НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 6.050102 «Комп'ютерна інженерія»

на тему: «Система моніторингу робочого часу працівників»

Виконав:
студент IV курсу, групи ІО-63
Зінкевич Павло Олегович _____

Керівник:
проф. д. т. н.
Луцький Георгій Михайлович _____

Консультант нормоконтроль:
проф. д. т. н.
Сімоненко Валерій Павлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 6.050102 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ СЕРГІЙ СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проект студенту
Зінкевичу Павлу Олеговичу

1. Тема проекту «Система моніторингу робочого часу працівників», керівник проекту Луцький Георгій Михайлович, проф., д.т.н., затверджені наказом по університету від «07» травня 2020 р. №1081-С

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту *Технічна документація. Теоретичні та статистичні дані. Середовище розробки Visual Studio Code, JavaScript. Фреймворк ReactJS, бібліотека Material-UI.*

4. Зміст пояснювальної записки *Аналіз предметної області, дослідження методики побудови веб-додатку на основі фреймворку ReactJS та бібліотеки Material-UI, система моніторингу робочого часу працівників.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) блок-схема алгоритму, функціональна схема, структурна схема сценарію використання

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	*Затвердження теми роботи*	*01.09.2019*	
2	*Вивчення та аналіз завдання*	*30.03.2020*	
3	*Розробка архітектури та загальної структури систем*	*10.04.2020*	
4	*Розробка структур окремих підсистем*	*22.04.2020*	
5	*Програмна реалізація системи*	*04.05.2020*	
6	*Оформлення пояснювальної записки*	*17.05.2020*	
7	*Передзахист*	*26.05.2020*	
8	*Захист*	*16.06.2020*	

Студент Павло ЗІНКЕВИЧ

Керівник Георгій ЛУЦЬКИЙ

Анотація

Дана дипломна робота присвячена розробці та аналізу системи для моніторингу робочого часу працівників, шляхом розробки спеціального веб-додатку для керівників в IT-компаніях, в малих бізнесах та підприємствах.

В даній роботі проаналізовано програмне забезпечення існуючих рішень та визначені оптимальні кроки для впровадження системи моніторингу робочого часу працівників на робочих місцях. Реалізовано веб-додаток з підключенням до мережі інтернет.

Для написання додатку, обрано мову JavaScript, обумовлене наявністю знань. Клієнтська частина розроблялася з використання фреймворку ReactJS, щоб вирішити проблему часткового оновлення веб-сторінки. В розробці інтерфейсів була використана версія популярної бібліотеки material-ui. Управління внутрішнім станом додатку здійснюється через Redux.

Annotation

This thesis is devoted to the development and analysis of time tracking system, by developing a special browser application for managers in IT-companies, small businesses and enterprises.

In this work the software of existing solutions is analyzed and the optimal steps for the implementation of a time tracking system for monitoring the working hours of employees in the workplace is determined. A browser application is implemented with an Internet connection.

To write an application, the language of javascript is selected based on availability of knowledge. The client part was developed using the ReactJS framework to solve the problem of partial web page refresh. In the development of interfaces a version of the popular library material-ui was used. The internal state of the application is managed through Redux.

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня__бакалавр___

на тему: “Система моніторингу робочого часу працівників”

Київ – 2020 року

Зміст

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Система моніторингу робочого часу працівників». Область застосування даної системи: альтернатива існуючим методам моніторингу робочого часу, який працівники провели на робочому місці.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки цієї системи є завдання на виконання дипломного проекту, системи моніторингу робочого часу працівників, затвердженого кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного дипломного проекту є розробка веб-додатку, що має відстежувати час, який працівник провів за виконанням роботи, та робити відповідні графіки щодо виконаної роботи та ефективності працівників.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки даного дипломного проекту є науково-технічна література, навчальні посібники по JavaScript, публікації в спеціалізованих виданнях, та публікації на цю тему в мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до веб-додатку, що розробляється:

- Моніторинг часу, проведеного працівником за роботою.
- Створення графіків, що показують кількість напрацьованих годин та ефективність працівника за вказаний проміжок часу.
- Створення команд, в склад яких входять підлеглі та лідер, що керує командою.

5.2. Вимоги до програмного забезпечення:

- Операційна система Windows, macOS, Linux.
- Доступ до мережі Інтернет.
- Будь-який браузер, який підтримує JavaScript (наприклад, Google Chrome, Opera, Firefox та інші).

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	30.01.2020
Складання і узгодження технічного завдання	11.02.2020
Створення модулів системи, що розробляється	14.03.2020
Тестування окремих модулів системи	10.04.2020
Доопрацювання, налагодження і виправлення помилок	04.05.2020
Оформлення документації дипломної роботи	17.05.2020

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

до дипломної роботи
освітньо-кваліфікаційного рівня бакалавр

на тему: “Система моніторингу робочого часу працівників”

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ДП.467100.001 ТЗ	Технічне завдання	4	
3	А4	ДП.467100.002 ВП	Відомість проекту	1	
4	А4	ДП.467100.003 ПЗ	Пояснювальна записка	60	
5	А4	ДП.467100.004 Д1	Схема структурна	1	
6	А4	ДП.467100.005 Д2	Блок-схема алгоритму	1	
7	А4	ДП.467100.006 Д3	Схема функціональна	1	

					ДП.467100.002 ВП		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Зінкевич П.О.*			*Відомість дипломного проекту*	*Літ.*	*Аркуш* / *Аркушів*
Перевір.		*Луцький Г.М.*					*1* / *1*
Н. Контр.		*Сімоненко В.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-63*	
Затв.							

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту

на тему: “Система моніторингу робочого часу працівників”

Київ – 2020 року

Зміст

					ДП.467100.003 ПЗ		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Зінкевич П.О.*			*Система моніторингу робочого часу працівників Пояснювальна записка*	*Літ.*	*Аркуш* / *Аркушів*
Перевір.		*Луцький Г.М.*				Т	*1* / *60*
Н. Контр.		*Сімоненко В.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-63*	
Затв.							

ПЕРЕЛІК СКОРОЧЕНЬ

JS – JavaScript

IT – Інформаційні технології

SPA – Single Page Application

MPA – Multi Page Application

SEO – Search engine optimization

AJAX – Asynchronous JavaScript and XML

XML – Extensible Markup Language

MVVM – Model-view-viewmodel

MVC – Model-view-controller

SaaS – Software as a service

CLI – Command-line interface

JSX – JavaScript XML

HTML – Hypertext Markup Language

PWA – Progressive Web App

CSS – Cascading Style Sheets

ОПП – об'єктно-орієнтоване програмування

VS Code – Visual Studio Code

ESLint – ECMAScript Lint

IDE – Integrated development environment

AGPL – Affero General Public License

ВСТУП

Моніторинг робочого часу співробітників – це один з найефективніших способів, щоб підтримувати якісну, продуктивну та злагоджену роботу між працівниками в колективі. Наразі, системи моніторингу робочого часу працівників набагато популярніші, ніж має здатися на перший погляд.

На сьогоднішній день більшість компаній світу використовують системи моніторингу робочого часу працівників, адже це найефективніший спосіб оцінки виконаної роботи співробітників. За допомогою системи моніторингу робочого часу працівників, можливо виявляти та вирішувати проблеми компанії ще до того, як ці проблеми стануть глобальними та приведуть до негативних наслідків.

Система моніторингу робочого часу робітників запобігає виникненню багатьох проблем. Це особливо важливо, якщо працівники отримують заробітну плату в залежності від кількості напрацьованих годин. Також, в залежності від функціоналу, система моніторингу робочого часу працівників також може запобігати конфлікти між колегами в командах, пов’язані із приналежністю виконаної роботи до конкретного працівника, адже системи моніторингу робочого часу працівників зазвичай включають в себе не тільки функцію стеження за кількістю відпрацьованих годин. В результаті технологічного розвитку, а також постійного наростання популярності IT-сфери в сучасному світі, набагато збільшилась і кількість працівників, які пов’язані з IT. Через це, необхідність в системах моніторингу робочого часу працівників значно зросла. Системи моніторингу робочого часу працівників знайдуть застосування у кожній сучасній IT-компанії.

Метою даного дипломного проекту є створення простої в використанні системи моніторингу робочого часу працівників у вигляді веб-додатку з інтуїтивним інтерфейсом.

Наукова новизна мого додатку полягає у легкій конструкції, що полегшує навантаження на сервер та базу даних. Це досягнуто завдяки використанню бібліотеки Material-UI на основі фреймворку React.js для створення компонентів додатку, а також завдяки використанню сучасної бібліотеки Redux.

РОЗДІЛ 1.

ТЕОРЕТИЧНА ЧАСТИНА

1.1 Аналіз предметної області

Моніторинг – система, за допомогою якої здійснюється постійне спостереження та оцінка змін стану будь-якого технічного, соціального, природного та інших об’єктів. В нашому випадку моніторинг (також можна використовувати слово «відстеження» для опису даної проблеми) застосовується задля визначення конкретної кількості величини – часу. Система моніторингу робочого часу працівників призначена для відстеження часу, під час якого працівник виконував свою роботу.

Але відстежувати тільки час буде не ефективним вирішенням проблеми, якщо завдання працівників будуть складними та потребуватимуть більше одного дня на виконання. Тому, окрім відстеження часу роботи, в системи моніторингу робочого часу працівників зазвичай додають й інші функції. Наприклад, хорошою практикою є додання в систему моніторингу робочого часу працівників функцію знімку екрану, яка працює через певну кількість часу. Можливо також додати в систему моніторингу робочого часу працівників функцію, яка відстежує кількість натиснутих клавіш у хвилину, або кількість кліків мишкою та інше.

Задля ефективного спостереження за своїми працівниками, керівникам (або іншим відповідальним за діяльність своїх підлеглих людям) необхідно отримати ряд питань, що є дуже складним завданням. По-перше, потрібно дізнатися, коли саме працівники починають та закінчують роботу. Також потрібно точно знати скільки фактично годин триває робочий день персоналу. Важливо враховувати й те, скільки часу працівники витрачають на виконання поставлених їм задач, а скільки – в особистих цілях. Потрібно знати чим займаються працівники протягом робочого дня та наскільки ефективно працівники використовують робочий час. На превеликий жаль, у більшості випадків пошук

відповідних рішень для моніторингу робочого часу працівників перетворюється у велику проблему. Саме тому і потрібні систему моніторингу робочого часу працівників, при правильній побудові, будуть здатні вирішити більшість вищевказаних проблем.

1.2 Аналіз існуючого програмного забезпечення

З розширенням IT-сфери та підвищенням кількості працівників в IT-сфері, зростає необхідність в простих в використанні та ефективних системах моніторингу робочого часу працівників. Оскільки проблема є неймовірно актуальною, вже з’явилися різні варіанти її вирішення.

Серед найпоширеніших рішень проблеми побудови ефективної системи моніторингу робочого часу працівників наявні проблеми, що вимагають додаткового доопрацювання і дорого проекту по впровадженню. У таблиці 1.1 наведені приклади програмного забезпечення для моніторингу робочого часу працівників.

Основною причиною для розробки власного спеціалізованого рішення стала відсутність якісного, простого та безкоштовного програмного забезпечення, яке задовольняє вимогам ефективної системи моніторингу робочого часу працівників.

Таблиця 1.1 – порівняння існуючих рішень

Найменування	Переваги	Недоліки
Timenotes	наявність системи звіту; безкоштовний період випробування (30 днів); наявність конфігурації оповіщень	висока вартість продукту; наявність помилок у програмному забезпечені, які постійно потребують виправлень: без попередньої оплати більшість функціоналу недоступна навіть у пробному режимі
Yaware	безкоштовний період випробування (14 днів); наявність обліку часу нарад та перерв; можливість виявлення найкращого та найгіршого співробітника	висока вартість продукту; наявність лише однієї мови інтерфейсу(російської); можливість у керівників слідкувати за веб-камерою працівника та усіма сайтами, на які заходить працівник, може погіршити мораль та довіру в команді, що в результаті позначиться на ефективності роботи
Harvest	підтримка від великої кількості малих бізнесів; безкоштовний період випробування (30 днів); наявність розкладу та функції конфігурування команд	висока вартість продукту; обмежений функціонал, який потребує додаткових доопрацювання; складний інтерфейс;

1.3 Постановка задачі

Аналіз предметної області дав зрозуміти, які саме функції повинна виконувати система моніторингу робочого часу працівників. Система моніторингу робочого часу працівників – це онлайн-система, призначена для всебічного моніторингу робочого часу працівників за комп'ютером та підвищення продуктивності його використання. Завдання системи моніторингу робочого часу працівників полягає в тому, щоб фіксувати час початку та закінчення роботи працівників за комп'ютером. Також необхідно здійснювати моніторинг часу виконання робочих завдань та відсутність на робочому місці. Необхідна можливість збирати інформацію щодо використання працівником додатків, програм і сайтів. Потрібно мати функцію розподілення ресурсів за категоріями, наприклад, «продуктивно», «середньо», «непродуктивно» виходячи з робочих обов'язків персоналу та результатів їх роботи.

Окрім того, що керівнику необхідно мати можливість отримувати оперативні звіти про діяльність персоналу за комп'ютером, самі працівники також повинні мати доступ до даних, що мають відношення до проведеної ними роботи. Цей функціонал необхідний, аби працівники мали мету вдосконалюватися, що в кінцевому результаті приводить до підвищення продуктивності в команді та ефективності виконання поставленого завдання.

Програмне забезпечення системи моніторингу робочого часу працівників повинне мати простий в користуванні та інтуїтивний інтерфейс. Потрібно мати можливість створення облікового запису та спосіб входу у вже створений обліковий запис. Весь функціонал повинен бути поділений на окремі розділи, що містять відповідний до назви розділу функціонал. Для цього необхідно мати спосіб переходу у різні розділи. Як показує досвід, найкращою практикою для вирішення цього питання є наявність простого меню зверху або зліва сторінки. Таким чином, наприклад, у розділі (вкладці) «Teams» повинен бути функціонал,

який пов’язаний зі створенням команд, їх конфігурацією та зв’язком між членами команди.

Серед вкрай важливих розділів необхідно мати розділ з профілем співробітника, У цьому розділі необхідна бути присутня основна та коротка інформація про працівника, аби мати змогу зв’язатися з ним та взаємодіяти у майбутньому. Також повинен бути розділ з таймером для відліку часу, який запускає працівник перед початком роботи. І останній, але не менш важливий розділ повинен мати усі збережені дати, що стосуються кількості напрацьованих годин та виконання завдань (проектів).

Таким чином, можна виділити основні пункти, які необхідні для вирішення заданої проблеми:

- Простий та інтуїтивний інтерфейс.
- Можливість створювати обліковий запис та входити в нього після реєстрації.
- Можливість відстеження кількості напрацьованих годин.
- Можливість перегляду збережених даних щодо виконаної роботи за день/тиждень/місяць.
- Можливість створювати команди, додавати нових працівників у команду та редагувати склад команди.
- Можливість редагувати особисту інформацію у розділі «Profile».

ВИСНОВКИ ДО РОЗДІЛУ 1

У сучасному світі, з постійним розвитком ІТ-технологій та ІТ-сфери в цілому, помітно зростає й кількість людей, які залучені в ІТ-сферу. А зі збільшенням кількості ІТ-працівників постало питання сучасного вирішення проблеми розподілу заробітної плати працівникам великих компаній та бізнесів.

Система моніторингу робочого часу працівників є найкращим вирішенням даної проблеми. Системи моніторингу робочого часу працівників дуже актуальні. Вони є важливими складовими в роботі ІТ-працівників, оскільки суттєво полегшують вирішення питання щодо ефективності кожного окремого працівника на робочому місці.

В даному розділі були розглянуті різноманітні системи моніторингу робочого часу працівників, а також наведені відповідні приклади. Розглянуті системи моніторингу робочого часу працівників мають великий обсяг функціоналу та можуть вирішувати різноманітні завдання, які їм поставлені. Незважаючи на це, існуючі системи моніторингу робочого часу працівників мають також і недоліки. Найбільшим недоліком серед яких є величезна вартість продукту, вслід чого система моніторингу робочого часу працівників стає недоступною для великого прошарку населення.

Через це, розробка дешевого веб- додатку. що включає в себе систему моніторингу робочого часу працівників, є актуальною проблемою на сьогодення. Дана система моніторингу робочого часу працівників повинна бути легка в використанні. Завдяки цьому, структуру веб-додатку можливо буде змінювати у відповідності до вимог користувачів.

