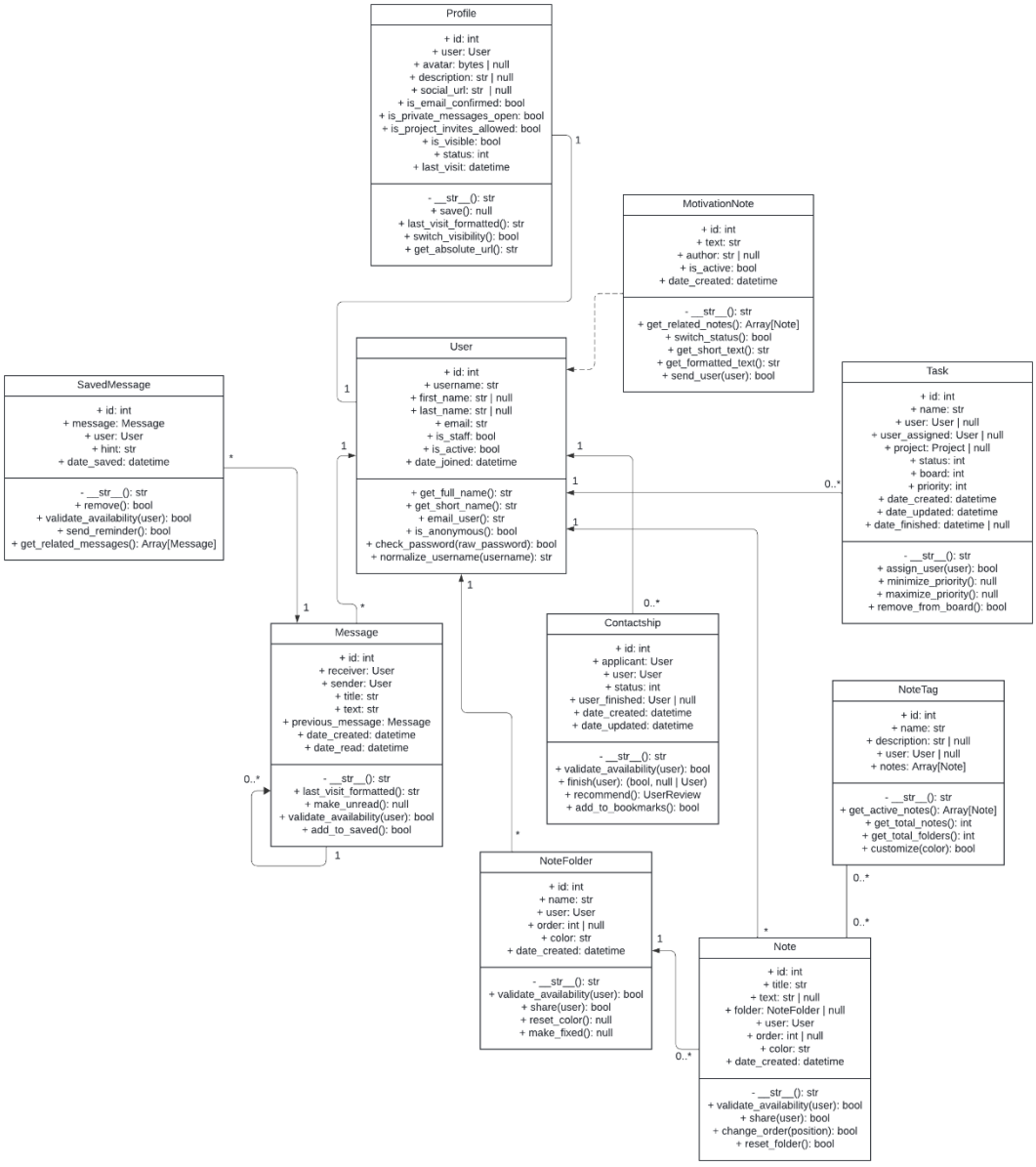

Рисунок 3.3 – Діаграма класів

В цій діаграмі зазначено класи з їх атрибутами та методами, що використовуються в застосунку. І атрибути, і методи мають зону видимості (публічні, приватні, захищені) та тип значення, а в других може бути список параметрів, що використовуються всередині.

Також всі класи пов’язані один з одним та мають такі зв’язки як «один-до-одного», «багато-до-багатьох» або ж «один-до-багатьох». Прикладом першого зв’язку є об’єкт користувача та профіль – він може існувати лише в одному екземплярі для кожного зареєстрованого користувача. «Багато-до-багатьох» зазвичай реалізується через додаткову таблицю в БД, де двома з полів є ідентифікатори пов’язаних об’єктів [13]. Зв’язок «один-до-багатьох» використовується тоді, коли певний об’єкт може мати ряд пов’язаних елементів, наприклад, тека може мати багато нотаток, проте нотатки не можуть мати інших тек. Схема бази даних наведена на рис. 3.4.

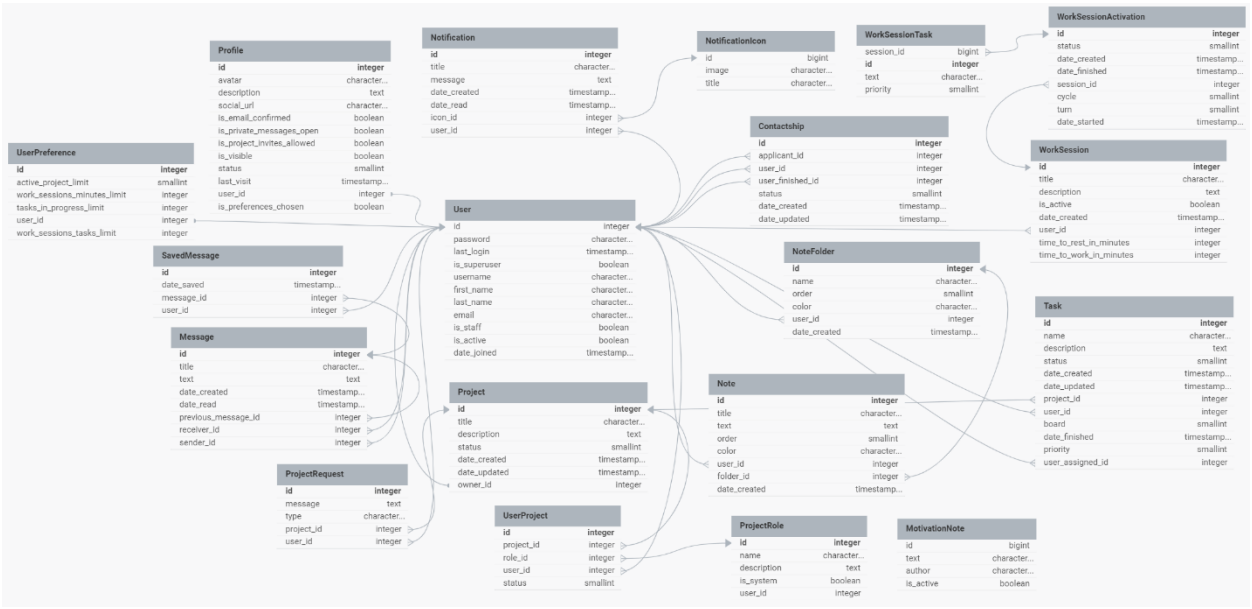

Рисунок 3.4 – Схема БД

Наступним кроком є розробка діаграм послідовності – випадком діаграм взаємодії, що забезпечують опис зв’язку елементів системи між собою в

процесі певних дій. На рис. 3.5-3.6 відображено діаграми послідовності для створення проєкту та запрошення у проєкт відповідно.

