МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ГНАТІВ Святослав Васильович

Програмний модуль обліку робочого часу із застосуванням IoT технологій / Software module for time keeping based on IoT technologies

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-43
С. В. Гнатів

_____

Науковий керівник:
к.т.н., доцент О.Р. Осолінський

_____

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н.М. Васильків

Тернопіль – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ГНАТІВ Святослав Васильович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Програмний модуль обліку робочого часу із застосуванням IoT технологій / Software module for time keeping based on IoT technologies
керівник роботи О.Р. Осолінський, к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- проаналізувати сучасні підходи до обліку робочого часу в організаціях з використанням IoT-технологій, ідентифікації користувачів та засобів комп'ютерного зору;
- сформувати основні функціональні та нефункціональні вимоги системи обліку часу на основі мікроконтролерів, модулів RFID, розпізнавання обличчя та веб-інтерфейсу адміністрування;
- розробити архітектуру IoT-модуля, яка включає апаратну частину на базі Arduino, засоби збереження локальних подій, підключення до мережі та взаємодію із сервером через API;
- реалізувати серверну частину з використанням Express.js і PostgreSQL, забезпечивши збереження та обробку подій, реєстрацію фальсифікацій, адміністрування користувачів і генерацію звітів;
- створити клієнтський веб-інтерфейс адміністрування для керування системою, перегляду подій, налаштування прав доступу та формування звітів про присутність і відпрацьований час;
- провести тестування розробленого модуля на реальних даних, оцінити її точність, надійність та зручність використання.
5. Перелік графічного матеріалу в роботі:
- діаграма функцій бізнес-процесів;
- архітектура програмного модуля для обліку робочого часу
- архітектура веб інтерфейсу модуля;

- схема роботи модуля для зчитування RFID;
- діаграма станів для запису на SD картку.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент_____ С. В. Гнатів
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ О.Р. Осолінський
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль обліку робочого часу із застосуванням IoT технологій» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 86 сторінок і містить 35 ілюстрації, 9 таблиць, 15 додатків і 33 використаних джерела.

Метою кваліфікаційної роботи є розробка програмного модуля для обліку робочого часу працівників, що поєднує ідентифікацію за допомогою RFID-карток та верифікацію особи за зображенням обличчя з використанням методів комп'ютерного зору.

Об'єктом дослідження є процеси автоматизованого обліку робочого часу персоналу на основі IoT-рішень.

Предмет дослідження – апаратно-програмний модуль обліку робочого часу із застосуванням RFID-ідентифікації та технологій розпізнавання обличчя.

Розроблено архітектуру IoT-системи, яка включає два мікроконтролери (Arduino Uno та MEGA 2560), RFID-зчитувач, модуль реального часу, а також веб-інтерфейс адміністрування, реалізований з використанням Express.js і React. Система дозволяє автономно реєструвати події входу/виходу працівників, здійснювати перевірку особи за зображенням обличчя, вести облік подій і спроб фальсифікації, а також генерувати звіти про відпрацьований час.

Реалізовано фізичну та логічну моделі бази даних у СКБД PostgreSQL, програмну реалізацію серверної частини та API для взаємодії з веб інтерфейсом. Забезпечено збереження даних у локальному режимі у випадку відсутності зв'язку з сервером.

Ключові слова: IoT, ОБЛІК РОБОЧОГО ЧАСУ, RFID, РОЗПІЗНАВАННЯ ОБЛИЧЧЯ, ARDUINO, PostgreSQL, React, Express.js.

ANNOTATION

Qualification work on the topic «Software module for time keeping based on IoT technologies» for obtaining a bachelor's degree in the specialty 122 «Computer Science» of the educational program «Computer Science» is written on 86 pages and contains 35 figures, 9 tables, 15 annexes, and 33 references.

The purpose of the qualification work is to develop a software module for employee time tracking that combines RFID-based identification and facial verification using computer vision techniques.

The object of research is the processes of automated employee time keeping based on IoT solutions.

The subject of the study is a hardware-software time keeping module using RFID identification and facial recognition technologies.

An IoT system architecture has been developed, which includes two microcontrollers (Arduino Uno and MEGA 2560), an RFID reader, a real-time clock module, and an administrative web interface implemented using Express.js and React. The system allows autonomous registration of employee check-in/check-out events, identity verification through facial images, logging of events and fraud attempts, as well as the generation of work time reports.

The physical and logical models of the database were implemented using the PostgreSQL DBMS, along with the server-side software and API for interaction with the web interface. Local data storage is ensured in case of server connection loss.

Keywords: IoT, WORKING TIME TRACKING, RFID, FACE RECOGNITION, ARDUINO, PostgreSQL, React, Express.js.

ЗМІСТ

ВСТУП

В сучасному світі інформаційні системи та технології різного призначення займають велику частину сфер людської діяльності та отримують широке розповсюдження та динамічні темпи розвитку. Однією з таких технологій є Інтернет речей (IoT), який впливає на життя людей не тільки в побуті, а й у промисловому виробництві [1-3]. Адже сучасний світ потребує автоматизації процесів доступу та обробки інформації. Останній кейс говорить про промисловий Інтернет речей [4-6].

В промисловості IoT системи найчастіше використовуються для автоматизації буденних завдань, прикладом є облік робочого часу [7-9]. Адже традиційні методи реєстрації часу роботи, такі як журнали обліку або пропускні системи з магнітними картками, мають недоліки, такі як людський фактор, ризики підробки даних та необхідність ручного опрацювання інформації. Проте головним недоліком є невідповідність вимогам безпеки. Тому більшість підприємств активно впроваджують IoT системи для зменшення ризиків та помилок. Крім того використання IoT технологій у системах контролю робочого часу дозволяє підприємствам отримувати більш точні та надійні дані про присутність працівників, мінімізувати адміністративні витрати та підвищити рівень безпеки [10, 11].

Більшість із цих програм мають вбудовані функції для виявлення лазівок у відвідуваності співробітників. Це допомагає керівництву впровадити нову політику або змінити її відповідно до сучасних реалій та потреб підприємства. Одним з рішень яке найчастіше використовують підприємства для вирішення цієї проблеми є використання технології RFID (англ. Radio Frequency Identification), яка виконує швидку та безконтактну ідентифікацію за допомогою карт або брелків [12, 16]. Проте навіть RFID має вразливості пов'язані з несанкціонованим використанням карти іншою особою у разі її втрати або передачі її іншій людині [17].

Також варто сказати, що традиційні IoT системи, які виконують дане завдання є дорогими і важкими у процесі підтримки роботи та обслуговування

системи [19, 20]. Тому є гостра необхідність в розробці рішень, які простіші в процесі підтримки та роботи проте мають ту саму функціональність. Отже тема кваліфікаційної роботи є актуальною [21].

