НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для автоматизованої системи табелювання співробітників

Виконав студент IV курсу, групи ІТ-01
(шифр групи)

Філоненко Владислав Павлович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник асистент Очеретяний О.К. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент ст. викладач кафедри ОТ, Алещенко О. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Філоненку Владиславу Павловичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для автоматизованої системи табелювання співробітників

керівник проєкту Очеретяний Олександр Костянтинович, асистент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_27_»квітня 2024 р. №2112-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Предпроєктне обстеження предметної області: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Розроблення вимог до програмного забезпечення: варіанти використання програмного забезпечення, аналіз системних вимог, розроблення функціональних та нефункціональних вимог.
3) Конструювання та розроблення програмного забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення.
4) Аналіз якості та тестування програмного забезпечення: аналіз якості, опис процесів тестування, опис контрольного прикладу.
5) Розгортання та супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема бази даних _____

3) Схема структурна класів програмного забезпечення _____

_____

_____

_____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «11» березня 2024 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв’язання задачі	24.03.2024	
3	Постановка та формалізація задачі	19.04.2024	
4	Розробка інформаційного забезпечення	30.04.2024	
5	Алгоритмізація задачі	10.05.2024	
6	Обґрунтування вибору використаних технічних засобів	14.05.2024	
7	Розробка програмного забезпечення	20.05.2024	
8	Налагодження програми	25.05.2024	
9	Виконання графічних документів	26.05.2024	
10	Оформлення пояснювальної записки	29.05.2024	
11	Подання ДП на попередній захист	03.06.2024	
12	Подання ДП рецензенту	05.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Владислав ФІЛОНЕНКО

(підпис) (ініціали, прізвище)

Керівник _____ Олександр ОЧЕРЕТЯНИЙ

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’ятьох розділів, містить 43 таблиць, 23 рисунків та 33 джерел – загалом 62 сторінки.

Дипломний проєкт присвячений розробці вебзастосунку для автоматизованої системи табелювання співробітників.

Мета дипломного проєкту є підвищення точності та мінімізація підробки табелювання співробітників, за рахунок розробки автоматизованої системі відслідковування працівників із використанням обробки відео камер відеоспостереження.

Об'єкт дослідження: розробка вебзастосунку для автоматизованої системи табелювання співробітників.

Предмет дослідження: вебзастосунок для автоматизованої системи табелювання співробітників, його функціональні можливості, тегнології та алгоритми розробки.

У розділі предпроєктне обстеження предметної області розглянуто предметну область, існуючих рішень програмних продуктів та було описані бізнес-процеси.

У розділі розроблення вимог до програмного забезпечення розглянуто варіанти використання програмного забезпечення, аналіз системних вимог та описані функціональні та нефункціональні вимоги.

У розділі конструювання та розроблення програмного забезпечення розглянуто розроблення архітектури програмного забезпечення та структура бази даних, а також проведений аналіз безпеки даних продукту.

У розділі аналіз якості та тестування програмного забезпечення було проведено оцінку якості кодової бази та здійснено комплексне тестування всього програмного забезпечення.

У розділі розгортання та супровід програмного забезпечення було проведено ефективне впровадження та підтримку вебзастосунку, використовуючи Docker Hub.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, АВТОМАТИЗАЦІЯ, ТАБЕЛЮВАННЯ, БАЗА ДАНИХ, АРХІТЕКТУРА.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 43 tables, 23 figures and 33 sources – in total 62 pages.

The diploma project is dedicated to the development of a web application for an automated employee attendance tracking system.

The purpose of the diploma project is to increase accuracy and minimize the possibility of falsifying employee attendance records by developing an automated tracking system that uses video surveillance camera footage processing.

Object of the research: Development of a web application for an automated employee attendance tracking system.

Subject of the research: Web application for an automated employee attendance tracking system, its functional capabilities, technologies, and development algorithms.

In the section on pre-project survey of the subject area, the subject area, existing software solutions, and business processes were examined.

In the section on development of software requirements, various usage scenarios, system requirements analysis, and functional and non-functional requirements were described.

In the section on software design and development, the software architecture and database structure were reviewed, and an analysis of the product's data security was conducted.

In the section on quality analysis and software testing, an assessment of the codebase quality was performed, and comprehensive testing of the entire software was carried out.

In the section on software deployment and maintenance, effective implementation and support of the web application using Docker Hub were carried out.

KEYWORDS: WEB APPLICATION, AUTOMATION, ATTENDANCE TRACKING, DATABASE, ARCHITECTURE.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-0320.045440.01.91	Технічне завдання	13	
3	А4	КПІ.ІТ-0320.045440.02.81	Пояснювальна записка	62	
4	А4	КПІ.ІТ-0320.045440.03.12	Текст програми	16	
5	А4	КПІ.ІТ-0320.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-0320.045440.05.34	Керівництво користувача	9	
7	А3	КПІ.ІТ-0320.045440.06.99	Графічний матеріал	4	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-0320.045440.00.90		
Розроб.		Філоненко В.П.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Очеретяний О.К.					
Н.контр.		Вітковська І.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-01	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТАБЕЛЮВАННЯ СПІВРОБІТНИКІВ

Технічне завдання

КПІ.ІТ-0320.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олександр ОЧЕРЕТЯНИЙ

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Владислав ФІЛОНЕНКО

Київ – 2024

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок для автоматизованої системи табелювання співробітників.

Галузь застосування:

Наведене технічне завдання поширюється на розробку вебзастосунку для автоматизованої системи табелювання співробітників [WebTimeTracker], котра використовується для збереження і керування інформацією про робочий час співробітників та призначена для автоматизації табелювання та оптимізації процесу управління персоналом в різних галузях, включаючи компанії, установи, та організації будь-якого розміру.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки вебзастосунок для автоматизованої системи табелювання співробітників є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації з табелювання співробітників і може використовуватися для будь яких компаній.

Метою розробки є підвищення точності та мінімізація підробки табелювання співробітників, за рахунок розробки автоматизованої системі відслідковування працівників із використанням обробки відео камер відеоспостереження.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– Форма авторизації користувача (рисунок 4.1);

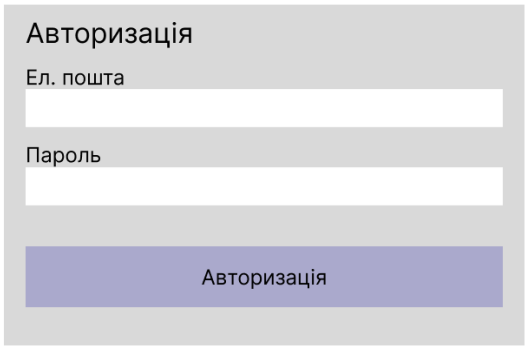


Рисунок 4.1 – Форма авторизації користувача

– Форма створення профілю робітника та список вже існуючих профілів працівників (рисунок 4.2).

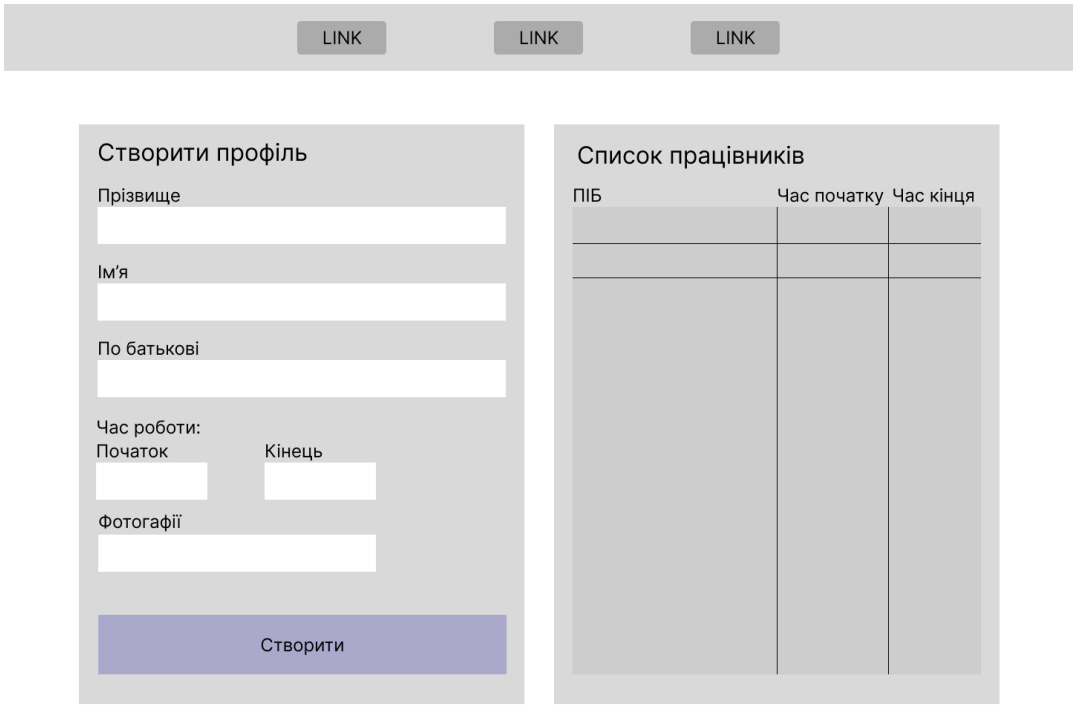


Рисунок 4.2 – Сторінка створення профілю робітника

– Сторінка профілю робітника та можливість редагувати дані і додавати нові фото (рисунок 4.3).

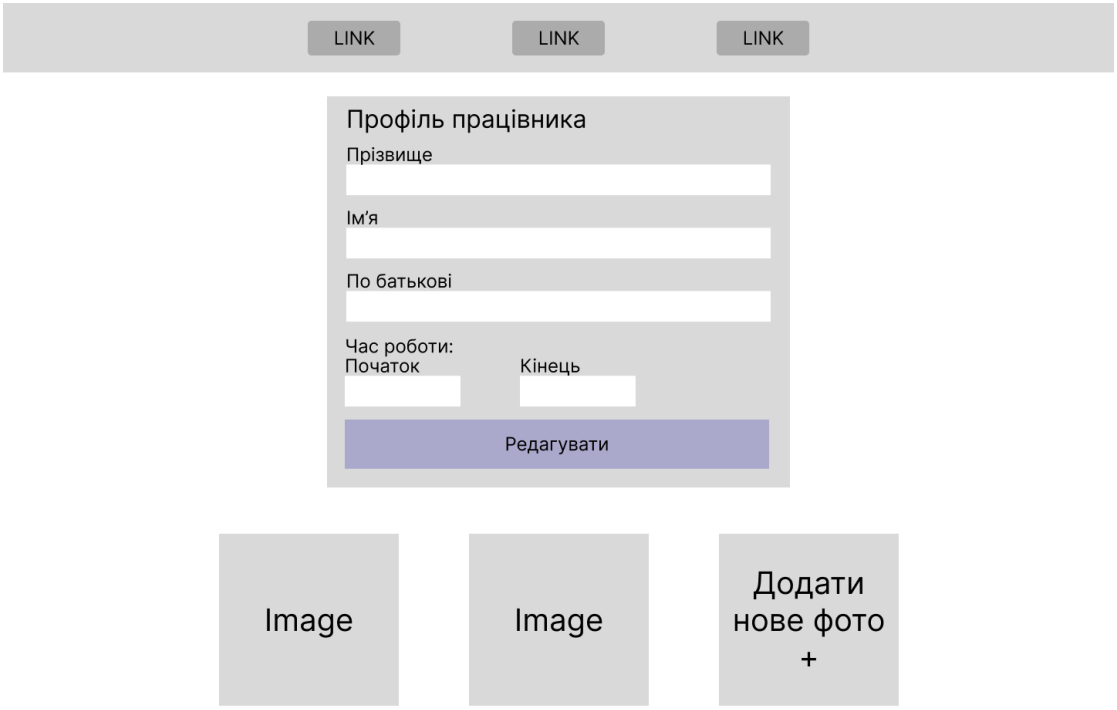


Рисунок 4.3 – Сторінка профілю робітника

– Сторінка для перегляду звіту виявлених працівників на відео та можливість фільтрувати звіт. Також завантажити Excel файл (рисунок 4.4).

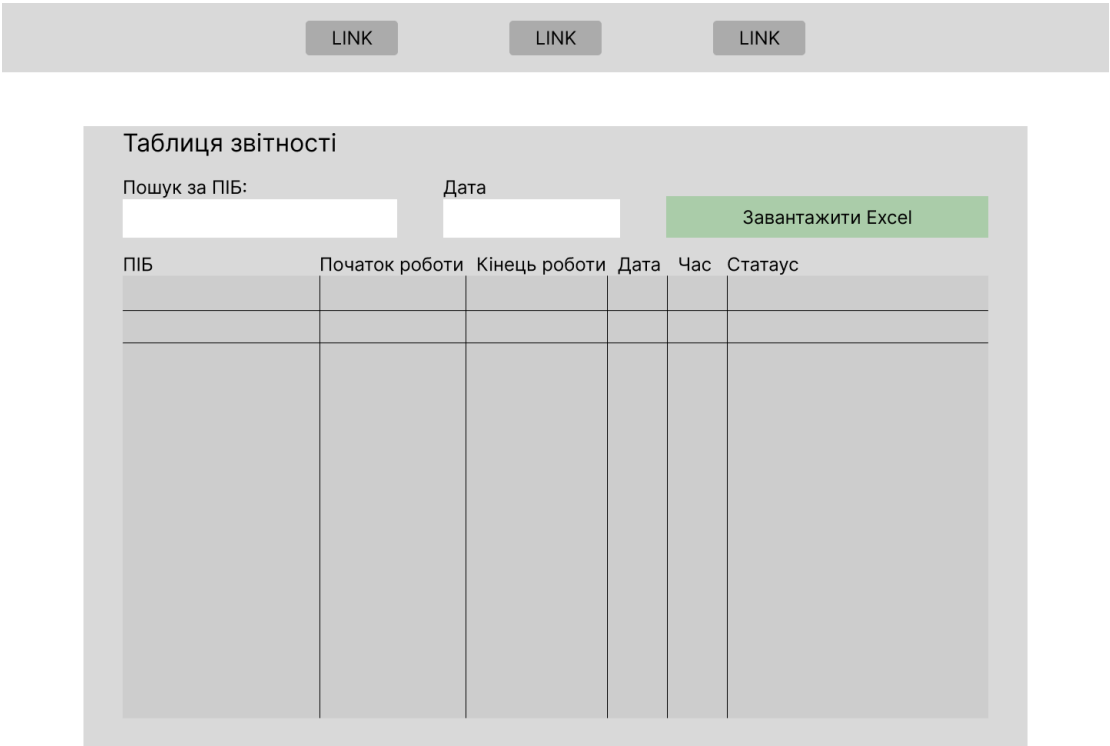


Рисунок 4.4 – Сторінка звіту

4.1.2 Для користувача:

– авторизуватися у системі;
– створювати, редагувати та видаляти співробітників;
– переглядати дані про співробітника;
– переглядати звіт;
– можливість фільтрувати та сортувати звіт;
– вивантажувати звіти в форматі Excel файлу;
– вихід з облікового запису;

4.1.3 Додаткові вимоги:

– Додаткові вимоги не передбачені.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на сучасних веб-браузерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: з частотою від 1 ГГц;

– об‘єм ОЗП: 512 Мб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати у браузерах Google Chrome, Mozilla Firefox та Microsoft Edge.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: .jpg, .png.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: .xlsx.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування JavaScript та середовища Node.js для серверної розробки та React.js для клієнтської розробки.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Visual Studio Code.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді репозиторію на GitHub.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема бази даних;
– схема структурна класів програмного забезпечення;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	11.03	
2.	Аналіз існуючих методів розв’язання задачі	24.03	Результат обраних методів
3.	Постановка та формалізація задачі	19.04	План розробки функціональних завдань
4.	Розробка інформаційного забезпечення	30.04	Інформаційне завдання
5.	Алгоритмізація задачі	10.05	Вибрані алгоритми розробки
6.	Обґрунтування вибору використаних технічних засобів	14.05	Вибрані технології розробки
7.	Розробка програмного забезпечення	20.05	Тексти програмного забезпечення
8.	Налагодження програми	25.05	Тести, результати тестування
9.	Виконання графічних документів	26.05	Графічний матеріал проекту
10.	Оформлення пояснювальної записки	29.05	Пояснювальна записка

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Вебзастосунок для автоматизованого табелювання співробітників**

КПІ.ІТ-0320.045440.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
ER	–	Entity-Relation diagram
БД	–	База даних.
ШІ	–	Штучний інтелект
HTTP	–	HyperText Transfer Protocol
SQL	–	Structured query language
UI	–	User interface

ВСТУП

Актуальність автоматизованих систем табелювання співробітників значно зросла в умовах сучасного цифрового світу. Ці системи дозволяють суттєво підвищити ефективність обліку робочого часу, знизити ймовірність помилок та забезпечити точний аналіз відвідуваності співробітників. Завдяки штучному інтелекту, набуває можливість аналізувати відео, для автоматизації процесу відслідковування людей, що підвищує точність даних.

Світові тенденції у вирішенні проблем автоматизації обліку робочого часу демонструють зростаючу інтеграцію технологій штучного інтелекту та машинного навчання. Провідні компанії використовують технології розпізнавання обличчя для ідентифікації співробітників та контролю їх присутності на робочому місці. Ці рішення дозволяють значно підвищити рівень безпеки та забезпечити більш точний облік робочого часу.

Сучасний стан розробки таких систем свідчить про активне впровадження технологій розпізнавання обличчя та аналізу відео. Провідні наукові установи та організації, такі як Google, Microsoft та Amazon, активно досліджують та вдосконалюють технології машинного навчання та комп'ютерного зору. Вчені та фахівці з цих установ досягли значних успіхів у розробці алгоритмів для точного та швидкого розпізнавання обличчя в реальному часі.

Можливі сфери застосування розробленої системи автоматизованого табелювання співробітників включають офісні приміщення, виробничі підприємства, навчальні заклади та інші організації, де необхідний точний облік робочого часу. Система може використовуватися як для внутрішніх потреб організації, так і в якості комерційного продукту, що пропонується іншим компаніям для підвищення ефективності їх роботи.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Предметна область автоматизованого табелювання співробітників є актуальною та важливою для сучасних підприємств незалежно від їх розміру та галузі діяльності. Основна мета автоматизації табелювання полягає в оптимізації процесів обліку робочого часу співробітників, збільшенні ефективності роботи та зменшенні ризику помилок.

В сучасному світі, де інформаційні технології набувають все більшого значення, автоматизоване табелювання стає необхідністю для підприємств будь-якої форми власності. Використання таких систем дозволяє ефективно відслідковувати робочий час співробітників, уникнути помилок у розрахунках оплати праці, а також забезпечує зручний доступ до статистичних даних щодо робочого часу та відпусток.

На сьогоднішній день багато підприємств використовують застарілі методи ручного табелювання, що призводить до значних затрат часу та можливих помилок. Це також може призвести до несправедливості у розподілі робочого часу та невідповідності нормативам та законодавству щодо робочого часу.

Одним з можливих шляхів вирішення цих проблем є впровадження сучасних веб-застосунків для автоматизованого табелювання співробітників. Такі застосунки можуть включати в себе функції відслідковування робочого часу, планування відпусток, генерації звітів та аналізу продуктивності. Вони також можуть бути легко інтегровані з існуючими системами управління персоналом та обліку робочого часу[1].

У рамках дипломного проєкту ми обрали шлях розробки веб-застосунку, який буде надійним, ефективним та простим у використанні. Наш застосунок буде забезпечувати автоматизоване табелювання співробітників з максимальною точністю та ефективністю, що сприятиме підвищенню продуктивності та оптимізації управління персоналом на підприємстві.

1.2 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації вебзастосунку для автоматизованого табелювання співробітників. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.2.1 Аналіз відомих програмних продуктів

На ринку існує кілька програмних продуктів, що частково чи повністю реалізують функціонал автоматизованого табелювання співробітників. Нижче наведено аналіз деяких з них.

Toggl Track - Відомий інструмент для відстеження часу, що дозволяє вести табелі робочого часу, генерувати звіти та інтегруватися з іншими системами[2].

Clockify - Безкоштовний інструмент для трекінгу часу, який пропонує аналогічний функціонал із можливістю створення та управління табелями робочого часу, генерування звітів і інтеграції з різними сторонніми сервісами[3].

Hubstaff - Інструмент для відстеження часу, що пропонує додаткові функції, такі як моніторинг активності, геолокаційний трекінг та можливість створення детальних звітів для аналізу продуктивності[4].

Для порівняння проєкту з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Дипломний проєкт	Toggl Track	Clockify	Hubstaff	Пояснення
Відстеження робочого часу	Так	Так	Так	Так	Продукти мають функціонал відстеження робочого часу.

Продовження таблиці 1.1

Функціонал	Дипломний проєкт	Toggl Track	Clockify	Hubstaff	Пояснення
Розпізнавання облич	Так	Ні	Ні	Ні	Дипломний проєкт включає функціонал розпізнавання облич для верифікації користувачів.
Створення профілів робітників	Так	Ні	Ні	Ні	Дипломний проєкт дозволяє зберігати ПІБ, час початку і кінця роботи та фотографії для розпізнавання.
Веб-інтерфейс	Так	Так	Так	Так	Продукти мають веб-інтерфейс для користувачів.
Звіти у форматі Excel	Так	Так	Так	Так	Продукти мають функціонал для генерування звітів у різних форматах, включаючи Excel.

Кінець таблиці 1.1

Функціонал	Дипломний проєкт	Toggl Track	Clockify	Hubstaff	Пояснення
Моніторинг активності	Ні	Ні	Ні	Так	Hubstaff пропонує додаткові функції моніторингу активності, які відсутні в інших продуктах.
Геолокаційний трекінг	Ні	Ні	Ні	Так	Hubstaff має унікальну функцію геолокаційного трекінгу, що не реалізовано у дипломному проєкті.

Дипломний проєкт виділяється серед інших рішень завдяки можливості створення детальних профілів робітників з фотографіями для розпізнавання облич. Це надає додатковий рівень безпеки та точності при табелюванні робочого часу. Тоді як такі продукти, як Toggl Track, Clockify і Hubstaff, забезпечують базовий функціонал для відстеження часу і генерування звітів, але не пропонують таких розширених можливостей для управління профілями робітників і розпізнавання обличчя.

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

Для реалізації автоматизованого табелювання співробітників буде використовуватися клієнт-серверна архітектура[5].

Клієнт-серверна архітектура розділяє систему на клієнтську (фронтенд) і серверну (бекенд) частини. Клієнт забезпечує взаємодію з користувачем, а сервер обробляє запити і взаємодіє з базою даних. Цей підхід забезпечує чітке розділення обов'язків, масштабованість та безпеку. Основні недоліки включають залежність від мережі та можливі затримки.

Таким чином, клієнт-серверна архітектура підходить для проектів, де необхідно розділити інтерфейс користувача і логіку обробки даних, забезпечуючи при цьому масштабованість і безпеку.

1.3 Опис бізнес-процесів

Для опису бізнес процесу використовується BPMN модель (рис. 1.1 – 1.3).

