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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія для обліку робочого часу працівників»

Виконав: студент 2-го курсу, групи
3КН-23м спеціальності
122 «Комп'ютерні науки»
_____ Валявський О.С.
(прізвище та ініціали)

Керівник: к.т.н., асистент каф. КН
_____ Богач І.В.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: к.т.н., доцент каф. АІІТ
_____ Гармаш В.В.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
«06» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.

11. 10 2024 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Валявському Олексію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія для обліку робочого часу працівників
керівник роботи Богач Ілона Віталіївна, к.т.н., асистент кафедри КН
затверджені наказом вищого навчального закладу від "17" 09. 2024 р.
№ 310
2. Строк подання студентом роботи 11.11. 2024 року

3. Вихідні дані до роботи:
тип програми – онлайн платформа; модель розробки – ітеративна; засоби організації даних програми – база даних MySQL; інформація про вакансії та працівників; середовище розробки – Visual Studio Code; мова програмування – Javascript, фреймворк React.
4. Зміст текстової частини:
Вступ, обґрунтування вибору методу розробки та постановка задач дослідження; розробка структури та алгоритмів роботи програмного продукту; розробка програмних компонент онлайн платформи; тестування програм економічна частина, висновки, перелік використаних джерел, додатки
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Алгоритм роботи програми, UML діаграма класів, робочі вікна програми, результати тестування програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Богач І. В., к.т.н, ас. кафедри КН		
5	Адлер О. О., к.е.н., доц. каф. ЕПВМ		

7. Дата видачі завдання 02.09 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка та аналіз задачі	02.09.24 – 24.09.24	
2	Побудова моделей	25.09.24 - 01.10.24	
3	Практичне застосування та оцінка ефективності розроблених моделей	02.10.24 – 28.10.24	
4	Підготовка економічної частини	29.10.24 - 01.11.24	
5	Апробація та/або впровадження результатів дослідження	02.11.24 – 04.11.24	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	05.11.24 - 11.11.24	

Студент _____ Валявський О. С.
(підпис)

Керівник роботи _____ Богач І. В.
(підпис)

АНОТАЦІЯ

УДК 004.413.2

Валявський О. С. Розробка інформаційної технології для обліку робочого часу працівників : магістерська кваліфікаційна робота зі спеціальності 122 Комп'ютерні науки, освітня програма – Системи штучного інтелекту. Вінниця: ВНТУ, 2024. 123 с.

На укр. мові. Бібліогр.: 25 назв; рис.: 28; табл.: 16.

У магістерській кваліфікаційній роботі проведено детальний аналіз сучасних методів обліку робочого часу працівників. Запропоновано створення веб-додатку для автоматизації обліку робочого часу з використанням фреймворка React. Проведено аналіз аналогічних систем, обґрунтовано актуальність розробки та визначено основні вимоги до програмного забезпечення.

Розроблено алгоритми та блок-схеми для функціональних модулів системи, спроектовано структуру програмного засобу. Веб-додаток створено з використанням мов програмування JavaScript, HTML та CSS, а також фреймворка React. Для розробки застосовано середовище Visual Studio Code.

Отримані результати можуть бути впроваджені на підприємствах різного типу для автоматизації процесів обліку робочого часу, що підвищить ефективність управління персоналом і спростить контроль над виконанням робочих завдань.

Ключові слова: облік робочого часу, інформаційна технологія, React, веб-додаток.

ABSTRACT

Valyavsky O. S. Development of information technology for the work hours of practitioners: master degree work in the specialty 122 Computer Science, graduate program - Artificial Intelligence Systems. Vinnytsia: VNTU, 2024. 123 p.

In Ukrainian movie Bibliography: 25 titles; Fig.: 28; table: 16.

The master's thesis carried out a detailed analysis of the current methods of working hours for doctors. A web application has been created to automate the work hours using the React framework. An analysis of similar systems was carried out, the relevance of the development was determined and the main benefits to software provision were identified.

Algorithms and block diagrams for the functional modules of the system were developed, and the structure of the software was designed. The web application was created using JavaScript, HTML and CSS programming, as well as the React framework. For the sake of clarity, the middle of Visual Studio Code has been put together.

The results can be used in various types of enterprises to automate processes related to working hours, which will increase the efficiency of personnel management and simplify control over the completion of work tasks.

Keywords: work environment, information technology, React, web add-on.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний розвиток інформаційних технологій значно впливає на всі сфери людської діяльності, включаючи управління та організацію робочих процесів. Одним із важливих аспектів ефективного функціонування підприємств і організацій є облік робочого часу працівників, який дозволяє оптимізувати управлінські процеси, підвищити продуктивність та забезпечити своєчасне виконання завдань.

Традиційні методи обліку робочого часу, засновані на ручних записах або застарілих програмних засобах, нерідко характеризуються низькою точністю, значними затратами часу та людськими помилками.

Використання спеціалізованих програмних засобів для обліку робочого часу працівників дає змогу зменшити вплив людського фактора, забезпечити швидкість обробки даних і високий рівень точності. Сучасні технології, такі як хмарні сервіси, мобільні додатки та інтеграція з біометричними системами, відкривають нові можливості для розробки таких систем.

Облік робочого часу є не лише важливим інструментом контролю, але й способом підвищення мотивації працівників. Забезпечення прозорості обліку сприяє зміцненню довіри між працівниками та керівництвом, а також стимулює працівників до більш ефективного виконання своїх обов'язків.

Таким чином, створення інформаційної технології для обліку робочого часу працівників є актуальним завданням, яке сприятиме вдосконаленню управління персоналом та підвищенню ефективності організації в цілому.

Зв’язок роботи з науковими програмами, планами, темами. Ця магістерська кваліфікаційна робота виконується в рамках досліджень кафедри комп’ютерних наук. Вона відповідає 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем»

Мета та завдання дослідження.

Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей інформаційної технології для обліку робочого

часу працівників, яка забезпечить автоматизацію процесу, підвищить його точність та мінімізує витрати часу на облік.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Провести аналіз існуючих підходів до обліку робочого часу.

2. Обґрунтувати вибір методів і технологій, які забезпечать ефективне вирішення задачі.

3. Розробити математичну модель інформаційної системи обліку робочого часу.

4. Розробити структуру, алгоритми та архітектуру програмного засобу.

5. Виконати програмну реалізацію інформаційної технології.

6. Провести тестування розробленого продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес структуризації знань галузі обліку робочого часу працівників.

Предмет дослідження – є інформаційна технологія та програмні засоби онтологічного моделювання бази знань обліку робочого часу.

Методи дослідження.

У роботі використано такі методи наукових досліджень:

- системний аналіз для вивчення процесу обліку робочого часу;
- об’єктно-орієнтоване програмування для створення програмного забезпечення;
- методи математичного моделювання для проектування структури системи;
- методи тестування програмних засобів для оцінки їх точності та ефективності.

Наукова новизна одержаних результатів.

1. Удосконалено інформаційну технологію обліку робочого часу працівників, яка відрізняється від існуючих інтеграцією з біометричними пристроями та мобільними додатками, що дозволило забезпечити високу точність, автоматизацію процесу та гнучкість у використанні.

2. Розроблено покращені алгоритми обробки даних, які забезпечують підвищення швидкості та надійності функціонування системи обліку.

Практичне значення роботи.

1. Розроблено алгоритм функціонування інформаційної технології обліку часу працівників.

2. Розроблено діаграму послідовності взаємодії модулів інформаційної технології обліку часу працівників.

3. Здійснено програмну реалізацію інформаційної технології обліку часу працівників.

Особистий внесок магістранта.

Усі результати, наведені у роботі, отримані автором самостійно.

Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел і додатків, у які винесено протокол перевірки кваліфікаційної роботи на наявність текстових запозичень, лістинг програми й ілюстративний матеріал до захисту роботи.

Апробація та публікації результатів роботи. Результати роботи було представлено на Всеукраїнській науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025) [1].

1 ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану розробки Web-системи з урахуванням часу працівників

У цьому розділі проведено всебічний аналіз стану розробки веб-систем для моніторингу часу роботи працівників на проєктах. Описано основні функціональні можливості, які вже реалізовані в існуючих системах, а також технології, що використовуються в таких рішеннях. Досліджено переваги та недоліки сучасних веб-систем, які допомагають оптимізувати облік робочого часу, і визначено можливі напрями їх вдосконалення [2].

Основні аспекти аналізу:

1. **Дослідження існуючих веб-систем:**
   - Проведено огляд доступних платформ, призначених для моніторингу часу працівників у проєктах.
   - Вивчено функціонал систем, таких як реєстрація робочих годин, фіксація часу для виконання конкретних завдань, відстеження прогресу проєкту, автоматична генерація звітів тощо.
   - Проведено оцінку технологічних рішень, які застосовуються у цих системах (зокрема, використання JavaScript, React, Node.js тощо), їх ефективності та відповідності сучасним стандартам розробки.
2. **Аналіз переваг та недоліків:**
   - Виявлено ключові переваги існуючих систем, серед яких: підвищення продуктивності працівників, покращення управління проєктами, точний облік витраченого часу, гнучке налаштування інтерфейсу.
   - Виділено недоліки: складність у використанні окремих систем, висока вартість впровадження, обмежена адаптивність до

специфічних потреб користувачів, високі вимоги до інфраструктури.

3. **Визначення нових функцій для розробки:**
    - На основі аналізу встановлено функції, яких бракує у сучасних системах, наприклад: інтеграція з популярними системами управління проєктами (Trello, Jira), підтримка мобільних пристроїв, розширені можливості аналітики.
    - Сформовано пріоритети для розробки нових функцій, які забезпечать максимальну вигоду користувачам.
4. **Вивчення впливу часу працівників на ефективність:**
    - Оцінено розподіл часу працівників між завданнями та проєктами.
    - Визначено часові обмеження для завдань, що дозволяє оцінити, чи доступні ресурси для реалізації нових функцій.
    - Враховано фактори швидкості виконання завдань і завантаженості працівників.
5. **Розробка стратегії вирішення проблем:**
    - Запропоновано рішення для усунення недоліків існуючих систем, таких як спрощення інтерфейсу, підвищення швидкості роботи програми, інтеграція з іншими популярними платформами.
    - Оцінено витрати на впровадження нових функцій та розраховано необхідні ресурси.
    - Визначено оптимальні шляхи розвитку системи, враховуючи часові та фінансові обмеження.

Цей детальний аналіз дозволяє не лише зрозуміти поточний стан ринку веб-систем для моніторингу робочого часу, але й формулює чіткі вимоги для розробки нової системи, яка враховуватиме потреби користувачів та усуватиме існуючі проблеми [3].

1.2 Порівняльний аналіз аналогів

Один з аналогів, який був розглянутий, це Toggle рисунок 1.1 [4]. Toggle є онлайн-сервісом, призначеним для відстеження робочого часу. Він надає широкий спектр функціональних можливостей, що включають створення проєктів, задач, ведення статистики виконаних робіт та експорт звітів у різні формати.

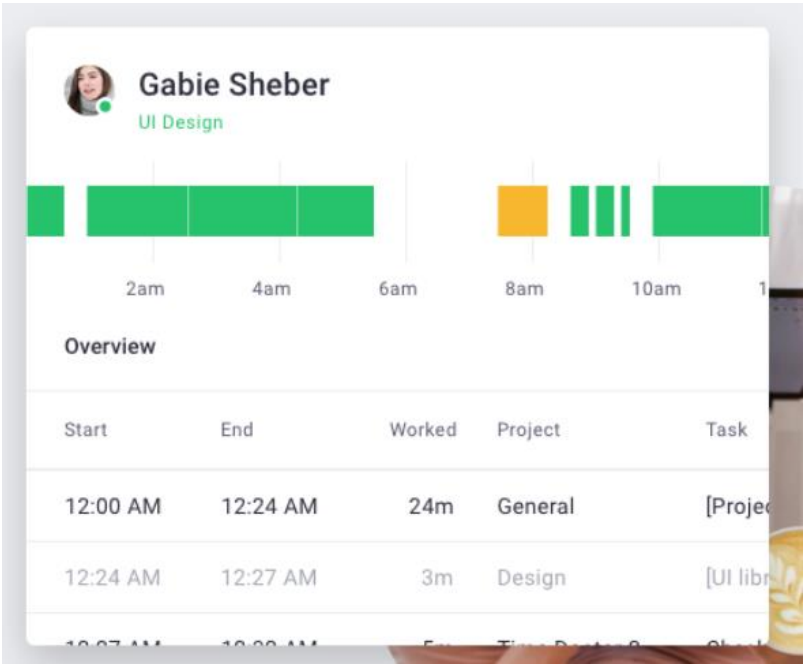

Рисунок 1.1 – Інтерфейс ресурсу Toggl

Однією з переваг Toggle є його широкий функціонал. Користувачі можуть створювати різні проєкти та задачі, які дозволяють категоризувати робочий час за різними критеріями. Крім того, Toggle забезпечує можливість вести детальну статистику виконаної роботи, що дозволяє аналізувати ефективність та продуктивність працівників [5].

Ще однією перевагою Toggle є можливість експортувати звіти у різні формати. Це дає змогу поділитися результатами роботи з колегами або керівництвом, а також використовувати дані для подальшого аналізу та обліку робочого часу.

Однак, наряду з перевагами, Toggle має й деякі недоліки. Наприклад, його використання може бути пов'язане з певними витратами, оскільки він є комерційним продуктом, і деякі його функціональності можуть бути доступні лише за певною підпискою або платою [6].

Загалом, Toggle є потужним інструментом для відстеження робочого часу та аналізу продуктивності. Його функціональні можливості та можливість експорту звітів роблять його привабливим варіантом для багатьох організацій і команд, що займаються управлінням проєктами та контролем робочого часу.

Між іншими аналогами системи моніторингу часу працівників на проєктах, який був проаналізований, є Harvest рисунок 1.2 [7]. Harvest є системою для відстеження робочого часу, що дозволяє реєструвати час, витрачений на конкретні проєкти, вести список задач і відстежувати прогрес виконання робіт.

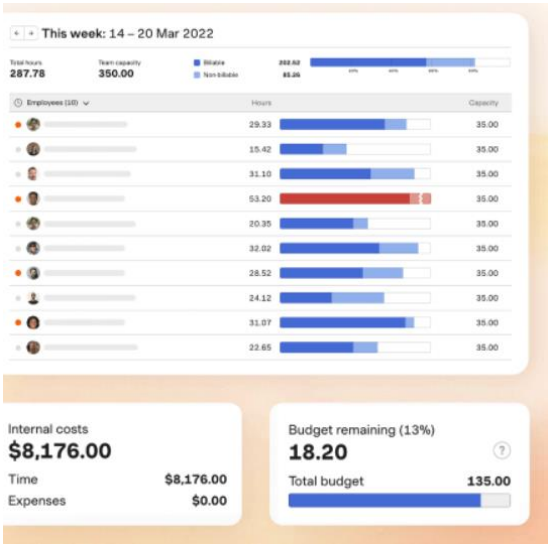

Рисунок 1.2 – Інтерфейс додатку harvest

Однією з ключових переваг Harvest є його простота використання. Інтерфейс користувача дуже зрозумілий і інтуїтивно зрозумілий, що дозволяє швидко орієнтуватися в системі і почати використовувати її без великих зусиль. Крім того, Harvest надає зручні інструменти для створення списку задач та організації роботи над проєктами [8].

Ще однією перевагою Harvest є його функціональність. Система дозволяє фіксувати час роботи на конкретних проєктах, що дозволяє точно відстежувати, скільки часу було витрачено на кожну задачу або проєкт. Крім того, Harvest надає можливість стежити за прогресом робіт і аналізувати продуктивність працівників [9].

Однак, Harvest також має свої недоліки. Наприклад, деякі користувачі відзначають, що інтерфейс системи може бути трохи застарілим і не настільки естетично привабливим, як у деяких інших аналогів. Крім того, певні функціональності Harvest можуть бути обмеженими у безкоштовній версії, і для доступу до повного спектру функцій може знадобитися підписка або платня.

Ще одним аналогом системи моніторингу часу працівників на проєктах є Hubstaff рисунок 1.3. Hubstaff є програмою для моніторингу часу роботи, яка надає широкі можливості для контролю та відстеження працівників [10].

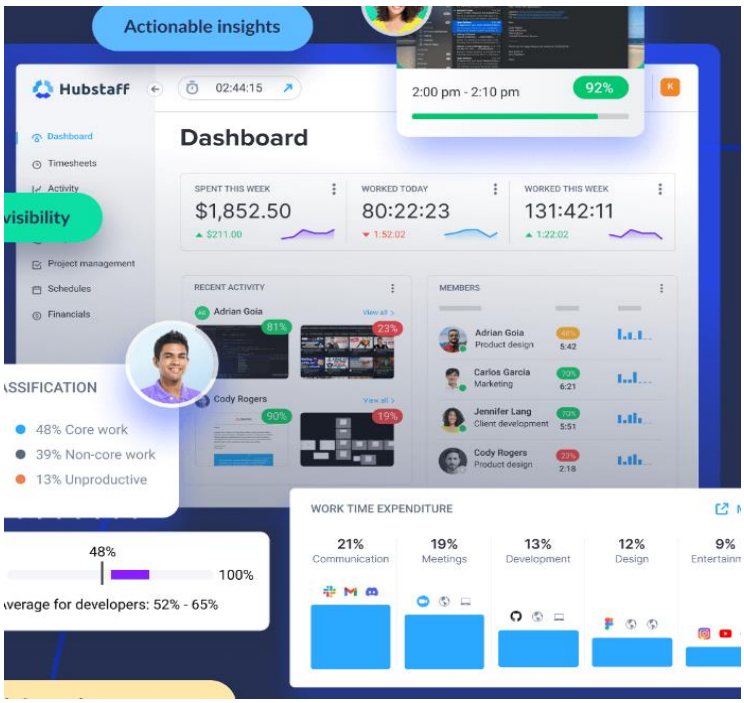

Рисунок 1.3 – Інтерфейс ресурсу Hubstaff

Однією з ключових особливостей Hubstaff є можливість вести облік робочого часу працівників. Програма автоматично фіксує час, проведений на

різних задачах та проєктах, що дозволяє точно відстежувати, як працівники витрачають свій робочий час. Крім того, Hubstaff надає можливість контролювати активність користувача на комп'ютері, фіксувати роботу з додатками та веб-сайтами, що дозволяє отримати детальну інформацію про продуктивність та ефективність роботи.

Проте, Hubstaff також має свої недоліки. Один з них - висока вартість. Використання повнофункціональних можливостей Hubstaff може вимагати підписки або плати, що може бути недоцільним для деяких команд або малих підприємств. Крім того, інтерфейс Hubstaff може бути складним для новачків, і може знадобитися певний час для ознайомлення з усіма функціями та налаштуваннями програми.

Отже, здійснимо порівняльний аналіз даних засобів та розробленого додатку TimeTracker (табл.1.1). В таблиці представлений порівняльний аналіз трьох програмних продуктів для моніторингу часу працівників на проєктах: Toggle, Harvest та Hubstaff, разом з розробленим додатком TimeTracker. Розроблений додаток TimeTracker включає усі основні функції, які присутні в аналогічних програмних продуктах, такі як відстеження часу роботи, статистика та звіти. Однак, додаток також має свої переваги, такі як простий інтерфейс, можливість налаштування та пристосування до потреб користувача, а також можливість інтеграції з іншими інструментами та сервісами [11].

Однак, важливо враховувати, що вибір програмного продукту для моніторингу часу працівників на проєктах залежить від потреб команди або підприємства. Кожен з аналогів має свої переваги та недоліки, і розроблення власного додатку може дати можливість налаштувати його під конкретні вимоги та потреби користувачів.

Таблиця 1.1 – Порівняльний аналіз аналогів

Характеристики / Функції	Toggl	Harvest	Hubstaff	TimeTracker
не надають детальної інформації про робочий час працівників на проєктах	+	–	–	+
Підтримка відео та аудіо	+	+	+	+
не мають можливості редагування вже створених записів	–	–	–	+
Ціна	Від $5999 на рік	Від €49 на місяць	Безкоштовно	Безкоштовно
Сума	2	1	2	4

В результаті проведеного аналізу встановлено, що розроблений програмний засіб, TimeTracker, виявився більш функціональним порівняно з розглянутими аналогами - Toggle, Harvest та Hubstaff.

TimeTracker має весь необхідний набір функцій для моніторингу часу працівників на проєктах, включаючи відстеження часу роботи, статистику та звіти. Крім того, він має переваги, такі як простий інтерфейс, можливість налаштування та пристосування до потреб користувача, а також можливість інтеграції з іншими інструментами та сервісами [12].

Порівняно з аналогами, TimeTracker надає широкий функціональний спектр і може бути більш універсальним інструментом для відстеження часу працівників на проєктах. Він дозволяє встановлювати та керувати завданнями, вести статистику та аналізувати продуктивність роботи, а також експортувати дані для подальшого аналізу.

Таким чином, розроблений програмний засіб TimeTracker може бути вигідним вибором для команд або підприємств, які потребують ефективного

моніторингу часу працівників на проєктах і бажають мати більш широкий функціональний набір, який може бути налаштований під їхні потреби.

1.3 Аналіз методів розв’язання поставленої задачі та обгрунтування вибору нейромережевого методу

Для реалізації функцій моніторингу часу працівників на проєктах, необхідно визначити підходи до розв'язання поставленої задачі. Одним з таких підходів може бути використання часових журналів (time tracking).

Time tracking дозволяє реєструвати час, витрачений працівниками на виконання конкретних завдань або проєктів. Це допомагає визначати кількість часу, витраченого на конкретний проєкт, а також ефективність роботи працівників.

Існує багато інструментів для time tracking, які можуть бути використані для розробки нашої системи. Одним з найбільш популярних є Toggl. Цей інструмент дозволяє вести облік часу, витраченого на роботу над конкретними завданнями. Крім того, він має можливості звітування та інтеграції з іншими інструментами.

Іншим підходом може бути використання системи тікетів (ticketing system). Системи тікетів, які використовуються для управління проєктами, дозволяють додавати завдання та назначати їх на конкретних працівників. Крім того, такі системи можуть дозволяти вести облік часу, витраченого на виконання кожного завдання.

Одним з недоліків цього підходу є те, що він може бути менш гнучким, оскільки завдання повинні бути відкриті та закриті відповідно до процесу управління проєктами, що може бути складним для деяких проєктів.

Загалом, обраний метод моніторингу часу працівників буде залежати від потреб конкретного проєкту та команди розробників.

Однак, в даній курсовій роботі для розробки програмного забезпечення моніторингу часу працівників на проєктах було вирішено використати підхід з

використанням бази даних та мови програмування JavaScript з використанням фреймворка React.

Такий вибір обґрунтований тим, що в наш час більшість програмних продуктів розробляються з використанням баз даних, оскільки це дозволяє зберігати інформацію про об'єкти та їх взаємозв'язки в структурованому вигляді та забезпечує високу ефективність операцій з даними. Також, мова програмування JavaScript є досить популярною серед програмістів та добре підходить для веб-розробки, а фреймворк React дозволяє швидко та зручно створювати веб-інтерфейси з високою рівнем інтерактивності та швидкодії [13].

Отже, обрано такий метод розробки програмного забезпечення, який передбачає використання бази даних та мови програмування JavaScript з фреймворком React. Цей метод є оптимальним з точки зору ефективності, швидкості та зручності розробки програмного продукту.

1.4 Аналіз методів розробки за допомогою фреймворка React

В розробці веб-додатку для моніторингу часу працівників на проєктах ми плануємо використовувати Redux для збереження даних та управління станом додатку. Redux надає централізоване сховище (store), в якому зберігаються дані, що використовуються в додатку. Це дозволяє нам легко отримувати доступ до даних з будь-якого компонента та забезпечує їх консистентність.

Для роботи з сервером та забезпечення взаємодії з API ми плануємо використовувати бібліотеку rtk query. Rtk query - це надбудова над Redux, яка надає швидку та просту роботу з взаємодією з API. Вона автоматично генерує необхідні запити та кешує дані, що дозволяє ефективно використовувати мережеві ресурси та покращує швидкість відгуку додатку.

Для роботи з календарем та датами в додатку ми плануємо використовувати бібліотеку date-fns. Date-fns - це набір функцій для роботи з датами у JavaScript. Вона надає зручні та потужні інструменти для роботи з

датами, такі як розрахунок різниці між датами, форматування дат, маніпуляції з часовими зонами та багато іншого. Використання date-fns допоможе нам забезпечити точну та надійну роботу з датами у нашому додатку.

Крім того, для стилізації та розміщення компонентів в інтерфейсі додатку планується використовувати бібліотеку Material-UI (Mui). Material-UI - це популярна бібліотека компонентів, яка надає набір готових стилізованих елементів інтерфейсу користувача, таких як кнопки, форми, таблиці та інші. Використання Material-UI дозволить нам швидко створювати красивий та сучасний інтерфейс додатку з мінімальними зусиллями.

Загалом, комбінація Redux для управління станом, rtk query для взаємодії з API, date-fns для роботи з датами та Material-UI для стилізації компонентів дозволить нам створити потужний, ефективний та привабливий веб-додаток для моніторингу часу працівників на проєктах.

Для хостингу нашого веб-додатку ми плануємо використовувати платформу Amazon Web Services (AWS). AWS є одним з провідних провайдерів хмарних послуг, який надає широкий спектр сервісів для розгортання, масштабування та управління додатками в хмарному середовищі.

Одним з ключових сервісів AWS, який ми плануємо використовувати, є Amazon Elastic Compute Cloud (EC2). EC2 дозволяє нам створювати та управляти віртуальними серверами, на яких буде розгортатися наш веб-додаток. Ми зможемо легко налаштувати потрібні обчислювальні ресурси, масштабувати сервери залежно від потреб та забезпечити надійність та доступність нашого додатку.

