

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБЗАСТОСУНКУ
ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ МІЖ ЗАМОВНИКОМ
І РОЗРОБНИКОМ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
DEVELOPMENT OF THE WEB APPLICATION CLIENT SIDE
TO AUTOMATE THE WORK BETWEEN A CUSTOMER
AND A SOFTWARE DEVELOPER

Виконав: студент(ка) 4 курсу, групи КНТ-110сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ДЯЧЕНКО М. О.

(ПРИЗВИЩЕ та ініціали)

Керівник КАПЛІЄНКО Т. І.

(ПРИЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДЯЧЕНКО Максима Олександровича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка клієнтської частини вебзастосунку для автоматизації роботи між замовником і розробником програмного забезпечення

керівник проєкту (роботи) КАПЛІЄНКО Тетяна Ігорівна, к.т.н., доцент,
(ПРИЗВИЩЕ, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 04 ” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 29 травня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КАПЛІЄНКО Т. І., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 06 » лютого 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Максим ДЯЧЕНКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КАПЛІЄНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 89 с., 26 рис., 3 табл., 1 дод., 24 джерел.

ВЕБЗАСТОСУНОК, КЛІЄНТСЬКА ЧАСТИНА, REACT, REST, REDUX, РОЗРОБНИК, КЛІЄНТ, ПЛАНУВАННЯ РОЗРОБКИ.

Об'єкт дослідження – програмні засоби для автоматизації процесу взаємодії між замовником та розробником програмних рішень.

Предмет дослідження – вебзастосунок, який допомагає автоматизувати та покращити ефективність взаємодії між замовником та розробником програмного забезпечення.

Мета роботи – дослідження сучасних front-end вебтехнологій та їх застосування на прикладі створення клієнтської частини вебсайту для автоматизації роботи над проєктом, з фокусом на поліпшенні комунікації між замовником та розробником програмного забезпечення.

Матеріали, методи та технічні засоби: структурне та компонентно-орієнтоване програмування, JavaScript бібліотека React, персональний комп'ютер з процесором Intel Core i7-9750H під управлінням операційної системи macOS Ventura 13.2.1.

Результати. Була розроблена клієнтська частина ПЗ, що є складовою програмного забезпечення, спрямованого на поліпшення комунікації між замовником та розробником програмного забезпечення.

Висновки. Розроблено клієнтську частину вебзастосунку з метою покращення взаємодії між замовником та розробником.

Галузь використання – покращення взаємодії між замовником та розробником програмного забезпечення в компаніях, які займаються створенням програмних продуктів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 26 figures, 3 tables, 1 appendixes, 24 sources.

WEB APPLICATION, CLIENT-SIDE, REACT, REST, REDUX, DEVELOPER, CLIENT, DEVELOPMENT PLANNING.

Research subject - software tools for automating the interaction process between the customer and the software solution developer.

The subject of the research is a web application that helps automate and improve the efficiency of interaction between the client and software developer.

The purpose of the work is to explore modern front-end web technologies and their application in creating the client-side of a website for automating project management, with a focus on improving communication between the client and software developer.

Materials, methods, and technical tools: structured and component-based programming, JavaScript library React, personal computer with Intel Core i7-9750H processor running macOS Ventura 13.2.1 operating system.

Results. A client-side component of a web application was developed, which is part of software aimed at improving communication between the client and software developer.

Conclusions. A client-side component of a web application has been developed with the aim of improving interaction between the client and the software developer.

Field of application - improving interaction between the client and software developer in companies engaged in software product development.

ЗМІСТ

ВСТУП	8
1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ	9
1.1 Основні поняття	9
1.2 Огляд програмних аналогів	11
1.3 Висновки за розділом 1	15
2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ	16
2.1 Вибір засобів проєктування і розробки	16
2.2 Визначення функціональних вимог до програмного забезпечення	19
2.3 Проєктування бази даних	21
2.4 Визначення структури програми	23
2.5 Висновки за розділом 2	25
3 РОЗРОБКА ПРОГРАМИ	26
3.1 Алгоритм функціонування програми	26
3.2 Опис реалізованих складових вебзастосунку	26
3.3 Висновки за розділом 3	33
4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ	34
4.1 Опис програми	34
4.2 Експлуатація програми	35
4.3 Тестування програми	45
4.4 Висновки за розділом 4	46
ВИСНОВКИ	47
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	48
ДОДАТОК А Текст програми	50
ДОДАТОК Б Слайди презентації	82

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface;
HTTP – HyperText Transfer Protocol;
IDE – Integrated Development Environment;
REST – Representational State Transfer;
ПЗ – програмне забезпечення.

ВСТУП

Сьогоднішній розвиток технологій дозволяє проводити глобальну автоматизацію різних сфер суспільного життя, що призводить до збільшення обсягу інформації, що зберігається. Однак, бази даних стають менш зручними в експлуатації, тому пошук необхідної інформації стає дедалі більш часо- і ресурсозатратним завданням для користувачів з усього світу.

Відповідно до цього, народжується потреба в зручному та швидкісному використанні великих обсягів даних незалежно від географічного положення користувачів. Таким рішенням став Інтернет - найбільш розвинена у світі інформаційна система, що забезпечує комунікацію між мільйонами людей та орієнтована на їх інформаційне забезпечення за допомогою вебтехнологій. Використання вебтехнологій дозволяє зручно обмінюватися інформацією, а їх постійний розвиток створює нові можливості.

Метою даної дипломної роботи є дослідження розвитку вебтехнології шляхом розробки клієнтської частини вебзастосунка, що допомагає поліпшити ефективність комунікації та співпраці між замовником та розробником програмного забезпечення. При цьому, наявність злагодженої роботи у мережі Інтернет є необхідним для покращення її ефективності та обміну інформацією між учасниками ринкового процесу. Виходячи з причин, описаних вище, темою дипломної роботи є розробка клієнтської частини вебзастосунку, що може бути використана для поліпшення взаємодії між замовником та розробником у компаніях, які займаються створенням програмних продуктів.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Предметна область визначається як конкретна галузь людської діяльності, яка описується за певними критеріями.

Невід’ємною характеристикою предметної області є модель. Модель, у свою чергу, є математичним об’єктом, що використовується для опису властивостей системи, що досліджується. У цьому випадку вона є системою, яка імітує структуру та функціонування реальної предметної області.

Кожен елемент предметної області має складну структуру, що складається з безлічі об’єктів та процесів, які використовують користувачі з різними поглядами на цю область. Для автоматизації процесу розробки програмного забезпечення і створення умов для ефективної комунікації між замовником та розробником необхідно мати чітке відображення сутності, мети та етапів проєкту. У даному випадку, завданням є створення клієнтської частини вебзастосунку, що дозволить автоматизувати процес роботи між замовником та розробником програмного забезпечення, для цього було проведено дослідження даної предметної області.

Для замовників не завжди є можливість швидко зв’язатися з розробниками програмного забезпечення та коригувати деталі проєкту. Також вони можуть мати різні погляди на предметну область, що ускладнює процес розробки програмного забезпечення. Щоб уникнути таких складнощів, фахівцю зі створення програмних засобів потрібно чітко розуміти сутність проєкту та його етапи.

Темою дипломної роботи є створення клієнтської частини вебзастосунку для автоматизації роботи між замовником та розробником.

Розробка програмного забезпечення є складним та багатограним процесом, який вимагає від фахівців знань з інформатики, математики, інженерії та інших областей знань. Цей процес передбачає послідовне виконання різних дисциплін, таких як аналіз вимог, проєктування,

програмування, тестування, інтеграція та впровадження програмного забезпечення.

Аналіз вимог є початковим етапом розробки програмного забезпечення, де визначаються потреби та вимоги замовника. На основі цих вимог проводиться проектування програмного забезпечення, де визначаються його архітектура, функціональність та інші параметри.

Програмування є безпосередньою розробкою програмного забезпечення та включає в себе написання коду, тестування та оптимізацію його роботи. Після цього проводиться тестування програмного забезпечення для перевірки його працездатності та виявлення можливих помилок.

Системна інтеграція передбачає поєднання різних компонентів програмного забезпечення в єдину систему, що забезпечує їх взаємодію та співпрацю.

Впровадження програмного забезпечення є завершальним етапом розробки, коли програмне забезпечення встановлюється на обчислювальну систему та готується до експлуатації. Після впровадження проводиться супровід програмного забезпечення, що включає підтримку його працездатності та надійності.

Розробники програмного забезпечення є фахівцями, які володіють знаннями та навичками у розробці програмного забезпечення для різних обчислювальних систем.

Програмне забезпечення є загальним терміном для різних видів програм, що використовуються для роботи з комп'ютерами та пов'язаними з ними пристроями. Вони дають можливість взаємодіяти з технологіями та забезпечують зручність та функціональність. Без програмного забезпечення багато комп'ютерних процесів були б складними або неможливими для виконання. Тому розробка програмного забезпечення є критично важливим елементом технологічного розвитку.

Клієнт-замовник - фізична або юридична особа, яка замовляє товари, послуги або роботи, що є цінностями, або подає заявку на їх придбання чи замовлення у майбутньому.

Результат призведе до створення сайту, що міститиме функціональні можливості для замовників та розробників програмного забезпечення з реалізованими можливостями доступу.

Вебзастосунок - це програмне забезпечення, що доступне через браузер та працює в Інтернеті. Це може бути будь-який тип сайту залежно від його класифікації.

Розроблюваний вебзастосунок буде віднесене до категорії закритих сайтів зовнішнього інтернет-представництва власників бізнесу. Воно матиме можливість надавати права доступу для замовників та розробників програмного забезпечення, які зможуть взаємодіяти з продуктом тільки після реєстрації та отримання відповідних дозволів.

Крім того, вебзастосунок може бути реалізований як інтернет-портал, вебсервіс або комбінований вебсервіс залежно від його функціональних можливостей та вимог замовників.

1.2 Огляд програмних аналогів

Системи управління проєктами є програмними засобами, призначеними для планування, координації та контролю за проєктами. Вони дозволяють командам працювати в одному місці, отримувати доступ до потрібних інструментів та ресурсів, вести облік робочого часу, виконувати завдання та ділитися інформацією.

У світі існує безліч систем управління проєктами, розглянемо три з найбільш популярних: Monday.com, Basecamp та JetBrains Space.

Monday.com - це засіб управління проєктами, що пропонує широкий спектр функцій та можливостей для команд будь-якого розміру [1]. Його функціонал включає в себе планування, управління завданнями, спільну

роботу, відстеження прогресу та аналіз даних. Monday.com приваблює користувачів своєю легкістю використання та настройки, а також інтуїтивним інтерфейсом (рисунок 1.1).

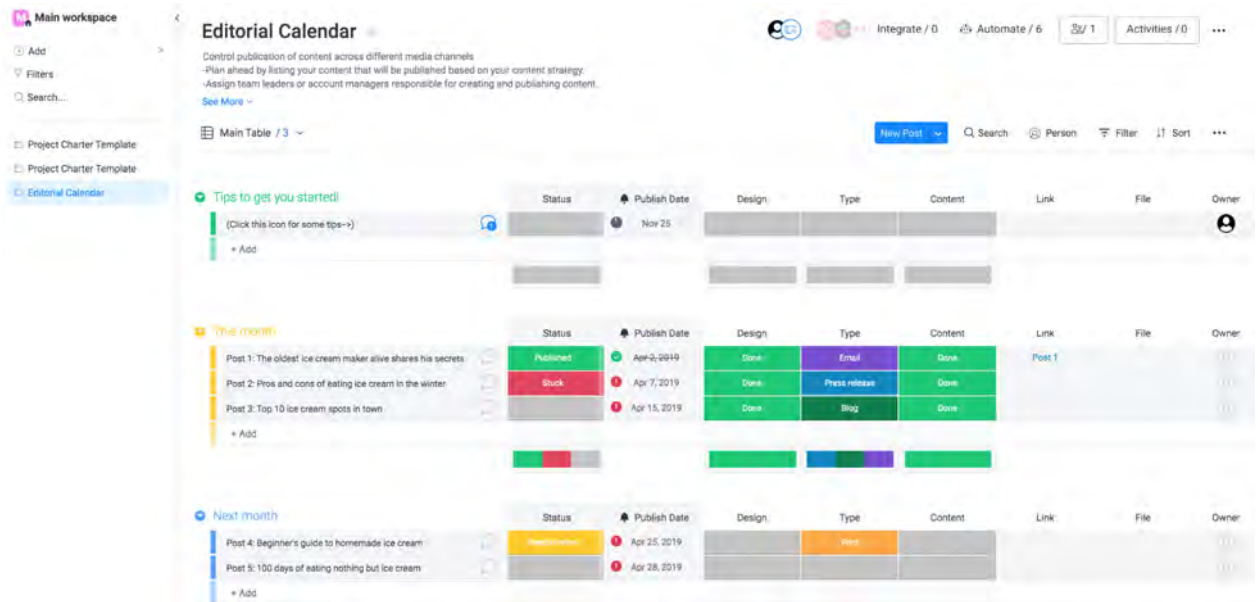


Рисунок 1.1 – Інтерфейс Monday.com

Basecamp - це проста та ефективна система управління проєктами, яка дозволяє командам легко організовувати свою роботу та зберігати всю необхідну інформацію на одній спільній платформі [2]. Basecamp пропонує інструменти для планування, спільної роботи, обміну файлами та зберігання важливої інформації в одному місці. Основна перевага Basecamp - його простота використання та легкий доступ до інформації. Інтерфейс зображено на рисунку 1.2.

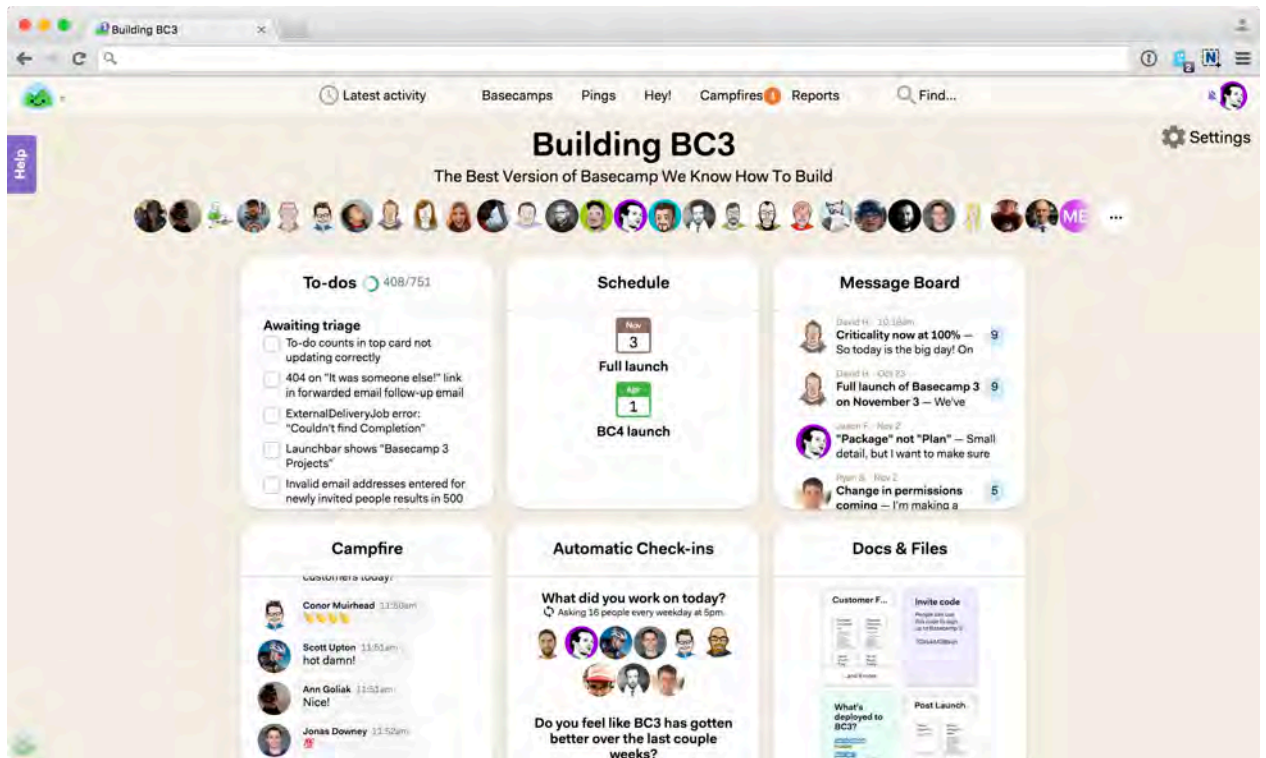


Рисунок 1.2 – Інтерфейс Basecamp

JetBrains Space - це комплексна платформа управління проектами для розробників, що забезпечує інтегрований підхід до розробки ПЗ [3]. Вона включає в себе інструменти для управління завданнями, проектами, репозиторіями, автоматизації та забезпечує відкритістю API для інтеграції з іншими системами. JetBrains Space пропонує широкі можливості для взаємодії та спільної роботи команди розробників, такі як код-ревію, автоматизоване тестування, CI/CD та інше. Інтерфейс зображено на рисунку 1.3.

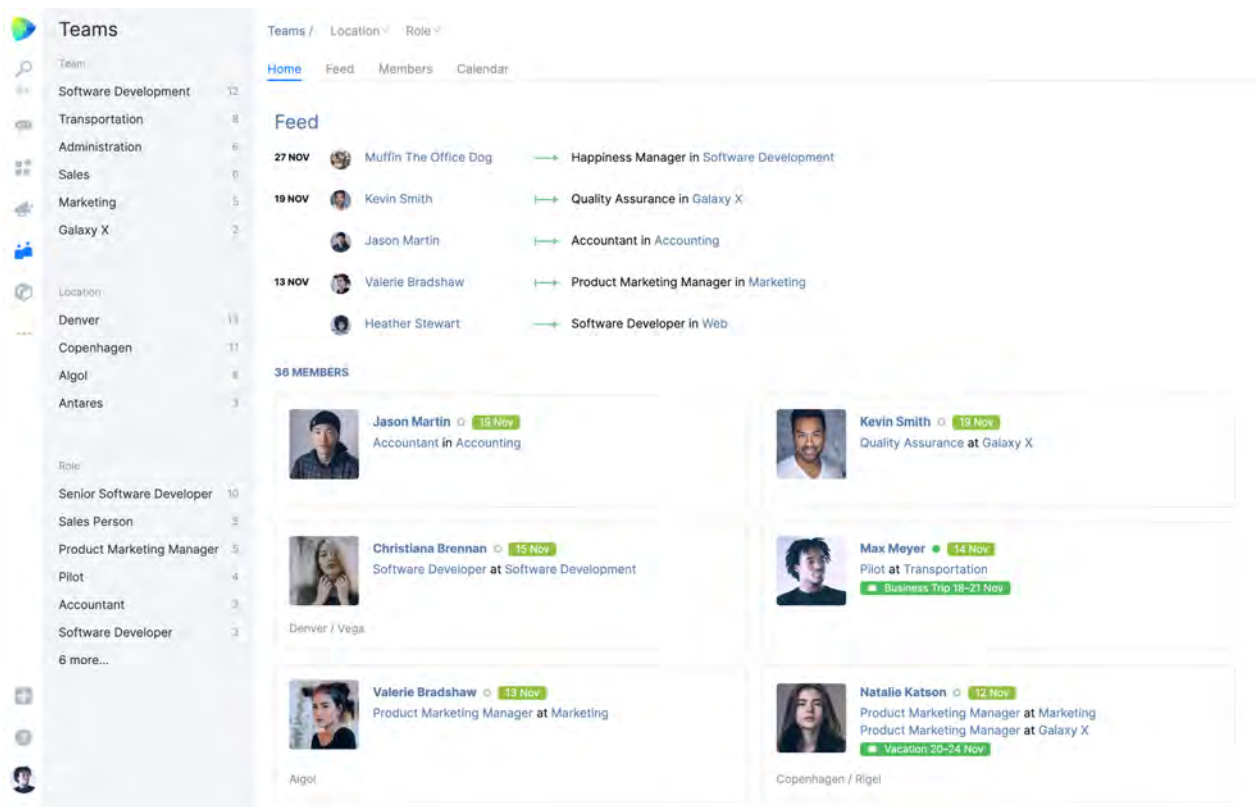


Рисунок 1.3 – Інтерфейс JetBrains Space

Отже, можна зробити висновок, що кожна з цих систем має свої переваги та недоліки, і вибір залежить від потреб команди та її масштабів. Monday.com та JetBrains Space більше придатні для великих компаній та команд розробників, які шукають інтегровані рішення для управління проєктами та розробкою ПЗ, включаючи розширення через API. Basecamp же, хоч і є простішою, але пропонує всі необхідні інструменти для спільної роботи, спілкування та організації проєктів.

Таким чином, вибір системи управління проєктами залежить від потреб команди та її масштабів, але незалежно від цього, кожна з цих трьох систем пропонує високу якість та можливості для успішного управління проєктами.

1.3 Висновки за розділом 1

В першому розділі проаналізовано предметну область, визначено основні поняття та зазначено, що цінність розроблюваного програмного забезпечення полягає в його здатності вирішувати основні завдання. Для досягнення цієї мети було проаналізовано конкурентів та виявлено, що для автоматизації роботи між замовником і розробником необхідно поєднання усіх можливостей аналогів.