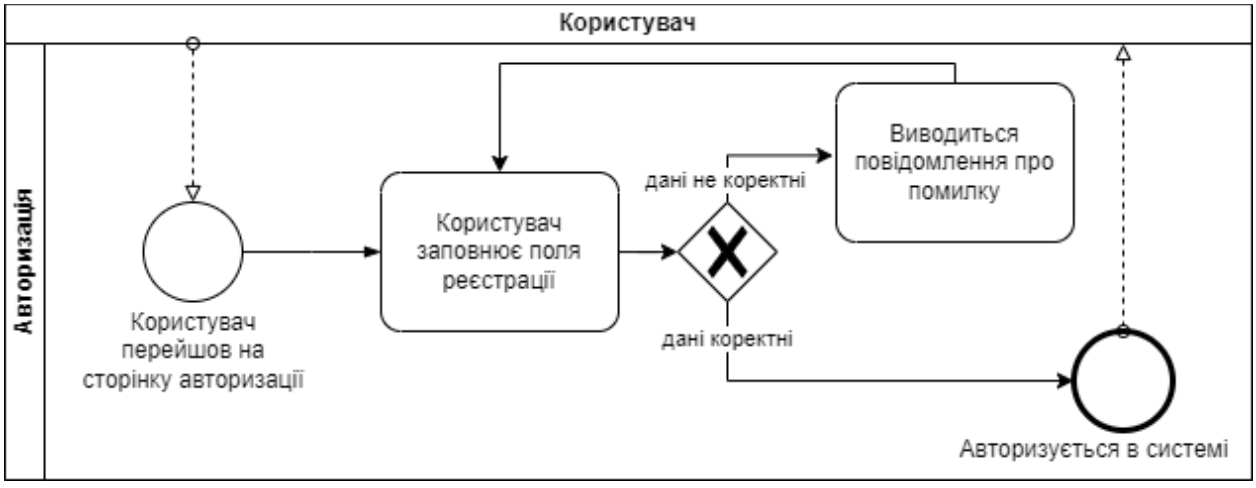


Рисунок 1.1 – Схема авторизації

Опис послідовності авторизації облікового запису користувача:

– користувач переходить на сторінку авторизації;
– користувач заповнює поля авторизації;

– якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою;

– якщо введені поля, відповідають шаблону заповнення на клієнтській стороні, користувач входить у систему.

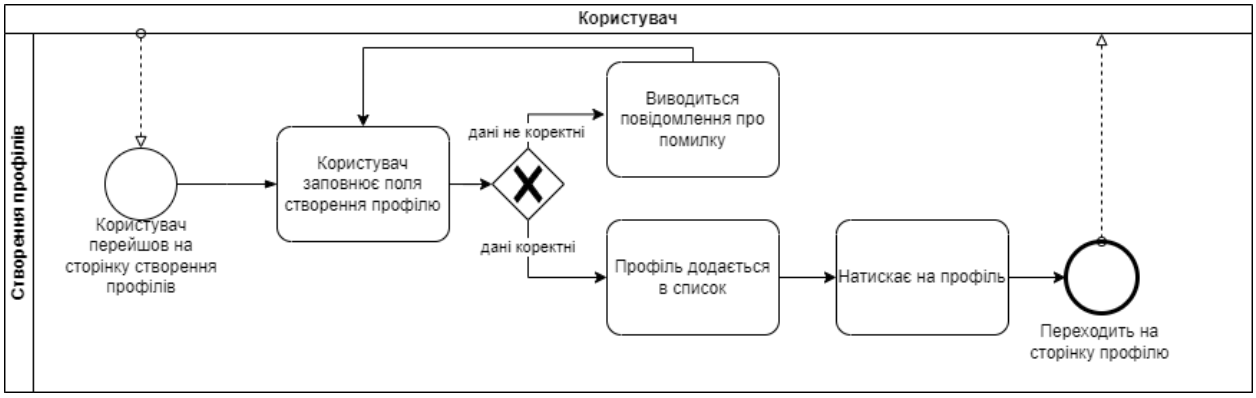


Рисунок 1.2 – Схема створення профілю

Опис послідовності створення профілю людей для табелювання:

– користувач переходить на сторінку створення профілю;

– користувач заповнює поля створення профілю;

– якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою.

– якщо введені поля, відповідають шаблону заповнення на клієнтській стороні, створюється профіль та додається до списку вже існуючих профілів;

– коли користувач натискає на профіль у списку, він переходить на сторінку з описом профілю.

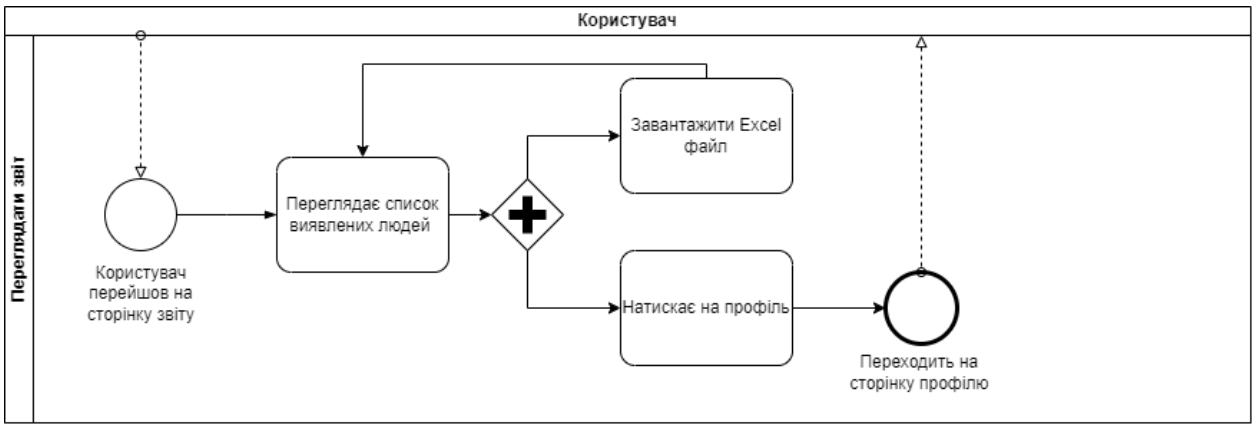


Рисунок 1.3 – Схема перегляду звіту

Опис послідовності перегляду звіту виявлених людей:

– користувач переходить на сторінку звіту;

– коли користувач натискає на профіль у списку, він переходить на сторінку з описом профілю;

– користувач завантажує звіт в форматі Excel файлу.

1.4 Постановка задачі

Метою дипломного проєкту є підвищення точності та мінімізація підробки табелювання співробітників, за рахунок розробки автоматизованої системі відслідковування працівників із використанням обробки відео камер відеоспостереження.

Завдяки нашій системі, користувач, може створювати профілі робітників, яких потрібно виявляти на відео, заповнивши їхню інформацію та завантаживши фотографії обличчя людини.

На виході можна переглядати список людей, які були виявлені на відео та час, коли було помічено. Також, завантажити звіт у форматі Excel файлу.

Висновки до розділу

У цьому розділі було проведено ґрунтовний аналіз предметної області автоматизованого табелювання співробітників, що є актуальним для сучасних підприємств незалежно від їх розміру та галузі діяльності. Основною метою автоматизації табелювання є оптимізація процесів обліку робочого часу,

підвищення ефективності роботи та зниження ризику помилок. Використання таких систем дозволяє підприємствам ефективно відстежувати робочий час співробітників, уникати помилок у розрахунках оплати праці та забезпечувати зручний доступ до статистичних даних.

Було також проведено аналіз існуючих рішень у даній області. Зокрема, розглянуто програмні продукти, такі як Toggl Track, Clockify та Hubstaff, які забезпечують базовий функціонал для відстеження часу та генерації звітів. Однак, ці продукти не пропонують розширених можливостей для управління профілями робітників та розпізнавання обличчя, що виділяє наш дипломний проєкт серед інших.

Розглянуто технічні рішення, зокрема, клієнт-серверну архітектуру, яка розділяє систему на клієнтську та серверну частини. Такий підхід забезпечує чітке розділення обов'язків, масштабованість та безпеку, хоча й має певні недоліки, як-от залежність від мережі та можливі затримки.

Для більш детального розуміння бізнес-процесів були створені BPMN моделі, які описують послідовності авторизації, створення профілю та перегляду звіту. Це забезпечує наочне уявлення про роботу системи та її взаємодію з користувачем.

Таким чином, у цьому розділі було проведено всебічний аналіз предметної області та існуючих рішень, визначено технічні підходи та бізнес-процеси, що забезпечує фундамент для подальшої розробки ефективного веб-застосунку для автоматизованого табелювання співробітників.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є автоматизоване табелювання співробітників завдяки ШІ, який аналізує відео. Також, можна створювати профілі співробітників, яких потрібно виявляти на відео та переглядати звіт про відвідування співробітників, більше функцій можна побачити у графічних матеріалів, креслення 1.

В таблицях 2.1 – 2.7 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 - Варіант використання UC-1

Use case name	Авторизація користувача
Use case ID	UC-01
Goals	Авторизація існуючого користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зайти в систему
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку авторизації. В поля для авторизації вводяться відповідні дані: логін користувача, пароль в системі. Після заповнення даних користувача натискає кнопку авторизації. Після цього користувач перенаправляється на сторінку звіту.
Extension	В випадку введення не коректних даних, кнопка авторизації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом.
Post-Condition	Вхід у систему користувача, перехід на сторінку звіту

Таблиця 2.2 - Варіант використання UC-2

Use case name	Вихід користувача
Use case ID	UC-01
Goals	Вихід користувача з системи
Actors	Користувач
Trigger	Користувач бажає вийти з системи
Pre-conditions	-
Flow of Events	Користувач знаходиться на любій сторінці вебзастосунку. В меню натискається кнопка виходу. Після натиснення на кнопку, користувача перенаправляє на сторінку авторизації.
Extension	-
Post-Condition	Вихід з системи користувача, перехід на сторінку авторизації

Таблиця 2.3 - Варіант використання UC-3

Use case name	Створення профіля співробітника
Use case ID	UC-03
Goals	Створити новий профіль співробітника
Actors	Користувач
Trigger	Користувач бажає створити новий профіль співробітника
Pre-conditions	-
Flow of Events	Користувач знаходиться на сторінці списку співробітників. В поля для створення вводяться відповідні дані: прізвище, ім’я, по батькові та обираються фотографії обличчя людини. Після натиснення на кнопку, виводиться повідомлення про успішно створеного профілю та додається до списку вже існуючих профілів.
Extension	-
Post-Condition	Створення нового профілю, оновлення списку

Таблиця 2.4 - Варіант використання UC-4

Use case name	Редагування профіля співробітника
Use case ID	UC-04
Goals	Редагувати профіль співробітника
Actors	Користувач
Trigger	Користувач бажає виправити або додати дані співробітника
Pre-conditions	-
Flow of Events	Користувач знаходиться на сторінці профілю співробітника. Після натиснення на кнопку редагувати, з'являється форма. В поля для редагування вводяться відповідні дані: прізвище, ім'я, по батькові, час початку роботи та кінець. Після натиснення на кнопку, виводиться повідомлення про успішно оновлені дані.
Extension	-
Post-Condition	Оновлення даних профілю

Таблиця 2.5 - Варіант використання UC-5

Use case name	Видалення профіля співробітника
Use case ID	UC-05
Goals	Видалити профіль співробітника
Actors	Користувач
Trigger	Користувач бажає видалити профіль співробітника
Pre-conditions	-
Flow of Events	Користувач знаходиться на сторінці профілю співробітника. Після натиснення на кнопку видалення профілю, виводиться повідомлення про успішно видалений профіль.
Extension	-
Post-Condition	Видалення профілю, переходить на сторінку списку співробітників

Таблиця 2.6 - Варіант використання UC-6

Use case name	Переглядати звіт
Use case ID	UC-06
Goals	Переглядати звіт про відвідування
Actors	Користувач
Trigger	Користувач бажає переглянути звіт про відвідування співробітників
Pre-conditions	-
Flow of Events	Користувач знаходиться на сторінці звіту. Можливість переглядати таблицю з часом відвідування співробітників. Також можна фільтрувати та сортувати данні. Після натиснення на рядок в таблиці, користувача перенаправляє на профіль співробітника.
Extension	-
Post-Condition	Переглядати звіт, переходить на сторінку профілю співробітника

Таблиця 2.7 - Варіант використання UC-7

Use case name	Завантажити звіт
Use case ID	UC-07
Goals	Завантажити звіт про відвідування
Actors	Користувач
Trigger	Користувач бажає завантажити звіт про відвідування співробітників
Pre-conditions	-
Flow of Events	Користувач знаходиться на сторінці звіту. Після натиснення на кнопку завантаження звіту, завантажується Excel файл на комп’ютер користувача
Extension	-
Post-Condition	Завантажується звіт в форматі Excel файлу

2.2 Аналіз системних вимог

Програмне забезпечення повинно функціонувати на сучасних веб-браузерах (Google Chrome, Mozilla Firefox та Microsoft Edge).

Мінімальна конфігурація технічних засобів:

– тип процесору: з частотою від 1 ГГц;

– об‘єм ОЗП: 512 Мб;

– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів < (та на якій виконувалась розробка) >:

– тип процесору: Intel Core i5;

– об‘єм ОЗП: 16 Гб;

– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

Також для серверної частини потрібно встановити Python [5], тому що, бібліотека яка використовується для ШІ, потребує його.

2.3 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.8 наведено загальну модель вимог, а в таблицях 2.9 – 2.26 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.1.

Таблиця 2.8 – Загальна модель вимог

№	Назва	ID вимоги	Пріоритети	Ризики
1	Система авторизації користувача.	FR-1	Високий	Високий
1.1	Авторизація користувача.	FR-2	Високий	Високий
1.1.1	Ввід пошти.	FR-3	Середній	Високий
1.1.2	Ввід паролю.	FR-4	Середній	Високий

Продовження таблиці 2.8

№	Назва	ID вимоги	Пріоритети	Ризики
1.2	Вихід із акаунту.	FR-5	Середній	Низький
2	Перегляд профілів співробітників	FR-6	Середній	Високий
2.1	Створити профіль.	FR-7	Високий	Високий
2.1.1	Ввести ПІБ.	FR-8	Середній	Низький
2.1.2	Ввести час періоду роботи.	FR-9	Середній	Низький
2.1.3	Завантажити фото.	FR-10	Середній	Високий
2.2	Редагувати профіль.	FR-11	Середній	Високий
2.2.1	Ввести нові дані.	FR-12	Середній	Низький
2.2.2	Додати нові фото.	FR-13	Високий	Високий
2.2.3	Видалити фото.	FR-14	Високий	Високий
3	Перегляд звіту.	FR-15	Високий	Низький
3.1	Завантажити звіт.	FR-16	Високий	Низький
3.2	Фільтрувати звіт.	FR-17	Середній	Низький
3.3	Сортувати звіт.	FR-18	Середній	Низький

Таблиця 2.9 – Функціональна вимога FR-1

Назва	Система авторизації користувача.
Опис	Неавторизований користувач має можливість авторизуватися у системі.

Таблиця 2.10 – Функціональна вимога FR-2

Назва	Авторизація користувача.
Опис	Вводить коректний логін та пароль, який потрібен для системи

Таблиця 2.11 – Функціональна вимога FR-3

Назва	Ввід пошти.
Опис	Ввести правильний формат пошти.

Таблиця 2.12 – Функціональна вимога FR-4

Назва	Ввід паролю.
Опис	Ввести правильний формат паролю.

Таблиця 2.13 – Функціональна вимога FR-5

Назва	Вихід із акаунту.
Опис	Можливість вийти з акаунту.

Таблиця 2.14 – Функціональна вимога FR-6

Назва	Перегляд профілів співробітників
Опис	Можливість переглядати список всіх профілів співробітників, які були додані в систему для табелювання.

Таблиця 2.15 – Функціональна вимога FR-7

Назва	Створити профіль.
Опис	Можливість створити новий профіль у системі..

Таблиця 2.16 – Функціональна вимога FR-8

Назва	Ввести ПІБ.
Опис	Ввести прізвище, ім’я, по батькові співробітника

Таблиця 2.17 – Функціональна вимога FR-9

Назва	Ввести час періоду роботи.
Опис	Ввести час о котрій робітник починає роботу та закінчує.

Таблиця 2.18 – Функціональна вимога FR-10

Назва	Завантажити фото.
Опис	Завантажити фотографії обличчя співробітника для подальшого розпізнавання людини на відео.

Таблиця 2.19 – Функціональна вимога FR-11

Назва	Редагувати профіль.
Опис	Можливість відредагувати старі дані або додати нові дані про обраного співробітника.

Таблиця 2.20 – Функціональна вимога FR-12

Назва	Ввести нові дані.
Опис	Можливість ввести відредаговані дані прізвище, ім’я, по батькові, час початку та кінця роботи .

Таблиця 2.21 – Функціональна вимога FR-13

Назва	Додати нові фото
Опис	Можливість додати нові фотографії обличчя співробітника.

Таблиця 2.22 – Функціональна вимога FR-14

Назва	Видалити фото.
Опис	Можливість видалити непотрібні фотографії співробітника.

Таблиця 2.23 – Функціональна вимога FR-15

Назва	Перегляд звіту.
Опис	Користувач бачить вебсторінку зі звітом. Користувач має можливість переглядати звіт про час відвідування співробітників.

Таблиця 2.24 – Функціональна вимога FR-16

Назва	Завантажити звіт.
Опис	Користувач бачить вебсторінку зі звітом. Користувач має можливість завантажити звіт у форматі Excel файлу (.xlsx).

Таблиця 2.25 – Функціональна вимога FR-17

Назва	Фільтрувати звіт.
Опис	Користувач бачить вебсторінку зі звітом. Користувач має можливість фільтрувати звіт по ПІБ.

Таблиця 2.26 – Функціональна вимога FR-18

Назва	Сортувати звіт.
Опис	Користувач бачить вебсторінку зі звітом. Користувач має можливість сортувати звіт по ПІБ, дата, час.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18
UC-1	+	+	+	+														
UC-2					+													
UC-3						+	+	+	+	+								
UC-4						+					+	+	+					
UC-5						+								+				
UC-6															+			
UC-7															+	+	+	+

Рисунок 2.1 – Матриця трасування вимог

2.4 Розроблення нефункціональних вимог

Нижче наведені вимоги до програмного забезпечення, які не стосуються його функціональності:

– Безпека: Забезпечення високого рівня безпеки є ключовим аспектом розробки програмного забезпечення. Це включає використання різних методів аутентифікації та авторизації, шифрування даних, захист від атак і зловживань, а також гарантування конфіденційності, цілісності та доступності даних.

– Продуктивність: Програмне забезпечення має забезпечувати високу продуктивність і швидкість роботи навіть при великих робочих навантаженнях та обробці великих обсягів даних. Важливо дотримуватися встановленого ліміту максимального часу відповіді, щоб забезпечити безперебійну та плавну роботу системи.

– Надійність: Програмне забезпечення повинно мати високу надійність і стійкість до відмов. Це досягається шляхом проведення тестування для виявлення помилок і вразливостей, резервного копіювання даних, наявності механізмів відновлення та забезпечення безперебійної роботи системи.

– Сумісність: Програмне забезпечення повинно бути сумісним з різними операційними системами, пристроями та браузерами, щоб забезпечити його роботу в різних середовищах. Це означає використання відповідних інтерфейсів, стандартів та протоколів, що дозволяють програмному забезпеченню функціонувати без проблем з різними системами та пристроями.

Висновки до розділу

На основі аналізу представленого матеріалу, можна зробити наступні висновки до розділу "Розроблення вимог до програмного забезпечення":

У цьому розділі було проведено детальний аналіз вимог до програмного забезпечення для автоматизованого табелювання співробітників з використанням системи розпізнавання обличчя. Основні напрямки включають авторизацію користувачів, створення та управління профілями співробітників, а також генерацію звітів про відвідування. Було розроблено діаграми варіантів використання (Use Case), що описують взаємодію користувачів з системою.

Аналіз системних вимог показав, що програмне забезпечення має підтримувати сучасні веб-браузери і працювати на різних типах обладнання, включаючи мінімальні та рекомендовані конфігурації для користувачів і серверної частини. Для серверної частини необхідно встановити Python для підтримки бібліотек штучного інтелекту, використаних для розпізнавання обличчя.

Функціональні вимоги програмного забезпечення були розділені на модулі, кожен з яких має свій набір функцій. Було розроблено матрицю трасування вимог, яка дозволяє відстежувати відповідність функціональних вимог до конкретних модулів програми.

Нефункціональні вимоги включають заходи безпеки для захисту від некоректних дій користувачів та забезпечення цілісності інформації в базі даних.

У підсумку, розроблено докладний план вимог до програмного забезпечення, який буде використаний для подальшої реалізації та тестування системи автоматизованого табелювання співробітників.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Клієнт-серверна архітектура [6] є одним з найпоширеніших підходів у розробці веб-застосунків. Вона розділяє систему на дві основні частини: клієнтську (фронтенд) і серверну (бекенд). Клієнт відповідає за взаємодію з користувачем і відправку запитів до сервера, який обробляє ці запити, взаємодіє з базою даних та повертає результати клієнту.

Переваги:

– Чітке розділення обов'язків: Клієнт і сервер виконують різні завдання, що спрощує розробку і підтримку.

– Масштабованість: Легко масштабувати серверну частину для обробки великої кількості запитів.

– Безпека: Сервер може виконувати складні обчислення та зберігати конфіденційну інформацію без необхідності передавати її клієнту.

Недоліки:

– Залежність від мережі: Потребує стабільного мережевого з'єднання між клієнтом і сервером.

– Можливі затримки: Затримки в мережі можуть впливати на швидкість відгуку.

Монолітна архітектура [7] об'єднує всі функції додатку в єдиному шарі. Всі компоненти тісно пов'язані між собою і працюють як єдине ціле.

Переваги:

– Простота розгортання: Всі компоненти розгортаються як єдине ціле, що спрощує процес розгортання.

– Менші витрати на розробку: Менші початкові витрати на розробку, оскільки всі компоненти інтегровані в один додаток.

Недоліки:

– Складність в масштабуванні: Важко масштабувати окремі частини системи незалежно.

– Менша гнучкість: Будь-яка зміна в одній частині системи може вимагати змін в інших частинах.

– Складність у підтримці: У міру зростання додатка його підтримка стає все складнішою.

Мікросервісна архітектура [8] розділяє додаток на набір дрібних, незалежних сервісів, кожен з яких виконує певну функцію. Ці сервіси можуть взаємодіяти один з одним через добре визначені інтерфейси.

Переваги:

– Масштабованість: Легко масштабувати окремі сервіси незалежно один від одного.

– Гнучкість: Легко впроваджувати нові технології і модифікувати окремі сервіси без впливу на інші.

– Підвищена стійкість: Збій в одному сервісі не обов'язково впливають на інші.

Недоліки:

– Складність управління: Управління множиною сервісів складніше, ніж управління монолітним додатком.

– Витрати на розробку: Потребує значних початкових витрат на розробку та налаштування інфраструктури.

– Залежність від мережі: Як і клієнт-серверна архітектура, потребує стабільного мережевого з'єднання для взаємодії між сервісами.

Для нашого проекту автоматизованого табелювання співробітників клієнт-серверна архітектура є оптимальним вибором, оскільки вона забезпечує чітке розділення обов'язків між клієнтом і сервером, дозволяє ефективно обробляти численні запити в реальному часі та забезпечує гнучкість і масштабованість системи, що є ключовими вимогами для цього типу застосунків. Модель використаної архітектури програмного забезпечення наведена на рисунку 3.1.