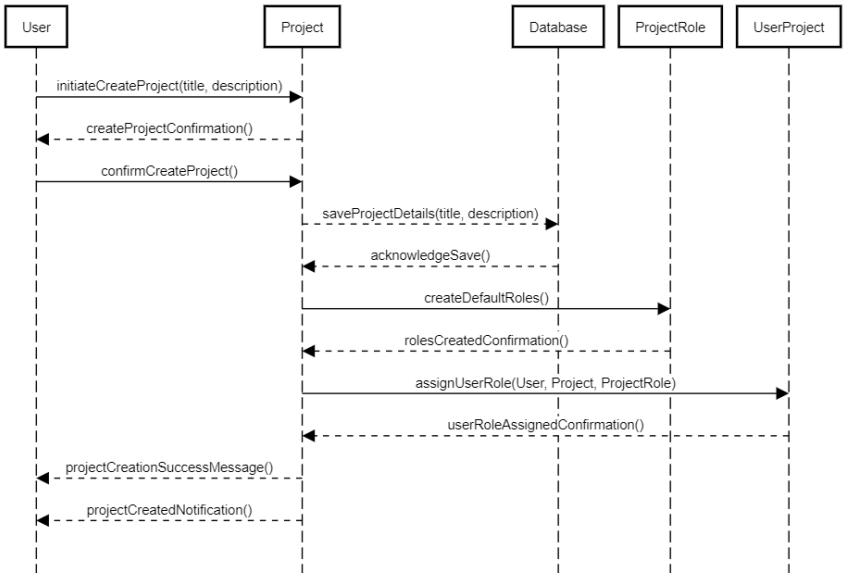

Рисунок 3.5 – Діаграма послідовності для створення проєкту

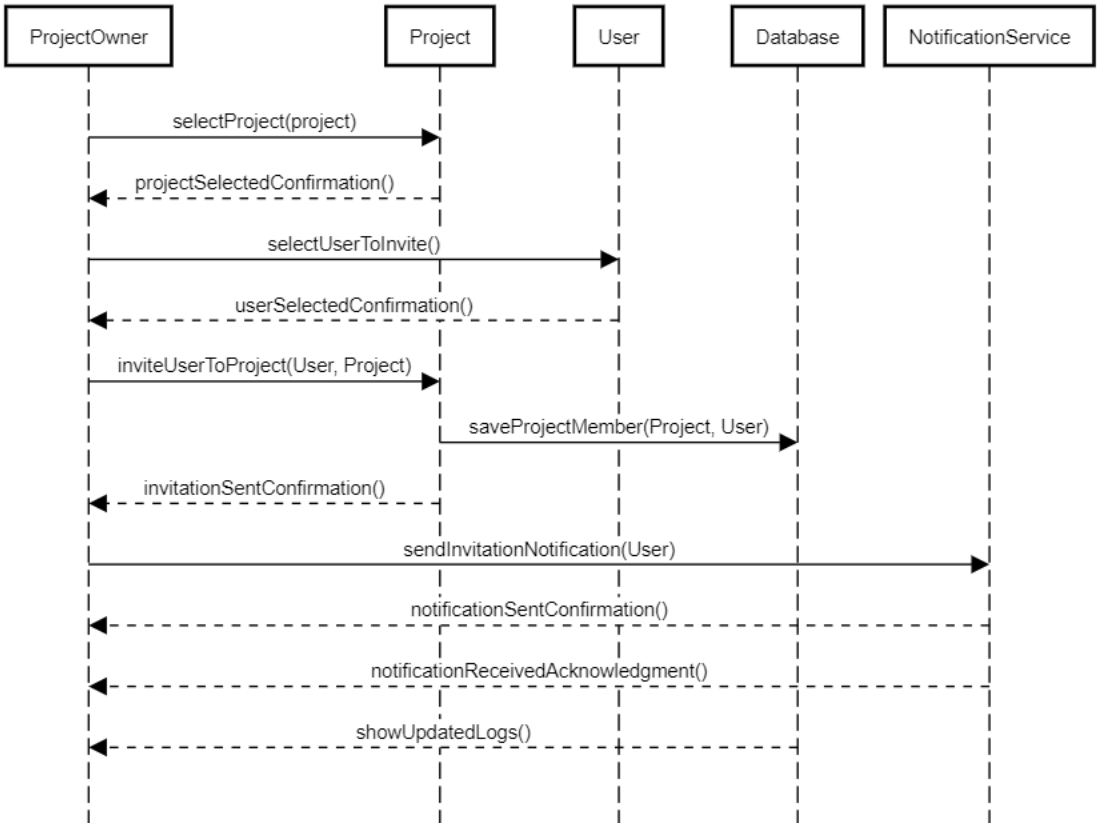

Рисунок 3.6 – Діаграма послідовності для запрошення в проєкт

На обох діаграмах відображені логічні сутності, тобто об’єкти, що взаємодіють між собою, а час роботи програми (час, протягом якого виконуються певні операції) позначено вертикальними пунктирними лініями. Горизонтальні лінії позначають операцію, а напрям стрілки – ініціатора та об’єкт-отримувач. Тобто, стрілки зліва направо називають прямими і вони відсилають повідомлення об’єкту-приймачу, а при поверненні повідомлення вони переходять у зворотні. Пунктирні лінії відображають повідомлення з результатом, звичайні ж його не повертають.

Далі потрібно розробити діаграми станів, що показують деякий сценарій при виконанні алгоритму дій, а також альтернативні сценарії з їх результатами. На рис. 3.7-3.8 відображено діаграми станів додавання користувача в проєкт та переходу з головної сторінки проєкту на інші відповідно.

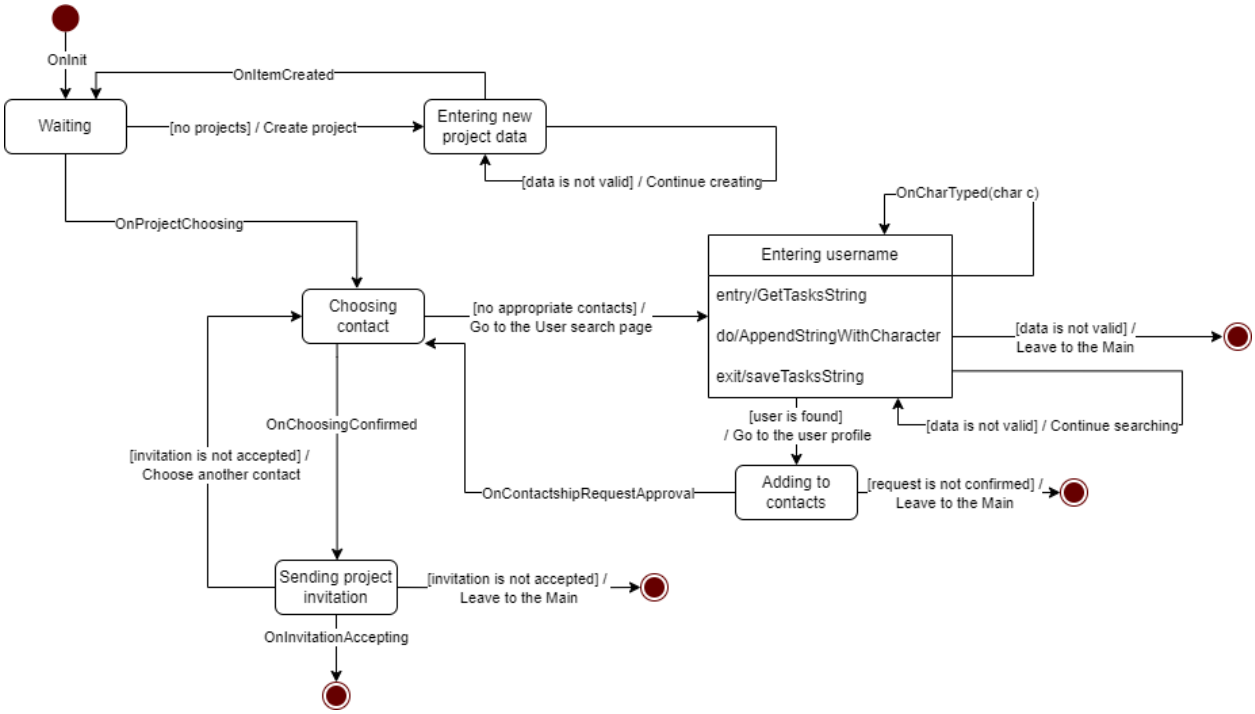

Рисунок 3.7 – Діаграма станів додавання користувача в проєкт

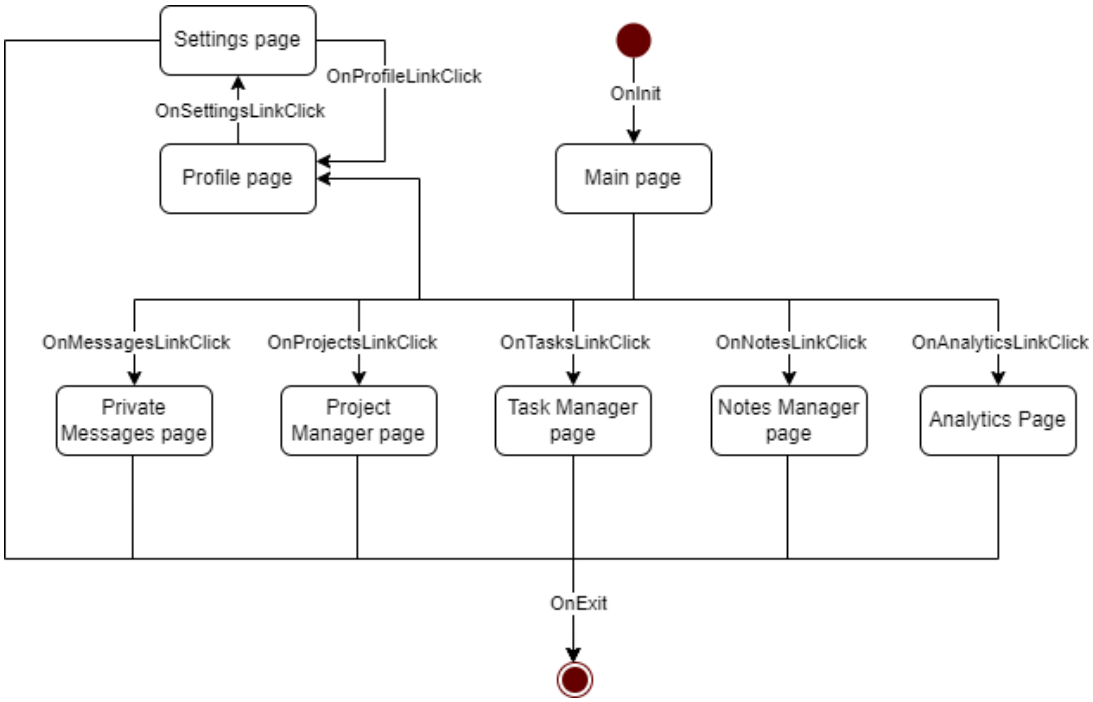

Рисунок 3.8 – Діаграма станів для переходу з головної сторінки на інші

Станами є сутності, що представляють логічні об’єкти для моделювання певного процесу чи дії. Діаграму також можна описати як граф, де ці стани є вершинами, а переходи (стрілки) є подіями, що призводять до зміни стану на інший. При цьому для переходу може вказуватися не лише назва дії, але й тригер, параметри події та граничні умови для здійснення операції. До того ж, можна виділити два види переходів – звичайний та рефлексивний, де другий відрізняється тим, що переходить в цей же стан при події.