Опис проєкту повинен бути доступним як для виконавців, так і для замовників, щоб задовольняти потреби обох сторін і не створювати додаткових складнощів під час використання. Окрім того, конфіденційність даних є важливим аспектом, тому застосунок має мати чіткі обмеження доступу.

Дослідження показало, що жодне з розглянутих програмних засобів не здатне забезпечити взаємодію між різними учасниками процесу, тому розроблюваний вебзастосунок має вирішувати цю проблему. Його основні завдання повинні бути спрямовані на автоматизацію роботи між замовником і розробником програмного забезпечення, забезпечення доступу до проєкту для обох сторін, забезпечення конфіденційності даних та забезпечення взаємодії між учасниками процесу. Таким чином, розроблюваний вебзастосунок має потенціал стати ефективним інструментом для автоматизації роботи замовника та розробника програмного забезпечення.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

Оскільки створення даного вебзастосунку має певну специфіку – передбачає роботу з дуже великими за розмірами наборами даних, таких як: проєкти, завдання та користувачі, з метою накопичення та обробки даних, було правильним рішенням обрати для розробки основну мову, яка на даний є дуже популярною та ефективною, – Javascript [5].

JavaScript – мова програмування, яку використовують для написання frontend- та backend-частин сайтів, мобільних застосунків.

JS підтримують усі популярні браузерери. У frontend-частини сайтів мову використовують для створення інтерактиву (анімацій, спливаючих форм, автозаповнення), оскільки він пов'язаний з HTML та CSS і може ними маніпулювати.

JS – мультипарадигменна мова, тому що вона підтримує різні парадигми (стилі) програмування:

- об'єктно-орієнтований – представляє програму як систему відносин між об'єктами – наборами даних та функцій;

- імперативний – у цьому стилі програмування використовують інструкції, які виконуються послідовно, а для доступу до даних використовуються змінні, значення яких може бути переписано під час обчислення;

- функціональний — у ньому обчислення ґрунтується на математичних функціях. Оскільки функціональне програмування не спирається на змінні, які можуть змінювати значення, результати запитів завжди будуть однаковими.

Вебсторінки частково обробляються за допомогою JavaScript на комп'ютері користувача. Це знижує навантаження на сервер: частина операцій виконується без запитів щодо нього, що заощаджує час і трафік.

У подальшому виникло питання вибору конкретного вебфреймворку серед React і Angular, за сукупністю показників, що представлені в таблиці 2.1, було обрано React.

Таблиця 2.1 – Вибір фреймворку розробки вебзастосунку

Критерій	React	Angular
Більше свободи в виборі сторонніх бібліотек	Так	Ні
Тестування	Гірше	Краще
Підтримує легку композицію компонентів	Так	Ні
Підтримує хукі для управління життєвим циклом компонентів	Так	Ні
Має вбудовані засоби для формування анімацій	Ні	Так

Для конструювання чіткої та послідовної схеми роботи та взаємодії складових React було обрано бібліотеку Redux а для посилення HTTP запитів Redux-Saga. Нижче представлено опис термінів.

Redux — це контейнер стану для застосувань JavaScript [6]. Технологія допомагає писати програми, які ведуть себе послідовно, запускаються в різних середовищах (клієнт, сервер), і їх легко перевірити. Є можливість використання Redux разом з React або з будь-якою іншою бібліотекою перегляду.

Redux-saga — це бібліотека резервного програмного забезпечення, створена для того, щоб зробити побічні ефекти (наприклад HTTP запит) у вашому застосуванні redux більш зручним та простим [7]. Це досягається

завдяки використанню генераторів (ES6), що дозволяє писати асинхронний код, який виглядає синхронним і який легко перевірити.

Проте за умови того, що у сучасній розробці програмного забезпечення використання ефективних бібліотек для управління станом застосунків є критично важливим, постає питання оптимального їх вибору.

Наразі, Redux-saga та MobX - дві популярні бібліотеки управління станом для React, що часто знаходяться в центрі обговорень вибору технології. Вони відрізняються за принципами роботи, зручністю використання, можливостями тестування та підтримкою великих застосунків. Проведемо детальне порівняння Redux-saga та MobX з метою визначення критеріїв для оптимального вибору в залежності від специфіки проєкту. Давайте розглянемо їх порівняння в таблиці 2.2:

Таблиця 2.2 – Порівняння Redux-saga та MobX

Критерій	Redux-saga	MobX
Використовує єдине глобальне сховище для усіх даних	Так	Ні
Легко тестувати та відлагоджувати, завдяки чистим функціям і використанню генераторів ES6	Так	Ні
Парадигма	Flux-архітектура, базується на діях, змінниках і сховищі	Реактивне програмування, базується на спостережуваних значеннях і діях
Продуктивність	Кращий для великих застосунків з великим обсягом даних	Кращий для невеликих та середніх застосунків
Мутація стану	Ні, стан незмінний	Так, дозволяє пряму мутацію стану

Після детального порівняння Redux-saga та MobX, можна зробити висновок, що Redux-saga показує себе як більш надійний інструмент. Вона пропонує строгу організацію стану, підтримку чистих функцій, прогнозованість та кращі можливості для тестування і відлагодження. Хоча Redux-saga може бути більш вимогливою до вивчення, вона надає сильну основу для стабільності та масштабування великих React-застосувань.

2.2 Визначення функціональних вимог до програмного забезпечення

Програмне забезпечення для автоматизації роботи між замовником і розробником програмного забезпечення повинно надавати можливість роботи з цілим набором функціоналу, а саме: створення проєктів розробника, виведення на головну сторінку усіх проєктів користувача, зміна статусу проєктів, редагування налаштувань необхідних при розробці, зручне маніпулювання даними (редагування, додавання, видалення інформації);, можливість завантаження необхідних файлів до проєкту, зміна власних налаштувань користувача, також має бути реалізовано чат для обміну інформацією між клієнтом та розробником.

У відповідності з даними функціональними можливостями визначено можливості замовника, описані у переліку, візуально зображені на рисунку 2.1:

- можливість перегляду проєктів та їх налаштувань;
- можливість завантаження необхідних файлів до проєкту;
- використання чату для обміну інформацією.



Рисунок 2.1 – Діаграма можливостей замовника

Можливості розробника описано у переліку, наведеному нижче та відображено на рисунку 2.2:

- виведення на головну сторінку усіх проєктів розробника;
- редагування даних проєкту;
- можливість завантаження необхідних файлів до проєкту;
- зручне маніпулювання даними (редагування, додавання, видалення інформації);
- зміна налаштувань облікового запису;
- зміна налаштувань розроблюваних проєктів;
- використання чату для обміну інформацією.

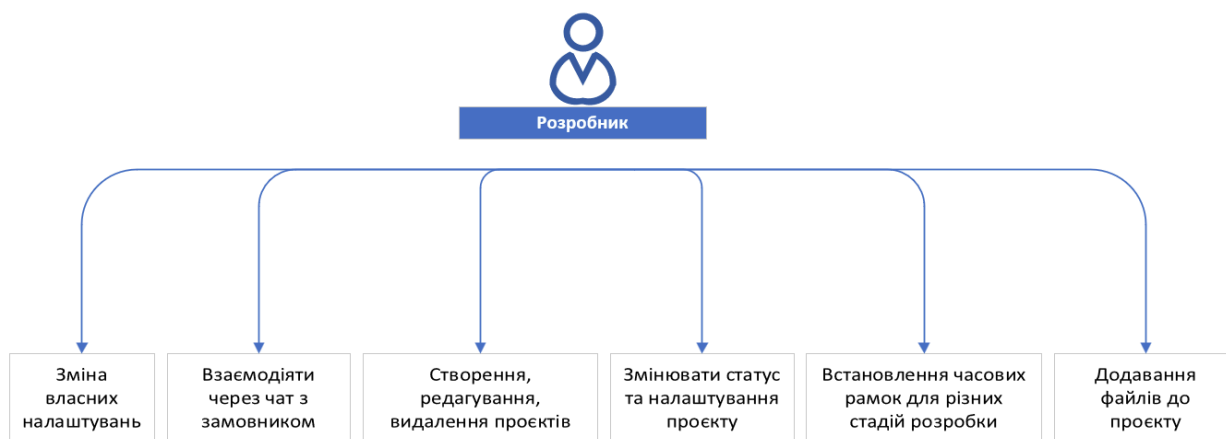


Рисунок 2.2 – Діаграма можливостей розробника

2.3 Проектування бази даних

Існують наступні види баз даних: реляційні та нереляційні. Розглянемо їх особливості, переваги та недоліки.

Реляційні (SQL) сховища – прості, стійкі, гнучкі, продуктивні, масштабовані, сумісні, проте громіздкі при зростанні кількості даних, що зберігаються.

Нереляційні (NoSQL) бази даних є документо-орієнтованими, схожими на папки, наповнені файлами. Вони краще масштабуються, не мають обмежень щодо типів, які можна зберігати разом, мають швидшу обробку даних, але є ризик того, що дані одного вузла можуть не синхронізуватися з даними іншого.

Переваги SQL бази даних:

- атомарність, узгодженість, ізолюваність;
- кожна таблиця містить одну, декілька категорій даних у стовпцях;
- кожен рядок містить унікальний екземпляр даних для категорій-стовпців;
- користувач може мати доступ до даних з бази даних, не знаючи структури таблиці.

Недоліки SQL бази даних:

- масштабування можливе лише на потужних серверах;
- дорогі системи зберігання.

Переваги NoSQL бази даних:

- можливе зберігання погано структурованих обсягів даних;
- швидка обробка даних;
- дешевші в обслуговуванні;
- можливість масштабування.

Недоліки NoSQL бази даних:

- узгодженість даних, ризики недосконалої синхронізації вузлів;

- відсутня стандартизація;
- погано розповсюджуються на інші вузли.

В якості бази даних для застосування було обрано NoSQL базу даних MongoDB, тому що є необхідність зберігати і структурувати інформацію, що має відмінні типи, необхідна можливість подальшого масштабування.

За результатами вивчення предметної області, виконаного в розділі 1, для побудови об'єктно-орієнтованої моделі було виділено наступні об'єктні множини:

- user – інформація про облікові записи користувачів;
- client – інформація про клієнтів;
- estimator – інформація про розробників;
- project – інформація про проєкти;
- section – інформація про секції розробки проєкту;
- task – інформація про пункти секції;
- settings – інформація про базові налаштування розробки проєктів, що належать певному розробнику;
- attachment – інформація про закріплені до проєкту файли;
- message – інформація про повідомлення;
- additional – інформація про налаштування проєкту;
- library – інформація про бібліотеки секцій користувача;
- lebrarySection – інформація про секції бібліотек користувача;
- libraryItem – інформація про пункти секцій бібліотек користувача;
- token – інформація про токени доступу користувача;
- emailToken – інформація про токени доступу для зміни пароля користувача;
- clientToken – інформація про токени доступу для клієнта.

Спроектowana об'єктно-орієнтована модель зображена на рисунку 2.3.

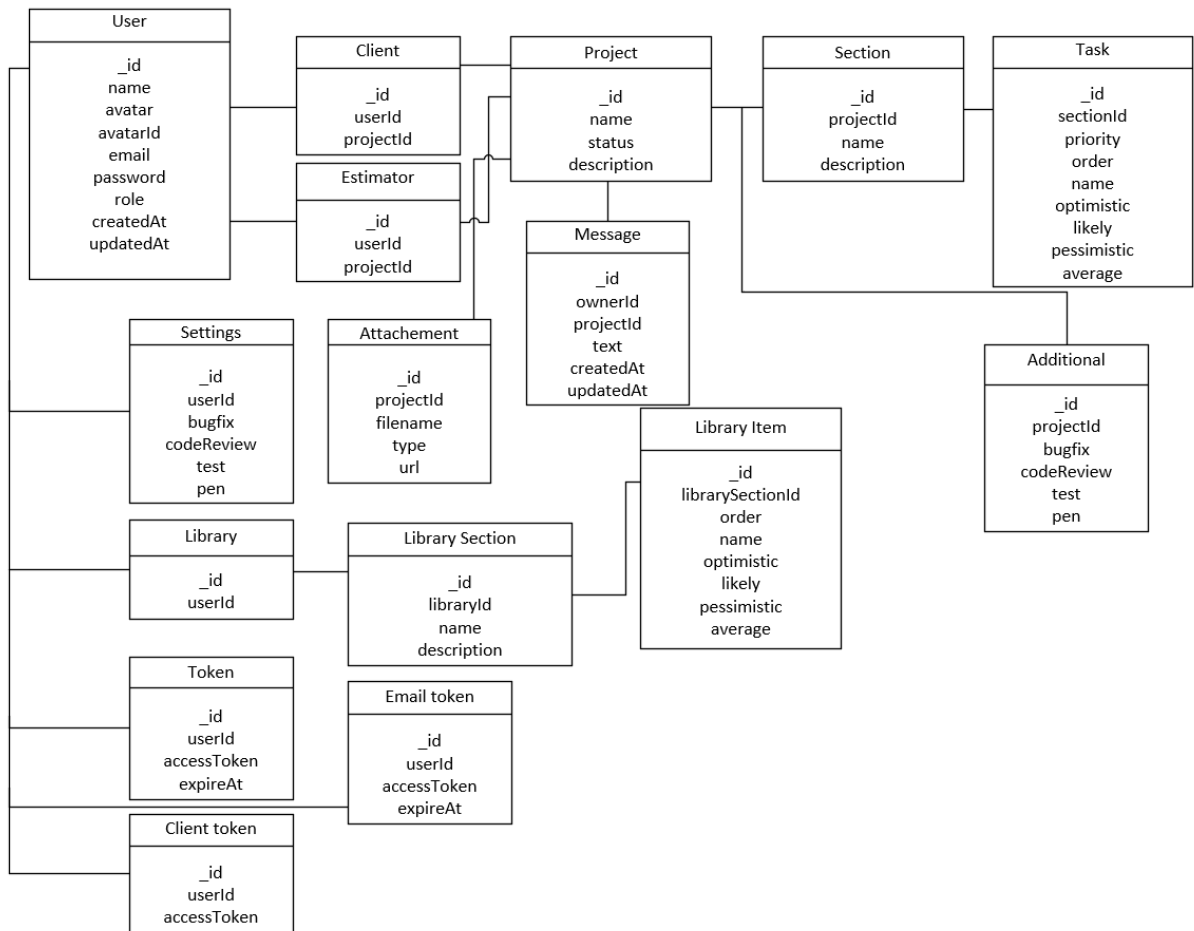


Рисунок 2.3 – Спроектована об'єктно-орієнтована модель

2.4 Визначення структури програми

Структурно клієнтська частина складається з наступних модулів:

- actions – описані основні події;
- components – описані основні компоненти застосування;
- config – містяться компоненти конфігурації ПЗ;
- constants – описані основні константи;
- containers – містяться контейнери даних;
- reducers – описані обробники даних;
- router – містяться усі шляхи сайту;
- saga – описані функції роботи з сервером;
- services – описані допоміжні функції роботи ПЗ;
- store – містяться змінні для тимчасових об'єктів;

- style – описані стилі інтерфейсу;
- tests – описані обробники результатів функцій;
- utils – описані утиліти для роботи клієнтської частини.

Візуальне відображення клієнтської частини здійснюється за допомогою React-компонентів, які описані у components. Дії з компонентами можливі завдяки функціям, їм властивим.

Відповідно до структури ПЗ, використовуючи Redux-Saga, кожна функція викликає відповідний метод, описаний у actions, після чого відбувається виклик генератора з saga, де відбувається запит до сервера, а згодом наступний виклик необхідної функції actions для роботи зі сховищем store, де доступ здійснюється за допомогою reducers завдяки переданим константам від actions (константи описані у constants). Ендпоінти описані у router, завдяки яким відбувається відображення компонентів, що обернені в containers. Алгоритм зображено на рисунку 2.4.

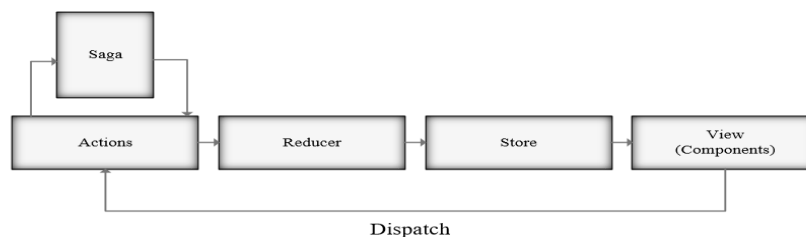


Рисунок 2.4 – Структурна схема клієнта

Клієнт та сервер спілкуються відповідно до архітектури REST. Кожен елемент інформації має своє відповідне посилання. Дії, пов'язані з даними, визначаються за допомогою методів HTTP-запиту (GET, POST, PUT, DELETE).

2.5 Висновки за розділом 2

Після порівняння з фреймворком Angular, React було обрано для розробки як найбільш популярний та ефективний інструмент для створення вебзастосунків на мові JavaScript. Встановлено основні сценарії взаємодії замовника і розробника з ПЗ. Створено модель бази даних та визначено архітектуру програми.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Основна функціональність програми для користувачів пов'язана з відображенням конкретної інформації, що відповідає їхній ролі. У цьому розділі детально описано логіку роботи розробників і замовників для визначення алгоритму функціонування застосування.

У процесі роботи з програмою, розробник має доступ до важливих функцій: від створення, редагування, видалення проєктів до встановлення часових рамок для різних стадій розробки. Розробник також може додавати файли, змінювати статус та налаштування проєкту, а також взаємодіяти через чат. Окрім цього, він має змогу керувати своїм профілем, включаючи зміну паролю, електронної пошти, аватару і так далі.

Замовник може вільно переглядати інформацію про проєкт, додавати необхідні файли, а також користуватися чатом для обміну повідомленнями з розробником. Щоб отримати доступ до цього вебзастосунку, замовник має пройти авторизацію, перейшовши по посиланню, яке він отримує на свою електронну пошту.

3.2 Опис реалізованих складових вебзастосунку

Розроблюване застосування надсилає запити на сервер за допомогою Axios. Axios — HTTP-клієнт для браузера. При чому цей запит стає побічним ефектом, саме тому було використано можливості Redux-Saga (рисунок 3.1).

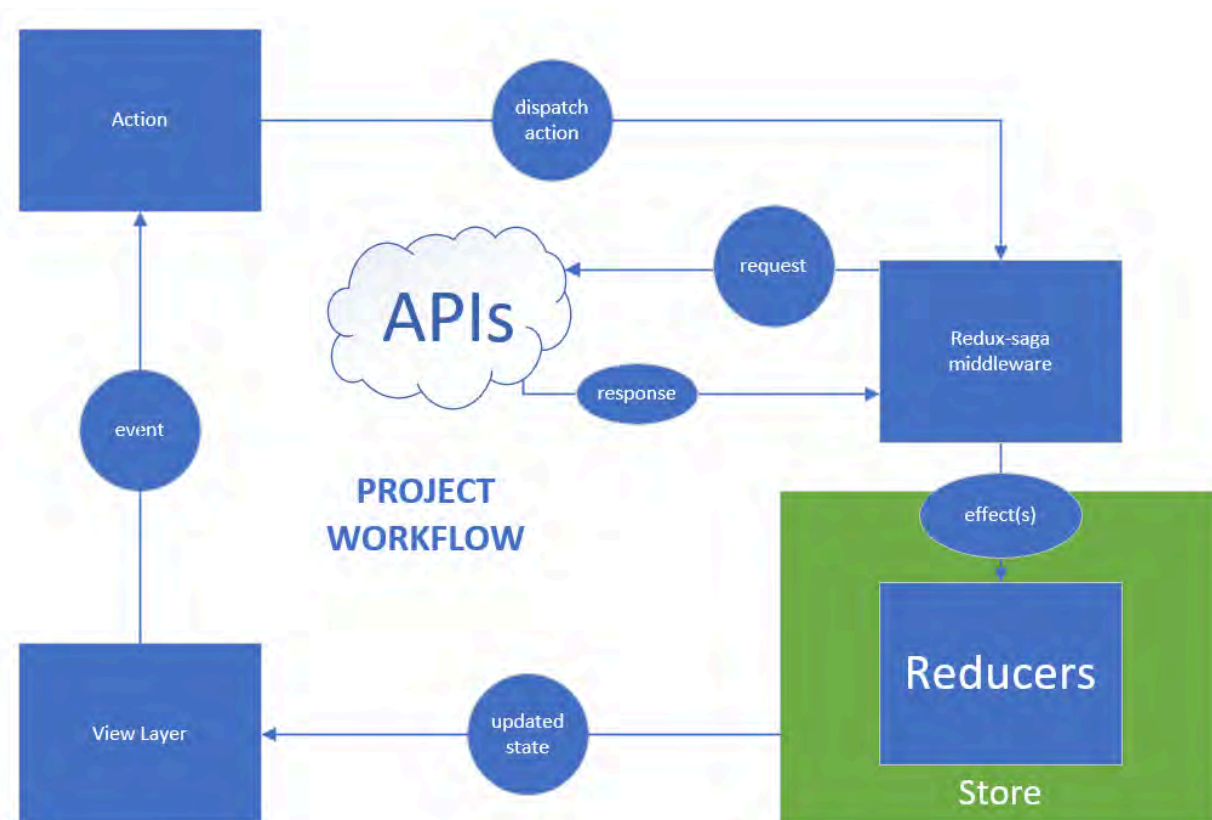


Рисунок 3.1 — Схема Redux-Saga

Існує два способи розробки вебзастосунку: односторінкові застосування (SPA) і багатосторінкові застосування (MPA) [8].