Метою роботи є розробка модуля обліку робочого часу працівників на базі Arduino з використанням RFID-технологій та технологій комп'ютерного зору для розпізнавання обличчя.

Щоб досягнути поставленої мети треба виконати наступні завдання:

- проаналізувати сучасні підходи до обліку робочого часу в організаціях;
- сформувати основні функціональні та нефункціональні вимоги системи обліку часу на основі мікроконтролерів, модулів RFID, розпізнавання обличчя та веб-інтерфейсу адміністрування;
- розробити архітектуру IoT-модуля, яка включає апаратну частину на базі Arduino, засоби збереження локальних подій, підключення до мережі та взаємодію із сервером через API;
- реалізувати серверну частину з використанням Express.js і PostgreSQL, забезпечивши збереження та обробку подій;
- створити клієнтський веб-інтерфейс для керування системою, перегляду подій;
- провести тестування розробленого модуля на реальних даних, оцінити точність, надійність та зручність використання.

Об'єктом дослідження виступатиме процес реалізації програмного модулю для обліку робочого часу співробітників на підприємстві.

Предметом дослідження є методи багаторівневої верифікації особи з використанням RFID та розпізнавання обличчя на основі комп'ютерного зору.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах студентської науково-практичної конференції "Інтелектуальні інформаційні технології в прикладних дослідженнях" (IITAR – 2025), м. Тернопіль, Україна, 27-29 травня 2025 р. (додаток С).

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сучасний світ активно розвивається і впроваджує нові технології у життя людей. Однією з таких технологій є Інтернет речей (IoT), який досить швидко розвивається та набуває популярності у світі[1-3]. Адже дана технологія дозволяє збирати та обмінюватися даними за допомогою мережі без будь якої взаємодії людини з комп'ютером.

Інтернет речей (IoT) - це набір фізичних пристроїв, транспортних засобів та інших фізичних об'єктів, які оснащенні мережевим з'єднанням та програмним забезпеченням, шо дозволяє проводити збір, обмін та обробку даних [3]. Зараз інтернет речей активно застосовується в медицині, побуті, транспорті та промисловості [4-6].

Якщо розглядати побутовий IoT то найчастіше його використовують для створення так званих розумних будинків. Які передбачають інтеграцію освітлення, опалення, системи безпеки та інших елементів, у будинку, в єдину мережу пристроїв, яка дозволяє виконувати віддалене керування та спостереження. Втім, це незначна частина можливостей, адже IoT технології інтегровані в життя людей набагато глибше.

У промисловості інтернет речей використовують для відстеження та моніторингу стану машин їх продуктивності та виявлення несправності[5]. Однак у промисловому IoT можна виділити один процес який найчастіше намагаються оптимізувати і це облік робочого часу, тобто процес реєстрації та моніторингу часу, який людина витрачає на виконання робочих завдань [7-9]. Адже оптимізувавши даний процес організація отримує можливість опрацьовувати дані про відпрацьовані години працівником, відгули, прогули, роботу понад норму у реальному часі з мінімальними затримками [10]. А отримані дані дозволяють виконати оцінку ефективності роботи організації, підрозділу організації або окремого працівника.

Отже створення рішення, яке допоможе ефективно та швидко та без помилок виконувати облік робочого часу, є актуальним завданням, що дозволить

краще дотримуватися трудового законодавства та зменшить вплив людини та кількість помилок при обліку робочого часу.

Однак перед тим як перейти до детального опису типів систем, прикладів використання, та їх зв'язком з IoT необхідно ознайомитися з поняттям обліку часу та системи обліку часу.

Облік робочого часу — це операція під час якої відбувається реєстрація та моніторингу часу, який працівники витрачають на виконання завдання[2]. Даний процес також включає не тільки робочий час а й і відсутність на робочому місці через відпустки, хвороби, відрядження та інші причини.

Система обліку робочого часу – це сукупність програмного забезпечення, яке автоматично фіксує, аналізує та звітує про час, витрачений працівниками н виконання конкретних завдань і проектів[5].

На даний момент існує наступна класифікація систем обліку робочого часу:

- ручний облік;
- системи з використанням біометричних даних;
- системи з використанням RFID-карт;
- цифровий облік.

Ручний облік - це традиційний метод обліку робочого часу, який вимагає багато часу та є трудомістким. Метод ручного обліку також створює серйозну небезпеку втрати даних. Адже даний метод передбачає ведення паперових журналів та проведення перекличок що суттєво збільшує ризик помилок та відволікає людей від роботи. Тому даний метод є недостатньо ефективним та застарілим хоча він був базовим багато років. Основною перевагою є конфіденційність адже метод не використовує жодних персональних даних. Основним недоліком є схильність до фальсифікації та ризик втрати даних.

Системи біометричної ідентифікації - це системи, які використовують спеціальні апаратні пристрої та алгоритми для розпізнавання, які розміщуються біля входу в офісні приміщення, виробничі зони, підприємстві. Дані системи працюють із використанням біометричних даних людини. Серед таких даних відбитки пальців, риси обличчя [9-15]. Даний тип систем виконує є швидку і точну ідентифікацію особи, що дозволяє здійснювати контроль доступу, а також

реєструвати час входу та виходу працівників [11]. Основною перевагою є висока точність та швидкість обробки даних. Основними недоліками є висока вартість впровадження і також постає питання конфіденційності адже система працює з персональними даними які потребують кращого захисту [21].

Системи з використанням RFID карт - це системи, які базуються на технології радіочастотної ідентифікації[12, 17, 18]. Дана технологія використовує передачу даних з карти за допомогою зчитувача за допомогою радіохвиль для визначення об'єкта. Процес ідентифікації займає декілька секунд. Основними перевагами даних систем є швидка інтеграція та інтуїтивно зрозуміле користування [17, 18]. Однак даний тип систем має декілька суттєвих недоліків. Першим вагомим недоліком є можливість передачі картки іншій особі, що допускає фальсифікацію даних. Другим недоліком є що працівники повинні зберігати RFID-карти і не губити їх.

Також варто зауважити що при порівнянні систем біометричної ідентифікації та RFID систем то біометричні сканують відбиток пальця або обличчя на відміну від набору даних записаних на мітку з якими працює RFID система. І тому можна зробити висновок, що біометричні системи надійніші за RFID проте вони значно дорожчі.