Останнім етапом є розробка діаграми компонентів (рис. 3.9), що дозволяє виокремити сукупності елементів з розроблених підсистем та організувати їх як частини застосунку, бізнес-логіки, представлення та доступу до бази даних.

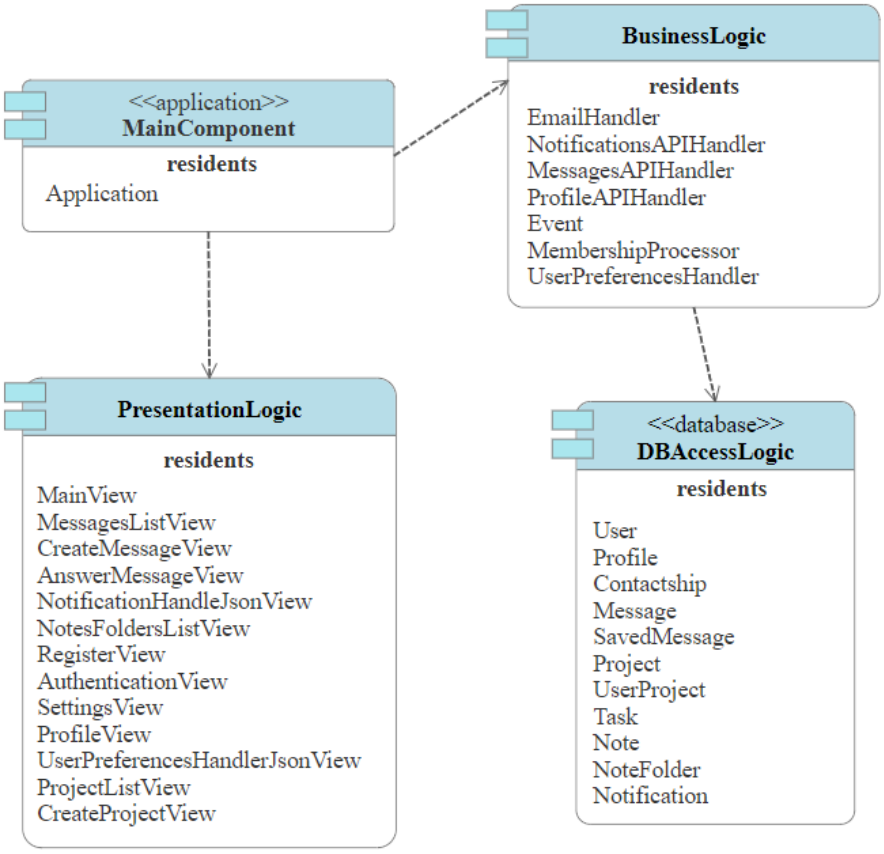

Рисунок 3.9 – Діаграма компонентів

Головний компонент зі стереотипом Application має єдиний клас, що централізує інші основні елементи. Таким чином застосунок простіше масштабувати та підтримувати, а також попереджувати та відслідковувати можливі помилки.

3.2 Розробка мокапів

Інтерфейс користувача грає велику роль для забезпечення успіху проєкту, оскільки це напряму впливає на комфорт користувачів та перехід на інші сторінки. Незрозумілий дизайн створить багато проблем, з якими багато хто не впорається і буде надавати перевагу варіанту з пошуком альтернативного застосунку.

UX/UI є дуже популярним підходом до створення дизайну [14], серед принципів якого можна виділити наступні:

– достатня простота та зрозумілість для всіх типів користувачів;
– спрямованість на цільову аудиторію;
– адаптованість для використання людьми з обмеженнями;
– наявність попереджень та зворотного зв’язку;
– відсутність перенасичення інформацією;
– адаптивність для різних екранів.

Далі представлено ряд мокапів, які було розроблено за допомогою Figma та графічного редактору Paint Tool SAI. На рис. 3.10 відображено дизайн сторінки реєстрації, а на рис. 3.11 – головної сторінки.

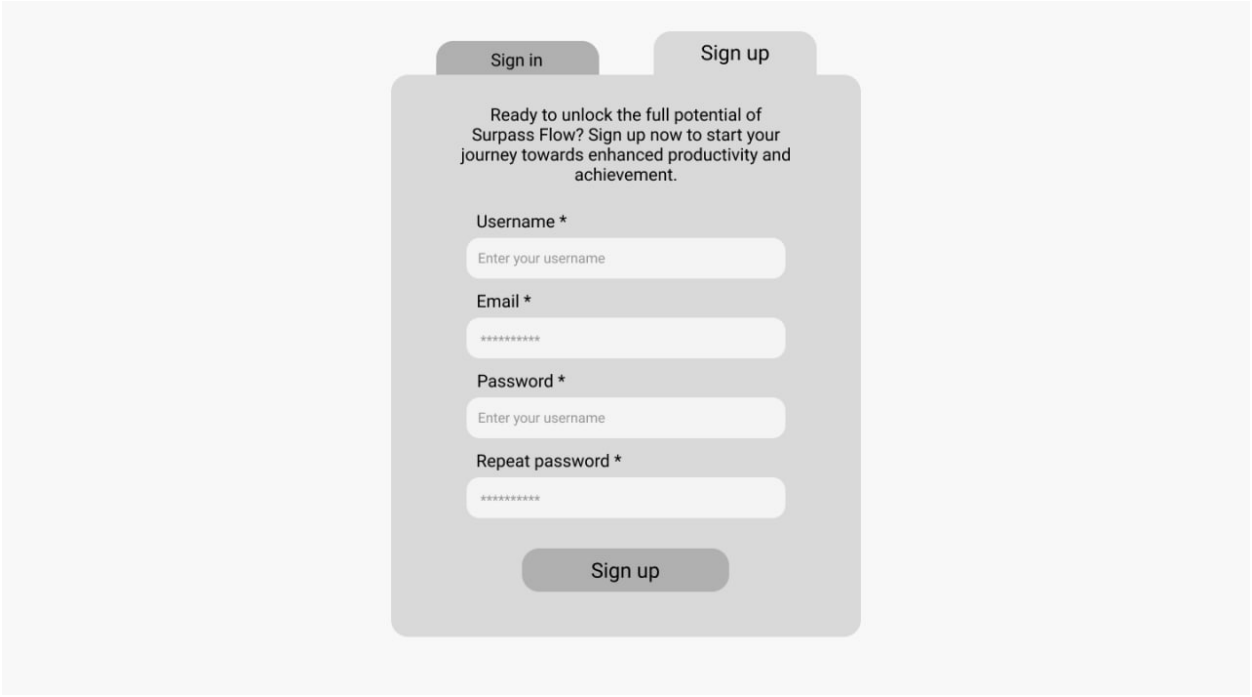

Рисунок 3.10 – Інтерфейс реєстрації

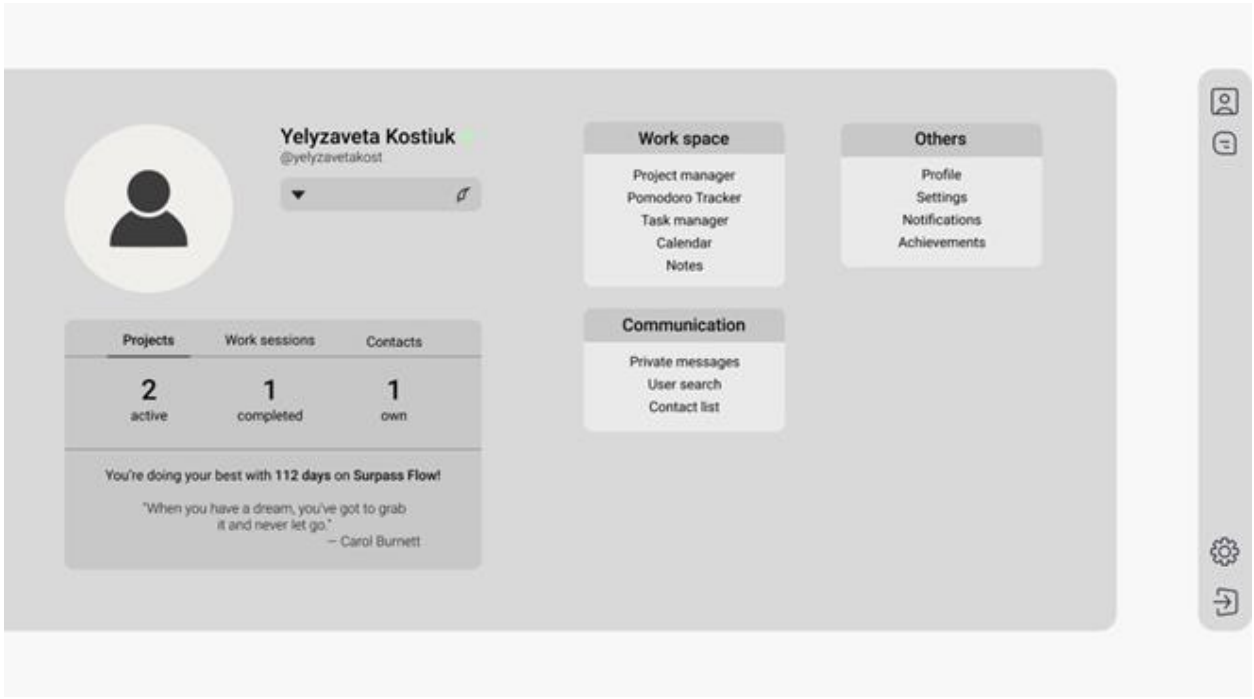

Рисунок 3.11 – Інтерфейс головної сторінки

Сторінка реєстрації представляє собою вікно з чотирма обов'язковими полями для вводу – ім'я користувача, електронна адреса, пароль та його підтвердження, а також містить кнопку, після натиснення якої користувач отримає або відповідні помилки, або потрапить на сторінку авторизації. Окрім цього, є кнопка для швидкого переходу на сторінку авторизації.