Крім того, ми можемо скористатися Amazon Simple Storage Service (S3) для зберігання статичних файлів, таких як зображення, стилі CSS та інші ресурси. S3 надає надійний та масштабований сервіс зберігання об'єктів, який дозволяє нам легко управляти та доставляти статичні файли до наших клієнтів.

Крім того, AWS також надає інші корисні сервіси, які можуть бути використані для підтримки нашого веб-додатку, такі як Amazon RDS для

управління базами даних, Amazon Route 53 для керування доменними іменами та багато інших.

Використання AWS для хостингу нашого веб-додатку дозволить нам отримати масштабоване, надійне та гнучке хмарне середовище, яке забезпечить швидку та стабільну роботу нашого додатку для моніторингу часу працівників на проєктах [14].

1.5 Постановка задач для розробки Web-системи

Розробка веб-системи для моніторингу робочого часу працівників на проєктах має на меті створити інструмент, який автоматизує та спрощує цей процес, забезпечуючи зручність і функціональність для користувачів. Для досягнення цієї мети визначено наступні основні задачі:

1. **Розробка бази даних:**
    - Створення структурованої бази даних для зберігання інформації про працівників, проєкти та відомості про відпрацьований час.
    - Забезпечення підтримки збереження історичних даних для подальшого аналізу.
2. **Інтерфейс для введення даних:**
    - Розробка інтуїтивно зрозумілого інтерфейсу для додавання нових проєктів, користувачів і записів про відпрацьований час.
    - Забезпечення можливості редагування й видалення даних у разі помилкового введення.
3. **Моніторинг робочого часу:**
    - Створення панелі управління для візуалізації робочого часу працівників у реальному часі.
    - Надання можливості фільтрації та сортування записів за проєктами, працівниками або періодами часу.
4. **Генерація звітів:**
    - Розробка функціоналу для створення звітів у форматах PDF і Excel

із даними про робочий час працівників.

- Забезпечення автоматичного надсилання звітів адміністраторам або керівникам проєктів за заданими параметрами.

5. **Доступність і кросплатформенність:**
   - Оптимізація веб-системи для роботи на різних пристроях (ПК, планшетах, смартфонах) та в різних браузерах.
   - Використання адаптивного дизайну для зручної роботи на мобільних пристроях.
6. **Безпека даних:**
   - Захист конфіденційної інформації про працівників і проєкти шляхом впровадження сучасних методів шифрування даних.
   - Реалізація багаторівневого доступу до системи із застосуванням ролей (адміністратор, менеджер, працівник).
   - Захист системи від несанкціонованого доступу, зокрема атаки типу SQL Injection та XSS.

Реалізація цих задач дозволить створити ефективну веб-систему, яка відповідатиме сучасним вимогам до моніторингу робочого часу та забезпечуватиме комфортний досвід для користувачів [15].

1.6 Висновок до розділу 1

У цьому розділі здійснено аналіз існуючих підходів до розробки веб-систем, спрямованих на моніторинг часу працівників, і сформульовано завдання для реалізації нового рішення, яке відповідає сучасним вимогам.

Результати аналізу вказали на те, що більшість наявних систем мають обмежену функціональність, низьку продуктивність або не забезпечують повноцінного виконання завдань, необхідних для ефективного управління робочим часом. Зокрема, було відзначено недостатню інтеграцію з сучасними технологіями та відсутність зручного інтерфейсу користувача.

Для розробки нової веб-системи обрано сучасні інструменти та технології:

- Фреймворк React:
    - Забезпечує динамічну взаємодію з користувачем завдяки використанню віртуального DOM.
    - Дозволяє створювати перевикористовувані компоненти, що спрощує підтримку й масштабування системи.
- Бібліотека Redux:
    - Полегшує управління станом додатку, забезпечуючи централізоване зберігання даних.
    - Допомагає відокремити бізнес-логіку від інтерфейсу, що підвищує гнучкість у розробці.

Окрім цього, розробку буде зосереджено на впровадженні адаптивного дизайну, що гарантує зручність використання системи на різних пристроях. Значна увага буде приділена безпеці даних, зокрема захисту інформації про працівників і проєкти.

Здійснений аналіз та обґрунтування вибору технологій є основою для розробки веб-системи, яка стане зручним і надійним інструментом для моніторингу робочого часу працівників.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМОГО ПРОДУКТУ

2.1 Аналіз даних

Розділ є ключовим для розробки програмного продукту моніторингу часупрацівників на проєктах. У цьому розділі розглядається аналіз вхідних даних, що будуть використовуватися в системі, та опис структури даних, що будуть зберігатися та оброблятися в системі.

У процесі розробки системи моніторингу часу працівників на проєктах, вхідними даними є інформація про робочі години працівників, їхні завдання на проєкті, а також додаткові відомості про проєкти та їхні складові. Для забезпечення повної функціональності системи, необхідно також зберігати дані про дати та терміни завершення проєктів, а також інформацію про клієнтів та їхні контакти [16].

Основна структура даних, що буде використовуватися в системі, буде базуватися на реляційній моделі зберігання даних. Реляційна модель є найбільш зручним та ефективним способом організації даних для даної задачі. Вона передбачає використання таблиць бази даних, які будуть пов'язані між собою за допомогою ключів.

У системі будуть створені наступні таблиці для зберігання даних:

- Таблиця "Працівники" - містить інформацію про працівників, включаючи їхнє ім'я, посаду та контактні дані.
- Таблиця "Проєкти" - містить дані про проєкти, включаючи назву, опис та дати завершення проєкту.
- Таблиця "Завдання" - містить інформацію про завдання, призначені працівникам на конкретних проєктах. Кожне завдання пов'язане зі своїм працівником та проєктом.
- Таблиця "Клієнти" - містить дані про клієнтів, замовників проєктів, включаючи їхнє ім'я та контактні дані.

Ці таблиці будуть мати відповідні стовпці для зберігання необхідних даних. При цьому, будуть використовуватися взаємозв'язки між таблицями за допомогою ключів, що дозволить забезпечити цілісність даних та зручний доступ до необхідної інформації. Таким чином, аналіз даних у цьому розділі включає визначення необхідних вхідних даних для системи моніторингу часу працівників на проєктах, а також опис структури та зв'язків між таблицями бази даних, що будуть використовуватися для зберігання та обробки цих даних. Такий аналіз є важливим етапом у розробці програмного продукту та допоможе забезпечити ефективну роботу системи моніторингу часу працівників на проєктах [17].

2.2 Розробка структури графічного інтерфейсу користувача

GUI або графічний інтерфейс користувача є формою інтерфейсу, що дозволяє взаємодіяти з програмним забезпеченням за допомогою візуальних об'єктів. Ці об'єкти можуть бути представлені як графічні примітиви, такі як кнопки, піктограми, або складніші елементи, наприклад, меню.

Забезпечення комфорту користувача в додатку не залежить тільки від якості реалізованого функціоналу, але також від зручності та інформативності GUI. Програмне забезпечення для управління конфігураціями потребує комплексного інтерфейсу, що дозволяє користувачу не заплутатися в різноманітних елементах системи. З цією метою була розроблена структурна схема головного вікна програми, де різні функції відокремлені за допомогою вкладок. Структурну схему можна побачити на рисунку 2.1.

Отже, розроблений GUI дозволяє користувачу зручно та ефективно взаємодіяти з програмним забезпеченням, забезпечуючи зрозумілість та наочність представлення функцій.

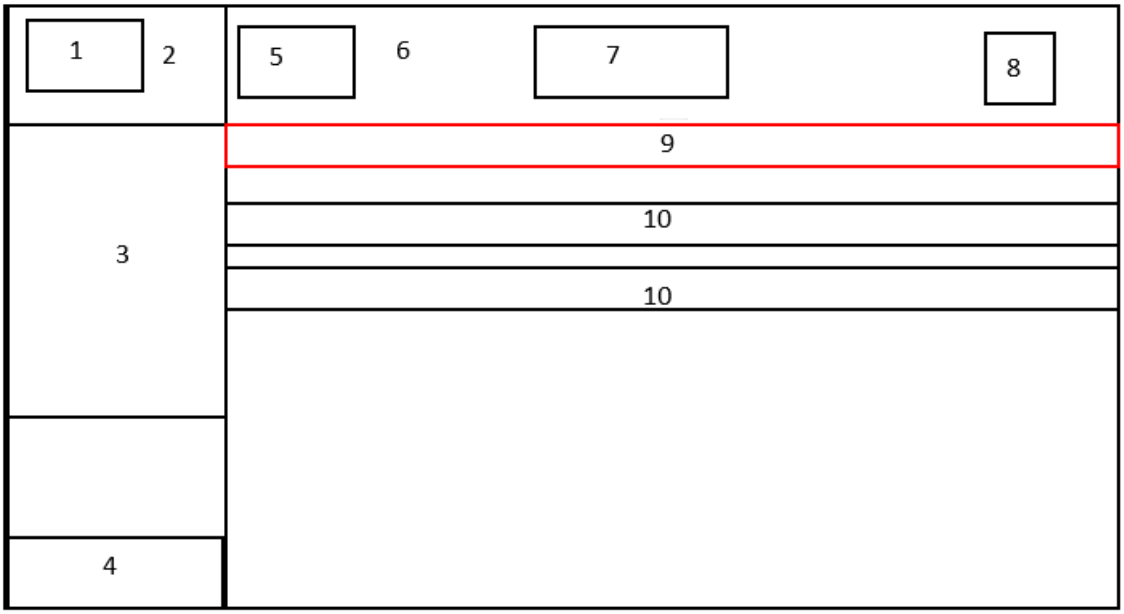

Рисунок 2.1 – Структурна схема головного вікна додатку

Основні елементи інтерфейсу головного вікна:

1) Логотип;

2) Сайдбар;

3) Навігація;

4) Профіль користувача;

5) Заголовок сторінки;

6) Шапка сторінки;

7) Календар;

8) Кнопка логування часу;

9) Шапка таблиці логованих проєктів;

10) Логовані проєкти.

Вміст центральної області вікна (9) залежить від обраної користувачем вкладки, що здійснюється за допомогою відповідних кнопок. Розглянемо більш детально схему інтерфейсу вкладки «Інвентар», що зображена на рисунку 2.2.

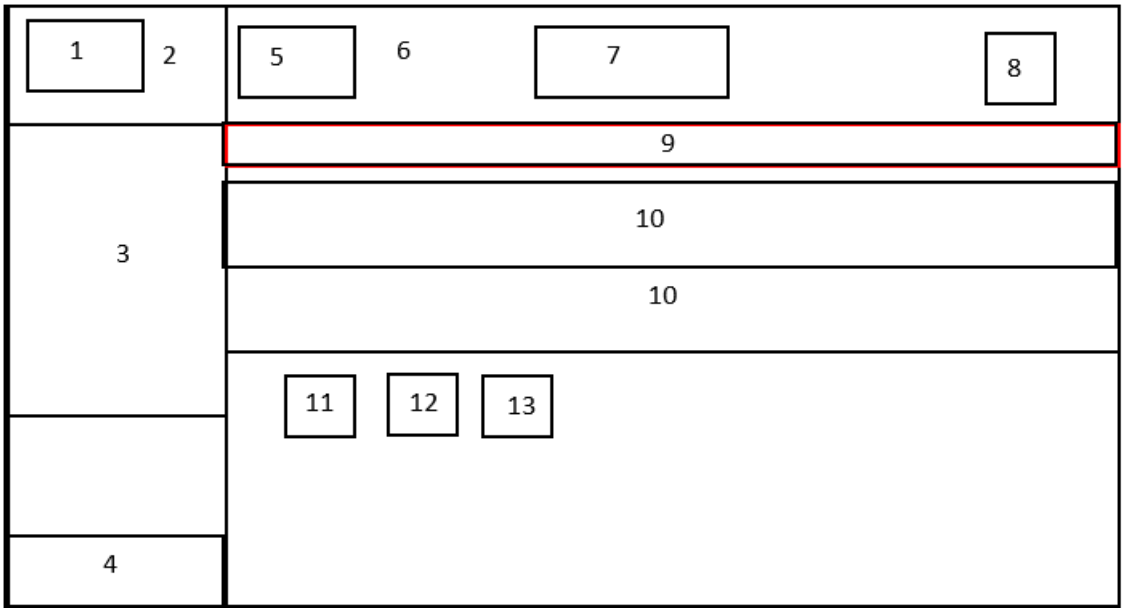

Рисунок 2.2 – Структурна схема вкладки «Логування за неділю»

Вкладка «Логування за неділю» включає в себе лише три елементи інтерфейсу:

1) Логотип;
2) Сайдбар;
3) Навігація;
4) Профіль користувача;
5) Заголовок сторінки;
6) Шапка сторінки;
7) Календар;
8) Селект бокс;
9) Шапка таблиці для заповнення логів на неділю;
10) Форма заповнення лога на неділю;
11) Кнопка для підтвердження даних;
12) Кнопка добавити рядок;
13) Кнопка скинути зміни.

Для логування часу за неділю потрібно заповнити всі поля форми і потім стане доступна кнопка 11.

Структурна схема вкладки «Статистика» зображена на рисунку 2.3.

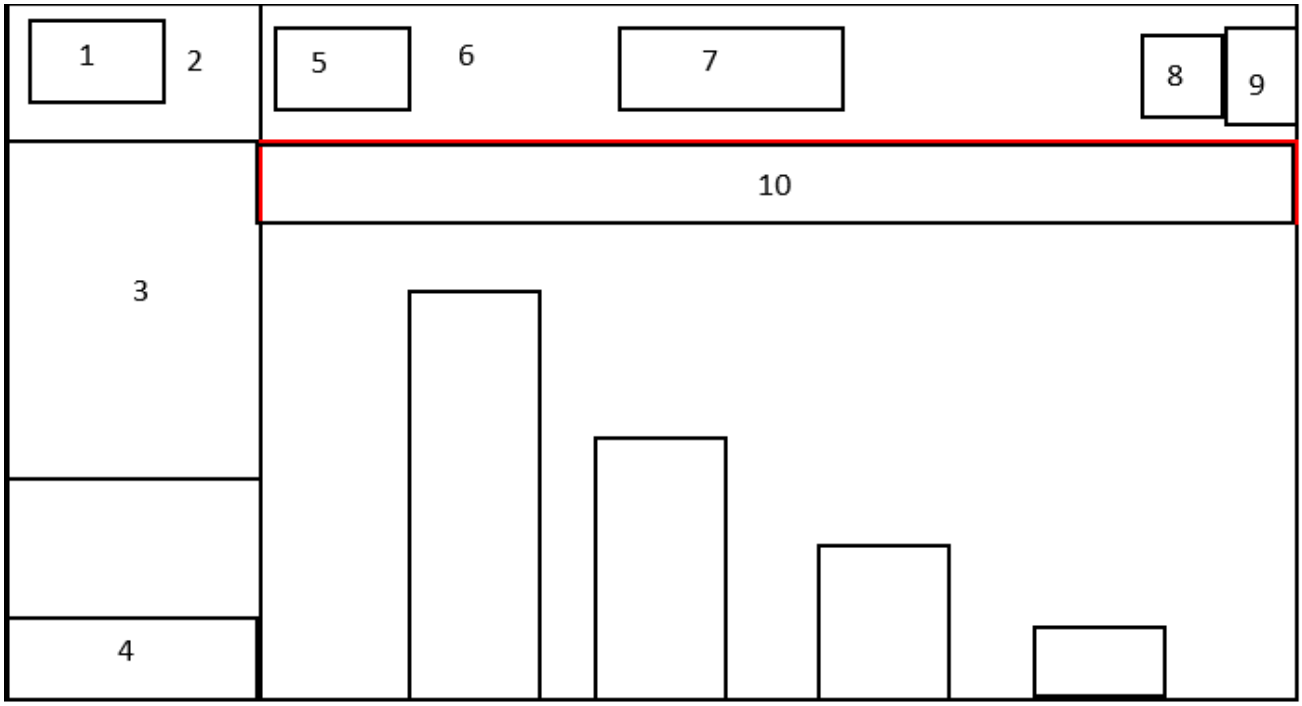

Рисунок 2.3 – Структурна схема вкладки «Статистика»

Вкладка «Статистика» включає в себе наступні елементи інтерфейсу:

1) Логотип;
2) Сайдбар;
3) Навігація;
4) Профіль користувача;
5) Заголовок сторінки;
6) Шапка сторінки;
7) Календар;
8) Модальне вікно фільтри;
9) Кнопка для завантажування звітів по користувачу;
10) Інформація про часи на проєкті;
11) Інформація про проєкти в часах.

На цій сторінці користувач може вибрати будь який проєкти для цього потрібно відкрити 8 модальне вікно фільтри та обрати проєкти або виконавці

проєктів по яким користувач хоче здійснити статистику та після вибору всьо необхідного потрібно здійснити клік на кнопку підтвердити що у модальному вікні та чекати на результат. Якщо результат буде відсутній то буде показано картинку що нічого не знайдено, також в календарі (7) можна змінити дату по яким числам можна буде дивитися статистику.

Схема сторінки «Користувачів» наведена на рисунку 2.4.

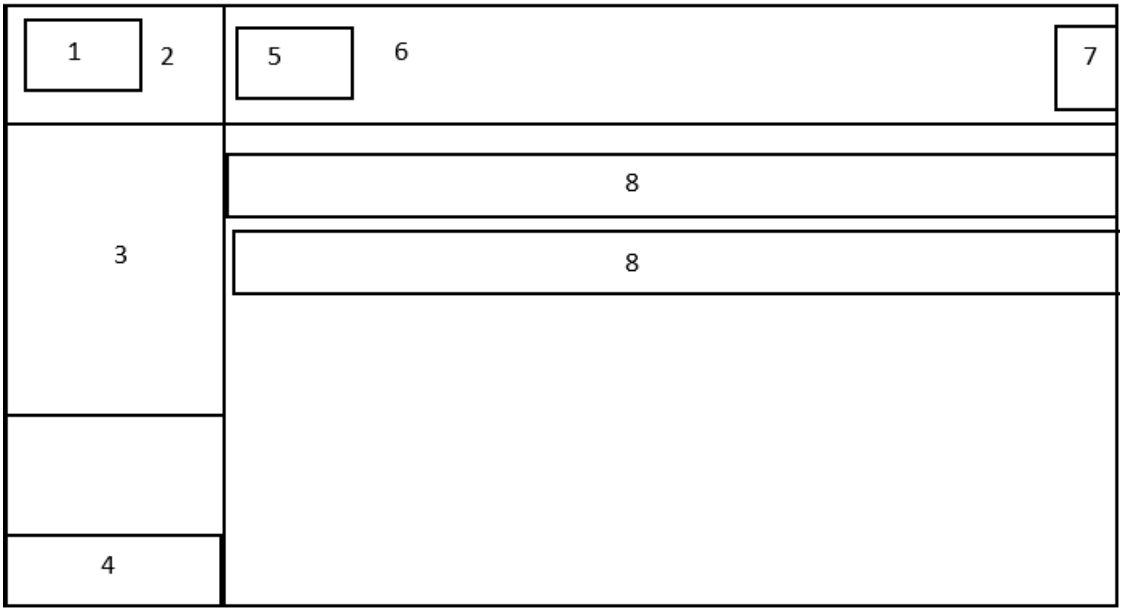

Рисунок 2.4 – Структурна схема вкладки «Виконання»

Елементи інтерфейсу, що знаходяться у цій вкладці, наступні:

1) Логотип;
2) Сайдбар;
3) Навігація;
4) Профіль користувача;
5) Пошук користувачів;
6) Шапка сторінки;
7) Кнопка додати нового користувача;
8) Інформація про користувача.

Сторінка Користувачів зроблена для того щоб адмін зміг добавити нового користувача та редагувати його адмін може добавити йому пароль, користувач має змогу потім змінити пароль також сторінка має пошук (5) по користувачам.

Отже, в даному підрозділі було розглянуто структуру графічного інтерфейсу користувача для створюваного додатку. Надано базові відомості про GUI, наведено особливості та вимоги при розробці інтерфейсу для програмного забезпечення для управління конфігураціями. Розглянуто й детально описано структурну схему головного вікна додатку, наведено структурні схеми для окремих вкладок. Описано призначення окремих елементів інтерфейсу при використанні різних функцій програмного продукт.

2.3 Розробка структури інтерфейсу програмного додатку

Після аналізу даних необхідно розробити структуру інтерфейсу програмного додатку. Інтерфейс повинен бути логічним і зрозумілим користувачеві. Його мета полягає в тому, щоб дозволити користувачеві взаємодіяти з програмним продуктом та надати зручний доступ до функцій та можливостей додатку. Основним елементом інтерфейсу є головне меню, яке містить посилання на всі функції додатку. При розробці інтерфейсу необхідно враховувати, що користувач повинен мати можливість легко знайти те, що йому потрібно, тому весь інтерфейс повинен бути зрозумілим і зручним у використанні [18].

Головна сторінка: головна сторінка веб-системи містить інформацію про Клієнтів, користувачів та проєктів. Макет головної сторінки зображено на рисунку 2.5.

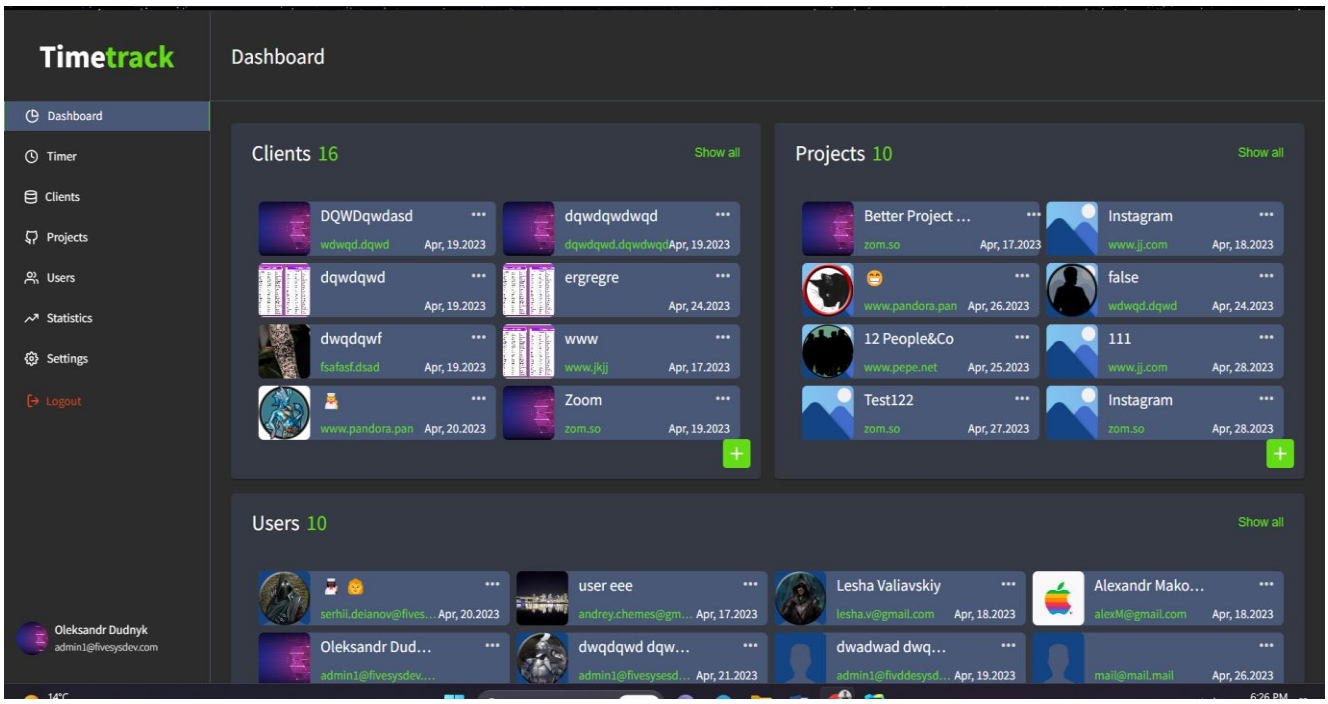

Рисунок 2.5 – Макет головної сторінки

Сторінка таймер: таймер містить інформацію про користувачів та їхні витрачані часи на проєкті .

На сторінці "Таймер" графічного інтерфейсу користувача надається інформація про користувачів та їх витрачаний час на проєкті. Ця сторінка має наступні елементи інтерфейсу:

- Логотип: це графічний елемент, який ідентифікує програмний продукт або компанію, що розробляє додаток.
- Сайдбар: це бічна панель, яка містить швидкий доступ до основних функцій та навігацію по іншим сторінкам додатку.
- Навігація: це розділ, де користувач може переміщатися між різними сторінками або вкладками програмного додатку.
- Профіль користувача: ця частина інтерфейсу відображає інформацію про користувача, таку як його ім'я, фотографію або інші додаткові дані.
- Заголовок сторінки: цей елемент показує назву сторінки "Таймер", щоб користувач міг легко розпізнати поточний контекст.
- Шапка сторінки: ця частина інтерфейсу містить додаткову інформацію або керуючі елементи, які стосуються сторінки "Таймер".