РОЗДІЛ 2.
ВИБІР ТЕХНОЛОГІЙ, МЕТОДІВ, ПЛАТФОРМ

2.1 Вибір інформаційної моделі

Першим кроком до вирішення поставлених завдань є визначення структури бази даних, в першу чергу слід визначити модель відносин бази даних.

Найпопулярнішою базою даних для сучасних додатків є MongoDB. MongoDB – це міжплатформна програма, з відкритим вихідним кодом, яка спеціально призначена для зберігання ієрархічних структур даних (документів), реалізована за допомогою підходу NoSQL та не потребує опису схеми таблиць.

MongoDB – це база даних документів із необхідною масштабованістю і гнучкістю, з наявністю необхідних для побудови системи моніторингу робочого часу працівників запитами та індексаціями. Модель документа MongoDB проста у вивченні та використанні, але все ж надає всі можливості, необхідні для задоволення найскладніших вимог у будь-якому масштабі.

Серед переваг, які надає база даних документів MongoDB, можна виділити такі:

- MongoDB зберігає дані у гнучких, схожих на JSON документах, тобто значення поля можуть змінюватися від документа до документа, а структура даних може змінюватися з часом.
- Модель документа відображає об’єкти у коді програми, що полегшує роботу з даними.
- Спеціальні запити, індексація та агрегація в режимі реального часу забезпечують потужні способи доступу та аналізу даних.
- MongoDB – це розподілена база даних по своїй суті, тому вбудована та зручна у використанні висока доступність, горизонтальне масштабування та географічний розподіл.

- MongoDB безкоштовний у використанні. Версії, випущені до 16 жовтня 2018 року, публікуються під AGPL. Усі версії, випущені після 16 жовтня 2018 року, включаючи виправлення для попередніх версій, публікуються під загальнодоступною ліцензією сервера (SSPL) v1.

Беручи до уваги усі вищевказані характеристики бази даних MongoDB, для побудови системи моніторингу робочого часу працівників дана база даних підходить. Відповідно, в якості серверної системи управління базами даних була обрана MongoDB.

2.2 Вибір засобів розробки

Першим кроком до вирішення поставлених завдань є визначення структури бази даних, в першу чергу слід визначити модель відносин бази даних.

Сьогодні існує величезний попит на складні веб-додатки, які з кожним днем все більше витісняють з ринку настільні програми. Хоча останнім часом програми, що працюють під браузером, поступово починають поступати свою популярність мобільним додаткам, на сьогоднішній день існують два основних шаблона розробки веб-додатків – SPA (Single Page Application) і MPA (Multi Page Application).

MPA – це традиційний веб-додаток. При такому підході кожний раз, коли додаток запитує дані або відправляє їх на сервер, воно змушене отримувати нову сторінку в повному обсязі, а потім візуалізувати її в браузері. Для відносно простих додатків такий підхід цілком підходить і не викликає незручностей. Але коли мова йде про складний додаток з об’ємним для користувача інтерфейсом, з високим ступенем інтерактивності, то на сторінці з'являється безліч користувальницьких інструментів. Все це веде до того, що розмір трафіку помітно збільшується, і при багатосторінковому підході неминуче з'являються проблеми з продуктивністю.

MPA мають більш класичну архітектуру. Кожна сторінка відправляє запит на сервер і повністю оновлює всі дані. Навіть якщо ці дані невеликі. Таким чином витрачається продуктивність на відображення одних і тих же елементів. Відповідно це впливає на швидкість і продуктивність.

Багато розробників, для того щоб підвищити швидкість і зменшити навантаження, використовують JavaScript/jQuery. Гарним прикладом є оновлення товарів без перезавантаження сторінки, при використанні фільтрів в інтернет магазині. Це набагато зручніше і головне швидше. Головні переваги MPA:

- Легка SEO оптимізація. Архітектура MPA дозволяє досить легко оптимізувати кожну сторінку під пошукові системи.
- Легка розробка. Як правило, для розробки багатосторінкового додатку потрібен менший стек технологій.
- Велика кількість рішень.

Використовуючи MPA ви можете знайти підходяще коробкове (відомі всім (масові) програмні продукти, які містять дуже мало помилок) рішення. Наприклад, використовувати Magento, OpenCart для розробки e-commerce веб-додатку або Dolphin, Elgg для розробки соціальних мереж. Серед недоліків MPA можна також виділити і такі пункти:

- Для розробки мобільних додатків буде потрібно набагато більше часу. У більшості випадків потрібно написання back-end з нуля.
- Складно розділити front-end і back-end. Як правило вони дуже тісно взаємодіють один з одним. Ускладнюється робота front-end і back-end розробників.

Основною перевагою MPA є хороша SEO оптимізація і величезна кількість коробкових рішень.

На початку 2000-х років MPA був вдосконалений з появою AJAX, який дозволяє оновлювати тільки частину сторінки, а не всю сторінку

цілком. З одного боку, це дозволило поліпшити продуктивність, з іншого ж - ускладнило веб-сторінку.

Згодом, через 10 років виник шаблон SPA. Насправді, SPA - це еволюція шаблону MPA + AJAX. При такому підході лише каркас веб-сторінки будується на сервері, все інше генерується засобами JavaScript.

SPA запитує розмітку і дані окремо, і візуалізує результати безпосередньо в браузері. Це стало можливим завдяки новим front-end - фреймворкам, що реалізують шаблон MVVM, таким як AngularJS і KnockoutJS.

SPA дозволяють імітувати роботу настільних додатків. Архітектура влаштована таким чином, що при переході на нову сторінку, оновлюється тільки частина контенту. Таким чином, немає необхідності повторно завантажувати одні й ті ж елементи. Це дуже зручно для розробників і користувачів. Для розробки SPA використовується одна з найпопулярніших мов програмування - JavaScript. Невеликий веб додаток можна зробити за допомогою бібліотеки jQuery. Але відразу варто зазначити, що jQuery дуже погано підходить для розробки великих проектів. Зазвичай рекомендується використовувати більш потужні технології для розробки SPA. Для цієї мети гарно підійде React.js, Angular.js, Vue.js та інші front-end фреймворки/бібліотеки. Їх архітектура дозволяє розробляти гнучкі веб-додатки. Більш того на базі фреймворків можна будувати мобільні додатки з повторним використанням коду. Такі можливості дає React-Native та Ionic. Серед основних переваг SPA можна відзначити такі:

- Продуктивність. Так як SPA не оновлює всю сторінку, а тільки потрібну частину, це істотно підвищує швидкість роботи.

- Висока швидкість розробки. Готові бібліотеки і фреймворки дають потужні інструменти для розробки веб-додатків. Над проектом можуть паралельно працювати back-end і front-end розробники. Завдяки чіткому поділу, вони не будуть заважати один одному.

- Мобільні додатки. SPA дозволяє легко розробити мобільний додаток на основі готового коду.

При всіх своїх перевагах, SPA має деякі недоліки, які стримують зростання популярності. Серед таких недоліків можна вказати такі:

- Погана SEO оптимізація. SPA працює на основі JavaScript і завантажує інформацію за запитом з боку клієнта. Пошукові системи насилу можуть імітувати дану поведінку. Тому більшість сторінок просто недоступні для сканування пошуковими ботами, тому просування сайту в 2020 може бути ускладнено.

- Неактивний JavaScript. Деякі користувачі відключають javascript в своїх браузерах, а без нього додаток не працюватиме.

- Низький рівень безпеки.

JavaScript має низький рівень безпеки, але якщо використовувати сучасні фреймворки, вони можу зробити веб-додаток безпечним. Але варто звернути увагу й на те, що використання jQuery може істотно знизити безпеку вашого проекту в 2020 році.

SPA веб-додатки добре підходять для розробки динамічних платформ, з невеликим обсягом даних. Крім того, якщо буде потрібно в майбутньому побудувати мобільний додаток, SPA відмінно підійде як основа. Основним недоліком, який стримує швидке зростання популярності SPA це погана SEO оптимізація. В проектах, в яких SEO має найважливіший пріоритет, варто використовувати MPA.

Беручи до уваги вищевказані плюси та мінуси стосовно MPA та SPA, підведемо підсумки щодо переваг та недоліків, які має SPA порівняно з MPA. Серед переваг SPA по відношенню до MPA можна виділити такі:

- Більш швидке завантаження сторінок.
- Нові маршрути (routes) генеруються за допомогою самого додатку.

- Покращене сприйняття призначеного для користувача інтерфейсу, оскільки завантаження даних з сервера відбувається у фоновому режимі.
- Немає необхідності писати код на сервері для візуалізації сторінки.
- Поділ на front-end та back-end розробку.
- Спрощена розробка під мобільні додатки; наявна можливість повторно використовувати один і той же серверний код для веб-додатків і мобільних додатків.

Тим не менш, SPA також має ряд недоліків, порівняно з MPA. Серед таких недоліків SPA по відношенню до MPA можна виділити такі:

- Складні клієнтські фреймворки, які потрібно завантажувати на кожен клієнт.
- Веб-форми не компілюються, а тому їх складніше налагоджувати і вони містять більше вразливостей, якими можуть скористатися зловмисники.
- SEO оптимізація. SPA ускладнює оптимізацію сайту під пошукові движки. Оскільки більша частина веб-сторінки будується на стороні клієнта, боти пошукових систем бачать сторінку зовсім інакше, ніж користувач.

Враховуючи усі вищевказані фактори, постає питання. А чи можливо створити гібридний варіант додатку? Було б ефективно розроблювати на MPA (як на більш безпечному шаблоні), в якому деякі сторінки були б реалізовані як SPA. Такий варіант можливий, але потрібно чітко усвідомлювати, наскільки правильно буде використовувати його для конкретного додатку. Інакше є ризик створити додаток, який буде поєднувати недоліки обох шаблонів. Серед таких недоліків можна виділити такі:

- Додаток може вийти повільним, з огляду на те, що частина сторінок буде формуватися на сервері, а для реалізації частини SPA знадобиться установка клієнтських фреймворків на клієнтську машину.
- Складніша розробка. Доведеться використовувати два фреймворки - MVVM для клієнта і MVC для сервера.
- Front-end і back-end будуть сильно пов'язані.
- Ви не зможете використовувати той самий back-end для мобільних додатків.

Таким чином, змішування SPA та MPA є поганою практикою і не рекомендується, якщо це не необхідно. Кожен з цих шаблонів гарно підходить для своєї області застосування, але їх поєднання зазвичай не є вигідним.

Враховуючи всі вищевказані плюси та мінуси SPA та MPA, необхідно вибрати ту структуру побудови веб-додатку, яка найбільше підійде для простої системи моніторингу робочого часу працівників. Кожна архітектура має свої переваги і недоліки і добре підходить для певного типу проекту. SPA відрізняється своєю швидкість і можливістю на базі готового коду розробляти мобільні додатки. Але в той же час, SPA має погану SEO оптимізацію. Таким чином, дана архітектура відмінно підходить для SaaS платформ, соціальних мереж, закритих спільнот, де пошукова оптимізація не має значення.

MPA більше підходить для створення великих інтернет магазинів, бізнес сайтів, каталогів, ринків і так далі. Добре оптимізована MPA має високу швидкість і продуктивність, але все ж не дозволяє легко розробити мобільний додаток. MPA і SPA з правильною архітектурою добре підходять для розробки масштабованих веб-додатків.

Серед популярних додатків, наприклад як Google Drive, Gmail, Facebook, Twitter, використовується SPA для побудови. З іншого боку, прикладами MPA є сайти Amazon, eBay, e-commerce, блоги, форуми та інші сайти, які продають продукти та різноманітні сервіси.

Беручи до уваги всі вищевказані констатування, для реалізації системи моніторингу робочого часу працівників правильним рішенням буде обрати SPA. Це обумовлено багатьма факторами. В першу чергу, до уваги слід узяти масштаб розроблюваного веб-додатку. Даний веб-додаток повинен мати не дуже великий обсяг контенту, який слід помістити в одну сторінку, тому SPA підійде краще в даній ситуації, ніж MPA. Крім того, SPA покращує сприйняття інтерфейсу, призначеного для користувача, що позитивно впливає на ефективність працівників.

Підводячи підсумки, для побудови системи моніторингу робочого часу працівників було обрано шаблон SPA.

2.3 Вибір мови програмування

JavaScript – мультипарадигмова (одночасно використовує багату сукупність ідей і понять, які визначають стиль написання комп'ютерних програм) інтерпретована мова програмування з динамічною типізацією (прийом, при якому змінна зв'язується з типом в момент надання значення, а не в момент оголошення змінної). Підтримує імперативний, функціональний і об'єктно-орієнтований стилі програмування. Є реалізацією специфікації ECMAScript.

При розробці метою було зробити мову, яка схожа на Java, але при цьому більш проста у використанні. Серед основних особливостей JavaScript можна виділити такі:

- Функції є об'єктами першого класу.
- Об'єкти з можливістю доступу до метаданих класів під час виконання програми.
- Наявність механізму, призначеного для обробки помилок часу виконання та інших можливих проблем (винятків), які можуть виникнути при виконанні програми.
- Функції є анонімними.
- Приведення типів здійснюється автоматично.

- Звільнення пам’яті від об’єктів, які не будуть використовуватися програмою в подальшому відбувається автоматично.

Система моніторингу робочого часу працівників передбачає собою веб-додаток, доступ до якого працівники можуть отримати через браузер. Оскільки SPA працює на основі JavaScript і завантажує інформацію за запитом з боку клієнта, то очевидним вибором мови програмування буде JavaScript.

Серед головних фреймворків на базі JavaScript можна виділити три основних – React.js, Angular.js, Vue.js.

Angular був розроблений Google та вперше випущений у 2010 році, що робить його найстарішим з даних фреймворків. Це фреймворк JavaScript на основі TypeScript. Суттєві зміни відбулися у 2016 році щодо випуску Angular 2 (та відділення “JS” від початкової назви - AngularJS). Остання стабільна версія - Angular 9, яка вийшла у лютому 2020 року.

Vue, також відомий як Vue.js, був розроблений колишнім співробітником Google Evan You у 2014 році. За останні три роки Vue набрав значної популярності, не дивлячись на відсутність підтримки великої компанії. Поточна стабільна версія – 2.6, випущена в лютому 2019 року (з тих пір випускаються невеликі додаткові оновлення). Вкладники Vue підтримуються за допомогою членської платформи Patreon. Vue 3, що наразі перебуває в альфа-фазі, планує перейти до TypeScript.

Незважаючи на те, що компанія Facebook, яка розробила технологію React (також відому як React.js), позиціонує React як бібліотеку, в дійсності, всі тенденції та кроки з розвитку React вказують на те, що React на даний момент вже ближче до фреймворку, ніж до бібліотеки. React був випущений у 2013 році. Facebook широко використовує React у своїх продуктах (Facebook, Instagram, WhatsApp). Поточна стабільна версія - 16.X, була випущена в листопаді 2018 року (з тих пір випускаються додаткові оновлення).

Тепер розглянемо складність ознайомлення з кожним із фреймворків, щоб визначитися з потрібним для побудови системи моніторингу робочого часу працівників фреймворком.

За думкою багатьох розробників, Vue, порівняно з React та Angular, являє собою найлегший для вивчення фреймворк. Vue - це найпростіша основа для навчання більшості веб-розробників. Він найближчий до основ HTML та JavaScript, тому він легкий для сприйняття. Початок роботи з Vue такий же простий, як і додавання єдиного імпорту до вашого HTML-документа. Vue стає складнішим під час створення складніших додатків. Для цього потрібна робота з файлами .vue, які потребують більш складної установки проекту (хоча Vue CLI може зробити це набагато простіше),внаслідок чого доводиться занурюватися в більш складні схеми.

Найскладнішим в освоєнні вважається Angular. Оскільки Angular використовує TypeScript, освоєння Angular видається складнішим, порівняно з іншими фреймворками. Компоненти, модулі та синтаксис, які використовуються, можуть виглядати зовсім не так, як синтаксис JavaScript, до якого легко звикнути. У Angular є багато вбудованих, потужних функцій, які змушують розробників до певних моделей кодування, які можуть бути дуже корисними при побудові додатків.

Говорячи про React, можна сказати, що він не складний у освоєнні, але й не надто легкий. Хоча він використовує підхід «JavaScript – це все», щодо React є два застереження. По-перше, це те, що він найкраще працює з синтаксисом ES6, що може бути складним для початківця розробника. По-друге, це використання JSX, синтаксичного гібрида HTML та JavaScript, який не використовується для типових проектів JavaScript. Спочатку JSX може бути складним, оскільки він схожий на HTML, але все ще залишається JavaScript, викликаючи певну плутанину в розумінні коду.