Односторінкові вебзастосунки дозволяють імітувати роботу десктоп-застосунків. Архітектура побудована таким чином, що при переході на нову сторінку, оновлюється тільки частина контенту. Таким чином, відсутня необхідність повторного завантажування елементів, дуже зручно для розробників і користувачів.

Розроблюваний проєкт було вирішено робити SPA-застосуванням, бо SPA характеризуються відмінною швидкістю, тому що більшість ресурсів, які вони використовують, завантажуються лише один раз протягом сесії використання застосування, SPA більш ефективні в кешуванні даних на локальних носіях.

Застосування має наступну схему роутів, що зображено на рисунку 3.2:

```

export default class MainRouter extends Component {

  render() {
    return (
      <Router history={history}>
        <Switch>
          <Route exact path="/" component={projects}/>
          <Route exact path="/login" component={LogIn} />
          <Route exact path="/signup" component={Signup}/>
          <Route exact path="/forgotpassword" component={forgotPassword} />
          <Route exact path="/setpassword" component={setPassword} />
          <Route exact path="/checkemail" component={checkEmail}/>
          <Route exact path="/profilesettings" component={profileSettings}/>
          <Route exact path="/settings" component={settings}/>
          <Route exact path="/projectpage/:projectId" component={projectPage}/>
          <Route exact path="/customerpage" component={customerView}/>
          <Route exact path="/customerpage/project/:projectId/:token" component={customerProjectPage}/>
        </Switch>
      </Router>
    );
  }
}

```

Рисунок 3.2 — Схема роутів

Кожен роут (шлях) відображає певний компонент, де виконуються певні HTTP-запити, надаючи дані частинами, що відповідає концепції SPA.

Шлях «/» відображає сторінку з усіма проектами користувача.

Шлях «/login» відображає сторінку авторизації.

Шлях «/signup» відображає сторінку реєстрації.

Шлях «/forgotpassword» відображає сторінку у разі забутого паролю.

Шлях «/setpassword» відображає сторінку відновлення паролю.

Шлях «/checkemail» відображає сторінку з повідомленням про необхідність перевірки пошти.

Шлях «/profilesttings» відображає сторінку з особистими налаштуваннями користувача.

Шлях «/settings» відображає сторінку з налаштуваннями проєктів користувача.

Шлях «/projectpage/:projectId» відображає сторінку обраного проєкту.

Шлях «/customerpage» відображає сторінку замовника.

Шлях «/customerpage/projectpage/:projected/:token» відображає сторінку певного проєкту замовника.

У контексті дослідження для цієї дипломної роботи, спочатку було використана бібліотека Axios для виконання HTTP-запитів. Проте, було виявлено, що Axios виявився недостатньо підходящим для завантаження зображень та файлів на сервер через його обмежену підтримку об'єкту FormData.

Об'єкт FormData є складовою веб API, який спрощує підготовку даних до відправки у форматі, схожому на форму. На відміну від Axios, Fetch API надає пряму підтримку об'єкту FormData. Це дозволяє легко створювати об'єкт FormData, додавати до нього файли та інші дані, а потім використовувати цей об'єкт як тіло запиту Fetch [9].

Axios, у свою чергу, не підтримує об'єкт FormData безпосередньо, як це зроблено в Fetch. В результаті, при використанні FormData з Axios, потрібно було явно встановити заголовки Content-Type та boundary. Без цих додаткових кроків, Axios може неправильно обробити об'єкт FormData, перетворивши його на рядок і призводячи до помилок при обробці на сервері.

Таким чином, розроблена реалізація завантаження файлів з допомогою Fetch представлена на рисунку 3.3

```

const handleChangeFile = async (event) => {

  const files = event.target.files;
  const fileToUpload = files[0];

  const token = localStorage.getItem('token');

  const formData = new FormData();
  formData.append('file', fileToUpload);

  try {
    await props.checkExpiredUserToken(token);
    const res = await fetch("http://localhost:3005/setavatar", {
      method: "POST",
      headers: {
        'Authorization': token,
      },
      body: formData,
    });
    if (res) {
      props.getUserData(token);
    }
  } catch (error) {
    throw (error)
  }
}

```

Рисунок 3.3 – Функціонал завантаження файлів з використанням бібліотеки fetch

У сучасному веброзробництві, бібліотека Redux-Form є потужним інструментом для управління станом форми в React-застосунках. Ця бібліотека дозволяє розробникам ефективно управляти станом форми, забезпечуючи високий рівень контролю та гнучкості.

Redux-Form використовує концепцію високорівневих компонентів для зв'язування форми з Redux Store. Це дозволяє нам зберігати стан форми в Store і використовувати Redux для управління цим станом. Централізоване управління станом форми з допомогою Redux-Form дозволяє легко виконувати різні завдання, такі як валідація, слідкування за станом полів форми і подібне [10].

Подивимося на конкретний випадок, де було використано Redux-Form у вебзастосунку, що розробляється. Одним з найбільш поширених випадків застосування Redux-Form є форми авторизації. Користувачам потрібно

заповнювати ці форми для входу в систему. Redux-Form може допомогти нам управляти станом цих форм, виконувати валідацію на стороні клієнта та обробляти відправку форми.

Redux-Form має вбудовану систему валідації. Є можливість визначити власні функції валідації для кожного поля в нашій формі. Наприклад, для форми авторизації можемо мати валідаційні правила для полів e-mail та пароль, що зображені на рисунку 3.4.

```
const validate = values => {  
  
  const errors = {}  
  
  if (!values.email) {  
    errors.email = 'Required';  
  } else if (!/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$/i.test(values.email)) {  
    errors.email = 'Invalid email address';  
  }  
  
  if (!values.password) {  
    errors.password = 'Required';  
  } else if (!schema.validate(values.password)) {  
    errors.password = 'Invalid password';  
  }  
  
  return errors  
}
```

Рисунок 3.4 – Правила валідації для форми авторизації

Якщо валідація не пройдена, повертається об'єкт з відповідними повідомленнями про помилки.

Коли в формі виникають помилки валідації, застосунок може відображати відповідні повідомлення про помилки. Redux-Form автоматично передає об'єкт помилок до форми, і в подальшому можна використовувати цей об'єкт для відображення повідомлень про помилки.

Для розробки інтерфейсу було обрано фреймворк Material-UI [11], який містить набір готових стилізованих React-компонентів, що дає змогу

зробити застосування з урахуванням вимог usability. Для розробки сайту було використано наступні компоненти, представлені у таблиці 3.1.

Таблиця 3.1 – Компоненти, використані при побудові інтерфейсу

Компонент	Опис призначення
AppBar	У верхній панелі застосувань відображається вміст і дії, пов'язані з поточним екраном. Вона використовується для брендингу, заголовків екрану, навігації і дій. Вона може використовуватися як контекстне меню або як навігаційна панель.
Typography	Дозволяє представити дизайн та текстовий вміст максимально чітко та ефектно.
IconButton	Дозволяє користувачам здійснювати дії та робити вибір одним натисканням на іконку
MenuItem	Дозволяє користувачам здійснювати дії та робити вибір одним натисканням на іконку у меню
FormLabel	Дозволяє представити дизайн та текстовий вміст максимально чітко та ефектно в межах форми
Divider	Тонка лінія, яка групує вміст у списки та макети
List	Суцільні, вертикальні покажчики тексту чи зображень
Box	Служить компонентом обгортки для більшості потреб CSS
Tabs	Полегшує переключення між різними вкладками
Tab	Слугує для відображення вкладинок
Avatar	Слугує для виведення аватару користувача
Drawer	Забезпечують доступ до пунктів у вашому застосуванні. Бічні аркуші - це поверхні, що містять додатковий вміст, які прикріплені до лівого або правого краю екрана.

3.3 Висновки за розділом 3

У даному розділі описано основні реалізовані одиниці програми з деталізацією їх структури та принципів роботи у проєкті.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис програми

Високоякісний діалог між замовником та розробником програмного забезпечення відіграє важливу роль, оскільки він сприяє підвищенню продуктивності та якості розробки. Автоматизація цього процесу, що є ціллю даної дипломної роботи, здійснена за допомогою спеціалізованого програмного забезпечення.

Вебзастосунок дозволяє автоматично контролювати статус проєктів, мати розуміння строків виконання, вести облік часу та розв'язувати конфліктні ситуації, що мінімізує можливість непорозумінь і покращує швидкість та якість виконання завдань.

Існують подібні програмні рішення, однак вони часто не надають зручних інструментів для взаємодії та відповідного зворотного зв'язку, наприклад, у формі чату. Такий зв'язок є важливим, оскільки він забезпечує ефективну та швидку взаємодію між розробником та замовником. Саме реалізація цієї функціональності є головним напрямом у роботі над розроблюваним застосунком.

Для нормальної роботи клієнтської частини потрібний персональний комп'ютер з доступом до мережі Інтернет, апаратна платформа повинна відповідати мінімальним системним вимогам браузера, в якому планується використання сайту.

Застосування нормально функціонуватиме під керуванням будь-якої операційної системи.

Для комп'ютерів з операційною системою Windows мінімальні вимоги наступні:

- Windows XP (2 пакет оновлень) або більш пізньої версії;
- процесор Intel i3 та аналогічні з підтримкою SSE2 [12].

Рекомендовані характеристики системи:

- Windows 7, Windows 8, Windows 8.1, Windows 10 або більш пізньої версії;

- процесор Core i5 2-го покоління (від 2 ГГц), Core i5 3-го або 4-го покоління чи аналогічні.

Комп'ютери з операційною системою MacOS повинні мати:

- OS X Yosemite 10.09 або більш пізньої версії.

Рекомендовані характеристики системи:

- Macbook Pro 2011, Macbook Air 2012 або більш нові версії.

Вимоги для пристроїв з операційною системою Linux:

- Ubuntu 14.04 (64-розрядної версії) або старше, Debian 8 або старше, openSUSE 13.3 або старше, Fedora Linux 24 або старше;

- процесор Intel Pentium 4 або старше з підтримкою SSE2 [13].

4.2 Експлуатація програми

Для використання сайту розробнику необхідно перейти за відповідним посиланням, при цьому з'явиться сторінка авторизації, зображено на рисунку 4.1.

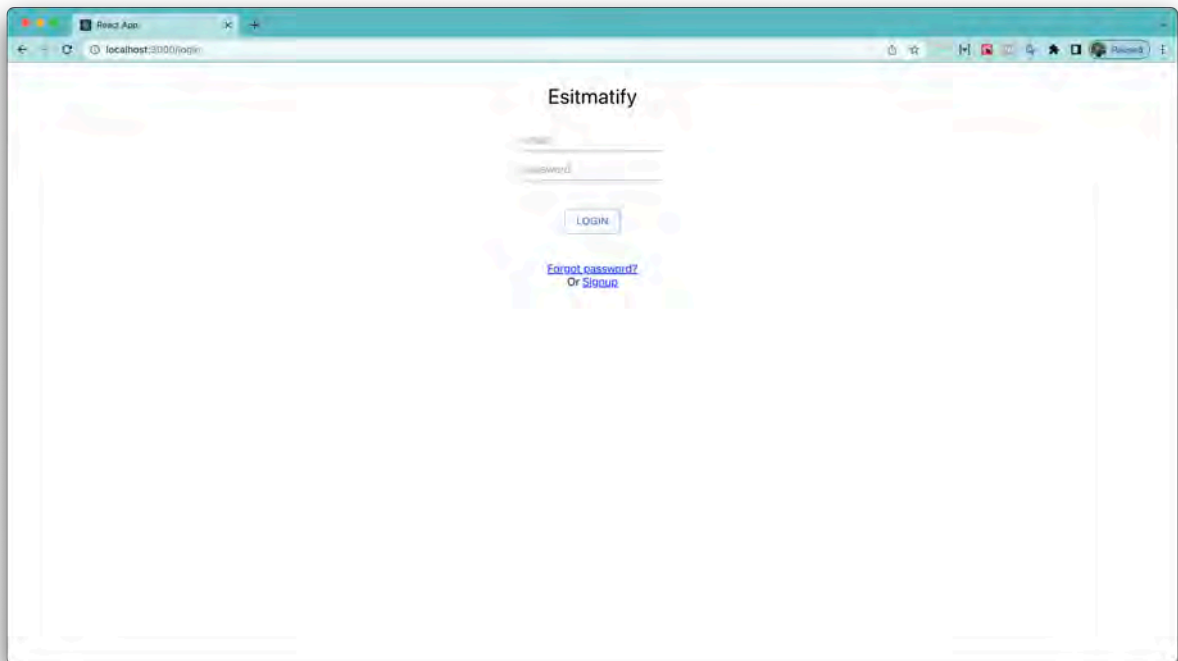


Рисунок 4.1 – Сторінка авторизації

Якщо облікового запису немає, то можна натиснути на посилання Signup та зареєструватися, заповнивши відповідні поля, зображено на рисунку 4.2.

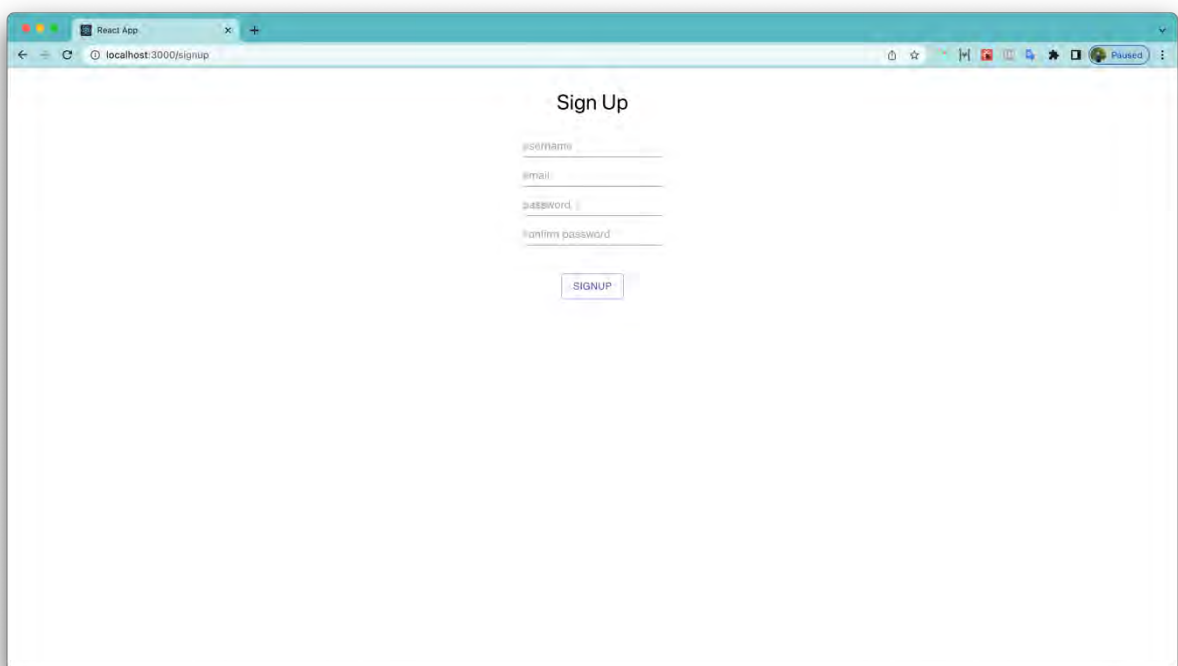


Рисунок 4.2 – Сторінка реєстрації

Якщо обліковий запис є, але користувач забув пароль, то передбачено його відновлення за допомогою електронної пошти. Сторінку зображено на рисунку 4.3.

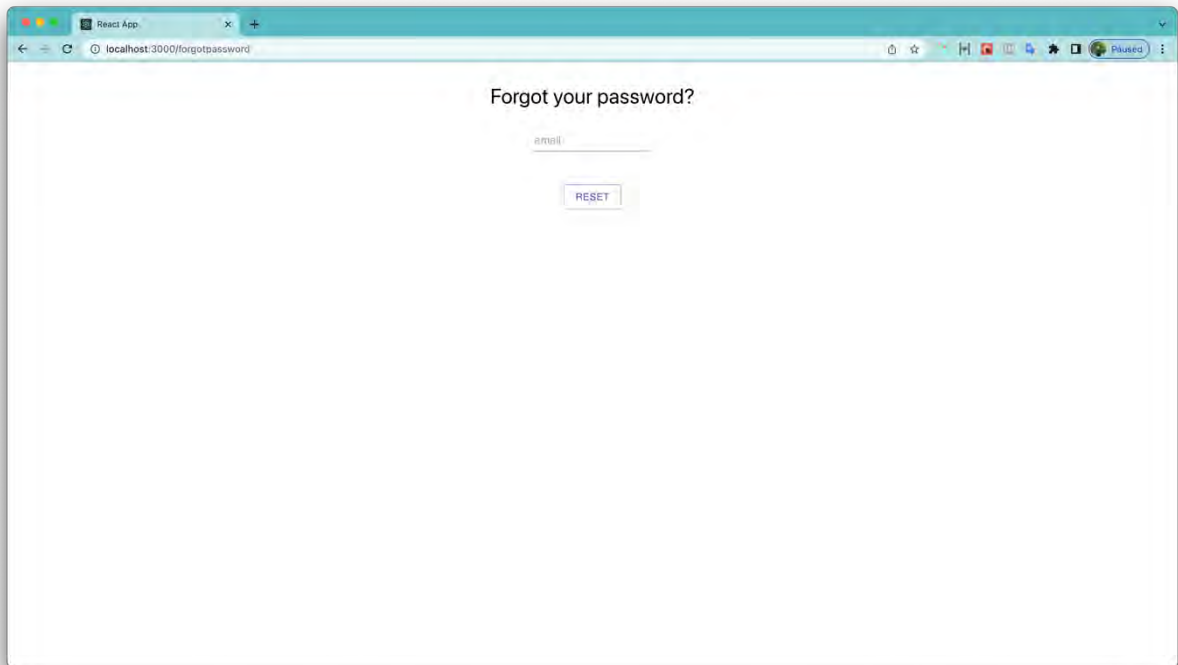


Рисунок 4.3 – Сторінка відновлення пароля

Після успішної авторизації відбувається перехід на головну сторінку сайту, де розміщені усі проєкти користувача. Сторінку зображено на рисунку 4.4.

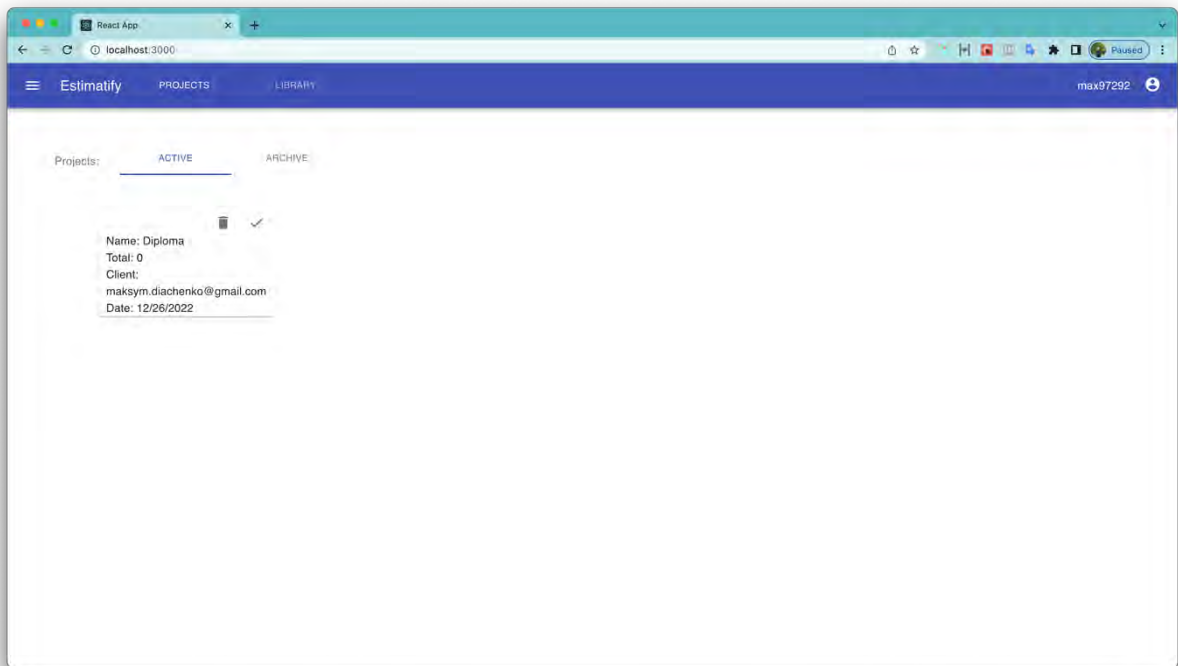


Рисунок 4.4 – Головна сторінка сайту

Натиснувши на кнопку у лівій частині екрану, відкриється меню, за допомогою якого можна обрати, який потрібний функціонал застосування. Наприклад, для перегляду та редагування облікового запису необхідно натиснути пункт Profile, модальне вікно зображено на рисунку 4.5.