Головна сторінка для авторизованого користувача містить коротку інформацію про нього (повне ім'я, аватар, ім'я в системі) а також показує закріплену нотатку при наявності, яку можна приховати та редагувати. Під цією інформацією є панель зі статистикою створених проєктів, пройдених робочих сесій та доданих контактів – інших користувачів, а нижче можна побачити кількість днів після реєстрації на сайті та випадкову мотиваційну фразу. Збоку знаходиться панель з посиланнями для переходу на відповідні сторінки, що поділені на категорії – робочий простір, комунікація та інше. Через «робочий простір» можна перейти на менеджер проєктів, таймер Pomodoro, менеджер завдань, календар та сторінку з нотатками. Через

«комунікацію» можна перейти на сторінку з приватними повідомленнями, знайти користувачів та відкрити список власних контактів. Через «інше» можна перейти на власний профіль, сторінку налаштувань, менеджер системних повідомлень та сторінку досягнень. Справа знаходиться фіксована панель для швидкого переходу, де за замовчуванням можна перейти на власний профіль, у приватні повідомлення, налаштування, чи вийти з облікового запису.

На рис. 3.12 відображено інтерфейс пошуку користувачів, де є поле для введення імені користувача або електронної адреси, за допомогою яких можна знайти іншого користувача, запис про якого з’явиться нижче. У знайденого користувача відображатиметься інформація про повне та системне імена, останній вхід на сайт, а також кнопки додавання в контакти, в збережені та написання приватного повідомлення. Окрім цього є можливість задати фільтри через відповідну кнопку під полем для пошуку.

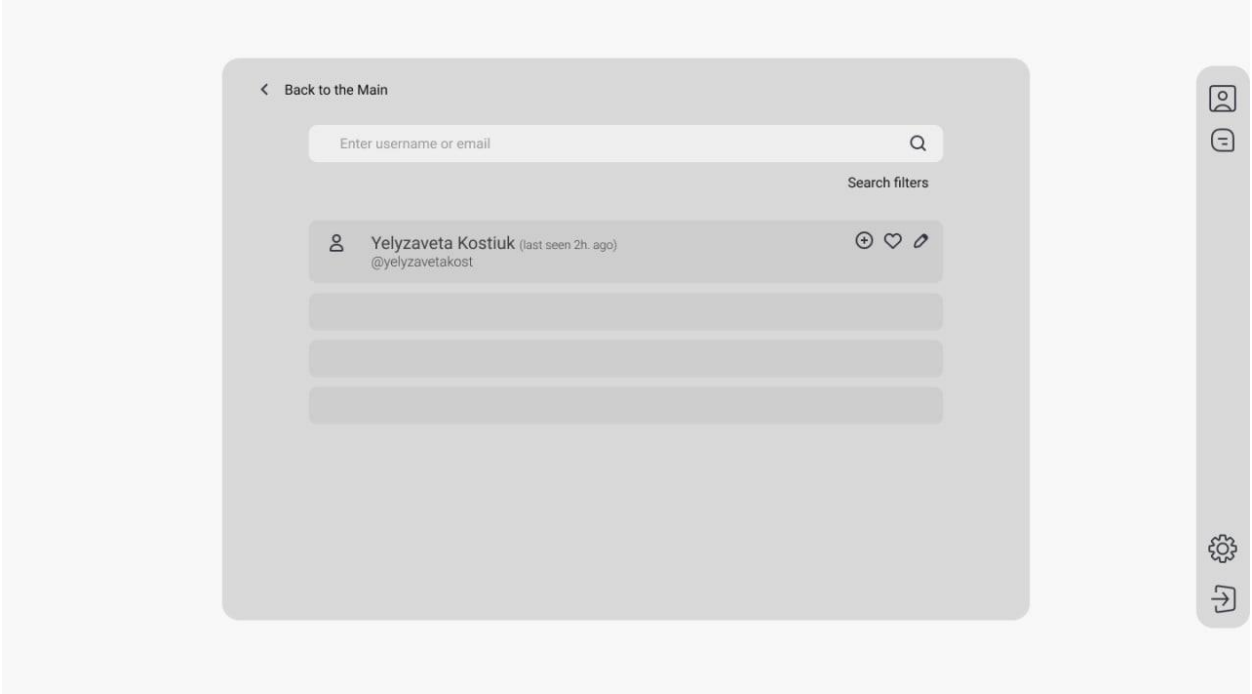

Рисунок 3.12 – Інтерфейс сторінки пошуку користувачів

На рис. 3.13-3.14 відображено сторінки зі списком наявних повідомлень та надсилання нового повідомлення відповідно.