До переваг фреймворку Angular варто віднести такі:

- Angular використовується разом з TypeScript. Даний фреймворк має виняткову підтримку для цього.
- Angular-language-service – забезпечує інтелектуальні можливості і автозаповнення шаблону HTML-компонента.
- Детальна документація, що дозволяє розробнику отримати всю необхідну інформацію. Однак це вимагає більше часу для навчання.
- Одностороння прив'язка даних, яка забезпечує виняткову поведінку додатка, що зводить до мінімуму ризик можливих помилок.
- Наявність MVVM, яка дозволяє розробникам працювати окремо над одним і тим же розділом програми, використовуючи один і той же набір даних.
- Впровадження залежностей від компонентів, пов'язаних з модулями і модульність в цілому.
- Структура і архітектура спеціально створені для великої масштабованості проекту.

Серед недоліків Angular можна виокремити такі:

- Різноманітність різних структур (Injectables, Components, Pipes, Modules і так далі) Ускладнює вивчення в порівнянні з React і Vue, у яких є тільки «Component».
- Відносно повільна продуктивність, враховуючи різні показники. З іншого боку, це можна легко вирішити, використовуючи так званий «ChangeDetectionStrategy», який допомагає вручну контролювати процес рендерингу компонентів.

До відомих компаній, які використовують Angular, належать: Microsoft, Autodesk, MacDonald's, UPS, Cisco Solution Partner Program, AT & T, Apple, Adobe, GoPro, ProtonMail, Clarity Design System, Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase та багато інших.

До переваг фреймворку Vue варто віднести такі:

- Посилений HTML. Це означає, що Vue має багато характеристик схожих з Angular, а це, завдяки використанню різних компонентів, допомагає оптимізації HTML- блоків.

- Детальна документація. Vue має дуже детальну документацію, яка може прискорити процес навчання для розробників і заощадити багато часу на розробку програми, використовуючи тільки базові знання HTML і JavaScript.

- Адаптивність. Може бути здійснений швидкий перехід від інших фреймворків до Vue через схожість з Angular і React з точки зору дизайну і архітектури.

- Приголомшлива інтеграція. Vue можна використовувати як для створення односторінкових додатків, так і для більш складних веб-додатків. Важливо, що невеликі інтерактивні елементи можна легко інтегрувати в існуючу інфраструктуру без негативних наслідків.

- Масштабування. Vue може допомогти в розробці досить великих шаблонів багаторазового використання, які можуть бути зроблені майже за той же час, що і більш прості.

- Крихітний розмір. Vue.js важить близько 20 КБ, зберігаючи при цьому свою швидкість і гнучкість, що дозволяє досягти набагато кращої продуктивності в порівнянні з іншими платформами.

Серед недоліків Vue можна виокремити такі:

- Недолік ресурсів. Vue як і раніше займає досить невелику частку ринку в порівнянні з React або Angular, що означає, що обмін знаннями в цьому середовищі все ще перебуває на початковій стадії.

- Ризик надмірної гнучкості. Іноді у Vue можуть виникнути проблеми при інтеграції в величезні проекти, і поки ще немає досвіду можливих рішень, але вони мають з'являться найближчим часом.

До відомих компаній, які використовують Vue, належать: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab та Laracasts, Adobe, Behance, Codeship, Reuters та інші.

До переваг фреймворку React варто віднести такі:

- Легко вивчити, завдяки простому дизайну, використанню JSX для шаблонів і дуже докладної документації. Розробники витрачають більше часу на написання сучасного JavaScript і менше турбуються про код, специфічний для фреймворку.
- Дуже швидкий, завдяки реалізації React Virtual DOM і різним оптимізаціям рендерингу.
- Відмінна підтримка рендерингу на стороні сервера, що робить його потужною платформою для контент-орієнтованих додатків.
- Першокласна підтримка PWA завдяки генератору додатків `create-react-app`.
- Прив'язка даних є односторонньою, що означає менше небажаних побічних ефектів та помилок.
- Наявність Redux – найпопулярнішої платформи для управління станом додатків в React, її легко вчити і використовувати.
- React реалізує концепції функціонального програмування, створюючи простий в тестуванні і багаторазово використовуваний код.
- Додатки можуть бути створені за допомогою TypeScript або Facebook's Flow, що мають вбудовану підтримку JSX.
- Перехід між версіями, як правило, дуже простий: Facebook надає «кодові модулі» для автоматизації більшої частини процесу.
- Навички, отримані в React, можуть бути застосовані для розробки на React Native.

Серед недоліків React можна виокремити такі:

- React не однозначний і залишає розробникам можливість вибирати кращий спосіб розвитку. Це може бути вирішено сильним лідерством проекту і хорошими процесами.
- Спільнота ділиться по способам написання CSS в React, які поділяються на традиційні таблиці стилів (CSS Modules) і CSS-in-JS (тобто Emotion і Styled Components).

- React відходить від компонентів на основі класів, що може стати перешкодою для розробників, яким більш комфортно працювати з ООП.
- Змішування шаблонів з логікою (JSX) може збити з пантелику деяких розробників при перших знайомствах з React.

До відомих компаній, які використовують React, належать: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft, Slack, Storybook і багато інших.

Підводячи підсумки, та зважуючи всі плюси та мінуси, був обраний фреймворк React.js у зв’язку з найбільш відповідними до заданих вимог характеристиками, а також обумовленою наявністю знань щодо даного фреймворку.

2.4 Вибір програмного середовища

Оскільки для побудови системи моніторингу робочого часу працівників була обрана мова JavaScript з використанням фреймворку React, для ефективного виконання завдання необхідно використовувати спеціальне середовище розробки, яке підходить під дані параметри.

Середовище розробки програмного забезпечення – система програмних засобів, яка використовується програмістами для розробки програмного забезпечення. Зазвичай середовище розробки включає в себе текстовий редактор, компілятор і/або інтерпретатор, засоби автоматизації збирання і відлагоджувач (debugger). Іноді також містить засоби для інтеграції з системами управління версіями і різноманітні інструменти для спрощення конструювання графічного інтерфейсу користувача. Багато сучасних середовищ розробки також включають браузер класів, інспектор об'єктів і діаграму ієрархії класів - для використання при об'єктно-орієнтованій розробці програмного забезпечення. Хоча й існують середовища розробки, призначені для декількох мов - такі як

Microsoft Visual Studio, зазвичай середовище розробки призначається для одного певного мови програмування - як наприклад, Visual Basic.

Серед найпопулярніших середовищ розробки, які підтримують розробку на мові JavaScript з використанням фреймворку React, можна виділити два середовища розробки – VS Code та WebStorm.

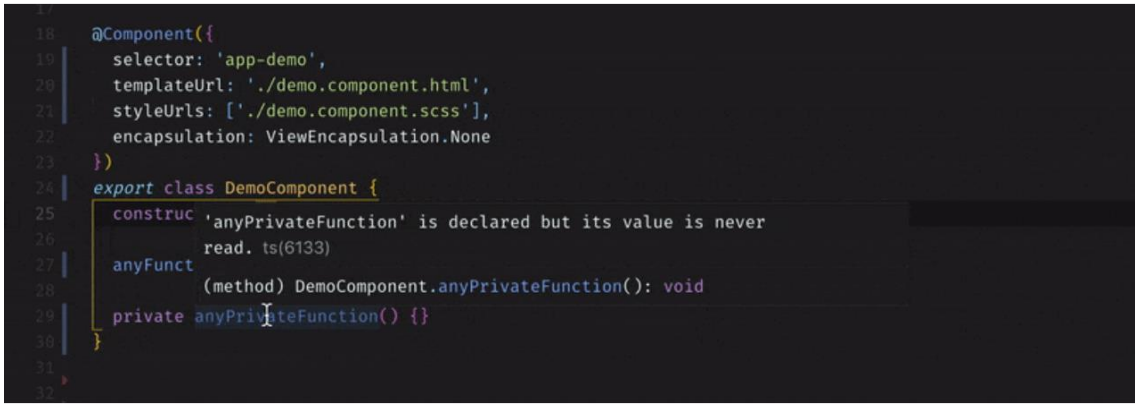


Рис. 2.1 – зовнішній вигляд редактору коду в Visual Studio Code

WebStorm забезпечує надійний, швидкий та гнучкий аналіз статичного коду. Цей аналіз виявляє помилки мови та часу виконання, пропонує виправлення та вдосконалення. Він також індексує весь ваш проект і може, наприклад, виявити всі невикористані методи, змінні та інше. Ви також можете виявити невикористані методи в методах JavaScript, використовуючи VS Code та ESLint. Але якщо ви, наприклад, використовуєте проект TypeScript (наприклад, Angular), VS Code не виявляє невикористані публічні методи. WebStorm також має інтегрований тестовий старт. Таким чином можливо запустити свої тести безпосередньо з IDE і навіть налагодити їх там.

VS Code за замовчуванням має досить просту інтеграцію git. Ви можете або використовувати розширення типу GitLens або використовувати додаткові інструменти, наприклад Sourcetree, якщо ви хочете використовувати графічний інтерфейс для складної роботи з git.

Крім того, VS Code не зберігає локальну історію змін, але може використовувати такі розширення, як Local History.

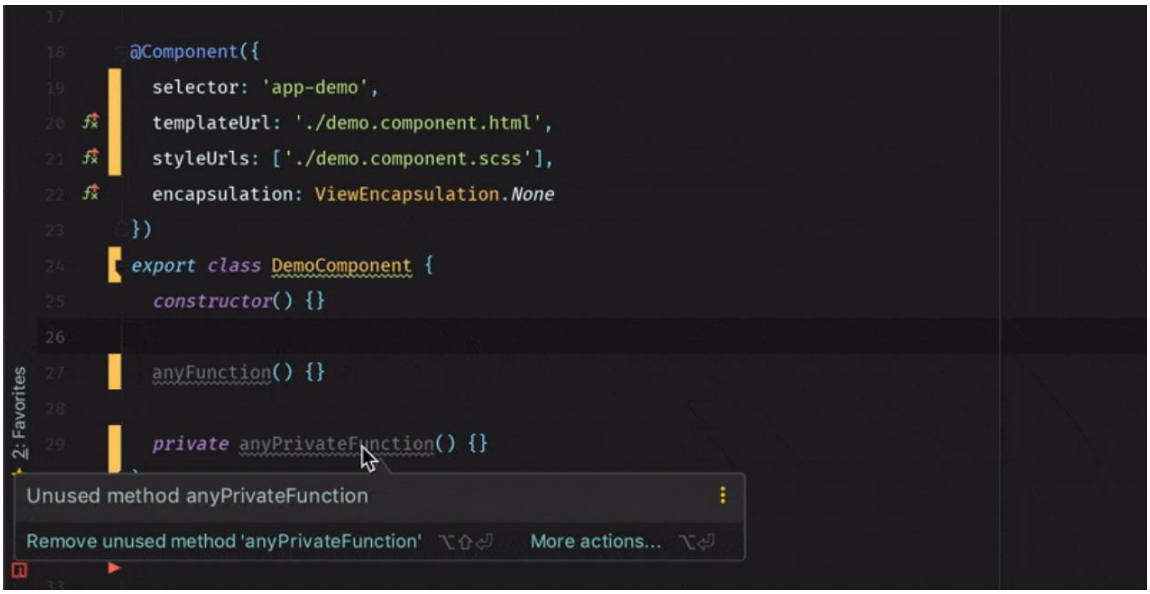


Рис. 2.2 – зовнішній вигляд редактору коду в WebStorm

Головним недоліком WebStorm є необхідність постійної оплати за користування. З роками вартість продукту WebStorm знижується для постійних покупців, але неплатоспроможні користувачі не здатні будуть користуватися даним середовищем розробки, хоча вона і має 30 днів безкоштовного періоду випробування.

З іншого боку, VS Code являє собою безкоштовне середовище розробки, яке може бути доступне для кожного. Відсутність деяких функцій, які наявні у WebStorm, можна компенсувати наявністю великої кількості різноманітних плагінів, які можна встановити в будь-який час, маючи лише доступ до інтернету.

Підводячи підсумки та зважуючи всі плюси та мінуси обох середовищ розробки, для побудови системи моніторингу робочого часу працівників було обрано використовувати середовище розробки програмного забезпечення Visual Studio Code.

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі були проведені дослідження та вибір інформаційної моделі, засобів розробки, мови програмування та програмного середовища розробки для побудови системи моніторингу робочого часу працівників. Було детально оглянуто характеристики кожного з аспектів, визначено переваги та недоліки кожного з них та вибрано найбільш підходящі для побудови системи моніторингу робочого часу працівників у веб-додатку.

Серед баз даних була обрана MongoDB через її спеціальне призначення для зберігання ієрархічних структур даних. Також MongoDB відображає об’єкти у коді програми, що полегшує роботу з даними, а спеціальні запити, індексація та агрегація в режимі реального часу забезпечують потужні способи доступу та аналізу даних.

Серед засобів розробки веб-додатку була обрана структура баз даних у вигляді SPA. Це обумовлено великої кількістю переваг, порівняно з MPA, серед яких в першу чергу було взято масштабованість розроблюваного веб-додатку, який необхідно помістити в одну сторінку. Це покращить сприйняття інтерфейсу, призначеного для користувача, що впливає позитивно на ефективність працівників.

Серед фреймворків на базі JavaScript був обраний фреймворк ReactJS. В першу чергу, це обумовлено наявністю знань, що я має по відношенню до даного фреймворку. Крім того, React.js дуже швидкий завдяки реалізації React Virtual Dom і різним оптимізаціям рендерингу.

Серед програмних середовищ розробки був обраний Visual Studio Code. Це обумовлено наявністю досвіду в використані даного програмного середовища розробки. Крім того, Visual Studio Code безкоштовний у використані, порівняно з іншими аналогами.

РОЗДІЛ 3.
ПОБУДОВА ПРОГРАМНОГО ПРОДУКТУ

3.1 Інформаційна модель

Як вже було зазначено вище, в якості серверної системи управління базами даних була обрана MongoDB. Структура отриманої моделі даних представлена на рисунку 3.1.

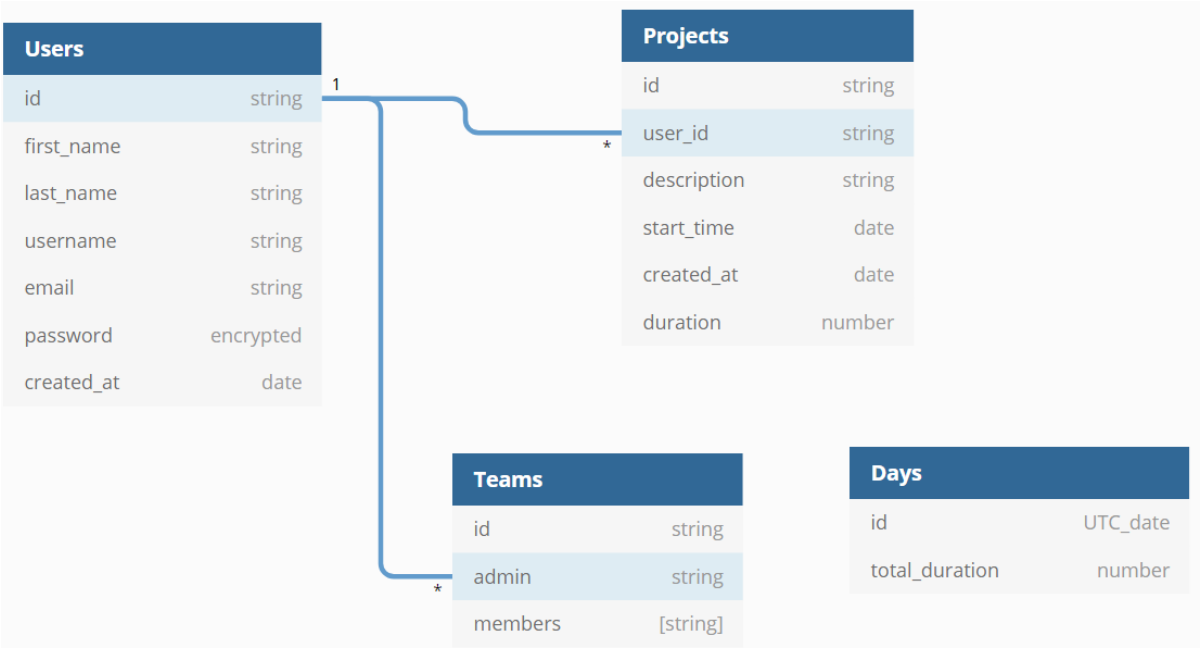


Рис. 3.1 – Модель даних

3.1.1 Користувач

Дана сутність включає в себе наступні поля:

- username – коротке ім’я користувача;
- email – адреса електронної пошти користувача для реєстрації та логіну у веб-додаток;
- password – пароль користувача, який хешується при збереженні в базу даних;
- createdAt – дата створення користувача у форматі ISO 8601.