- Календар: цей елемент дозволяє користувачеві вибрати конкретну дату, для якої він хоче переглянути витрачений час на проєкті.
- Таблиця з даними про витрачений час: це основний компонент сторінки "Таймер". Вона містить інформацію про користувачів та їх витрачаний час на проєкті. Кожен рядок таблиці представляє окремого користувача і відображає його ім'я та час, який він витратив на проєкт.
- Кнопка запуску/зупинки таймера: цей елемент дозволяє користувачеві почати або зупинити відлік часу на проєкті. Користувач може натиснути на цю кнопку, щоб почати роботу над проєктом і почати відлік часу, а потім натиснути її знову, щоб зупинити відлік, коли закінчить роботу.
- Кнопка скидання таймера: цей елемент дозволяє користувачеві скинути витрачений час на проєкт і розпочати заново.
- Структура сторінки "Таймер" допомагає користувачеві легко відстежувати свій час, витрачений на проєкт, і отримувати необхідну інформацію для подальшого аналізу та управління. Користувач може бачити загальний час, який він витратив, а також порівнювати свій внесок з іншими учасниками проєкту.

Ця сторінка забезпечує зручну та інформативну взаємодію з додатком, дозволяючи користувачеві точно виміряти свій час та ефективно керувати своїми робочими завданнями на рисунку 2.6.

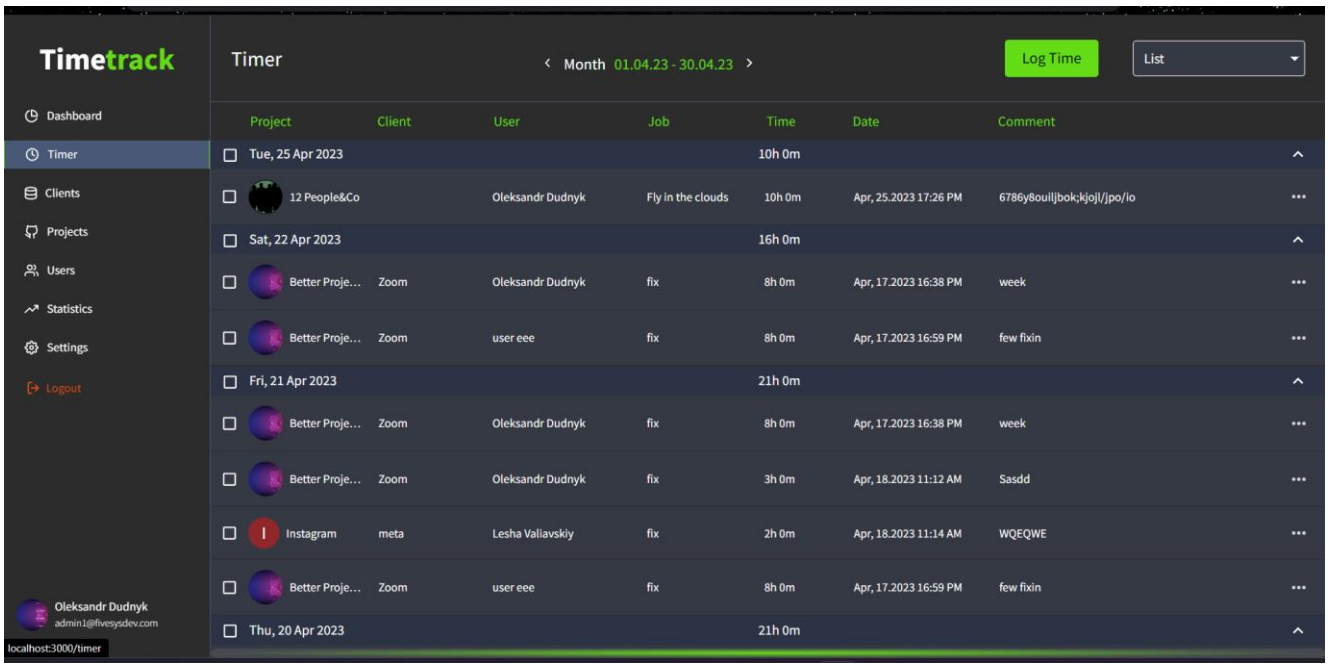

Рисунок 2.6 – Макет Time page

Навігаційні елементи: для зручності користувача інтерфейс може містити навігаційні елементи, такі як кнопки домашня сторінка, каталог, пошук, фільтри. Інтерфейс Tracker time. На сторінці "Таймер" можуть бути додані додаткові навігаційні елементи для забезпечення більшої зручності користувача. Ось кілька прикладів навігаційних елементів, які можуть бути включені в інтерфейс Tracker time:

- Кнопка "Домашня сторінка": ця кнопка перенесе користувача на головну сторінку додатку або панель управління, де він може отримати доступ до інших функцій та інформації.
- Кнопка "Каталог": цей елемент навігації дозволяє користувачеві переглядати список доступних проєктів або завдань. Він може бути корисним, якщо користувач працює над кількома проєктами і хоче швидко перемикатися між ними.
- Кнопка "Пошук": цей навігаційний елемент дозволяє користувачеві здійснювати пошук за різними параметрами, наприклад, за іменем користувача або за конкретною датою. Він полегшує знаходження певних даних або записів про витрачений час.

- Елемент "Фільтри": цей елемент дозволяє користувачеві застосовувати фільтри до таблиці з даними про витрачений час. Він може дозволяти користувачеві відфільтровувати записи за певними критеріями, такими як часовий діапазон, категорія або тип завдання [19].

Ці навігаційні елементи спрощують взаємодію користувача зі сторінкою "Таймер" і допомагають йому знаходити необхідну інформацію швидко і зручно. Вони додають більшу функціональність інтерфейсу і поліпшують загальний досвід користувача.

У програмі також передбачено можливість фільтрації статистики за різними параметрами, такими як користувачі, проекти, часові проміжки тощо в календарику рисунок 2.7. Користувач може вибрати одного або кількох користувачів або проектів і отримати статистику, що стосується саме цих обраних об'єктів [20].

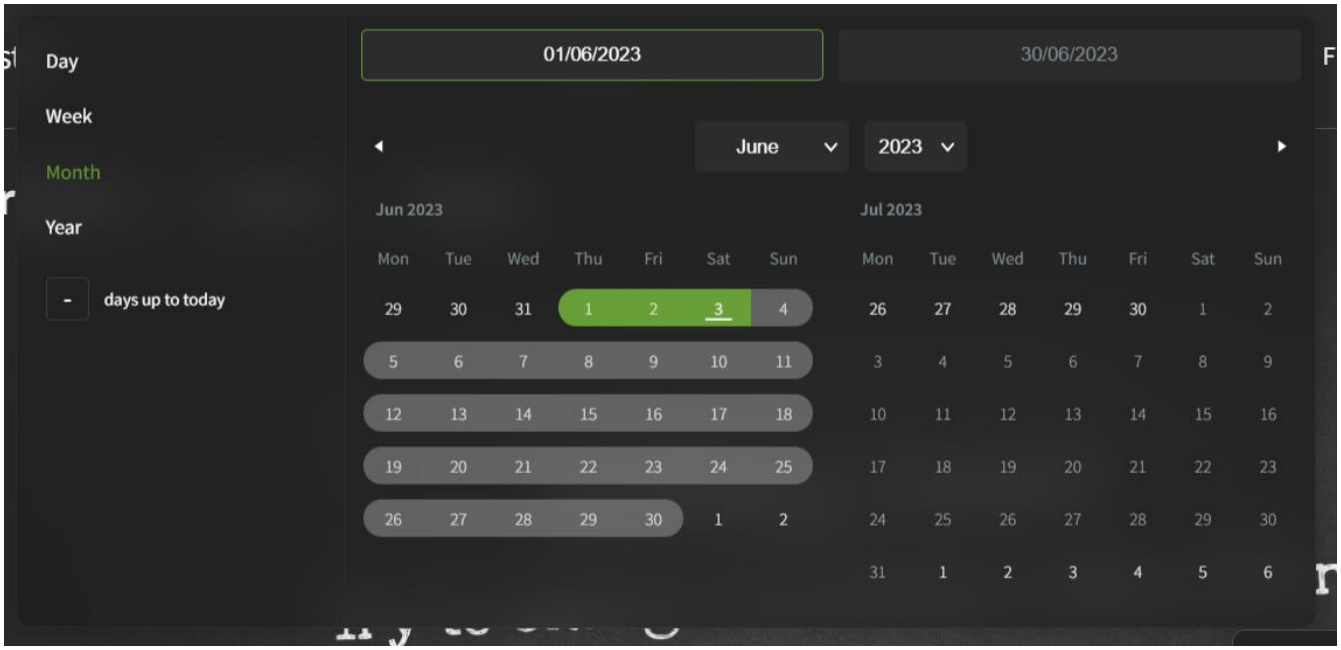

Рисунок 2.7 – Календарик

Також в програмі доступна функція завантаження статистичних даних у форматі Excel, що дозволяє зручно зберігати та обробляти ці дані в поза програмному середовищі. Це надає більші можливості для аналізу та

подальшого використання статистичних даних, наприклад, для підготовки звітів або подальшого дослідження.

Всі ці функціональні можливості роблять програму зручним інструментом для аналізу статистики проєктів та управління робочими годинами, дозволяючи збільшити продуктивність та ефективність роботи команди.

За допомогою модального вікна користувач може отримати детальну інформацію про обраний проект. Наприклад, відображення кількості витрачених годин на кожного користувача в рамках даного проекту, загальна тривалість проекту, список завдань та їх стан (виконано, невиконано, в процесі) тощо. Це дозволяє керівникам та учасникам проекту аналізувати та оцінювати продуктивність та ефективність роботи рисунок 2.8.

Розробка графічного інтерфейсу Web-системи є важливим етапом в процесі створення зручного та привабливого користувацького досвіду. У даному випадку, розробка графічного інтерфейсу проводилася у графічному редакторі Figma. Figma є потужним інструментом, що дозволяє дизайнерам створювати та прототипувати інтерфейси веб-додатків та мобільних додатків.

Figma надає можливість створення інтерактивних прототипів, де можна відтворити роботу різних функцій та навігації в системі [21]. Це дозволяє дизайнерам та розробникам зрозуміти, як буде виглядати та працювати кінцевий продукт. Приклад зображення інтерфейсу наведено на рисунку 2.8.

Приклад зображення інтерфейсу сторінки статистики наведено на рисунку 2.9.

Після розробки графічного інтерфейсу в Figma, наступним кроком було реалізувати його в реакті, використовуючи UI бібліотеку Material UI. Material UI є популярною бібліотекою компонентів, яка забезпечує готові елементи дизайну згідно з принципами Material Design, розробленими компанією Google.

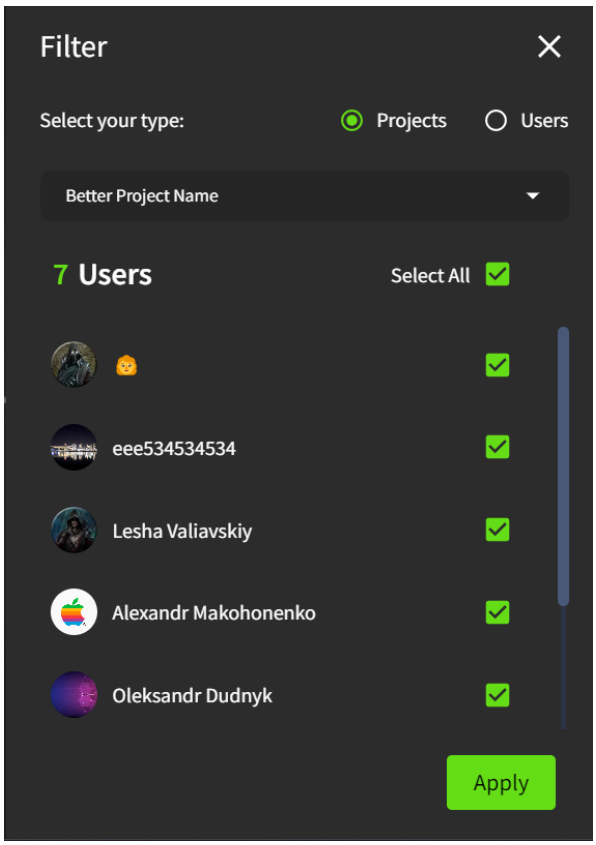

Рисунок 2.8 – Фільтри

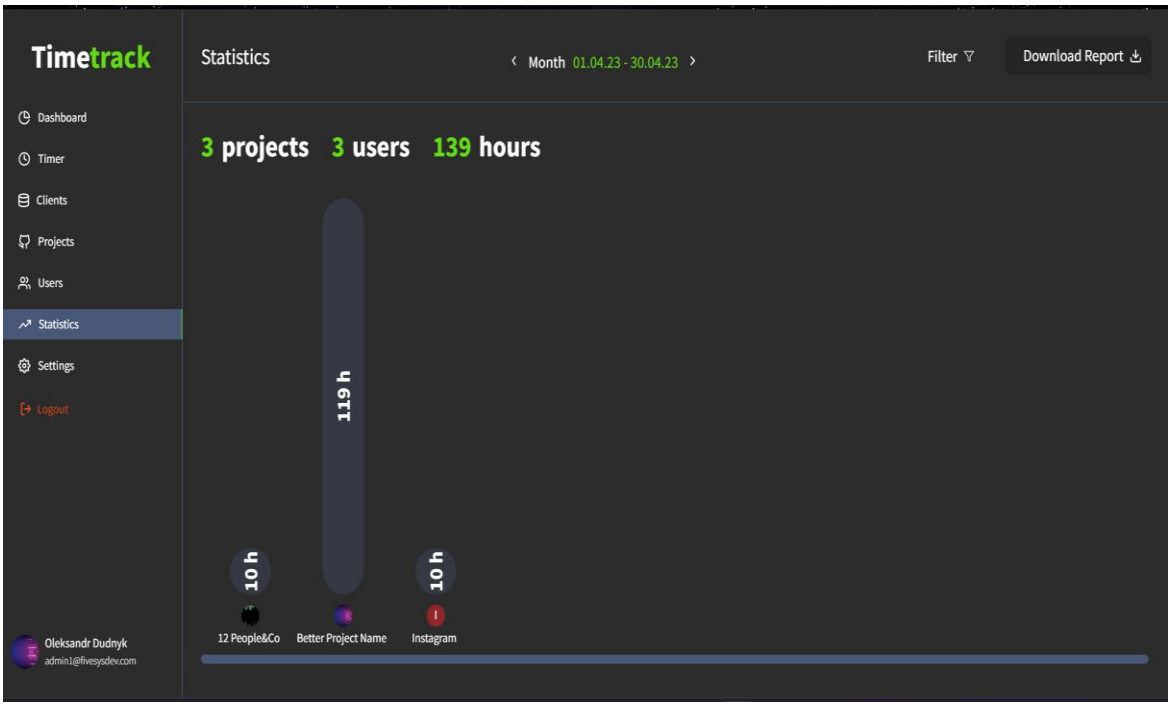

Рисунок 2.9 – Приклад зображення інтерфейсу

Material UI надає широкий вибір готових компонентів, таких як кнопки, форми, таблиці, меню та багато інших. Використання цієї бібліотеки дозволяє ефективно створювати стильний та сучасний інтерфейс, забезпечувати єдність вигляду елементів і полегшувати роботу з розміщенням та стилізацією.

За допомогою Material UI розробники можуть швидко та ефективно створювати компоненти, налаштовувати їх зовнішній вигляд та взаємодію, а також реагувати на події користувача. Це спрощує процес розробки і дозволяє швидше довести продукт до користувачів.

Завдяки поєднанню графічного редактора Figma та бібліотеки Material UI, розробники можуть ефективно створювати та реалізовувати привабливий та функціональний графічний інтерфейс для Web-системи. Це дозволяє забезпечити зручний та приємний користувацький досвід, сприяючи успішному впровадженню системи та задоволенню потреб користувачів.

2.4 Розробка моделі системи

Для кращого розуміння роботи онлайн платформи було вирішено використати модель роботи системи на прикладі UML діаграм. Однією з таких діаграм є діаграма класів, яка відображає основні класи та взаємодії між ними. На рисунку 2.4 представлена діаграма класів, яка ілюструє структуру системи та відношення між класами.

У цій діаграмі присутні три основні класи: "log time" (журнал часу), "User" (користувач) і "Project" (проєкт). Клас "log time" має атрибути, такі як id (ідентифікатор), date (дата), user (користувач), project (проєкт) та time (час). Цей клас відображає інформацію про часові записи, які пов'язані з конкретним користувачем і проєктом.

Клас "User" містить атрибути, які описують користувача системи, такі як ID (ідентифікатор), name (ім'я), email (електронна пошта) та password (пароль). Цей клас відображає дані про користувача, які можуть бути використані для авторизації та ідентифікації.

Клас "Project" представляє проєкт, який пов'язаний з користувачами системи. Він має зв'язок з класом "User", що вказує на те, що кожен проєкт може бути пов'язаний з одним або багатьма користувачами.

Діаграма класів зображена на рисунку 2.10.

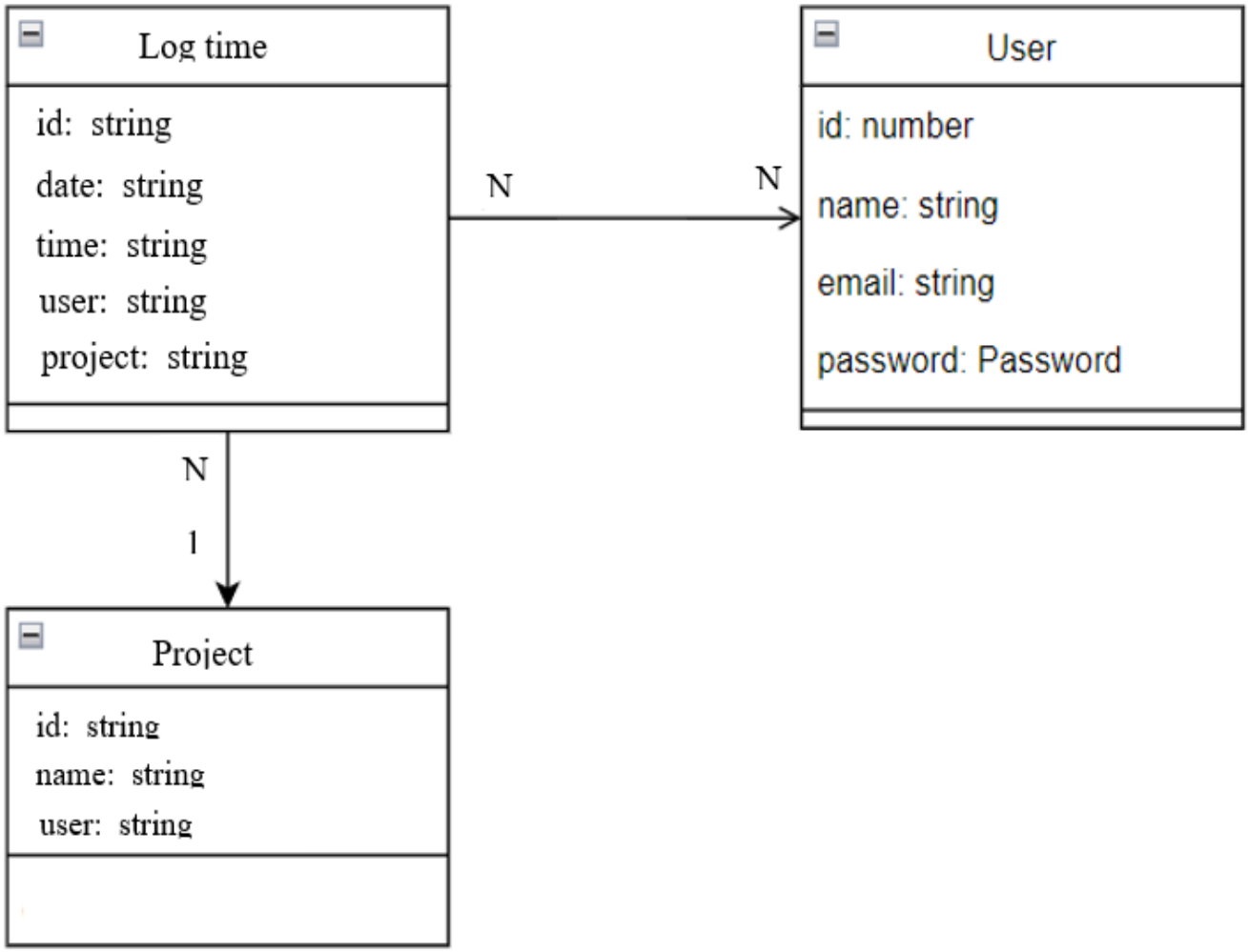

Рисунок 2.10 – Діаграма класів

Ця діаграма класів дозволяє зрозуміти структуру системи та взаємозв'язки між класами. Вона може бути використана як основа для подальшого проектування та розробки системи онлайн платформи, що ураховує час роботи працівників на проєктах.

Використання UML діаграм допомагає уточнити вимоги до системи, визначити її структуру та зв'язки між компонентами. Вона сприяє зрозумінню

системи як для розробників, так і для замовників проекту, що полегшує комунікацію та співпрацю між різними сторонами.

Дана діаграма класів є лише однією з можливих інтерпретацій структури системи і може бути доповнена або змінена залежно від конкретних потреб проекту та деталей його реалізації.

Діаграма діяльності, зображена на рисунку 2.11, надає візуальне уявлення про взаємодію користувача з сайтом при логуванні свого часу на проєкті. Ця діаграма допомагає краще зрозуміти послідовність кроків, які виконує користувач для фіксації своєї робочої часу.

На початку процесу користувач відкриває сайт і переходить на сторінку, де доступна функціональність логування робочого часу. Наступним кроком є введення обраних користувачем авторизаційних даних, таких як ім'я користувача та пароль. Після введення цих даних користувач натискає кнопку "Увійти".

Після успішного входу в систему користувач переходить на сторінку, де можна вибрати проєкт, на якому він працював. Це може бути список доступних проєктів або календарний інтерфейс, де користувач може вибрати конкретну дату та побачити проекти, що були активними у цей день.

Після вибору проєкту користувач може вказати час, який він витратив на цей проєкт. Це може бути введенням кількості годин або хвилин у відповідні поле або за допомогою інтерфейсу, де можна вибрати час з вже встановленими інтервалами.

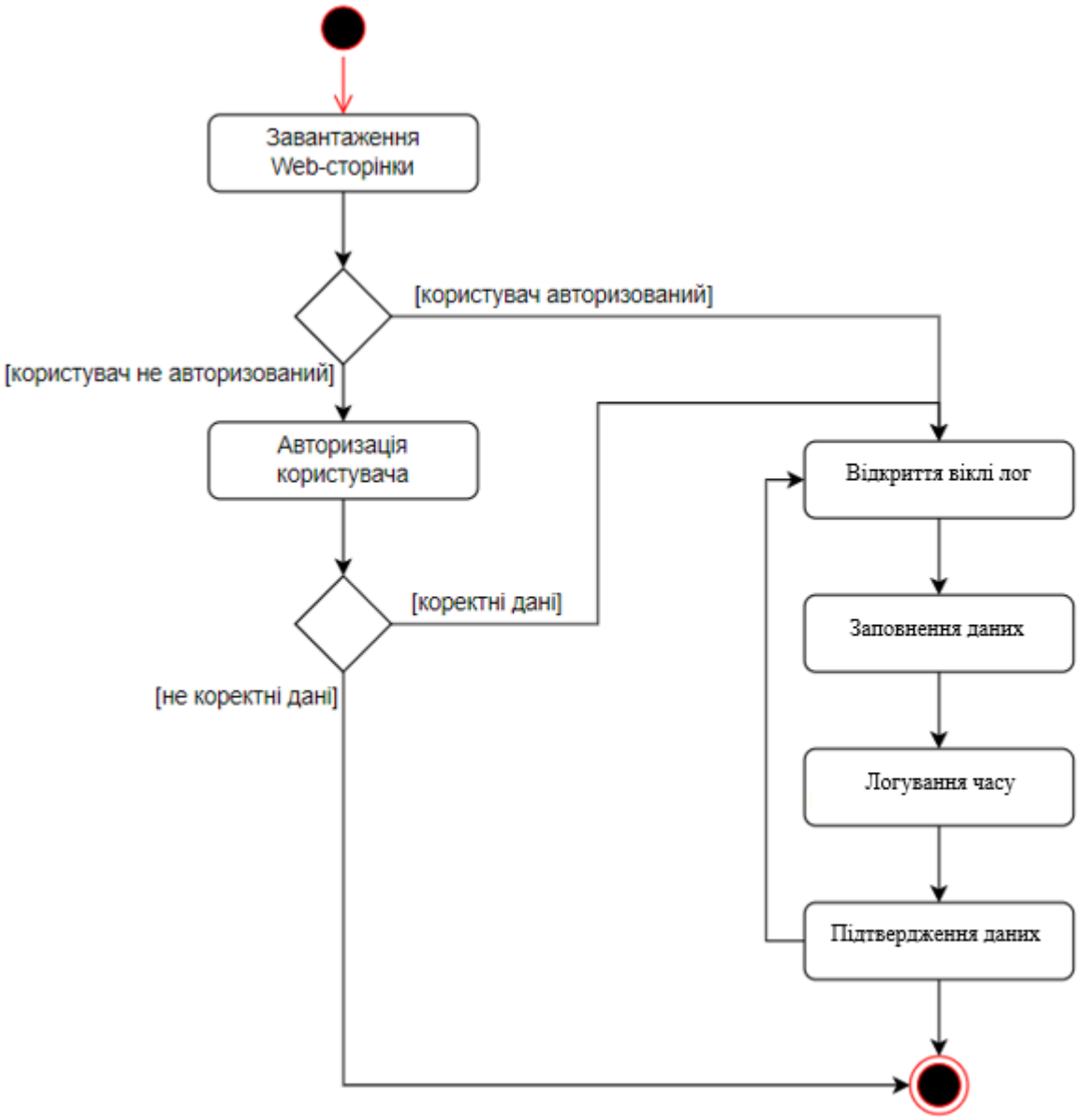

Рисунок 2.11 – Діаграма діяльності

Далі, після вказання часу, користувач може надати додаткову інформацію про свою роботу на проєкті, наприклад, опис завдань або коментарі. Це допомагає зберегти важливу інформацію про проделану роботу та зробити її більш детальною.