Цифрові системи - це системи, які включають використання хмарних та електронних рішень. Це системи які у більшості працюють через веб інтерфейси і мобільні пристрої та не потребують спеціальних датчиків та зчитувачів. Принцип роботи таких систем можна описати так що працівник реєструє свій робочий час, за допомогою реєстрації через мобільний додаток або браузер, і відмічає початок, кінець робочого дня, перерви тощо [4]. Дані одразу надсилається на хмарний сервіс, який проводить аналітику Однак у даному типі систем є які виконують облік робочого часу автоматично при ввімкненні комп'ютера і виявленні активності. Основною перевагами є точність та гнучкість і віддалений та швидкість обробки даних. Основними недоліками є питання конфіденційності адже існує ризик несанкціонованого доступу до веб сервісу, та навчання персоналу, існує ймовірність того що при впровадженні таких систем деякі працівники можуть допускати помилки.

Дана класифікація є узагальненою і демонструє ключові категорії систем. І саме така класифікація дозволяє показує як можна використати IoT для обліку робочого часу. Адже по факту системи з використанням RFID карток чи біометрії є набором апаратно-програмних комплексів, що зазвичай виступають складовою частиною IoT-систем виконують збір даних та передачу даних на локальну або хмарну обробку [7-9]. Цифрові системи також можна інтегрувати в IoT проте це не можна назвати повноцінною складовою частиною.

Дослідивши класифікацією, розглянемо найпоширеніший варіант IoT архітектури , яка продемонстрована на рисунку 1.1.

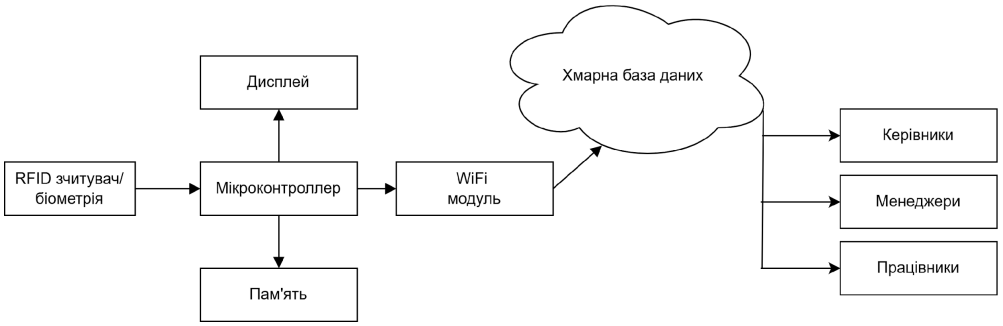

Рисунок 1.1 – Архітектура IoT системи для обліку робочого часу

На рисунку 1.1 представлено загальну архітектуру IoT-модулю робочого часу за допомогою RFID-технологій або біометрії. Дана архітектура передбачає в собі центральний мікроконтролер, зберігає інформацію в пам’яті, відображає її на екрані та передає через Wi-Fi модуль у хмарну базу даних, яка забезпечує централізоване зберігання подій і доступ до них для різних категорій користувачів Така архітектура дозволяє з отримати доступ до статистики та автоматизацію процесу обліку без потреби в ручному введенні даних.

Основні функції бізнес-процесу для модуля обліку рочого часу з використанням IoT-технологій:

- ідентифікація працівника;
- перевірка автентичності;
- обробка події;
- реєстрація події;
- реакції на порушення;

- адміністрування системи;
- формування звітності.

Діаграма функцій бізнес-процесів, де детальніше описані ці процеси зображена на рисунку 1.2.

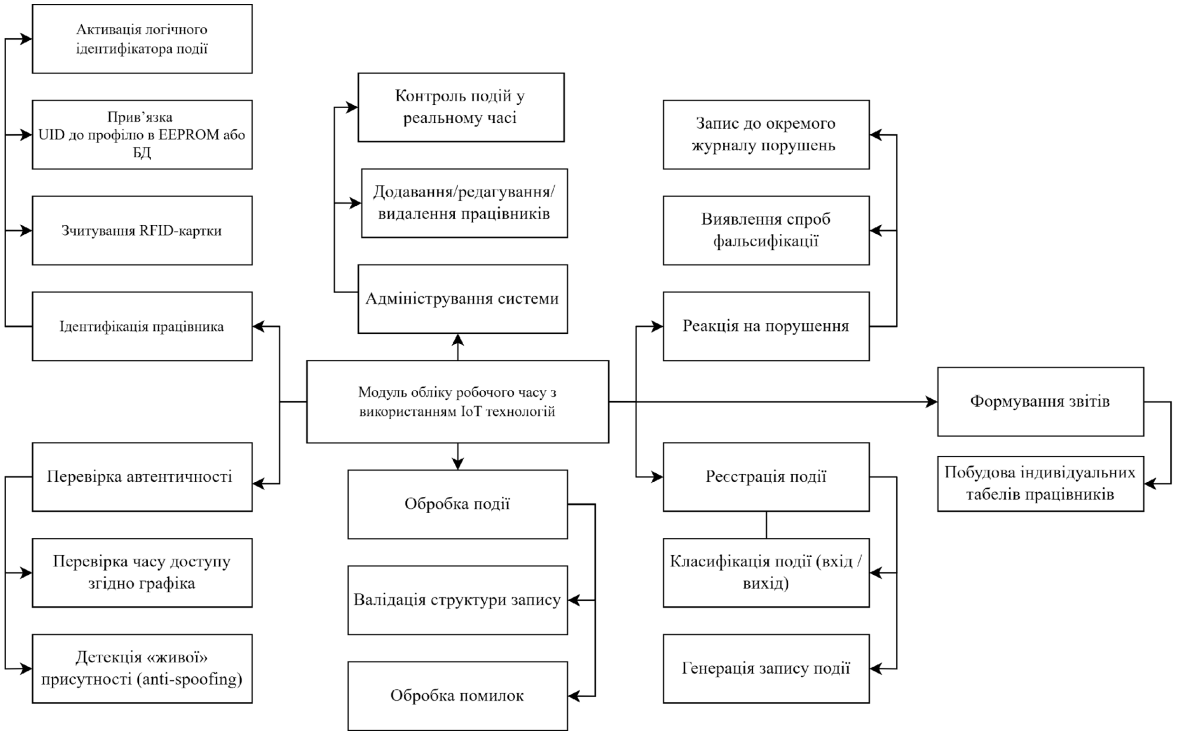

Рисунок 1.2 – Діаграма функцій бізнес-процесів

Також варто зазначити для функціонування та коректної роботи модулю є дві складові транзакційна та аналітична.Транзакційна складова виконує дії на рівні пристрою та забезпечує обмін даними. Аналітична складова включає обробку зібраних даних.