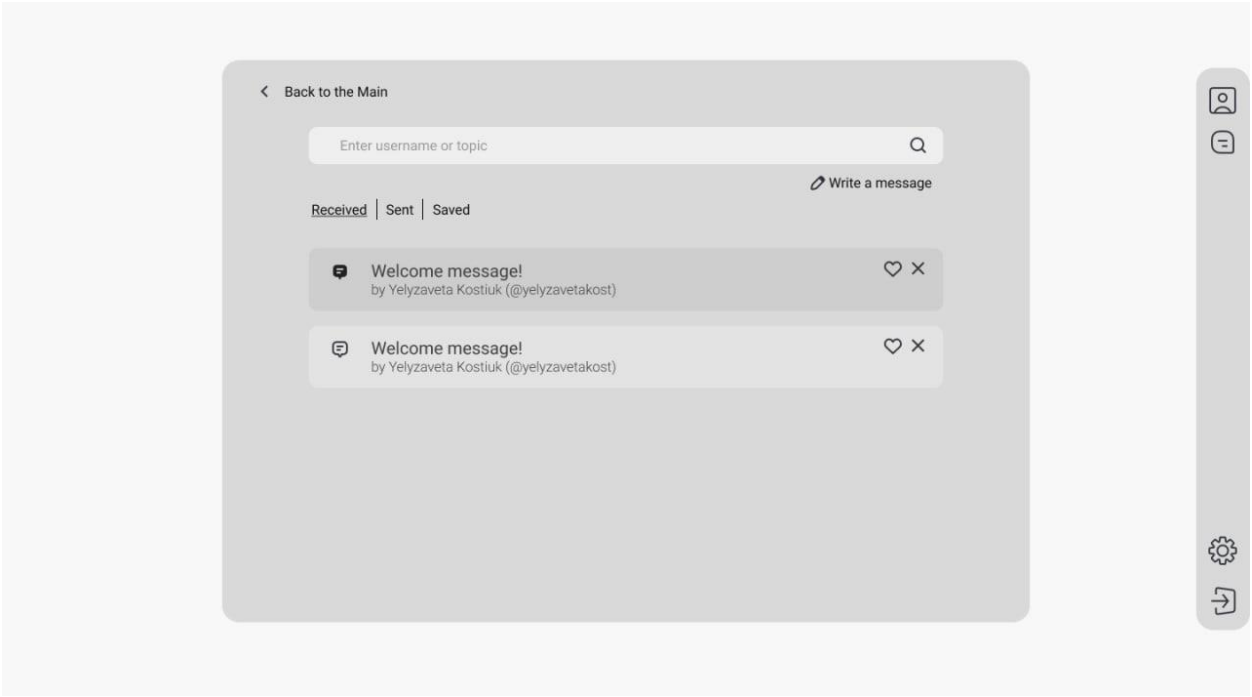

Рисунок 3.13 – Інтерфейс сторінки пошуку користувачів

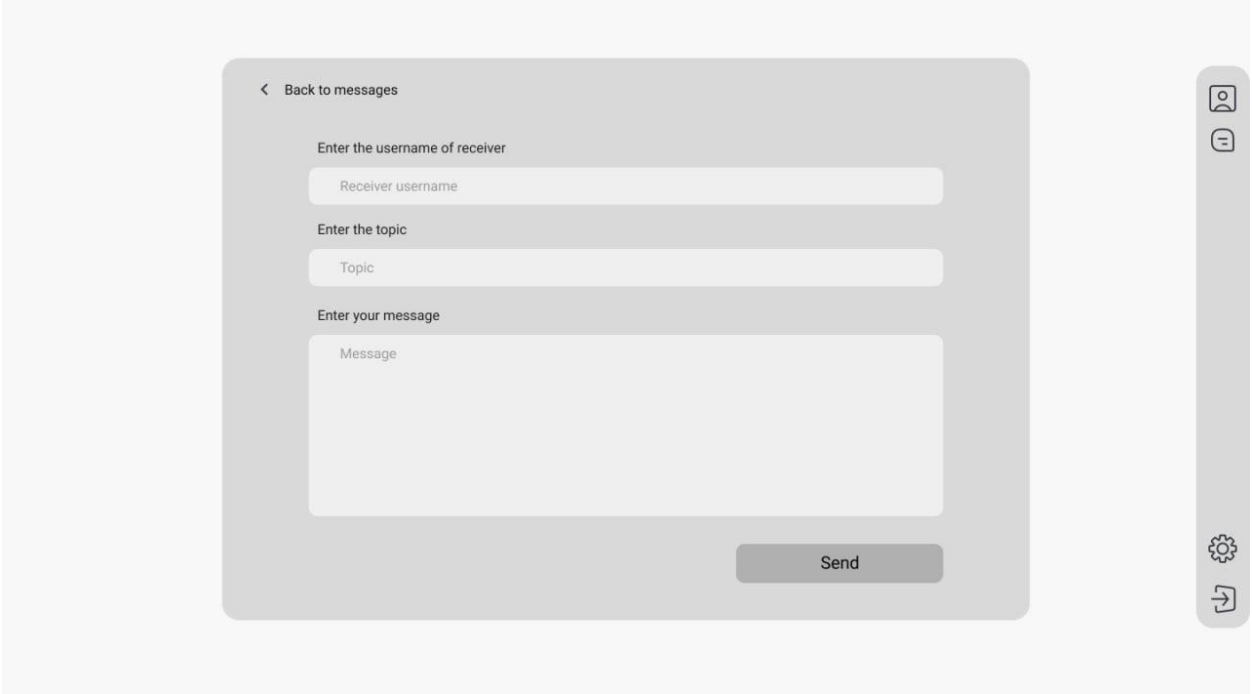

Рисунок 3.14 – Інтерфейс сторінки пошуку користувачів

На сторінці з приватними повідомленнями є поле для пошуку конкретних повідомлень через ім'я користувача (отримувача або відправника) або тему. Під цим полем знаходиться посилання на сторінку створення нового повідомлення, а нижче є кнопки вибору категорії повідомлення – отримані, відправлені або збережені. Кожне повідомлення показує тему, на яку можна натиснути для його відкриття, відправника, а також кнопки збереження та видалення. Прочитані та непрочитані повідомлення відрізняються кольором. Справа знаходиться фіксована панель для швидкого переходу, де за замовчуванням можна перейти на власний профіль, у приватні повідомлення, налаштування, чи вийти з облікового запису.

На сторінці створення повідомлення є три поля для введення – ім'я користувача, тема повідомлення та саме повідомлення, при цьому останнє поле є найбільшим.

Також є кнопка відправки повідомлення, після натиснення на яку будуть показані або відповідні помилки (неіснуючий користувач або порожні поля), або повідомлення про успішну операцію. Окрім цього, є посилання для повернення на сторінку з усіма повідомленнями.

На рис. 3.15-3.16 відображені сторінки для нотаток з теками та форми створення нової нотатки – окремого діалогового вікна, що відкривається при натисканні на відповідну кнопку біля надпису «Мої нотатки». Такий підхід дозволяє зменшити кількість зайвих сторінок та покращити користувацький досвід, оскільки одразу ж після створення нового об’єкту в БД список оновлюється, тому користувач може бачити результат.

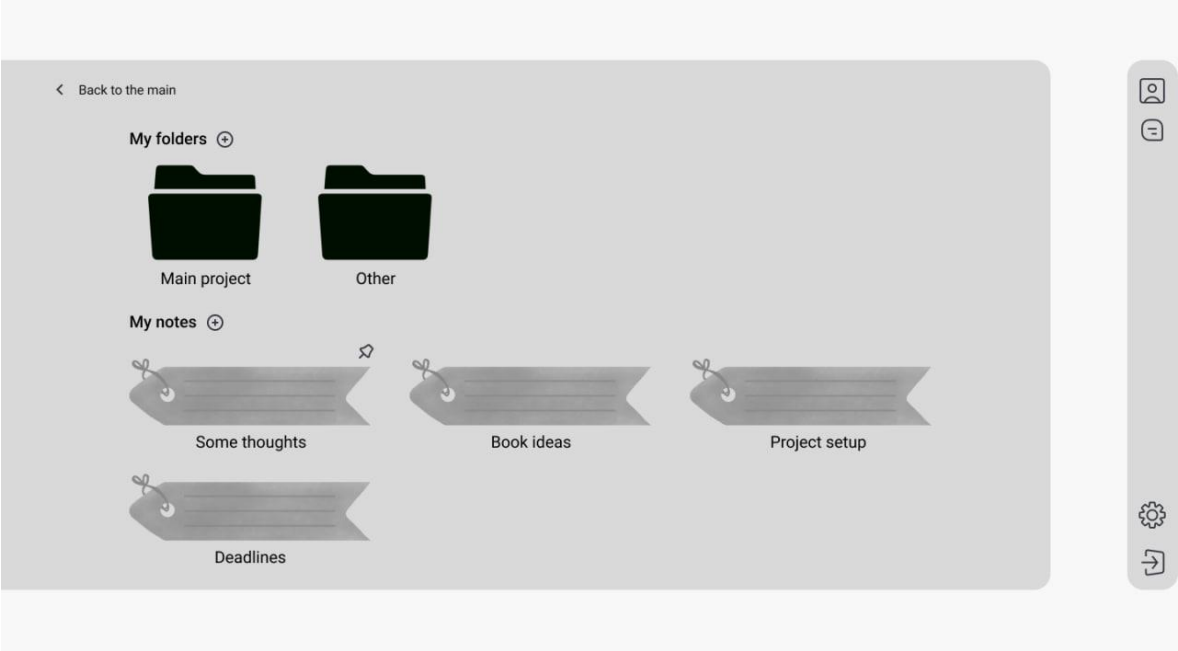

Рисунок 3.15 – Інтерфейс нотаток

Рисунок 3.16 – Інтерфейс створення нової нотатки

На сторінці з нотатками знаходяться теки для нотаток та нотатки без категорії. Поряд з назвами категорій є кнопки створення, що відкривають відповідне вікно.

Теки та нотатки відображаються за допомогою персоналізованих користувачем іконок та назв, а також можуть закріплюватися користувачем за бажанням, що потім відображатиметься окремим значком зверху. Нотатка редагується в окремому спливаючому вікні, яке можна закрити, якщо натиснути на фон позаду нього. У формі знаходяться поля з назвою нотатки, текстом, можливим кольором для персоналізації та чекбокс для закріплення.

Окрім цього, є кнопка видалення та збереження – в першому випадку нотатка видаляється з системи після підтвердження, в другому – показуються відповідні помилки за наявності або ж нотатка успішно зберігається, при цьому усі зміни у списку відображаються одразу ж без необхідності оновлювати сторінку.

3.3 Огляд стеку технологій

Мова програмування Python одною з найбільш популярних для використання у багатьох сферах, в тому числі й у веброзробці, саме тому вона буде використовуватись у проєкті. Її популярність обумовлена простотою синтаксису, великою кількістю бібліотек і фреймворків, а також активною спільнотою розробників.

За даними JetBrains за 2021 рік трьома найпопулярнішими вебфрейворками є Django, Flask та FastAPI [15], які мають ряд власних особливостей, переваг та недоліків, з якими потрібно ознайомитись перед вибором для розробки.

В табл. 3.1 проведено порівняльну характеристику наведених вебфреймворків.

Таблиця 3.1 – Порівняння вебфреймворків Python

Характеристика	**Django**	**Flask**	**FastAPI**
Рік створення	2005	2010	2018
Дата оновлення версії на 04.05.2024	03.04.2024	07.04.2024	03.05.2024
Публічні репозиторії GitHub	62,644	46,816	11,216
Інтерфейс адміністратора	Є	Немає	Немає
Документація	Повна	Достатня	Повна
Асинхронність	Є	Немає (потрібні сторонні бібліотеки)	Є
Використання	Великі вебзастосунки по типу CMS та e-commerce	Мікросервіси та менші вебзастосунки	API та невеликі вебзастосунки
Приклади використання	Instagram, Pinterest, Coursera, Udemy	Netflix, Reddit, Patreon	Netflix, Uber, Microsoft

З огляду на проведений аналіз найкращим вибором для розробки вебзастосунку відслідковування та підтримки продуктивності є Django. Цей вебфреймворк є має багато вбудованого функціоналу і активно підтримується спільнотою, що дає можливість швидко вирішувати певні питання чи проблеми. Окрім Django використовується django-rest-framework – бібліотека

для створення API. За іншим опитуванням JetBrains за 2022 вона посідає перше місце серед найбільш популярних пакетів та набрала 60% [16].

У якості БД використовується PostgreSQL, що в тому ж опитуванні є найбільш популярною серед Django-розробників та має 79%. Ця БД офіційно підтримується, тому проблем з ORM не має виникати.

Для front-end частини в проєкті використовується JavaScript та його бібліотека jQuery, що значно полегшує взаємодію з елементами DOM на сторінці та надає розширений інструментарій [17]. Окрім цього, у якості препроцесору стилів застосовується Stylus з динамічною системою типізації. Він знаходиться в трійці найбільш популярних препроцесорів за 2021 рік [18].

Висновки до розділу 3

В третьому розділі було досліджено ряд діаграм, що допомагають при розробці проєктів – діаграми класів, пакетів, розгортання, взаємодії, станів і переходів. Всі вони організовують дані та структурують наявний план системи, даючи змогу дивитись глибше для усунення недоліків та оптимізації робочих процесів. Розроблено ряд мокапів, що використовуватимуться під час верстки вебсторінок, та враховано найкращі UI/UX практики для гарного користувацького досвіду.