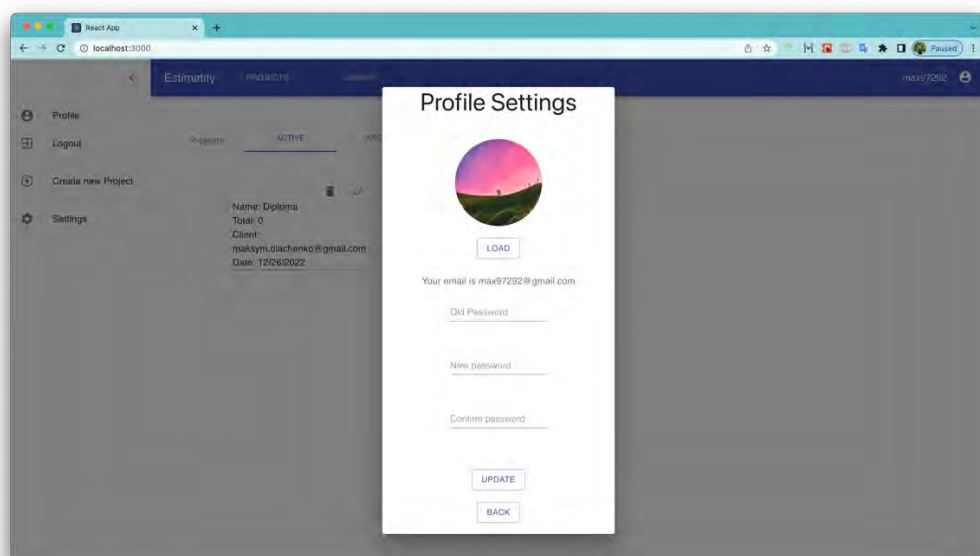


Рисунок 4.5 – Модальне вікно для редагування облікового запису користувача

При натисканні пункту Logout відбувається вихід з облікового запису та перехід на сторінку авторизації.

Якщо натиснути пункт Create new Project, то відкриється модальне вікно для створення проєкту, де необхідно заповнити відповідні поля. Модальне вікно зображено на рисунку 4.6.

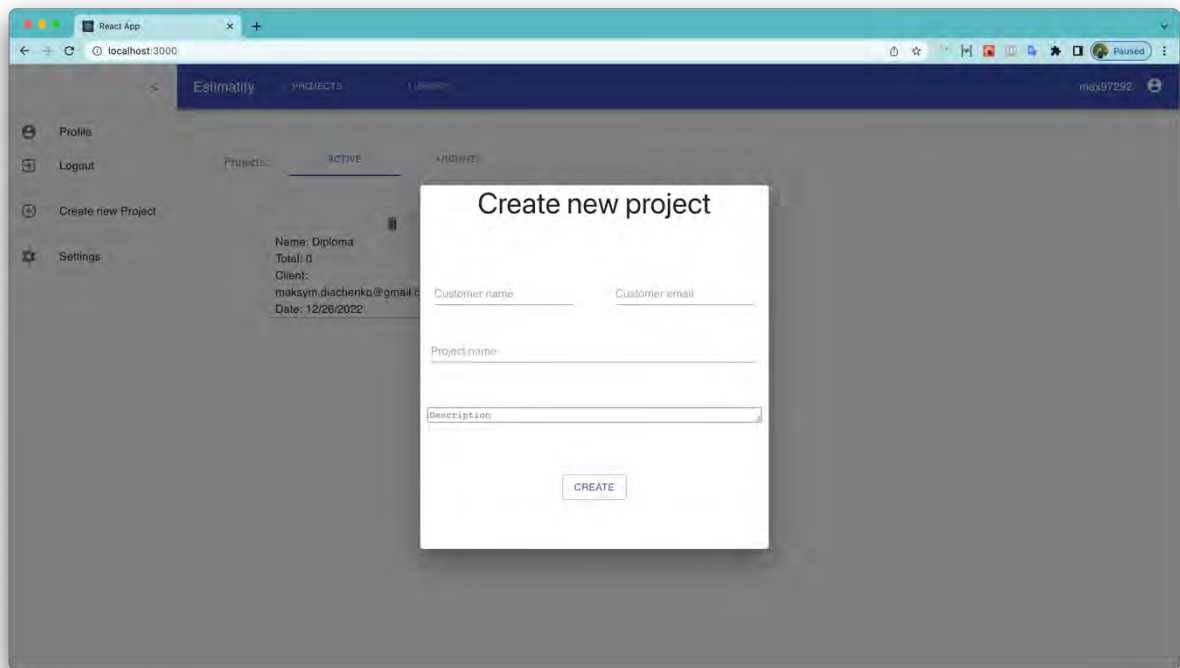


Рисунок 4.6 – Модальне вікно створення проєкту

Для зміни налаштувань створюваних проєктів необхідно натиснути на пункт Settings та змінити значення у полях модального вікна, що зображено на рисунку 4.7.

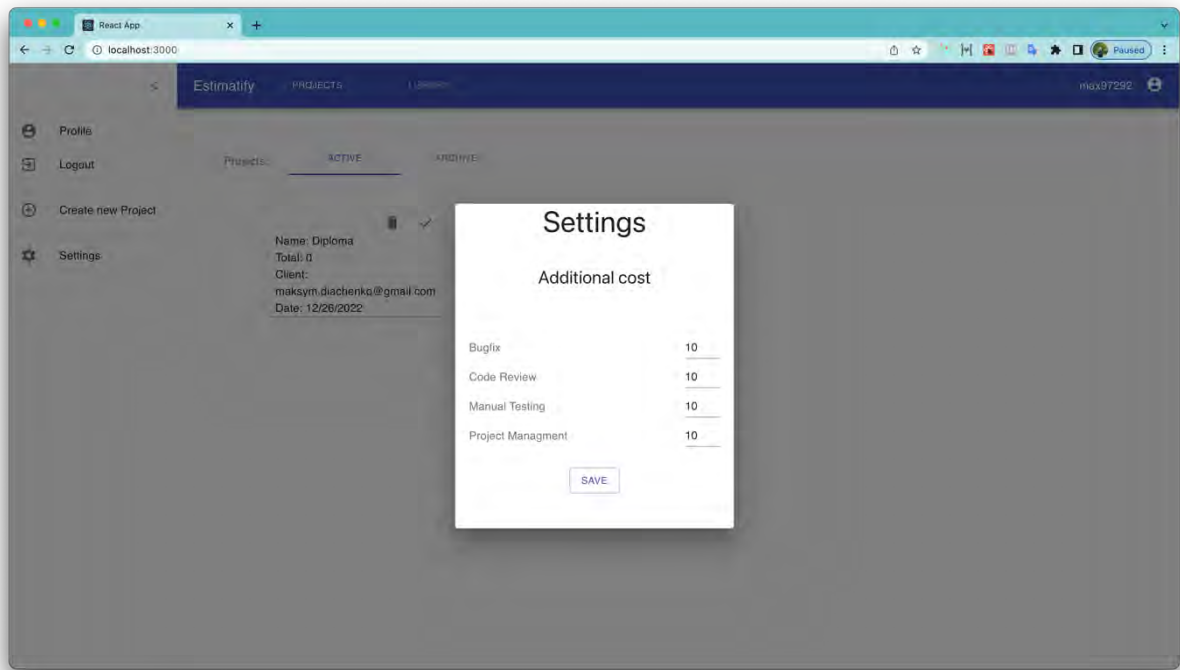


Рисунок 4.7 – Модальне вікно налаштувань проєктів

Якщо необхідно переглянути більш детальну інформацію про проєкт, то треба натиснути на поле необхідного проєкту, після чого відкриється сторінка проєкту, що зображена на рисунку 4.8.

Є можливість редагування даних, додавання, завантаження та видалення закріплених файлів до проєкту.

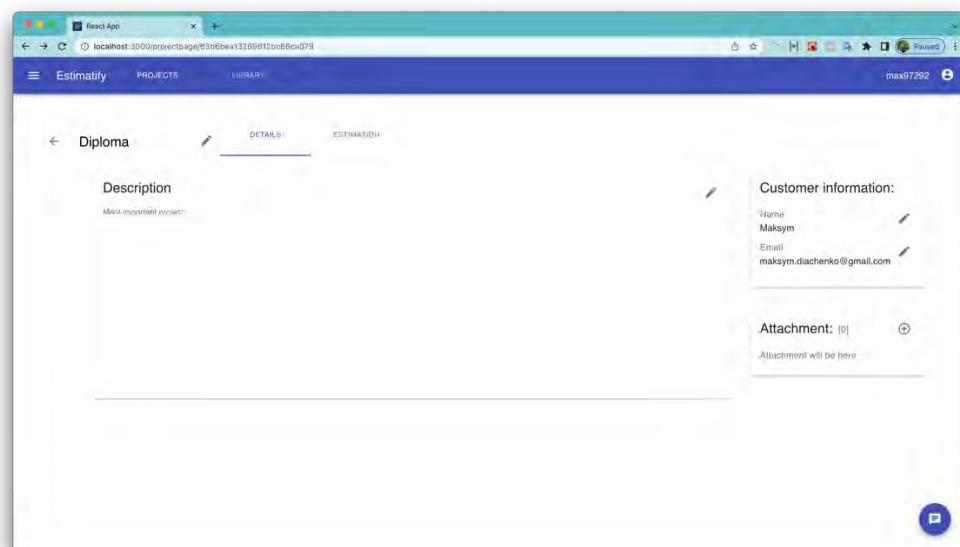


Рисунок 4.8 – Сторінка проєкту

При виборі вкладки Estimation можна продивитися стан та умови розробки проєкту. Сторінка зображена на рисунку 4.9.

Є можливість додавання, видалення, редагування, завантаження з Library секцій, а також додавання, редагування, видалення, завантаження з Library, зміни порядку та пріоритетності пунктів секцій.

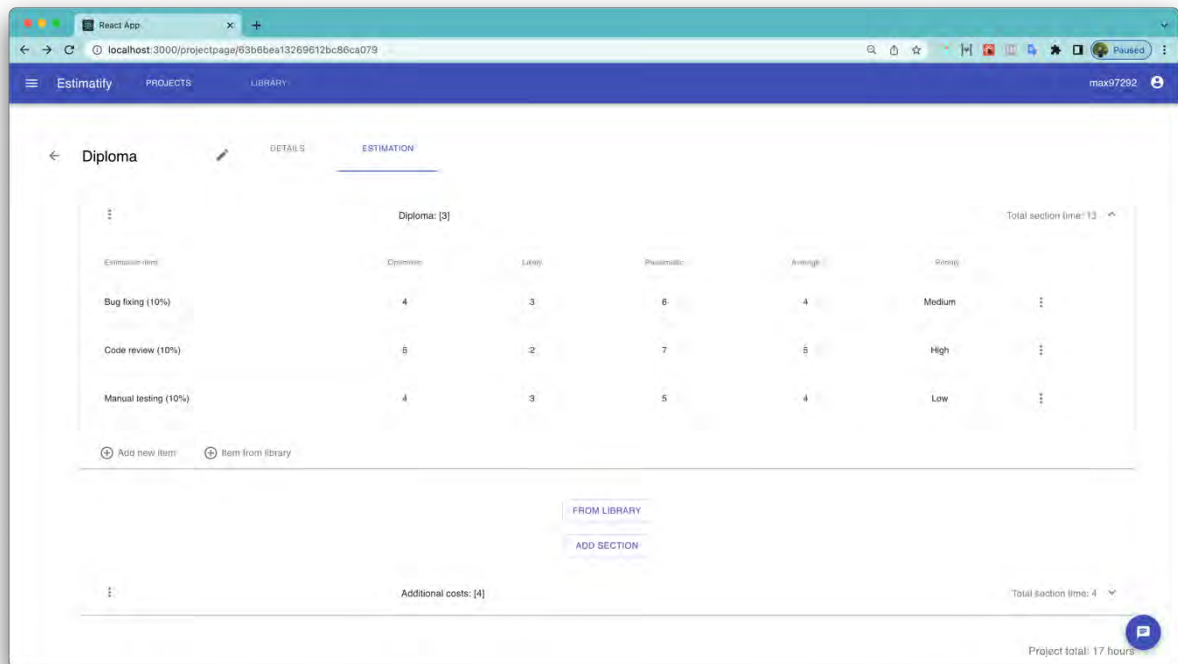


Рисунок 4.9 – Сторінка інформації про розробку проєкту

Для перегляду бібліотек, важливих певному користувачу, необхідно натиснути на пункт верхньої панелі Library. Сторінка зображена на рисунку 4.10.

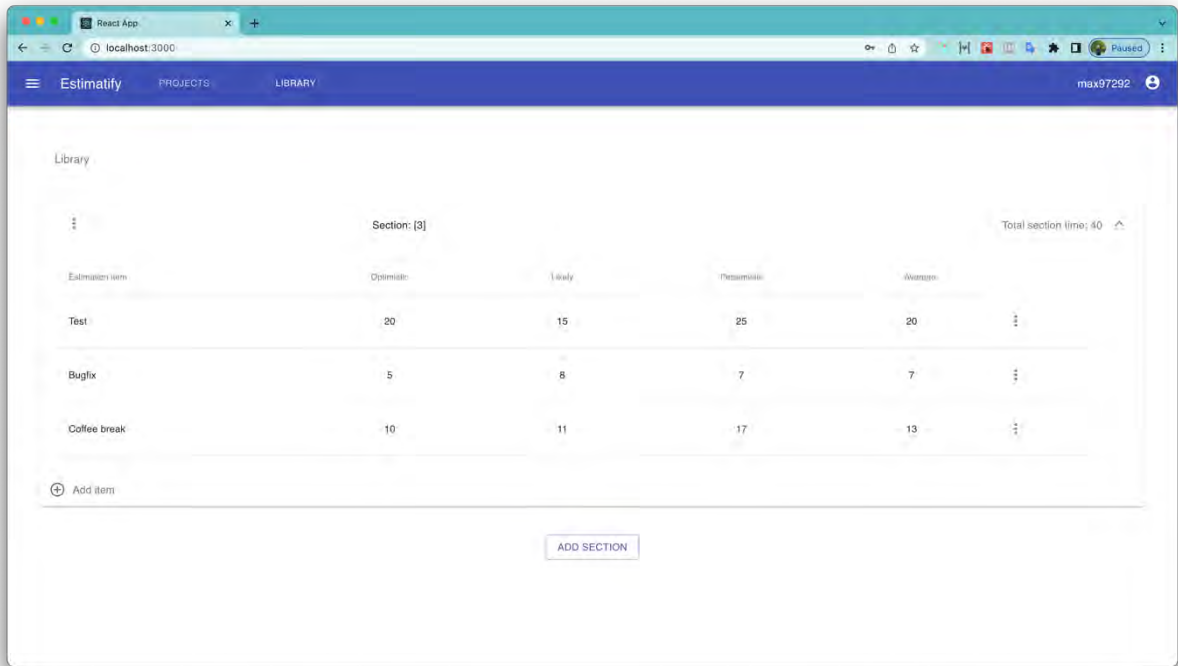


Рисунок 4.10 – Сторінка бібліотек користувача

Клієнт може переглянути проєкт та список проєктів, перейшовши за посиланням, що надійде йому на електронну пошту під час створення проєкту. Вигляд повідомлення зображено на рисунку 4.11.

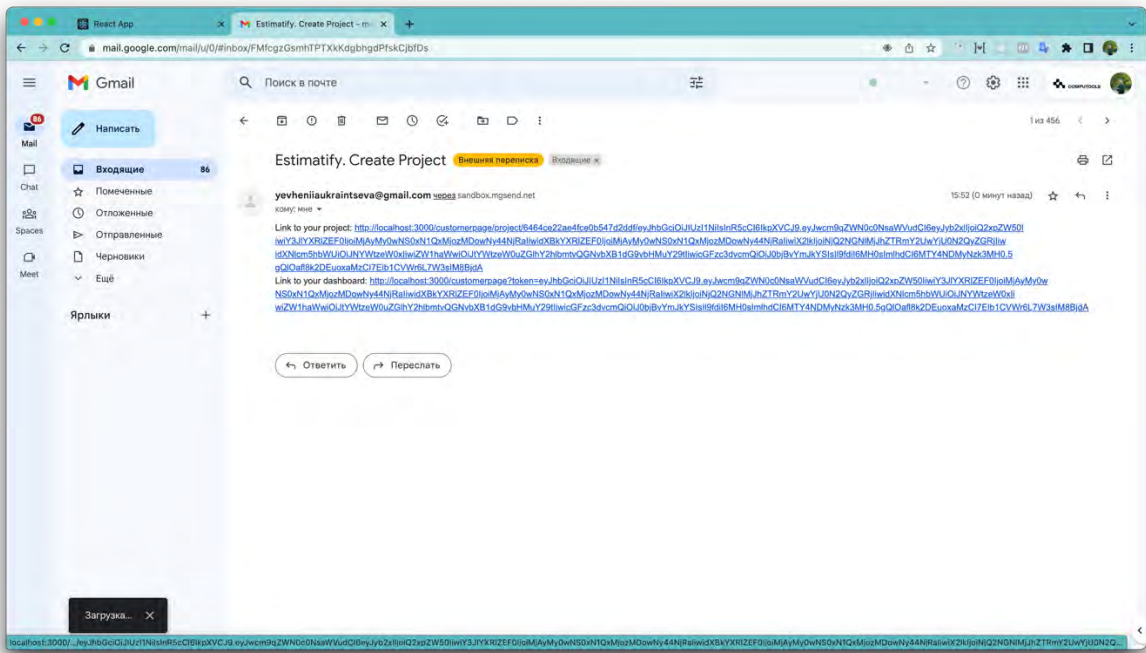


Рисунок 4.11 – Повідомлення з посиланнями для користувача

При натисканні на посилання «Link to your project» відбувається перехід на сторінку проекту. Сторінка зображена на рисунку 4.12.

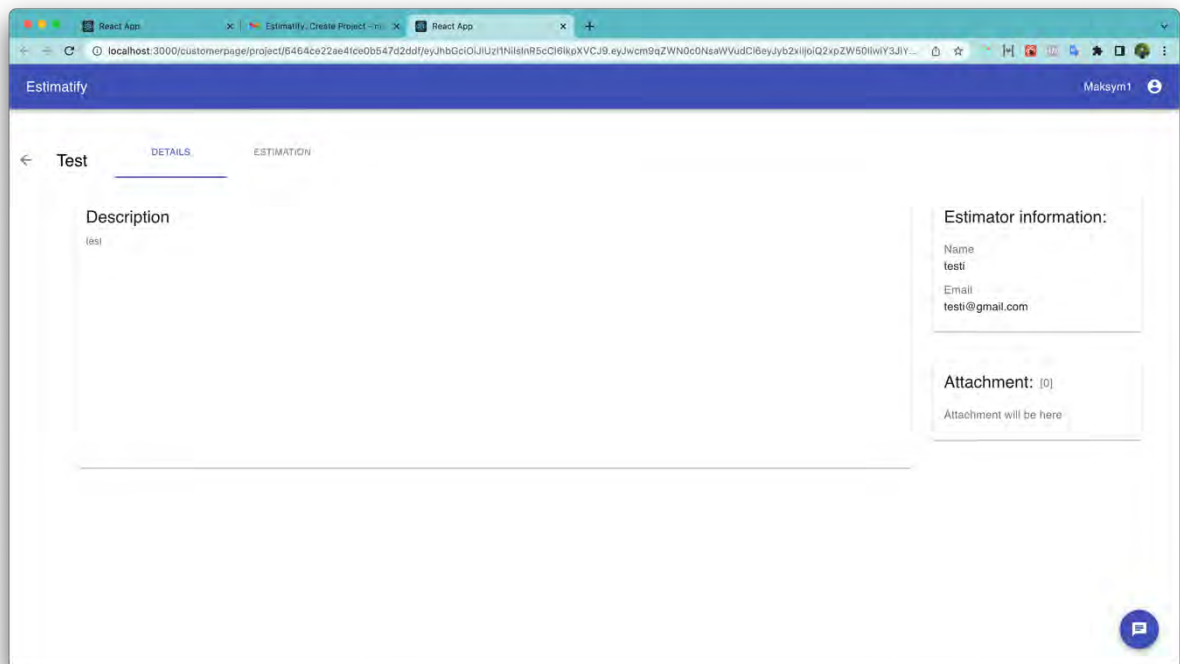


Рисунок 4.12 – Сторінка проекту з правами доступу клієнта

Натиснувши посилання «Link to your dashboard» відбувається перехід на головну сторінку, де знаходиться список усіх проєктів клієнта. Сторінка зображена на рисунку 4.13.

При натисканні на проєкт можна переглянути більш детальну інформацію про сам проєкт та його розробку.