Отже, опис предметної області демонструє актуальність задачі обліку робочого часу на підприємствах різного масштабу з використанням IoT-рішень, зокрема модулів на базі мікроконтролерів з підтримкою RFID або біометричної ідентифікації, які зменшують вплив людського фактора та створюють передумови для підвищення ефективності управлінських рішень, оптимізації використання трудових ресурсів і дотримання трудового законодавства.

1.2 Огляд і аналіз існуючих рішень

На сьогоднішній день ринок пропонує багато комерційних рішень та досліджень які виконують контроль обліку робочого часу працівника на підприємстві. Відбір систем для аналізу проводився за такими критеріями:
тип ідентифікації(RFID, біометрія, біометрія+RFID);

- швидкість ідентифікації;
- місткість бази даних для зберігання даних працівників;
- інтеграція з іншими платформами або API;
- захист від фальсифікацій;
- підтримка мережевого підключення.

Варто зазначити, що обрані рішення не є повноцінними IoT-системами у класичному розумінні, оскільки зазвичай не реалізують повний цикл збору, передачі та обробки даних у розподіленому середовищі. Проте завдяки підтримці мережевих інтерфейсів, API та можливості віддаленого управління, ці рішення можуть бути інтегровані у більш складні IoT-архітектури та використовуватись як частина розумної інфраструктури підприємства. Як існуючі рішення було проаналізовано:

1. SpeedFace V5;
2. Suprema FaceStation F2;
3. ZKTeco uFace800.

SpeedFace V5 — це термінал від компанії ZKTeco, призначений для контролю доступу та обліку робочого часу [23]. Пристрій створений для використання в умовах підвищеної безпеки та інтенсивного навантаження, включаючи промислові підприємства, офісні будівлі, медичні установи та навчальні заклади.

Аналіз пристрою за відібраними вище критеріями:

− підтримуваний тип ідентифікації розпізнавання обличчя, долоні, відбитка пальця (опціонально), а також RFID-картки та QR-коди також є комбіновані сценарії доступу (біометрія + RFID або біометрія + PIN).

− швидкість ідентифікації для обличчя менше 0.3 секунди, долоні

близько 0.35 секунди верифікація відбитка пальця до 0.3 секунди

− місткість бази даних сягає до 10 000 шаблонів обличчя, 4 000 шаблонів відбитків пальців, 10 000 RFID-карток, 200 000 записів подій;

− інтеграція з підтримується ZKTeco SDK, ZKBioAccess, ZKBioTime, а також через Web API;

− захист від фальсифікацій відбувається за допомогою алгоритму Live Face Detection, запобігає ідентифікації по фото відео, інфрачервона камера забезпечує виявлення живої присутності;

− підтримка мережевого підключення за відбувається допомогою Ethernet (LAN), Wi-Fi (опційно), 3G модем (опційно), USB, Wiegand, RS485.

Комплектацію даного девайсу представлено на рисунку 1.3.

Рисунок 1.3 - Комплектація модуля ZKTeco SpeedFace-V5.

Suprema FaceStation F2 — це термінал для контролю доступу та обліку робочого часу, який поєднує високоточне розпізнавання обличчя з підтримкою RFID, відбитків пальців (у відповідних моделях) та мобільних ідентифікаторів [22]. Пристрій призначений для великих підприємств, корпоративного середовища, промислових об’єктів, державних установ і офісів, де важливі безконтактність, висока точність та масштабованість.

Аналіз пристрою за критеріями:

− підтримуваний тип ідентифікації: обличчя, RFID-картки , відбитки пальців (у версії з сенсором), мобільні ідентифікатори (через BLE/NFC), QR-коди;

підтримується багатофакторна аутентифікація (наприклад, обличчя + карта або обличчя + мобільний ключ);

- швидкість ідентифікації: менше 0.5 секунди;
- місткість бази даних до 100 000 користувачів (в режимі 1:1), до 50 000 (в режимі 1:N), до 5 000 000 логів подій, до 50 000 зображень;
- інтеграція з іншими підтримується через відкритий API до платформи BioStar 2 (RESTful API);
- доступні інтерфейси TCP/IP, RS-485, Wiegand, USB та підтримка інтеграції з відеосистемами через ONVIF;
- захист від фальсифікацій: реалізована технологія виявлення "живості" (Liveness Detection), ІЧ-камера й подвійна обробка забезпечують високий рівень надійності; для додаткового захисту біометричні шаблони шифруються у вбудованій пам’яті;
- підтримка мережевого підключення: підтримується через Ethernet (LAN), Wi-Fi (опційно), RS-485, Wiegand, USB-хост, мобільні сервіси через BLE/NFC, що забезпечує гнучкість у розгортанні як у локальних, так і в хмарних інфраструктурах.

Зовнішній вигляд даного девайсу представлено на рисунку 1.4.

Рисунок 1.4 – Вигляд Suprema FaceStation F2.

ZKTeco uFace800 - це біометричний термінал для контролю доступу та обліку робочого часу, який поєднує в собі передові технології розпізнавання

обличчя, відбитків пальців та RFID [24]. Пристрій призначений для середніх та великих підприємств, офісів, навчальних закладів і промислових об’єктів. Зовнішній вигляд даного девайсу представлено на рисунку 1.5.

Аналіз пристрою за відібраними вище критеріями:

- підтримуваний тип ідентифікації обличчя, відбиток пальця, долоня (у версії Plus) та RFID-картки, Підтримується як поодинока ідентифікація, так і комбінована (наприклад, обличчя + карта або відбиток + код;
- швидкість ідентифікації менше 1 секунди (в алгоритмах нового покоління та в останніх версіях) ;
- місткість бази даних сягає до 3 000 шаблонів обличчя, 4 000 шаблонів відбитків пальців, 10 000 RFID-карток 100 000 записів подій (у версії Plus – до 3,000 шаблонів долонь);
- інтеграція з підтримується для SDK ZKTeco, протокол ADMS, а також програмні платформи ZKBioTime, ZKAccess. Можлива інтеграція через Web API, TCP/IP, а також вбудований Web Server;
- захист від фальсифікацій відбувається за допомогою алгоритму Live Face Detection, що запобігає ідентифікації по фото чи відео, сканер відбитків пальців з оптичним сенсором, тобто розпізнає штучні копії;
- підтримка мережевого підключення за відбувається допомогою Ethernet (LAN), Wi-Fi (опційно), 3G модем (опційно), USB, Wiegand, RS485.

Рисунок 1.5 – Вигляд ZKTeco uFace800

Проведений аналіз комерційних рішень (ZKTeco SpeedFace V5, Suprema FaceStation F2, ZKTeco uFace800) дозволив виявити як сильні сторони, так і

обмеження сучасних систем обліку робочого часу. Незважаючи на широкий функціонал, більшість рішень демонструють високу залежність від хмарних сервісів і потребують постійного мережевого з'єднання. Частина пристроїв має складну процедуру синхронізації RFID-карток і обмежену гнучкість налаштувань. У низці випадків відсутнє локальне збереження подій, що знижує надійність при перебоях зв'язку.