Перерахованих вище даних достатньо, щоб організувати процес реєстрації і аутентифікації користувача. Крім того, вищевказані дані

знаходяться у вкладці «Profile» користувача та мають можливість бути редагованими.

3.1.2 Проект

Дана сутність включає в себе наступні поля:

- userid – використовується для визначення відповідального за виконаний проект;
- description – залишений працівником, короткий опис виконуваного проекту;
- createdAt – дата створення проекту;
- startTime – час початку виконання проекту;
- duration – загальний час, який було витрачено на роботу над проектом.

За допомогою userid визначається приналежність виконаного проекту до працівника, який його виконував. За допомогою createdAt зберігається дата початку виконання проекту. startTime позначає дату та час початку роботи над проектами. duration визначає загальну кількість часу, який було витрачено на проект, та обраховується відніманням startTime від окремої змінної endTime.

3.1.3 День

Дана сутність включає в себе наступні поля:

- date – використовує міжнародний стандарт ISO 8601 для відображення дати й часу;
- totalDuration – загальний час, який було витрачено на виконання роботи на протязі усього дня.

Дана таблиця уміщує в себе значення, які не потребують окремого відображення, але використовуються в інших розділах, зокрема «Project».

3.1.4 Команда

Дана сутність включає в себе наступні поля:

- admin – користувач, учасник команди з особливими правами, який може бути тільки один у команді;
- members – користувачі, які знаходяться у складі команді, створеної іншим користувачем.

Користувач, який створює команду, автоматично набуває особливі права – admin. Користувач з правами admin може додавати або видаляти з команди інших членів – members. Крім того, admin може передати свої права будь-якому члену команди. Передаючи права, admin втрачає свої права і стає звичайним членом команди (members), а той, кому передали права, в свою чергу стає admin. Кожен учасник команди має свій userid, за допомогою якого можна подивитись на розділ «Project» кожного з учасників команди.

3.2 Алгоритмізація задач

В даному підрозділі представлені сценарії роботи програми у вигляді алгоритмів, які, будучи перенесені в код, є вирішенням в поставлених в п.1.3 задач.

Алгоритм підготовки додатку до роботи представлений на рисунках 3.2-3.4.

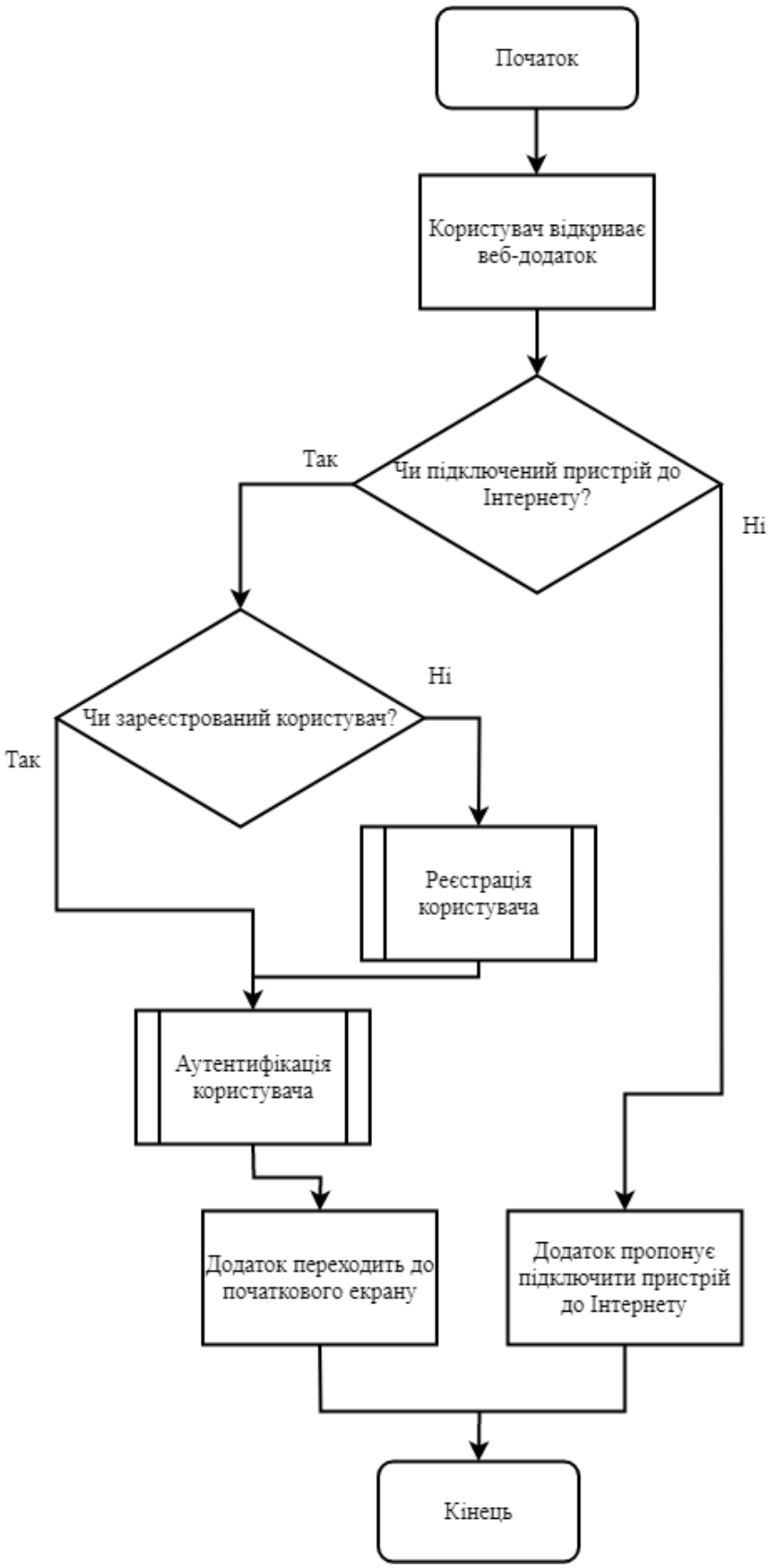


Рис. 3.2 – Алгоритм підготовки програми до роботи (основний процес)

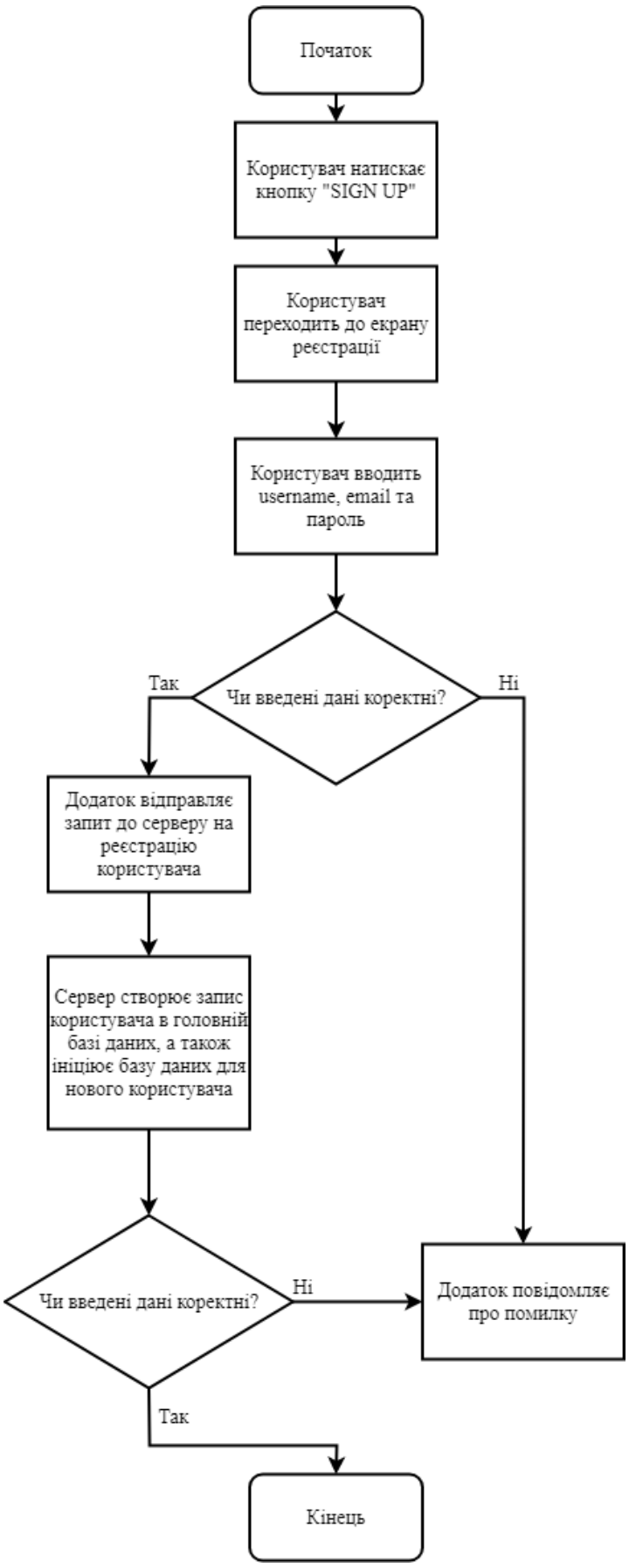


Рис. 3.3 – Процес реєстрації

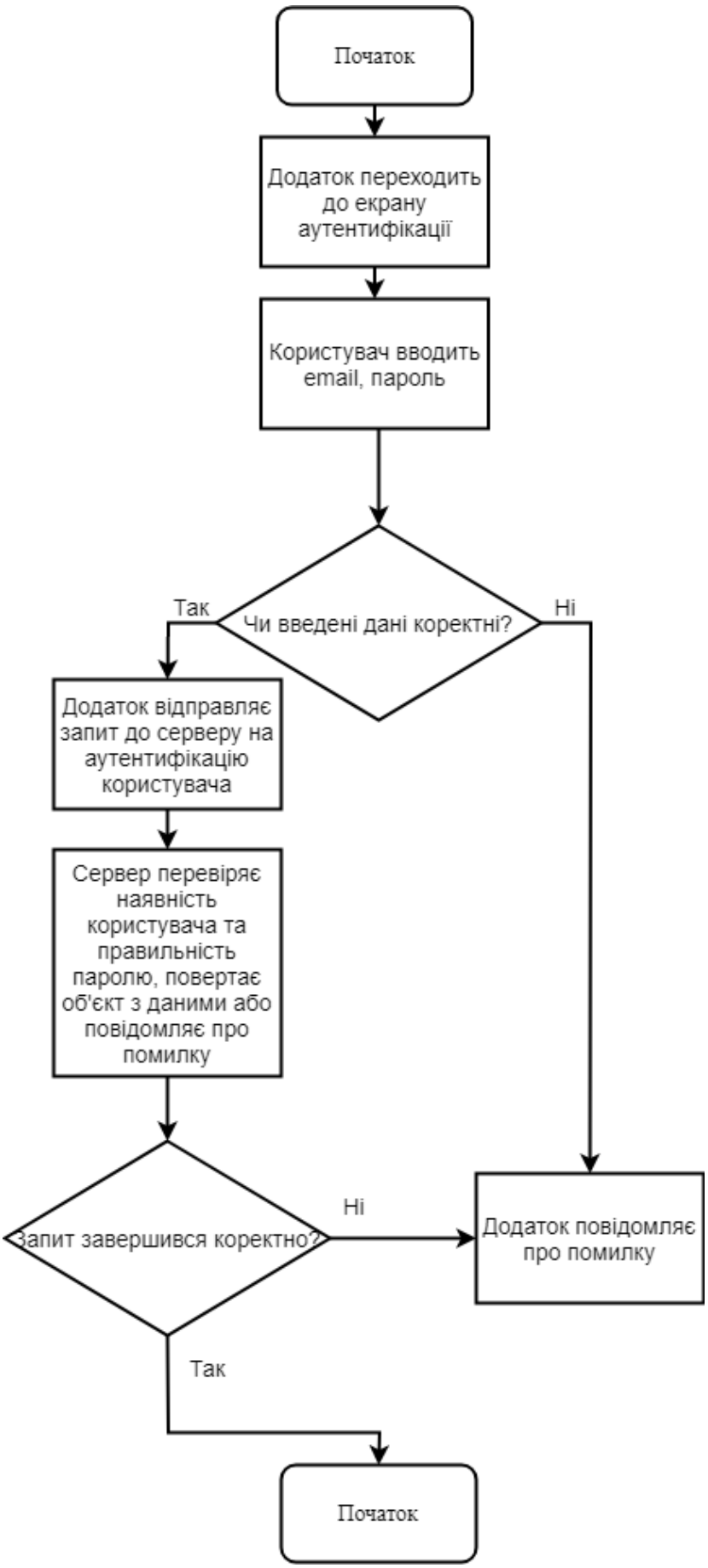


Рис. 3.4 – Процес аутентифікації

3.3 Програмна реалізація

3.3.1 Архітектура програми

Додаток складається з декількох загальних модулів, що забезпечують роботу інших, більш спеціалізованих:

– store.js (сховище) - сховище стану програми, забезпечує атомарність і відстеження змін стану;
– routes.js (маршрути) - модуль, який відповідає за переходи між різними екранами додатку;
– app.js - головний модуль, точка входу в додаток, відповідає за ініціалізацію всіх інших модулів і бібліотек при запуску.

Після завантаження програми ініціалізується app.js і store.js, після ініціалізації стану програми управління передається в код модуля routes.js.

Роутер на підставі стану програми може передавати управління одному з наступних модулів:

– Login.js - екран входу та реєстрації в додаток;
– Dashboard.js – екран статистичних даних користувача;
– Profile.js – екран, що відображає особисту інформацію користувача;
– Projects.js - екран списку виконаних (або у стані виконання) проектів;
– TimeTracker.js – екран, що відповідає за початок роботи додатку;
– Teams.js - екран команд користувача.

Кожен з цих модулів містить в собі опис користувальницького інтерфейсу, а також логіки роботи програми.

Для роботи з внутрішнім станом клієнтської частини мі використовуємо бібліотеку Redux. Ця бібліотека імплементує flux architecture. В основі такої структури є store, який зберігає дані та в залежності від actions змінює внутрішній стан. Елементи нашого додатку, що підписуються на store, отримують дані, які змінюються автоматично зі зміною внутрішнього стану. Щоб спростити роботу з внутрішнім станом, кореневий reducer розбили на кілька редюсерів:

- auth – зберігає дані користувача і має наступну форму:

```
const initialState = {
  user: null,
  isLoggedIn: false,
  isLoading: false,
  isChecking: true,
};
```

Рис. 3.5 – auth reducer

- projects – зберігає дані про проект і має наступну форму:

```
const initialState = {
  list: [],
  selected: '',
  isLoading: false,
};
```

Рис. 3.6 – projects reducer

Потім ці редюсери комбінуються в один кореневий reducer за допомогою функції combineReducers і присвоюються відповідні назви кожного з редюсерів. Аби редюсери змінювали внутрішній стан, потрібно відправляти action до store. В залежності від action.type, ми обробляємо і повертаємо певну частину внутрішнього стану. Для прикладу, розглянемо login action. Оскільки Redux бібліотека синхронна, для виконання асинхронних запитів потрібно додати Redux thunk middleware. Він дає змогу повернути dispatch, який можна використати для виклику інших actions. Спочатку ми створили Redux actions loginStart, loginSuccess, loginError. При dispatch login action з компоненту, ми викликаємо вказані actions в потрібному порядку. loginStart помічає флаг isLoading у редюсері, відправляємо запит до сервера по user/login endpoint. Якщо сервер повернув помилку, loginError action встановить isLoading: false і присвоїть Error поле у внутрішній стан. Якщо запит успішний, ми

зберігаємо token у локальному сховищі браузера та викликаємо loginSuccess user, що зберігає користувача у внутрішній стан auth редюсера.

```
      const loginStart = () => ({ type: AUTH.LOGIN_START });
      const loginSuccess = (user) => ({ type: AUTH.LOGIN_SUCCESS, payload: {
user } });
      const loginError = (error) => ({ type: AUTH.LOGIN_ERROR, payload: { err
or } });

      const login = (email, password, history) => async (dispatch) => {
        dispatch(loginStart());
        try {
          const { result, error } = await callApi('user/login', { email, pass
word });

          if (error) {
            dispatch(loginError(error));
          } else {
            const { user, token } = result;
            localStorage.setItem('token', token);
            dispatch(loginSuccess(user));

            history.push('/dashboard');
          }
        } catch (err) {
          console.error('error', err.message);
          dispatch(loginError());
        }
      };
```

```
const handleSubmit = (e) => {
  e.preventDefault();
  if (isLogin) {
    dispatch(login(email, password, history));
  } else {
    dispatch(signup({
      username,
      email,
      password,
    }, history));
  }
};
```

Рис. 3.7. – Приклад виклику actions у компоненті

```
export default (state = initialState, action) => {
  switch (action.type) {
    case AUTH.LOGIN_START:
    case AUTH.SIGNUP_START:
    case AUTH.LOGOUT_START:
      return { ...state, isLoading: true };

    case AUTH.LOGIN_ERROR:
    case AUTH.SIGNUP_ERROR:
    case AUTH.LOGOUT_ERROR:
      return {
        ...state,
        error: action.payload.error,
        isLoading: false,
      };

    case AUTH.SIGNUP_SUCCESS:
    case AUTH.LOGIN_SUCCESS:
      return {
        ...state,
        user: action.payload.user,
        isLoggedIn: true,
        isLoading: false,
      };
```

Рис. 3.8 – Приклад обробки внутрішнього стану у auth редюсері

3.3.2 Архітектура серверної частини

При написанні серверу використовуємо Express фреймворк, слухаємо на порту 4000 та передаємо декілька middleware функцій(проміжні функції між запитами), одна з яких відповідає за взаємодію з клієнтом в не залежності від домену (задаючи заголовки відповідей Access-Control-Allow-Origin) та bodyParser, шо повертає відповіді у JSON-форматі. В якості аутентифікації запитів використовується middleware функція, яка перевіряє token кожного запиту та, у випадку успіху, передає керування функції з відповідного endpoint.