Окрім цього, обрано та описано стек технологій, що буде використовуватися при розробці. Основними мовами програмування для створення застосунку є Python та JavaScript, а серед їх фреймворків використовуватимуться Django та jQuery відповідно. В якості БД використовуватиметься PostgreSQL.

4 ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА ОГЛЯД ЗАСТОСУНКУ

Після етапів моделювання та проєктування важливих системних аспектів та виокремлення компонентів, що будуть використовуватися, потрібно переходити до кодування – написання програмної частини, тестування створеного функціоналу. Під час кодування застосунку використовуватиметься стек технологій, розроблений раніше, а також створений в Figma дизайн.

Оскільки на початкових стадіях розробки система може містити ряд помилок, що можуть призвести до погіршення користувацького досвіду, тестування є важливим кроком перед релізом проєкту.

4.1 Розробка програмних компонентів

Усі підсистеми вебзастосунку зберігаються в директорії apps, що знаходиться в кореневій. Окрім цього, з кореневої директорії можна перейти до шаблонів, статичних файлів, медіа-файлів, скриптів та компонентів Stylus. З файлів можна виділити requirements.txt, в якому містяться всі пакети Python з їх версіями для коректного завантаження та безпомилкової роботи у разі оновлень. В цьому ж місці зберігається файл manage.py, який відповідає за запуск застосунку та централізує всі інші компоненти. Окремо потрібно виділити файл змінних середовища .env, в якому зберігаються деякі дані з Django (секретний ключ для доступу та параметр Debug) та бази даних (ім’я, користувач, пароль, сервер та порт).

Оскільки застосунок буде розгортатися за допомогою Docker, то спочатку потрібно створити файли з кодом, що підійматимуть контейнери. На рис. 4.1 показано код з Dockerfile Django-частини проєкту, в якому додаються всі необхідні залежності (пакети) та налаштовується середовище.

Dockerfile

```
You, 2 months ago | 1 author (You)
1  FROM python:3.11-alpine
2  LABEL maintainer="kttel"
3        You, 2 months ago • project init;
4  ENV PYTHONDONTWRITEBYTECODE 1
5  ENV PYTHONUNBUFFERED 1
6
7  COPY ./requirements.txt /tmp/requirements.txt
8  COPY ./requirements.dev.txt /tmp/requirements.dev.txt
9  COPY ./scripts /scripts
10 COPY ./surpass_flow /app
11
12 WORKDIR /app
13 EXPOSE 8000
14
15 ARG DEV=false
16 RUN python -m venv /py && \
17     /py/bin/pip install --upgrade pip && \
18     apk add --update --no-cache postgresql-client jpeg-dev && \
19     apk add --update --no-cache --virtual .tmp-build-deps \
20         build-base postgresql-dev musl-dev zlib zlib-dev linux-headers && \
21     /py/bin/pip install -r /tmp/requirements.txt && \
22     if [ $DEV = "true" ]; \
23         then /py/bin/pip install -r /tmp/requirements.dev.txt; \
24     fi && \
25     rm -rf /tmp && \
26     apk del .tmp-build-deps && \
27     adduser \
28         --disabled-password \
29         --no-create-home \
30         django-user && \
31     mkdir -p /vol/web/media && \
32     mkdir -p /vol/web/static && \
33     chown -R django-user:django-user /vol && \
34     chmod -R 755 /vol && \
35     chmod -R +x /scripts
36
37 ENV PATH="/scripts:/py/bin:$PATH"
38 USER root
39 CMD ["run.sh"]
```

Рисунок 4.1 – Dockerfile вебзастосунку

В майже кожній з підсистем (модулів) є пакет з моделями – класами, що містять логіку для певних об’єктів. Одним з таких моделей є повідомлення – Message, що містить 7 атрибутів та 3 методи (один з яких є властивістю).

Виходячи з назви та наповнення, цей клас відповідає за комунікацію між користувачами в системі, а код наведено на рис. 4.2.

```
You, 2 months ago | 1 author (You)
class Message(models.Model):
    receiver = models.ForeignKey(User, related_name="received_messages", on_delete=models.CASCADE)
    sender = models.ForeignKey(User,
                               related_name="sent_messages",
                               on_delete=models.CASCADE,
                               null=True, blank=True)
    title = models.CharField(max_length=255)
    text = models.TextField()
    previous_message = models.ForeignKey("Message", on_delete=models.SET_NULL, null=True, blank=True)
    date_created = models.DateTimeField(default=timezone.now)
    date_read = models.DateTimeField(null=True, blank=True)

    def __str__(self):
        return f"{self.title} by {self.sender} to {self.receiver}"

    @property
    def last_visit_formatted(self):
        intervals = get_time_intervals(self.date_created)
        for value, unit in intervals:
            if value > 0:
                return f"{value} {unit} ago"
        return "just now"

    def make_unread(self):
        self.date_read = None
```

Рисунок 4.2 – Клас Message

Для відображення списку представлених повідомлень додано View – представлення (рис. 4.3), що відповідає за збір даних для шаблону.

```
class MessagesList(LoginRequiredMixin, ListView):
    template_name = "internal_messages/list.html"
    context_object_name = "messages"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        user = self.request.user
        filtering = self.get_filtering()
        context.update({
            "saved_messages": [m.message for m in M.SavedMessage.objects.filter(
                user=user,
                message__in=M.Message.objects.filter(*filtering),
            )],
            "sent_messages": M.Message.objects.filter(*filtering, sender=user)\
                .annotate(is_saved=Exists(M.SavedMessage.objects.filter(message=OuterRef("id")))),
        })
        return context

    def get_filtering(self):
        query = self.request.GET.get("search")
        filtering = []
        if query:
            filtering.append(
                Q(title__icontains=query)|Q(sender__username__icontains=query)|Q(receiver__username__icontains=query)
            )
        return filtering

    def get_queryset(self):
        filtering = self.get_filtering()
        objects = M.Message.objects.filter(*filtering, receiver=self.request.user)\
            .annotate(is_saved=Exists(M.SavedMessage.objects.filter(message=OuterRef("id"))))
        return objects
```

Рисунок 4.3 – Представлення списку повідомлень

При завантаженні сторінки всі дані з представлення додаються в шаблон, який вказано в змінній класу template_name. На рис. 4.4 показано частину коду цього шаблону, в якій генерується саме окреме повідомлення. Для позначення CSS класів використовується методологія БЕМ, яка є популярним набором правил для подальшого задання стилів за допомогою препроцесорів і полягає у розбитті інтерфейсу на складові, які компонуються і складають систему, в якій легко орієнтуватися:

– блок – незалежний компонент, що може існувати самостійно;

– елемент – частина блоку, що не може існувати поза ним та позначається через подвійне нижнє підкреслення;

– модифікатор – варіант блоку чи елементу, що змінює його вигляд чи поведінку та позначається через подвійний дефіс.

```
<div class="messages-content__messages__item js-internal-message {% if message.date_read %} messages-content__messages__item--read{% endif %}"
    data-id="{{ message.pk }}">
    <div class="messages-content__messages__item__icon {% if message.date_read %}messages-content__messages__item__icon--read{% endif %}"></div>
    <div class="messages-content__messages__item__data">
        {% include "widgets/delete_object.html" with object=message obj_type="message-delete" %}
        {% include "widgets/save_object.html" with object=message obj_type="message-save" %}
        <div class="messages-content__messages__item__title">
            <a href="{% url 'internal_messages:details' message.pk %}">{{ message.title }}</a>
        </div>
        <div class="messages-content__messages__item__author">
            by {% firstof message.sender "system" %} to {{ message.receiver }} ({{ message.last_visit_formatted }})
        </div>
    </div>
    <div class="messages-content__messages__item__actions">
        {% if not message.is_saved and not type == "saved" %}
            <div class="messages-content__messages__item__action messages-content__messages__item__action--save js-save-message"
                 data-id="{{ message.pk }}"
                 title="Save"></div>
        {% endif %}
        <div class="messages-content__messages__item__action messages-content__messages__item__action--delete js-delete-message"
             data-id="{{ message.pk }}"
             title="Delete"></div>
    </div>
</div>      You, 2 months ago • internal messages update;
```

Рисунок 4.4 – Шаблон відображення повідомлення

Окрім цього підсистема повідомлень використовує API (рис. 4.5) – це потрібно, щоб надсилати AJAX-запити на сервер і динамічно оновлювати дані в залежності від відповіді, яка отримується [19].

```python
You, 2 months ago | 1 author (You)
class InternalMessagesViewSet(ModelViewSet):
    permission_classes = (IsAuthenticated,)
    serializer_class = MessageSerializer

    def get_queryset(self):
        user = self.request.user
        messages = M.Message.objects.filter(Q(sender=user)|Q(receiver=user))
        return messages

    @action(detail=True, methods=["post"])
    def save_message(self, request, pk=None):
        message = self.get_object()
        try:
            M.SavedMessage.objects.get_or_create(user=request.user, message=message)
        except:
            return Response({"status": False})
        return Response({"status": True})
```

Рисунок 4.5 – Шаблон відображення повідомлення

Щодо AJAX-запитів – вони здійснюються в файлах JS після натискання на певні кнопки чи спрацьовуванні деяких умов при подіях. Якщо розглядати описане вище API, а саме метод збереження повідомлення, то його запит можна побачити на рис. 4.6.

```javascript
saveMessageConfirmation.on("click", (e) => {
    const element = $(e.target);
    const objId = element.attr("data-id");

    $.ajax({
        method: "POST",
        url: `${apiUrl}${objId}/save_message/`,
        headers: {"X-CSRFToken": CSRFTOKEN},
        dataType: "json",
        contentType: "application/json; charset=utf-8",
        error: function(data) {
            Toastify(defaultErrorData).showToast();
        },
        success: function(data) {
            $(".js-message-popup").addClass("hidden");
            const savedMessage = element.parents('.js-internal-message').first();
            savedMessage.find(".js-save-message").remove();
            savedMessage.clone().appendTo(savedMessaged);
            Toastify({...defaultSuccessData, text: "Your message was saved successfully!"}).showToast();
        }
    });
});
```

Рисунок 4.6 – Запит збереження повідомлення

Елемент, до якого прикріплено обробник подій, є кнопкою для підтвердження збереження – по її натисканню функція отримує data-атрибут id відповідного повідомлення, після чого надсилає запит з деякими параметрами. Формування URL відбувається за допомогою базового URL цього API, унікального ідентифікатора та доповнення save_message, яке було додано в попередню функцію на python через декоратор action. CSRF-токен в заголовках потрібен для безпеки даних користувачів від CRSF-атак, що спрямовані на сприймання браузером чужого втручання з метою викрадення даних.