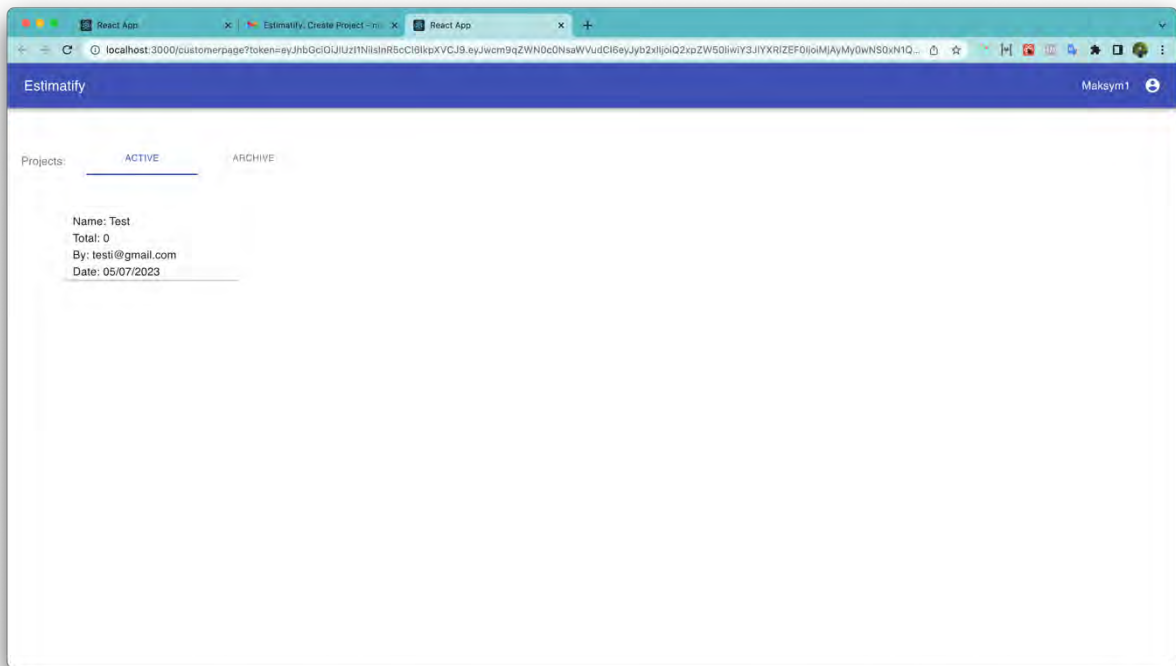


Рисунок 4.13 – Головна сторінка зі списком проєктів клієнта

У правому нижньому куті сторінки кожного проєкту ви знайдете кнопку, що відкриває можливість для прямого спілкування між розробником та замовником. Приклад такого діалогового вікна чату наведено на рисунку 4.14.

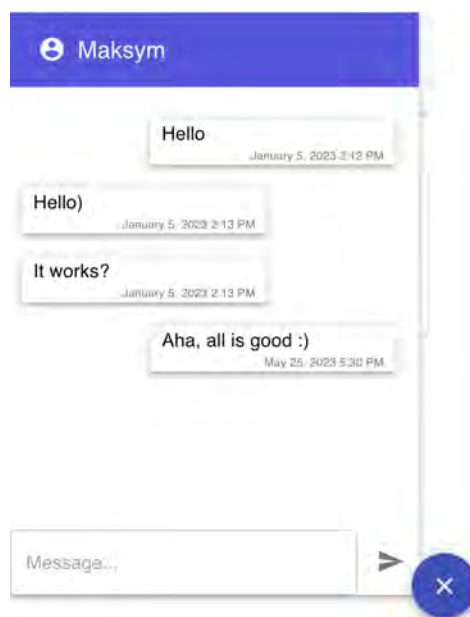


Рисунок 4.14 – Діалогове вікно чату між замовником та розробником

Крім того, замовник має можливість переглянути закріплені файли конкретного проєкту та, при потребі, видалити їх. Цей аспект функціональності демонструється на рисунку 4.15.

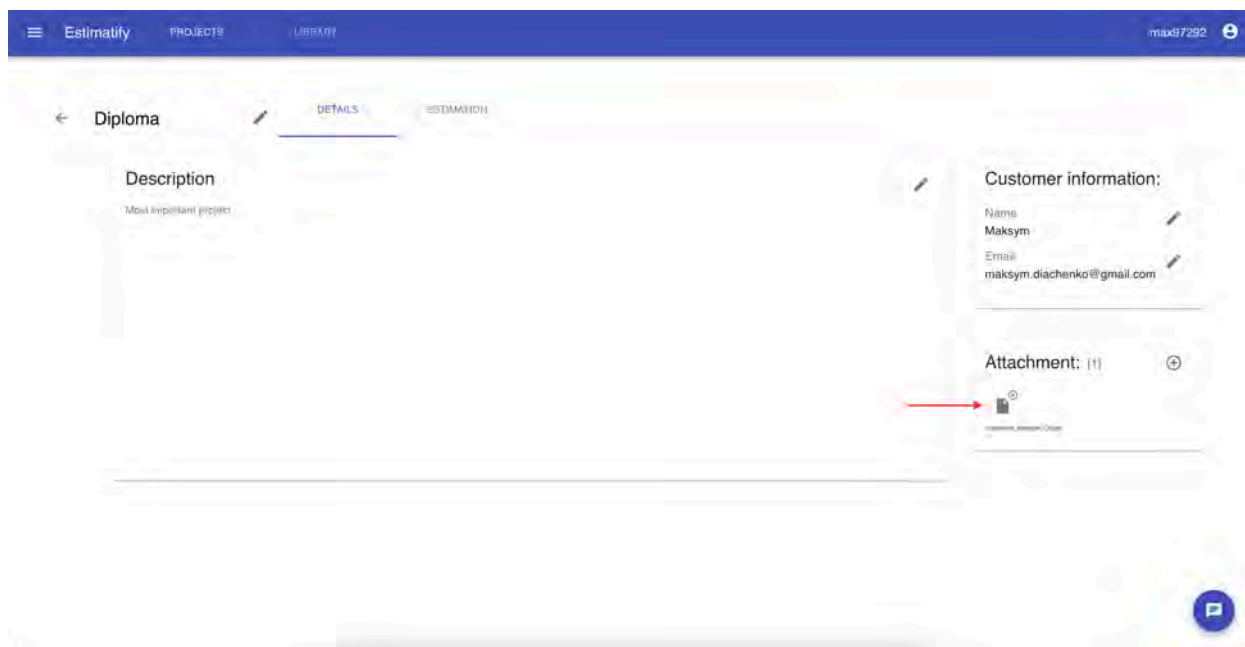


Рисунок 4.15 – Демонстрація функціоналу прикріплення файлів

4.3 Тестування програми

Тестування програми було здійснено за двома напрямками.

Перший напрямок включав перевірку реалізації функціональних вимог застосуванням. Для цього спочатку було розроблено набір тестів, що, зокрема, включали перевірку створення, видалення, редагування проєктів, листування між користувачами, зміни особистих налаштувань. Після завершення роботи було окремо перевірено повністю весь список функціональних вимог для підтвердження працездатності.

Другий напрямок включав роботу з вебзастосунком через різні браузери. Перевірка включала браузери Google Chrome (зокрема версія 113.0.5672.126), Mozilla Firefox (зокрема версія 109.0), Microsoft Edge (зокрема версія 113.0.1774.35). Результати підтвердили, що програма є працездатною у всіх протестованих варіантах.

4.4 Висновки за розділом 4

Програма працює в режимах розробника та замовника, кожен з яких було описано у пояснювальній записці. Виконано успішне тестування вебзастосунку.

ВИСНОВКИ

У ході виконання цієї дипломної роботи було створено клієнтську частину ПЗ для автоматизації взаємодії між замовником та розробником програмного забезпечення. Цей застосунок відзначається здатністю підвищити ефективність та якість взаємодії між замовником і розробником, спрощуючи процеси угод, переговорів та обробки даних.

Цей вебзастосунок надає можливість огляду всіх проєктів, які виконуються замовником та розробником, дозволяючи переглядати, редагувати, видаляти різноманітні дані проєкту, а також налаштовувати особистий обліковий запис для зручного та ефективного користування.

Під час написання дипломної роботи було виконано аналіз предметної області, описано основні рішення щодо реалізації компонентів системи та розроблено методику взаємодії користувача та програміста з системою.

Програмний продукт було розроблено в середовищі програмування Visual Studio Code. В якості технології для розробки клієнтської частини застосування було вибрано JavaScript-бібліотеку з відкритим вихідним кодом React.

ПЗ може ефективно функціонувати на будь-якій операційній системі. Інтерфейс забезпечення є інтуїтивно зрозумілим та зручним у використанні, тому користувачам не потрібно витрачати багато часу на вивчення принципів роботи з ним.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Monday. Документація. – [Електронний ресурс]. – Режим доступу: <https://support.monday.com/>
2. Basecamp. "Basecamp Help." Документація. – [Електронний ресурс]. – Режим доступу: <https://help.basecamp.com/>
3. JetBrains Space Documentation. Документація. – [Електронний ресурс]. – Режим доступу: <https://www.jetbrains.com/help/space>
4. Яковлев, Ігор. "Вивчаємо Monday.com: Управління проектами і командами." O'Reilly Media, Inc., 2022, 350с.
5. Ryan Florence. The Future of React Router and @reach/router – [Електронний ресурс]. – Режим доступу: <https://reacttraining.com/blog/reach-react-router-future/>.
6. Redux. Документація. – [Електронний ресурс]. – Режим доступу: <https://redux.js.org/>.
7. Redux-saga. Документація. – [Електронний ресурс]. – Режим доступу: <https://ru.redux-saga.js.org/>.
8. Malahov, M. Single-page application vs. multiple-page application – [Електронний ресурс]. – Режим доступу: <https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58>
9. MDN. Using Fetch – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
10. Redux Form. Документація. – [Електронний ресурс]. – Режим доступу: <https://redux-form.com/8.2.2/>.
11. Material-UI. Документація. – [Електронний ресурс]. – Режим доступу: <https://material-ui.com/>.
12. Tania Rascia. Setting Up a React Environment on Windows – [Електронний ресурс]. – Режим доступу: <https://www.taniarascia.com/getting->

started-with-react/

13. Aleksandr Shevchenko. Setting Up a React Environment on Ubuntu – [Електронний ресурс]. – Режим доступу: <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-react-project-with-create-react-app>
14. Nichols, S., Foster, J. "Professional JavaScript for Web Developers". Wrox, 2019, 900с.
15. Gasston, P. "The Modern Web: Multi-Device Web Development with HTML5, CSS3, and JavaScript". No Starch Press, 2013, 264с
16. Stefanov, S. "JavaScript Patterns: Build Better Applications with Coding and Design Patterns". O'Reilly Media, 2010, 236с.
17. Redux – [Електронний ресурс]. – Режим доступу: <https://metanit.com/web/react/5.3.php>.
18. Fowler, M. "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence". Addison-Wesley Professional, 2012, 192с.
19. Tauro, C., Aravindh, S., Bhat, A. «Порівняльне дослідження баз даних SQL і NoSQL». 2020, 7с.
20. Керзнер, Г. "Project Management: A Systems Approach to Planning, Scheduling, and Controlling." John Wiley & Sons, 2017, 1296с.
21. Testing Library – [Електронний ресурс]. – Режим доступу: <https://testing-library.com/>
22. Simpson, K. "You Don't Know JS: ES6 & Beyond". O'Reilly Media, 2015, 278с.
23. Мартін, Р.С. "Clean Code: A Handbook of Agile Software Craftsmanship." Prentice Hall, 2008, 464с.
24. Banks, A., Porcello, E. "Learning React: Functional Web Development with React and Redux." O'Reilly Media, 2017, 304с.

ДОДАТОК А
Текст програми

```

src/actions/user.js
import {
  LOGIN_USER, LOGOUT_USER, SIGN_USER, SET_USER, FORGOT_PASSWORD, SET_PASSWORD,
  CHECK_EMAIL, EXPIRE_EMAIL_TOKEN, SET_SETTINGS,
  EXPIRE_USER_TOKEN, NEW_PASSWORD_SETTINGS, CREATE_PROJECT, ALL_PROJECT, GET_USER,
  ARCHIVE_PROJECT, DELETE_AVATAR,
  DELETE_PROJECT, FIND_PROJECT, CHANGE_PROJECT_NAME, CHANGE_PROJECT_DESCRIPTION,
  CHANGE_CLIENT_NAME, CHANGE_CLIENT_EMAIL,
  CREATE_LIBRARY_SECTION, DELETE_ATTACHMENT, GET_ALL_LIBRARY_SECTION,
  CREATE_LIBRARY_SECTION_ITEM, CREATE_SECTION, CREATE_SECTION_ITEM,
  EDIT_SECTION, DELETE_SECTION, EDIT_LIBRARY_SECTION, DELETE_LIBRARY_SECTION,
  TRANSFER_TO_LIBRARY, DOWNLOAD_FROM_LIBRARY, GET_CUSTOMER,
  GET_CUSTOMER_PROJECTS, GET_PROJECT_SECTION, EDIT_LIBRARY_SECTION_ITEM,
  DELETE_LIBRARY_SECTION_ITEM, EDIT_SECTION_ITEM, DELETE_SECTION_ITEM,
  UPDATE_ORDER_LIBRARY, UPDATE_ORDER_SECTION, EDIT_PRIORITY,
  DOWNLOAD_ITEM_FROM_LIBRARY, EDIT_ADDITIONALS, GET_ALL_MESSAGES
} from '../constants/user';

export const loginUser = data => ({
  type: LOGIN_USER,
  data,
});

export const logoutUser = data => ({
  type: LOGOUT_USER,
  data,
});

export const signUser = data => ({
  type: SIGN_USER,
  data,
});

export const setUser = data => ({
  type: SET_USER,
  data,
});

export const getUser = data => ({
  type: GET_USER,
  data,
});

export const forgotPassword = data => ({
  type: FORGOT_PASSWORD,
  data
});

export const setPassword = data => ({
  type: SET_PASSWORD,
  data
});

export const checkEmail = data => ({
  type: CHECK_EMAIL,
  data
});

export const expireEmailToken = data => ({
  type: EXPIRE_EMAIL_TOKEN,
  data,
});

export const expireUserToken = data => ({
  type: EXPIRE_USER_TOKEN,
  data,
});

export const newPasswordSettings = data => ({

```

```

    type: NEW_PASSWORD_SETTINGS,
    data,
  });

export const newSettings = data => ({
  type: SET_SETTINGS,
  data,
});

export const createProject = data => ({
  type: CREATE_PROJECT,
  data,
});

export const getAllProject = data => ({
  type: ALL_PROJECT,
  data,
});

export const archiveProject = data => ({
  type: ARCHIVE_PROJECT,
  data,
});

export const deleteProject = data => ({
  type: DELETE_PROJECT,
  data,
});

export const deleteAvatar = data => ({
  type: DELETE_AVATAR,
  data,
});

export const findProjectById = data => ({
  type: FIND_PROJECT,
  data,
});

export const changeProjectName = data => ({
  type: CHANGE_PROJECT_NAME,
  data,
});

export const changeProjectDescription = data => ({
  type: CHANGE_PROJECT_DESCRIPTION,
  data,
});

export const changeClientName = data => ({
  type: CHANGE_CLIENT_NAME,
  data,
});

export const changeClientEmail = data => ({
  type: CHANGE_CLIENT_EMAIL,
  data,
});

export const deleteAttachment = data => ({
  type: DELETE_ATTACHMENT,
  data,
});

export const getAllLibrarySection = data => ({
  type: GET_ALL_LIBRARY_SECTION,
  data,
});

```

```

export const createLibrarySection = data => ({
  type: CREATE_LIBRARY_SECTION,
  data,
});

export const createLibrarySectionItem = data => ({
  type: CREATE_LIBRARY_SECTION_ITEM,
  data,
});

export const editLibrarySection = data => ({
  type: EDIT_LIBRARY_SECTION,
  data,
});

export const deleteLibrarySection = data => ({
  type: DELETE_LIBRARY_SECTION,
  data,
});

export const createSection = data => ({
  type: CREATE_SECTION,
  data,
});

export const createSectionItem = data => ({
  type: CREATE_SECTION_ITEM,
  data,
});

export const editSection = data => ({
  type: EDIT_SECTION,
  data,
});

export const deleteSection = data => ({
  type: DELETE_SECTION,
  data,
});

export const transferToLibrary = data => ({
  type: TRANSFER_TO_LIBRARY,
  data,
});

export const downloadFromLibrary = data => ({
  type: DOWNLOAD_FROM_LIBRARY,
  data,
});

export const getCustomer = data => ({
  type: GET_CUSTOMER,
  data,
});

export const getCustomerProjects = data => ({
  type: GET_CUSTOMER_PROJECTS,
  data,
});

export const getProjectSection = data => ({
  type: GET_PROJECT_SECTION,
  data,
});

export const editLibrarySectionItem = data => ({
  type: EDIT_LIBRARY_SECTION_ITEM,
  data,
});

```

```

export const deleteLibrarySectionItem = data => ({
  type: DELETE_LIBRARY_SECTION_ITEM,
  data,
});

export const editSectionItem = data => ({
  type: EDIT_SECTION_ITEM,
  data
});

export const deleteSectionItem = data => ({
  type: DELETE_SECTION_ITEM,
  data,
});

export const updateOrderLibrary = data => ({
  type: UPDATE_ORDER_LIBRARY,
  data,
});

export const updateOrderSection = data => ({
  type: UPDATE_ORDER_SECTION,
  data,
});

export const editPriority = data => ({
  type: EDIT_PRIORITY,
  data,
});

export const downloadItemFromLibrary = data => ({
  type: DOWNLOAD_ITEM_FROM_LIBRARY,
  data,
});

export const editAdditional = data => ({
  type: EDIT_ADDITIONAL,
  data,
});

export const getMessages = data => ({
  type: GET_ALL_MESSAGES,
  data
});

src/constants/user.js
export const SET_USER = 'SET_USER';
export const GET_USER = 'GET_USER';
export const LOGIN_USER = 'LOGIN_USER';
export const LOGOUT_USER = 'LOGOUT_USER';
export const TOKEN = 'TOKEN';
export const DROP_TOKEN = 'DROP_TOKEN';
export const SIGN_USER = 'SIGN_USER';
export const FAIL = 'FAIL';
export const FORGOT_PASSWORD = 'FORGOT_PASSWORD';
export const SET_PASSWORD = 'SET_PASSWORD';
export const FIND_EMAIL = 'FIND_EMAIL';
export const CHECK_EMAIL = 'CHECK_EMAIL';
export const SERVER_EXCEPTION = 'SERVER_EXCEPTION';
export const SET_EXCEPTION = 'SET_EXCEPTION';
export const EXPIRE_EMAIL_TOKEN = 'EXPIRE_TOKEN';
export const EXPIRE_USER_TOKEN = 'EXPIRE_USER_TOKEN';
export const NEW_PASSWORD_SETTINGS = 'NEW_PASSWORD_SETTINGS';
export const SET_AVATAR = 'SET_AVATAR';
export const SETTINGS = 'SETTINGS';
export const SET_SETTINGS = 'SET_SETTINGS';
export const CREATE_PROJECT = 'CREATE_PROJECT';
export const SET_PROJECT = 'SET_PROJECT';

```

```

export const DROP_EXCEPTION = 'DROP_EXCEPTION';
export const LOGIN_EXCEPTION = 'LOGIN_EXCEPTION';
export const DROP_LOGIN_EXCEPTION = 'DROP_LOGIN_EXCEPTION';
export const SET_SUCCESS = 'SET_SUCCESS';
export const DROP_SUCCESS = 'DROP_SUCCESS';
export const ALL_PROJECT = 'ALL_PROJECT';
export const SET_PROJECTS = 'SET_PROJECTS';
export const ARCHIVE_PROJECT = 'ARCHIVE_PROJECT';
export const DELETE_PROJECT = 'DELETE_PROJECT';
export const DELETE_AVATAR = 'DELETE_AVATAR';
export const FIND_PROJECT = 'FIND_PROJECT';
export const CHANGE_PROJECT_NAME = 'CHANGE_PROJECT_NAME';
export const CHANGE_PROJECT_DESCRIPTION = 'CHANGE_PROJECT_DESCRIPTION';
export const CHANGE_CLIENT_NAME = 'CHANGE_CLIENT_NAME';
export const CHANGE_CLIENT_EMAIL = 'CHANGE_CLIENT_EMAIL';
export const CREATE_LIBRARY_SECTION = 'CREATE_LIBRARY_SECTION';
export const DELETE_ATTACHMENT = 'DELETE_ATTACHMENT';
export const SET_LIBRARY = 'SET_LIBRARY';
export const GET_ALL_LIBRARY_SECTION = 'GET_ALL_LIBRARY_SECTION';
export const CREATE_LIBRARY_SECTION_ITEM = 'CREATE_LIBRARY_SECTION_ITEM';
export const SET_SECTION = 'SET_SECTION';
export const CREATE_SECTION = 'CREATE_SECTION';
export const CREATE_SECTION_ITEM = 'CREATE_SECTION_ITEM';
export const EDIT_SECTION = 'EDIT_SECTION';
export const DELETE_SECTION = 'DELETE_SECTION';
export const EDIT_LIBRARY_SECTION = 'EDIT_LIBRARY_SECTION';
export const DELETE_LIBRARY_SECTION = 'DELETE_LIBRARY_SECTION';
export const TRANSFER_TO_LIBRARY = 'TRANSFER_TO_LIBRARY';
export const DOWNLOAD_FROM_LIBRARY = 'DOWNLOAD_FROM_LIBRARY';
export const GET_CUSTOMER = 'GET_CUSTOMER';
export const GET_CUSTOMER_PROJECTS = 'GET_CUSTOMER_PROJECTS';
export const GET_PROJECT_SECTION = 'GET_PROJECT_SECTION';
export const EDIT_LIBRARY_SECTION_ITEM = 'EDIT_LIBRARY_SECTION_ITEM';
export const DELETE_LIBRARY_SECTION_ITEM = 'DELETE_LIBRARY_SECTION_ITEM';
export const EDIT_SECTION_ITEM = 'EDIT_SECTION_ITEM';
export const DELETE_SECTION_ITEM = 'DELETE_SECTION_ITEM';
export const UPDATE_ORDER_LIBRARY = 'UPDATE_ORDER_LIBRARY';
export const UPDATE_ORDER_SECTION = 'UPDATE_ORDER_SECTION';
export const EDIT_PRIORITY = 'EDIT_PRIORITY';
export const DOWNLOAD_ITEM_FROM_LIBRARY = 'DOWNLOAD_ITEM_FROM_LIBRARY';
export const EDIT_ADDITIONALS = 'EDIT_ADDITIONALS';
export const GET_ALL_MESSAGES = 'GET_ALL_MESSAGES';
export const SET_MESSAGE = 'SET_MESSAGE';
export const UPDATE_MESSAGE = 'UPDATE_MESSAGE';
export const DELETE_MESSAGE = 'DELETE_MESSAGE';