У сервері імплементовано кілька основних маршрутів на відповідні API endpoint (кінцеві точки):

- userRoute – route (маршрут), який відповідає за логін та реєстрацію користувача;
- projectRoute – відповідає за операції над проектами.

userRoute обробляє три кінцеві точки (endpoint).

```
router.post(
  '/signup',
  [
    check('username', 'Please enter a valid username')
      .not()
      .isEmpty(),
    check('email', 'Please enter a valid email').isEmail(),
    check('password', 'Please enter a valid password').isLength({ min: 6 }),
  ],
  handleUserSignup,
);
```

Рис. 3.9 – /signup endpoint

В залежності від результату валідації, функція handleUserSignup повертає помилку зі статусом 400 або дістає дані з тіла запиту. Відбувається пошук користувача з переданою електронною поштою і, якщо такий користувач вже є в базі, повертаємо помилку: «User Already Exists». Якщо пройшли попередні перевірки, то створюємо об’єкт з отриманими даними, хешуємо пароль за допомогою бібліотеки Bcrypt, та зберігаємо користувача у базі даних. Генеруємо authorization token з id отриманого користувача та відправляємо клієнту разом з об’єктом користувача.

```
const handleUserSignup = async (req, res) => {
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(400).json({
      error: normalizeValidationResult(errors.array()),
    });
  }

  const { username, email, password } = req.body;

  try {
    let user = await User.findOne({ email });
    if (user) {
      return res.status(400).json({ error: 'User Already Exists' });
    }

    user = new User({ username, email, password });
```

```
    const salt = await bcrypt.genSalt();
    user.password = await bcrypt.hash(password, salt);

    await user.save();

    const token = generateToken({ id: user.id });

    res.status(200).json({
      user,
      token,
    });
  } catch (err) {
    console.error(err.message);
    res.status(500).json({ error: 'Error in Saving' });
  }
};
```

Функція handleUserLogin спочатку валідує email та пароль. Якщо валідація не пройшла, повертає помилку зі статусом 400. Інакше, ці дані дістаються з тіла запиту та шукаємо користувача з такою електронною адресою. Якщо користувача не знайдено в базі даних, клієнту повертається помилка: «User Not Exists». Коли користувача знайдено, йде перевірка пароля: в Bcrypt бібліотеці викликаємо функцію compare, куди передаємо захешований пароль з бази даних та отриманий. Якщо паролі не співпадають, повертаємо помилку: «Incorrect Password». Так само, як і в попередній раз, генеруємо authorization token з id отриманого користувача та відправляємо клієнту разом з об’єктом користувача.

```
router.post(
  '/login',
  [
    check('email', 'Please enter a valid email').isEmail(),
    check('password', 'Please enter a valid password').isLength({ min: 6 }),
  ],
  handleUserLogin,
);
```

Рис. 3.10 – /login endpoint

```
const handleUserLogin = async (req, res) => {
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(400).json({
      error: normalizeValidationResult(errors.array()),
    });
```

```
  }

  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(400).json({
        error: 'User Not Exist',
      });
    }

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.status(400).json({
        error: 'Incorrect Password !',
      });
    }

    const token = generateToken({ id: user.id });

    res.status(200).json({
      user,
      token,
    });
  } catch (e) {
    console.error(e);
    res.status(500).json({
      error: 'Server Error',
    });
  }
};
```

```
router.get('/me', auth, async (req, res) => {
  try {
    // request.user is getting fetched from Middleware after token authentication
    const user = await User.findById(req.id);
    res.json(user);
  } catch (e) {
    res.send({ message: 'Error in Fetching user' });
  }
});
```

Рис. 3.11 – /me endpoint

Перевірка запитів на захищені ресурси відбувається через endpoint /me, за допомогою middleware функції auth. З header запиту дістається поле authorization. Якщо такого немає: «Auth Error». В іншому випадку, дістаємо token та перевіряємо на валідність за допомогою jsonbwebtoken бібліотеки. Якщо валідація не пройшла, сервер видає помилку: «Invalid Error». Якщо валідація успішна, з token декодується id користувача та

передається у наступну функцію. По отриманому id шукається користувач в базі даних, та відправляється клієнту в JSON-форматі.

```
const jwt = require('jsonwebtoken');
const { JWT_SECRET } = require('../config/config');

module.exports = function (req, res, next) {
  const { authorization } = req.headers;
  if (!authorization) return res.status(401).json({ error: 'Auth Error' });

  try {
    const token = authorization.replace('Bearer ', '');
    const decoded = jwt.verify(token, JWT_SECRET);
    req.id = decoded.id;
    next();
  } catch (e) {
    console.error(e);
    res.status(500).send({ error: 'Invalid Token' });
  }
};
```

projectRoute також обробляє три кінцеві точки (endpoint).

```
const express = require('express');
const {
  handleAddProject,
  handleGetProject,
  handleGetProjectsByUser,
} = require('./handlers');
const auth = require('../../middleware/auth');

const router = express.Router();

router.post('/', auth, handleAddProject);
router.get('/:projectId', auth, handleGetProject);
router.get('/', auth, handleGetProjectsByUser);

module.exports = router;
```

Рис. 3.12 – кінцеві точки, які оброблює projectRoute

Якщо прийшов POST запит, перевіривши функцією auth валідність авторизації, викликаємо функцію handleAddProject. Дістаємо description з тіла запиту та id користувача з самого запиту. Шукаємо проект для користувача з цим id. Якщо проект не знайдено – створюємо новий. Зберігаємо проект в базі даних та повертаємо клієнту.

```
const handleAddProject = async (req, res) => {
  const { description } = req.body;
  console.log(description)
  const { id: userId } = req;

  try {
    let project = await Project.findOne({ userId });
    if (!project) {
      project = new Project({ description, userId });
    }

    await project.save();

    res.status(200).json({
      project,
    });
  } catch (err) {
    console.error(err.message);
    res.status(500).json({ error: 'Error creating project' });
  }
};
```

Якщо прийшов get запит з параметрами, викликаємо функцію handleGetProject. Якщо в параметрах не знайшли поля projectId, повертаємо помилку зі статусом 400. Інакше, в базі даних шукаємо проект з id рівнем projectId. Знайшовши проект, відправляємо користувачу.

```
const handleGetProject = async (req, res) => {
  try {
    if (!req.params.projectId) {
      res.status(400).json({ error: 'Bad request, no params' });
    }
    const project = await Project.findOne({ _id: req.params.projectId });
    if (!project) {
      res.status(400).json({ error: 'No project found' });
    }

    res.status(200).json({
      project,
    });
  } catch (err) {
    console.error(err.message);
    res.status(500).json({ error: 'Error creating project' });
  }
};
```

Якщо прийшов get запит без параметрів, перевіривши функцією auth валідність авторизації, викликаємо handleGetProgectsByUser. Якщо запит не містить id користувача, повертаємо помилку зі статусом 400. Шукаємо в базі всі проекти з отриманим id користувача і повертаємо їх.

```
const handleGetProjectsByUser = async (req, res) => {
  try {
    if (!req.id) {
      res.status(400).json({ error: 'Bad request, no user id' });
    }
    const projects = await Project.find({ userId: req.id });
    if (!projects) {
      res.status(400).json({ error: 'No projects for this user' });
    }

    res.status(200).json({
      projects,
    });
  } catch (err) {
    console.error(err.message);
    res.status(500).json({ error: 'Error creating project' });
  }
};
```

Кінцева точка (endpoint) - це один кінець каналу зв'язку. Коли API взаємодіє з іншою системою, точки дотику цього повідомлення вважаються кінцевими точками. Для API, кінцева точка містить URL-адресу сервера. Кожна кінцева точка - це місце, з якого API можуть отримати доступ до ресурсів, необхідних для виконання своєї функції.

API працюють з використанням «запитів» та «відповідей». Коли API вимагає інформацію від веб-програми чи веб-сервера, він отримає відповідь. Місце, в якому API надсилають запити, і де «живе» ресурс, називається кінцевою точкою.

Кожна функція складається з шаблонного модуля, що здійснює рутинні операції по прийому HTTP запиту та відправки відповіді.

Всі дані зберігаються у хмарному сховищі MongoDB Atlas на кластері. За допомогою MongoDB Atlas створюється кластер MongoDB для будь-якого основного постачальника хмарних обчислень за нашим вибором. Після чого ми можемо почати використовувати цей кластер за лічені хвилини. Використовуючи призначений для користувача інтерфейс на основі браузера Atlas, також можливо інтуїтивно налаштувати кластер і контролювати його продуктивність.

Зв’язок з MongoDB, що зберігає дані як документи, відбувається за допомогою Mongoose – бібліотеки JavaScript. Всі документи мають JSON-

формат. Однією з головних особливостей MongoDB є гнучкість її структури даних. Властивості об’єкта не є обов’язковими для інших об’єктів колекції. Саме це відрізняє MongoDB від баз даних SQL (* Structured Query Language - мова структурованих запитів), наприклад, MySQL чи Microsoft SQL Server, в яких для кожного об'єкту, що зберігається в базі даних, необхідна фіксована схема. Таким чином, ми визначаємо об'єкти за строго-типізованою схемою, що відповідає документу MongoDB. Створили моделі «Day», «Project», «Team», «User».

ВИСНОВКИ ДО РОЗДІЛУ 3

У даному розділі була проведена розробка програмного продукту для виконання завдання по створенню системи моніторингу робочого часу працівників.

Весь процес створення системи, а саме, написання програмного коду, було виконано, використовуючи програмне середовище розробки Visual Studio Code, яке було обрано заздалегідь у попередньому розділі через зазначені вище переваги. Серверна частина додатку написана на Express фреймворку – середовища Node.js. Серверна частина виконує операції з даними за допомогою MongoDB, а потім повертає клієнту. На клієнті реалізований внутрішній стан, що делегується бібліотекою Redux. Створені компоненти підписуються на дані внутрішнього стану (state) та відображають інформацію динамічно. Крім того, за допомогою фреймворку React.js, який було обрано у попередньому розділі, з використанням бібліотеки Material-UI, було створено багато різних компонентів для взаємодії користувача із веб-додатком. Було створено повністю функціонуючий веб-додаток, серед можливостей якого є такі:

- Можливість входу та реєстрації користувача;
- Можливість редагування особистої інформації користувача, а також функція зміни паролю;
- Можливість створювати, редагувати та обирати потрібний для роботи проект, а також можливість відстежувати кількість витраченого часу на кожний з проектів;
- Можливість визначати ефективність працівників, за допомогою відповідних графіків;
- Можливість створювати свою та вступати у команди інших користувачів для подальшої роботи.

Додаток створено таким чином, що його функціонал можна доповнювати або розширювати за необхідності в будь-який момент часу.

РОЗДІЛ 4.

ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

4.1 Опис початкового екрану додатку

На рисунку 4.1 представлена форма входу в додаток. Дана форма містить два поля введення, які беруть відповідно адресу електронної пошти та пароль користувача. Кнопка «SIGN UP» дозволяє перейти до екрану реєстрації нового користувача. Кнопка «LOG IN» запускає процес аутентифікації після введення валідних значень у відповідні поля.

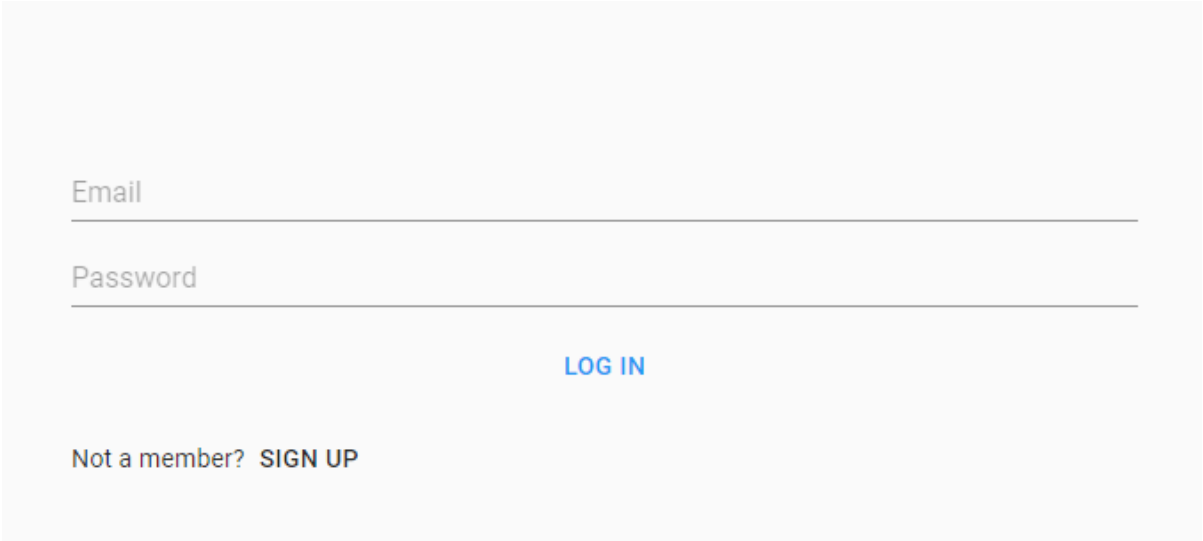


Рис. 4.1 – Екран входу LOG IN

На малюнку 4.2 представлена форма реєстрації нового користувача. Дана форма містить три поля введення, відповідно ім’я (Username), адреса електронної пошти (Email) та пароль (Password) користувача. Кнопка «SIGN UP» запускає процес реєстрації після введення валідних значень. Кнопка «LOG IN» дозволяє повернутися до екрану входу в додаток.