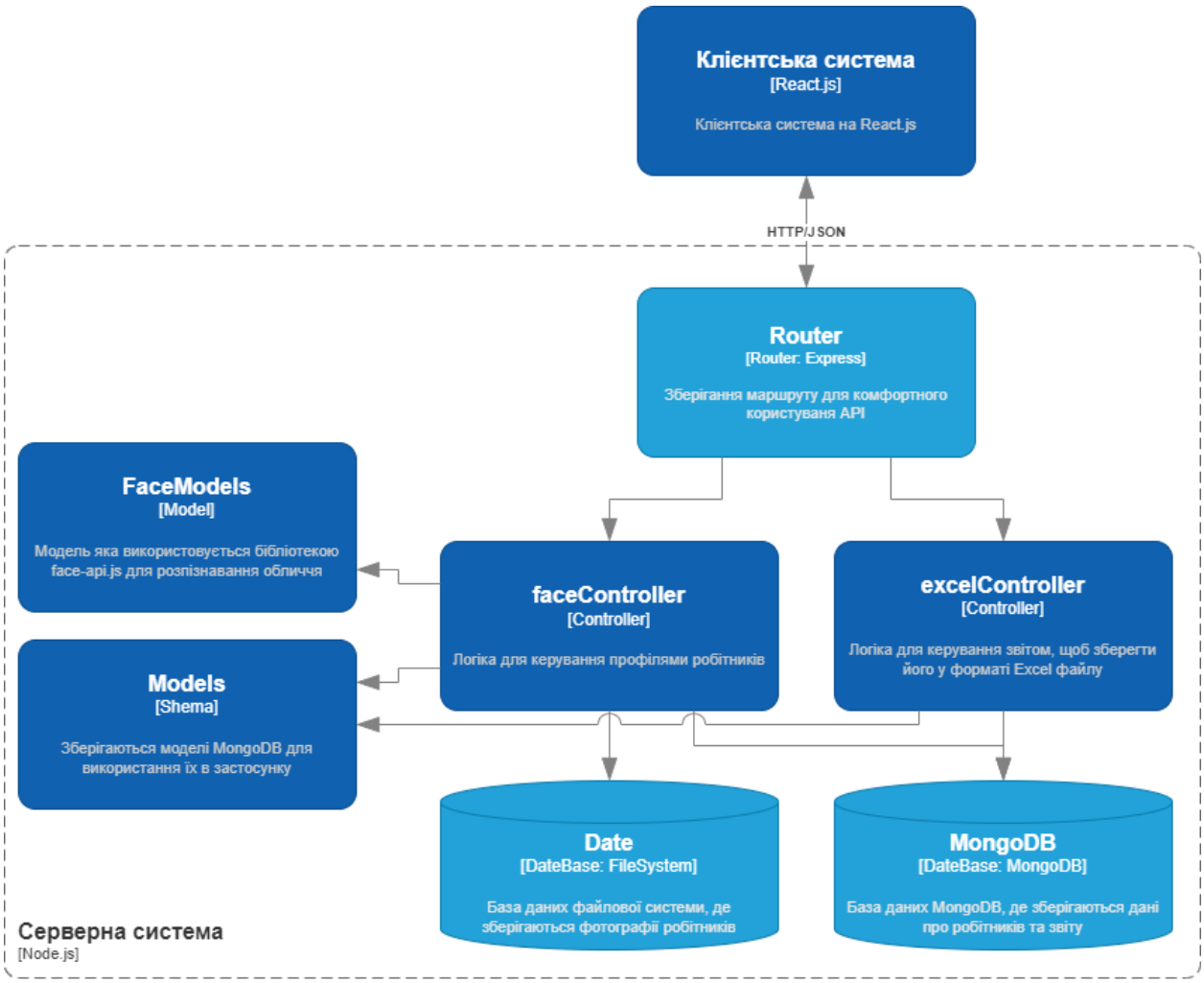


Рисунок 3.1 – Модель архітектури серверної системи

На рисунку можна побачити, що в нас є дві моделі FaceModels та Models. FaceModels нам потрібна для бібліотеки face-aps.js, щоб вона коректно працювала з зображеннями та відео, які потрібно обробити, щоб знайти обличчя та порівняти їх з даними, які вже зберігаються в MongoDB. Models – це написані схеми, якими створюємо колекції, які знаходять в базі даних MongoDB.

Можна побачити компоненту Date, вона описує збереження зображень в файловій системі, де знаходиться сервер. Таке рішення було прийнятим тим, щоб було комфортніше користуватися MongoDB, де ми будемо тільки зберігати посилання на зображення, яке знаходиться в файловій системі.

3.2 Обґрунтування вибору засобів розробки

Для реалізації автоматизованого табелювання співробітників були обрані Node.js [9], React [10] та MongoDB [11]. Ці засоби розробки були вибрані через їх ефективність, продуктивність та сумісність для створення сучасних веб-застосунків.

Node.js був обраний для серверної частини додатку. Це середовище виконання на базі JavaScript, яке дозволяє розробникам використовувати одну мову програмування для фронтенду і бекенду. Node.js відомий своєю високою продуктивністю завдяки неблокуючій, подієвій архітектурі. Він забезпечує швидке виконання коду, що важливо для обробки численних одночасних запитів. Альтернативи, такі як Python (з використанням Flask[12] або Django[13]) або Ruby [14], також популярні, але Node.js має перевагу у швидкості і можливості використовувати JavaScript [15] на обох сторонах розробки.

React був обраний для клієнтської частини додатку. Ця бібліотека для створення інтерфейсів користувача надає можливість створювати динамічні та інтерактивні UI. React забезпечує високу продуктивність завдяки використанню віртуального DOM, що робить його ідеальним для сучасних веб-застосунків. Альтернативи, такі як Angular [16] або Vue.js [17], також мають свої переваги. Angular, наприклад, надає більше функціональності «з коробки», але має більш круту криву навчання. Vue.js є більш простим у вивченні, але менш поширеним у великих корпоративних проектах порівняно з React. React був обраний через його популярність, широке співтовариство і багатий набір інструментів та бібліотек.

MongoDB був обраний як база даних. Це документо-орієнтована база даних, яка забезпечує гнучкість у зберіганні та обробці даних. Вона добре підходить для роботи з великими обсягами даних та динамічними схемами. MongoDB дозволяє легко масштабувати систему та забезпечує високу продуктивність. Альтернативи, такі як PostgreSQL [18] або MySQL [19], також популярні, особливо для реляційних даних. Проте, для цього проекту

MongoDB був обраний через його гнучкість і можливість ефективно обробляти неструктуровані дані.

Для розробки була обрана IDE Visual Studio Code [20]. Це безкоштовний, легкий редактор коду з великою кількістю розширень, які підтримують розробку на JavaScript, Node.js, і React. Він забезпечує зручний інтерфейс і інтеграцію з системами контролю версій, такими як Git [21]. Альтернативи, такі як WebStorm [22] або Atom [23], також мають свої переваги. WebStorm пропонує більш інтегровані інструменти, але є комерційним продуктом. Atom, створений GitHub [24], є схожим на Visual Studio Code, але менш популярним. Visual Studio Code був обраний за його високу продуктивність, гнучкість та підтримку спільноти.

Таким чином, вибір Node.js, React, MongoDB та Visual Studio Code базується на їх здатності забезпечити високу продуктивність, гнучкість та зручність розробки, що є ключовими вимогами для нашого проекту автоматизованого табелювання співробітників.

Для виконання специфічних завдань у проекті також будуть використовуватися бібліотеки face-api.js [25], @tensorflow/tfjs-node [26] та exceljs [27].

face-api.js є бібліотекою для розпізнавання облич на основі глибокого навчання. Вона забезпечує можливості детекції облич, розпізнавання їхніх рис, ідентифікації облич та оцінки емоцій. Ця бібліотека базується на TensorFlow.js і забезпечує високу точність і продуктивність при обробці зображень у реальному часі. Використання face-api.js дозволяє інтегрувати функції біометричної аутентифікації у веб-застосунок, що підвищує безпеку і зручність користувачів.

@tensorflow/tfjs-node є серверною версією TensorFlow.js, яка дозволяє використовувати можливості машинного навчання на стороні сервера. Це забезпечує високоефективну обробку даних і виконання моделей машинного навчання безпосередньо у Node.js середовищі. @tensorflow/tfjs-node підходить для задач, що потребують значних обчислювальних ресурсів і високої

продуктивності, таких як розпізнавання облич, класифікація зображень та інші.

exceljs є бібліотекою для роботи з файлами Excel [28] у Node.js. Вона дозволяє читати, створювати і змінювати файли Excel, що є корисним для генерації звітів і обробки табличних даних. Використання exceljs дозволяє автоматизувати процес створення табелів обліку робочого часу, що значно спрощує адміністрування і знижує ймовірність помилок.

Вибір цих бібліотек базується на їх здатності забезпечити необхідну функціональність для проекту автоматизованого табелювання співробітників, підвищити продуктивність і забезпечити високу якість реалізації специфічних завдань.

3.3 Конструювання програмного забезпечення

У графічних матеріалів, креслення 3, наведена діаграма класів серверної частини ПЗ.

Для серверної частини застосунку було обрано паттерн MVC (Model-View-Controller). Використання цього паттерну забезпечує чітке розділення логіки програми на три основні компоненти: модель, представлення та контролер. Це сприяє більшій модульності коду, спрощує його супровід та тестування, а також полегшує розвиток застосунку, схему цього шаблону можна побачити на рисунку 3.2.

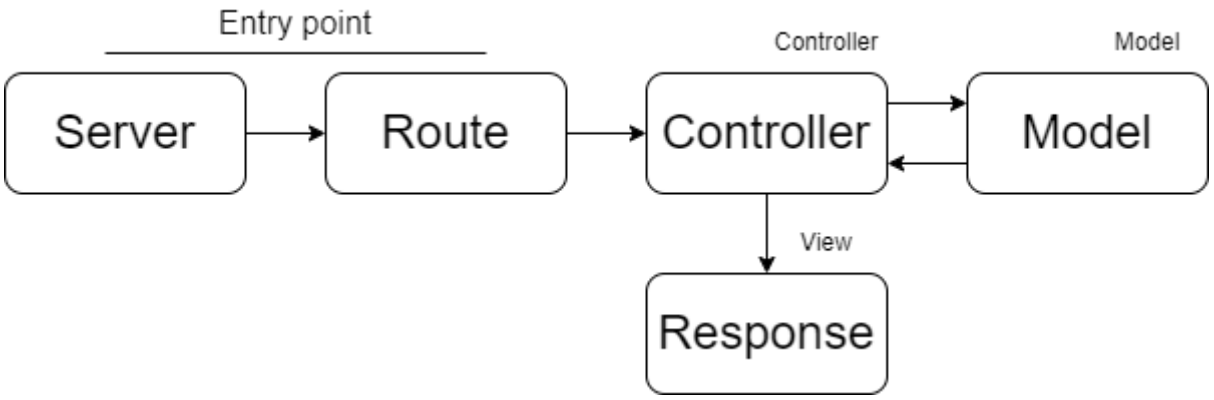


Рисунок 3.2 – Схема MVC паттерну

Модель (Model) представляє собою рівень даних застосунку. Вона відповідає за взаємодію з базою даних, зберігання та обробку даних. У даному

проекті модель включає визначення схем бази даних MongoDB та методів для роботи з ними. Це дозволяє зручно маніпулювати даними користувачів, їх профілями та іншою інформацією, необхідною для роботи системи.

Представлення (View) є рівнем, який відповідає за відображення даних користувачу. У випадку серверної частини це, як правило, формування відповіді у форматі, який буде зрозумілим клієнтській частині застосунку, наприклад, у форматі JSON. Представлення не містить бізнес-логіки, а лише відповідає за презентацію даних, отриманих від моделі.

Контролер (Controller) виступає посередником між моделлю та представленням. Він обробляє вхідні запити від користувачів, взаємодіє з моделлю для отримання або зміни даних і передає ці дані у відповідне представлення. Контролер містить бізнес-логіку застосунку, забезпечуючи правильну обробку запитів та координацію дій між моделлю та представленням.

Загалом, застосування паттерну MVC у розробці серверної частини забезпечує організовану структуру коду, підвищує його якість та спрощує підтримку, що є ключовим для успішної реалізації та подальшого розвитку системи автоматизованого табелювання співробітників.

В якості системи управління базами даних використовується MongoDB. База даних призначена для зберігання даних про робітників, а також дані про їх відвідування в звіті, а точніше, коли були помічені на відео. Опис колекцій бази даних наведено у таблицях 3.1 - 3.3. Модель бази даних наведена у графічних матеріалів, креслення 2.

Таблиця 3.1 – Опис колекції users

Назва поля	Тип даних	Опис
_id	objectId	ідентифікаційний номер документу user
firstName	string	ім’я користувача
lastName	string	прізвище користувача

Продовження таблиці 3.1

Назва поля	Тип даних	Опис
middleName	string	по батькові користувача
images	[image:ObjectId]	зберігає масив ідентифікаційних номерів документів колекції image
entryTime	string	час початку роботи робітника
outTime	string	час кінеця роботи робітника

Таблиця 3.2 – Опис колекції images

Назва поля	Тип даних	Опис
_id	objectId	ідентифікаційний номер документу image
url	string	путь збережених фотографій в локальній файловій системі
descriptions	array	Масив опису фотографій людини, які використовуються для порівняння обличчя

Таблиця 3.3 – Опис колекції detectedusers

Назва поля	Тип даних	Опис
_id	objectId	ідентифікаційний номер документу detectedusers
userID	[user:ObjectId]	ідентифікаційних номерів документів колекції user
date	string	день коли було помічено робітника
time	string	час коли було помічено робітника
status	string	статус, запізнився робітник чи ні

Продовження таблиці 3.3

Назва поля	Тип даних	Опис
createAt	data	час коли було створено документ

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.4.

Таблиця 3.4 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Visual Studio Code	Головне середовище розробки програмного забезпечення серверної та клієнтської частини дипломної роботи.
2	Postman	Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів.
3	MongoDB Online	Хмарне програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних.
4	express	Легка та гнучка веб-структура для Node.js, яка використовується для побудови серверних додатків та API.
5	express-fileupload	Мідлвер для обробки завантаження файлів у додатках Express, що дозволяє легко інтегрувати функціонал завантаження файлів.

Продовження таблиці 3.4

№ п/п	Назва утиліти	Опис застосування
6	mongoose	ORM (Object-Relational Mapping) бібліотека для MongoDB і Node.js, яка забезпечує легку роботу з базами даних за допомогою об'єктно-орієнтованого підходу.
7	nodemon	Інструмент, що автоматично перезапускає Node.js додаток при внесенні змін до файлів проекту, що значно покращує процес розробки.
8	dotenv	Бібліотека для завантаження змінних середовища з .env файлу, що дозволяє зберігати конфіденційну інформацію окремо від коду.
9	canvas	Бібліотека для роботи з HTML5 Canvas у Node.js, яка дозволяє створювати та обробляти графічні зображення.
10	uuid	Бібліотека для генерації унікальних ідентифікаторів (UUID), яка забезпечує створення унікальних значень для ідентифікації ресурсів.
11	exceljs	Бібліотека для роботи з Excel-файлами у Node.js, що дозволяє створювати, редагувати та читати документи Excel.
12	face-api.js	Бібліотека для розпізнавання облич на JavaScript, яка використовує глибоке навчання для виявлення та аналізу облич.
13	@tensorflow/tfjs-node	Бібліотека TensorFlow.js для Node.js, яка дозволяє використовувати моделі машинного навчання безпосередньо у серверних додатках.

Кінець таблиці 3.4

№ п/п	Назва утиліти	Опис застосування
14	jsonwebtoken	Бібліотека для створення і валідації JSON Web Tokens (JWT), яка забезпечує аутентифікацію та авторизацію користувачів.
15	bcrypt	Бібліотека для хешування паролів, яка забезпечує безпечне зберігання та перевірку паролів користувачів.
16	multer	Мідлвер для обробки завантаження файлів у додатках Express, що дозволяє працювати з файлами, які завантажуються через форми.
17	ws	Бібліотека для роботи з WebSockets у Node.js, що дозволяє реалізувати двосторонній зв'язок між клієнтом та сервером у реальному часі.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Аналіз безпеки даних є критично важливим для веб-застосунку автоматизованого табелювання співробітників. У цьому підрозділі розглянемо основні заходи, щоб забезпечити захист даних.

Захист паролів: Паролі користувачів повинні зберігатися у зашифрованому вигляді з використанням надійних алгоритмів хешування, таких як bcrypt. Це забезпечує захист паролів навіть у випадку витоку бази даних.

Валідація вхідних даних: Всі вхідні дані повинні бути ретельно перевірені перед використанням. Використання бібліотек для валідації даних допомагає запобігти атакам типу SQL Injection.

Безпечне управління сесіями: Використання захищених сесійних токенів, зберігання їх у HTTP-only куках і налаштування сесійного тайм-ауту допомагає захистити сесії користувачів від захоплення.

Висновки до розділу

У розділі "Конструювання та розроблення програмного забезпечення" було проведено детальний аналіз архітектури системи, обгрунтовано вибір засобів розробки та представлено структуру класів серверної частини програми. Також наведено опис бази даних та використовуваних утиліт, бібліотек і іншого програмного забезпечення.

Клієнт-серверна архітектура була обрана як оптимальний вибір для проекту автоматизованого табелювання співробітників, оскільки вона забезпечує ефективну взаємодію між клієнтом і сервером, розділення обов'язків і гнучкість системи.

Вибір засобів розробки, таких як Node.js для серверної частини, React для клієнтської частини та MongoDB для бази даних, обгрунтовано їхньою ефективністю, продуктивністю та сумісністю для створення веб-застосунків.

Діаграма класів серверної частини дозволяє чітко представити структуру програмного забезпечення. База даних MongoDB була обрана для зберігання даних про співробітників та їх відвідування.

Представлені утиліти, бібліотеки та інше програмне забезпечення, такі як Visual Studio Code для розробки, Postman для тестування API та MongoDB Online для доступу до бази даних, допомагають у зручній та продуктивній розробці програми.

У розділі також проведено аналіз безпеки даних, де визначено основні заходи для захисту даних користувачів, такі як захист паролів, валідація вхідних даних та безпечне управління сесіями.

Отже, результати даного розділу дозволяють чітко зрозуміти архітектуру, інструменти та заходи безпеки, які використовуються у процесі розроблення програмного забезпечення для автоматизованого табелювання співробітників.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення включає оцінку його характеристик за різними метриками. Це дозволяє виявити сильні та слабкі сторони системи, забезпечити відповідність функціональним та нефункціональним вимогам, а також підвищити загальну якість ПЗ.

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, …);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

Метриками для оцінки якості ПЗ обрано наступні:

– швидкість: вимірюється час відгуку системи на запити користувачів та швидкість обробки даних. Висока продуктивність є критично важливою для забезпечення безперебійної роботи та зручності використання.

– надійність: оцінюється здатність системи працювати без збоїв протягом певного часу. Надійність включає перевірку статусу відповідей від серверу.

Для статичного аналізу коду було обрано сервіс SonarCloud[30]. Для того, щоб почати використовувати його перевірки, потрібно було підключити сервіс до репозиторію GitHub з кодом. Після, було доступно інтерфейс з інформацією про статус на правильність структури коду. Результати можна побачити на рисунку 4.1.

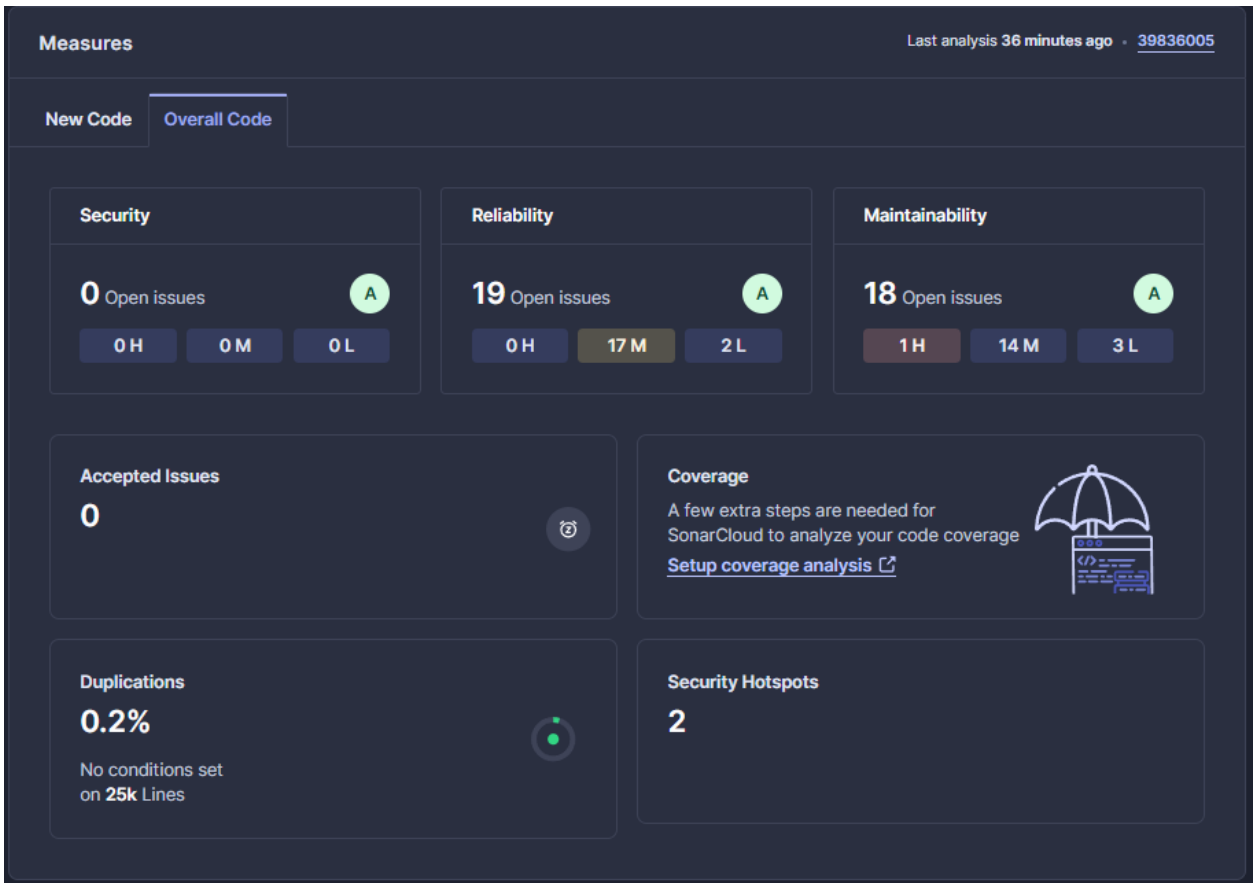


Рисунок 4.1 – Результат перевірки SonarCloud

Для динамічного аналізу коду було обрано бібліотеку express-status-monitor[31]. Це простий самостійний модуль для звітів про метрики сервера в реальному часі для серверів вузлів на базі Express. Результати можна побачити на рисунку 4.2.