Завершальним кроком є натискання кнопки "Зберегти" або "Логувати час", яка підтверджує запис часу на проєкт. Ця інформація потім зберігається в системі та використовується для обчислення статистики та аналізу продуктивності.

Ця діаграма діяльності допомагає візуалізувати весь процес логування робочого часу на проєкті та зрозуміти послідовність кроків, які користувач повинен здійснити для успішного внесення цих даних. Вона сприяє зручності в роботі зі системою та ефективному відстеженню робочих годин на проєктах.

2.5 Розробка алгоритмів роботи додатку

Для створення ефективної системи **TimeTrack**, яка забезпечує точне та зручне логування робочого часу, необхідно розробити детальний алгоритм роботи. Він визначає взаємодію користувача із системою, послідовність дій, а також відповідає за обробку даних. Алгоритм поділяється на кілька ключових етапів.

Основні етапи алгоритму

1. **Ініціалізація системи:**
    - Користувач відкриває сайт або запускає мобільний додаток TimeTrack.
    - Відбувається завантаження інтерфейсу, який перевіряє наявність активної сесії.
    - Якщо сесія активна, користувача автоматично перенаправляють до головної сторінки.
2. **Авторизація/Реєстрація:**
    - У разі відсутності активної сесії користувач проходить процедуру авторизації.
    - Для нових користувачів передбачена реєстрація, під час якої необхідно вказати персональні дані: ім'я, електронну пошту, пароль.
    - Після успішного входу система вітає користувача та надає доступ до функціоналу.
3. **Робота з проектами:**

- Користувач може створювати нові проекти, вибирати існуючі або переглядати загальний список.
- Для кожного проекту доступні такі функції:
  - додавання задач,
  - встановлення пріоритетів,
  - визначення дедлайнів.

4. **Логування часу:**
   - Користувач обирає проект із доступного списку.
   - Зазначає тривалість роботи (час початку та завершення) або загальну кількість годин.
   - За потреби додає коментарі, що описують виконані завдання або проблеми, які виникли під час роботи.
5. **Підтвердження даних:**
   - Після введення часу натискається кнопка "Зберегти" або "Логувати час".
   - Система перевіряє коректність введених даних, запобігаючи дублюванню або помилковому запису.
   - Дані записуються до бази, а користувачу надається повідомлення про успішне збереження.
6. **Аналіз даних:**
   - Система проводить обчислення, оновлює статистику для проектів та окремих користувачів.
   - Доступні функції:
     - перегляд загальної кількості годин,
     - аналіз продуктивності за проектами,
     - створення звітів у форматах PDF чи Excel.
7. **Редагування та коригування:**
   - Користувач має можливість редагувати внесені записи, видаляти некоректні дані або додавати нову інформацію.
   - Всі зміни автоматично відображаються у звітах та статистиці.

8. **Завершення роботи:**
    - По завершенні роботи з додатком користувач може вийти із системи або залишити її працювати у фоновому режимі.

Особливості алгоритму

Алгоритм роботи враховує необхідність підтримки багатокористувацького режиму та забезпечує зручність роботи з великими обсягами даних. Для інтерактивності та швидкої роботи передбачено використання AJAX-запитів, які мінімізують час завантаження сторінок.

На рисунку 2.12 подано схематичне представлення основних етапів роботи додатку TimeTrack у вигляді блок-схеми.

Загалом, розроблений алгоритм забезпечує зручну та логічну послідовність дій для користувачів системи TimeTrack при логуванні їх робочого часу на проєктах. Цей алгоритм сприяє ефективному використанню системи та полегшує процес відстеження та аналізу робочих годин.

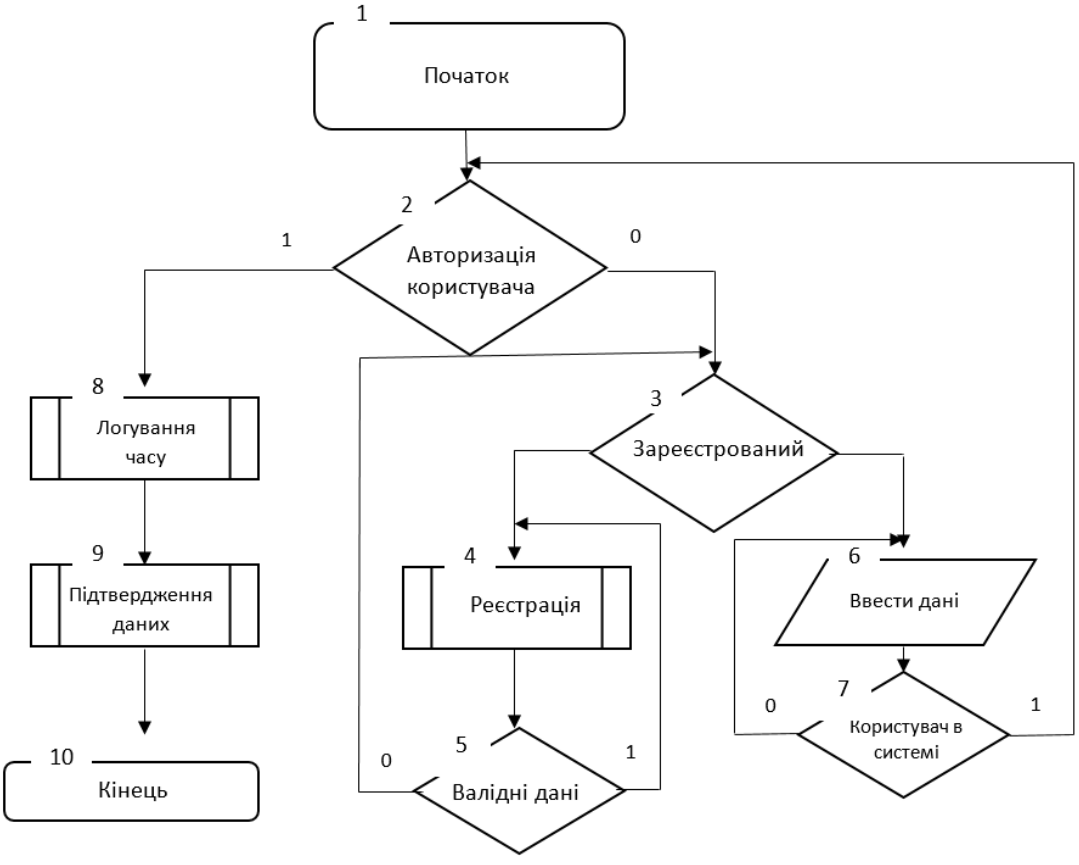

Рисунок 2.12 – Загальний алгоритм додатку

2.6 Висновок до розділу 2

У цьому розділі було детально проаналізовано процеси, пов’язані зі створенням та функціонуванням системи TimeTrack, яка спрямована на автоматизацію логування робочого часу та управління проектами. Особлива увага приділялась розробці вхідних та вихідних даних, які є ключовими елементами для забезпечення точності та ефективності системи.

Було визначено, що для коректного функціонування системи необхідно ретельно опрацювати логіку роботи з даними. Вхідні дані включають інформацію про користувачів, проекти, задачі та робочі години, тоді як вихідні дані мають надавати користувачам звіти, статистику, аналітику та інші підсумкові результати. Забезпечення цілісності даних і запобігання дублюванню є важливими аспектами роботи системи.

Розробка макета та дизайну платформи стала одним із основних етапів. Вона включала створення зручного, інтуїтивно зрозумілого інтерфейсу, який відповідає сучасним вимогам до UX/UI-дизайну. Особливий акцент зроблено на простоті взаємодії користувача із системою, адже зручність та доступність є важливими факторами для підвищення продуктивності.

3 РОЗРОБКА МОДУЛЯ ГОЛОВНОЇ СТОРІНКИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Ефективна розробка запланованої Web-системи напряму залежить від обраного засобу реалізації: від обраної мови програмування, від обраного середовища розробки.

Серед можливих мов розробки розглянемо Python, JavaScript, PHP.

PHP – скриптова мова програмування для веб-розробки на серверному боці.

Переваги: Висока поширеність, вбудована підтримка веб-технологій, простота вивчення.

Недоліки: Обмежені можливості поза веб-розробкою, застарілі підходи.

JavaScript– скриптова мова програмування для веб-розробки на стороні клієнта.

Переваги: Універсальність, можливість взаємодії з користувачем на стороні клієнта, велика спільнота розробників.

Недоліки: Залежність від браузера, безпекові ризики.

Python– універсальна скриптова мова програмування з великою кількістю бібліотек.

Переваги: Простота синтаксису, широкі можливості застосування, велика кількість бібліотек.

Недоліки: Менша продуктивність порівняно з іншими мовами, не є найкращим вибором для веб-розробки на клієнтському боці.

Щодо обґрунтування вибору використання JavaScript для реалізації веб-системи. Клієнтська сторона: JavaScript використовується для розробки клієнтської частини веб-додатків, що виконується безпосередньо на браузері користувача. Це дозволяє створювати веб-системи, які можуть бути доступні з

різних пристроїв та платформ, включаючи комп'ютери, планшети та мобільні пристрої.

Широкий вибір фреймворків та бібліотек: JavaScript має велику кількість фреймворків та бібліотек, таких як React, Angular, Vue.js та інші, які спрощують розробку веб-інтерфейсу та взаємодії з користувачем.

Асинхронність: JavaScript має вбудовану підтримку асинхронного програмування, що дозволяє реалізовувати ефективні веб-системи, які взаємодіють з сервером, отримуючи та обробляючи дані асинхронним чином.

Розширені можливості: JavaScript є мовою програмування з багатьма розширеними можливостями, такими як маніпулювання DOM (Document Object Model), взаємодія з API серверу, можливості геолокації, анімації, робота з камерою та іншими веб-технологіями. Ці можливості дозволяють створювати веб-систему для перегляду з різноманітними функціями, такими як реалізація 3D-графіки, взаємодія з користувачем, адаптація до різних розмірів екрану та інші [22].

Підтримка браузерами: JavaScript є однією з основних мов, які підтримуються браузерами, що дозволяє виконувати веб-додатки на різних браузерах без необхідності встановлення додаткових плагінів чи розширень.

Отже, як основну мову було обрано Java Script, враховуючи порівняно високу продуктивність.

Серед середовищ розробки можна виділити Visual Studio Code, WebStorm, PhpStorm. Розглянемо дані середовища.

Visual Studio Code– це безкоштовний відкритий редактор коду, розроблений Microsoft, який став дуже популярним серед розробників веб-застосунків на JS. VS Code має велику кількість розширень, які полегшують розробку на JS, такі як підсвічування синтаксису, автодоповнення, відладка, інтеграція з системами контролю версій, тестування та інші. Він має також зручний інтерфейс та можливості розширення функціональності.

WebStorm – це інтегроване середовище розробки (IDE), розроблене компанією JetBrains, спеціалізоване на розробці веб-застосунків на JS.

WebStorm має розширені функції, такі як відступи, підсвічування синтаксису, автодоповнення, відладка, вбудовані інструменти для тестування та інтеграцію з популярними фреймворками для розробки на JS, такими як React, Angular, Vue та інші.

PhpStorm – це інша продукт компанії JetBrains, який спеціалізується на розробці веб-застосунків на PHP, але також має велику підтримку для розробки на JS. PhpStorm надає можливості, такі як відступи, підсвічування синтаксису, автодоповнення, відладка, вбудовані інструменти для тестування, інтеграцію з системами контролю версій та підтримку популярних фреймворків, таких як React, Angular, Vue та інші.

Отже, середовищем для розробки було обрано Visual Studio Code, оскільки середовище підтримується компанією Microsoft, а також має багато корисних функцій, таких як відступи, автодоповнення, відладка, інтеграція з системами контролю версій та інші, що допоможуть зробити розробку більш зручною та ефективною. Іншими причинами вибору середовища є безкоштовна версія та широкий функціонал [23]

.

Отже, вибраними засобами реалізації модуля є Java Script та Visual Studio Code.

3.2 Розробка програмного модуля

Модуль Timer є ключовим елементом системи відстеження робочого часу, забезпечуючи широкий функціонал для працівників та адміністраторів. Його основна мета — спростити процес реєстрації робочого часу та зробити його максимально зручним і інтуїтивно зрозумілим.

Основні можливості модуля

Цей компонент надає користувачам можливість:

- Логувати робочий час із зазначенням виконаної роботи, дати, коли вона була здійснена, та інших деталей, таких як:

- Поле "Job" — короткий опис типу виконаної роботи, наприклад, *"Рефакторинг коду"*, *"Доопрацювання дизайну"*, або *"Написання документації"*.
- Коментарі — детальний опис виконаних завдань, що надає контекст для адміністраторів або колег.

- Реєструвати дані як для одного дня, так і для довших періодів, наприклад, у тижневому форматі.

Особливістю модуля є вкладка "Weekly Log", що дозволяє швидко внести дані про робочий час за цілий тиждень. Ця функція є незамінною для працівників, які виконують багато завдань і потребують інструменту для зручного агрегування даних. Щоб зберегти внесені записи, достатньо натиснути кнопку "Apply", після чого дані будуть автоматично додані до системи.

Зручність використання та інтерфейс

На рисунку 3.1 представлено інтерфейс сторінки, де працівники можуть вводити дані про робочий час.

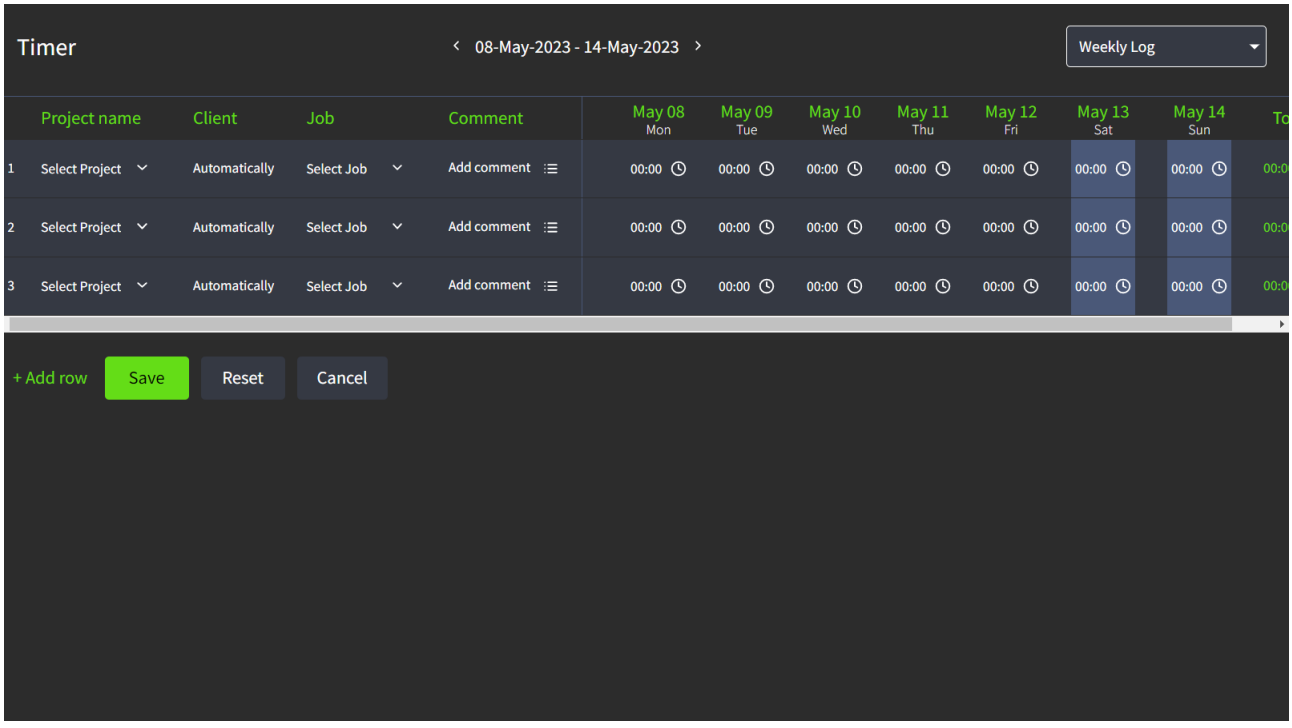

Рисунок 3.1 – Weekly log

Сторінка містить:

- Вкладки навігації, які дозволяють легко перемикатися між щоденними та тижневими логами, а також іншими розділами системи.
- Форми для введення даних, що мають мінімалістичний, інтуїтивно зрозумілий дизайн із підказками, що спрощує роботу навіть для новачків.

Модуль також передбачає механізми перевірки коректності введених даних, попереджаючи користувачів про помилки та надаючи можливість їх виправлення ще до збереження.

Переваги для адміністраторів

Модуль Timer забезпечує не лише зручність для працівників, але й надає адміністраторам корисний інструмент для моніторингу:

- Аналіз динаміки роботи — адміністратори можуть переглядати загальний прогрес по проєктах, оцінювати ефективність роботи кожного працівника або команди загалом.
- Прозорість даних — кожен запис супроводжується коментарями, що дозволяє зрозуміти, на що витрачався робочий час.
- Експорт звітів — отримані дані легко перетворити у формати, зручні для аналізу, наприклад, у файли Excel чи PDF.

Вплив на продуктивність команди

Завдяки інтуїтивно зрозумілому інтерфейсу та широкому набору функцій, модуль сприяє підвищенню ефективності команди:

- Мінімізація помилок у логуванні часу завдяки підказкам і перевіркам.
- Швидкість реєстрації даних, що звільняє час для виконання основних робочих завдань.
- Полегшення комунікації між працівниками та адміністраторами через зрозумілі й деталізовані записи.

Узагальнення

Отже, модуль Timer є потужним інструментом для реєстрації, обробки та аналізу робочого часу. Його впровадження в систему дозволяє:

1. Спрощувати процес логування часу для працівників, навіть якщо вони працюють із великими обсягами даних.

2. Забезпечувати гнучкість використання завдяки підтримці як щоденного, так і тижневого форматів роботи.

На рисунку 3.2 представлено код сторінок програми.

```
export const MainRouts = () => {
  const { isAuthorised, role } = useAppSelector(authSelector);

  return (
    <Routes>
      <Route element={<PrivateRoutes />}>
        <Route path="*" element={<Page404 />} />

        <Route path="/" element={<MainLayout />}>
          <Route
            index
            element={
              role === ADMIN ? (
                <Navigate to={ROUTE.privateFieldsAdmin.dashboard} />
              ) : (
                <Navigate to={ROUTE.base.timer} />
              )
            }
          />
          <Route path="/" element={<PrivateRoutesAdmin />}>
            <Route path={ROUTE.privateFieldsAdmin.dashboard} element={<Dashboard />} />
            <Route path={ROUTE.privateFieldsAdmin.clients} element={<Clients />} />
            <Route path={ROUTE.privateFieldsAdmin.projects} element={<Projects />} />
            <Route path={ROUTE.privateFieldsAdmin.users} element={<Users />} />
            <Route path={ROUTE.privateFieldsAdmin.statistics} element={<Statistics />} />
          </Route>
          <Route path={ROUTE.base.timer} element={<Timer />} />
          <Route path={ROUTE.base.settings} element={<Settings />} />
        </Route>
      </Route>
      {!isAuthorised && <Route path={ROUTE.signIn} element={<Login />} />}
    </Routes>
  );
};
```

Рисунок 3.2 – Код головного модуля

На рисунку 3.3 представлена сторінка таймера, яка надає можливість переглядати, коригувати та створювати логи робочого часу. Ця сторінка містить різні функціональні елементи, що дозволяють взаємодіяти з даними про робочий час.

Рисунок 3.3 – Сторінка таймер

На сторінці таймера ви можете бачити списки логів робочого часу, які включають інформацію про дату, тривалість роботи та опис виконаної роботи. Кожен запис логу може бути корегований, що дозволяє вам внести зміни до даних про робочий час, якщо потрібно.

Для корегування логу робочого часу ви можете використовувати різні функціональні кнопки або елементи керування, такі як кнопки редагування, видалення або збереження змін. Це дозволяє вам оновлювати та змінювати дані про робочий час зручним способом.

Крім корегування, ви також можете створювати нові записи логів робочого часу. Для цього ви можете використовувати відповідну функціональну кнопку або елемент керування, які запропоновані на сторінці. Після створення нового запису логу ви зможете вказати дату, тривалість роботи та додатковий опис.

Сторінка таймера надає зручність та ефективність при роботі з даними про робочий час. Ви можете легко переглядати, коригувати та створювати логи, що сприяє точному відстеженню робочого часу та полегшує аналіз та управління часом роботи.

На рисунку 3.4 зображено код сторінки таймера. Запропонований код є частиною роботи і відповідає за функціональність таймера в системі. Наведений код написаний мовою програмування JavaScript з використанням бібліотеки React для розробки інтерфейсу користувача.

```
    endDate: endOfMonth(new Date()),
    key: 'selection',
  });
  const [isOpenEditPanel, setIsOpenPanel] = useState<boolean>(false);
  const [isOpenPopup, setIsOpenPopup] = useState<boolean>(false);

  const { data: loggedDaysList = [], isLoading } = useGetLoggedDaysQuery(timerRange);
  const [deleteSeveralLoggedDays] = useDeleteMultipleLoggedDaysMutation();

  const handleOpenPanel = () => setIsOpenPanel(true);
  const handleClosePanel = () => setIsOpenPanel(false);

  const deleteLoggedDaysHandler = () => {
    deleteSeveralLoggedDays({ ids: selected });
    dispatch(setSelected([]));
    closePopupHandler();
  };

  useEffect(() => {
    selected.length && dispatch(setEmptySelected());
  }, [currentMode]);

  const openPopupHandler = () => setIsOpenPopup(true);
  const closePopupHandler = () => setIsOpenPopup(false);

  return (
    <>
      <TimerHeader
        timerRange={timerRange}
        setTimerRange={setTimerRange}
        deleteLoggedDays={openPopupHandler}
        selected={!!selected.length}
      />
      {currentMode === 'list' ? (
        <MonthlyLog
          timerRange={timerRange}
          loggedDaysList={loggedDaysList}
          isLoading={isLoading}
          handleOpenPanel={handleOpenPanel}
          handleClosePanel={handleClosePanel}
          isOpenEditPanel={isOpenEditPanel}
        />
      ) : (
        <WeeklyLog />
      )}
      <DeletePopup
        closePopup={closePopupHandler}
        openPopup={isOpenPopup}
        confirm={deleteLoggedDaysHandler}
        itemType="log item"
      />
    </>
  );
};

```

Рисунок 3.4 – Код сторінки таймер

У даному коді використовуються наступні імпорти:

- useEffect та useState з бібліотеки react для використання ефектів та стану компонентів.
- useDeleteMultipleLoggedDaysMutation та useGetLoggedDaysQuery з store/services/LoggedDaysApi для роботи з API для видалення та отримання даних про зареєстровані дні.
- Range з react-date-range для представлення діапазону дат.
- DeletePopup з components для відображення підтвердження видалення.
- TimerHeader з pages для відображення заголовка таймера.
- WeeklyLog та MonthlyLog для відображення щотижневого та щомісячного журналів відповідно.
- useAppDispatch та useAppSelector з hooks для отримання функції диспетчера та вибору стану зі збереженого магазину.
- timerSelector з store/selectors для вибору поточного режиму таймера.
- setEmptySelected та setSelected з store/timer/timerSlice для зміни вибраних елементів у магазині.
- endOfMonth та startOfMonth з date-fns для отримання початку та кінця поточного місяця.

Компонент Timer є основним компонентом цієї частини системи. Він містить логіку для відображення інформації на проєктах, включаючи вибір діапазону дат, отримання та видалення зареєстрованих логів, керування відкриттям та закриттям панелі редагування та підтвердження видалення. Залежно від поточного режиму таймера відображається щотижневий або щомісячний журнал. Компонент Timer також використовує інші компоненти, такі як TimerHeader, MonthlyLog, WeeklyLog та DeletePopup, для відображення різних частин інтерфейсу та взаємодії з користувачем. TimerHeader відповідає за відображення заголовка таймера та обробку дій користувача, таких як вибір діапазону дат або видалення зареєстрованих днів. MonthlyLog та WeeklyLog відповідають за відображення зареєстрованих днів в місячному або тижневому

форматі відповідно. DeletePopup відображає підтверджувальне вікно для видалення зареєстрованих днів.

Цей код показує, як компоненти взаємодіють між собою та як вони використовуються для забезпечення функціональності системи відстеження робочого часу працівників. Він також демонструє використання зовнішніх бібліотек та роботу зі станом компонентів для забезпечення коректної роботи системи.

3.3 Використання штучного інтелекту для автоматичного обліку робочого часу

Автоматизація обліку робочого часу є одним із ключових напрямків сучасних інформаційних технологій, який спрямований на підвищення ефективності роботи співробітників, зменшення впливу людського фактору та оптимізацію бізнес-процесів. Впровадження штучного інтелекту (ШІ) у цю сферу відкриває нові можливості для аналізу поведінки користувачів, покращення продуктивності та автоматичного створення детальних звітів [24].

Основні можливості ШІ-систем для обліку робочого часу:

1. Відстеження дій у реальному часі. ШІ-системи здатні відстежувати активність користувачів у реальному часі, фіксуючи запуск додатків, взаємодію з ними, періоди бездіяльності тощо. Наприклад, система може визначити, скільки часу працівник витрачає на роботу з професійними інструментами, а скільки — на сторонні задачі або соціальні мережі. Цей підхід забезпечує прозорість використання робочого часу, дозволяючи швидко виявляти проблемні зони.