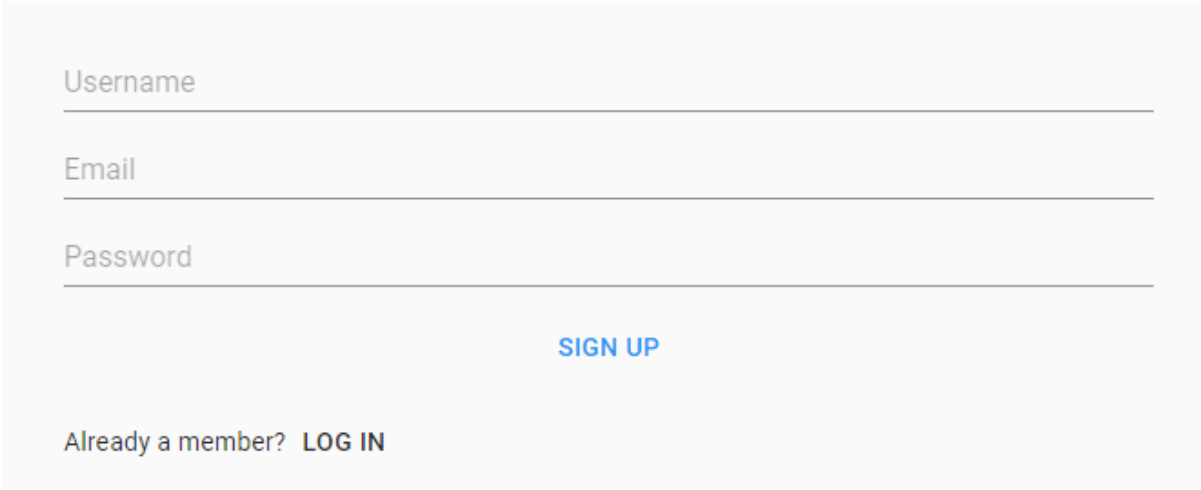


Рис. 4.2 – Екран реєстрації SIGN UP

На рисунку 4.3 представлений екран завантаження. Даний екран не містить активних елементів і призначений для відображення прогресу завантаження даних з Redux Store.

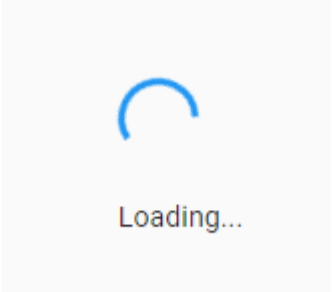


Рис. 4.3 – Екран завантаження LOADING

4.2 Опис екрану Dashboard та панелі меню

На рисунку 4.4 представлений початковий екран додатку «Dashboard» після входу. Дана форма містить кілька основних структурних елементів:

- панель програми – містить кнопку меню і кнопку виходу «LOG OUT» із облікового запису користувача;
- панель вибору дати – містить дві кнопки «Start Date» і «End Date», кожна з яких відповідно починає процес вибору початкової дати та кінцевої дати часового проміжку;

- панель графіку – містить графік, що відображає кількість напрацьованих годин у часовому проміжку, який вибрали за допомогою кнопок «Start Date» і «End Date».

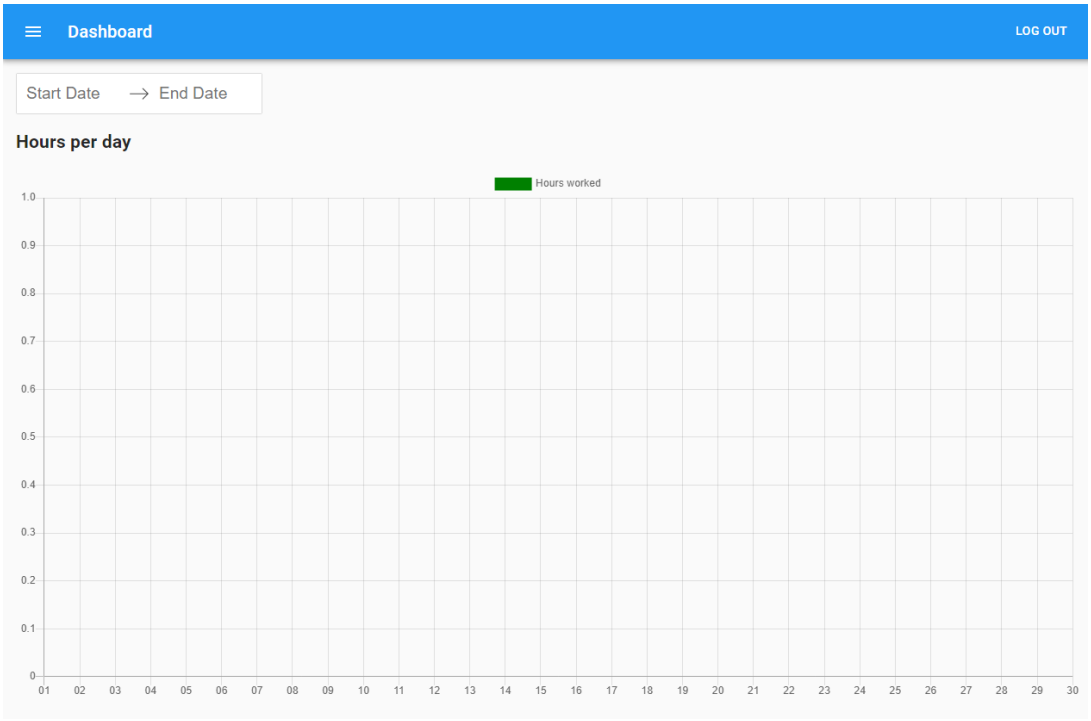


Рис. 4.4 – Екран Dashboard

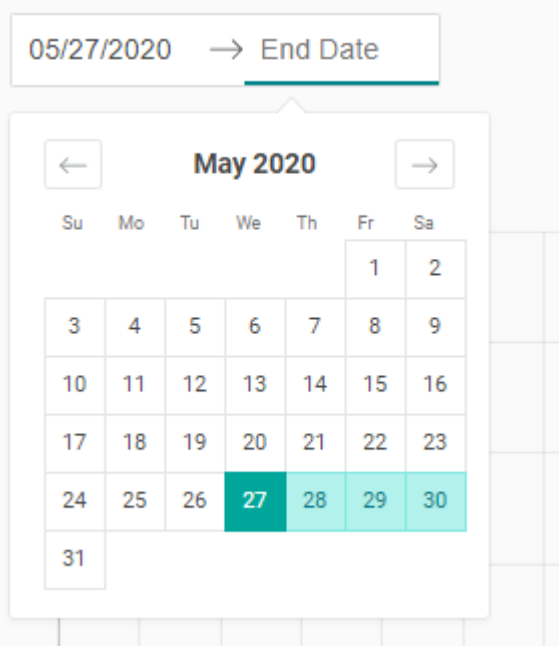


Рис. 4.5 – Вибір періоду часу за допомогою кнопок Start Date і End Date

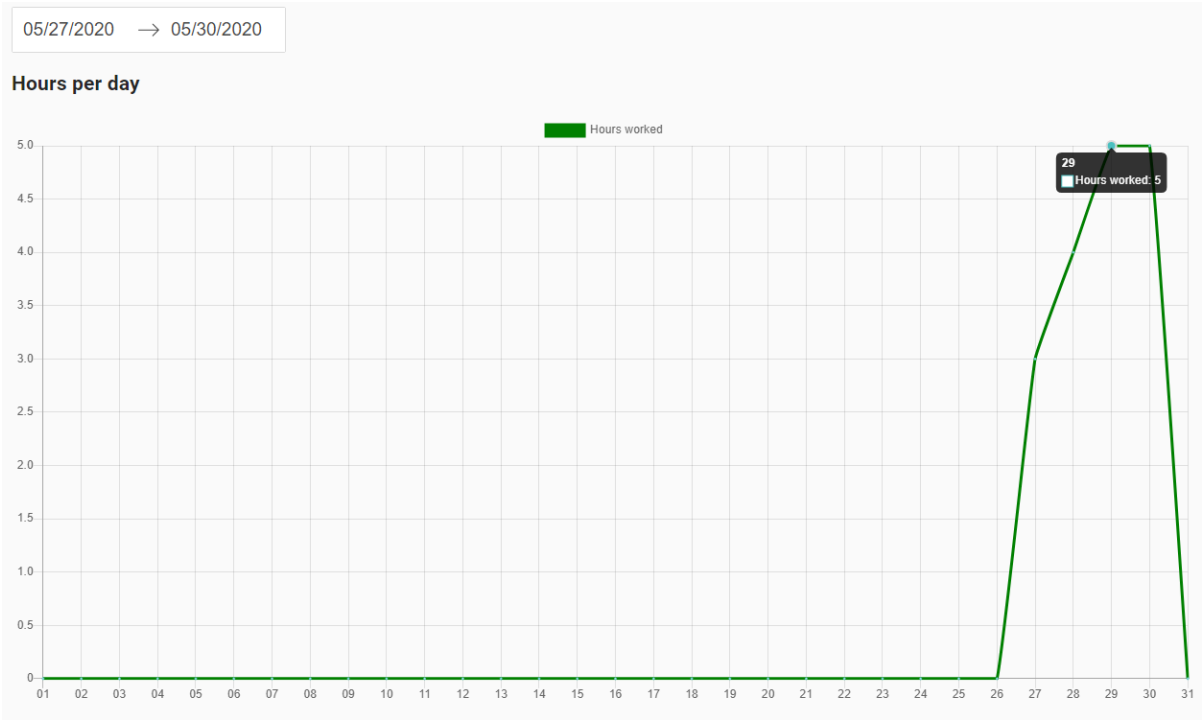


Рис. 4.6 – Графік напрацьованих годин відповідно до вибраного періоду часу

На рисунку 4.7 представлена панель меню. Дана форма містить посилання, що дозволяють перейти до інших категорій додатку, серед яких є «Profile», який у меню відображає поточні ім’я користувача та електронну адресу, «Time Tracker», «Dashboard», «Projects» і «Teams».

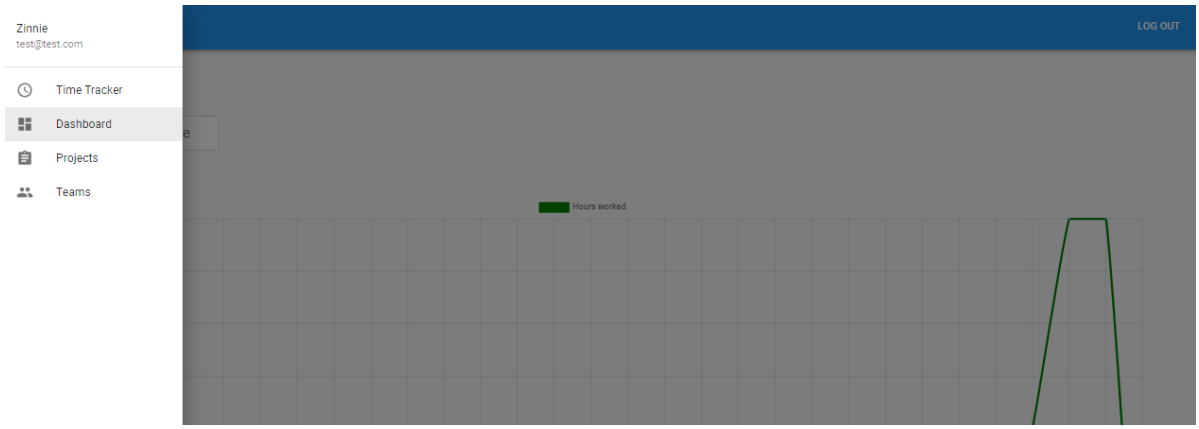


Рис. 4.7 – Панель меню

4.3 Опис екрану Profile

На рисунку 4.8 представлений екран «Profile», який містить персональні дані користувача. Дана форма дозволяє також змінювати інформацію та пароль по бажанню користувача. Серед наявних полів присутні такі:

- First Name – ім’я користувача;
- Last Name – прізвище користувача;
- Username – ідентифікаційне ім’я користувача;
- Email – електронна адреса користувача, яка використовується для запрошення у команди;
- New Password – поле для вводу нового паролю користувача з метою змінити поточний пароль;
- Confirm Password – поле, що відповідає за підтвердження зміни паролю і не дозволяє змінити пароль, якщо відмінне від поля New Password;
- SAVE CHANGES – кнопка, за допомогою якої відбувається підтвердження всіх введених змін, та відправка нових даних у Redux Store.

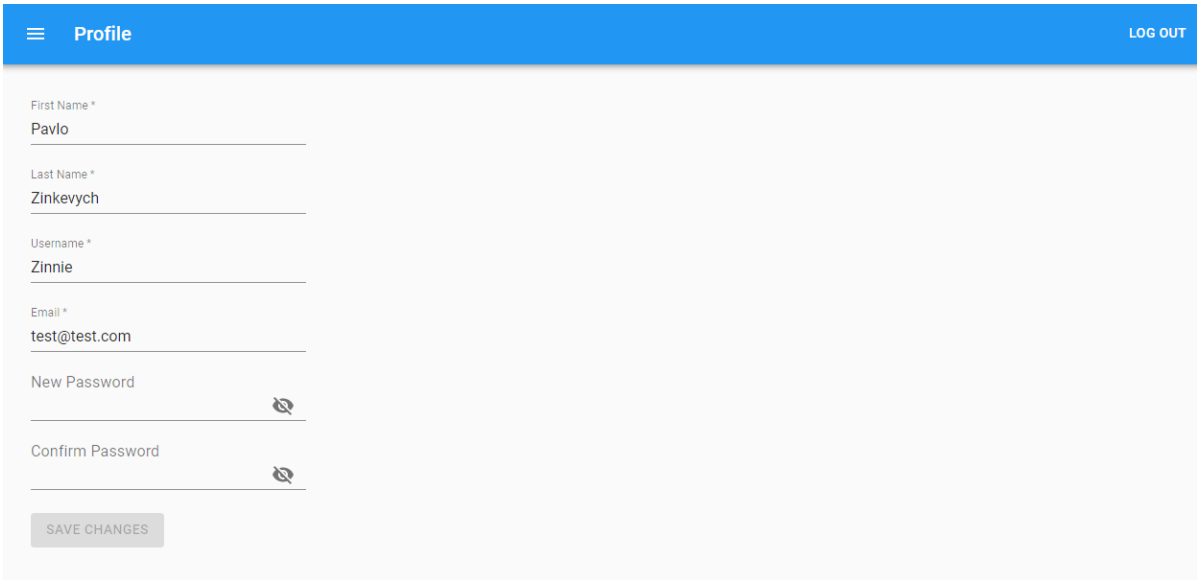


Рис. 4.8 – Екран профілю користувача Profile

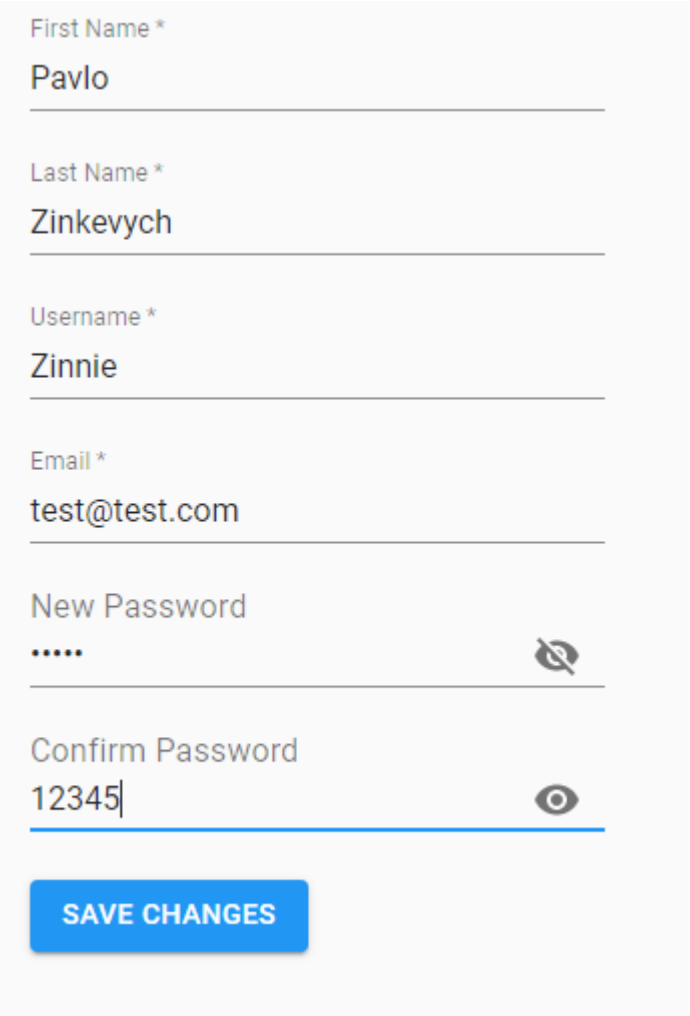


Рис. 4.9 – Процес зміни паролю користувача

4.4 Опис екрану Time Tracker

На рисунку 4.10 представлено екран запуску моніторингу робочого часу працівників «Time Tracker». Дана форма містить такі структурні елементи:

- Total – відображає загальну кількість часу, який було проведено за роботою у поточний день;
- ADD PROJECT – кнопка, що дозволяє додавати нові проекти, над якими працює користувач, за допомогою текстового поля «What are you working on?». Якщо вже існують інші проекти, або поле «What are you working on?» пусте, то замість кнопки ADD PROJECT буде відображатися кнопка, за допомогою якої можна вибрати інші, вже існуючі проекти. Кожен зі створених проектів з'являється у розділі

«Projects» з можливістю огляду кількості часу, який було затрачено на кожний проект;

- START – кнопка, що відповідає за початок роботи й починає відлік часу. Після натискання на цю кнопку, лічильник зліва починає відраховувати час роботи, а сама кнопка START змінюється на кнопку STOP, яка відповідає за зупинку відліку часу.