Хоча всі аналізовані пристрої підтримують багатофакторну ідентифікацію та мають високий рівень захисту від фальсифікацій, їх впровадження є дорогим і технічно складним, особливо для середніх і малих підприємств.

1.3 Постановка задачі дослідження

Провівши аналіз предметної області визначено основні обмеження існуючих рішень, які суттєво впливають на процес автоматизації обліку робочого часу. Основні проблеми які характерні для описаних вище аналогів можна виділити:

- залежність від постійного підключення до мережі Інтернет та хмарного сервісу;
- недостатній захист від фальсифікацій (наприклад, передача картки іншій особі);
- висока вартість ліцензійного ПЗ та обмежена гнучкість у налаштуванні;
- відсутність локального збереження подій, що унеможливлює роботу при збої мережі;
- складна синхронізація RFID-карток.

Враховуючи описані недоліки сформовано функціональні вимоги до модуля і його роботи .

Вимоги щодо функцій модуля:

- зчитування даних з RFID-картки та перевірка існування даних карти у внутрішній пам'яті (EEPROM) контроллера MEGA 2560;
- відправка даних для запуску процедури розпізнавання обличчя;
- збереження фотографії з камери при підозрі на фальсифікацію;

- синхронізація даних карток із сервером за допомогою запиту, який надсилається на сервер з певною періодичністю;
- резервування подій у локальній пам’яті при відсутності з’єднання з сервером;
- можливість автономної роботи при збоях роботи;
- зручна веб-адміністрація працівників, логів та звітів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- сформувати основні функціональні та нефункціональні вимоги системи обліку часу на основі мікроконтролерів, модулів RFID, розпізнавання обличчя та веб-інтерфейсу адміністрування;
- розробити архітектуру IoT-модуля, яка включає апаратну частину на базі Arduino, засоби збереження локальних подій, підключення до мережі та взаємодію із сервером через API;
- реалізувати серверну частину з використанням Express.js і PostgreSQL, забезпечивши збереження та обробку подій;
- створити клієнтський веб-інтерфейс для керування системою, перегляду подій;
- провести тестування розробленого модуля на реальних даних, оцінити точність, надійність та зручність використання.

Отже у даній роботі буде реалізовано програмний модуль обліку робочого часу працівників на підприємстві з використанням IoT технологій, що дозволить:

- підвищити точність обліку робочого часу на підприємстві;
- підвищити безпеку на підприємстві;
- зменшити вплив людського фактору та кількість фальсифікацію;
- розробити гнучку та автономну архітектуру для легкої інтеграції у виробничі процеси та підсистеми підприємства;
- розробити просте рішення яке легко масштабувати та локально обробляти дані.

В результаті реалізації проектований модуль дозволить зменшити навантаження на адміністративний персонал та підвищити швидкість та точність обробки даних.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

2.1 Загальна структура проєктованого модуля

Модуль для обліку робочого часу з використанням IoT базуватиметься на архітектурі яка побудована на взаємодії трьох рівнів:

- апаратного рівня;
- програмного рівня ;
- аналітичного рівня.

Дана структура забезпечує розмежування функціональності модуля, обробки даних та аналізу. Реалізацію функціональної архітектури пропонованого рішення продемонстровано на рисунку 2.1.

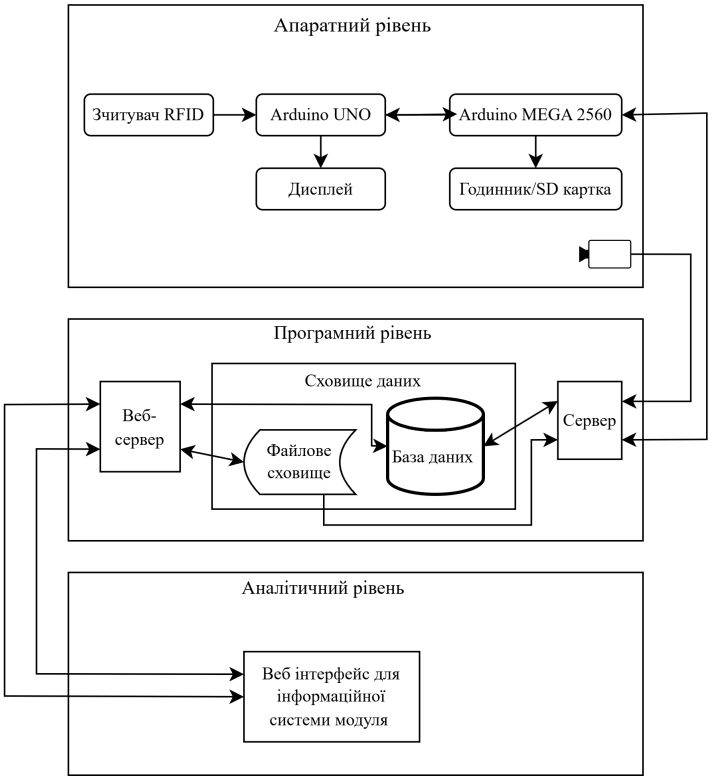

Рисунок 2.1 – Загальна архітектура модуля для обліку робочого часу;

Побудована архітектура використовує структурно-орієнтованого підходу, що має в основі ієрархічну декомпозицію компонентів модуля та відображення потоків даних між ними.

Основними функціями модуля є:

– зчитування даних працівника за допомогою RFID-картки;

- передача зчитаного коду працівника з Arduino UNO на Arduino MEGA;
- отримання поточного часу з модуля реального часу;
- виведення на LCD-дисплей імені працівника, його коду та часу події;
- перевірка коду на відповідність у вбудованій пам’яті або на сервері;
- запис події входу/виходу до SD-карти з фіксацією часу;
- отримання коду від апаратної частини через послідовний порт;
- перевірка наявності коду працівника у базі даних;
- запуск процесу розпізнавання обличчя через підключену камеру;
- порівняння отриманого зображення з еталонними фото працівників;
- запис результатів (успішний вхід або спроба фальсифікації) до бази даних;
- перегляд журналів подій: вхід, вихід, спроби фальсифікації;
- формування звітів за день, тиждень або місяць;
- завантаження та збереження фотографій працівників у фотоархів.

Інформаційні зв’язки у системі - Працівник → RFID-картка → Arduino UNO → MEGA (UART) → SD/RTC/EEPROM → MEGA → Сервер (UART) → PostgreSQL / Камера / Фотоархів.