2. Аналіз продуктивності. Використання алгоритмів машинного навчання відкриває можливість автоматичного аналізу виконаних задач. Система здатна:

- класифікувати завдання за рівнем складності;
- оцінювати середній час, необхідний для їх виконання;

- надавати рекомендації для оптимізації робочого процесу. Це допомагає як працівникам, так і керівникам знаходити шляхи для покращення ефективності.

3. Автоматичне логування часу. Завдяки інтелектуальним алгоритмам, потреба в ручному введенні даних відпадає. ШІ самостійно формує логи, що містять детальну інформацію про те, як було розподілено робочий час протягом дня. Така функція не лише економить час співробітників, але й мінімізує ймовірність помилок або навмисного спотворення даних.

4. Інтелектуальні звіти. Сучасні ШІ-системи дозволяють автоматично створювати детальні звіти з такими елементами:

- графіки використання часу;
- показники ефективності;
- аналіз завантаження працівників.

Це дає менеджерам змогу оцінювати результати роботи команди в реальному часі, виявляти зони для вдосконалення та приймати обґрунтовані управлінські рішення.

Переваги використання ШІ для автоматизації обліку часу:

- Зменшення навантаження на працівників: системи автоматично виконують рутинні операції, такі як фіксація часу та формування звітів.
- Прозорість процесів: відсутність суб’єктивного впливу дозволяє об'єктивно оцінювати продуктивність.
- Підвищення ефективності управління: керівники отримують доступ до аналітики, що допомагає оптимізувати робочі процеси.

Візуалізація системи

На рисунку нижче зображено інтерфейс сучасної системи обліку часу з інтеграцією штучного інтелекту. Інтерфейс подібний до класичних програм, проте доповнений інструментами для аналітики та прогнозування.

Такі інноваційні рішення стають незамінними для підприємств, які прагнуть залишатися конкурентоспроможними в умовах цифрової трансформації.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 3.1.

Таблиця 3.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу інформаційної технології.

Критерії	Експерти		
	Hubstaff	Harvest	TimeTracker
Точність	2	3	3
Можливість офлайн роботи	2	3	3
Ai	0	0	3
Ціна	3	2	3
Ринкові переваги (експлуатаційні витрати)	3	2	3

За результатами розрахунків, наведених в таблиці 3.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології – вище середнього.

3.4 Висновок до розділу 3

У третьому розділі здійснено глибокий аналіз вибору технологій та інструментів для розробки модуля Timer, який є ключовим компонентом системи відстеження робочого часу. Цей аналіз охоплював різні аспекти, включаючи функціональність, зручність використання, продуктивність, підтримку сучасних стандартів розробки та інтеграцію з іншими компонентами системи.

Особливу увагу приділено компоненту Timer, який виконує критично важливу роль у відображенні інформації про робочі записи. Він реалізує логіку вибору діапазонів дат, отримання та видалення логів, а також керування функціями редагування та видалення записів. Залежно від вибраного режиму

(щотижневий або щомісячний), компонент відображає журнал із відповідною деталізацією.

Розробка компонента базується на розумній модульності. Використання підкомпонентів, таких як TimerHeader, MonthlyLog, WeeklyLog і DeletePopup, забезпечує чіткий розподіл функціональності між елементами системи:

- TimerHeader відповідає за заголовок таймера та обробку дій користувача, таких як вибір діапазону дат або видалення записів.
- MonthlyLog і WeeklyLog відображають інформацію про логування у форматі місячного або тижневого журналу відповідно, що забезпечує гнучкість у використанні системи.
- DeletePopup додає підтверджувальне вікно для забезпечення безпеки при видаленні даних, мінімізуючи ризик випадкового видалення інформації.

Крім того, реалізація компонента використовує сучасні підходи до роботи зі станом додатку, зовнішні бібліотеки для підвищення ефективності та надійності, а також забезпечує коректну взаємодію між частинами системи. Це дозволяє уникати конфліктів і забезпечує гладкий досвід для користувача.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування онлайн платформи

Тестування веб-платформи є одним із ключових етапів процесу розробки, адже саме воно гарантує високу якість роботи продукту, забезпечує стабільність його функціонування та відповідність технічним і користувацьким вимогам. Це дозволяє виявити можливі дефекти, помилки чи неефективності в роботі додатку ще до моменту його впровадження у виробниче середовище.

Мета тестування

Основними завданнями тестування є:

- Перевірка функціональності веб-додатку, щоб переконатися, що всі компоненти працюють відповідно до специфікації.
- Забезпечення надійності та коректності роботи навіть у разі значного навантаження або помилкових дій користувачів.
- Оцінка швидкості завантаження сторінок та оптимізація продуктивності.
- Аналіз користувацького досвіду (UX), щоб гарантувати інтуїтивність і простоту використання платформи.
- Тестування сумісності з різними пристроями, браузерами та операційними системами.
- Перевірка безпеки, щоб забезпечити захист даних користувачів та захист від можливих атак.

Етапи тестування

1. Перевірка швидкості завантаження веб-сторінок:

- Швидкість завантаження веб-додатків є важливим показником, оскільки вона безпосередньо впливає на користувацький досвід. Для тестування швидкості використовуються вбудовані інструменти браузера, такі як DevTools, а також спеціалізовані сервіси, наприклад, Google PageSpeed Insights [25].

- Як приклад, було протестовано сторінку зі списком вакансій. Загальний час завантаження становив 1.48 секунди, що є чудовим результатом, враховуючи сучасні стандарти швидкодії. Це вказує на високу оптимізацію ресурсу та відсутність критичних вузьких місць у коді або серверних запитах.

2. Функціональне тестування:

- Всі ключові функції веб-платформи, зокрема реєстрація користувачів, авторизація, додавання, редагування й видалення записів, були перевірені на коректність виконання.
- Для забезпечення покриття всіх сценаріїв використовувалися як ручні методи тестування, так і автоматизовані тести з використанням бібліотек, таких як Selenium чи Cypress.

3. Тестування сумісності:

- Перевірка роботи платформи на різних браузерах (Google Chrome, Mozilla Firefox, Safari, Edge) та пристроях (настільні комп’ютери, планшети, смартфони).
- Забезпечено адаптивність дизайну для коректного відображення елементів інтерфейсу незалежно від розміру екрана.

4. Тестування безпеки:

- Перевірка платформи на вразливості, такі як SQL Injection, XSS (міжсайтовий скриптинг), CSRF (підробка міжсайтових запитів) та інші поширені атаки.
- Використання спеціалізованих інструментів, таких як OWASP ZAP або Burp Suite, для пошуку вразливостей у веб-додатку.

5. Тестування продуктивності:

- Проведено навантажувальні тести для визначення стійкості системи під час значної кількості одночасних запитів. Це дозволило оцінити, як система реагує на пікові навантаження, та знайти потенційні точки покращення.
- Оптимізація часу виконання запитів до сервера і зменшення обсягу переданих даних шляхом використання кешування та мінімізації скриптів.

6. Тестування UX/UI:

- Оцінка зручності використання інтерфейсу та доступності основних функцій.
- Проведено опитування серед тестової групи користувачів для отримання зворотного зв’язку щодо зручності навігації, зрозумілості форм і швидкості виконання дій.

Результати тестування

На основі проведеного тестування виявлено такі результати:

- Веб-додаток демонструє високу швидкодію, час завантаження ключових сторінок знаходиться в межах 1–2 секунд, що відповідає сучасним стандартам.
- Функціональність системи працює коректно, жодних критичних помилок чи збоїв не виявлено.
- Веб-платформа успішно проходить перевірку на сумісність із різними браузерами та пристроями, а також відповідає вимогам адаптивного дизайну.
- Безпека системи забезпечена, вразливостей, які б становили значну загрозу, не знайдено.

Приклад завантаження сторінки наведено на рисунку 4.1.

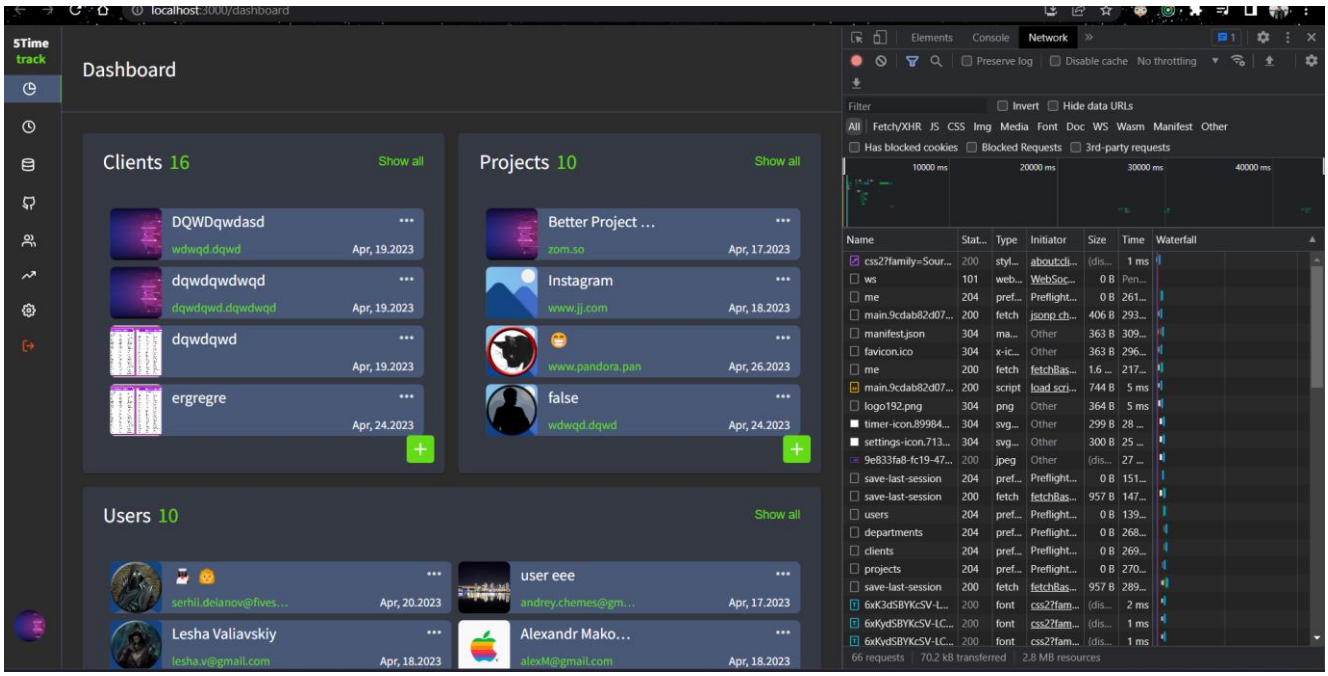

Рисунок 4.1 – Завантаження сторінки

Під час тестування сторінки реєстрації користувача було виконано спробу зареєструвати користувача з порожнім полем імені. В результаті, система відобразила сповіщення, що користувач не може бути зареєстрований без введеного імені. Ця поведінка системи вірна, оскільки інформація про ім'я є обов'язковою для подальшої роботи користувача. Результати цього тестування можна побачити на рисунку 4.2.

Переконавшись, що система не дозволяє реєструвати користувача без обов'язкових полів, спробуємо ввести неправильне значення в поле електронної адреси. В результаті тестування, система відобразить сповіщення про неправильну електронну адресу, яке можна побачити на рисунку 4.2.

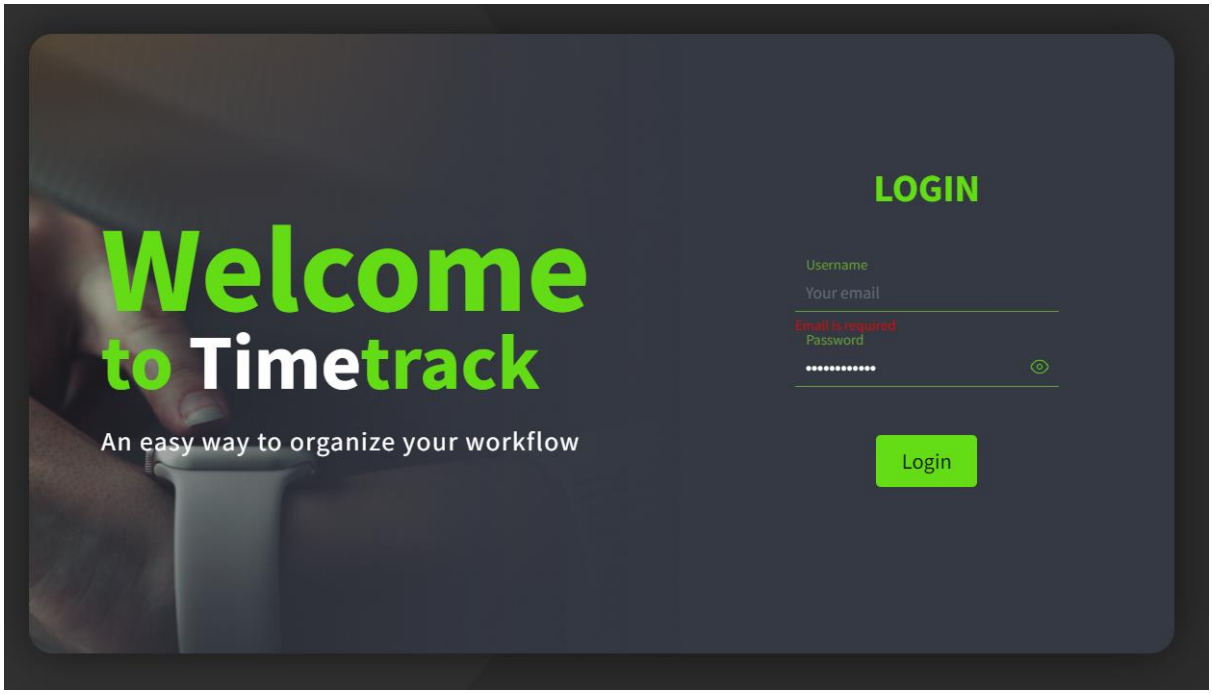

Рисунок 4.2 – Реєстрація користувача

Після перевірки введення недійсних даних, ми спробуємо правильно заповнити всі поля та перевірити, чи зареєструє система користувача. В результаті цієї перевірки, система перенаправить користувача на сторінку dashboard, що підтвердить успішну реєстрацію користувача. Результат тестування можна побачити на рисунку 4.3.

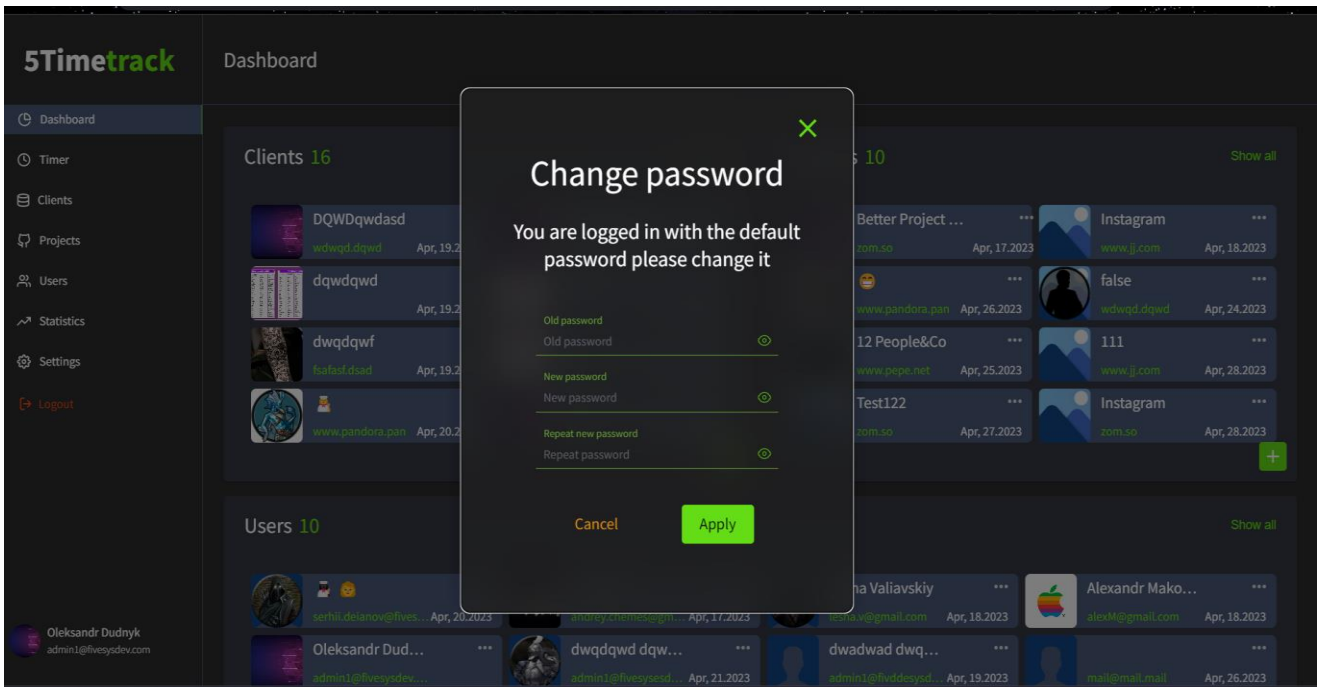

Рисунок 4.3 – Результат успішної реєстрації користувача

Юніт-тестування є одним із найважливіших компонентів процесу розробки програмного забезпечення. Воно дозволяє перевірити окремі функціональні одиниці коду, такі як функції, методи або класи, з метою виявлення помилок і недоліків. Юніт-тести пишуться розробниками програмного забезпечення та виконуються автоматично без взаємодії з іншими частинами системи.

Для забезпечення ефективного тестування проєкту був створений тестовий набір, який включає різні тести для перевірки функціональності окремих модулів. Тестовий набір розділений на підкатегорії відповідно до модулів, що тестуються, з метою забезпечення чіткої структури та організації тестів.

Огляд Jest і юніт-тестування в React: Jest є одним із найпопулярніших фреймворків для юніт-тестування в середовищі React. Він надає потужний інструментарій для створення тестів, включаючи підтримку асинхронного коду, відслідковування покриття коду тестами та зручний інтерфейс для перевірки результатів.

Використання Jest для юніт-тестування в React дозволяє розробникам перевірити правильність роботи компонентів, їх взаємодію та відповідність очікуваному результату. Тестування в React вимагає спеціалізованих підходів, оскільки компоненти взаємодіють зі станом, властивостями та подіями.

Написання тестів для основного функціоналу рисунка 4.4 допомагає переконатися, що цей функціонал працює правильно і відповідає очікуваному результату. Тести можуть перевіряти різні аспекти, такі як коректність обробки вхідних даних, правильність відображення результату та взаємодію з іншими компонентами.

Написання тестів у Jest вимагає визначення очікуваного результату і перевірки його з фактичним результатом виконання функціоналу. За допомогою спеціальних функцій та методів Jest, таких як **expect**, можна визначити умови успішного проходження тесту.

При виконанні тестів у Jest можна також використовувати різні розширення та додаткові інструменти для полегшення процесу тестування, такі як модульні моки, шпигуни (spies) та інші.

Загальною метою написання тестів є забезпечення стабільності та якості програмного забезпечення шляхом перевірки різних аспектів його функціоналу. Юніт-тестування допомагає виявити помилки та недоліки на ранніх етапах розробки, що дозволяє їх швидко виправити та покращити якість продукту.

Отже, використання Jest і юніт-тестування в React дозволяє розробникам ефективно перевіряти функціональність та якість коду, покращувати стабільність та надійність програмного забезпечення, а також спрощує процес виявлення та виправлення помилок. Завдяки цьому, розробка проєкту стає більш ефективною та надійною.