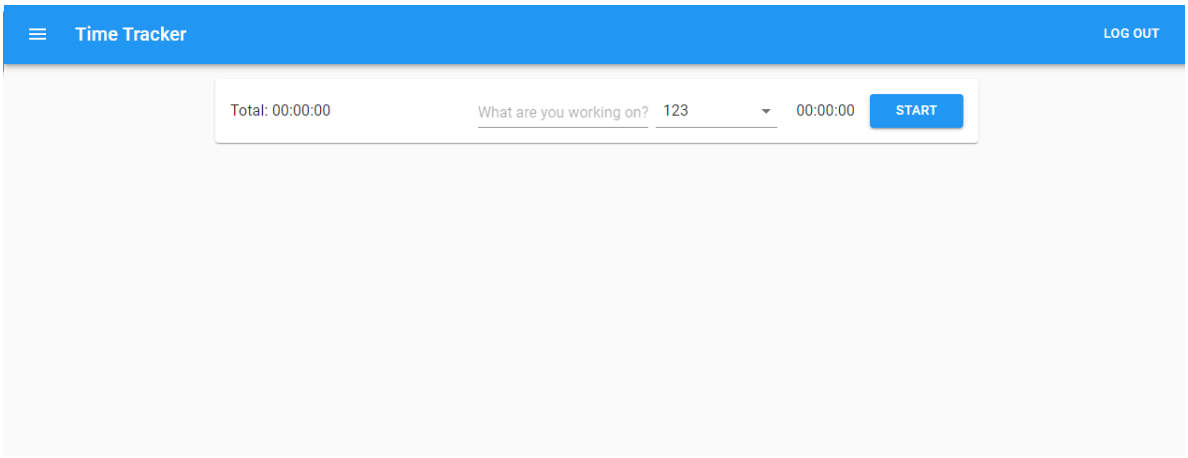


Рис. 4.10 – Екран відліку часу Time Tracker

Рис. 4.11 – Створення нового проекту за допомогою кнопки ADD PROJECT

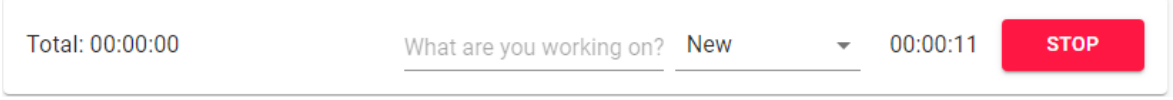


Рис. 4.12 – Початок відліку часу роботи за допомогою кнопки START

4.5 Опис екрану Projects

На рисунку 4.13 представлено екран «Projects», який відображає усі проекти, які створив користувач. Усі створені проекти мають вигляд форм, які містять назву проекту, дату створення, а також час, який було витрачено на роботу над даним проектом. Присутня функція створення нових проектів, а також функція пошуку проекту за його назвою. Крім того, є можливість змінити ім'я проекту, або зовсім видалити його.

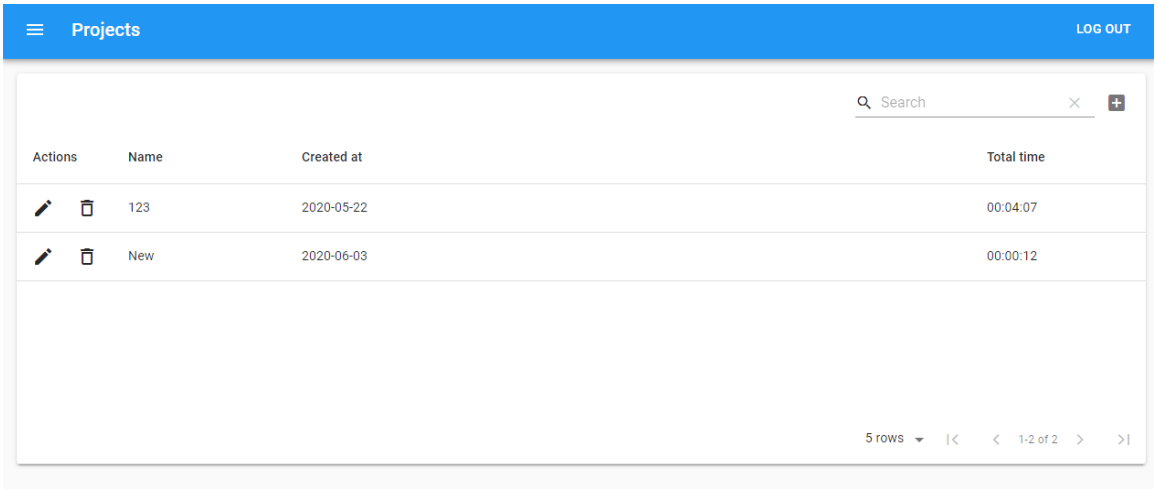


Рис. 4.13 – Екран керування проектами Projects

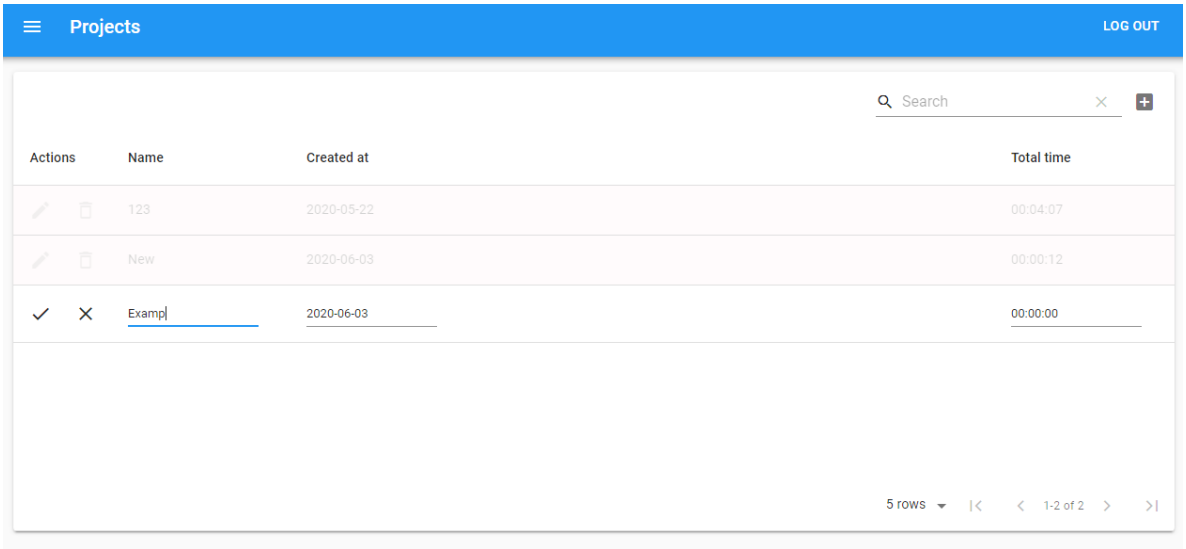


Рис. 4.14 – Процес створення нового проекту

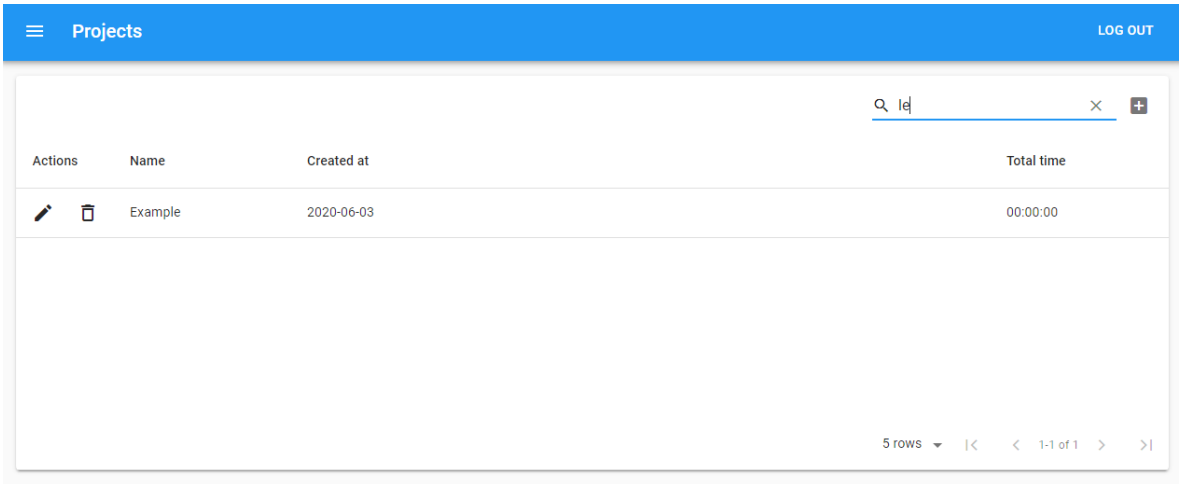


Рис. 4.15 – Процес пошуку за назвою проекту

4.6 Опис екрану Teams

На рисунку 4.16 представлено екран «Teams», який відображає поточну команду користувача. Якщо користувач не є членом іншої команди, то він автоматично є членом своєї команди, розмір якої 1 користувач, а також він набуває права «Admin». Якщо користувач вступає у команду до іншого користувача, то він набуває права «Member». Дана форма містить такі структурні елементи:

- Search bar – виконує функцію пошуку користувача зі списку членів команди за допомогою Username або Email;
- ADD MEMBER – кнопка, яка відповідає за відправку запрошення на вступ до команди іншим користувачам. Відправка здійснюється за допомогою Email.

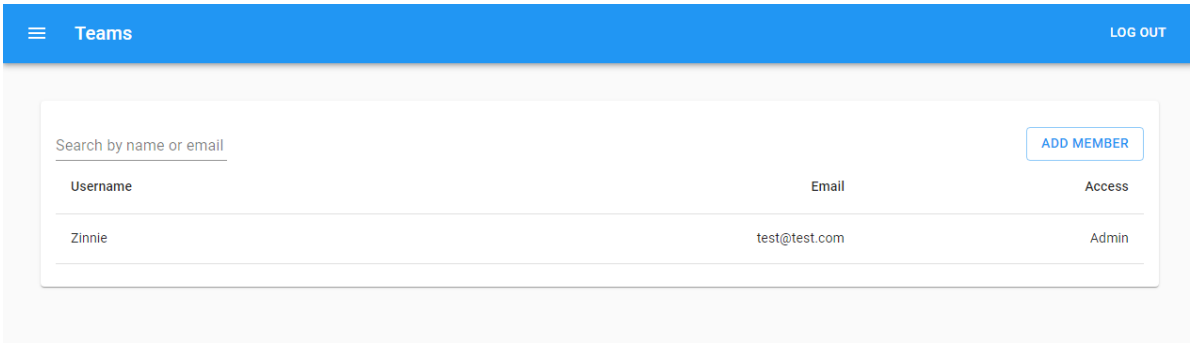


Рис. 4.16 – Екран команди користувача Teams

На рисунку 4.17 представлено повідомлення, яке приходить до користувача, після того як інший користувач відправив запрошення на вступ до своєї команди. Користувач може прийняти запрошення, ставши членом команди іншого користувача, або відхилити запрошення.

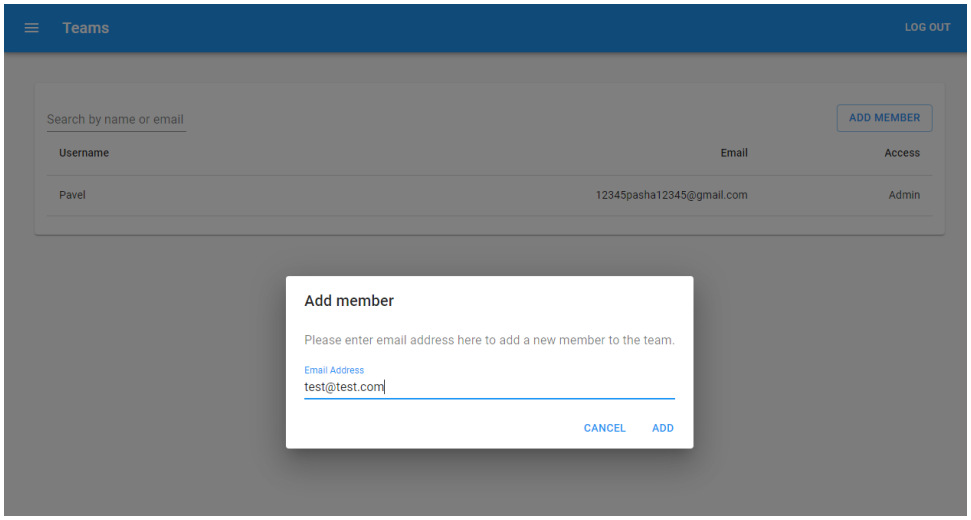


Рис. 4.17 – Екран відправлення запрошення на вступ до команди

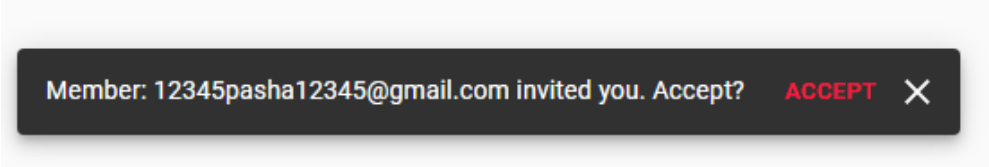


Рис. 4.18 – Вигляд запрошення, що з'являється після входу в обліковий запис

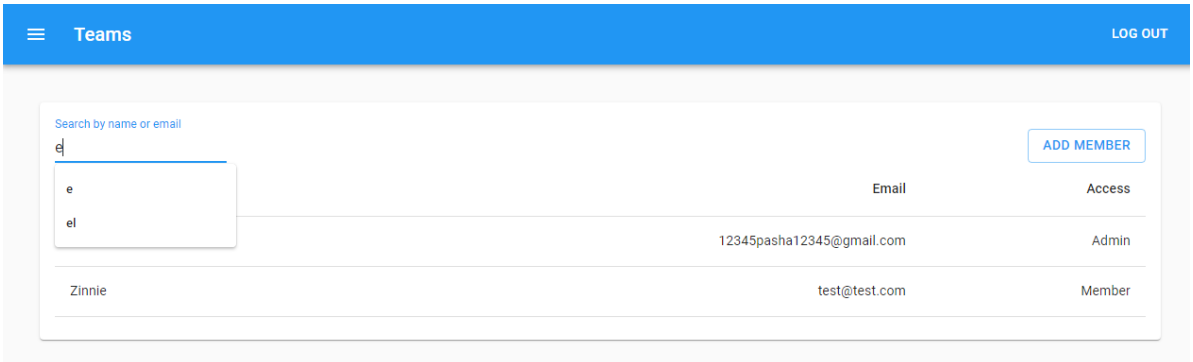


Рис. 4.19 – Процес пошуку серед членів команди

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі був проведений опис додатку, щоб перевірити на правильність функціонування та знайти можливі помилки, якщо такі є.

В результаті перевірки, було визначено, що даний веб-додаток не містить помилок, а також функціонує згідно з вимогами, які необхідні для роботи системи моніторингу робочого часу працівників. Було показано та описано користувацький інтерфейс, який є дуже зрозумілим та простим у використанні.

Також, завдяки гнучкій структурі, даний додаток можна розширювати й додавати інший функціонал за необхідністю у будь-який час.

ВИСНОВКИ

В даній бакалаврській роботі було досліджено, створену та проаналізовано роботу системи моніторингу робочого часу працівників. Дана система була побудована у вигляді SPA веб-додатку за допомогою програмного середовища розробки Visual Studio Code.

Для написання програмного коду використовувався фреймворк React.js на основі мови JavaScript. Крім того, більшість компонентів для взаємодії з клієнтом було побудовано за допомогою популярної бібліотеки React – Material-UI. Для збереження стану додатку використовувалась бібліотека Redux.