```

```

src/components/projects/projects.js
import React, { useEffect, useState } from 'react';
import { connect } from 'react-redux';
import PropTypes from 'prop-types';
import { Redirect } from "react-router"

import clsx from 'clsx';
import { makeStyles, useTheme } from '@material-ui/core/styles';
import AppBar from '@material-ui/core/AppBar';
import Toolbar from '@material-ui/core/Toolbar';
import Typography from '@material-ui/core/Typography';
import IconButton from '@material-ui/core/IconButton';
import MenuIcon from '@material-ui/icons/Menu';
import AccountCircle from '@material-ui/icons/AccountCircle';
import FormLabel from '@material-ui/core/FormLabel';
import ChevronLeftIcon from '@material-ui/icons/ChevronLeft';
import ChevronRightIcon from '@material-ui/icons/ChevronRight';
import Divider from '@material-ui/core/Divider';
import List from '@material-ui/core/List';
import Box from '@material-ui/core/Box';
import Tabs from '@material-ui/core/Tabs';
import Tab from '@material-ui/core/Tab';
import Avatar from '@material-ui/core/Avatar';

```

```

import Drawer from '@material-ui/core/Drawer';

import Settings from '../settings/settings';
import ProfileSettings from '../profile/profileSettings';
import CreateProject from '../createProject/createProject';
import LogOut from '../logout/logout';
import ProjectStatusPanel from '../projectContent/projectStatusPanel';
import Library from '../library/library';

import { expireUserToken, getAllProject } from '../actions/user';

const drawerWidth = 240;

const useStyles = makeStyles(theme => ({
  root: {
    flexGrow: 1,
    display: 'flex',
  },
  title: {
    paddingRight: 10,
  },
  pages: {
    flexGrow: 1,
  },
  username: {
    color: 'white',
    marginRight: 20,
  },
  appBar: {
    transition: theme.transitions.create(['margin', 'width'], {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.leavingScreen,
    }),
  },
  appBarShift: {
    width: `calc(100% - ${drawerWidth}px)`,
    marginLeft: drawerWidth,
    transition: theme.transitions.create(['margin', 'width'], {
      easing: theme.transitions.easing.easeOut,
      duration: theme.transitions.duration.enteringScreen,
    }),
  },
  menuButton: {
    marginRight: theme.spacing(2),
  },
  hide: {
    display: 'none',
  },
  drawer: {
    width: drawerWidth,
    flexShrink: 0,
  },
  drawerPaper: {
    width: drawerWidth,
  },
  drawerHeader: {
    display: 'flex',
    alignItems: 'center',
    padding: theme.spacing(0, 1),
    ...theme.mixins.toolbar,
    justifyContent: 'flex-end',
  },
  content: {
    flexGrow: 1,
    padding: theme.spacing(3),
    transition: theme.transitions.create('margin', {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.leavingScreen,
    }),
  },
}));

```

```

    marginLeft: -drawerWidth,
  },
  contentShift: {
    transition: theme.transitions.create('margin', {
      easing: theme.transitions.easing.easeOut,
      duration: theme.transitions.duration.enteringScreen,
    }),
    marginLeft: 0,
  },
  smallAvatar: {
    width: 40,
    height: 40,
  },
});

```

```

function TabPanel(props) {
  const { children, value, index, ...other } = props;

  return (
    <Typography
      component="div"
      role="tabpanel"
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
    >
      <Box p={3}>{children}</Box>
    </Typography>
  );
}

```

```

TabPanel.propTypes = {
  children: PropTypes.node,
  index: PropTypes.any.isRequired,
  value: PropTypes.any.isRequired,
};

```

```

function allyProps(index) {
  return {
    id: `simple-tab-${index}`,
    'aria-controls': `simple-tabpanel-${index}`,
  };
}

```

```

function Projects(props) {

  const [redirect, setRedirect] = useState(false);

  useEffect(() => {
    const token = localStorage.getItem('token');
    if (token) {
      props.checkExpiredUserToken(token);
    } else {
      setRedirect(true);
    }
  }, []);

  useEffect(() => {
    const token = localStorage.getItem('token');
    props.getAllProjectData(token);
  }, []);

  const classes = useStyles();
  const theme = useTheme();

  const [open, setOpen] = React.useState(false);
  const [value, setValue] = React.useState(0);

```

```

function handleDrawerOpen() {
  setOpen(true);
}

function handleDrawerClose() {
  setOpen(false);
}

function handleChange(event, newValue) {
  setValue(newValue);
}

return (
  redirect ? <Redirect to='/login' /> :
  <div className={classes.root}>
    <AppBar position="fixed"
      className={clsx(classes.appBar, {
        [classes.appBarShift]: open,
      })}>
      <Toolbar>
        <div>
          <IconButton
            color="inherit"
            aria-label="open drawer"
            onClick={handleDrawerOpen}
            edge="start"
            className={clsx(classes.menuButton, open && classes.hide)}
          >
            <MenuIcon />
          </IconButton>
        </div>

        <Typography variant="h6" className={classes.title}> Estimatify
      </Typography>

      <Tabs
        value={value}
        onChange={handleChange}
        aria-label="simple tabs example"
        indicatorColor="primary"
        variant="scrollable"
        scrollButtons="auto"
        className={classes.pages}
      >
        <Tab label="Projects" {...allProps(0)} />
        <Tab label="Library" {...allProps(1)} />
      </Tabs>

      <FormLabel className={classes.username}>{props.user ?
        `${props.user.username}` : ''}</FormLabel>

        <div>
          {props.user ?
            props.user.avatar ? <Avatar alt="" className={classes.smallAvatar}
src={props.user.avatar} /> : <AccountCircle />
            : <AccountCircle />}
          </div>

        </Toolbar>
      </AppBar>

      <Drawer
        className={classes.drawer}
        variant="persistent"
        anchor="left"
        open={open}
        classes={{
          paper: classes.drawerPaper,

```

```

    }}
  >
    <div className={classes.drawerHeader}>
      <IconButton onClick={handleDrawerClose}>
        {theme.direction === 'ltr' ? <ChevronLeftIcon /> : <ChevronRightIcon />}
      </IconButton>
    </div>
    <Divider />

    <List>
      <ProfileSettings />
      <LogOut />
    </List>

    <Divider />

    <List>
      <CreateProject />
    </List>

    <Divider />

    <List>
      <Settings />
    </List>

  </Drawer>

  <main
    className={clsx(classes.content, {
      [classes.contentShift]: open,
    })}
  >
    <div className={classes.drawerHeader} />

    <TabPanel value={value} index={0}>
      <ProjectStatusPanel />
    </TabPanel>
    <TabPanel value={value} index={1}>
      <Library />
    </TabPanel>

  </main>
</div>
)
}

Projects.defaultProps = {
  user: null,
  projects: null,
  auth: null,
}

Projects.propTypes = {
  checkExpiredUserToken: PropTypes.func.isRequired,
  getAllProjectData: PropTypes.func.isRequired,
}

const mapStateToProps = store => ({
  user: store.user,
  auth: store.auth,
  settings: store.settings,
  projects: store.projects,
});

const mapDispatchToProps = dispatch => ({
  checkExpiredUserToken: data => dispatch(expireUserToken(data)),
  getAllProjectData: data => dispatch(getAllProject(data)),
});

```

```

export default connect(mapStateToProps, mapDispatchToProps)(Projects);

src/components/projectPage/projectPage.js
import React, { useEffect, useState } from 'react';
import { Redirect } from 'react-router';
import PropTypes from 'prop-types';
import { connect } from 'react-redux';

import { makeStyles } from '@material-ui/core/styles';
import Tabs from '@material-ui/core/Tabs';
import Tab from '@material-ui/core/Tab';
import Typography from '@material-ui/core/Typography';
import Box from '@material-ui/core/Box';
import Input from '@material-ui/core/Input';

import EditIcon from '@material-ui/icons/Edit';
import Tooltip from '@material-ui/core/Tooltip';
import IconButton from '@material-ui/core/IconButton';

import ArrowBackIcon from '@material-ui/icons/ArrowBack';
import DoneIcon from '@material-ui/icons/Done';

import ProjectDetails from './details';
import CreateSection from './estimation/section/createSection';
import SectionContainer from './estimation/sectionTemplate/sectionContainer';
import FromLibrary from '../fromLibrary/index';
import AdditionalCosts from './estimation/additionalCosts/additionalCosts';
import Chat from '../chat/chat';

import { changeProjectName, getProjectSection } from '../../actions/user';

function TabPanel(props) {
  const { children, value, index, ...other } = props;

  return (
    <Typography
      component="div"
      role="tabpanel"
      hidden={value !== index}
      id={`scrollable-auto-tabpanel-${index}`}
      aria-labelledby={`scrollable-auto-tab-${index}`}
      {...other}
    >
      <Box p={3}>{children}</Box>
    </Typography>
  );
}

TabPanel.propTypes = {
  children: PropTypes.node,
  index: PropTypes.any.isRequired,
  value: PropTypes.any.isRequired,
};

function allyProps(index) {
  return {
    id: `scrollable-auto-tab-${index}`,
    'aria-controls': `scrollable-auto-tabpanel-${index}`,
  };
}

const useStyles = makeStyles(theme => ({
  root: {
    flexGrow: 1,
    width: '100%',
    backgroundColor: theme.palette.background.paper,
  },
  label: {

```

```

        marginTop: 20,
        marginLeft: 20,
        width: 200,
      },
      tabPanel: {
        marginLeft: 40,
      },
      description: {
        width: 1150,
        marginLeft: 30,
        marginRight: 30,
      },
      additional: {
        width: 500,
        marginRight: 30,
        marginLeft: 30,
        marginBottom: 40,
      },
      growElement: {
        flexGrow: 1,
      },
      buttons: {
        paddingLeft: 12,
        paddingRight: 12,
        marginTop: 10,
      },
      smallEdit: {
        height: 20,
        width: 20,
      },
      additionalTitle: {
        marginTop: 10,
      },
      count: {
        marginLeft: 10,
        marginTop: 5,
      },
      input: {
        margin: theme.spacing(1),
        width: 200,
      },
      button: {
        marginTop: 20,
        marginBottom: 20,
      },
      secondaryHeading: {
        fontSize: theme.typography.pxToRem(17),
        color: theme.palette.text.secondary,
        marginTop: 15,
      },
      totalTime: {
        marginTop: 30,
      }
    }
  ));

```

```
function ProjectContentPanel(props) {
```

```

  const classes = useStyles();
  const [value, setValue] = useState(0);
  const [redirect, setRedirect] = useState(false);
  const [redirectToLogin, setRedirectToLogin] = useState(false);
  const [redirectUrl, setRedirectUrl] = useState(null);
  const [edit, setEdit] = useState(false);
  const [projectName, setProjectName] = useState('');
  const [totalTime, setTotalTime] = useState(0);
  const [averageTime, setAverageTime] = useState(0);

```

```

  useEffect(() => {
    const token = localStorage.getItem('token');

```

```

    if (!token) {
      setRedirectToLogin(true);
    }
  }, []);

  useEffect(() => {
    if (props.project) {
      const project = props.project.projectId;
      props.getProjectSectionData(project);
    }
  }, [props.project]);

  useEffect(() => {
    let time = 0;
    setTotalTime(0);
    if (props.section) {
      props.section.map((st) => {
        return st.task.map((ts) => {
          time = time + ts.average;
          return setTotalTime(time);
        })
      })
    }
  })

  function handleChange(event, newValue) {
    setValue(newValue);
  }

  const handleOpenProjects = () => {
    setRedirectUrl('/');
    setRedirect(true);
  };

  const handleEdit = () => {
    setEdit(true);
  };

  const handleEditStop = () => {
    if (projectName) {
      const userToken = localStorage.getItem('token');
      props.changeProjectNameData({ token: userToken, projectsName: projectName,
projectId: props.project.projectId });
      setEdit(false);
    } else {
      setEdit(false);
    }
  };

  return (
    redirectToLogin ? <Redirect to='/login' /> :
    (redirect && redirectUrl) ? <Redirect to={redirectUrl} /> :
    <div className={classes.root}>
      <Box display="flex" flexDirection="row">
        <Tooltip title="Back">
          <IconButton aria-label="Back" className={classes.buttons}
onClick={handleOpenProjects}>
            <ArrowBackIcon />
          </IconButton>
        </Tooltip>

        {
          edit ?
            <Input
              defaultValue={props.project
props.project.projects.name : ''}
              className={classes.input}
              inputProps={{
                'aria-label': 'description',

```

```

    }}
    onChange={(e) => setProjectName(e.target.value)}
  /> :
  <Typography gutterBottom variant="h5" component="h2"
className={classes.label}>
    {props.project ? props.project.projects.name : ''}
  </Typography>
}

<div>
  {
    edit ?
    <IconButton                                aria-label="Edit"
className={classes.buttons} onClick={handleEditStop}>
      <DoneIcon />
    </IconButton>
    :
    <IconButton                                aria-label="Edit"
className={classes.buttons} onClick={handleEdit}>
      <EditIcon />
    </IconButton>
  }
</div>

<Tabs
  value={value}
  onChange={handleChange}
  indicatorColor="primary"
  textColor="primary"
  variant="scrollable"
  scrollButtons="auto"
  aria-label="scrollable auto tabs example"
>
  <Tab label="Details" {...allProps(0)} />
  <Tab label="Estimation" {...allProps(1)} />
</Tabs>
</Box>

<Box className={classes.tabPanel}>
  <TabPanel value={value} index={0}>
    <ProjectDetails project={props.project} />
  </TabPanel>

  <TabPanel value={value} index={1}>
    <SectionContainer sections={props.section} />

    <FromLibrary project={props.project} />
    <CreateSection project={props.project} />

    <AdditionalCosts                                project={props.project}
sections={props.section} setAverageTime={setAverageTime}/>

    <Box                                display="flex"                                flexDirection="row"
justifyContent="flex-end" className={classes.totalTime}>
      <Typography
className={classes.secondaryHeading}>Project      total:      {Math.round(totalTime      +
averageTime)} hours</Typography>
    </Box>

  </TabPanel>

  <Chat project={props.project} user={props.user}/>
</Box>

</div>
);
}

```

```

ProjectContentPanel.defaultProps = {
  user: null,
  projects: null,
  section: null,
}

ProjectContentPanel.propTypes = {
  changeProjectNameData: PropTypes.func.isRequired,
  getProjectSectionData: PropTypes.func.isRequired,
}

const mapStateToProps = store => ({
  user: store.user,
  projects: store.projects,
  section: store.section,
});

const mapDispatchToProps = dispatch => ({
  changeProjectNameData: data => dispatch(changeProjectName(data)),
  getProjectSectionData: data => dispatch(getProjectSection(data)),
});

export default connect(mapStateToProps, mapDispatchToProps)(ProjectContentPanel);

src/components/projectPage/details.js
import React, { useEffect, useState } from 'react';
import PropTypes from 'prop-types';
import { connect } from 'react-redux';

import validator from 'validator';

import EditIcon from '@material-ui/icons/Edit';
import DoneIcon from '@material-ui/icons/Done';

import { makeStyles } from '@material-ui/core/styles';
import IconButton from '@material-ui/core/IconButton';
import Card from '@material-ui/core/Card';
import Typography from '@material-ui/core/Typography';
import TextareaAutosize from '@material-ui/core/TextareaAutosize';
import Input from '@material-ui/core/Input';
import { CardContent } from '@material-ui/core';
import Box from '@material-ui/core/Box';

import { changeProjectDescription, changeClientName, changeClientEmail } from
'../../actions/user';
import { dropException } from '../../actions/exception';

import Attachment from './attachment/attachment';

const useStyles = makeStyles(theme => ({
  root: {
    flexGrow: 1,
    width: '100%',
    backgroundColor: theme.palette.background.paper,
  },
  label: {
    marginTop: 20,
    marginLeft: 20,
  },
  tabPanel: {
    marginLeft: 40,
  },
  description: {
    width: "80%",
    marginLeft: 30,
    marginRight: 30,
  },
  additional: {
    width: "100%",

```

```

        marginBottom: 40,
      },
      growElement: {
        flexGrow: 1,
      },
      edit: {
        paddingLeft: 12,
        paddingRight: 12,
        marginTop: 10,
      },
      smallEdit: {
        height: 20,
        width: 20,
      },
      additionalTitle: {
        marginTop: 10,
      },
      count: {
        marginLeft: 10,
        marginTop: 5,
      },
      textArea: {
        width: "100%",
      },
      message: {
        fontSize: 10,
        color: "red",
        textAlign: "center",
        marginTop: 10,
      }
    }
  ));

function ProjectDetails(props) {
  const classes = useStyles();

  const [editDescription, setEditDescription] = useState(false);
  const [editName, setEditName] = useState(false);
  const [editEmail, setEditEmail] = useState(false);
  const [isItEmail, setIsItEmail] = useState(true);
  const [projectDescription, setProjectDescription] = useState('');
  const [clientName, setClientName] = useState('');
  const [clientEmail, setClientEmail] = useState('');

  useEffect(() => {
    if (props.serverException) {
      props.dropServerExceptionData(null);
    }

    return function resetServerException() {
      if (props.serverException) {
        props.dropServerExceptionData(null);
      }
    }
  }, []);

  const handleEditDescription = () => {
    setEditDescription(true);
  };

  const handleStopEditDescription = () => {
    if (projectDescription) {
      const userToken = localStorage.getItem('token');

      props.changeProjectDescriptionData({
        token: userToken,
        projectsDescription: projectDescription,
        projectId: props.project.projectId
      });
    }
  };

```

```

        setEditDescription(false);
    } else {
        setEditDescription(false);
    }
};

const handleEditName = () => {
    if (props.serverException) {
        props.dropServerExceptionData(null);
    }
    setEditName(true);
};

const handleStopEditName = () => {
    if (clientName) {
        const userToken = localStorage.getItem('token');

        props.changeClientNameData({
            token: userToken,
            clientsName: clientName,
            projectsId: props.project.projectId
        });

        setClientName('');
        setEditName(false);
    } else {
        setEditName(false);
    }
};

const handleEditEmail = () => {
    if (props.serverException) {
        props.dropServerExceptionData(null);
    }
    setEditEmail(true);
};

const handleStopEditEmail = () => {
    if (clientEmail) {
        if (validator.isEmail(clientEmail)) {
            const userToken = localStorage.getItem('token');

            props.changeClientEmailData({
                token: userToken,
                clientsEmail: clientEmail,
                projectsId: props.project.projectId
            });
        } else {
            setIsItEmail(false);
        }
        setClientEmail('');
        setEditEmail(false);
    } else {
        setIsItEmail(true);
        setEditEmail(false);
    }
};

return (
    <Box display="flex" flexDirection="row">
        <Card className={classes.description}>
            <CardContent>
                <Box display="flex" flexDirection="row" justifyContent="space-
between">
                    <Typography gutterBottom variant="h5" component="h2">
                        Description
                    </Typography>

```

```

        <div>
          {
            editDescription ?
              <IconButton                                aria-label="Edit"
onClick={handleStopEditDescription}>
                <DoneIcon />
              </IconButton>
              :
              <IconButton                                aria-label="Edit"
onClick={handleEditDescription}>
                <EditIcon />
              </IconButton>
            }
          </div>
        </Box>

        {
          editDescription ?
            <TextareaAutosize
              aria-label="maximum height"
              defaultValue={props.project
props.project.projects.descriptions : ''}
              className={classes.textArea}
              onChange={(e) =>
setProjectDescription(e.target.value)}
            />
            :
            <Typography variant="body2" color="textSecondary"
component="p">
              {props.project ? props.project.projects.descriptions :
''}
            </Typography>
          }
        </CardContent>
      </Card>

      <div className={classes.growElement} />
      <Box display="flex" flexDirection="column">
        <Card className={classes.additional}>
          <CardContent>
            <Typography gutterBottom variant="h5" component="h2">Customer
information:</Typography>
            <Box display="flex" flexDirection="row">
              <Box flexGrow={1}>
                <Typography color="textSecondary" component="p"
className={classes.additionalTitle}>Name</Typography>
                {
                  editName ?
                    <Input
                      defaultValue={props.project
props.project.users.username : ''}
                      inputProps={{
                        'aria-label': 'username',
                      }}
                      onChange={(e) =>
setClientName(e.target.value)}
                    />
                    :
                    <Typography component="p">
                      {props.project
props.project.users.username : ''}
                    </Typography>
                }
              </Box>
            }
          </CardContent>
          {
            editName ?