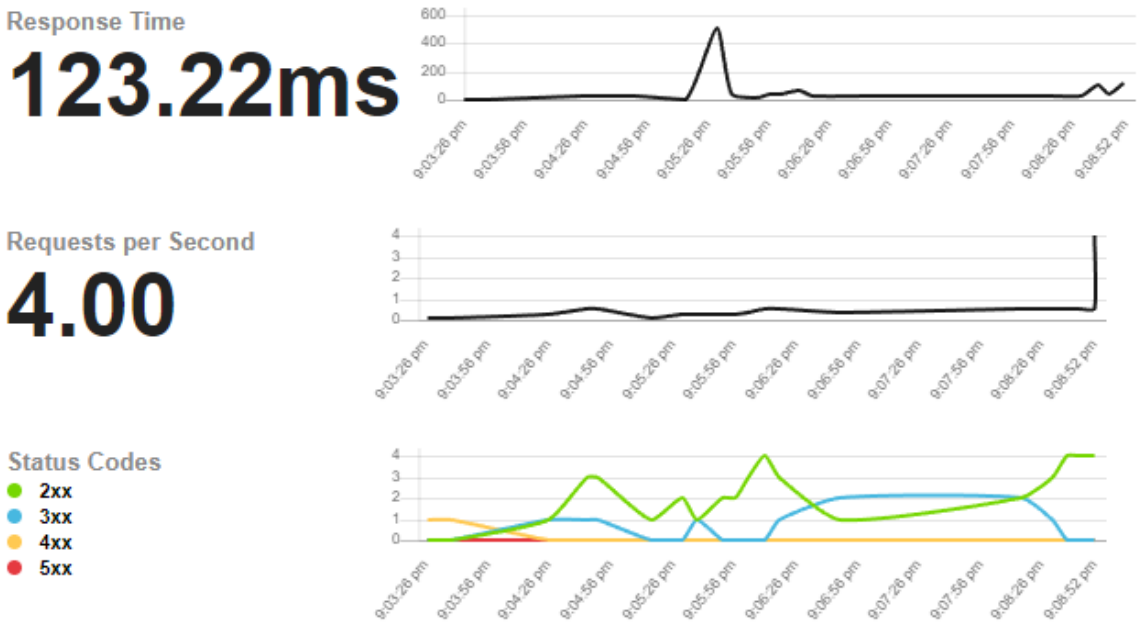


Рисунок 4.2 – Результати звіту про метрики сервера

Отже, під час аналізу якості програмного забезпечення було виявлено позитивні результати за всіма метриками, визначеними для оцінки. Програмне забезпечення успішно пройшло тестування і виявилося надійним, безпечним, швидким та зручним у використанні.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1 – 4.12.

Таблиця 4.1 – Тест 1.1 Авторизація користувача

Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка до цього була зареєстрована в системі, пароль від 1 до 64 символів, який містить хоча б з одну англійську літеру, одне число і один спеціальний символ. Після цього натискається кнопка підтвердження авторизації.
Очікуваний результат	Авторизація проходить успішно, користувач перенаправляється на сторінку створення профілю робітника.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 1.2 Не коректна авторизація користувача

Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: не коректна електронна пошта, яка до цього не була зареєстрована в системі. Після цього натискається кнопка підтвердження авторизації.
Очікуваний результат	Авторизація проходить не успішно, користувачу на екран виводиться повідомлення про неправильно введену ел. пошту чи пароль.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 1.3 Створення профілю робітника

Початковий стан системи	Користувач знаходиться на сторінці створення профілю
Вхідні дані	Прізвище, ім’я, по батькові, час початку та кінця роботи, фотографії.
Опис проведення тесту	У відповідні поля вводяться: прізвище, ім’я, по батькові, тільки текстом, час початку та кінця роботи, тільки цифрами, обираються мінімум 1 фотографія. Після цього натискається кнопка створення.
Очікуваний результат	Користувачу на екран виводиться повідомлення про успішне збережені дані та в список робітників, додається посилання на профіль робітника.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.4 Переглянути профіль робітника

Початковий стан системи	Користувач знаходиться на сторінці створення профілю
Вхідні дані	-
Опис проведення тесту	У списку робітників натискаємо на посилання профілю працівника.
Очікуваний результат	Користувач перенаправляється на сторінку профілю робітника.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 1.5 Редагувати профіль робітника

Початковий стан системи	Користувач знаходиться на сторінці профілю робітника
Вхідні дані	Прізвище, ім’я, по батькові, час початку та кінця роботи
Опис проведення тесту	У відповідні поля вводяться: прізвище, ім’я, по батькові, тільки текстом, час початку та кінця роботи, тільки цифрами. Після цього натискається кнопка зберегти зміни.
Очікуваний результат	Користувачу на екран виводиться повідомлення про успішне оновлені дані.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 1.6 Видалити профіль робітника

Початковий стан системи	Користувач знаходиться на сторінці профілю робітника
Вхідні дані	-
Опис проведення тесту	Натискаємо на кнопку видалити профіль. Після з’являється кнопка з підтвердженням, натискаємо підтвердити.
Очікуваний результат	Профіль робітника видаляється та користувач перенаправляється на сторінку створити профіль.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 1.7 Переглядати звіт

Початковий стан системи	Користувач знаходиться на сторінці звіту
Вхідні дані	-
Опис проведення тесту	Перейти на сторінку звіту.
Очікуваний результат	З’являється табличка зі звітністю відвідування робітників.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 1.8 Фільтрувати звіт за ПІБ

Початковий стан системи	Користувач знаходиться на сторінці звіту
Вхідні дані	Прізвище, ім’я, по батькові.
Опис проведення тесту	В поле пошук за ПІБ вводимо такі дані: прізвище, ім’я, по батькові тільки текстом.
Очікуваний результат	Оновлюється табличка звіту відносно ПІБ.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 1.9 Фільтрувати звіт за датою

Початковий стан системи	Користувач знаходиться на сторінці звіту
Вхідні дані	Дата.
Опис проведення тесту	В поле пошук за датою, обираємо день або вводимо вручну.
Очікуваний результат	Оновлюється табличка звіту відносно дати, яка була вказана в фільтрі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 1.10 Сортувати звіт

Початковий стан системи	Користувач знаходиться на сторінці звіту
Вхідні дані	-
Опис проведення тесту	Після натиснення на заголовок в таблиці: ПІБ, початок роботи, кінець роботи, дата, час. Буде оновлюватися таблиця.
Очікуваний результат	Оновлюється табличка звіту відносно заголовку, якого була вибрана сортування.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.11 – Тест 1.11 Редагувати звіт

Початковий стан системи	Користувач знаходиться на сторінці звіту
Вхідні дані	-
Опис проведення тесту	Після натиснення на кнопку видалити у рядку з даними, хто був помічений у табличку звіт, видаляється рядок.
Очікуваний результат	Оновлюється табличка звіту з видаленим рядком.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.12 – Тест 1.12 Завантажити звіт

Початковий стан системи	Користувач знаходиться на сторінці звіту
Вхідні дані	-

Продовження таблиці 4.12

Опис проведення тесту	Після натиснення на кнопку завантажити, завантажується файл Excel формату зі звітом. Якщо був використаний фільтр, тоді файл завантажується з відфільтрованою табличкою.
Очікуваний результат	На комп’ютер користувача завантажується файл Excel.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

Для перевірки правильності роботи програмного забезпечення автоматизованого табелювання співробітників, було обрано контрольний приклад, який включає всі можливі розгалуження та особливості системи. Нижче наведено повний опис цього прикладу, доповнений рисунками 4.3 – 4.5 для кращого розуміння.

Неавторизований користувач знаходиться на сторінці авторизації рисунок 4.3.

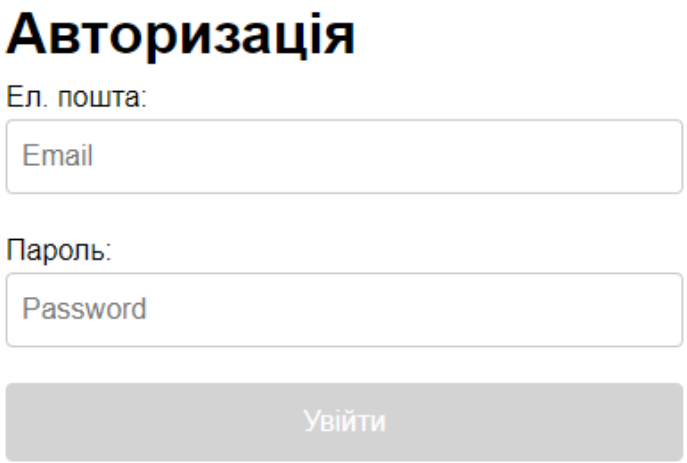


Рисунок 4.3 – Сторінка авторизації

Неавторизований користувач має можливість авторизуватися, якщо його акаунт є у системі рисунок 4.4.

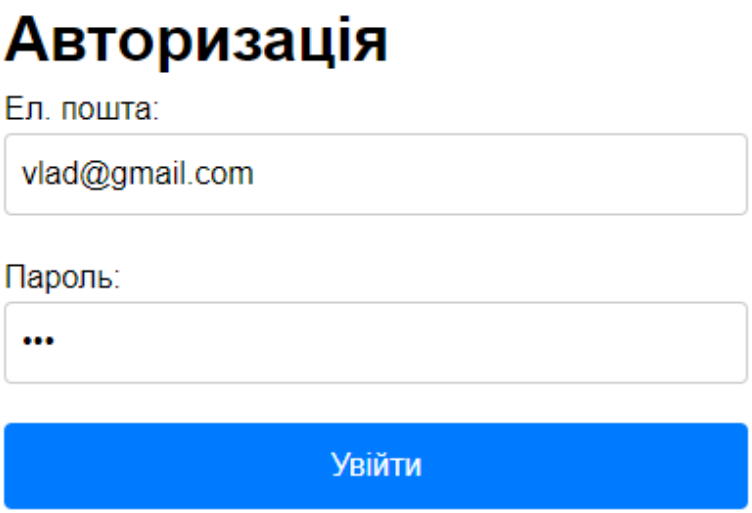


Рисунок 4.4 – Сторінка авторизації з даними

Після входу у систему, користувача направляє на сторінку створення профілю робітника рисунок 4.5.

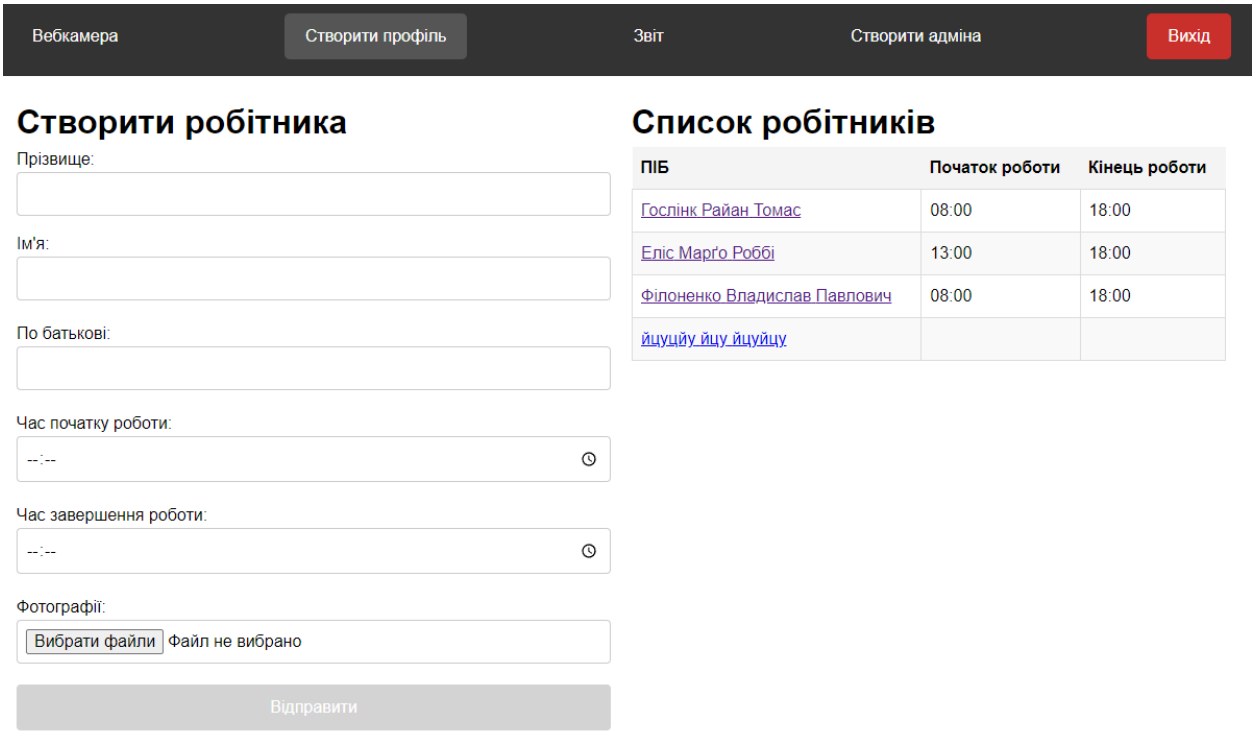


Рисунок 4.5 – Сторінка створення профілю робітника

Після заповнення форми створення робітника можна надіслати дані, потім профіль працівника створюється та додається до списку робітників, рисунок 4.6 – 4.7.

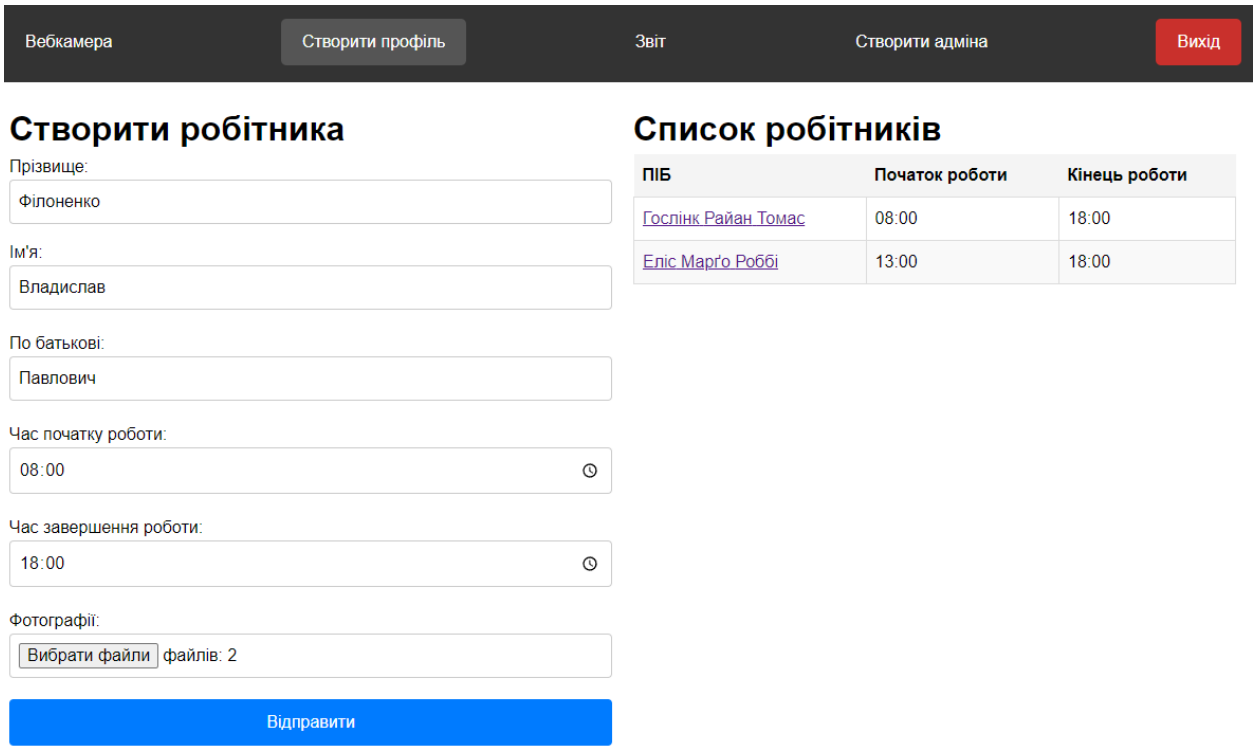


Рисунок 4.6 – Сторінка створення профілю робітника з заповненою формою

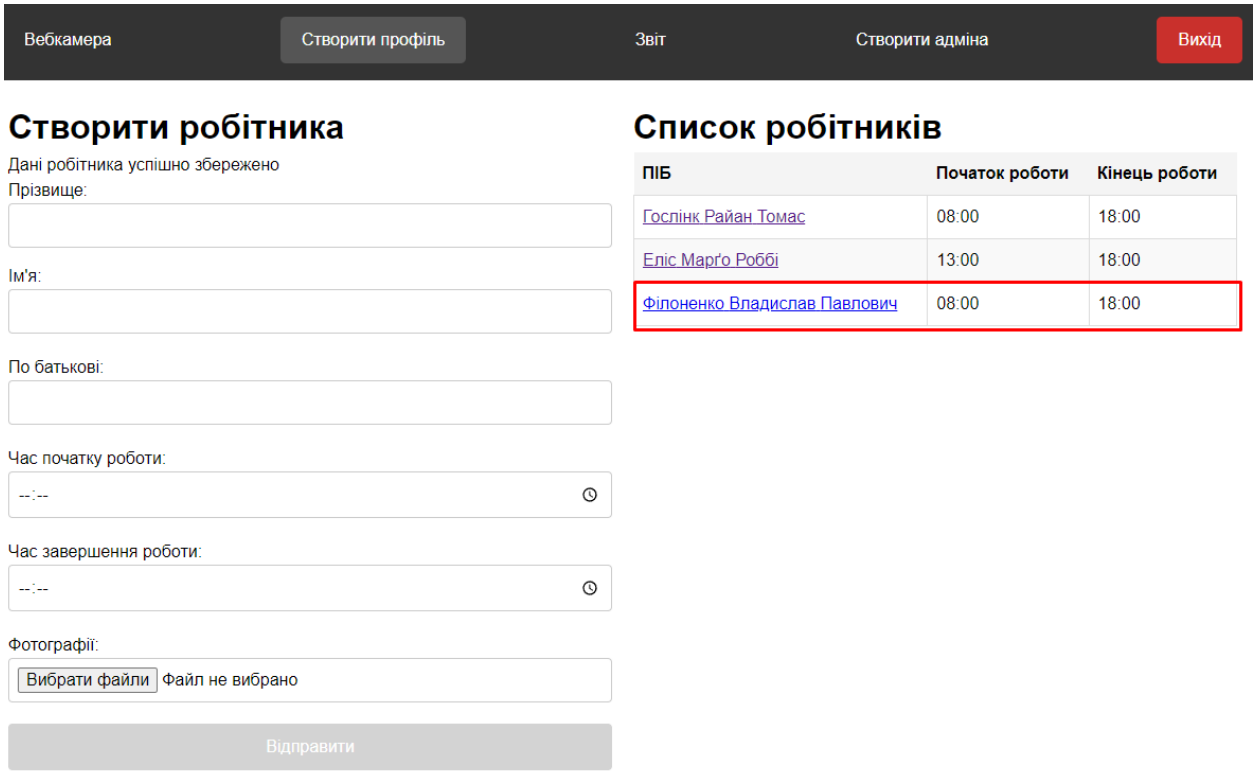


Рисунок 4.7 – Профіль додано до списку

Після натиснення на посилання у списку, користувача перенаправляє на сторінку профілю обраного робітника рисунок 4.8.

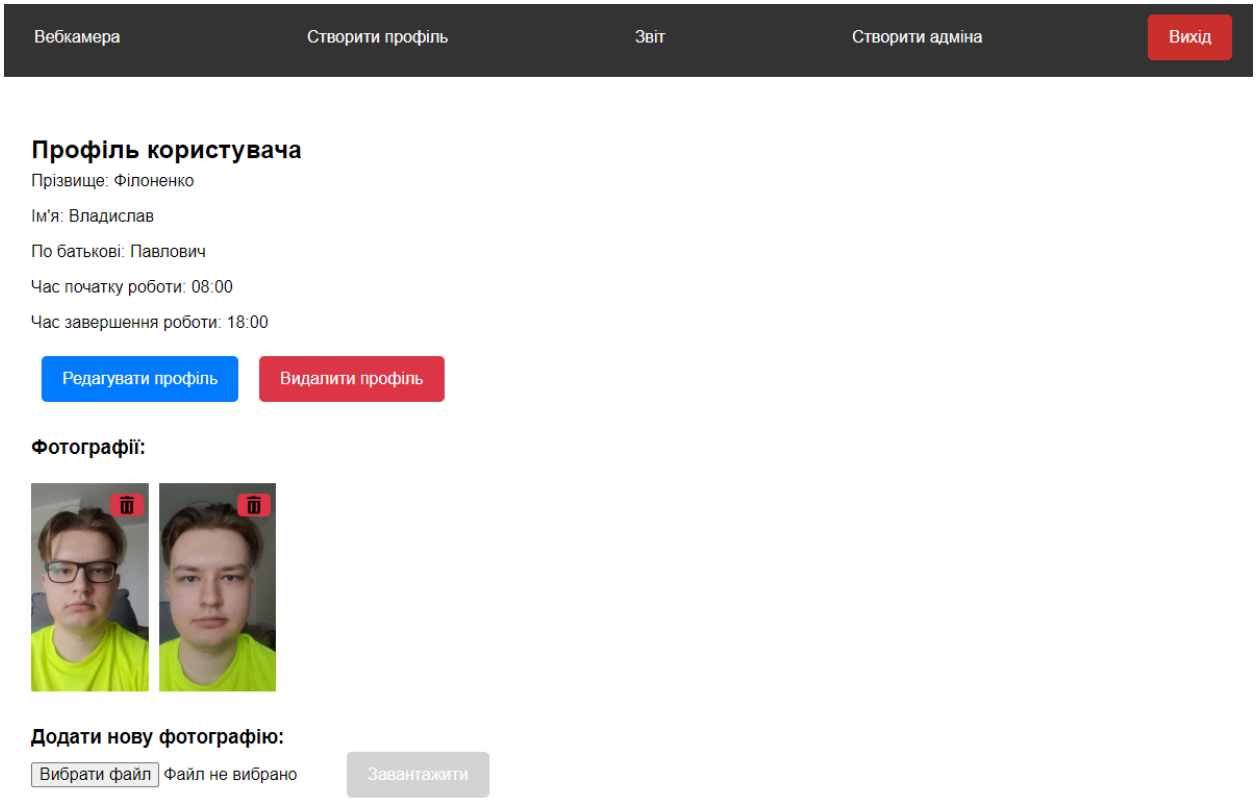


Рисунок 4.8 – Сторінка профілю робітника

Перебуваючи на сторінці профілю робітника, користувач може редагувати дані, додати нові фото обличчя та видалити профіль. Весь описаний функціонал можна побачити на рисунках 4.9 – 4.11.