Для написання серверної частини використовувався Express фреймворк, який прослуховується на порту 4000 та передає декілька middleware функцій, одна з яких відповідає за взаємодію з клієнтом в не залежності від домену (задаючи заголовки відповідей Access-Control-Allow-Origin) та bodyParser, шо повертає відповіді у JSON-форматі. В якості аутентифікації запитів використовується middleware функція, яка перевіряє token кожного запиту та, у випадку успіху, передає керування функції з відповідного endpoint.

У хмарному сховищі MongoDB Atlas всі дані зберігаються на кластері. Кластер MongoDB створюється для будь-якого основного постачальника хмарних обчислень за нашим вибором, За допомогою MongoDB Atlas. Використовуючи призначений для користувача інтерфейс на основі браузера Atlas, також можливо інтуїтивно налаштувати кластер і контролювати його продуктивність. Зв’язок з MongoDB, що зберігає дані як документи, відбувається за допомогою Mongoose – бібліотеки JavaScript. Всі документи мають JSON-формат.

Додаток не містить помилок і виконує поставлене перед ним завдання щодо створення системи моніторингу робочого часу працівників. Але даний веб-додаток не є ідеальним, оскільки його можна покращувати, додаючи більш широкий спектр функціоналу. Гнучкість

даного веб-додатку дозволяє додавати функціонал в будь-який момент часу за необхідністю. Тому, в майбутньому, має місце додати й такий функціонал:

- Можливо додати чат між членами команди, для того щоб користувачі не відволікались на інші додатки, а також могли спілкуватися безпосередньо в одному додатку;
- Якщо користувач не зайшов у свій обліковий запис і йому прийшло запрошення на вступ до команди, можна додати повідомлення, яке відправляється в один з популярних месенджерів (наприклад, Viber чи Telegram);
- Додати функцію відслідковування кількості натиснутих клавіш, щоб покращити оцінку ефективності роботи працівників;
- Якщо начальник більш строгого стеження, можливо додати запис екрану працівників.

Таким чином, даний веб-додаток є дуже перспективним і його можна розроблювати й надалі, якщо того будуть потребувати користувачі. Усі поставлені завдання і цілі були успішно виконані.

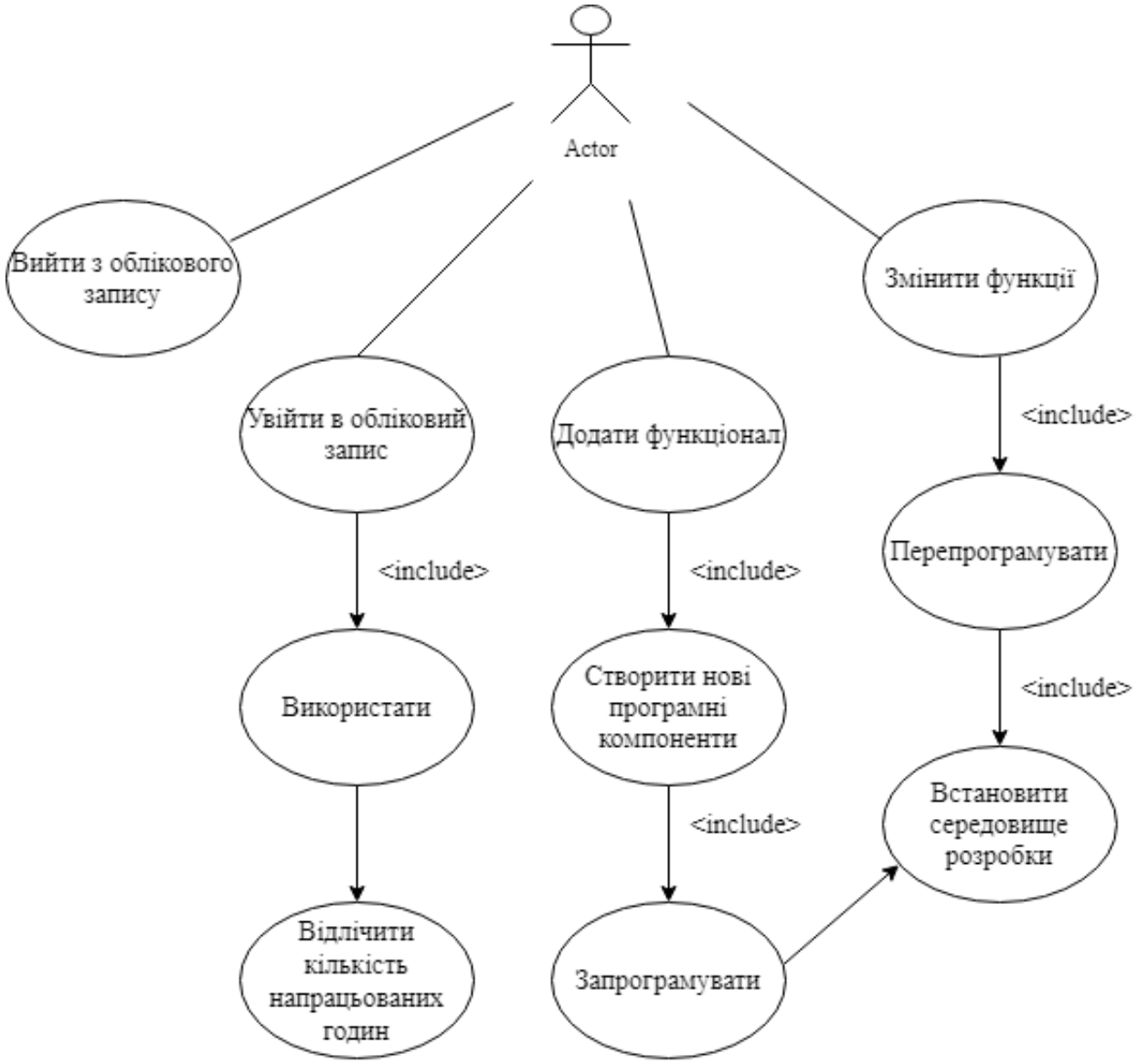
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. https://www.bamboohr.com/hr-glossary/time-tracking/
2. https://timetracker.yaware.com.ua/uk/
3. https://www.getharvest.com/harvest-time-tracking
4. https://timenotes.io/features-list
5. https://www.actitime.com/time-tracking/what-is-time-tracking/
6. https://habr.com/post/322532/
7. https://metanit.com/nosql/mongodb/1.1.php
8. https://metanit.com/nosql/mongodb/2.1.php
9. https://proselyte.net/tutorials/mongodb/
10. https://code.tutsplus.com/ru/tutorials/create-a-database-cluster-in-the-cloud-with-mongodb-atlas--cms-31840
11. https://code.tutsplus.com/uk/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527
12. https://merehead.com/ru/blog/single-page-application-vs-multi-page-application/
13. http://shiftoffproblem.com/what-is-spa-and-mpa/
14. https://echoinnovateit.com/single-page-app-vs-multi-page-app/
15. https://medium.com/better-programming/js-vanilla-script-spa-1b29b43ea475
16. https://clockwise.software/blog/single-page-applications-are-they-a-good-choice-for-your-project/
17. https://angular.io/guide/architecture
18. https://docs.angularjs.org/guide/introduction
19. https://vuejs.org/v2/guide/
20. https://router.vuejs.org/
21. https://blog.udemy.com/react-js-vs-angular-vs-vue-js-which-is-the-best-javascript-framework/
22. https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

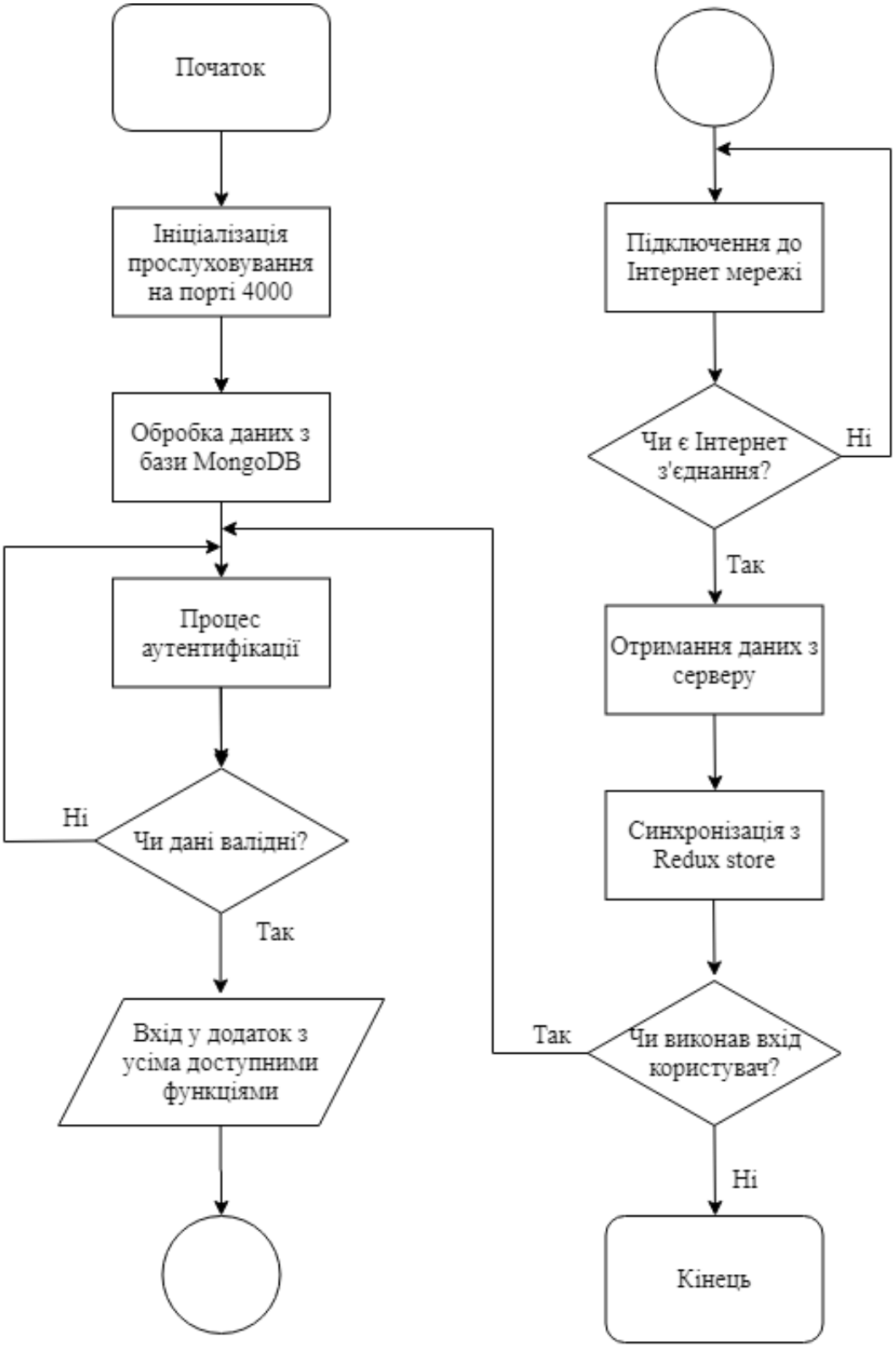
23. https://binariks.com/blog/what-to-choose-in-2020-angular-vs-react-vs-vuejs-comparison/
24. https://stackshare.io/stackups/visual-studio-code-vs-webstorm
25. https://dev.to/mokkapps/why-i-switched-from-visual-studio-code-to-jetbrains-webstorm-939
26. https://www.jetbrains.com/webstorm/promo/
27. https://code.visualstudio.com/
28. https://reactjs.org/docs/hooks-state.html
29. https://medium.com/@marina.kovalyova/flux-the-react-js-application-architecture-773f515d068d
30. https://redux.js.org/introduction/getting-started
31. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin
32. https://www.npmjs.com/package/express
33. https://www.npmjs.com/package/bcrypt
34. https://material-ui.com/

ДОДАТОК А

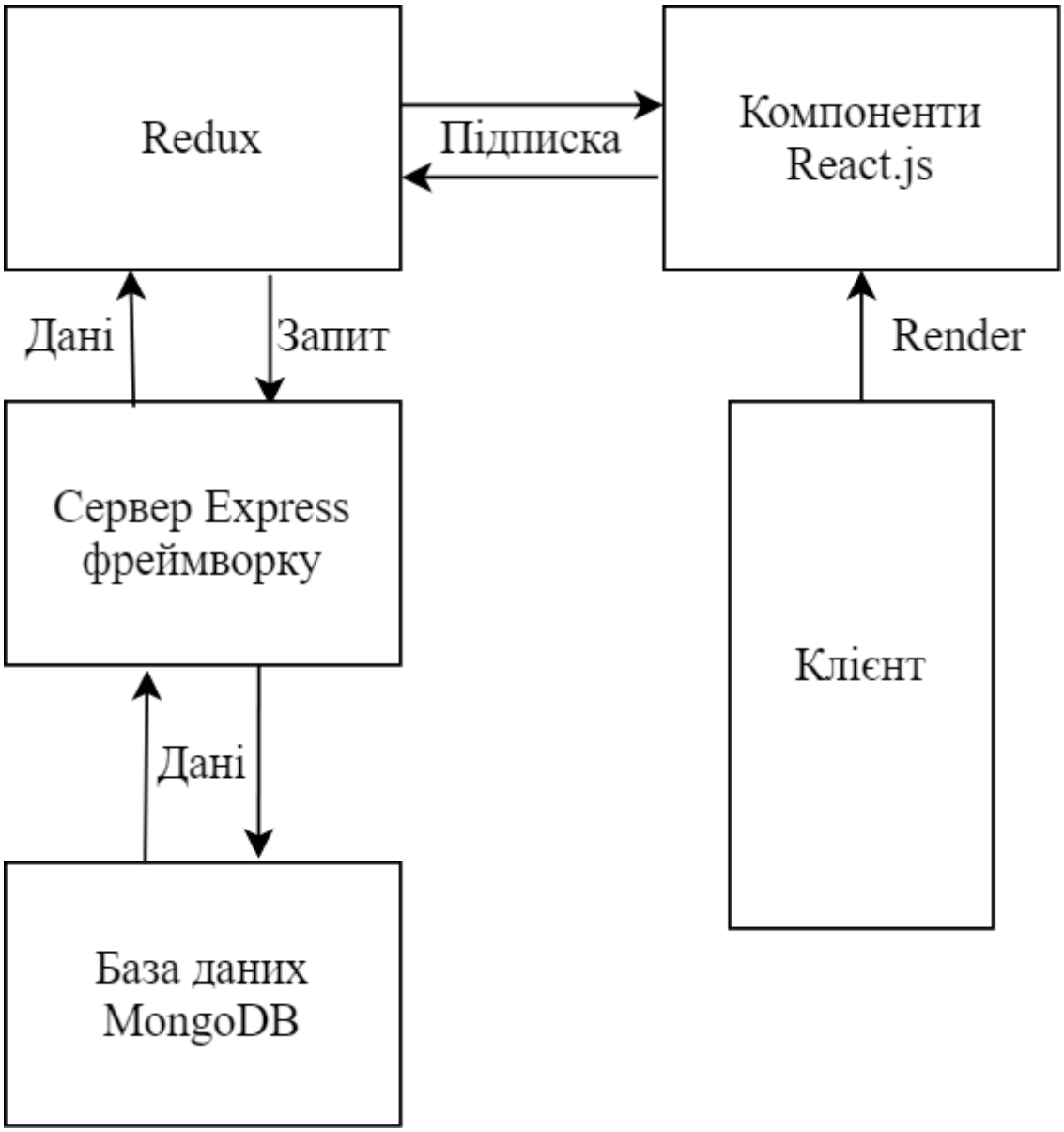
Київ – 2020 року



					ІАЛЦ.467100.004 Д1	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Зінкевич П.О.*			*Система моніторингу робочого часу працівників Структурна схема сценарію виконання*	*Літ.* / *Аркуш* / *Аркушів*
Перевір.		*Луцький Г.М.*				
Н. Контр.		*Сімоненко В.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-63*
Затв.						



					ІАЛЦ.467100.004 Д2	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Зінкевич П.О.*			*Система моніторингу робочого часу працівників Блок-схема алгоритму*	*Літ.* / *Аркуш* / *Аркушів*
Перевір.		*Луцький Г.М.*				
Н. Контр.		*Сімоненко В.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-63*
Затв.						



					ІАЛЦ.467100.004 Д3	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Зінкевич П.О.			Система моніторингу робочого часу працівників Функціональна схема алгоритму	Літ. / Аркуш / Аркушів
Перевір.		Луцький Г.М.				
Н. Контр.		Сімоненко В.				НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-63
Затв.						