```

```

onClick={handleStopEditName}>
    <IconButton
        <DoneIcon />
    </IconButton>
    :
    <IconButton
        <EditIcon />
    </IconButton>
    aria-label="Edit"
onClick={handleEditName}>
    <EditIcon />
    </IconButton>
    aria-label="Edit"
    }
</Box>

<Box display="flex" flexDirection="row">
    <Box flexGrow={1}>
        <Typography color="textSecondary" component="p"
className={classes.additionalTitle}>Email</Typography>
        {
            editEmail ?
            <Input
                defaultValue={props.project
                ?
                props.project.users.email : ''}
                inputProps={{
                    'aria-label': 'email',
                }}
                onChange={(e) =>
                    setClientEmail(e.target.value)}
            </Input> :
            <Typography component="p">
                {props.project ? props.project.users.email
                : ''}
            </Typography>
        }
    </Box>

    {
        editEmail ?
        <IconButton
            <DoneIcon />
        </IconButton>
        <IconButton
            <EditIcon />
        </IconButton>
        aria-label="Edit"
        onClick={handleStopEditEmail} className={classes.buttons}>
        <IconButton
            <EditIcon />
        </IconButton>
        aria-label="Edit"
        onClick={handleEditEmail}>
        <EditIcon />
        </IconButton>
    }
</Box>

{
    isItEmail ? '' : <div className={classes.message}>Invalid
email</div>
}

<div className={classes.message}>{props.serverException ?
`${props.serverException.error}` : ''}</div>

</CardContent>
</Card>

<Attachment project={props.project}/>

</Box>
</Box>
)
}

ProjectDetails.defaultProps = {

```

```

    user: null,
    projects: null,
    serverException: null,
  }

  ProjectDetails.propTypes = {
    changeProjectDescriptionData: PropTypes.func.isRequired,
    changeClientNameData: PropTypes.func.isRequired,
    changeClientEmailData: PropTypes.func.isRequired,
    dropServerExceptionData: PropTypes.func.isRequired,
  }

  const mapStateToProps = store => ({
    user: store.user,
    projects: store.projects,
    serverException: store.serverException,
  });

  const mapDispatchToProps = dispatch => ({
    changeProjectDescriptionData: data => dispatch(changeProjectDescription(data)),
    changeClientNameData: data => dispatch(changeClientName(data)),
    changeClientEmailData: data => dispatch(changeClientEmail(data)),
    dropServerExceptionData: data => dispatch(dropException(data)),
  });

  export default connect(mapStateToProps, mapDispatchToProps)(ProjectDetails);

```

```

src/containers/projectPage.js
import React, { useEffect, useState } from 'react';
import { connect } from 'react-redux';
import PropTypes from 'prop-types';
import { Redirect } from 'react-router';

import IndexProjectPage from '../components/projectPage/index';

import { expireUserToken } from '../actions/user';

const getProject = (array, id) => {
  for (let iterator of array) {
    if (iterator.projectId === id) {
      return iterator;
    }
  }
  return null;
}

function projectPage(props) {
  const [project, setProject] = useState(null);
  const [redirect, setRedirect] = useState(false);

  const tryToSetProject = () => {
    const projectId = props.match.params.projectId;
    const result = getProject(props.projects, projectId);
    setProject(result);
  };

  useEffect(() => {
    if (!props.projects) {
      setRedirect(true);
      const token = localStorage.getItem('token');
      props.checkExpiredUserToken(token);
      return;
    }
    tryToSetProject();
  }, [props.projects]);

  useEffect(() => {
    if (props.projects && !project) {

```

```

        tryToSetProject()
      }
    }, [props.projects]);

    return (
      <div>
        {redirect ? <Redirect to='/login' /> : project ? <IndexProjectPage
project={project} /> : ''}
      </div>
    )
  }

  projectPage.defaultProps = {
    user: null,
    projects: null,
    auth: null,
  }

  projectPage.propTypes = {
    checkExpiredUserToken: PropTypes.func.isRequired,
  }

  const mapStateToProps = store => ({
    user: store.user,
    auth: store.auth,
    projects: store.projects,
  });

  const mapDispatchToProps = dispatch => ({
    checkExpiredUserToken: data => dispatch(expireUserToken(data)),
  });

  export default connect(mapStateToProps, mapDispatchToProps)(projectPage);

src/reducers/user.js
import { SET_USER, FAIL } from '../constants/user';

const initialState = null;

const userReducer = (state = initialState, action) => {
  switch (action.type) {
    case SET_USER:
      return action.data;
    case FAIL:
      return '';
    default:
      return state;
  }
}

export default userReducer;

src/router/index.js
import React, { Component } from 'react';
import { Router, Route, Switch } from 'react-router-dom';
import history from '../utils/history';

import LogIn from '../containers/authorization';
import Signup from '../containers/signup';
import forgotPassword from '../containers/forgotpassword';
import setPassword from '../containers/setpassword';
import checkEmail from '../containers/email';
import projects from '../containers/projects';
import profileSettings from '../containers/profileSettings';
import settings from '../containers/settings';
import projectPage from '../containers/projectPage';
import customerView from '../containers/customerView';
import customerProjectPage from '../containers/customerProject';

```

```

export default class MainRouter extends Component {
  render() {
    return (
      <Router history={history}>
        <Switch>
          <Route exact path="/" component={projects}/>
          <Route exact path="/login" component={LogIn} />
          <Route exact path="/signup" component={Signup}/>
          <Route exact path="/forgotpassword" component={forgotPassword} />
          <Route exact path="/setpassword" component={setPassword} />
          <Route exact path="/checkemail" component={checkEmail}/>
          <Route exact path="/profilesettings" component={profileSettings}/>
          <Route exact path="/settings" component={settings}/>
          <Route exact path="/projectpage/:projectId"
component={projectPage}/>
          <Route exact path="/customerpage" component={customerView}/>
          <Route exact path="/customerpage/project/:projectId/:token"
component={customerProjectPage}/>
        </Switch>
      </Router>
    );
  }
}

src/saga/sagas.js
import { call, put, takeLatest } from 'redux-saga/effects';

import {
  LOGIN_USER, SIGN_USER, LOGOUT_USER, FORGOT_PASSWORD, SET_PASSWORD,
  EXPIRE_EMAIL_TOKEN, EXPIRE_USER_TOKEN, NEW_PASSWORD_SETTINGS,
  SET_SETTINGS, CREATE_PROJECT, ALL_PROJECT, GET_USER, ARCHIVE_PROJECT,
  DELETE_PROJECT, DELETE_AVATAR, FIND_PROJECT, CHANGE_PROJECT_NAME,
  CHANGE_PROJECT_DESCRIPTION, CHANGE_CLIENT_NAME, CHANGE_CLIENT_EMAIL,
  CREATE_LIBRARY_SECTION, DELETE_ATTACHMENT, GET_ALL_LIBRARY_SECTION,
  CREATE_LIBRARY_SECTION_ITEM, CREATE_SECTION, CREATE_SECTION_ITEM, EDIT_SECTION,
  DELETE_SECTION, EDIT_LIBRARY_SECTION, DELETE_LIBRARY_SECTION,
  DOWNLOAD_FROM_LIBRARY, TRANSFER_TO_LIBRARY, GET_CUSTOMER, GET_CUSTOMER_PROJECTS,
  GET_PROJECT_SECTION, EDIT_LIBRARY_SECTION_ITEM,
  DELETE_LIBRARY_SECTION_ITEM, EDIT_SECTION_ITEM, DELETE_SECTION_ITEM,
  UPDATE_ORDER_LIBRARY, UPDATE_ORDER_SECTION, EDIT_PRIORITY,
  DOWNLOAD_ITEM_FROM_LIBRARY, EDIT_ADDITIONALS, GET_ALL_MESSAGES, UPDATE_MESSAGE,
  DELETE_MESSAGE
} from '../constants/user';

import { setUser } from '../actions/user';
import { getToken, dropToken } from '../actions/token';
import { setException, loginException } from '../actions/exception';
import { setSettings } from '../actions/settings';
import { setProject, setProjects } from '../actions/project';
import { setSuccess } from '../actions/success';
import { setLibrary } from '../actions/library';
import { setSection } from '../actions/section';
import { setMessage } from '../actions/message';

import axios from 'axios';
import history from '../utils/history';

function* login({ data }) {
  try {
    const res = yield call(axios.post, "http://localhost:3005/login", data);
    yield put(setUser(res.data));
    yield put(getToken(res.data));
    history.push('/');
  } catch (error) {
    yield put(loginException({ error: error.response.data }));
  }
}

```

```

function* signup({ data }) {
  try {
    const res = yield call(axios.post, "http://localhost:3005/create", data);
    yield put(setUser(res.data));
    yield put(setSettings(res.data));

    history.push('/login');
  } catch (error) {
    yield put(setException({ error: error.response.data }));
  }
}

function* logout({ data }) {
  try {
    const res = yield call(axios.delete, `http://localhost:3005/logout/${data}`);
    yield put(setUser(res.data));
    yield put(dropToken());

    history.push('/login');
  } catch (error) {
    yield put({ type: 'FAIL', error })
  }
}

function* findemail({ data }) {
  try {
    const res = yield call(axios.post, "http://localhost:3005/findemail", data);
    yield put(setUser(res.data.user));

    history.push('/checkemail')
  } catch (error) {
    yield put(setException({ error: error.response.data }));
  }
}

function* setpassword({ data }) {
  try {
    const res = yield call(axios.put, "http://localhost:3005/setpassword", data);
    yield put(setUser(res.data));

    history.push('/login');
  } catch (error) {
    yield put({ type: 'FAIL', error })
  }
}

function* setexpiredtoken({ data }) {
  try {
    const res = yield call(axios.post,
`http://localhost:3005/expiretoken/${data}`);
    yield put(setException(res.data));

  } catch (error) {
    yield put({ type: 'FAIL', error })
  }
}

function* checkexpiredusertoken({ data }) {
  try {
    const config = {
      headers: { 'Authorization': `Bearer ${data}` },
    }
    const res = yield call(axios.get, "http://localhost:3005/checkExpireToken",
config);
  }
}

```

```

        yield put(setUser(res.data.user));
        yield put(setSettings(res.data.settings));
    } catch (error) {
        history.push('/login');
    }
}

function* changepasswordinsettings({ data }) {
    try {
        const { token } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.put, "http://localhost:3005/changepasswordsettings", data);
        yield put(setSuccess({ message: 'Password was successfully changed' }));
    } catch (error) {
        yield put(setException({ error: error.response.data }));
    }
}

function* settings({ data }) {
    try {
        const { token } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        const res = yield call(axios.put, "http://localhost:3005/settings", data);
        yield put(setSettings(res.data));
        yield put(setSuccess({ message: 'Settings changed successfully' }));

        history.push('/');
    } catch (error) {
        yield put({ type: 'FAIL', error });
    }
}

function* createproject({ data }) {
    try {
        const { token } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        const res = yield call(axios.post, "http://localhost:3005/createproject",
data);
        yield put(setProject(res.data));
        yield put(setSuccess({ message: 'Project created successfully' }));
        yield put({ type: ALL_PROJECT, data: token });
    } catch (error) {
        yield put(setException({ error: error.response.data }));
    }
}

function* allesimatorsprojects({ data }) {
    try {
        yield put({ type: EXPIRE_USER_TOKEN, data });
        const res = yield call(axios.get,
`http://localhost:3005/estimatorsproject/${data}`);
        yield put(setProjects(res.data));
    } catch (error) {
        yield put({ type: 'FAIL', error });
    }
}

function* getuser({ data }) {
    try {
        const res = yield call(axios.get, `http://localhost:3005/getuser/${data}`);
        yield put(setUser(res.data));
    } catch (error) {
        yield put({ type: 'FAIL', error });
    }
}

```

```

function* archiveproject({ data }) {
  try {
    const { token } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/changeprojectstatus", data);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* deleteproject({ data }) {
  try {
    const { id, token } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.delete, `http://localhost:3005/deleteproject/${id}`);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* deleteavatar({ data }) {
  try {
    yield put({ type: EXPIRE_USER_TOKEN, data });
    yield call(axios.delete, `http://localhost:3005/deleteavatar/${data}`);
    yield put({ type: GET_USER, data });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* findprojectbyid({ data }) {
  try {
    const { id, token } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.get, `http://localhost:3005/findprojectbyid/${id}`);
  } catch (error) {
    history.push('/login');
  }
}

function* changeprojectname({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/changeprojectname", other);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* changeprojectdescription({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/changeprojectdescription",
other);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* changeclientname({ data }) {
  try {
    const { token, ...other } = data;

```

```

        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.put, "http://localhost:3005/changeclientname", other);
        yield put({ type: ALL_PROJECT, data: token });
      } catch (error) {
        yield put(setException({ error: error.response.data }));
      }
    }

function* changeclientemail({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/changeclientemail", other);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put(setException({ error: error.response.data }));
  }
}

function* deleteattachment({ data }) {
  try {
    const { token, attachmentId } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.delete,
`http://localhost:3005/deleteattachment/${attachmentId}`);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* allsections({ data }) {
  try {
    yield put({ type: EXPIRE_USER_TOKEN, data });
    const res = call(axios.get,
`http://localhost:3005/getlibraries/${data}`);
    yield put(setLibrary(res.data));
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* createlibrarysection({ data }) {
  try {
    const { token } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.post, "http://localhost:3005/createlibrarysection", data);
    yield put(setSuccess({ message: 'Section created successfully' }));
    yield put({ type: GET_ALL_LIBRARY_SECTION, data: token });
  } catch (error) {
    yield put(setException({ error: error.response.data }));
  }
}

function* editlibrarysection({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/editlibrarysection", other);
    yield put(setSuccess({ message: 'Section changed successfully' }));
    yield put({ type: GET_ALL_LIBRARY_SECTION, data: token });
  } catch (error) {
    yield put(setException({ error: error.response.data }));
  }
}

function* deletelibrarysection({ data }) {
  try {

```

```

        const { token, sectionId } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.delete,
`http://localhost:3005/deletelibrarysection/${sectionId}`);
        yield put({ type: GET_ALL_LIBRARY_SECTION, data: token });
    } catch (error) {
        yield put({ type: 'FAIL', error });
    }
}

function* createlibrarysectionitem({ data }) {
    try {
        const { token, ...other } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.post, "http://localhost:3005/createlibrarysectionitem",
other);
        yield put(setSuccess({ message: 'Item created successfully' }));
        yield put({ type: GET_ALL_LIBRARY_SECTION, data: token });
    } catch (error) {
        yield put(setException({ error: error.response.data }));
    }
}

function* createsection({ data }) {
    try {
        const { token, ...other } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.post, "http://localhost:3005/createsection", other);
        yield put(setSuccess({ message: 'Section created successfully' }));
        yield put({ type: ALL_PROJECT, data: token });
    } catch (error) {
        yield put(setException({ error: error.response.data }));
    }
}

function* createsectionitem({ data }) {
    try {
        const { token } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.post, "http://localhost:3005/createsectionitem", data);
        yield put(setSuccess({ message: 'Item created successfully' }));
        yield put({ type: ALL_PROJECT, data: token });
    } catch (error) {
        yield put(setException({ error: error.response.data }));
    }
}

function* editsection({ data }) {
    try {
        const { token, ...other } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.put, "http://localhost:3005/editsection", other);
        yield put(setSuccess({ message: 'Section changed successfully' }));
        yield put({ type: ALL_PROJECT, data: token });
    } catch (error) {
        yield put(setException({ error: error.response.data }));
    }
}

function* deletesection({ data }) {
    try {
        const { token, sectionId } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.delete, `http://localhost:3005/deletesection/${sectionId}`);
        yield put({ type: ALL_PROJECT, data: token });
    } catch (error) {
        yield put({ type: 'FAIL', error });
    }
}

```

```

function* fromlibrary({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.post, "http://localhost:3005/fromlibrary", other);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* tolibrary({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.post, "http://localhost:3005/tolibrary", other);
    yield put({ type: GET_ALL_LIBRARY_SECTION, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* getcustomer({ data }) {
  try {
    const res = yield call(axios.get,
`http://localhost:3005/getcustomer/${data}`);
    yield put(setUser(res.data.user));
    yield put(getToken(res.data));
    const project = yield call(axios.get,
`http://localhost:3005/customersproject/${data}`);
    yield put(setProjects(project.data));
  } catch (error) {
    history.push('/login');
  }
}

function* allcustomerprojects({ data }) {
  try {
    const res = yield call(axios.get,
`http://localhost:3005/customersproject/${data}`);
    yield put(setProjects(res.data));
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* getprojectsections({ data }) {
  try {
    const res = yield call(axios.get,
`http://localhost:3005/getprojectsections/${data}`);
    yield put(setSection(res.data));
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* editlibraryitem({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/editlibraryitem", other);
    yield put(setSuccess({ message: 'Item changed successfully' }));
    yield put({ type: GET_ALL_LIBRARY_SECTION, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

```

```

function* deletelibraryitem({ data }) {
  try {
    const { token, itemId } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.delete, `http://localhost:3005/deletelibraryitem/${itemId}`);
    yield put({ type: GET_ALL_LIBRARY_SECTION, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* editsectionitem({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/editsectionitem", other);
    yield put(setSuccess({ message: 'Item changed successfully' }));
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* deletesectionitem({ data }) {
  try {
    const { token, itemId } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.delete, `http://localhost:3005/deletesectionitem/${itemId}`);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* updateorderlibrary({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/updateorderlibrary", other);
    yield put({ type: GET_ALL_LIBRARY_SECTION, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* updateordersection({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/updateordersection", other);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* editpriority({ data }) {
  try {
    const { token, ...other } = data;
    yield put({ type: EXPIRE_USER_TOKEN, data: token });
    yield call(axios.put, "http://localhost:3005/editpriority", other);
    yield put({ type: ALL_PROJECT, data: token });
  } catch (error) {
    yield put({ type: 'FAIL', error });
  }
}

function* downloaditemfromlibrary({ data }) {
  try {

```

```

        const { token, ...other } = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.post, "http://localhost:3005/downloaditemfromlibrary",
other);
        yield put({ type: ALL_PROJECT, data: token });
    } catch(error) {
        yield put({ type: 'FAIL', error });
    }
}

function* editadditional({ data }) {
    try {
        const {token, ...other} = data;
        yield put({ type: EXPIRE_USER_TOKEN, data: token });
        yield call(axios.put, "http://localhost:3005/editadditional", other);
        yield put(setSuccess({ message: 'Additional changed successfully' }));
        yield put({ type: ALL_PROJECT, data: token });
    } catch (error) {
        yield put({ type: 'FAIL', error });
    }
}

function* getmessages({ data }) {
    try {
        const { projectId } = data;
        const res = yield call(axios.get,
`http://localhost:3005/getmessages/${projectId}`);
        yield put(setMessage(res.data));
    } catch(error) {
        yield put({ type: 'FAIL', error });
    }
}

function* updatemessagetext({ data }) {
    try {
        const { project, ...other} = data;
        yield call(axios.put, "http://localhost:3005/updatemessagetext", other);
        yield put({ type: GET_ALL_MESSAGES, data: {projectId: project} });
    } catch(error) {
        yield put({ type: 'FAIL', error });
    }
}

function* deletemessage({ data }) {
    try {
        const { project, id} = data;
        yield call(axios.delete, `http://localhost:3005/deletemessage/${id}`);
        yield put({ type: GET_ALL_MESSAGES, data: {projectId: project} });
    } catch(error) {
        yield put({ type: 'FAIL', error });
    }
}

export default function* Watch() {
    yield takeLatest(LOGIN_USER, login);
    yield takeLatest(SIGN_USER, signup);
    yield takeLatest(LOGOUT_USER, logout);
    yield takeLatest(FORGOT_PASSWORD, findemail);
    yield takeLatest(SET_PASSWORD, setpassword);
    yield takeLatest(EXPIRE_EMAIL_TOKEN, setexpiredtoken);
    yield takeLatest(EXPIRE_USER_TOKEN, checkexpiredusertoken);
    yield takeLatest(NEW_PASSWORD_SETTINGS, changepasswordinsettings);
    yield takeLatest(SET_SETTINGS, settings);
    yield takeLatest(CREATE_PROJECT, createproject);
    yield takeLatest(ALL_PROJECT, allesimatorsprojects);
    yield takeLatest(GET_USER, getuser);
    yield takeLatest(ARCHIVE_PROJECT, archiveproject);