На рисунку 4.4 зображено пройдені тести це значить що все працює.

```
PASS  src/core/Tests/ModalSearchItem.test.tsx (17.816 s)
PASS  src/core/Tests/Projects.test.tsx (23.416 s)
PASS  src/core/Tests/Statistics.test.tsx (22.201 s)
PASS  src/core/Tests/WeeklyLog.test.tsx (24.814 s)
PASS  src/core/Tests/MonthlyLogItem.test.tsx (6.741 s)

Test Suites: 5 passed, 5 total
Tests:       10 passed, 10 total
Snapshots:   1 passed, 1 total
Time:        35.173 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Рисунок 4.4 – Результати тестів

На рисунку 4.5 представлена форма, яка використовується для логування часу. У цій формі присутні різні поля, які необхідно заповнити для введення даних про робочий час.

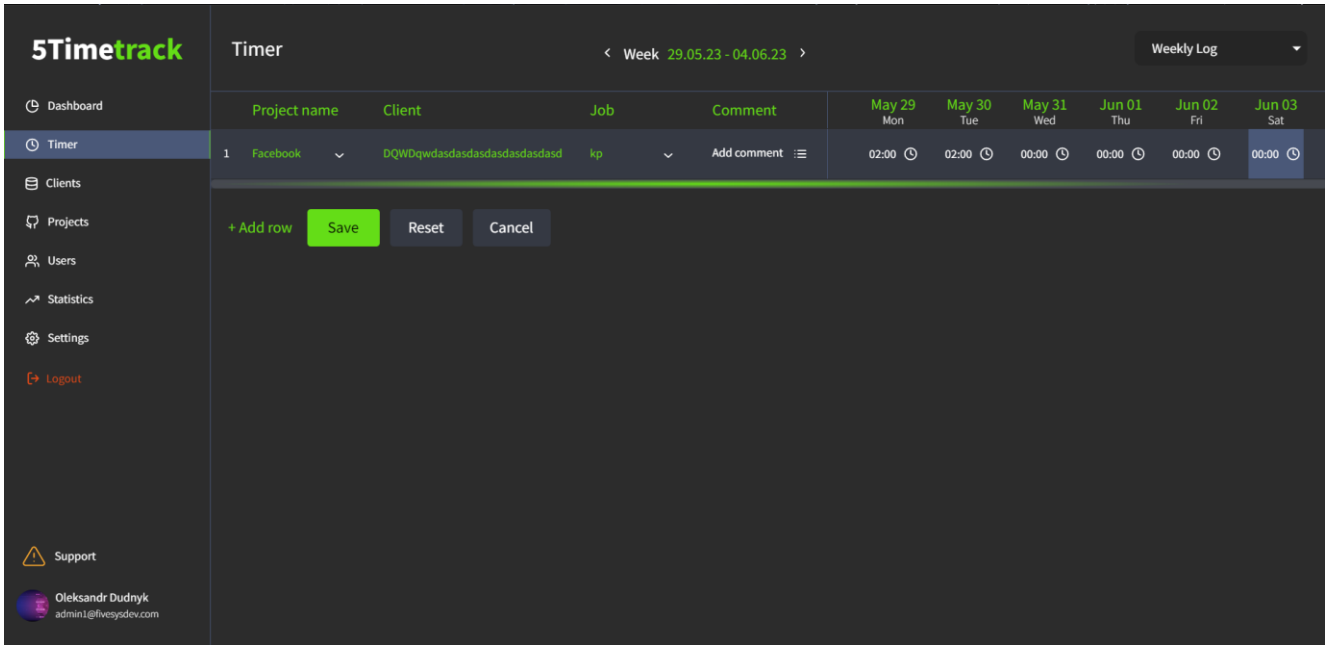

Рисунок 4.5 – Логування часу

Процес тестування логування часу передбачає заповнення всіх полів форми, перевірку коректності введених даних та відправку цих даних на сервер для подальшого оброблення.

Для початку, ми заповнимо поле "Проєкт" обираючи з доступних варіантів проєкту, на якому працює працівник. Після цього, в поле "Дата" введемо поточну дату, а в поле "Початок" та "Кінець" введемо час початку та закінчення робочого періоду відповідно.

Для перевірки коректності введених даних, ми будемо використовувати валідацію полів форми. Наприклад, поле "Дата" повинно бути валідною датою, а поля "Початок" та "Кінець" повинні містити коректний час. Якщо будь-яке з полів не відповідає вимогам, форма повинна відобразити відповідне повідомлення про помилку.

Після заповнення форми та перевірки коректності даних, натиснення кнопки "Відправити" спричинить відправку даних на сервер. У цей момент буде здійснено перевірку з'єднання з сервером та відправка даних. У разі успішної відправки, сервер повинен обробити дані та зберегти їх відповідно до встановлених правил.

Тестування логування часу є важливим етапом розробки, оскільки від нього залежить коректність та достовірність збережених даних про час працівників. Після успішного проходження цього тесту, можна переходити до наступних етапів тестування та розробки веб-системи для моніторингу часу працівників.

Після натискання кнопки "Save" в формі логування часу, може з'явитися алерт, який повідомляє про результат збереження даних. На рисунку 4.6 видно, що колір алерту є червоним, а повідомлення вказує на обов'язкові поля для логування часу - коментар та час.

Такий алерт відображається, коли не всі необхідні поля були заповнені перед натисканням кнопки "Save". Це допомагає користувачу уникнути збереження неповних або некоректних даних. Коментар та час є обов'язковими полями, які повинні бути заповнені для точного логування часу працівника.

Завдяки такому алерту користувач може легко виявити та виправити невірні дані, додати обов'язкові поля та повторно натиснути кнопку "Save", щоб зберегти правильні дані. Це забезпечує точність та достовірність збережених даних про час працівника в системі моніторингу.

Крім того, червоний колір алерту використовується для виділення помилок або некоректних вводів, що допомагає визначити проблемні області та вжити необхідні дії для їх виправлення.

В цілому, відображення червоного алерту з вказівкою на обов'язкові поля допомагає забезпечити коректну та повну інформацію про логування часу працівників, підвищує точність та надійність системи моніторингу та полегшує користувачам внесення правильних даних. На рисунку 4.6 видно, що колір алерту красний.

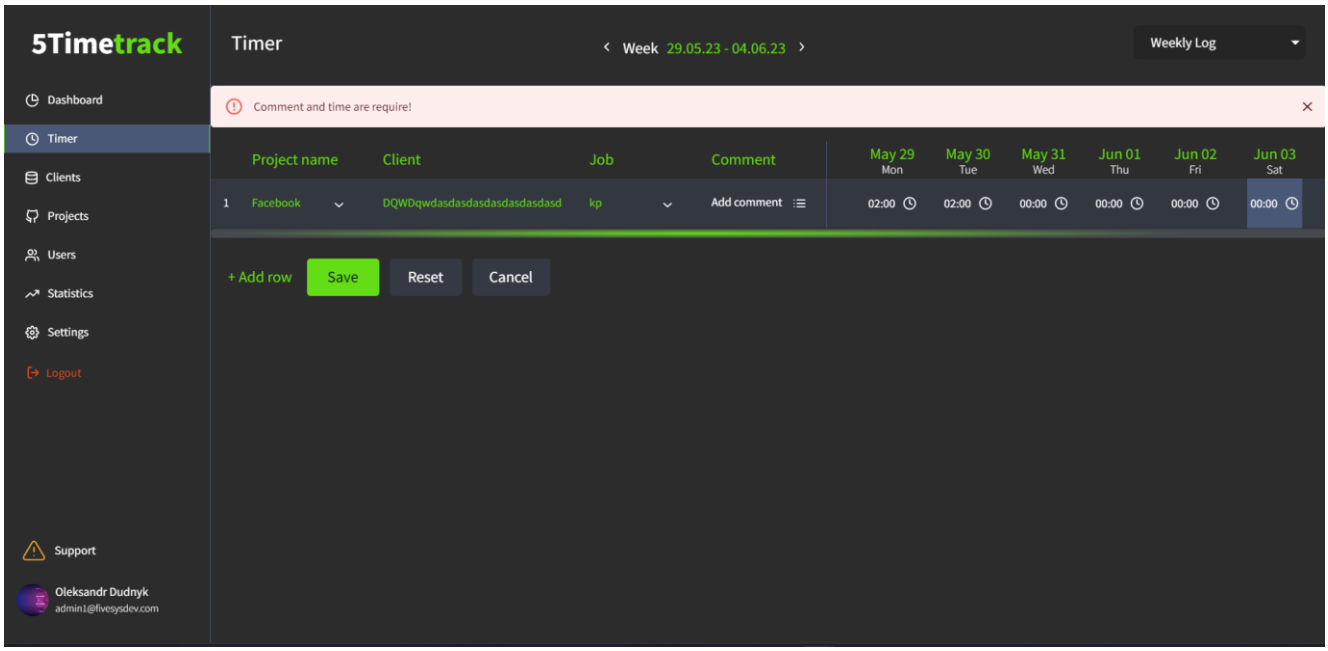

Рисунок 4.6 – Логування часу

Тому заповнимо всі поля форми та подивимося на результат тестування.

На рисунку 4.7 видно, що колір алерту є зеленим, а повідомлення вказує на те, що дія, яку ми виконали, була успішною. Це означає, що ми успішно залогували наші часи на проєкті.

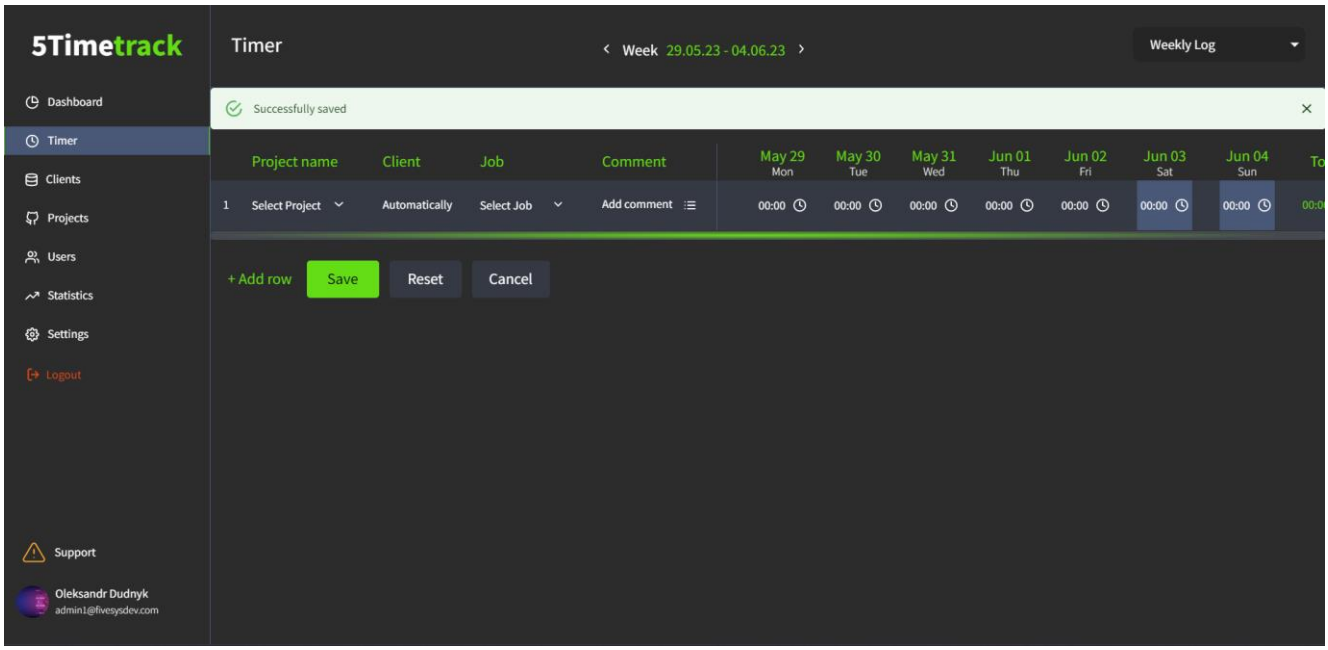

Рисунок 4.7 – Логування часу з успішністю

Зелений колір алерту традиційно використовується для позитивних повідомлень, що свідчить про успішне виконання дії або завершення процесу. У цьому випадку, він підтверджує, що дані про логування часу були успішно збережені та відправлені до системи.

Крім того, на рисунку 4.7 також видно наявність кнопок, що дозволяють додати кількість рядків для логування часу. Це дає користувачу можливість додати більше записів про час працівника на проєкті. Така функціональність корисна, коли потрібно внести інформацію про багато різних робочих годин або різні види роботи на проєкті.

Крім того, існує кнопка "Reset All", яка дозволяє скинути всі введені дані та повернутися до початкового стану форми. Це зручна функція, яка дає можливість користувачам почати заново, якщо вони зробили помилки або хочуть почати логування часу з чистого листа.

Такі функції дозволяють полегшити процес логування часу працівників, надають більшу гнучкість та зручність користувачам та забезпечують точну та повну інформацію про час працівників на проєкті.

На рисунку 4.8 показана вкладка "Timer -> List", де ми можемо отримати більш детальну інформацію про залоговані часи на проєктах. Ця вкладка надає зручний спосіб перегляду та управління залогованими часами працівників.

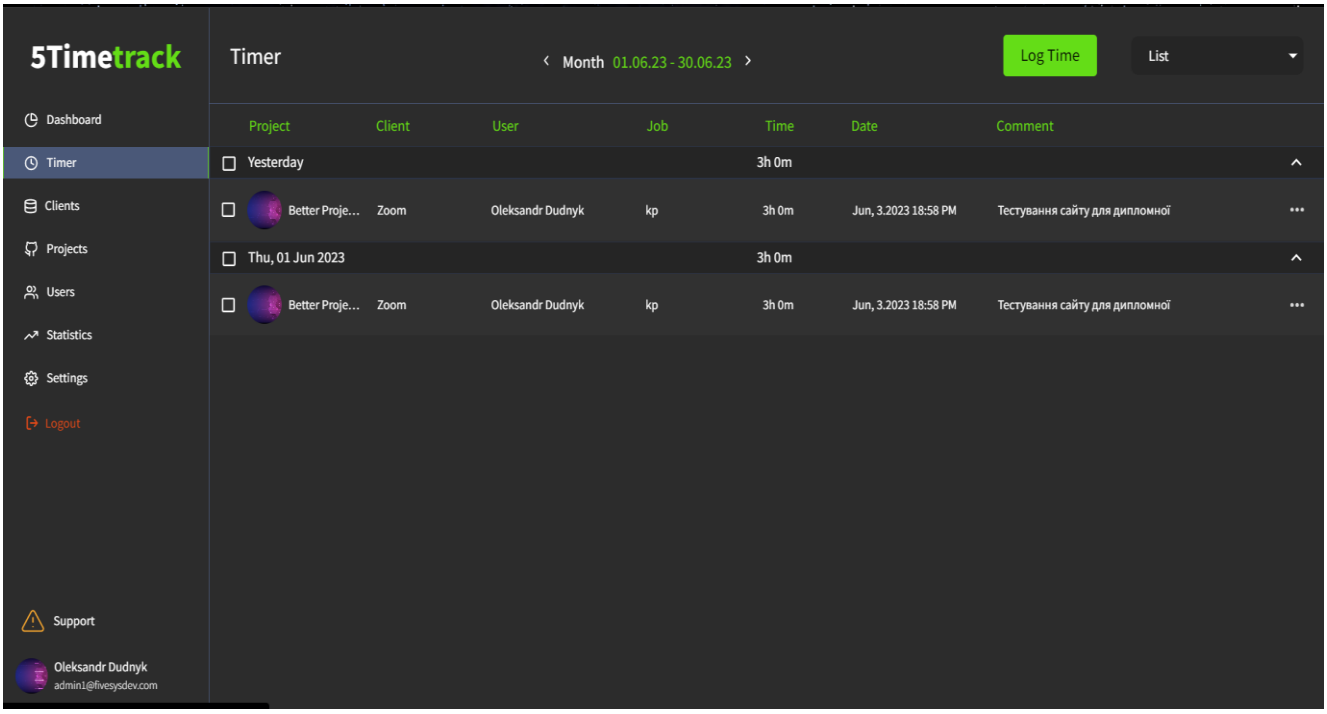

Рисунок 4.8 – Інформація про часи

На даному рисунку можна побачити список залогованих часів на різних проєктах. Кожен запис включає інформацію про проєкт, дату, час початку та закінчення, а також загальний час, проведений на проєкті. Це дозволяє нам відстежувати, скільки часу працівники витратили на кожен проєкт і як розподілилася їх робоча активність протягом періоду.

Крім того, на рисунку 4.8 видно, що список часів може бути сортованим за різними параметрами, наприклад, за датою або за проєктом. Це дозволяє зручно організувати та фільтрувати дані залогованих часів, щоб швидко знайти необхідну інформацію.

Ця вкладка також може надати можливість виконати додаткові дії з залогованими часами, наприклад, редагувати або видаляти записи. Це дозволяє користувачам коригувати інформацію, якщо виникла потреба внести зміни в залоговані часи або виправити помилки.

В цілому, вкладка "Timer -> List" надає зручний інтерфейс для перегляду та управління залогованими часами на проєктах. Вона дозволяє нам більш детально аналізувати та контролювати часові ресурси працівників, що сприяє ефективному управлінню проєктами та розподілу робочого часу.

На рисунку 4.9 можна побачити, що всі запити пройшли успішно, що підтверджує коректну роботу веб-платформи. Кожен із запитів, як до серверної частини, так і до бази даних, виконується без помилок і з очікуваними результатами. Це свідчить про те, що система здатна правильно обробляти запити користувачів та ефективно взаємодіяти з усіма її компонентами, що є важливим аспектом для стабільної роботи платформи. Важливо, що усі запити були оброблені в межах оптимального часу, що також свідчить про високу продуктивність системи під навантаженням.

Успішне виконання тестів запитів є важливим етапом перевірки функціональності та забезпечення коректного обміну даними між клієнтською частиною та сервером. Це підтверджує, що платформа готова до реального використання та може працювати на великих обсягах даних і при великій кількості одночасних користувачів. Виявлення та виправлення помилок на цьому етапі дозволяє значно підвищити надійність системи перед її остаточним впровадженням.

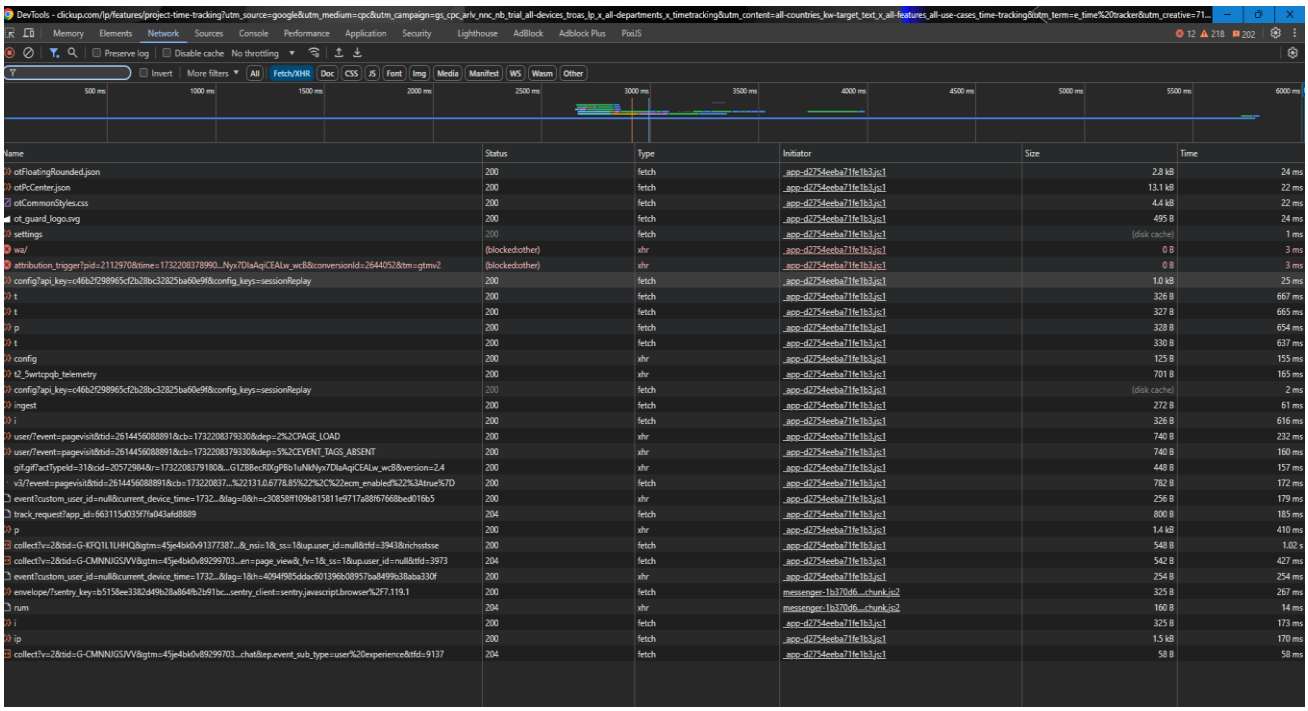

Рисунок 4.9 – Запити загрузки сайту

Також на рисунку 4.10 можна побачити процес створення нового користувача та налаштування його профілю в системі. На цьому етапі користувач вводить необхідну інформацію, таку як ім'я, прізвище, адреса електронної пошти та інші персональні дані, що дозволяє системі коректно ідентифікувати користувача та надавати йому доступ до необхідних функцій. Крім того, на цьому етапі можна налаштувати додаткові параметри, такі як рівень доступу, роль у системі або налаштування сповіщень.

Налаштування профілю є важливим етапом, оскільки воно дозволяє кожному користувачу персоналізувати свій досвід роботи з платформою, а також визначити, які саме функції та сервіси будуть для нього доступні. Наприклад, адміністратор може призначати ролі, що дозволяє контролювати доступ до чутливих даних і налаштовувати дозволи для різних категорій користувачів.

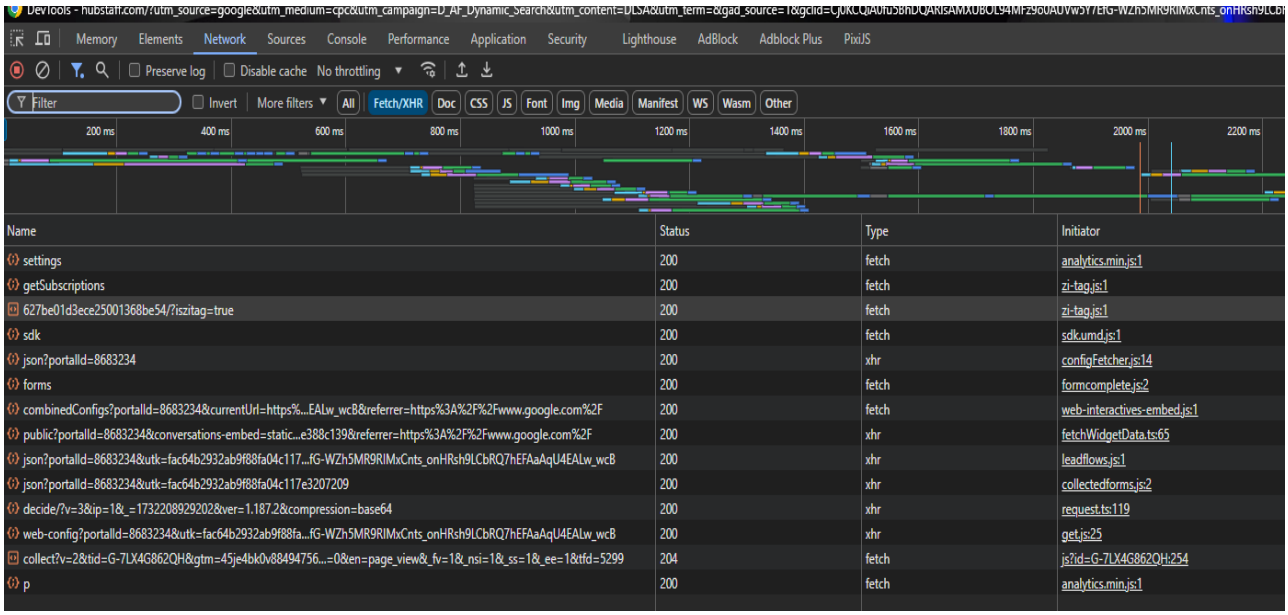

Рисунок 4.10 – Запити логування часу

Цей етап створення та налаштування користувача забезпечує не лише зручність у використанні платформи, але й допомагає підтримувати безпеку системи, гарантуючи, що кожен користувач має доступ тільки до тих функцій, що відповідають його правам та обов'язкам.

4.2 Розробка інструкції користувача

У інструкції користувача передбачено визначення технічних вимог для запуску програмного продукту. Це важливий крок, що допомагає забезпечити оптимальну продуктивність та стабільну роботу системи.

Технічні вимоги включають мінімальну та рекомендовану конфігурацію серверного комп'ютера, необхідну для встановлення та ефективної роботи програмного продукту.

У таблиці 4.1 наведено мінімальну конфігурацію серверного комп'ютера. Ці параметри представляють найнижчі вимоги, які необхідні для запуску програмного продукту. Наприклад, це може включати певний обсяг оперативної пам'яті, мінімальну версію операційної системи, потужність

процесора та інші параметри, які впливають на продуктивність та сумісність системи.

Визначення технічних вимог є важливим кроком для користувачів, оскільки вони допомагають забезпечити оптимальну продуктивність та уникнути можливих проблем зі сумісністю. Користувачі повинні уважно ознайомитися з цими вимогами та переконатися, що їх серверний комп'ютер відповідає рекомендованим параметрам.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1 ГГц
Об’єм оперативної пам’яті	4 ГБ
Місце на жорсткому диску	20 ГБ
Операційна система	Windows 7

У таблиці 4.2 наведено рекомендовану конфігурацію серверного комп'ютера. Ці параметри є оптимальними для забезпечення найвищої продуктивності та надійності системи. Вони можуть включати в себе великий обсяг оперативної пам'яті, потужний процесор, швидкий диск та інші компоненти, які забезпечують швидку та ефективну роботу програмного продукту.

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	64-розрядний (x64) процесор з тактовою частотою 2 ГГц
Об’єм оперативної пам’яті	8 ГБ
Розмір жорсткого диску	512 ГБ
Операційна система	Windows 10

Для встановлення програмного продукту потрібно використовувати Docker – це платформа, яка дозволяє швидко розробляти, тестувати і розгортати додатки. Docker упаковує програмне забезпечення в контейнери, кожен з яких містить усе необхідне для роботи програми, такі як бібліотеки, системні інструменти, код і середу виконання . Це дозволяє користувачеві швидко розгортати та масштабувати програмний продукт в будь-якому середовищі, зберігаючи впевненість, що код буде працювати. Використання Docker зменшує необхідність вручну налаштовувати середовище, оскільки користувачу достатньо виконати команди для запуску контейнерів на онлайн-платформі.

Після запуску платформи користувач повинен авторизуватися в системі та перейти на головну сторінку для вибору логування свого часу на проєкті.

4.3 Висновок до розділу 4

У четвертому розділі проведено детальне тестування функціоналу логування часу в програмному продукті TimeTrack. Під час тестування було акцентовано на декількох аспектах, включаючи швидкість виконання запитів на отримання всіх логів часу користувачів, валідацію полів введення інформації та правильне відображення елементів сторінки.

Один з важливих аспектів тестування полягав у перевірці швидкості виконання запитів на отримання всіх логів часу користувачів. Це важливо, оскільки великий обсяг даних може вплинути на продуктивність програми. Тестування включало створення значного обсягу логів часу та перевірку часу відповіді системи на запити їх отримання. Результати тестування допомогли виявити ефективність системи обробки та виконання запитів, а також визначити можливі проблеми з продуктивністю.

Крім того, проводилася перевірка валідації полів введення інформації, пов'язаної з логуванням часу. Це включало перевірку правильності обробки

даних, таких як дата, час та інші параметри. Тестування полягало в спробі ввести некоректні дані та перевірку, як система обробляє такі випадки. Це допомогло забезпечити коректну роботу системи та виявити можливі помилки у валідації.

Окрім функціонального тестування, також проводилося тестування відображення елементів сторінки, пов'язаних з функціоналом логування часу. Це включало перевірку наявності та правильного розташування необхідних елементів, як-то форми для введення даних, кнопок та повідомлень про успішну операцію. Тестування допомогло переконатися, що інтерфейс користувача відображається належним чином та користувач може легко взаємодіяти з функціоналом логування часу.

5 ЕКОНОМІЧНА ЧАСТИНА

Для впровадження науково-технічної розробки важливо, щоб вона відповідала вимогам сучасного технічного прогресу та була економічно ефективною. Оцінка можливих економічних переваг від її реалізації є ключовою для визначення доцільності проекту [25].

Магістерська робота на тему «Інформаційна технологія для обліку робочого часу працівників» є прикладом інноваційної розробки, яка має потенціал для комерціалізації. Впровадження цієї розробки дозволить підвищити ефективність обліку робочого часу, що дасть компаніям можливість отримати значні економічні вигоди. Для успішної реалізації ідеї необхідно переконати інвесторів у її життєздатності та перспективності.

У процесі виконання роботи передбачено наступні етапи:

1. Оцінка розробки для визначення її технологічної готовності та ринкових можливостей.
2. Визначення витрат, необхідних для реалізації проекту.
3. Розрахунок економічного ефекту і обґрунтування доцільності комерціалізації.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія для обліку робочого часу працівників» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюється із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в таблиці 5.1

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Ринкові перспективи					
	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1

	0	1	2	3	4
	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
0	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуют ь ся у військово промисловому	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс
1	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
2	Необхідна розробка регламентних документів та отримання великої кількості продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки зведені до таблиці 5.2. Участь в опитуванні брали експерти з ВНТУ, кафедри комп’ютерних наук: к.т.н., доцент Озеранський Володимир Сергійович, к.т.н., доцент Крилик Людмила Вікторівна, к.т.н., проф. Колесницький Олег Костянтинович.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Озеранський В.С.	Крилик Л.В.	Колесницький О. К
	Бали		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	4	3	4
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	4	3	3
5. Ринкові переваги (експлуатаційні витрати)	4	4	3
6. Ринкові перспективи (розмір ринку)	4	4	4
7. Ринкові перспективи (конкуренція)	4	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	4	4	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	$СБ_1$ =45	$СБ_2$=42	$СБ_3$ =43
Середньоарифметична сума балів $СБ_c$	43,3		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в таблиці 5.3.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно з проведеними дослідженнями рівень комерційного потенціалу розробки за темою «Інформаційна технологія для обліку робочого часу працівників» становить 43,3 бали, які, відповідно до таблиці 5.3, свідчать про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки «Високий».

Дана розробка призначена для користувачів, підприємства які хочуть збільшити ефективність роботи працівників.

5.2 Визначення рівня конкурентоспроможності розробки

Визначаючи економічну ефективність науково-технічної розробки є доцільним провести прогноз рівня конкурентоспроможності за сукупністю параметрів, які підлягають оцінюванню.

Одиничний параметричний індекс розраховується за формулою [25]:

$$q_i = \frac{P_i}{P_{баз\,i}}, \tag{5.1}$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.4.

Таблиця 5.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
Точність	73	> 89	1,55	0,6
Кількисть запитів за хвилину	2	60	30	0,4

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [25]:

$$I_{НП} = \prod_{i=1}^{n} q_i \ , \tag{5.2}$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблювальна інформаційна технологія відповідає вимогам ДСТУ, тому $I_{нп}$ = 1.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [25]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \,, \tag{5.3}$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 5.4.

$$I_{тп} = 1{,}55 \cdot 0{,}6 + 30 \cdot 0{,}4 = 12{,}93.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [25]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \,, \tag{5.4}$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i – економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці.

$$I_{ЕП} = 0,82 \cdot 0,5 + 0,68 \cdot 0,5 = 0,75.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [25]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{5.5}$$

$$K_{ІНТ} = 1 \cdot 12,99 / 0,75 = 17.32.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія для обліку робочого часу працівників», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою 5.6:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 30000{,}00 \cdot 63 / 22 = 85909{,}09 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.4.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проєкту	30000	1363,64	63	85909,09
Інженер-розробник програмного забезпечення	25000	1136,36	63	71590,91
Консультант	17000	772,73	63	48681,82
Всього				206181,82

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою 5.7:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \,, \qquad (5.7)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою 5.8:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \qquad (5.8)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$$C_1 = 8000{,}00 \cdot 1{,}10 \cdot 1{,}65 / (22 \cdot 8) = 82{,}50 \text{грн.}$$

$$З_{р1} = 82.50 \cdot 6{,}00 = 495.00{,}00 \text{грн.}$$

Величина витрат на основну заробітну плату робітників наведена в табл. 5.5.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
Підготовка робочого місця дослідника	6	2	82,50	495.00
Інсталяція програмного забезпечення	4	5	106,78	424,00
Тестування	8	5	106,78	848,00
				1767,00

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою 5.9:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$$З_{дод} = (206181,82 + 1695,94) \cdot 11 / 100\% = 22874,37 грн.$$

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою 5.10:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн$ = (206181,82 + 1695,94+ 22874,37) · 22 / 100% = 50765.47 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою 5.11:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}\ , \qquad (5.11)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$$М_1 = 3 \cdot 195{,}00 \cdot 1{,}1 - 0{,}000 \cdot 0{,}00 = 643{,}5 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір	195	3	0	0	643,5
Канцелярське приладдя (набір офісного працівника)	3	0	0	511,5	3
Flesh-пам'ять	150	3	0	0	495
Всього					1650

5.3.4 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою [25]:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (5.12)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, (грн);

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$$B_{спец} = 40000{,}00 \cdot 1 \cdot 1{,}11 = 44400 \text{ (грн.)}.$$

Отримані результати зведемо до таблиці 5.8:

Таблиця 5.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
1. Комп’ютер	1	40000	44400
2. Клавіатура Razer 87	1	1150	1276,5
3. Миша Razer P93	1	999	1108,89
4. Монітор 24`	2	10000	22200
Всього			68985.39

5.3.5 Програмне забезпечення для наукових робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проєктування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою 5.13:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.13)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *15000,00 · 1 · 1,12 = 50400 грн.*

Отримані результати зведемо до таблиці 5.9.

Таблиця 5.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
IntelliJ IDEA	1	5500	6160
Операційна система Microsoft Windows 10 Pro for Workstations	3	15000	50400
Всього			56560

5.3.6 Амортизація обладнання, програмних засобів та приміщень

У спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою 5.14:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.14)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$A_{обл} = (25000{,}00 \cdot 3) / (4 \cdot 12) = 1562{,}5 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.8.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
1	2	3	4	5
ПК DELL OptiPlex 7010 SFF (N001O7010SFF UBU), монітор	25 000	4	3	1562,50
ПК Expert PC Ultimate (I11400F.16.H1S2.1650. A3836), монітор	26 000	4	3	1625,00

Продовження таблиці 5.8.

1	2	3	4	5
ПК 2E Integer Intel C J4005, монітор	12 000	4	3	750,00
Оргтехніка	7000	4	3	437,50
Операційна система Microsoft Windows 10 Pro for Workstations	50400	1	3	12600,00
IntelliJ IDEA	6160	1	3	1540,00
Всього				18515,00

5.3.7 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою 5.15:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (5.15)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e = 7{,}50$ грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0{,}3 \cdot 480{,}0 \cdot 7{,}50 \cdot 0{,}95 / 0{,}97 = 1057{,}73$ *грн.*

Проведені розрахунки зведемо до таблиці 5.9.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
ПК DELL OptiPlex	0,3	480	1057,73
ПК Expert PC Ultimate	0,3	504	1110,62
ПК 2E Integer Intel C J4005	0,2	400	587,63
Оргтехніка	0,45	15	49,58
Всього			2805,56

5.3.8 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою 5.16:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (5.16)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св}$ = (206181,82+ 1695,94) · 20 / 100% = 41575,55 грн.

Витрати на роботи, які виконують сторонні підприємства, установи і організації

5.3.9 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 5.17:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.17)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 70%.

$I_в$ = (206181,82+ 1695,94) · 70 / 100% = 145514,43 грн.

5.3.10 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою 5.18:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.18)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$В_{нзв}$ = (206181,82+ 1695,94) · 120 / 100% = 249 453,31 грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою 5.19:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}.$$

(5.14)

$B_{заг}$=*964939,12грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою 5.19:

$$ЗВ = \frac{B_{заг}}{\eta}, \qquad (5.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,7.

ЗВ = 964939,12/ 0,7 = 1378484,46 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

У ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

У цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 1000 проєктних груп;

2-й рік – 3000 проєктних груп;

3-й рік – 2500 проєктних груп.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 5000 проєктних груп;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 250000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 20000,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою 5.20:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 25\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (20000{,}00 \cdot 5000{,}00 + 270000 \cdot 1000) \cdot 0{,}83 \cdot 0{,}25 \cdot (1\text{-}0{,}18/100\%) =$ 63183050 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (20000{,}00 \cdot 5000{,}00 + 270000 \cdot (1000 + 3000)) \cdot 0{,}83 \cdot 0{,}25 \cdot (1\text{-}0{,}18/100\%) =$ 201502700 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (20000{,}00 \cdot 5000{,}00 + 270000 \cdot (1000+3000+2500)) \cdot 0{,}83 \cdot 0{,}25 \cdot (1-0{,}18/100\%) = 316769075$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки, знаходиться за формулою 5.21:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,3;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП = 63183050 /(1+0{,}3)^1 + 201502700 /(1+0{,}3)^2 + 316769075 /(1+0{,}3)^3 =$ 312017268,78 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки, знаходиться за формулою 5.22:

$$PV = k_{інв} \cdot ЗВ, \qquad (5.22)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=4;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1378484,46 грн.

$PV = k_{інв} \cdot ЗВ$ = 4· 1378484,46 = 5513937,821.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме за формулою 5.23:

$$E_{абс} = ПП - PV \tag{5.23}$$

де *ПП* – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 312017268,78 грн;

PV – теперішня вартість початкових інвестицій, 5513937,821 грн.

$E_{абс} = ПП - PV$ = 312017268,78 - 5513937,821 = 306503330,95 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, знаходиться за формулою 5.24:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.24}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 306503330,95 грн;

PV – теперішня вартість початкових інвестицій, 5513937,821 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1 + 306503330,95 / 5513937,821)^{1/3} = 2,84.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$, знаходиться за формулою 5.25:

$$\tau_{мін} = d + f, \tag{5.25}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,11+0,25 = 0,36 < 2,84 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія для обліку робочого часу працівників» доцільно.

Період окупності інвестицій $T_{ок}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розрахуємо за формулою 5.27:

$$T_{ок} = \frac{1}{E_в}, \tag{5.26}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 2,84 = 0,35 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.5 Висновок до розділу 5

За проведеними дослідженнями рівень комерційного потенціалу розробки за темою «Інформаційна технологія для обліку робочого часу працівників» становить 43,3 бали, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

Також термін окупності становить 0,35 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія для обліку робочого часу працівників».

ВИСНОВКИ

У магістерській кваліфікаційній роботі було розроблено web систему для урахування часу працівників на поєктах під назвою “Time Track”. Розроблене ПЗ призначено для підвищення ефективності працюванні працівників

В робот було проведено аналіз сфери керування електронними ресурсами та розглянуто існуючі аналоги, доведено доцільність розробки власного програмного продукту. Було розроблено архітектуру створюваного ПЗ, розроблено схему компонентів і їх взаємодії, UML-діаграму діяльності та алгоритм формування змінних хостів, що дає можливість виконувати більш складні конфігурації. Розроблено алгоритм запуску конфігураційних скриптів.

Управління робочим часом є важливим для бізнесу, але ручні методи часто неточні та забирають багато часу. Впровадження програмного забезпечення для обліку часу, як доведено дослідженнями, підвищує точність на понад 30% і скорочує витрати часу на адміністрування до 40%.

Розширення функціональних можливостей інформаційної технології для обліку робочого часу працівників, яка забезпечить автоматизацію процесу, підвищить його точність та мінімізує витрати часу на облік.

Мета дослідження – підвищення точного логування часу, інтегрування ШІ, для точного вираховування часу працівників на проєктах, досягається за рахунок того, що в результаті тестування, модель показала високі оцінки за рядом критеріїв, таких як швидкодія, зручність, точність, що стало можливим завдяки покращенню інформаційної технології.

Результати дослідження підтверджують доцільність впровадження розробленої технології та відкривають перспективи для подальшого вдосконалення та розширення її функціональних можливостей.

Рівень комерційного потенціалу розробки становить 43,3 бали, що, свідчить про комерційну важливість проведення даних досліджень Термін окупності становить 0,35 р.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Валявський О.С., Богач І.В. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ ОБЛІКУ РОБОЧОГО ЧАСУ ПРАЦІВНИКІВ // Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22780.

2. Toggl [Електронний ресурс] – Режим доступу до ресурсу: https://toggl.com/

3. Harvest [Електронний ресурс] – Режим доступу до ресурсу: https://hubstaff.com/

4. Hubstaff [Електронний ресурс] – Режим доступу до ресурсу:

5. Alan Forbes. The Joy of PHP – 2015. – 128 с.

6. Douglas Crockford. JavaScript: The Good Parts – 2013. – 176 с.

7. Paul Barry. Head First Python: A Brain-Friendly Guide – 2016. – 622 с.

8. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/.

9. WebStorm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/.

10. PhpStorm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/phpstorm/.

11. Docker [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/docker/.

12. JavaScript [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/JavaScript.

13. Інформаційні системи і технології в обліку: програма практики / [уклад. О. В. Клименко]. – Вінниця: ВТЕІ, 2023. – 30 с.

14. Martin Fowler. Refactoring: Improving the Design of Existing Code - 1999. - 431 p.

15. Robert C. Martin. Clean Code: A Handbook of Agile Software

Craftsmanship - 2008. - 464 p.

16. Eric Evans. Domain-Driven Design: Tackling Complexity in the Heart of Software - 2003. - 560 p.

17. Kent Beck. Test-Driven Development by Example - 2002. - 240 p.

18. Steve McConnell. Code Complete: A Practical Handbook of Software Construction - 1993. - 960 p.

19. Scott Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs - 2005. - 320 p.

20. OWL [Електронний ресурс]. – URL: https://homes.cs.washington.edu/~billhowe/cs410/owl_overview.pdf

21. Створення SPARQL запитів [Електронний ресурс]. – URL: https://sachipcranasinghe.medium.com/intro-to-sparql-part1-103957d8e854

22. Humanitarian Assistance Ontology for Emergency Disaster Response [Електронний ресурс]. – URL: https://ieeexplore.ieee.org/document/6813390

23. Social Participation Ontology [Електронний ресурс]. – URL: https://arxiv.org/abs/1501.02662

24. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

25. Як розраховується прибуток [Електронний ресурс]. – URL: https://support.joinposter.com/uk/articles/7067856.

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Інформаційна технологія для обліку робочого часу працівників

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

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

Показники звіту подібності Unicheck

Оригінальність 96% Схожість 4%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

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

Автор роботи Валявський О. С.

Керівник роботи Богач І. В.

Додаток Б (обов'язковий)

Лістинг основних функцій програми