Окремо на рисунку 2.2 зображено архітектуру роботи веб інтерфейсу .

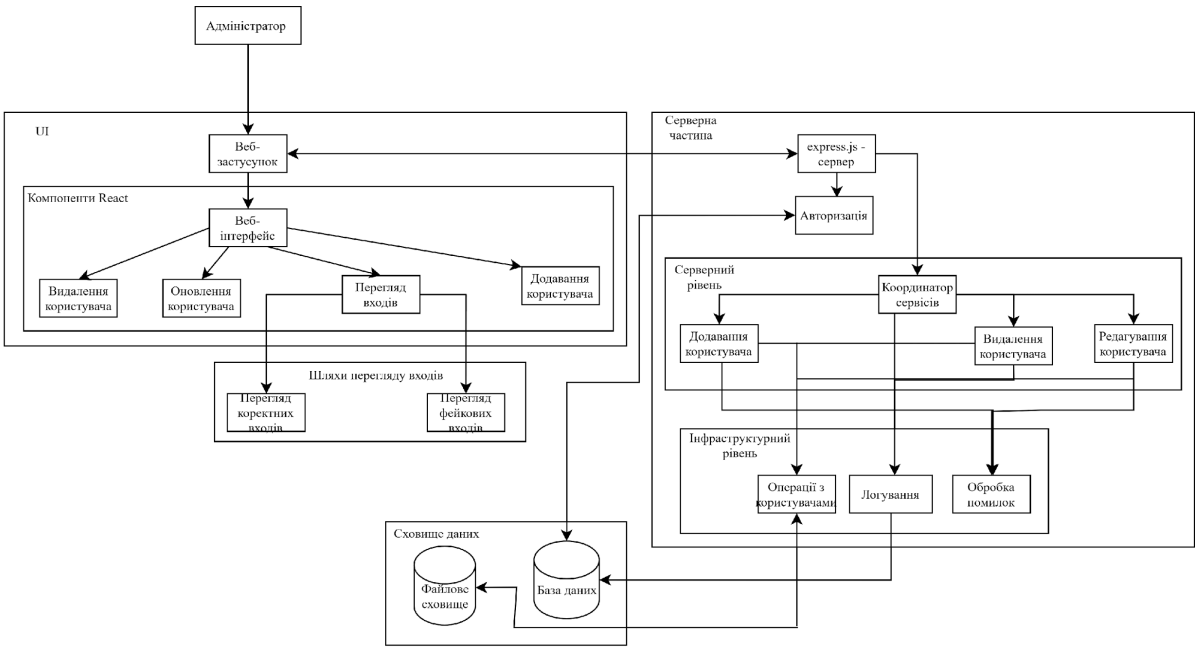

Рисунок 2.2 – Архітектура веб інтерфейсу модуля для обліку робочого часу

Рисунок 2.2 описує частину, яка дозволяє переглядати дані отримані із схеми на рисунку 2.1. Адміністратор взаємодіє з інтерфейсом, реалізованим на базі React.js, що має функціонал для додавання, оновлення та видалення користувачів, а також для перегляду подій входу виходу. Взаємодія між інтерфейсом і сервером відбувається за допомогою запитів, які передаються від компонентів React до відповідних маршрутів Express.js [27].

Стрілки демонструють потоки даних від адміністратора через веб-інтерфейс до серверної частини, а також взаємодію серверу з базою даних та файловим сховищем. Це забезпечує повний цикл адміністрування доступу, журналювання подій і виявлення порушень.

Отже, дані діаграми дають повне уявлення про архітектуру модуля для обліку робочого часу з використанням IoT-технологій, демонструючи її компоненти, їх взаємозв'язки та шляхи обробки даних.

2.2 Схема алгоритму роботи модуля для обліку робочого часу

Враховуючи архітектуру програмного модуля, реалізовано наступний варіант функціонування системи побудований на компонентному підході, який враховує особливості апаратних та програмних засобів. Алгоритми роботи кожного програмного модуля мають гарантувати високу надійність, забезпечувати гнучке масштабування та підтримку обробки даних від пристроїв Інтернету речей (IoT) у режимі реального часу.

Першою компонентою. описаного модуля є процес обробки RFID-картки після її прикладання до зчитувача. Після запуску системи пристрій переходить у режим очікування нової карти. У разі її виявлення виконується автентифікація сектору за допомогою попередньо визначеного ключа доступу. Цей етап є важливим для захисту інформації, оскільки дозволяє працювати лише з тими картками, що відповідають очікуваному формату доступу.

Після успішної автентифікації система переходить до зчитування даних із визначених блоків карти — зокрема, імені працівника (блок 4) та унікального коду

(блок 5). Зчитаний код передається на центральний контролер Arduino Mega за допомогою UART-зв’язку, де він проходить перевірку в базі працівників.

Наступним кроком є очікування відповіді від Mega. Вона може містити:

- часову мітку та напрямок руху у форматі <Entry 14:03>;
- повідомлення <NOCARD>, якщо UID відсутній у базі.

Залежно від типу отриманої відповіді пристрій виводить відповідне повідомлення на LCD-дисплей. У разі успішного доступу відображаються ім’я працівника, його код та час. Якщо ж картку не знайдено в базі, відображається повідомлення про відмову в доступі. Алгоритм роботи даного компонента зображено на рисунку 2.3

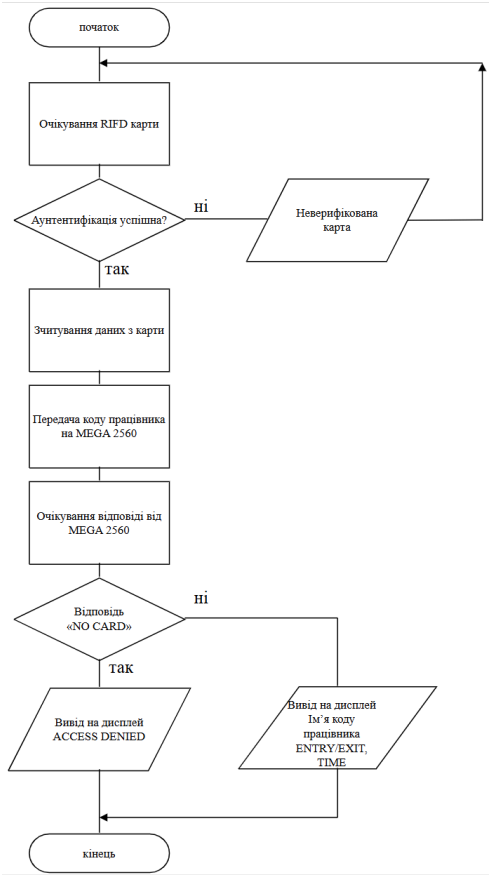

Рисунок 2.3 – Схема роботи модуля для зчитування RFID

Другою компонентою є модуль для обробки даних. Після зчитування та отримання даних ці дані аналізуються у локальній пам’яті даного модуля. Отже механізм збереження даних входу, який відповідає за локальне логування ідентифікованих входів і виходів працівників основною задачею має формування

структурованих записів (dbString), які включають дату, час, cardID працівника та напрям руху (вхід/вихід), і збереження цих подій у вигляді текстових файлів на SD-картці. Записи структуруються за окремими файлами відповідно до дати у форматі ДДММРР.txt.