Наступним елементом є тип даних, що відправлятиметься і отримуватиметься – json, що є доволі популярним при роботі з REST API [20].

Далі написані функції обробки результату при помилці або ж при успішній обробці запиту. В обох випадках користувач отримує системне повідомлення зі зворотнім зв’язком до своїх дій за допомогою бібліотеки Toastify, а у випадку успіху спливаюче вікно з підтвердженням зникає, в той час як користувацьке повідомлення додається до секції збережених і не видалятиметься з системи через певний час.

Більш цікавим прикладом використання JavaScript є функціонал таймеру Pomodoro, який передбачає його запуск, оновлення та виконання задач. На рис. 4.7 показано функціонал встановлення та запуску таймеру, який змінює цикл роботи та відпочинку з потрібним для них часом, що було задано користувачем при створенні сесії.

```
const setTimer = () => {
    var timer = workDuration, minutes, seconds;
    minutes = parseInt(timer / 60, 10)
    seconds = parseInt(timer % 60, 10);

    minutes = minutes < 10 ? "0" + minutes : minutes;
    seconds = seconds < 10 ? "0" + seconds : seconds;

    display.textContent = minutes + ":" + seconds;
}

const startTimer = (duration) => {
    var timer = duration, minutes, seconds;
    timerInterval = setInterval(function() {
        minutes = parseInt(timer / 60, 10);
        seconds = parseInt(timer % 60, 10);

        minutes = minutes < 10 ? "0" + minutes : minutes;
        seconds = seconds < 10 ? "0" + seconds : seconds;

        display.textContent = minutes + ":" + seconds;

        if (--timer < 0) {
            timer = duration;
            clearInterval(timerInterval);
            if (turn == "work") {
                turn = "rest";
                startTimer(restDuration);
            } else {
                if (CYCLE + 1 > 3) {
                    $(".js-turn").text("Finished!");
                    clearInterval(timerInterval);
                    finishWorkSession();
                    return;
                }
                increaseCycle();
                turn = "work";
                startTimer(workDuration);
            }
            switchTurn();
            $(".js-turn").addClass("hidden")
            $(`.js-turn[data-turn=${turn}]`).removeClass("hidden");
        }
    }, 1000);
}
```

Рисунок 4.7 – Функціонал таймеру

Окрім цього, кожна зміна циклу та завершення всього таймеру передбачає зміни в БД, що означає використання подібних AJAX-запитів.

Використання Stylus відображено на рис. 4.8.

```
.messages-content
  content-default()
  &__search
    width 100%
    display flex
    flex-direction column
    justify-content center
    form
      width 100%
      display flex
      justify-content center
    &__field
      border-radius $border-radius-average
      background-color var(--color-input-default)
      padding .75rem 2.25rem
      border none
      width 80%
      @media screen and (max-width: $width-phone)
        padding .75rem 1.25rem
  &__details
    display flex
    flex-direction column
    width 100%
  &__title
    font-weight 700
    font-size 2rem
    @media screen and (max-width: $width-tablet)
      font-size 1.25rem
```

Рисунок 4.8 – Частина коду задання стилів

Подібний принцип написання коду використовується у всіх підсистемах проєкту, що забезпечує зрозумілість та цілісність системи.

4.2 Тестування застосунку

Будь-який програмний продукт може містити баги, які можуть по-різному впливати на систему – деякі перехоплюються та ігноруються або ж логуються, але інколи можуть бути серйозні проблеми, що призводять до падіння всього вебсайту. Етап тестування потрібен, щоб впевнитися в коректності основного функціоналу та пройтися по всім можливим сценаріям поведінки користувача, щоб зменшити шанси наявності певної помилки та вберегти клієнтів від можливих недоліків неперевіреного релізу.

В таблицях 4.1-4.6 представлені сценарії та результати ручного тестування застосунку відслідковування та підтримки продуктивності ІТ-фахівців.

Таблиця 4.1 – Реєстрація користувача

Діючі особи	Неавторизований користувач, вебзастосунок		
Мета	Зареєструвати обліковий запис		
Передумова	В системі не зареєстровано користувача з цільовою електронною адресою		
Пріоритет	Високий	**Дата**	04.05.2024
Успішний сценарій: – користувач переходить на сторінку реєстрації; – користувач вводить ім’я, адресу, пароль та його підтвердження; – користувач натискає на кнопку «Зареєструватися»; – дані користувача записуються в БД; – користувач перенаправляється на сторінку авторизації; – користувач повідомляється про успішну реєстрацію.			
Результат	Успішно		
Розширення			
1а	Користувач не вводить одне з полів, тому система не дозволяє зареєструватися та просить заповнити дані.		
2а	Користувач помиляється в підтвердженні паролю, тому система виводить помилку.		
3а	Введена користувачем електронна адреса вже існує в БД та не може бути використана повторно, тому система запропонує користувачу або ж відновити пароль, або ж обрати іншу електронну адресу.		

Таблиця 4.2 – Авторизація користувача

Діючі особи	Неавторизований користувач, вебзастосунок		
Мета	Увійти в існуючий обліковий запис		
Передумова	В системі зареєстровано користувача за відповідними даними.		
Пріоритет	Високий	**Дата**	04.05.2024
Успішний сценарій: – користувач переходить на сторінку авторизації; – користувач вводить ім’я та пароль, вказані при реєстрації; – користувач натискає на кнопку «Увійти»; – система перевіряє введені дані; – користувач перенаправляється на головну сторінку; – користувач повідомляється про успішну авторизацію.			
Результат	Успішно		
Розширення			
1а	Користувач не вводить одне з полів, тому система не дозволяє авторизуватися та просить заповнити дані.		
2а	Користувач вводить дані, яких немає в БД, тому система повідомляє про помилку у введенні імені користувача та (або) паролю.		

Таблиця 4.3 – Додавання користувача в проєкт

Діючі особи	Творець проєкту, запрошений користувач		
Мета	Додати користувача у власний проєкт		
Передумова	Користувач має мати або власний проєкт, або проєкт з достатніми правами, а також мати запрошеного в контактах.		
Пріоритет	Середній	**Дата**	04.05.2024

Кінець таблиці 4.3

Успішний сценарій: – користувач переходить на сторінку проєкту; – користувач натискає на кнопку «Запросити користувача»; – користувач обирає користувача зі списку контактів та натискає на кнопку «Запросити»; – запрошений користувач отримує системне повідомлення про запрошення; – запрошений користувач відкриває сторінку з проєктами; – запрошений користувач натискає на кнопку «Прийняти»; – запрошений користувач перенаправляється на сторінку проєкту.	
Результат	Успішно
Розширення	
1а	Користувач не має цільового користувача в контактах і не може надіслати запрошення.
2а	Запрошений користувач відхиляє запрошення в проєкт. Система повідомляє того, хто запросив, про відхилене запрошення.
3а	Запрошений користувач має встановлений ліміт активних проєктів, а проєкт, в який його було запрошено, має такий статус. Система надсилає цільовому користувачу повідомлення про спробу отримати запрошення і підказку щодо збільшення ліміту активних проєктів.
4а	Запрошений користувач не приймає запрошення протягом 12 годин і воно анулюється. Той, хто запрошував, отримає системне повідомлення.

Таблиця 4.4 – Перетягування завдання на Kanban-дошці

Діючі особи	Авторизований користувач, вебзастосунок		
Мета	Змінити поточний статус завдання		
Передумова	Користувач має створене завдання		
Пріоритет	Середній	**Дата**	04.05.2024
Успішний сценарій: – користувач переходить на сторінку керування задачами; – користувач перетягує задачу з однієї дошки на іншу; – задача закріплюється за новим статусом; – користувач повідомляється про успішну зміну статусу задачі.			
Результат	Успішно		
Розширення			
1а	Користувач перетягує задачу на дошку «В процесі», але в нього закінчився ліміт таких одночасних задач. Система повідомляє користувача про помилку та не змінює статус задачі.		
2а	Користувач відпустив завдання не над дошкою. Система повертає завдання на старе місце.		