```
{ Navigate, Route, Routes } from 'react-router-dom';
// Constants
import { ROUTE } from 'core/routers';
import { ADMIN } from 'core/constants';

// Store
import { useAppSelector } from 'hooks';
import { authSelector } from 'store/selectors';
// Components
import {
  Clients,
  Dashboard,
  Login,
  Projects,
  Settings,
  Statistics,
  Timer,
  Users,
  Page404,
} from 'pages';
import { MainLayout, PrivateRoutes, PrivateRoutesAdmin } from 'layouts';

export const MainRouts = () => {
  const { isAuthorised, role } = useAppSelector(authSelector);

  return (
    <Routes>
      <Route element={<PrivateRoutes />}>
        <Route path="*" element={<Page404 />} />

        <Route path="/" element={<MainLayout />}>
          <Route
            index
            element={
              role === ADMIN ? (
                <Navigate to={ROUTE.privateFieldsAdmin.dashboard} />
              ) : (
                <Navigate to={ROUTE.base.timer} />
              )
            }
          />
          <Route path="/" element={<PrivateRoutesAdmin />}>
            <Route path={ROUTE.privateFieldsAdmin.dashboard}
element={<Dashboard />} />
            <Route path={ROUTE.privateFieldsAdmin.clients} element={<Clients
/>} />
            <Route path={ROUTE.privateFieldsAdmin.projects} element={<Projects
/>} />
            <Route path={ROUTE.privateFieldsAdmin.users} element={<Users />}
/>
            <Route path={ROUTE.privateFieldsAdmin.statistics}
element={<Statistics />} />
          </Route>
          <Route path={ROUTE.base.timer} element={<Timer />} />
          <Route path={ROUTE.base.settings} element={<Settings />} />
        </Route>
      </Route>
```

```
      {!isAuthorised && <Route path={ROUTE.signIn} element={<Login />} />}
    </Routes>
  );
};
```

DashBoard Page

```
import { useMemo, useState } from 'react';

import styles from './Dashboard.module.scss';

import { ROUTE } from 'core/routers';

import { dashboardListReducer } from 'core/utils/dashboardListReducer';

import { IClient, IProject, IUser } from 'store/services/types';

// Api hooks

import { useGetAllClientsQuery } from 'store/services/ClientApi';

import { useGetAllProjectsQuery } from 'store/services/ProjectApi';

import { useGetAllUsersQuery } from 'store/services/UserApi';

// img

import clientImg from 'assets/images/client.png';

import projectImg from 'assets/images/project.png';

import userImg from 'assets/images/user.png';

// Components

import { Grid } from '@mui/material';

import { DashboardCard } from 'pages';

import { Header, ClientsModal, ProjectModal, AddUserModal } from 'components';



export const Dashboard = () => {

  const { data: clientsList = [], isLoading: clientListLoading } =
useGetAllClientsQuery();

  const { data: projectsList = [], isLoading: projectListLoading } =
useGetAllProjectsQuery();

  const { data: usersList = [], isLoading: userListLoading } =
useGetAllUsersQuery();



  const [isOpenClientModal, setIsOpenClientModal] = useState<boolean>(false);

  const [isOpenProjectModal, setIsOpenProjectModal] =
useState<boolean>(false);

  const [isOpenUserModal, setIsOpenUserModal] = useState<boolean>(false);

  const [screenWidth] = useState<number>(window.innerWidth);
```

```
  const handleOpenClientModal = () => setIsOpenClientModal(true);

  const handleCloseClientModal = () => setIsOpenClientModal(false);


  const handleOpenProjectModal = () => setIsOpenProjectModal(true);

  const handleCloseProjectModal = () => setIsOpenProjectModal(false);


  const handleOpenUserModal = () => setIsOpenUserModal(true);

  const handleCloseUserModal = () => setIsOpenUserModal(false);

  const computedClients = useMemo(() => {

    const reducedArray = dashboardListReducer<IClient>(clientsList,
screenWidth);


    return reducedArray.map((item) => {

      return {

        id: item.id,

        image: item.avatarPath,

        title: item.name,

        created: item.createdAt,

        site: item.website,

      };

    });

  }, [clientsList, screenWidth]);


  const computedProjects = useMemo(() => {

    const reducedArray = dashboardListReducer<IProject>(projectsList,
screenWidth);


    return reducedArray.map((item) => {

      return {

        id: item.id,

        image: item.imagePath,

        title: item.name,
```

```
        created: item.createdAt,
        site: item?.client?.website,
      };
    });
  }, [projectsList, screenWidth]);

  const computedUsers = useMemo(() => {
    const reducedArray = dashboardListReducer<IUser>(usersList, screenWidth,
true);

    return reducedArray.map((item) => {
      return {
        id: item.id,
        image: item.avatarPath,
        title: `${item.firstName} ${item.lastName}`,
        created: item.createdAt,
        site: item.email,
      };
    });
  }, [usersList, screenWidth]);

  return (
    <div>
      <Header title="Dashboard"></Header>
      <div className={styles.dashboard__content}>
        <Grid container spacing={2}>
          <Grid container item={true} sm={12} md={6} height="410px"
borderRadius="10px">
            <DashboardCard
              handleOpenModal={handleOpenClientModal}
              cardData={computedClients}
              caption="client"
              length={clientsList.length}
```

```
            path={ROUTE.privateFieldsAdmin.clients}
            isLoading={clientListLoading}
            clientData={clientsList}
            userData={usersList}
            projectData={projectsList}
            defaultImg={clientImg}
          />
          <ClientsModal
            open={isOpenClientModal}
            image={clientImg}
            handleCloseModal={handleCloseClientModal}
          />
        </Grid>
        <Grid container item={true} sm={12} md={6} height="410px"
borderRadius="10px">
          <DashboardCard
            handleOpenModal={handleOpenProjectModal}
            cardData={computedProjects}
            caption="project"
            length={projectsList.length}
            path={ROUTE.privateFieldsAdmin.projects}
            isLoading={projectListLoading}
            clientData={clientsList}
            projectData={projectsList}
            userData={usersList}
            defaultImg={projectImg}
          />
          <ProjectModal
            openModal={isOpenProjectModal}
            closeModal={handleCloseProjectModal}
            clientsList={clientsList}
          />
```

```
          </Grid>

          <Grid container item={true} xs={12} height="390px"
borderRadius="10px">

            <DashboardCard

              handleOpenModal={handleOpenUserModal}

              cardData={computedUsers}

              caption="user"

              length={usersList.length}

              path={ROUTE.privateFieldsAdmin.users}

              isLoading={userListLoading}

              clientData={clientsList}

              userData={usersList}

              projectData={projectsList}

              defaultImg={userImg}

            />

            <AddUserModal open={isOpenUserModal}
handleClose={handleCloseUserModal} />

          </Grid>

        </Grid>

      </div>

    </div>

  );

};
```

Timer Page

```
import { useEffect, useState } from 'react';

import {

  useDeleteMultipleLoggedDaysMutation,

  useGetLoggedDaysQuery,

} from 'store/services/LoggedDaysApi';

import { Range } from 'react-date-range';
```

```
// Components
import { DeletePopup } from 'components';
import { TimerHeader } from 'pages';
import { WeeklyLog } from './WeeklyLog';
import { MonthlyLog } from './MonthlyLog';
// Store
import { useAppDispatch, useAppSelector } from 'hooks';
import { timerSelector } from 'store/selectors';
import { setEmptySelected, setSelected } from 'store/timer/timerSlice';
import { endOfMonth, startOfMonth } from 'date-fns';


export const Timer = () => {
  const dispatch = useAppDispatch();
  const { selected, currentMode } = useAppSelector(timerSelector);
  const [timerRange, setTimerRange] = useState<Range>({
    startDate: startOfMonth(new Date()),
    endDate: endOfMonth(new Date()),
    key: 'selection',
  });
  const [isOpenEditPanel, setIsOpenPanel] = useState<boolean>(false);
  const [isOpenPopup, setIsOpenPopup] = useState<boolean>(false);


  const { data: loggedDaysList = [], isLoading } =
useGetLoggedDaysQuery(timerRange);
  const [deleteSeveralLoggedDays] = useDeleteMultipleLoggedDaysMutation();


  const handleOpenPanel = () => setIsOpenPanel(true);
  const handleClosePanel = () => setIsOpenPanel(false);


  const deleteLoggedDaysHandler = () => {
    deleteSeveralLoggedDays({ ids: selected });
    dispatch(setSelected([]));
```

```
  closePopupHandler();
};

useEffect(() => {
  selected.length && dispatch(setEmptySelected());
}, [currentMode]);

const openPopupHandler = () => setIsOpenPopup(true);
const closePopupHandler = () => setIsOpenPopup(false);

return (
  <>
    <TimerHeader
      timerRange={timerRange}
      setTimerRange={setTimerRange}
      deleteLoggedDays={openPopupHandler}
      selected={!!selected.length}
    />
    {currentMode === 'list' ? (
      <MonthlyLog
        timerRange={timerRange}
        loggedDaysList={loggedDaysList}
        isLoading={isLoading}
        handleOpenPanel={handleOpenPanel}
        handleClosePanel={handleClosePanel}
        isOpenEditPanel={isOpenEditPanel}
      />
    ) : (
      <WeeklyLog />
    )}
    <DeletePopup
      closePopup={closePopupHandler}
```

```
        openPopup={isOpenPopup}
        confirm={deleteLoggedDaysHandler}
        itemType="log item"
      />
    </>
  );
};


import { useCallback } from 'react';
import styles from './WeeklyLog.module.scss';
// API hooks
import { useGetAllProjectsQuery } from 'store/services/ProjectApi';
import { useGetAllJobsQuery } from 'store/services/JobApi';
// Icons
import { ReactComponent as GreenClockIcon } from 'assets/icons/clock-
green.svg';
import projectAvatar from 'assets/images/project.png';
// Components
import { TableCell, TableRow } from '@mui/material';
import { StyledAutocomplete } from 'components';
import { CommentCell, WeeklyLogDay } from './';
// Store
import { useAppSelector, useAppDispatch } from 'hooks';
import { WeeklyTimerSelector } from 'store/selectors';
import { setProject, setJob } from 'store/weeklyTimer/weeklySlice';
import { IJob, IProject } from 'store/services/types';

export const WeeklyLogRow = () => {
  const dispatch = useAppDispatch();

  const { data: projectList = [] } = useGetAllProjectsQuery();
  const { data: jobList = [] } = useGetAllJobsQuery();
```

```
const { week } = useAppSelector(WeeklyTimerSelector);

const setProjectHandler = (value: IProject | null, index: number) => {
  dispatch(setProject({ value, index }));
};

const setJobHandler = (value: IJob | null, index: number) => {
  dispatch(setJob({ value, index }));
};

const totalTimeOut = useCallback(
  (rowIndex: number) => {
    const { hours, minutes } = week[rowIndex].total;
    let viewedHours = hours;
    let viewedMins = minutes;

    if (viewedHours.toString().length === 1) {
      viewedHours = '0' + hours;
    }
    if (viewedMins.toString().length === 1) {
      viewedMins = '0' + viewedMins;
    }

    return `${viewedHours}:${viewedMins}`;
  },
  [week],
);

return (
  <>
    {week.map((item, rowIndex) => {
```

```
        return (
          <TableRow key={item.id} hover role="checkbox">
            <TableCell
              padding="checkbox"
              sx={{ color: '#ffffff', borderColor: '#2C2C2C', textAlign:
'center' }}
            >
              {rowIndex + 1}
            </TableCell>
            <TableCell sx={{ borderColor: '#2C2C2C', padding: '14px 16px' }}>
              <StyledAutocomplete
                label="Select Project"
                options={projectList}
                setValue={(value: IProject | null) => setProjectHandler(value,
rowIndex)}
                value={item.project}
                defaultAvatar={item?.project?.imagePath ?
item.project?.imagePath : projectAvatar}
                width={120}
                textColor="#64dd17"
              />
            </TableCell>
            <TableCell
              sx={{
                color: item.project?.client?.name ? '#64dd17' : '#ffffff',
                borderColor: '#2C2C2C',
                whiteSpace: 'nowrap',
              }}
            >
              {item.project?.client?.name || 'Automatically'}
            </TableCell>
            <TableCell sx={{ borderColor: '#2C2C2C', padding: '14px 16px' }}>
              <StyledAutocomplete
```

```
                label="Select Job"
                options={jobList}
                setValue={(value: IJob | null) => setJobHandler(value,
rowIndex)}
                value={item.job}
                width={110}
                textColor="#64dd17"
              />
            </TableCell>
            <CommentCell text={item.comment} index={rowIndex} />
            <TableCell
              sx={{
                color: '#ffffff',
                borderColor: '#2C2C2C',
                borderLeft: '1px solid #4A5878',
                width: '48px',
              }}
            />
            {item.times.map((day, dayIndex) => (
              <WeeklyLogDay key={dayIndex} day={day} rowIndex={rowIndex}
dayIndex={dayIndex} />
            ))}
            <TableCell sx={{ color: '#64DD17', borderColor: '#2C2C2C' }}>
              <div className={styles.timer__time}>
                <span className={styles['timer__time-
text']}>{totalTimeOut(rowIndex)}</span>
                <GreenClockIcon />
              </div>
            </TableCell>
          </TableRow>
        );
      })}
    </>
```

```
  );
};
```

Statistics Page

```
import React, { useEffect, useMemo, useState } from 'react';

import dayjs from 'dayjs';

import styles from './Filter/Statistics.module.scss';

import { saveAs } from 'file-saver';

import { useAppDispatch, useAppSelector } from '../../hooks';

import { format, startOfMonth, endOfMonth } from 'date-fns';