Перед виконанням кожного запису модуль перевіряє готовність SD-карти та уникає дублювання вже записаних подій шляхом порівняння з останнім збереженим кодом та напрямком руху. Це дозволяє знизити надлишковість даних і оптимізувати використання пам’яті. Такий підхід забезпечує надійне та автономне збереження критичних подій у випадку тимчасової втрати зв’язку із сервером.

Таким чином, модуль SD-збереження виконує функцію локального журналу подій, що є важливою частиною безпечної та автономної IoT модуля обліку часу. Діаграма станів даного механізму зображено на рисунку 2.4.

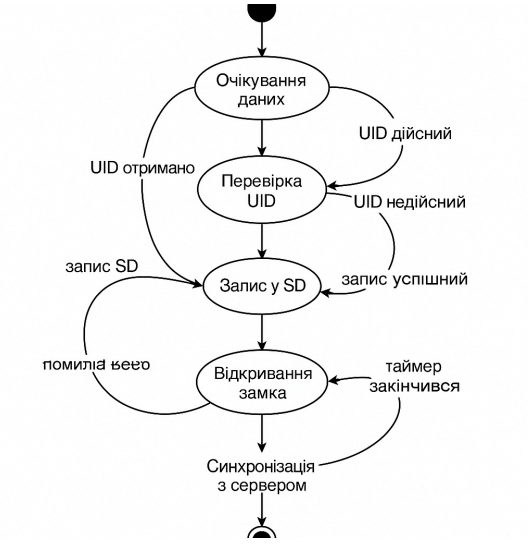

Рисунок 2.4 – Діаграма станів для запису на SD картку

Третьою компонентною частиною системи є модуль верифікації особи на основі детекції та розпізнавання обличчя, реалізований на стороні серверного програмного забезпечення. Основною функцією даного модуля є підтвердження ідентичності працівника після зчитування RFID-картки, шляхом порівняння зображення з камери з еталонним зображенням, що зберігається у базі даних.

Процес роботи модуля активується у відповідь на рядок dbString, отриманий через послідовний порт від модуля Arduino Mega обробки даних RFID-картки.

Зокрема, сервер отримує унікальний ідентифікатор картки (cardID) та ініціює сесію розпізнавання. Протягом обмеженого проміжку часу (5–10 секунд) з камери отримується поточне зображення обличчя користувача, після чого застосовується алгоритм розпізнавання face_recognition [25, 15], щоб зіставити обличчя з базою працівників PostgreSQL. У разі успішної ідентифікації обличчя виконує логування події у базі даних accessLogs з міткою часу. Якщо ідентифікація обличчя не пройшла успішно (відсутність збігу або низька якість зображення), дані події заносяться до журналу фальсифікацій (fakeLogs) разом із фотографією особи, що намагалась отримати доступ.

Таким чином, дана компонента виконує функцію багатофакторної автентифікації, яка значно підвищує рівень безпеки системи обліку робочого часу. Її інтеграція з сервером, базою даних і апаратною частиною забезпечує повну замкнуту архітектуру перевірки в реальному часі.

Четвертою, і фінальною, є компонентною частиною модуля обліку робочого часу є веб-інтерфейс керування та моніторингу, який надає користувачу можливість взаємодії з системою через браузер. В основі реалізації цієї компоненти лежить клієнт-серверна архітектура, де серверна частина створена на базі Express.js, а інтерфейс користувача реалізований з використанням бібліотеки React [27-30]. Після запуску сервер забезпечує роботу HTTP API, через які доступна уся необхідна інформація: список працівників, журнали входу та виходу, журнал фальсифікацій, відпрацьовані години, аналітика за періодами тощо. Уся інформація зчитується з бази даних PostgreSQL, де вона зберігається після обробки на рівні контролерів Arduino та модуля розпізнавання обличчя[27-29]. Інтерфейс на React автоматично підключається до API сервера та надає такі функціональні можливості:

- аналіз відпрацьованих годин за день, тиждень, місяць;
- перегляд логів подій доступу;
- управління записами (додавання, оновлення, видалення працівників);
- вивантаження звітів.

Користувачеві достатньо підключитись до локальної мережі підприємства та відкрити браузер за відповідною IP-адресою сервера. Авторизація у

веб-інтерфейсі забезпечується механізмом логіна й пароля з хешуванням через bcrypt, що гарантує безпечний доступ лише авторизованим особам.

Таким чином, дана компонента відіграє роль інструменту керування та аналітики, що забезпечує зручну візуалізацію стану системи, оперативний доступ до інформації та централізоване адміністрування IoT-системи обліку робочого часу.

2.3 Опис інформаційного забезпечення

2.3.1 Опис вхідної та вихідної інформації

Проєктований модуль обліку робочого часу функціонує на основі багаторівневої архітектури обробки інформації, яка охоплює етапи зчитування, виведення, перевірки, збереження та аналітичної обробки. На кожному з рівнів реалізовано обробку відповідних потоків даних, що забезпечує цілісність функціонального циклу системи.

До вхідної інформації, яка надходить до модуля, належать:

- ідентифікаційні дані працівника, зчитані з RFID-картки (прізвище, ім’я, унікальний код);
- зображення обличчя працівника, зафіксоване камерою для подальшої верифікації з базою даних;
- часові мітки, сформовані модулем реального часу (RTC) у момент прикладання картки;
- авторизаційні облікові дані користувача для доступу до веб-інтерфейсу адміністратора.

Вихідна інформація включає:

- сформований код події (dbString), що містить дату, час, cardID та напрям руху (вхід/вихід);
- повідомлення, що виводяться на дисплей (LCD) для інформування користувача;
- записані журнали доступу в базі даних, де зберігаються підтверджені події входу/виходу;

- лог подій фальсифікацій, який включає фотоосіб, що не пройшли верифікацію;
- дублікати подій у вигляді лог-файлів на SD-карті, що зберігаються на контролері Arduino Mega.

Окрім основних каналів введення/виведення, система оперує також проміжною інформацією, яка забезпечує коректну роботу модуля:

- статус синхронізації EEPROM із базою cardID;
- службові повідомлення про події (вхід, вихід, доступ заборонено);
- результати розпізнавання обличчя для підтвердження особи працівника.