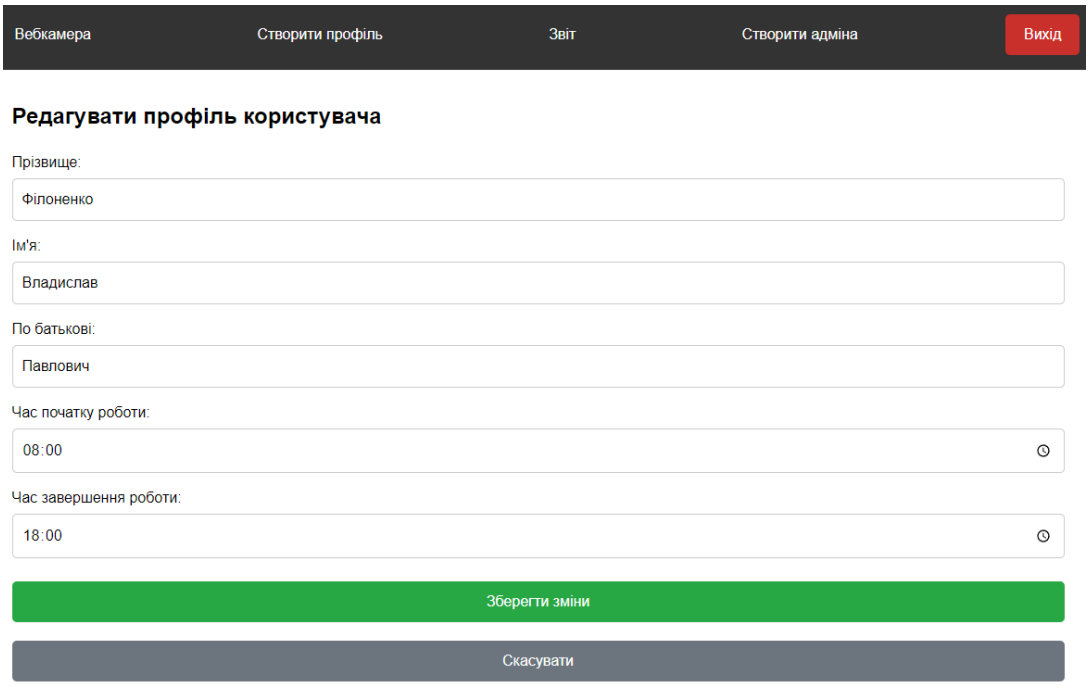


Рисунок 4.9 – Форма редагування профілю робітника

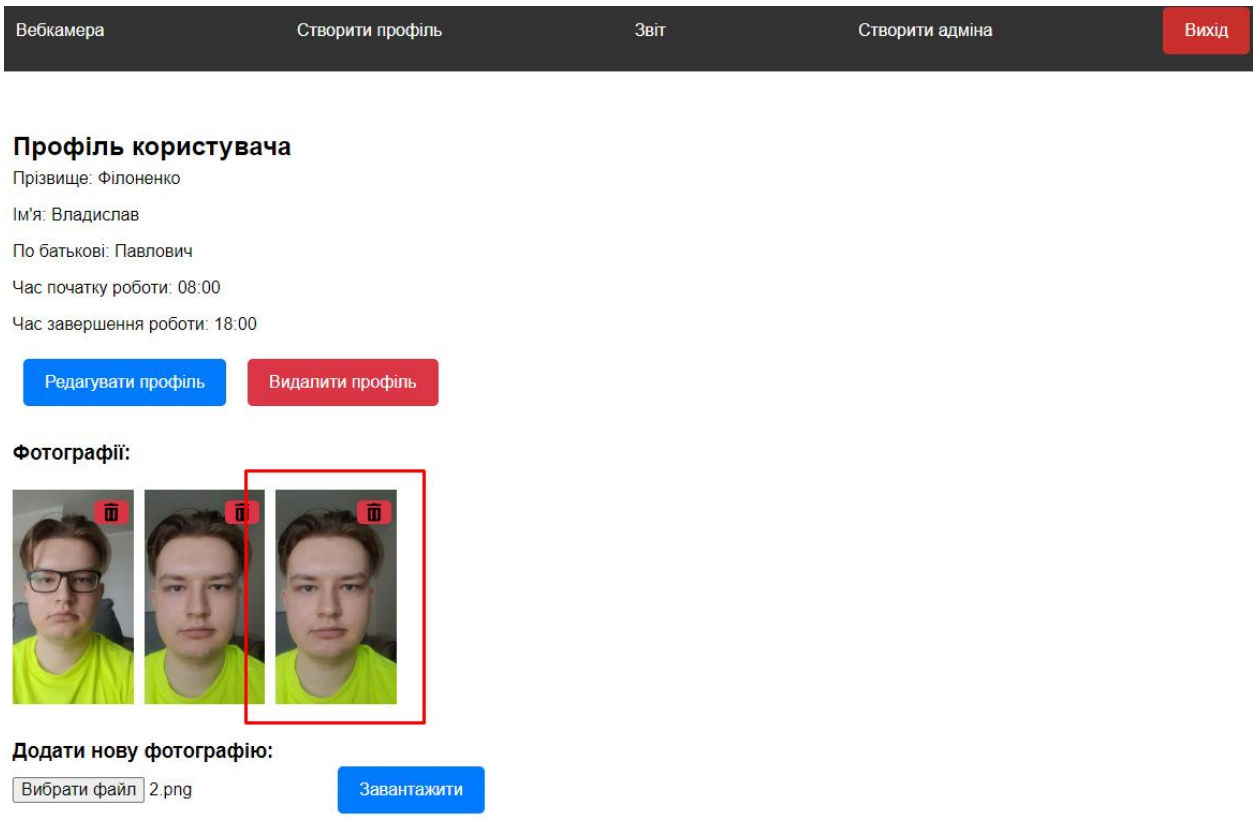


Рисунок 4.10 – Додане нове фото до профілю робітника

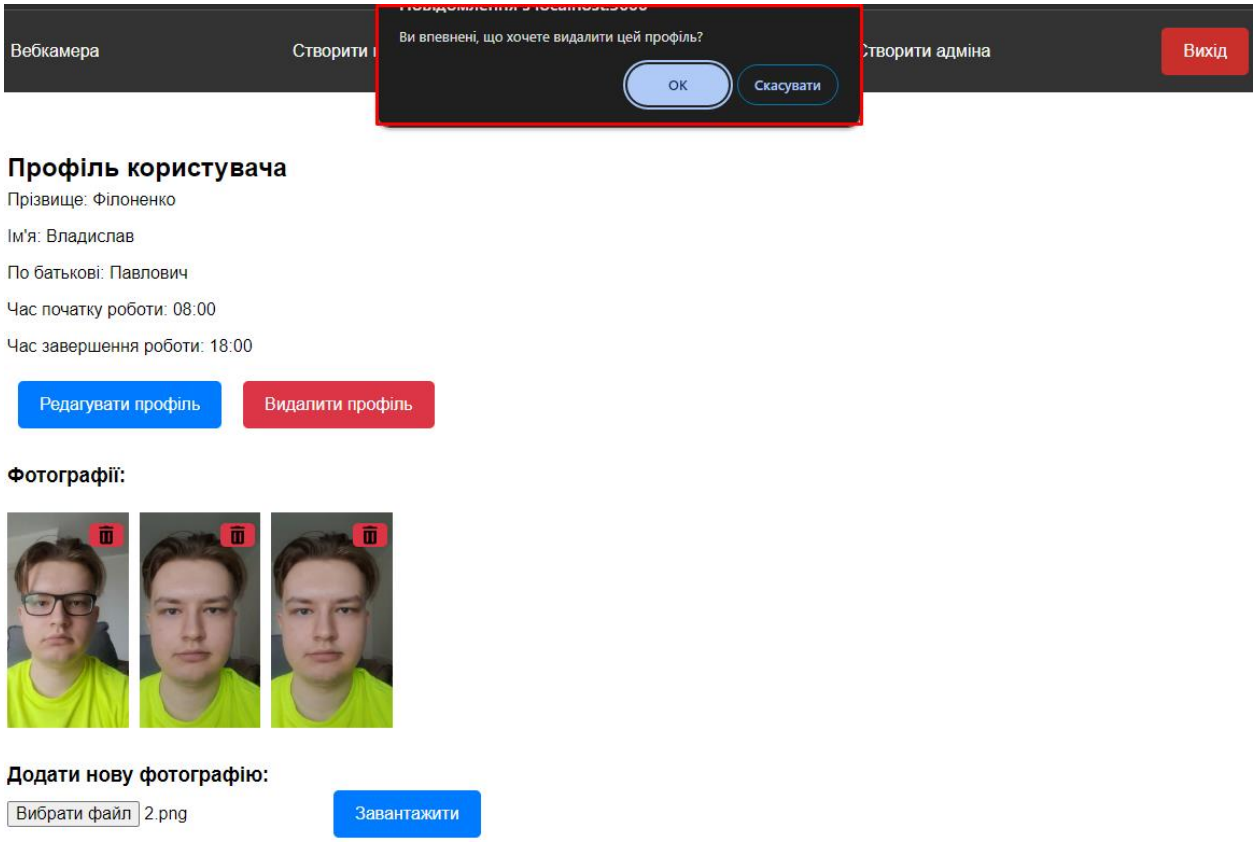


Рисунок 4.11 – Видалення профілю робітника

Користувач на сторінці звіту може побачити табличку зі всіма робітниками, яких було помічено на відеокамері, рисунок 4.12. Після

натиснення на кнопку завантажити Excel, на комп'ютер користувача завантажується файл, рисунок 4.13.

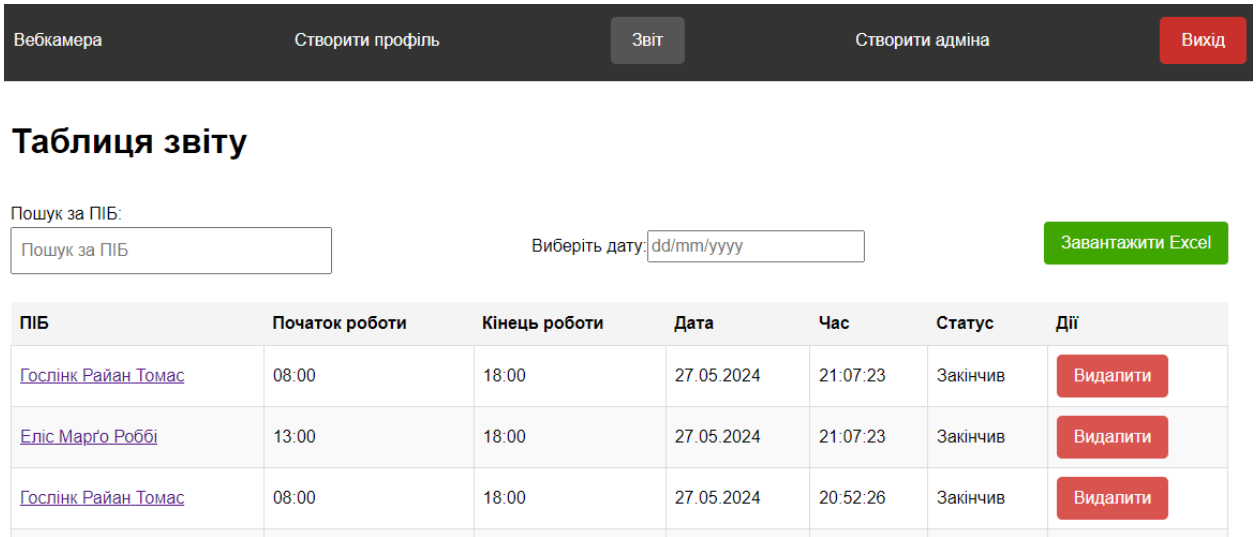


Рисунок 4.12 – Сторінка звіту

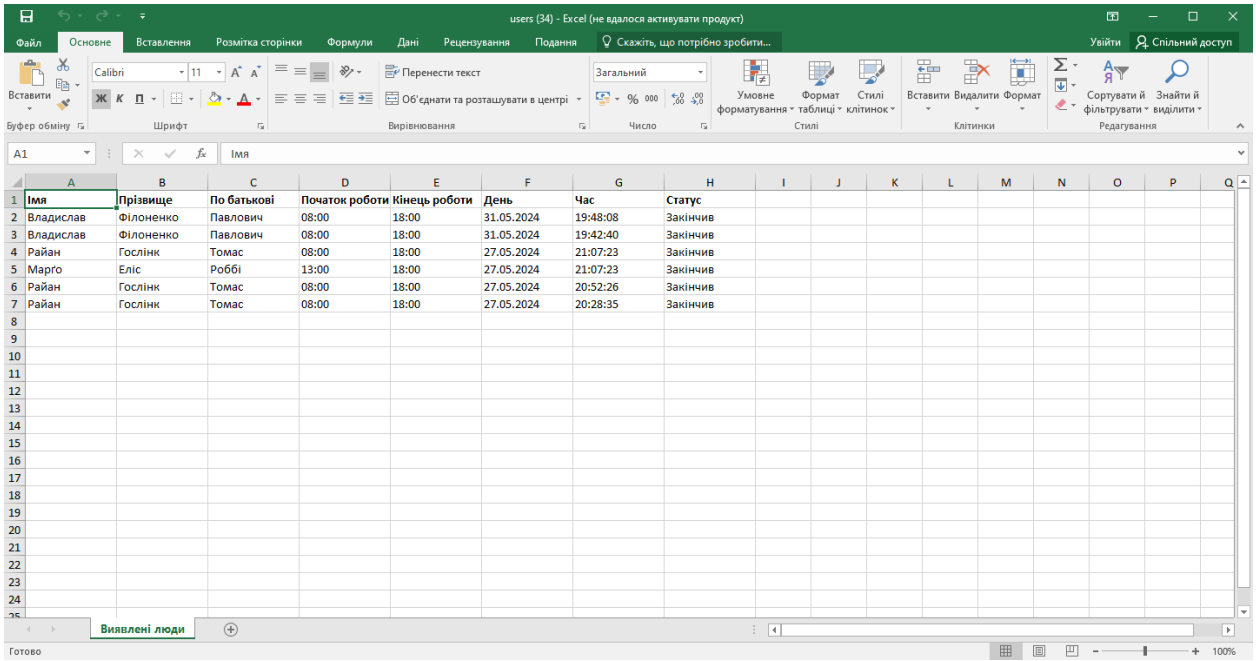


Рисунок 4.13 – Звантажений звіт у форматі Excel

Висновки до розділу

Було детально розглянуто процес аналізу якості та тестування програмного забезпечення для автоматизованого табелювання співробітників. Основну увагу приділено перевірці відповідності ПЗ функціональним та

нефункціональним вимогам, зокрема перевірено правильність роботи, збереження даних, сумісність із сучасними браузерами, зручність графічного інтерфейсу та виявлення помилок для їх усунення.

В рамках аналізу якості було обрано наступні метрики: швидкість та надійність. Швидкість оцінювалася за часом відгуку системи та швидкістю обробки даних, що є критично важливим для забезпечення безперебійної роботи та зручності використання. Надійність оцінювалася здатністю системи працювати без збоїв протягом певного часу, включаючи перевірку статусу відповідей від сервера.

Для статичного аналізу коду було використано сервіс SonarCloud, який забезпечив перевірку правильності структури коду та виявив можливі помилки. Результати аналізу показали, що структура коду відповідає встановленим стандартам. Для динамічного аналізу коду було обрано бібліотеку express-status-monitor, яка забезпечила звіти про метрики сервера в реальному часі, підтвердивши стабільність і продуктивність роботи сервера.

Завершенням розділу є опис контрольного прикладу, що продемонстрував усі ключові функціональні можливості системи автоматизованого табелювання співробітників. Описаний приклад включав усі можливі розгалуження та особливості системи, що забезпечило повне тестування основних функціональних можливостей.

Таким чином, аналіз якості та тестування програмного забезпечення підтвердили його відповідність встановленим вимогам, забезпечивши високу якість, надійність та продуктивність системи для автоматизованого табелювання співробітників.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

У цьому підрозділі розглядається процес розгортання програмного забезпечення, включаючи аналіз засобів розгортання та покрокову інструкцію. Це дозволяє забезпечити безперебійне встановлення та конфігурацію системи на робочих середовищах.

Для розгортання програмного забезпечення обрано Docker[32] та Docker Hub. Docker дозволяє створювати контейнери, які містять всі необхідні компоненти ПЗ, а Docker Hub надає зручний спосіб зберігання та розповсюдження Docker-образів. Використання Docker Hub спрощує процес розгортання, дозволяючи легко отримувати оновлення образів та ділитися ними з іншими розробниками.

Розглянемо покрокове розгортання:

– Встановіть Docker на сервер

– Переконайтеся, що ваш сервер відповідає вимогам для запуску Docker-контейнерів.

– У клієнтській та серверній директорії проекту створюємо файл Dockerfile з необхідною конфігурацією для додатку.

– Виконаємо команду “docker build –t name .” для створення Docker-образу в клієнтській та серверній директорії, тоді в нас повинно вийти два образи рисунок 5.1.

– Для того щоб надіслати наші образи до Docker Hub, потрібно для початку увійти в свій акаунт, це можна зробити завдяки Docker Desktop. Потім потрібно виконати команди “docker tag name fl1nk/name” та “docker push fl1nk/name”, як для директорії клієнтської, так і для серверної.

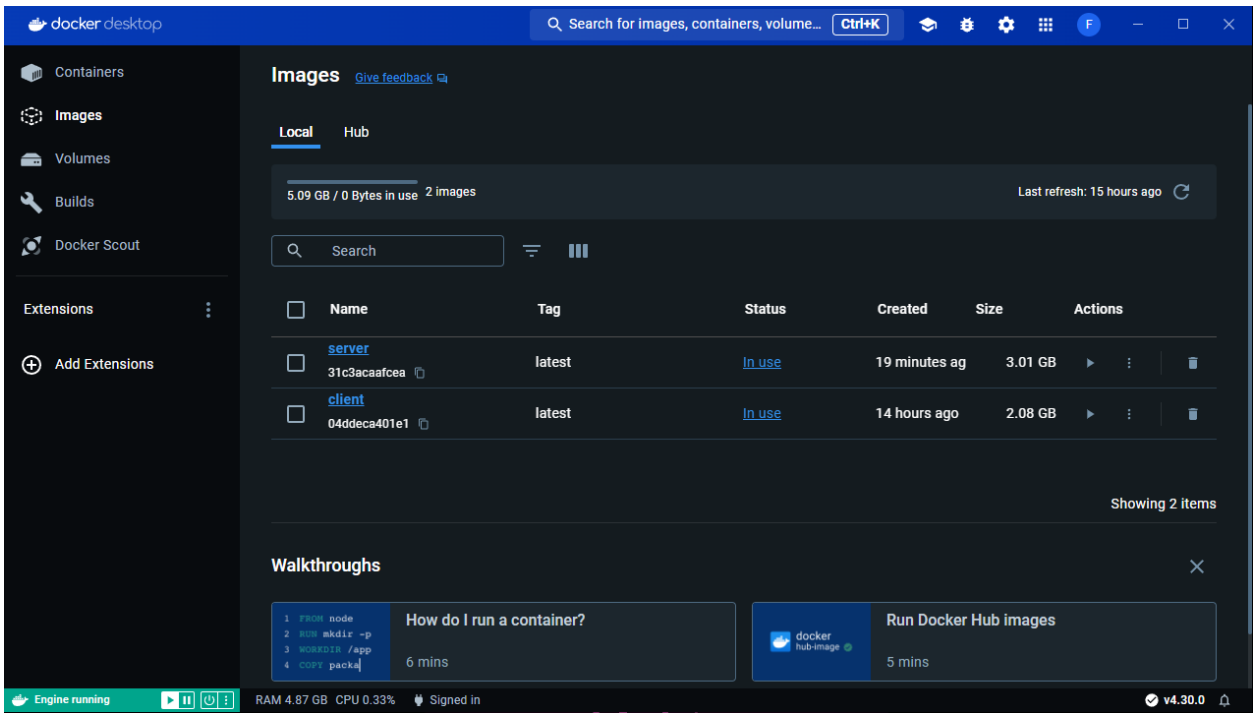


Рисунок 5.1 – Docker-образи клієнтьсокї та серверної частини

Створемо файл docker-compose.yml у кореневій директорії проекту з наступною конфігурацією рисунок 5.2 та виконаємо команду запуску “docker-compose up -d”. Docker Compose автоматично створить та запустить необхідні контейнери, результат можна побачити на рисунку 5.3.

```
docker-compose.yml
1   version: '3'
2   services:
3     server:
4       image: fl1nk/server:latest
5       ports:
6         - '5000:3001'
7     client:
8       image: fl1nk/client:latest
9       ports:
10        - '3000:3000'
11
```

Рисунок 5.2 – Конфігурація docker-compose.yml

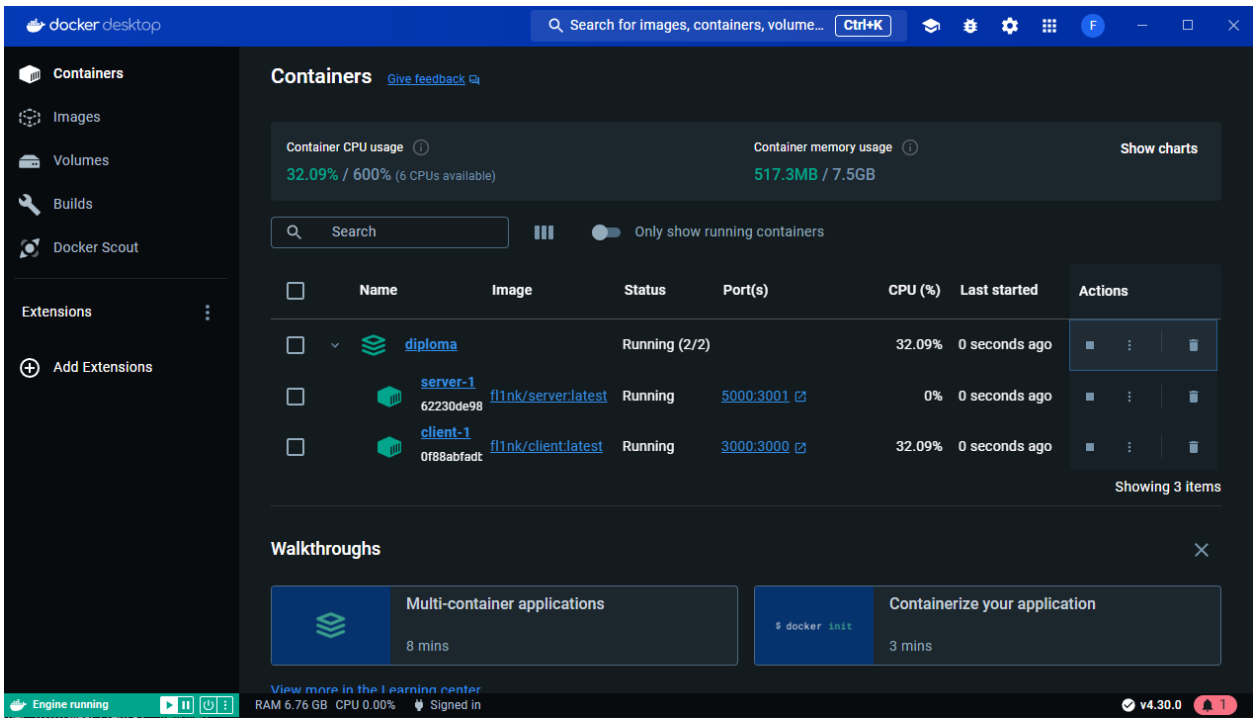


Рисунок 5.3 – Запущені контейнери завдяки docker-compose.yml

Для того щоб переконатися у правильних діях з docker для запуску застосунку, відкриваємо браузер та введемо URL нашого клієнта. Програмне забезпечення має бути доступним для користувача.

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі.

Оновлення системи за допомогою Docker дозволяє мінімізувати простої та зменшити ризики помилок, пов'язаних з оновленням програмного забезпечення. Процес включає кілька основних кроків: створення нового Docker-образу, завантаження його до Docker Hub, зупинка поточних контейнерів та запуск нових контейнерів з оновленими образами.

Розглянемо покрокове оновлення:

– Виконаємо команду “docker build –t name .” для створення Docker-образу в клієнтській та серверній директорії, тоді в нас повинно вийти два образи.

– Для того щоб надіслати наші образи до Docker Hub, потрібно для початку увійти в свій акаунт, це можна зробити завдяки Docker Desktop. Потім потрібно виконати команди “docker tag name fl1nk/name” та “docker push

fl1nk/name", як для директорії клієнтської, так і для серверної. На рисунку 5.4 можна побачити надіслані Docker-образи до Docker Hub.