// Components

import { Bar, BarChart, LabelList, ResponsiveContainer, XAxis } from
'recharts';

import { Header, Calendar, ModalSearchItem } from 'components';

import { ReactComponent as DownloadIcon } from 'assets/icons/download-
icon.svg';

import { ReactComponent as FilterIcon } from 'assets/icons/filterIcon.svg';

import { ReactComponent as ActiveFilterIcon } from
'assets/icons/activeFilter.svg';

import StatisticFilterPanel from './Filter/StatisticFilterPanel';

import NoData from '../../assets/images/NoData.jpg';

import StatisticContentHeader from './StatisticContentHeader';

import CustomizedAxisTick from './CustomizedAxisTick';

import { Slide } from '@mui/material';

// Api hooks

import {

  useGetAllStatisticCountsQuery,

  useGetAllStatisticQuery,

  useGetDownLoadExelQuery,

} from 'store/services/StatisticApi';

import { useGetAllUsersQuery } from 'store/services/UserApi';

import { useGetAllProjectsQuery } from 'store/services/ProjectApi';

import {

  clearFilter,

  filterSelector,
```

```
  setActiveFilterIcon,
  setShowFilterModal,
} from '../../store/Statistic/statisticFilterSlice';
// Types
import { ClearFilterType } from '../../store/Statistic/config';
import { IUser } from '../../store/services/types';
import { Range } from 'react-date-range';
import { ModalTypes } from './types';
import { getModalParticipant } from '../../core/utils/getModalParticipant';


export const Statistics = () => {
  const [dateRange, setDateRange] = useState<Range>({
    startDate: startOfMonth(new Date()),
    endDate: endOfMonth(new Date()),
    key: 'selection',
  });
  const [queryString, setQueryString] = useState<string>(
    `?dateFrom=${format(dateRange.startDate || new Date(), 'yyyy-MM-
dd')}&dateTo=${format(
      dateRange.endDate || new Date(),
      'yyyy-MM-dd',
    )}`,
  );
  const [queryStringExelDownLoad, setQueryStringExelDownLoad] = useState('');
  const dispatch = useAppDispatch();
  const [fileExcelName, setFileExcelName] = useState('');
  const { isShowModalFilter, isActiveFilter, selectedUser, selectedProject } =
    useAppSelector(filterSelector);
  const { data: projectList = [] } = useGetAllProjectsQuery();
  const { data: userList = [] } = useGetAllUsersQuery();
  const { data: allCount, refetch: refetchCounts } =
useGetAllStatisticCountsQuery(queryString);
  const [showModalUserDownload, setShowModalUserDownload] = useState(false);
```

```
  const {
    data: dataExel,
    refetch: refetchExedDownLoad,
    isFetching: isFetchingUserExcel,
  } = useGetDownLoadExelQuery(queryStringExelDownLoad, {
    skip: !queryStringExelDownLoad,
  });
  const [searchInputText, setSearchInputText] = useState({
    filterText: '',
    filteredSelectedUsers: '',
    downLoadInputText: '',
  });
  const [selected, setSelected] = useState({
    selectedUsers: selectedUser,
    selectedProject: selectedProject,
  });


  const containerRef = React.useRef(null);


  const {
    data: allStatistic = [],
    refetch: refetchStatistic,
    isLoading,
    isFetching,
  } = useGetAllStatisticQuery(queryString);


  const onClickFilterApply = () => {
    setSelected({ selectedUsers: selectedUser, selectedProject:
selectedProject });
    const usersIds = selectedUser.map((user) => user.id);
    const projectsIds = selectedProject.map((project) => project.id);
    const startMonth = format(dateRange.startDate || new Date(), 'yyyy-MM-
dd');
```

```
    const endMonth = format(dateRange.endDate || new Date(), 'yyyy-MM-dd');


    setQueryString(

`?dateFrom=${startMonth}&dateTo=${endMonth}&userIds=${usersIds}&projectIds=${p
rojectsIds}`,

    );

    refetchCounts();

    refetchStatistic();

    dispatch(setActiveFilterIcon(true));

    dispatch(setShowFilterModal(false));

  };


  const refetchExelDownLoad = () => {

    if (queryStringExelDownLoad) {

      refetchExedDownLoad();

    }

  };

  const onClickDownLoadExel = () => {

    setShowModalUserDownload(true);

  };


  const onClickUserDownLoadExcel = (id: string, name?: string) => {

    const startMonth = dayjs(dateRange.startDate).format('YYYY-MM-DD');

    const endMonth = dayjs(dateRange.endDate).format('YYYY-MM-DD');

    let queryString = '';


    queryString = `?userId=${id}&dateFrom=${startMonth}&dateTo=${endMonth}`;

    setQueryStringExelDownLoad(queryString);

    refetchExelDownLoad();

    setFileExcelName(name + '_' + startMonth + ' - ' + endMonth);

    setShowModalUserDownload(false);

  };
```

```
  const closeExcelDownLoadModal = () => setShowModalUserDownload(false);


  useEffect(() => {
    if (dataExel && !isFetchingUserExcel) {
      saveAs(dataExel, `${fileExcelName}.xls`);
    }


    return () => {
      dispatch(clearFilter(ClearFilterType.All));
      dispatch(setActiveFilterIcon(false));
      dispatch(setShowFilterModal(false));
    };
  }, [isFetchingUserExcel]);


  useEffect(() => {
    !showModalUserDownload &&
      setSearchInputText({ downLoadInputText: '', filterText: '',
filteredSelectedUsers: '' });
  }, [showModalUserDownload]);


  const getFilterUsers = () => {
    if (searchInputText.filterText && !searchInputText.downLoadInputText) {
      return userList.filter((user: IUser) => {
        const fullName = `${user.firstName} ${user.lastName}`;
        return
fullName.toLocaleLowerCase().includes(searchInputText.filterText);
      });
    } else if (searchInputText.downLoadInputText &&
!searchInputText.filterText) {
      return userList.filter((user: IUser) => {
        const fullName = `${user.firstName} ${user.lastName}`;
        return
fullName.toLocaleLowerCase().includes(searchInputText.downLoadInputText);
```

```
      });
    } else {
      return userList;
    }
  };

  const filteredUsers = useMemo(() => {
    return getFilterUsers();
  }, [userList, searchInputText.filterText]);

  const filteredDonwloadUsers = useMemo(() => {
    const filteredUsers = getFilterUsers();
    return getModalParticipant(ModalTypes.DOWNLOAD_REPORT, [], filteredUsers);
  }, [userList, searchInputText.downLoadInputText]);

  const statisticHours = useMemo(() => {
    return allStatistic.map((item) => {
      const fullName = item.firstName + ' ' + item.lastName;
      return {
        name: item?.name
          ? item.name
          : item?.firstName
            ? fullName
            : dayjs(item.date).format('DD.MM.YYYY'),
        hours: +item.hours,
        title: item.minutes ? `${item.hours} h ${item.minutes} m` :
`${item.hours} h`,
        imageUrl: item.url,
      };
    });
  }, [allStatistic]);
  const isShowInfo = useMemo(() => {
```

```
    return (
      selected.selectedUsers.length === 1 && selected.selectedProject.length
=== 1 && isActiveFilter
    );
  }, [selectedUser, selectedProject, selected.selectedUsers]);
  const isShowNoDataImg = useMemo(() => {
    return !isLoading && !statisticHours.length;
  }, [statisticHours]);
  return (
    <div className={styles.statistics}>
      <Header title="Statistics">
        <div className={styles.tools}>
          <Calendar dateRange={dateRange} setDateRange={setDateRange} />
          <div className={styles.header__right}>
            <div
              className={styles.tools__report}
              onClick={() => dispatch(setShowFilterModal(true))}
            >
              <p className={styles['tools-filter-title']}>Filter</p>
              {isActiveFilter ? (
                <ActiveFilterIcon className={styles['tools__filter__active--
icon']} />
              ) : (
                <FilterIcon className={styles.tools__filter__icon} width={13}
height={12} />
              )}
            </div>
            <div
              className={styles.tools__report + ' ' +
styles.tools__report__download}
              onClick={onClickDownLoadExel}
            >
              <p
                className={{
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ ОБЛІКУ РОБОЧОГО ЧАСУ ПРАЦІВНИКІВ

Виконав: студент 2-го курсу,
групи ЗКН-23м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Валявський О.С.
(прізвище та ініціали)

Керівник: к.т.н., ас. каф. КН
Богач І. В.
(прізвище та ініціали)
«» 05. 12. 2024 р.

Вінниця ВНТУ - 2024 рік

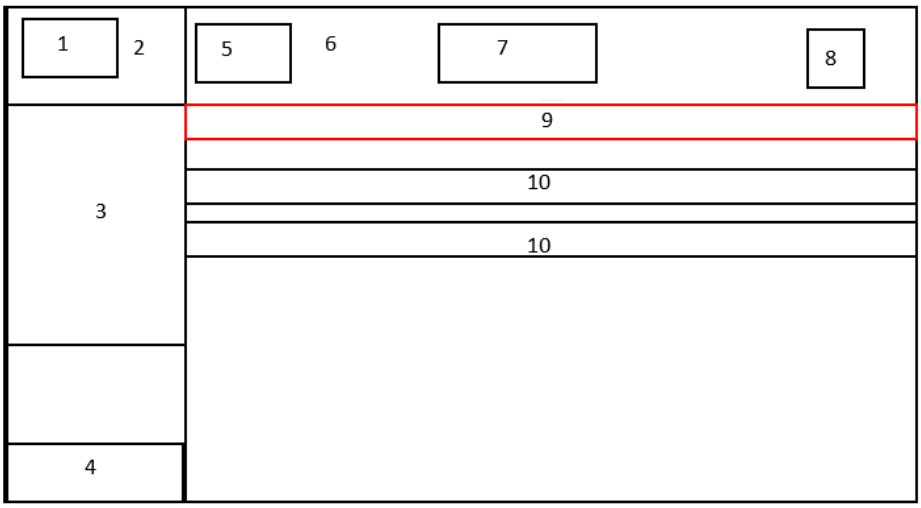

Рисунок В 1.1 – Структурна схема головного вікна додатку

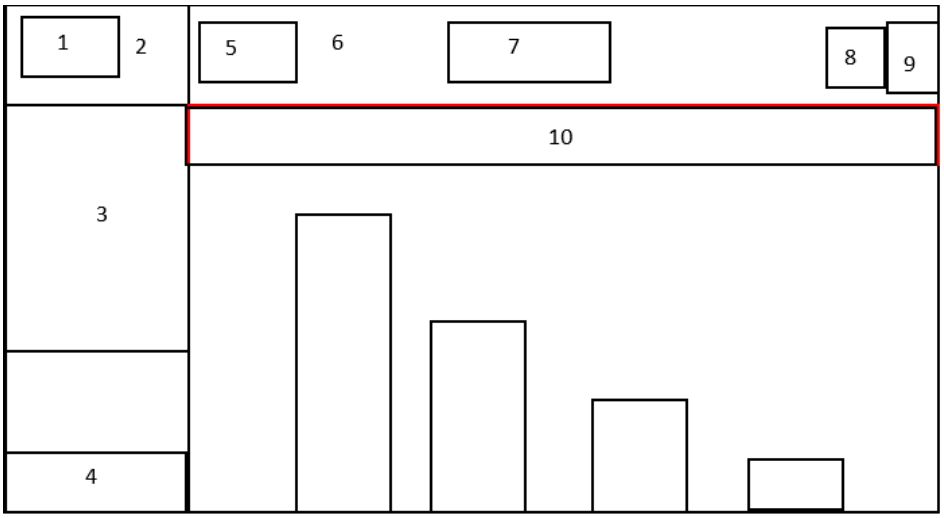

Рисунок В 1.2 – Структурна схема головного вікна додатку

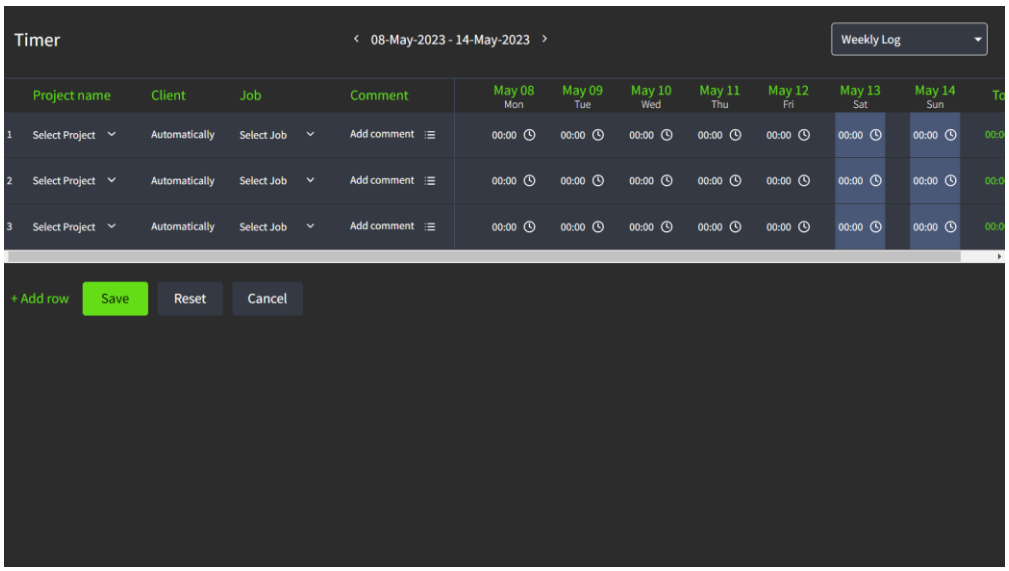

Рисунок В 1.3 – Weekly log

Рисунок В 1.4 – Сторінка таймер

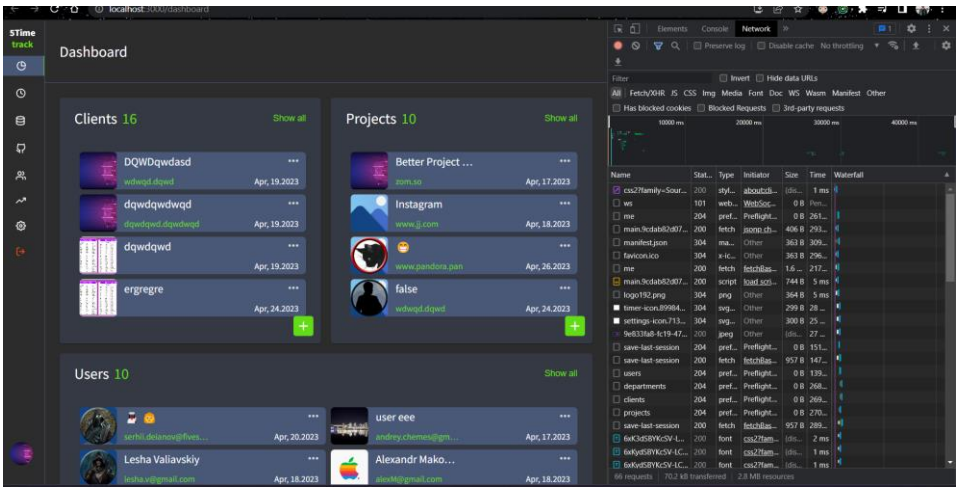

Рисунок В 1.5 – Завантаження сторінки

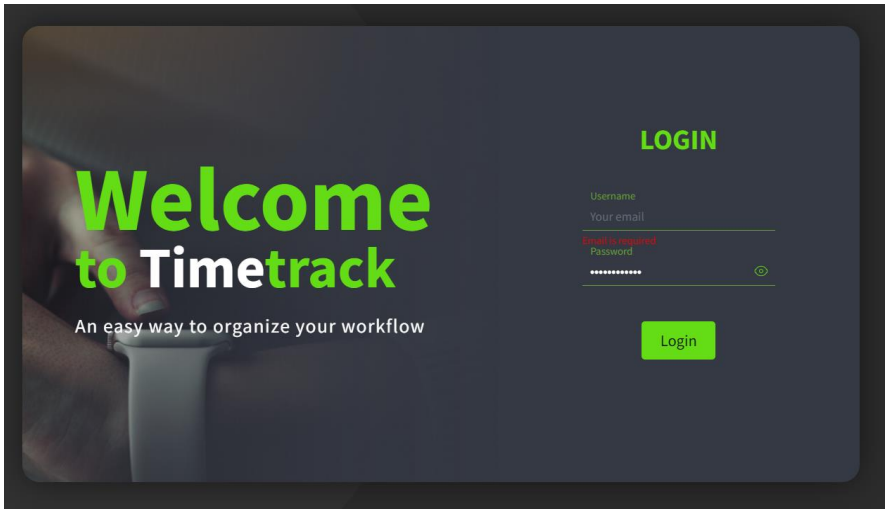

Рисунок В 1.6 – Реєстрація користувача

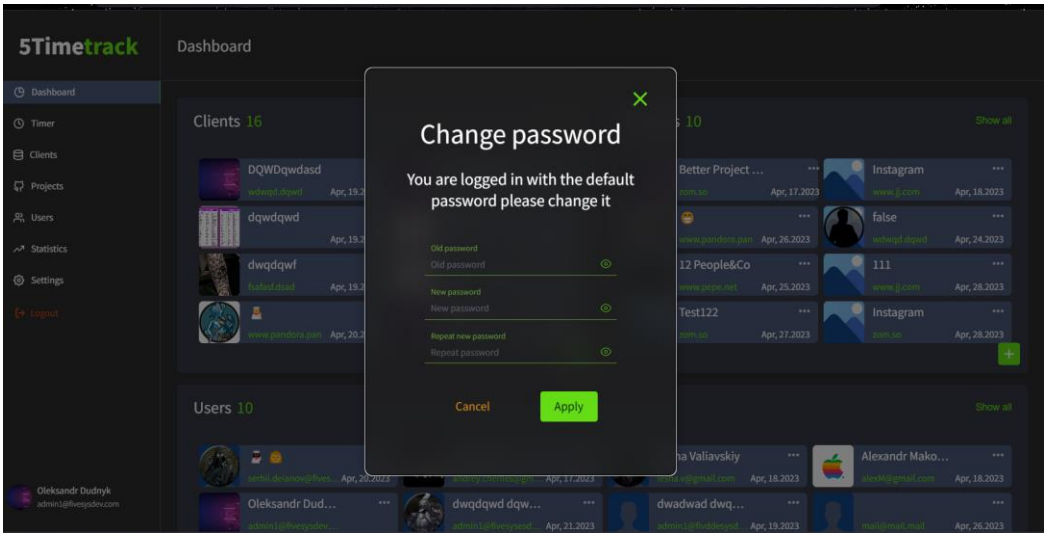

Рисунок В 1.7 – Результат успішної реєстрації користувача

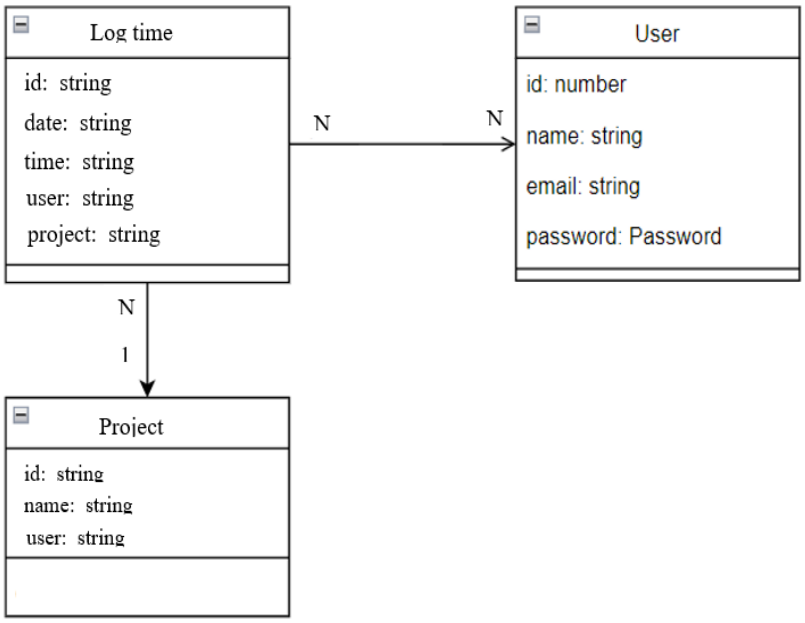

Рисунок В 1.8 – Діаграма класів

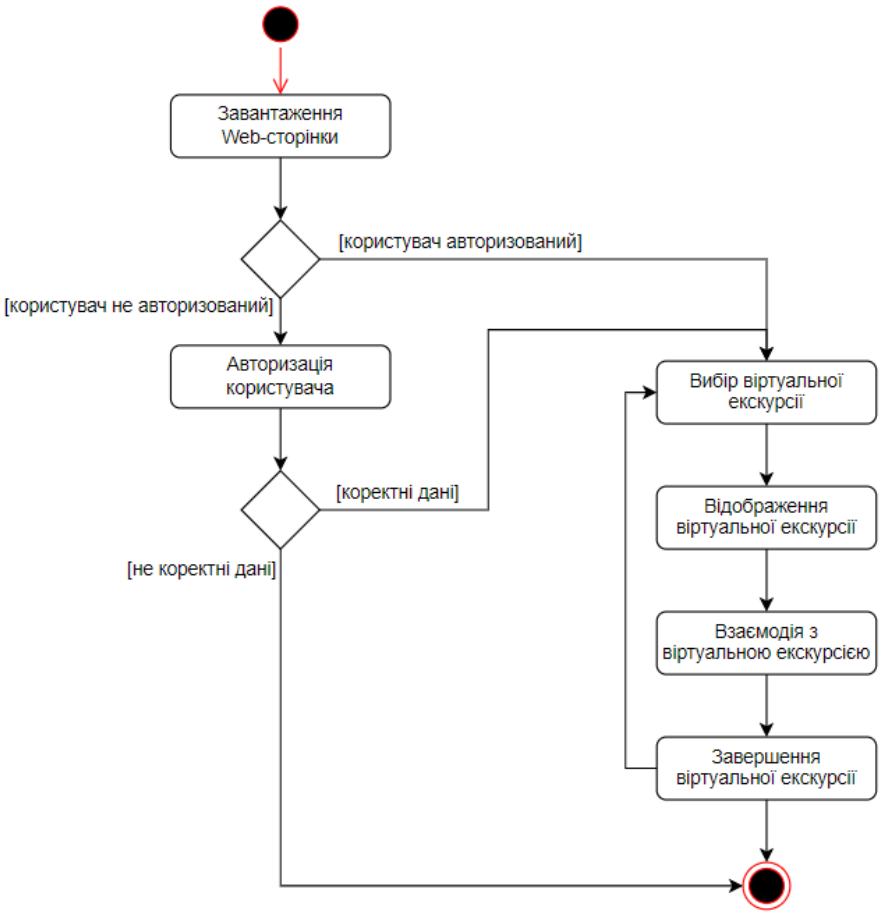

Рисунок В 1.9 – Діаграма діяльності

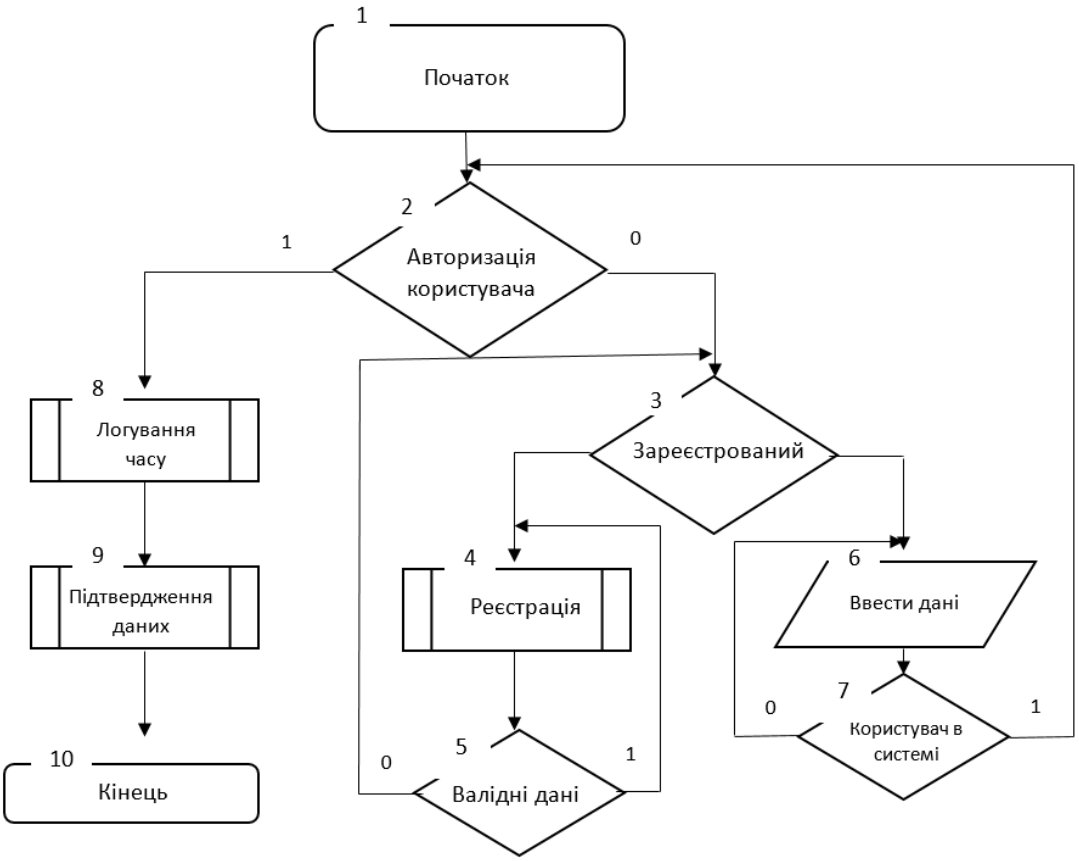

Рисунок В 1.10 – Загальний алгоритм додатку