Таблиця 4.5 – Початок робочої сесії Pomodoro

Діючі особи	Авторизований користувач, вебзастосунок		
Мета	Розпочати працювати з таймером робочої сесії		
Передумова	Користувач має створену сесію		
Пріоритет	Середній	**Дата**	04.05.2024
Успішний сценарій: – користувач переходить на сторінку робочих сесій; – користувач обирає робочу сесію; – користувач вводить завдання на робочий період (необов’язково);			

Кінець таблиці 4.5

– користувач натискає на кнопку «Зберегти»; – користувач натискає на кнопку «Почати сесію».	
Результат	Успішно
Розширення	
1а	Користувач має запущену сесію, яка не була завершена. Система запропонує достроково закінчити почату сесію натисканням на відповідну кнопку або почекати на її самостійне завершення.
2а	Користувач має вичерпаний ліміт в кількості робочих сесій на день. Система повідомляє його про необхідність почекати наступного дня або просить змінити ліміт.

Таблиця 4.6 – Додавання користувача до контактів

Діючі особи	Авторизований користувач, цільовий користувач-контакт		
Мета	Надіслати запит на додавання користувача до контактів та отримати згоду		
Передумова	В системі має бути зареєстровано цільового користувача з відкритим профілем		
Пріоритет	Середній	**Дата**	04.05.2024
Успішний сценарій: – користувач переходить на сторінку пошуку користувачів; – користувач вводить ім’я або електронну адресу цільового контакту; – користувач натискає на кнопку «Перейти в профіль» в списку дій після імені цільового контакту; – користувач перенаправляється на сторінку цільового контакту;			

Кінець таблиці 4.6

– користувач натискає на кнопку «Додати в контакти»; – система надсилає цільовому користувачу повідомлення про запит на додавання в контакти; – цільовий користувач переходить на сторінку контактів; – цільовий користувач натискає на кнопку «Прийняти» біля імені користувача, що надіслав заявку; – система надсилає повідомлення обом користувачам про новий контакт.	
Результат	Успішно
Розширення	
1а	Користувач заблокований цільовим контактом і не може натиснути на кнопку «Додати в контакти». Система повідомляє про це при наведенні на неактивну кнопку.
2а	Цільовий контакт відхиляє заявку. Система надсилає повідомлення користувачу, що надіслав заявку, про відхилення.
3а	Цільовий контакт заблокований у системі, тому при надсиланні заявки стається помилка. Система повідомляє про те, що операція неможлива.
4а	Користувач або цільовий контакт видалили свої профілі після надсилання заявки. Система повідомляє користувача про те, що операція неможлива через видалення профілю.
5а	Користувач намагається додати себе в контакти. Система відображає повідомлення про те, що операція неможлива.

Проведення подібних тестувань допомагає впевнитися у відсутності критичних помилок, з якими користувач може зіткнутися під час роботи з вебзастосунком.

Розробка розширень покращує розуміння можливих альтернативних сценаріїв і дозволяє обробити специфічний випадок. Це, в свою чергу, підвищує якість кінцевого продукту, забезпечує кращий користувацький досвід і сприяє більшій задоволеності користувачів. Регулярні тести та аналіз розширень допомагають виявляти й усувати потенційні проблеми на ранніх етапах, що сприяє стабільній роботі системи та її відповідності потребам користувачів.

Висновки до розділу 4

У четвертому розділі написано програмний код застосунку відслідковування та підтримки продуктивності роботи ІТ-фахівців з використанням обраного стеку технологій, серед яких основною мовою програмування для back-end частини є Python з його вебфреймворком Django, а на front-end боці – jQuery зі Stylus. Після цього проведено ряд тестувань, щоб мінімізувати шанс отримання помилок користувачами після релізу.

Окрім цього, описано основні функції для роботи із застосунком, які можуть знадобитися при знайомстві з системою. Розроблені успішні сценарії можна переписати як частини документації, що допоможе як і майбутнім користувачам ПЗ, так і новим розробникам.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра створено інструмент для покращення роботи ІТ-фахівців – вебзастосунок відслідковування та підтримки продуктивності. Для досягнення поставленої мети виконано ряд завдань, а саме:

– досліджено інструменти досягнення продуктивності шляхом аналізу застосунків-аналогів;

– розроблено UI/UX дизайн для найкращого користувацького досвіду;

– змодельовано та спроєктовано систему;

– обрано стек технологій та розроблено вебзастосунок;

– додано функціонал попередження вигорання;

– протестовано вебзастосунок.

Дослідження теми показало, що продуктивність ІТ-фахівців є важливим аспектом успіху компаній, тому впровадження інструментів її підвищення та підтримки може бути результативним рішенням. Виокремлення особливостей, переваг та недоліків застосунків-аналогів підтвердило необхідність наявності основного функціоналу, що розроблявся, а також підкреслило актуальність важливість розкриття теми для попередження вигорання.

На етапі моделювання створено ряд UML-діаграм, що спростили процес розробки усієї системи та прискорили кодування, яке було розпочато після вибору стеку технологій для front-end та back-end частин застосунку. Результатом виконаної роботи є вебзастосунок відслідковування та підтримки продуктивності ІТ-фахівців.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. N. Raman, M. Cao, Y. Tsvetkov, C. Kästner, etc. Stress and burnout in open source: toward finding, understanding, and mitigating unhealthy interactions. ACM/IEEE 42nd International Conference on Software Engineering, 2020. URL: https://doi.org/10.1145/3377816.3381732 (date of access: 20.03.2024).

2. A. J. Thadhani. Factors affecting programmer productivity during application development. IBM Systems Journa, vol. 23, issue 1, 1984. URL: https://ieeexplore.ieee.org/document/5387803 (date of access: 20.03.2024).

3. P. Jalote, D. Kamma. Studying Task Processes for Improving Programmer Productivity. IEEE Transactions on Software Engineering, vol. 47, no. 4, pp. 801-817, 2021. URL: http://dx.doi.org/10.1109/TSE.2019.2904230 (date of access: 21.03.2024).

4. J. Münch, J. Heidrich. Software project control centers: concepts and approaches, Journal of Systems and Software, vol. 70, 2004. URL: https://doi.org/10.1016/S0164-1212(02)00138-3 (date of access: 21.03.2024).

5. P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta. Agile Software Development Methods: Review and Analysis, 2017. URL: https://doi.org/10.48550/arXiv.1709.08439 (date of access: 21.03.2024).

6. T. Dingsoyr, S. Nerur, B. Balijepally, N. Brede Moe. A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, vol. 85, 2012. URL: https://doi.org/10.1016/j.jss.2012.02.033 (date of access: 21.03.2024).

7. V. Murali, P. M. Devidas. Impact of high performance work practices on burnout intervened by job overload with respect to a select IT firm. International Conference on Science Engineering and Management Research (ICSEMR), Chennai, India, 2014. URL: https://ieeexplore.ieee.org/document/7043653 (date of access: 24.03.2024).

8. A. Cockburn. Structuring Use cases with goals. Journal of object-oriented programming, 10 (5), 1997. URL: https://www.researchgate.net/profile/Alistair-Cockburn/publication/2807676_Structuring_Use_cases_with_goals/links/56d434b908ae2ea08cf8e07b/Structuring-Use-cases-with-goals.pdf (date of access: 24.03.2024).

9. K. Bittner, I. Spence. Use case modeling. Addison-Wesley Professional, 2003. URL: https://archive.org/details/usecasemodeling00kurt (date of access: 24.03.2024).

10. J. Deacon. Model-view-controller (MVC) architecture, 2009. URL: https://www.academia.edu/30077059/Model_View_Controller_MVC_Architecture (date of access: 24.03.2024).

11. H. Gore, R. K. Singh, V. Jagota, M. Shabaz, etc. Django: Web Development Simple & Fast. Annals of the Romanian Society for Cell Biology, vol.6, issue 6. 2021. URL: https://annalsofrscb.ro/index.php/journal/article/view/6301 (date of access: 15.04.2024).

12. N. M. Idris, C. F. Foozy, P. Shamala. A Generic Review of Web Technology: DJango and Flask. International Journal of Advanced Science Computing and Engineering, vol. 2, 2021. URL: https://doi.org/10.30630/ijasce.2.3.29 (date of access: 15.04.2024).

13. W. Nutt, S. Razniewski. Completeness of queries over SQL databases. ACM International conference on Information and knowledge management, 2012. Association for Computing Machinery, New York. URL: https://doi.org/10.1145/2396761.2396875 (date of access: 15.04.2024).

14. W. S. L. Nasution, P. Nusa. UI/UX Design Web-Based Learning Application Using Design Thinking Method. ARRUS Journal of Engineering and

Technology, pp. 18-27, 2021. URL: https://doi.org/10.35877/jetech532 (date of access: 03.05.2024).

15. Python Developers Survey 2021 Results. URL: https://lp.jetbrains.com/python-developers-survey-2021/ (date of access: 20.04.2024).

16. Django Developers Survey 2022 Results. URL: https://lp.jetbrains.com/django-developer-survey-2022/ (date of access: 20.04.2024).

17. B. Bibeault, Y. Katz. (2015). jQuery in Action. Simon and Schuster, 2015. URL: https://docs.niwa.co.nz/library/public/1933988355update.pdf (date of access: 10.05.2024).

18. The State of CSS 2021: Pre-/Post-processors. URL: https://2021.stateofcss.com/en-US/technologies/pre-post-processors/ (date of access: 10.05.2024).

19. I. Garcia and A. Bellogin. 2018. Towards an open, collaborative REST API for recommender systems. ACM Conference on Recommender Systems, 2018. Association for Computing Machinery, New York. URL: https://doi.org/10.1145/3240323.3241615 (date of access: 19.04.2024).

20. D. Rubio. REST services with Django. Beginning Django, pp 549–566, 2017. URL: https://doi.org/10.1007/978-1-4842-2787-9_12 (date of access: 01.05.2024).