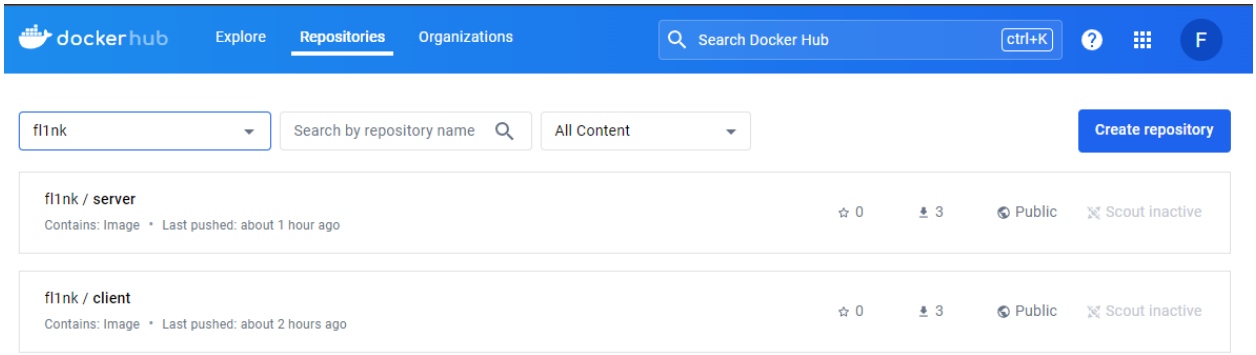


Рисунок 5.4 – Надіслані образи до Docker Hub

Для завантаження останніх версій з Docker Hub та використання на своєму сервері, потрібно виконати команду "docker-compose pull", якщо ми використовуємо конфігурацію docker-compose.yml з попереднього розділу.

Висновки до розділу

У цьому розділі було детально розглянуто процеси розгортання та супроводу програмного забезпечення, що дозволяють забезпечити ефективне та безперебійне функціонування системи. Розгортання програмного забезпечення здійснювалося за допомогою Docker та Docker Hub, що надали можливість створення ізольованих середовищ для додатків та забезпечили легкість у керуванні версіями програмного забезпечення. Docker дозволяє створювати контейнери, які включають всі необхідні компоненти програмного забезпечення, а Docker Hub забезпечує зручний спосіб зберігання та розповсюдження цих контейнерів.

Покрокова інструкція з розгортання включала встановлення Docker на сервер, створення файлів Dockerfile для клієнтської та серверної частин додатку, створення Docker-образів, завантаження їх до Docker Hub, створення та конфігурацію файлу docker-compose.yml для автоматизації запуску контейнерів та перевірку доступності програмного забезпечення через браузер. Ці кроки дозволили забезпечити швидке та безпечне розгортання системи, мінімізуючи ризики помилок та збоїв.

Супровід програмного забезпечення включав інструкцію користувача, наведена в окремому документі, що забезпечує зручність користування системою. Оновлення програмного забезпечення за допомогою Docker. Це дозволило мінімізувати простої та забезпечити безперебійне оновлення системи з мінімальними ризиками помилок та збоїв.

Результати впровадження описаних процесів розгортання та супроводу програмного забезпечення дозволили забезпечити ефективне та безперебійне функціонування системи, можливість швидкого оновлення додатку та значно спростили ці процеси, роблячи їх більш зручними та надійними. Використання Docker та Docker Hub значно спростило ці процеси, забезпечуючи надійність та безпеку розгортання та супроводу програмного забезпечення.

ВИСНОВКИ

В результаті виконання дипломного проєкту було спроєктовано та реалізовано автоматизовану систему табелювання співробітників. У процесі роботи було досягнуто важливих практичних результатів, що відповідають сучасному рівню технічних знань.

У першому розділі було проведено детальний аналіз предметної області та визначено основні вимоги до автоматизованої системи табелювання співробітників. Було виявлено ключові процеси, що підлягають автоматизації, а також встановлено необхідні функціональні та нефункціональні вимоги до системи. Це дозволило створити чітке уявлення про майбутню систему та закласти міцний фундамент для її розробки.

У другому розділі було сформульовано детальні вимоги до програмного забезпечення, включаючи функціональні вимоги, які визначають основні операції та процеси, що підтримуються системою, а також нефункціональні вимоги, такі як продуктивність, надійність та безпека. Було також визначено вимоги до інтерфейсу користувача та сумісності з різними браузерами та пристроями. Це забезпечило структурований підхід до розробки та зменшило ризики виникнення помилок у процесі реалізації.

У третьому розділі було створено архітектуру системи, яка включає серверну частину, клієнтську частину та базу. Було реалізовано всі необхідні функції, включаючи авторизацію користувачів, створення та редагування профілів співробітників, перегляд та завантаження звітів.

В якості середовища розробки обрано Node.js та React, що забезпечує ефективний та гнучкий підхід до розробки як серверної, так і клієнтської частин додатку.

У якості БД використано MongoDB, що дозволило забезпечити зручне та надійне зберігання даних про користувачів і результати їхньої діяльності.

У четвертому розділі Проведено детальний аналіз якості програмного забезпечення за допомогою статичних та динамічних методів тестування. Також було проведено мануальне тестування вебзастосунку.

Після реалізації застосунку він був протестований на різних пристроях, щоб переконатися, що додаток акуратно відображається на різних екранах і працює коректно в різних браузерах. Це забезпечило високу якість кінцевого продукту та відповідність функціональним вимогам.

У п’ятому розділі на етапі розгортання було використано технології Docker та Docker Hub, що забезпечило зручність та ефективність процесу розгортання програмного забезпечення на різних середовищах. Створення Docker-образів та їх зберігання на Docker Hub спростило управління версіями програмного забезпечення та забезпечило можливість швидкого оновлення системи. У розділі супроводу описано покроковий процес оновлення системи за допомогою Docker, що мінімізує простої та забезпечує безперервність роботи.

В результаті виконання дипломного проєкту було спроєктовано та реалізовано автоматизовану систему табелювання співробітників, яка відповідає сучасним вимогам та забезпечує високу продуктивність, надійність та зручність використання. Використання передових технологій, таких як Node.js, React, MongoDB та Docker, дозволило створити гнучку та масштабовану систему, яку легко розгорнути та підтримувати. Подальші дослідження та розвиток системи можуть бути спрямовані на додавання нових функціональних можливостей та адаптацію до потреб різних підприємств та організацій, що сприятиме підвищенню ефективності управління персоналом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Автоматичний табель обліку робочого часу: Принцип роботи // Вільний блог. URL: https://yaware.com.ua/uk/blog/avtomatichnij-tabel-obliku-robochogo-chasu-princzip-roboti/ (дата звернення 01.05.2024).

2) Toggl. Toggl Track: Time Tracking Software for Any Workflow. URL: https://toggl.com/ (дата звернення 01.05.2024).

3) Clockify - FREE Time Tracking Software. Clockify. URL: https://clockify.me/ (дата звернення 01.05.2024).

4) Hubstaff | Time Tracking and Productivity Monitoring Tool. Hubstaff. URL: https://hubstaff.com/ (дата звернення 01.05.2024).

5) Client-server model // Вікіпедія: вільна енциклопедія. URL: https://en.wikipedia.org/wiki/Client%E2%80%93server_model (дата звернення 01.05.2024).

6) Python. Official documentation. URL: https://docs.python.org/3/ (дата звернення 01.05.2024).

7) Monolith First. Martinfowler. URL: https://martinfowler.com/bliki/MonolithFirst.html (дата звернення 01.05.2024).

8) Microservise. Martinfowler. URL: https://martinfowler.com/articles/microservices.html (дата звернення 01.05.2024).

9) NodeJS. Official documentation. URL: https://nodejs.org/en (дата звернення 01.05.2024).

10) React. Official documentation. URL: https://react.dev/learn (дата звернення 01.05.2024).

11) MongoDB. Official documentation. URL: https://www.mongodb.com/docs/manual/ (дата звернення 01.05.2024).

12) Flask. Official documentation. URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення 01.05.2024).

13) Gjango. Official documentation. URL: https://docs.djangoproject.com/en/5.0/ (дата звернення 01.05.2024).

14) Ruby. Офіційний сайт. URL: https://www.ruby-lang.org/ru/ (дата звернення 01.05.2024).

15) JavaScript // Вікіпедія: вільна енциклопедія. URL: https://en.wikipedia.org/wiki/JavaScript (дата звернення 01.05.2024).

16) Angular. Офіційний сайт. URL: https://angular.io/ (дата звернення 01.05.2024).

17) VueJS. Офіційний сайт. URL: https://vuejs.org/ (дата звернення 01.05.2024).

18) PostgresSQL. Офіційний сайт. URL: https://www.postgresql.org/ (дата звернення 01.05.2024).

19) MySQL. Офіційний сайт. URL: https://www.mysql.com/ (дата звернення 01.05.2024).

20) Visual Studio Code. Офіційний сайт. URL: https://code.visualstudio.com/ (дата звернення 01.05.2024).

21) Git. Офіційний сайт. URL: https://git-scm.com/ (дата звернення 01.05.2024).

22) WebStorm. Офіційний сайт. URL: https://www.jetbrains.com/ru-ru/webstorm/ (дата звернення 01.05.2024).

23) Atom. Офіційний сайт. URL: https://atom-editor.cc/ (дата звернення 01.05.2024).

24) GitHub. Офіційний сайт. URL: https://github.com/ (дата звернення 01.05.2024).

25) FaceAPI. Публічна бібліотека npm. URL: https://www.npmjs.com/package/@vladmandic/face-api?activeTab=readme (дата звернення 01.05.2024).

26) TensorFlow. Публічна бібліотека npm. URL: https://www.npmjs.com/package/@tensorflow/tfjs-node (дата звернення 01.05.2024).

27) ExcelJS. Публічна бібліотека npm. URL: https://www.npmjs.com/package/exceljs (дата звернення 01.05.2024).

28) Excel. Офіційний сайт. URL: https://www.microsoft.com/uk-ua/microsoft-365/excel (дата звернення 01.05.2024).

29) SonarCloud. Офіційний сайт. URL: https://www.sonarsource.com/products/sonarcloud/ (дата звернення 01.05.2024).

30) Express-status-monitor. Публічна бібліотека npm. URL: https://www.npmjs.com/package/express-status-monitor (дата звернення 01.05.2024).

31) Docker. Офіційний сайт. URL: https://www.docker.com/ (дата звернення 01.05.2024).

32) Computer vision based employee activities analysis / Hoque Md Rezwanul та ін. *BracU IR*. URL: https://dspace.bracu.ac.bd/xmlui/handle/10361/2747 (дата звернення 01.05.2024).

33) Neuhausen M., Teizer J., König M. Construction Worker Detection and Tracking in Bird's-Eye View Camera Images. URL: https://www.researchgate.net/publication/326689558_Construction_Worker_Detection_and_Tracking_in_Bird's-Eye_View_Camera_Images (дата звернення 01.05.2024).

ДОДАТКИ

UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016334648

Дата перевірки:
08.06.2024 19:33:50 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2024 07:48:20 EEST

ID користувача:
76913

Назва документа: ІТ-01_Філоненко_ПЗ

Кількість сторінок: 55 Кількість слів: 7504 Кількість символів: 62626 Розмір файлу: 1.65 MB ID файлу: 1016135103

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

19%
Схожість

Найбільша схожість: 7.76% з джерелом з Бібліотеки (ID файлу: 1016100816)

4.36% Джерела з Інтернету 163 ..... Сторінка 57

18.6% Джерела з Бібліотеки 380 ..... Сторінка 58

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 12 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТАБЕЛЮВАННЯ СПІВРОБІТНИКІВ

Текст програми

КПІ. ІТ-0320.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр ОЧЕРЕТЯНИЙ

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Владислав ФІЛОНЕНКО

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://github.com/fl1nK/diploma

Файл authController.js
Реалізація функціональної вимоги авторизації користувача у систему.

```
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken')
const Admin = require('../models/Admin');
const registration = async (req, res) => {
    try{
        const {password} = req.body;
        const email = req.body.email.toString();
        const admin = await Admin.findOne({ email: email });
        if (admin) {
            return res.status(400).json({message: `Адмін з таким логіном
${email} вже існує!`})
        }
        const hashPassword = bcrypt.hashSync(password, 7);
        const user = new Admin({email, password: hashPassword})
        await user.save()
        return res.status(201).json({message: "Адміна було створено!"})
    } catch (e) {
        console.log(e)
        res.send({message: "Server error"})
    }
}

const login = async (req, res) => {
    try{
        const {password} = req.body;
        const email = req.body.email.toString();
        const admin = await Admin.findOne({ email: email });
        if (!admin) {
            return res.status(404).json({message: "Такого адмін не існує!"})
        }
        const isPassValid = bcrypt.compareSync(password, admin.password);
        if (!isPassValid) {
            return res.status(400).json({message: "Невірний логін чи
пароль"})
        }
        const token = jwt.sign({id: admin.id}, process.env.SECRET_KEY,
{expiresIn: "1h"})
```

```
        res.json({
            token:token,
            message: 'Користувач успішно авторизувався'
        })
    } catch (e) {
        console.log(e)
        res.send({message: "Server error"})
    }
}
module.exports = {
    registration,
    login,
};
```

Файл faceController.js

Реалізація функціональної вимоги створення, редагування, видалення профілю робітника та можливість перегляду звіту.

```
const fs = require('fs');
const path = require('path');
const faceapi = require('@vladmandic/face-api');
require('@tensorflow/tfjs-node');
const { Canvas, Image } = require('canvas');
faceapi.env.monkeyPatch({ Canvas, Image });
const canvas = require('canvas');
const FaceModel = require('../models/Face');
const ImageModel = require('../models/Image');
const DetectedUser = require('../models/DetectedUser');
const { createUserDB, savePhoto } = require('./utils/faceUtils');

async function getAllUser(req, res) {
  try {
    const faces = await FaceModel.find();
    return res.json(faces);
  } catch (error) {
    return res.status(500).json({ error: error.message });
  }
}

async function createUser(req, res) {
  const label = req.body;
  const result = await createUserDB(req.files, label);
  if (result) {
```

```
    return res.status(200).json({ message: 'Дані робітника успішно збережено'
});
  } else {
    return res.status(500).json({ message: 'Щось пішло не так, спробуйте ще
раз.' });
  }
}
async function getUser(req, res) {
  const { id } = req.params;
  try {
    const user = await FaceModel.findById(id);
    if (!user) {
      return res.status(404).json({ message: 'Користувач не знайдений' });
    }
    const images = await ImageModel.find({ _id: { $in: user.images } });
    user.images = images;
    return res.json(user);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
}

async function putUser(req, res) {
  const { id } = req.params;
  const { firstName, lastName, middleName, entryTime, outTime } = req.body;
  try {
    const user = await FaceModel.findByIdAndUpdate(
      id,
      { firstName, lastName, middleName, entryTime, outTime },
      { new: true },
    );
    if (!user) {
      return res.status(404).json({ message: 'Користувач не знайдений' });
    }
    res.status(200).json(user);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
}

async function deleteUser(req, res) {
```

```
  try {
    const { id } = req.params;
    const user = await FaceModel.findById(id);
    if (!user) {
      return res.status(404).json({ message: 'Користувач не знайдений' });
    }
    for (const imageId of user.images) {
      const image = await ImageModel.findById(imageId);
      if (image) {
        const filePath = path.join(__dirname, '../data', image.url);
        fs.unlink(filePath, (err) => {
          if (err) {
            console.error(`Не вдалося видалити файл: ${filePath}`, err);
          }
        });
        await ImageModel.findByIdAndDelete(imageId);
      }
    }
    await DetectedUser.deleteMany({ userID: id });
    await FaceModel.findByIdAndDelete(id);
    return res.status(200).json({ message: 'Дані успішно видалені' });
  } catch (e) {
    console.log(e);
    return res.status(500).json({ message: 'Server error' });
  }
}

async function uploadImageForUser(req, res) {
  const { id } = req.params;
  try {
    const user = await FaceModel.findById(id);
    if (!user) {
      return res.status(404).json({ message: 'Користувач не знайдений' });
    }
    const { files } = req;
    if (!files || Object.keys(files).length === 0) {
      return res.status(400).json({ message: 'Файл не було завантажено' });
    }
    const fieldName = `image`;
    const img = await canvas.loadImage(files[fieldName].tempFilePath);
    const detections = await
faceapi.detectSingleFace(img).withFaceLandmarks().withFaceDescriptor();
    let url = await savePhoto(files[fieldName]);
```

```
    const newImage = new ImageModel({
      url: url,
      descriptions: detections.descriptor,
    });
    const newImageModel = await newImage.save();
    user.images.push(newImageModel);
    await user.save();
    res.status(200).json(user);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
}

async function deleteImageForUser(req, res) {
  const { userId, imageId } = req.params;
  try {
    const user = await FaceModel.findById(userId);
    if (!user) {
      return res.status(404).json({ message: 'Користувач не знайдений' });
    }
    const image = await ImageModel.findById(imageId);
    if (!image) {
      return res.status(404).json({ message: 'Фотографія не знайдена' });
    }
    const filePath = path.join(__dirname, '../data', image.url);
    fs.unlink(filePath, (err) => {
      if (err) {
        console.error(`Не вдалося видалити файл: ${filePath}`, err);
      }
    });
    user.images.pull(imageId);
    await user.save();
    await ImageModel.findByIdAndDelete(imageId);
    res.status(200).json({ message: 'Фотографія успішно видалена' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
}

async function getAllDescriptors(req, res) {
  try {
```

```
    let users = await FaceModel.find().lean();
    users.forEach((obj) => {
      obj['descriptions'] = [];
    });
    for (const user of users) {
      for (const imageId of user.images) {
        const image = await ImageModel.findById(imageId);
        if (image) {
          user.descriptions.push(image.descriptions[0]);
        }
      }
      user.images = '';
    }
    // console.log(users);
    return res.json(users);
  } catch (error) {
    return res.status(500).json({ error: error.message });
  }
}

async function setDetectedUser(req, res) {
  try {
    const { id } = req.body;

    const user = await FaceModel.findById(id);

    if (!user) {
      return res.status(404).json({ message: 'Користувач не знайдений' });
    }
    const entryTime = user.entryTime;
    const outTime = user.outTime;
    const currentDate = new Date();
    const currentTime = currentDate.getHours() * 60 +
currentDate.getMinutes(); // Переведемо поточний час в хвилини для зручності
порівняння
    const entryTimeMinutes = parseInt(entryTime.slice(0, 2)) * 60 +
parseInt(entryTime.slice(3, 5));
    const outTimeMinutes = parseInt(outTime.slice(0, 2)) * 60 +
parseInt(outTime.slice(3, 5));
    let status;
    if (currentTime < entryTimeMinutes) {
      status = 'Раніше';
```

```
    } else if (currentTime >= entryTimeMinutes && currentTime <=
outTimeMinutes) {
      status = 'Запізнюється';
    } else if (currentTime > outTimeMinutes) {
      status = 'Закінчив';
    }
    const formattedDate = currentDate.toLocaleDateString('uk-UA');
    const formattedTime = currentDate.toLocaleTimeString('uk-UA');
    const createFace = new DetectedUser({
      userID: id,
      date: formattedDate,
      time: formattedTime,
      status: status,
    });
    await createFace.save();
  } catch (error) {
    return res.status(500).json({ error: error.message });
  }
}
async function getDetectedAllUsers(req, res) {
  try {
    const detectedUsers = await DetectedUser.find()
      .populate('userID', 'firstName lastName middleName entryTime outTime')
      .sort({ date: -1, time: -1 });
    return res.json(detectedUsers);
  } catch (error) {
    return res.status(500).json({ error: error.message });
  }
}

async function deleteDetectedUser(req, res) {
  try {
    const { id } = req.params;
    const detectedUser = await DetectedUser.findByIdAndDelete(id);
    if (!detectedUser) {
      return res.status(404).json({ message: 'Користувача не знайдено' });
    }
    return res.status(200).json({ message: 'Дані успішно видалені' });
  } catch (error) {
    return res.status(500).json({ error: error.message });
  }
}
module.exports = {
```

```
  getAllUser,
  getUser,
  createUser,
  putUser,
  deleteUser,
  uploadImageForUser,
  deleteImageForUser,
  getAllDescriptors,
  setDetectedUser,
  getDetectedAllUsers,
  deleteDetectedUser,
};
```

Файл faceUtils.js

```
const FaceModel = require('../../models/Face');
const ImageModel = require('../../models/Image');
const fs = require('fs');
const path = require('path');
const { v4: uuidv4 } = require('uuid');
const faceapi = require('@vladmandic/face-api');
require('@tensorflow/tfjs-node');
const { Canvas, Image } = require('canvas');
faceapi.env.monkeyPatch({ Canvas, Image });
const canvas = require('canvas');

async function getAllDescriptorsFromDB() {
  let faces = await FaceModel.find();
  for (let i = 0; i < faces.length; i++) {
    for (let j = 0; j < faces[i].descriptions.length; j++) {
      faces[i].descriptions[j] = new
Float32Array(Object.values(faces[i].descriptions[j]));
    }
    faces[i] = new faceapi.LabeledFaceDescriptors(faces[i].label,
faces[i].descriptions);
  }
  return faces;
}

async function createUserDB(files, label) {
  try {
    const filesCount = Object.keys(files).length;
    const { firstName, lastName, middleName, entryTime, outTime } = label;
    const arrayIdImageModel = [];
```

```
    for (let i = 1; i <= filesCount; i++) {
      const fieldName = `photo${i}`;

      const img = await canvas.loadImage(files[fieldName].tempFilePath);

      const detections = await faceapi
        .detectSingleFace(img)
        .withFaceLandmarks()
        .withFaceDescriptor();
      let url = await savePhoto(files[fieldName]);
      const newImage = new ImageModel({
        url: url,
        descriptions: detections.descriptor,
      });
      const newImageModel = await newImage.save();
      arrayIdImageModel.push(newImageModel._id);
    }

    const createFace = new FaceModel({
      firstName: firstName,
      lastName: lastName,
      middleName: middleName,
      images: arrayIdImageModel,
      entryTime: entryTime,
      outTime: outTime,
    });
    await createFace.save();
    return true;
  } catch (error) {
    console.log(error);
    return false;
  }
}

async function savePhoto(photo) {
  return new Promise((resolve, reject) => {
    const tempPath = photo.tempFilePath;
    const uniqueFilename = uuidv4() + path.extname(photo.name);

    const targetPath = path.join('data', uniqueFilename);
    const readStream = fs.createReadStream(tempPath);
    const writeStream = fs.createWriteStream(targetPath);
    readStream.on('error', (err) => {
```

```
      reject(err);
    });
    writeStream.on('error', (err) => {
      reject(err);
    });
    writeStream.on('finish', () => {
      resolve(uniqueFilename);
    });
    readStream.pipe(writeStream);
  });
}

async function loadModels() {
  await faceapi.nets.faceRecognitionNet.loadFromDisk('./models/faceApi');
  await faceapi.nets.faceLandmark68Net.loadFromDisk('./models/faceApi');
  await faceapi.nets.ssdMobilenetv1.loadFromDisk('./models/faceApi');
}

module.exports = {
  loadModels,
  getAllDescriptorsFromDB,
  createUserDB,
  savePhoto,
};
```