```

```

    yield takeLatest(DELETE_PROJECT, deleteproject);
    yield takeLatest(DELETE_AVATAR, deleteavatar);
    yield takeLatest(FIND_PROJECT, findprojectbyid);
    yield takeLatest(CHANGE_PROJECT_NAME, changeprojectname);
    yield takeLatest(CHANGE_PROJECT_DESCRIPTION, changeprojectdescription);
    yield takeLatest(CHANGE_CLIENT_NAME, changeclientname);
    yield takeLatest(CHANGE_CLIENT_EMAIL, changeclientemail);
    yield takeLatest(CREATE_LIBRARY_SECTION, createlibrarysection);
    yield takeLatest(DELETE_ATTACHMENT, deleteattachment);
    yield takeLatest(GET_ALL_LIBRARY_SECTION, allsections);
    yield takeLatest(CREATE_LIBRARY_SECTION_ITEM, createlibrarysectionitem);
    yield takeLatest(CREATE_SECTION, createsection);
    yield takeLatest(CREATE_SECTION_ITEM, createsectionitem);
    yield takeLatest(EDIT_SECTION, editsection);
    yield takeLatest(DELETE_SECTION, deletesection);
    yield takeLatest(EDIT_LIBRARY_SECTION, editlibrarysection);
    yield takeLatest(DELETE_LIBRARY_SECTION, deletelibrarysection);
    yield takeLatest(DOWNLOAD_FROM_LIBRARY, fromlibrary);
    yield takeLatest(TRANSFER_TO_LIBRARY, tolibrary);
    yield takeLatest(GET_CUSTOMER, getcustomer);
    yield takeLatest(GET_CUSTOMER_PROJECTS, allcustomerprojects);
    yield takeLatest(GET_PROJECT_SECTION, getprojectsections);
    yield takeLatest(EDIT_LIBRARY_SECTION_ITEM, editlibraryitem);
    yield takeLatest(DELETE_LIBRARY_SECTION_ITEM, deletelibraryitem);
    yield takeLatest(EDIT_SECTION_ITEM, editsectionitem);
    yield takeLatest(DELETE_SECTION_ITEM, deletesectionitem);
    yield takeLatest(UPDATE_ORDER_LIBRARY, updateorderlibrary);
    yield takeLatest(UPDATE_ORDER_SECTION, updateordersection);
    yield takeLatest(EDIT_PRIORITY, editpriority);
    yield takeLatest(DOWNLOAD_ITEM_FROM_LIBRARY, downloaditemfromlibrary);
    yield takeLatest(EDIT_ADDITIONALS, editadditional);
    yield takeLatest(GET_ALL_MESSAGES, getmessages);
    yield takeLatest(UPDATE_MESSAGE, updatemessagetext);
    yield takeLatest(DELETE_MESSAGE, deletemessage);
  };
};

src/store/configureStore.js
import {
  createStore,
  combineReducers,
  compose,
  applyMiddleware,
} from 'redux';

import createSagaMiddleware from 'redux-saga';
import { reducer as form } from 'redux-form';
import { routerReducer, routerMiddleware } from 'react-router-redux';

import history from '../utils/history';

import userReducer from '../reducers/user';
import tokenReducer from '../reducers/auth';
import exceptionReducer from '../reducers/serverException';
import settingsReducer from '../reducers/settings';
import projectReducer from '../reducers/project';
import loginExceptionReducer from '../reducers/serverExceptionLogin';
import successReducer from '../reducers/success';
import projectsReducer from '../reducers/projects';
import libraryReducer from '../reducers/library';
import sectionReducer from '../reducers/section';
import messageReducer from '../reducers/message';

const rootReducer = combineReducers({
  form,
  router: routerReducer,
  user: userReducer,
  auth: tokenReducer,
  serverException: exceptionReducer,
  loginException: loginExceptionReducer,
});

```

```
    success: successReducer,  
    settings: settingsReducer,  
    project: projectReducer,  
    projects: projectsReducer,  
    library: libraryReducer,  
    section: sectionReducer,  
    message: messageReducer,  
  });  
  
const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ || compose;  
const router = routerMiddleware(history);  
  
export const sagaMiddleware = createSagaMiddleware()  
  
const initStore = () => {  
  return createStore(rootReducer, composeEnhancers(applyMiddleware(sagaMiddleware,  
    router)));  
};  
  
export default initStore;
```

ДОДАТОК Б
Слайди презентації

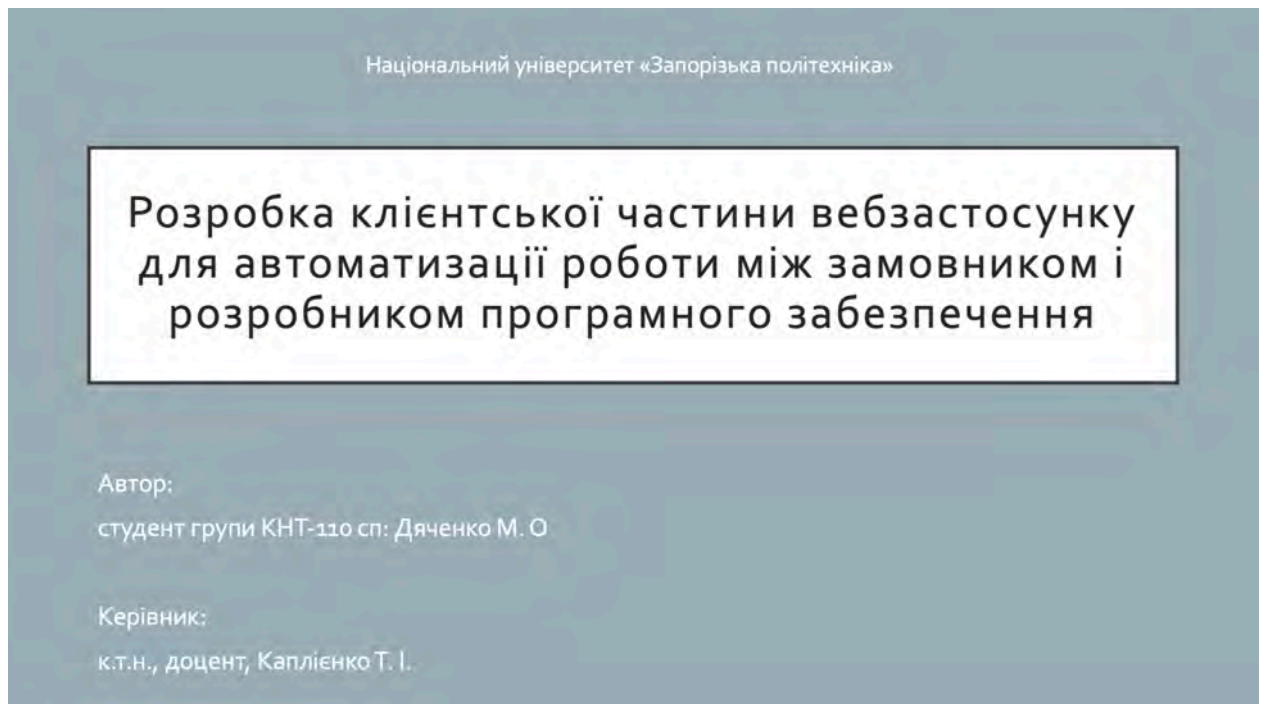


Рисунок Б.1 – Слайд № 1

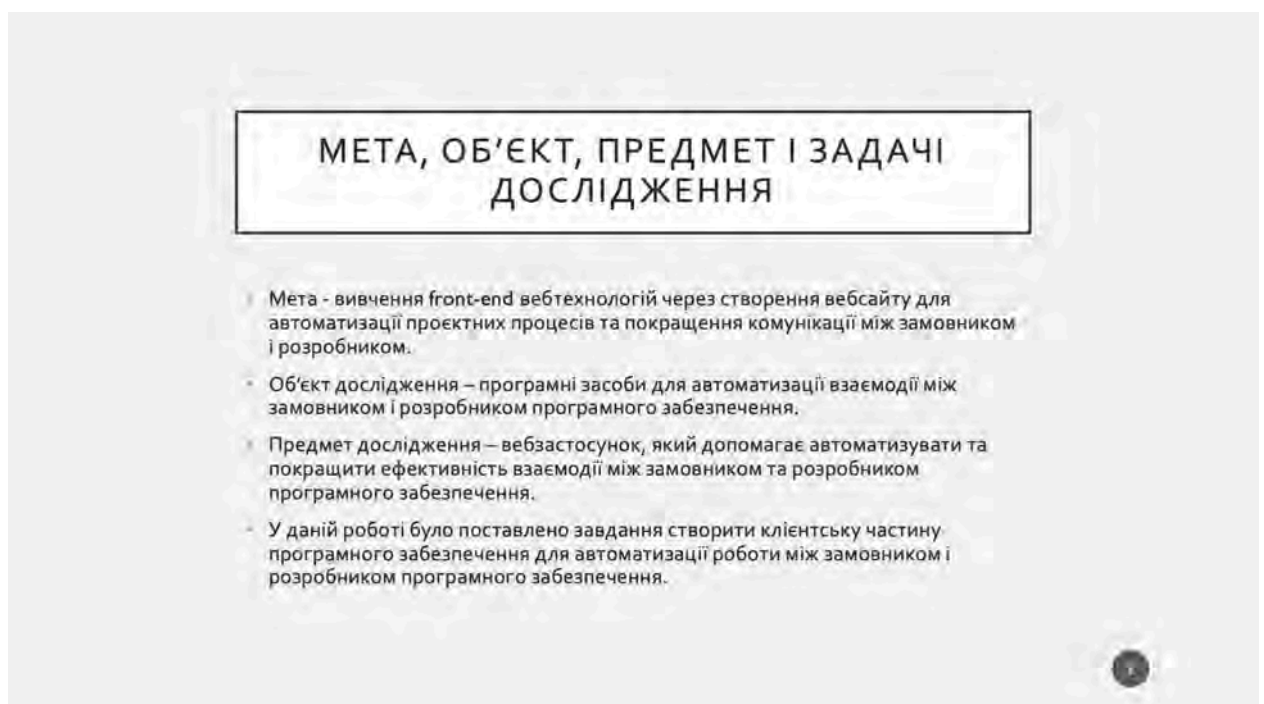


Рисунок Б.2 – Слайд № 2

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Визначено зміст сутностей:

- розробники програмного забезпечення;
- програмне забезпечення;
- розробка програмного забезпечення;
- вимоги до програмного забезпечення;
- програмування;
- системна інтеграція;
- впровадження програмного забезпечення;
- клієнт-замовник;
- вебзастосування.

Рисунок Б.3 – Слайд № 3

ОГЛЯД ПРОГРАМНИХ АНАЛОГІВ

Система управління проектами	Опис	Переваги	Масштаби команди
Monday.com	Широкий спектр функцій та можливостей для команд будь-якого розміру.	- Легкість використання та настройки; - Інтуїтивний інтерфейс.	Різнi розміри команди.
Basecamp	Проста та ефективна система управління проектами з усіма необхідними інструментами на одній платформі.	+ Простота використання та легкий доступ до інформації; - Зручні інструменти для спільної роботи та зберігання інформації.	Різнi розміри команди.
JetBrains Space	Комплексна платформа управління проектами для розробників, що забезпечує широкий функціонал та можливості.	- Інтегрований підхід до розробки ПЗ; - Широкі можливості для взаємодії та спільної роботи команди розробників.	Великі компанії, команди розробників.

Рисунок Б.4 – Слайд № 4

ВИБІР ФРЕЙМВОРКУ РОЗРОБКИ ВЕБЗАСТОСУНКУ		
Критерій	React	Angular
Більше свободи в виборі сторонніх бібліотек	Так	Ні
Тестування	Гірше	Краще
Підтримує легку композицію компонентів	Так	Ні
Підтримує хукі для управління життєвим циклом компонентів	Так	Ні
Має вбудовані засоби для формування анімацій	Ні	Так

Рисунок Б.5 – Слайд № 5



Рисунок Б.6 – Слайд № 6



Рисунок Б.7 – Слайд № 7

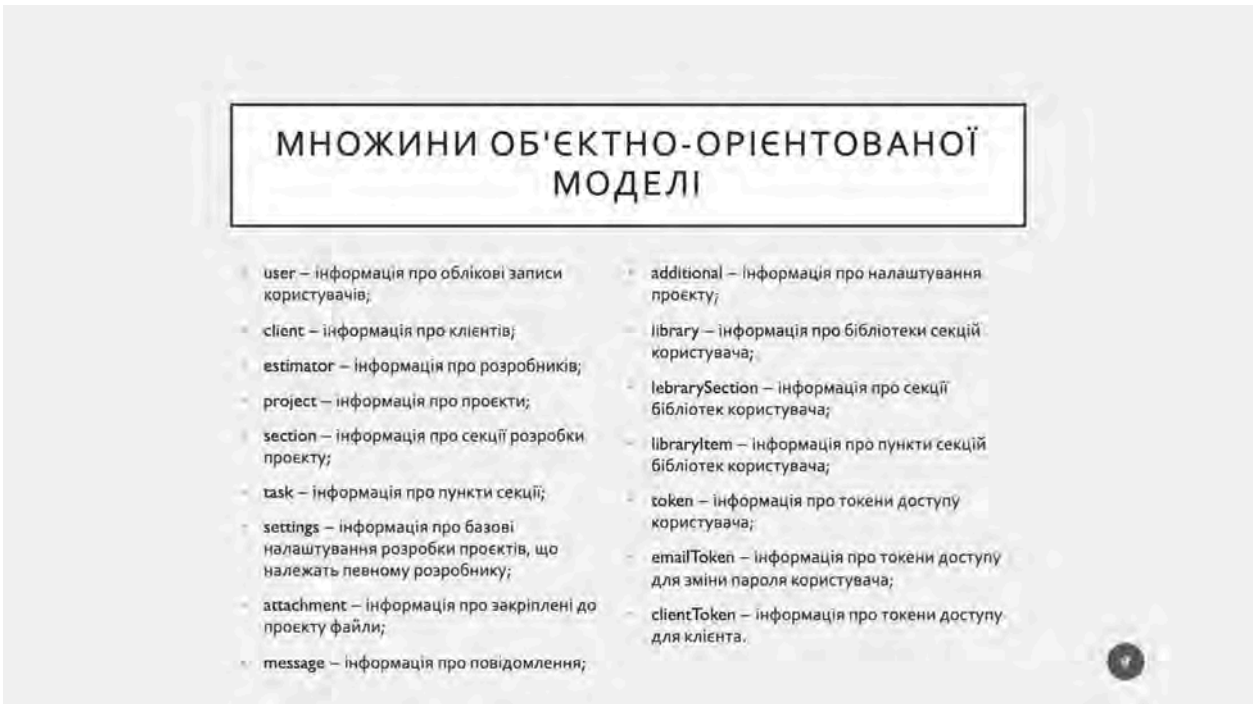


Рисунок Б.8 – Слайд № 8

ВИБІР БІБЛІОТЕКИ КЕРУВАННЯ СТАНОМ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБЗАСТОСУНКУ		
Критерій	Redux-saga	MobX
Використовує єдине глобальне сховище для усіх даних	Так	Ні
Легко тестувати та відлагоджувати, завдяки чистим функціям і використанню генераторів ES6	Так	Ні
Парадигма	Flux-архітектура, базується на діях, змінниках і сховищі	Реактивне програмування, базується на спостережуваних значеннях і діях
Продуктивність	Кращий для великих застосунків з великим обсягом даних	Кращий для невеликих та середніх застосунків
Мутація стану	Ні, стан незмінний	Так, дозволяє пряму мутацію стану

Рисунок Б.9 – Слайд № 9

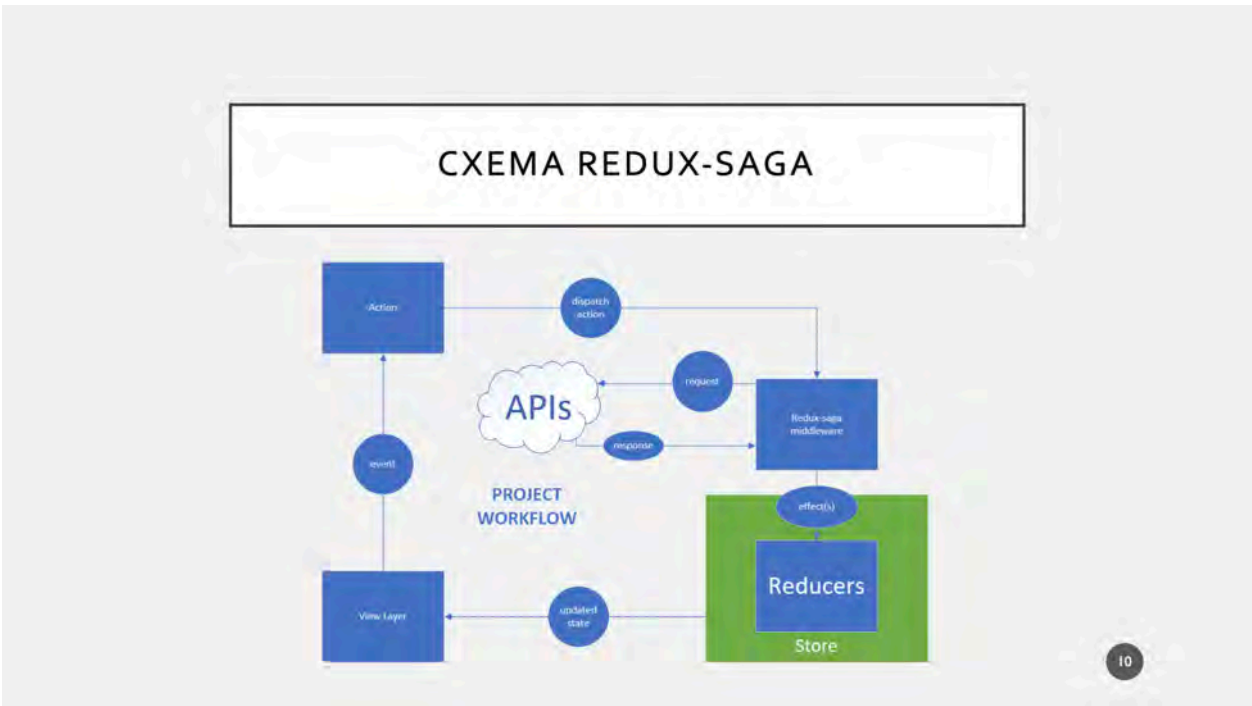


Рисунок Б.10 – Слайд № 10

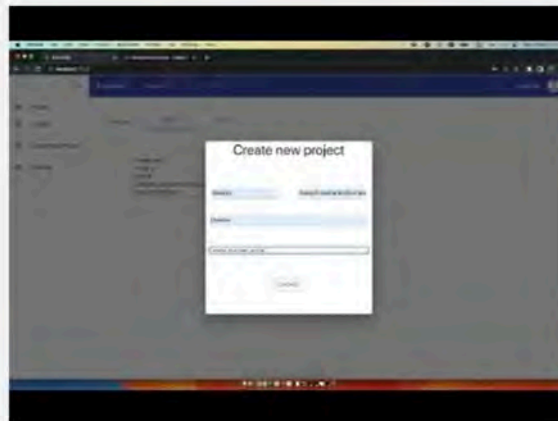
СХЕМА ШЛЯХІВ

```
export default class MainRouter extends Component {
  render() {
    return (
      <Router history={history}>
        <Switch>
          <Route exact path="/" component={projects} //сторінка з усіма проектами користувача
          <Route exact path="/login" component={Login} //відображає сторінку авторизації
          <Route exact path="/signup" component={Signup} // відображає сторінку реєстрації
          <Route exact path="/forgotpassword" component={forgotPassword} //відображає сторінку у разі забутого паролю
          <Route exact path="/setpassword" component={setPassword} //відображає сторінку відновлення паролю
          <Route exact path="/checkemail" component={checkEmail} //відображає сторінку з повідомленням про необхідність перевірки пошти
          <Route exact path="/profilesettings" component={profileSettings} //відображає сторінку з особистими налаштуваннями користувача
          <Route exact path="/settings" component={settings} //відображає сторінку з налаштуваннями проектів користувача
          <Route exact path="/projectpage/:projectId" component={projectPage} //відображає сторінку обраного проекту
          <Route exact path="/customerpage" component={customerView} //відображає сторінку замовника
          <Route
            exact
            path="/customerpage/project/:projectId/token" //відображає сторінку певного проекту замовника
            component={customerProjectPage}
          />
        </Switch>
      </Router>
    );
  }
}
```

11

Рисунок Б.11 – Слайд № 11

ІНТЕРФЕЙС ЗАСТОСУВАННЯ



12

Рисунок Б.12 – Слайд № 12

ВИСНОВКИ

- Застосунок відзначається здатністю підвищити ефективність та якість взаємодії між замовником і розробником, спрощуючи процеси угод, переговорів та обробки даних.
- Цей вебзастосунок надає можливість огляду всіх проектів, які виконуються замовником та розробником, дозволяючи переглядати, редагувати, видаляти різноманітні дані проекту, а також налаштовувати особистий обліковий запис для зручного та ефективного користування.
- ПЗ може ефективно функціонувати на будь-якій операційній системі. Інтерфейс забезпечення є інтуїтивно зрозумілим та зручним у використанні, тому користувачам не потрібно витрачати багато часу на вивчення принципів роботи з ним.
- Розроблений застосунок не має альтернатив на сучасному ринку програмних продуктів, поєднує кращі методи аналогів та доповнений власними принципами, заснованими на спостереженнях.

Рисунок Б.13 – Слайд № 13