Файл excelController.js

Реалізація функціональної вимоги завантаження звіту у форматі Excel файлу.

```
const excelJS = require('exceljs');
const exportUser = async (req, res) => {
  const User = req.body;
  const workbook = new excelJS.Workbook();
  const worksheet = workbook.addWorksheet('Виявлені люди');
  worksheet.columns = [
    { header: 'Імя', key: 'firstName', width: 15 },
    { header: 'Прізвище', key: 'lastName', width: 15 },
    { header: 'По батькові', key: 'middleName', width: 15 },
    { header: 'Початок роботи', key: 'entryTime', width: 15 },
    { header: 'Кінець роботи', key: 'outTime', width: 15 },
    { header: 'День', key: 'date', width: 15 },
    { header: 'Час', key: 'time', width: 15 },
    { header: 'Статус', key: 'status', width: 15 },
  ];
```

```
  let counter = 1;
  User.forEach((user) => {
    worksheet.addRow({
      firstName: user.userID.firstName,
      lastName: user.userID.lastName,
      middleName: user.userID.middleName,
      entryTime: user.userID.entryTime,
      outTime: user.userID.outTime,
      date: user.date,
      time: user.time,
      status: user.status,
    });
    counter++;
  });
  worksheet.getRow(1).eachCell((cell) => {
    cell.font = { bold: true };
  });
  try {
    res.setHeader(
      'Content-Type',
      'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',
    );
    res.setHeader('Content-Disposition', `attachment; filename=users.xlsx`);
    return workbook.xlsx.write(res).then(() => {
      res.status(200);
    });
  } catch (err) {
    res.send({
      status: 'error',
      message: 'Something went wrong',
    });
  }
};
module.exports = { exportUser };
```

Файл websocketServer.js

Реалізація функціональної вимоги виявлення робітників на відео.

```
const { Server } = require('ws');
const tf = require('@tensorflow/tfjs-node');
const faceapi = require('@vladmandic/face-api');
const http = require('http');
```

```
const FaceModel = require('./models/Face');
const DetectedUser = require('./models/DetectedUser');

const loadModels = async () => {
  await faceapi.nets.ssdMobilenetv1.loadFromDisk('./models/faceApi');
  await faceapi.nets.faceLandmark68Net.loadFromDisk('./models/faceApi');
  await faceapi.nets.faceRecognitionNet.loadFromDisk('./models/faceApi');
};

loadModels();

const startWebSocketServer = (app) => {
  const server = http.createServer(app);
  const wss = new Server({ server });

  wss.on('connection', (ws) => {
    ws.on('message', async (message) => {
      try {
        const videoBuffer = Buffer.from(message);
        const videoTensor = tf.node.decodeImage(videoBuffer);

        const faces = await getAllDescriptors();

        const labeledFaceDescriptors = faces.map((face) => {
          const descriptors = face.descriptions.map(
            (desc) => new Float32Array(Object.values(desc)),
          );
              return  new  faceapi.LabeledFaceDescriptors(face._id.toString(),
descriptors);
        });

        const faceMatcher = new faceapi.FaceMatcher(labeledFaceDescriptors);

        if (videoTensor.shape.length === 3 && videoTensor.shape[2] === 3) {
          const detections = await faceapi
            .detectAllFaces(videoTensor)
            .withFaceLandmarks()
            .withFaceDescriptors();

          const results = detections.map((detection) =>
            faceMatcher.findBestMatch(detection.descriptor),
          );
```

```
          for (const result of results) {
            if (result.label !== 'unknown') {
                                const    face    =    faces.find((face)    =>
face._id.equals(result.label));

              await setDetectedUser(result.label);
              ws.send(JSON.stringify({ user: face.lastName }));
            } else {
              ws.send(JSON.stringify({ error: 'Unknown user.' }));
            }
          }
        } else {
          ws.send(JSON.stringify({ error: 'Unexpected tensor shape.' }));
        }
      } catch (error) {
        console.error(error);
        ws.send(JSON.stringify({ error: 'Error processing video.' }));
      }
    });
  });

  const port = process.env.WS_PORT || 5001;
  server.listen(port, () => {
    console.log(`WebSocket server running at http://localhost:${port}`);
  });
};

async function getAllDescriptors() {
  try {
    let users = await FaceModel.find().populate('images').lean();

    for (const user of users) {
      user.descriptions = user.images.map((image) => image.descriptions[0]);
      user.images = '';
    }

    return users;
  } catch (error) {
    console.log(error);
  }
}

async function setDetectedUser(id) {
```

```
  try {
    // Перевірте, чи вже є помітка для цього користувача за останні 5 хвилин
    const fiveMinutesAgo = new Date(Date.now() - 5 * 60 * 1000);

    const existingDetection = await DetectedUser.findOne({
      userID: id,
      createdAt: { $gt: fiveMinutesAgo },
    });

    if (existingDetection) {
      console.log(`Already detected user ${id} within the last 5 minutes.`);
      return;
    }

    const currentDate = new Date();
          const currentTime = currentDate.getHours() * 60 +
currentDate.getMinutes();

    const user = await FaceModel.findById(id);

    const entryTime = user.entryTime;
    const outTime = user.outTime;

       const entryTimeMinutes = parseInt(entryTime.slice(0, 2)) * 60 +
parseInt(entryTime.slice(3, 5));
       const outTimeMinutes = parseInt(outTime.slice(0, 2)) * 60 +
parseInt(outTime.slice(3, 5));

    let status;
    if (currentTime < entryTimeMinutes) {
      status = 'Раніше';
       } else if (currentTime >= entryTimeMinutes && currentTime <=
outTimeMinutes) {
      status = 'Запізнюється';
    } else if (currentTime > outTimeMinutes) {
      status = 'Закінчив';
    }

    const formattedDate = currentDate.toLocaleDateString('uk-UA');
    const formattedTime = currentDate.toLocaleTimeString('uk-UA');

    const createFace = new DetectedUser({
      userID: id,
```

```
      date: formattedDate,
      time: formattedTime,
      status: status,
    });
    await createFace.save();
  } catch (error) {
    console.log(error);
  }
}

module.exports = startWebSocketServer;
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТАБЕЛЮВАННЯ СПІВРОБІТНИКІВ

Програма та методика тестування

КПІ. ІТ-0320. 045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр ОЧЕРЕТЯНИЙ

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Владислав ФІЛОНЕНКО

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є вебзастосунок для автоматизованої системи табелювання співробітників, що являє собою клієнт-серверним вебзастосунком та розроблене для різних операційних систем з підтримкою сучасних браузерів.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування інтерфейсу користувача;

– тестування зручності використання;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТАБЕЛЮВАННЯ СПІВРОБІТНИКІВ

Керівництво користувача

КПІ.ІТ-0320.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр ОЧЕРЕТЯНИЙ

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Владислав ФІЛОНЕНКО

Київ – 2024

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Розробка вебзастосунок для автоматизованої системи табелювання співробітників призначення для підвищення точності та мінімізації підробки табелювання співробітників, за рахунок розробки автоматизованої системі відслідковування працівників із використанням обробки відео камер відеоспостереження. Може використовуватись підприємцями та власниками бізнесу.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

Мінімальна конфігурація технічних засобів:

– тип процесору: з частотою від 1 ГГц;

– об‘єм ОЗП: 512 Мб;

– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

– наявність сучасного браузера, такого як Google Chrome або Mozilla Firefox;

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i5;

– об‘єм ОЗП: 16 Гб;

– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

– наявність сучасного браузера, такого як Google Chrome або Mozilla Firefox;

2.2 Завантаження застосунку

Для отримання доступу до вебзастосунку в будь-якому браузері потрібно відкрити адресу сторінки сайту.

2.3 Перевірка коректної роботи

При успішному завантаженні застосунку повинна відображатись сторінка авторизації для входу у систему.

3 ВИКОНАННЯ ПРОГРАМИ

При переході за посиланням на сторінку вебзастосунку користувачу буде відображено два поля для вводу електронної пошти та паролю для входу у систему (рисунок 3.1).

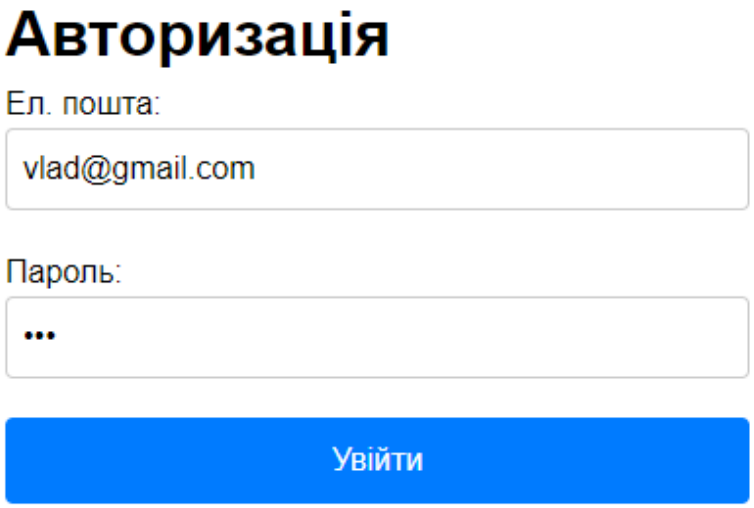


Рисунок 3.1 – Початкова сторінка авторизації

Після успішної авторизації користувач отримує повний доступ до функціоналу програмного забезпечення та направляє на сторінку створення профілю робітника (рисунок 3.2).

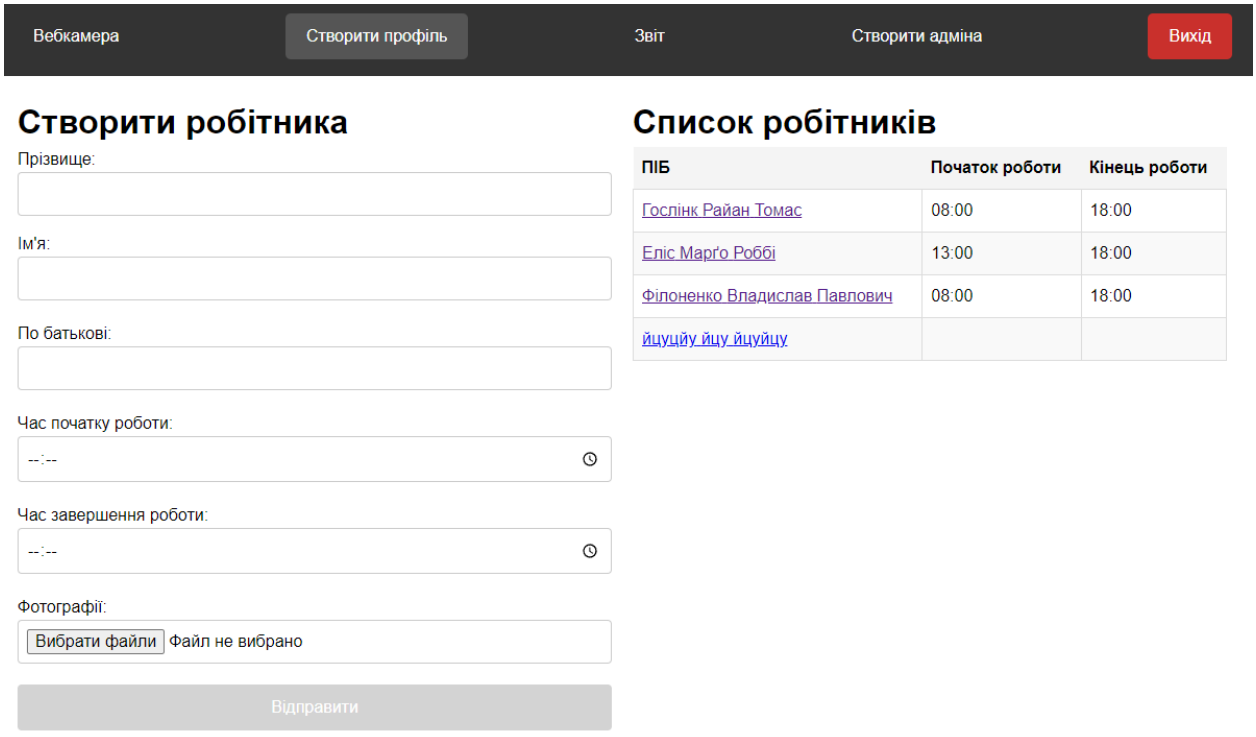


Рисунок 3.2 – Сторінка створення профілю робітника

Користувач може створити профіль робітника заповнюючи форму для створення профілю, потім профіль працівника створюється та додається до списку робітників (рисунок 3.3 – 3.4).

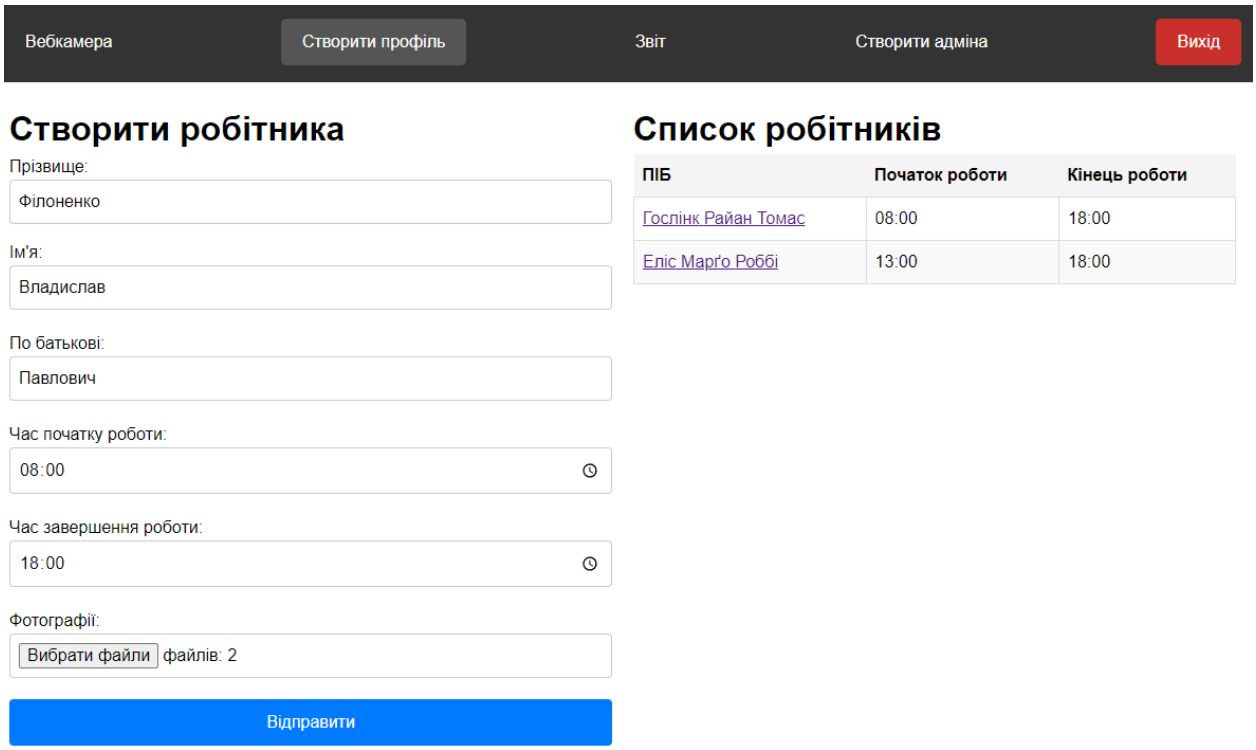


Рисунок 3.3 – Сторінка створення профілю з заповненою формою

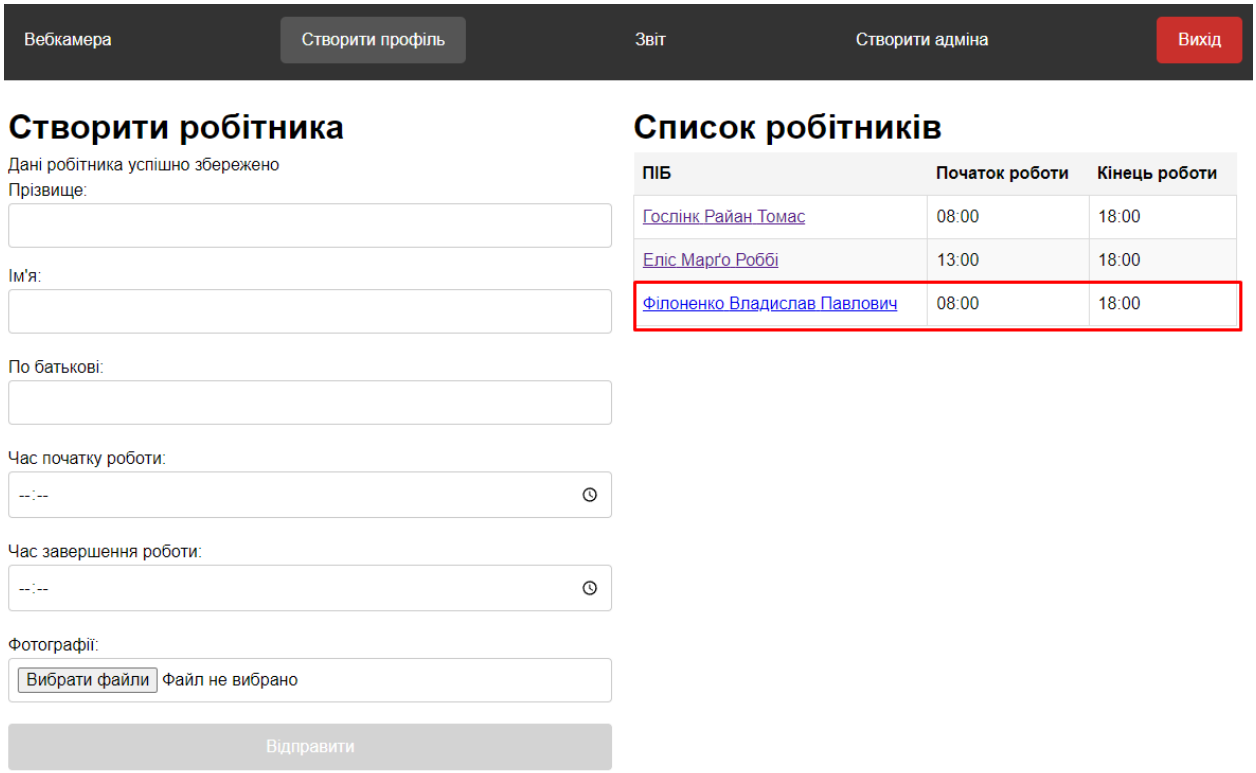


Рисунок 3.4 – Профіль додано до списку

Користувач має можливість перейти на профіль робітника натиснувши на посилання у списку (рисунок 3.5).

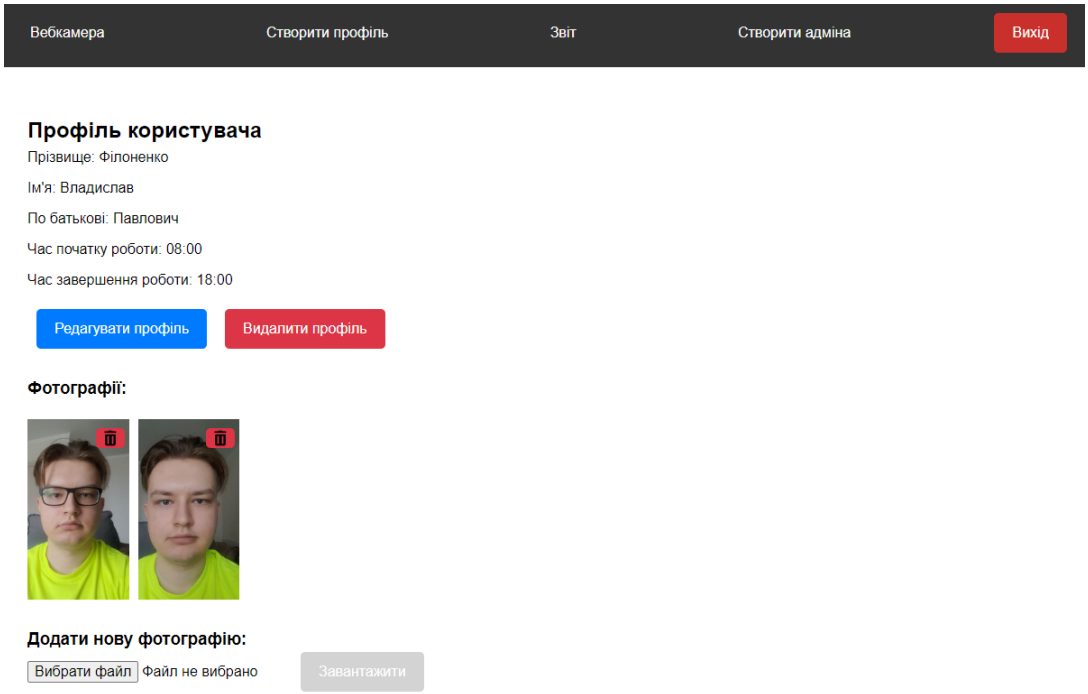


Рисунок 3.5 – Сторінка профілю робітника

На сторінці профілю робітника, користувач має можливість редагувати профіль, натиснувши на кнопку редагувати профіль (рисунок 3.6), видалити профіль, натиснувши кнопку видалити (рисунок 3.7) та можливість завантажити нове фото (рисунок 3.8).

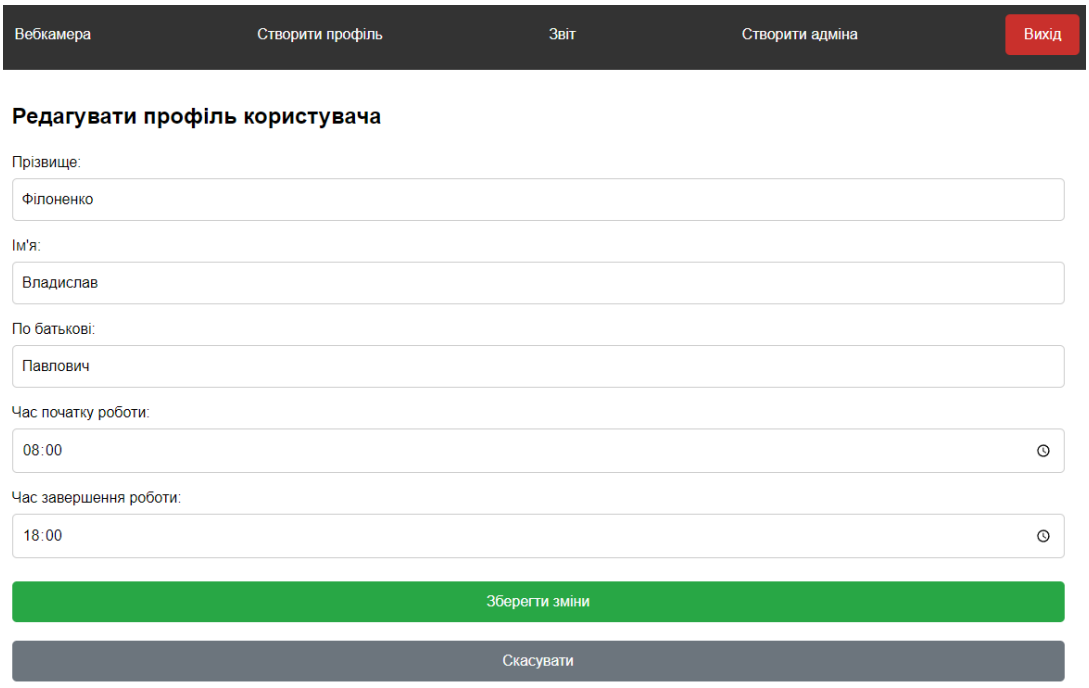


Рисунок 3.6 – Форма редагування профілю робітника

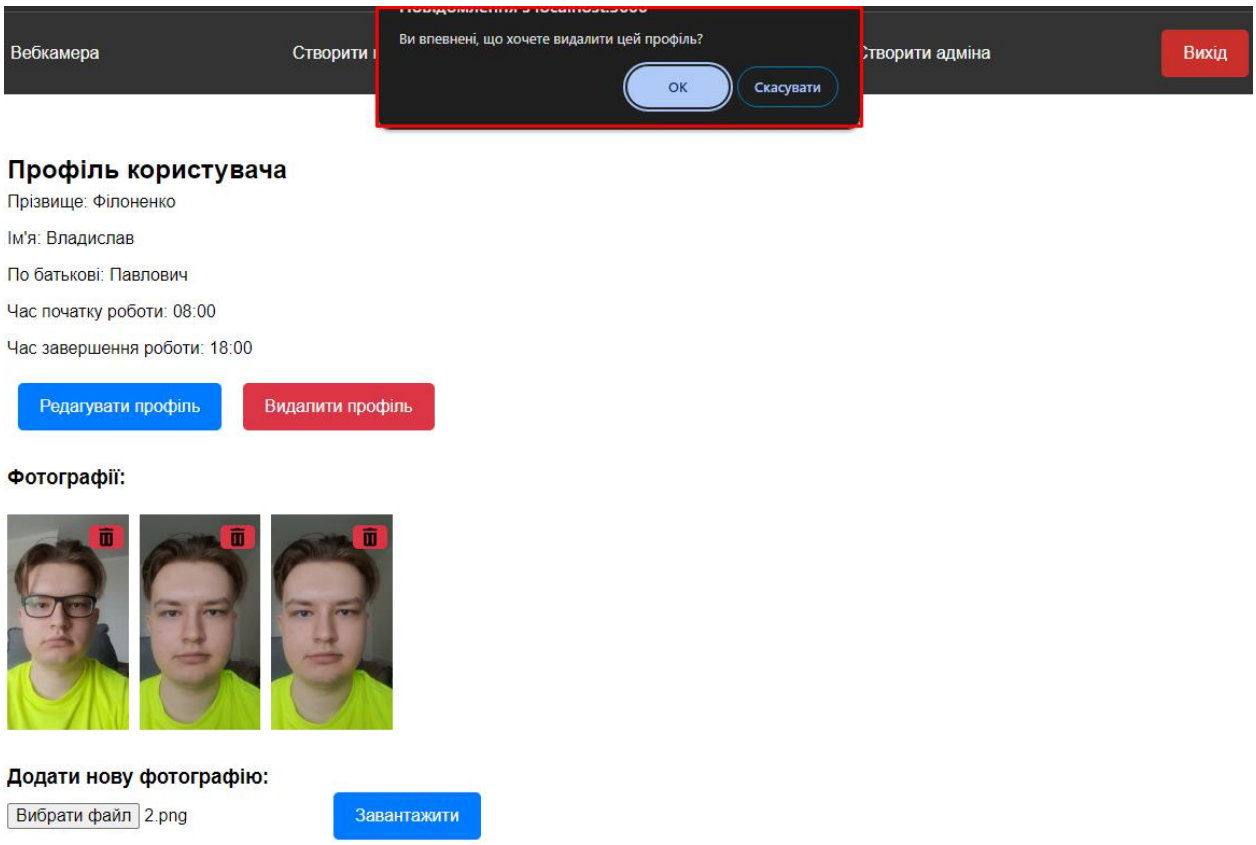


Рисунок 3.7 – Видалення профілю робітника

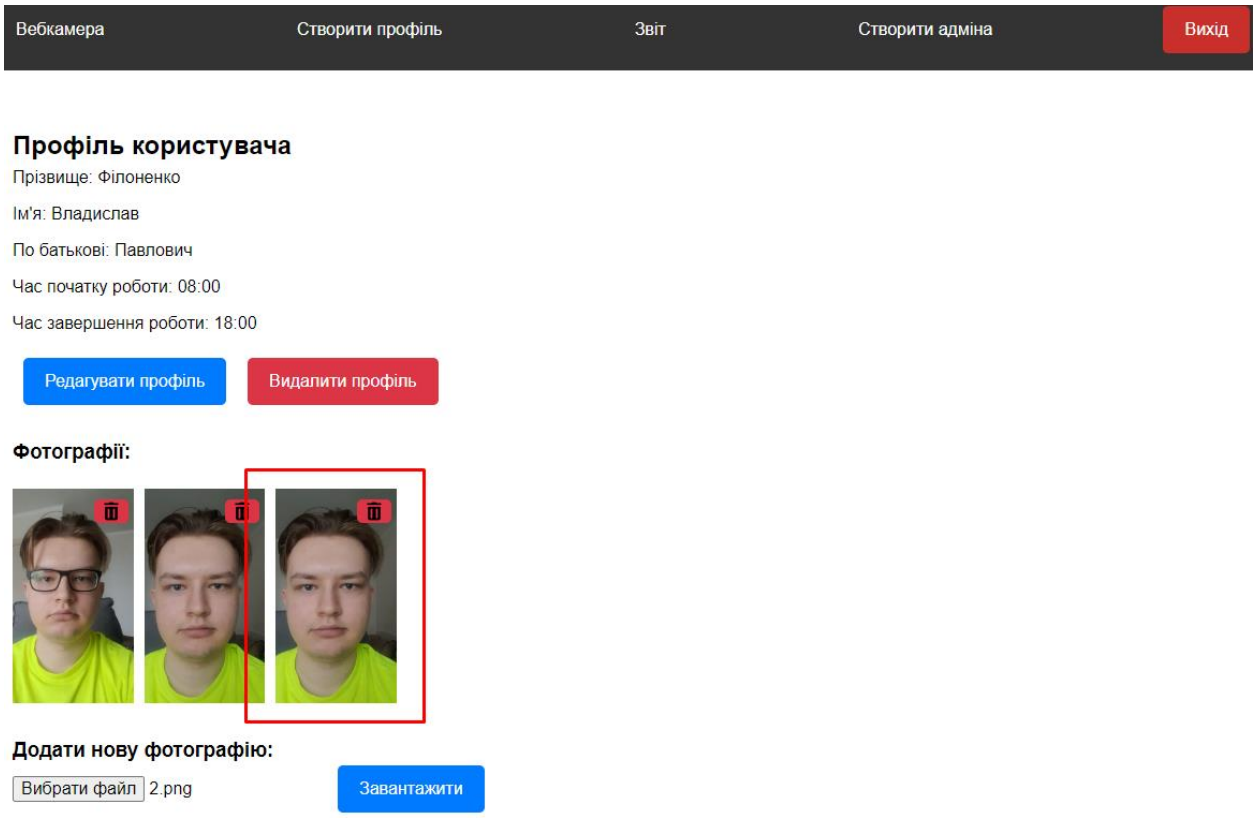


Рисунок 3.8 – Додане нове фото до профілю робітника

Користувач має можливість перейти на сторінку звіту, де може побачити табличку зі всіма робітниками, яких було помічено на відеокамері, також користувач має можливість фільтрувати і сортувати дані (рисунок 3.9).

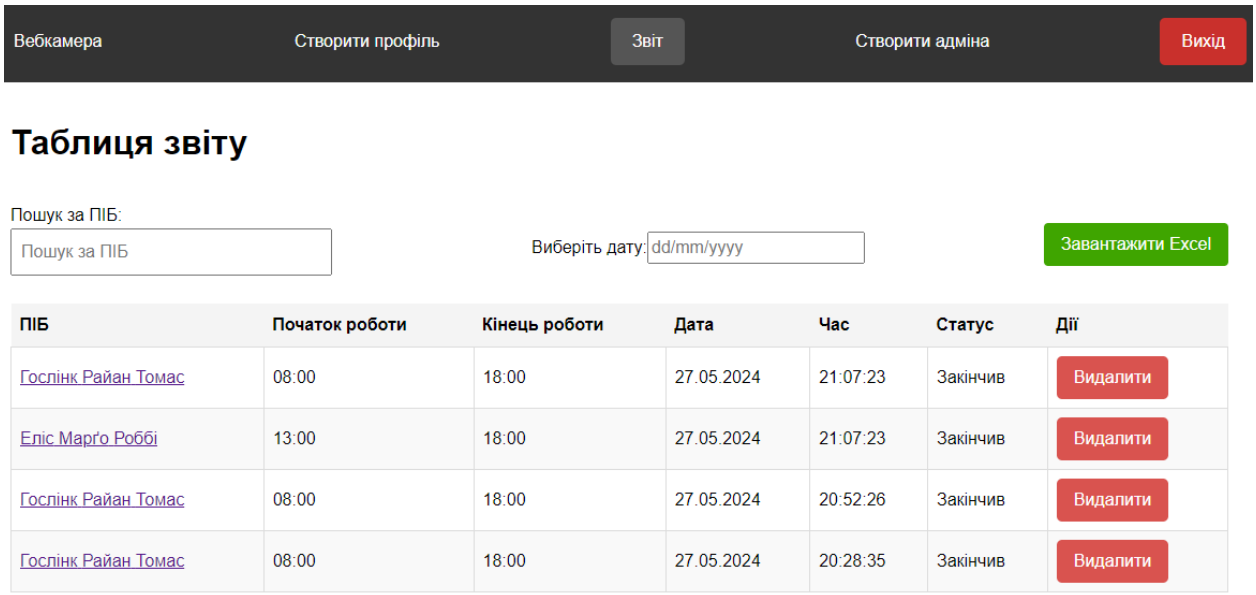


Рисунок 3.9 – Сторінка звіту

Після натиснення на кнопку завантажити Excel, на комп’ютер користувача завантажується файл у форматі Excel (рисунок 3.10).

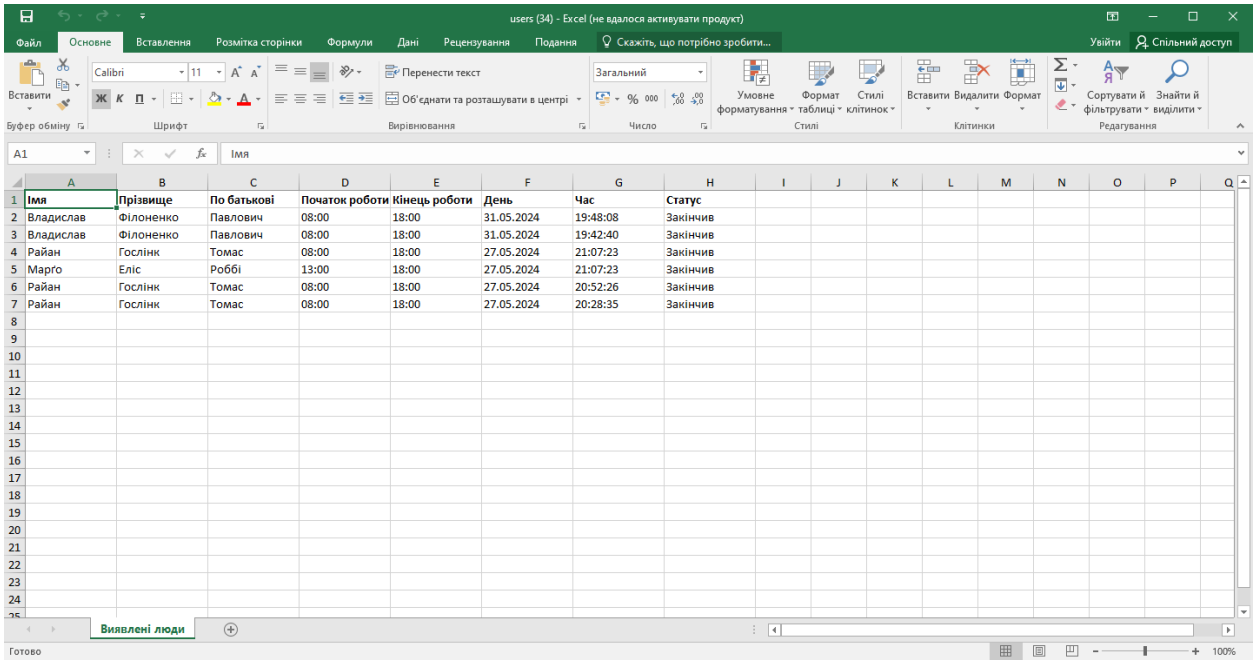


Рисунок 3.10 – Звантажений звіт у форматі Excel

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТАБЕЛЮВАННЯ СПІВРОБІТНИКІВ

Графічний матеріал

КПІ.ІТ-0320.045440.06.99

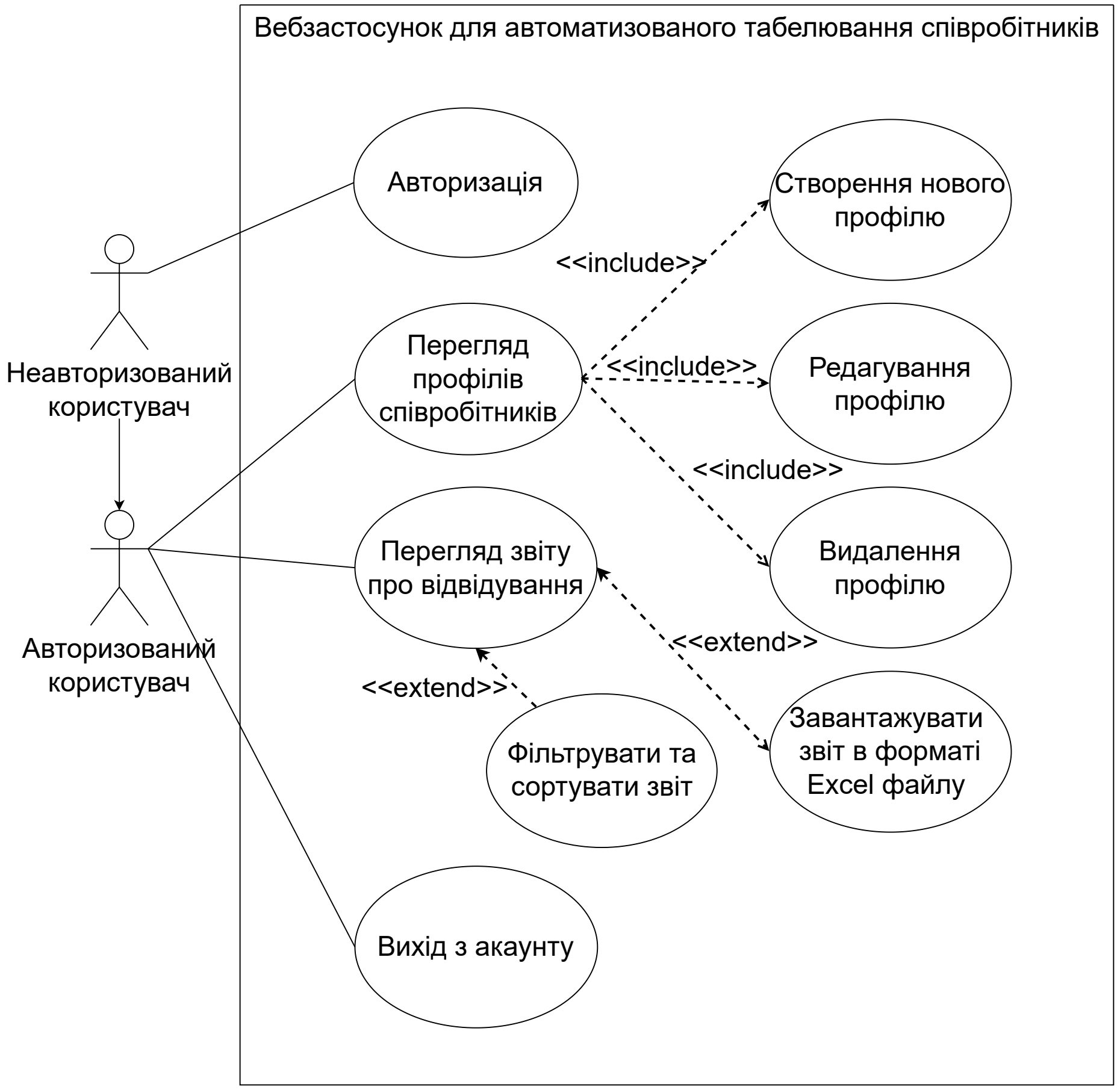
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олександр ОЧЕРЕТЯНИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Владислав ФІЛОНЕНКО

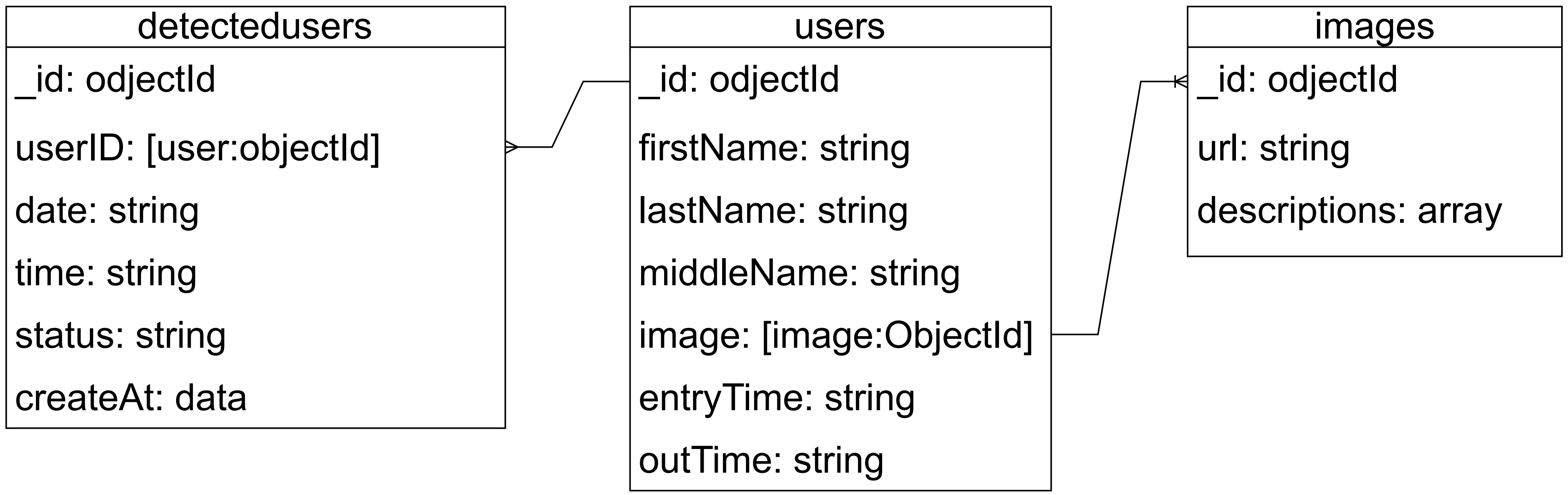
Київ – 2024

КПІ.ІТ-0320.045440.06.99.ССВ



					КПІ.ІТ-0320.045440.06.99.ССВ			
					Схема структурна варіантів використань	Лит.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата				
Розробив		Філоненко В.П.				Аркуш	Аркушів	
Перевірив		Очеретяний О.К.						
Т. контр.					Вебзастосунок для автоматизованої системи табелювання співробітників	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01		
Н. контр.		Вітковська І.І.						
Затвердив		Жаріков Е.В.						

КПІ.ІТ-0320.045440.06.99.СБД



Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Філоненко В.П.		
Перевірив		Очеретяний О.К.		
Т. контр.				
Н. контр.		Вітковська І.І.		
Затвердив		Жаріков Е.В.		

КПІ.ІТ-0320.045440.06.99.СБД

Схема бази даних

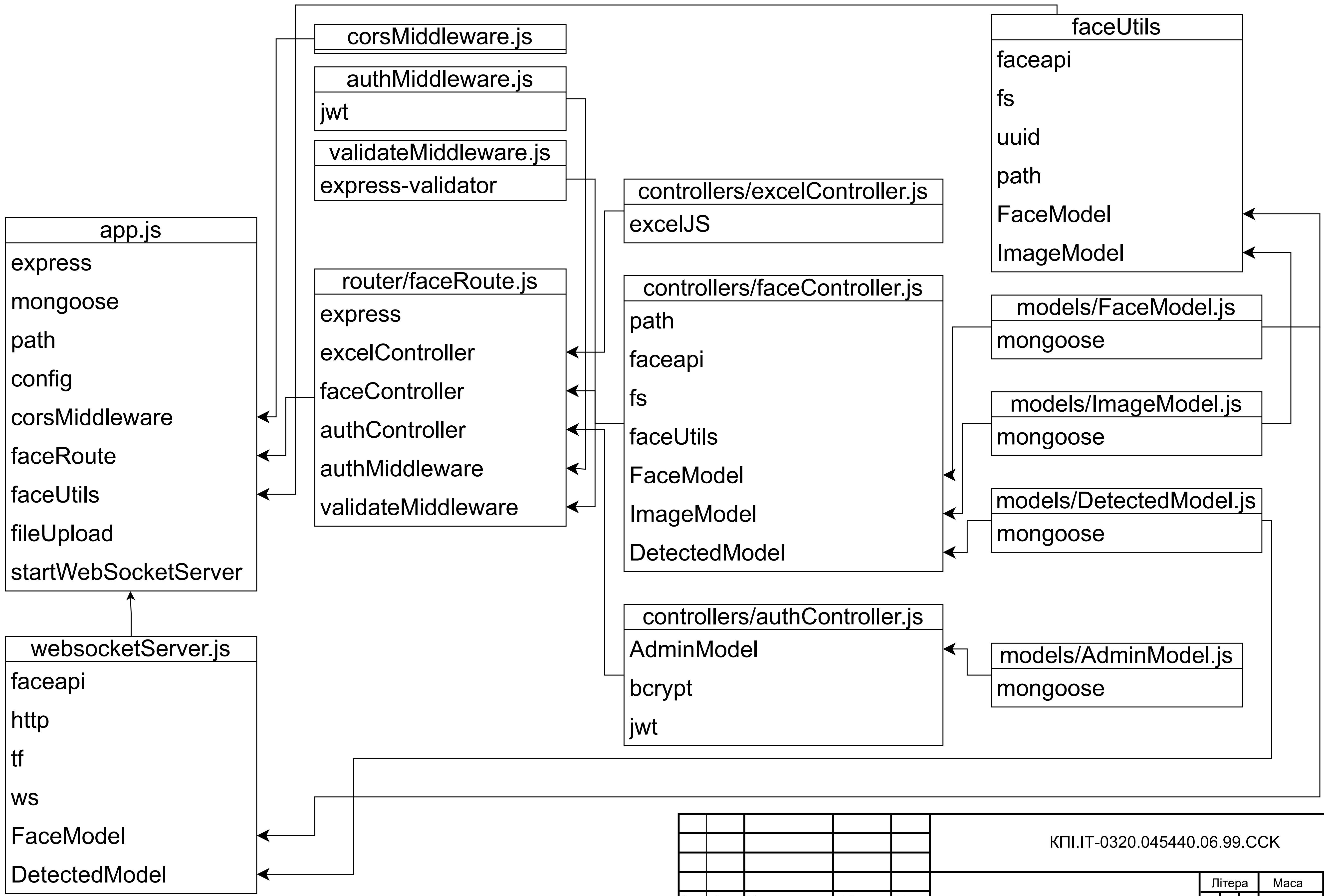
Літера	Маса	Масштаб

Аркуш	Аркушів

Вебзастосунок для автоматизованої системи табелювання співробітників

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-01

КПІ.ІТ-0320.045440.06.99.ССК



КПІ.ІТ-0320.045440.06.99.ССК

Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Філоненко В.П.		
Перевірив		Очеретяний О.К.		
Т. контр.				
Н. контр.		Вітковська І.І.		
Затвердив		Жаріков Е.В.		

Схема структурна класів програмного забезпечення

Літера	Маса	Масштаб

Аркуш	Аркушів

Вебзастосунок для автоматизованої системи табелювання співробітників

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-01