2.3.2 Інфологічне проєктування інформаційної моделі

Проаналізувавши вхідну та вихідну інформацію системи для інфологічної моделі системи було виділено п’ять основних сутностей:

- Роль користувача (userRoles) - це таблиця, яка представляє перелік можливих ролей у системі з обмеженим набором допустимих значень з наступними сутностями, як: ID ролі (rolesID) (PK), назва ролі (roleName);
- Працівники (workers) - таблиця представляє основну інформацію про співробітників та має наступні сутності: ID працівника (workerID) (PK), прізвище працівника (workerSurname), ім’я працівника (workerName), ID картки працівника (cardID), ID ролі працівника (roleID) (FK), шлях до фотографії працівника (photoPath);
- Логіни користувачів (userLogins) - таблиця представляє облікові дані користувачів для доступу до внутрішнього сервісу моніторингу, та містить наступні сутності: ID логіну (loginID) (PK), ID працівника (workerID) (FK), логін працівника (login), хешований пароль (passwordHash);
- Журнал доступу (accessLogs) - таблиця представляє журнал подій, який зберігає успішно ідентифіковані спроби входу або виходу, та має наступні сутності: ID події (logID) (PK), ID працівника (workerID) (FK), дата та час (timestamp), напрям руху (direction);
- Журнал фальшивих спроб (fakeLogs) - таблиця представляє журнал

подій, який зберігає неуспішні спроби входу або виходу, та має наступні сутності: ID фейкової події (fakeLogID) (PK), ID картки працівника (cardID) (FK), ім’я працівника (workerName), дата та час (timestamp), шлях до фотографії працівника (photoPath).

2.3.3 Опис даталогічної моделі даних

На рисунку 2.5 зображено ER-діаграму для бази даних, яка відображає структуру збереження інформації для даного модуля

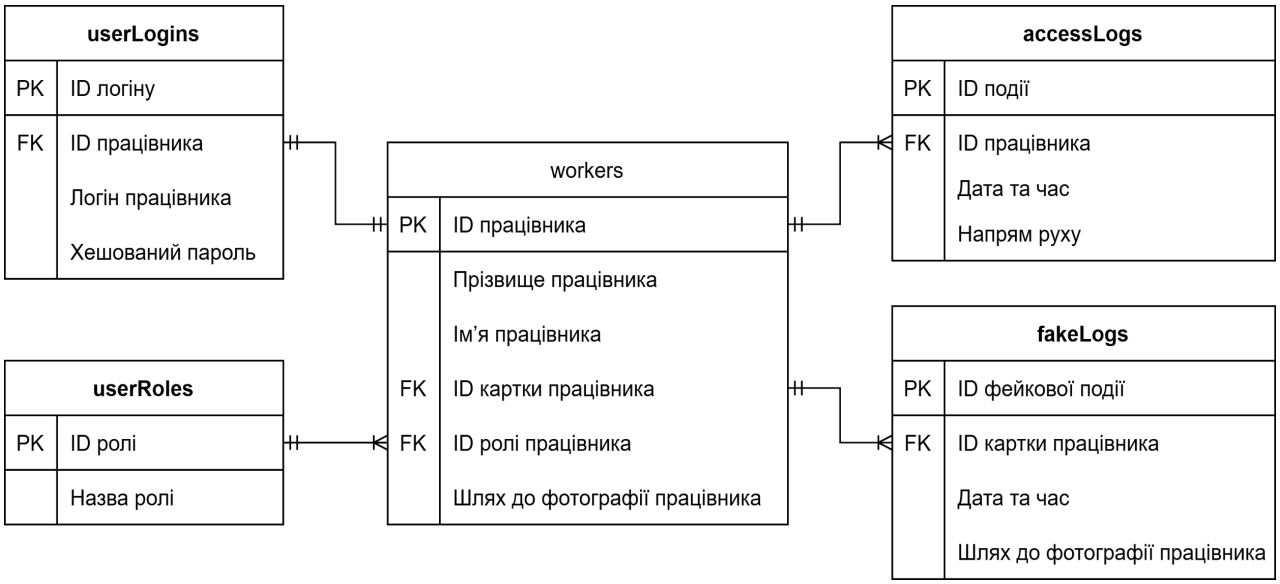

Рисунок 2.5 – ER-діаграма бази даних обліку робочого часу

У таблицях 2.1 - 2.5 які знаходяться нижче відображено сутності бази даних системи для обліку робочого часу.

Таблиця 2.1 – Сутність Роль користувача (userRoles)

Ключ	Сутність	Опис сутності
PK	ID ролі	Унікальний ідентифікатор ролі
	Назва ролі	Назва ролі (ролі: administrator, manager, user)

Таблиця 2.2 – Сутність Працівники (workers)

Ключ	Сутність	Опис сутності
PK	ID працівника	Унікальний ідентифікатор працівника
	Прізвище працівника	Прізвище працівника
	Ім’я працівника	Ім’я працівника
FK	ID картки працівника	Унікальний RFID-ідентифікатор картки
FK	ID ролі працівника	Посилання на таблицю ролей (userRoles)
	Шлях до фотографії працівника	Шлях до фотографії працівника

Таблиця 2.3 – Сутність Логіни користувачів (userLogins)

Ключ	Сутність	Опис сутності
PK	ID логіну	Унікальний ідентифікатор облікового запису
FK	ID працівника	Посилання на таблицю працівників (workers)
	Логін працівника	Логін користувача
	Хешований пароль	Хешований пароль

Таблиця 2.4 – Сутність Журнал доступу (accessLogs)

Ключ	Сутність	Опис сутності
PK	ID події	Унікальний ідентифікатор запису
FK	ID працівника	Посилання на таблицю працівників (workers)
	Дата та час	Час події
	Напрям руху	Напрям руху (0 – вихід, 1 – вхід)

Таблиця 2.5 – Сутність Журнал фальшивих спроб (fakeLogs)

Ключ	Сутність	Опис сутності
PK	ID фейкової події	Унікальний ідентифікатор фальшивої події
FK	ID картки працівника	UID-картки, що була використана
	Ім’я працівника	Ім’я працівника який зайшов
	Дата та час	Час виявлення спроби
	Шлях до фотографії працівника	Шлях до зображення особи під час спроби

Нижче наведено опис логічних зв’язків між таблицями у базі даних інформаційної системи:

- зв’язок між таблицями userRoles та workers – один тип ролі може мати багато працівників. Тому в даному випадку використовується зв’язок один до багатьох (1:N);
- зв’язок між таблицями workers та userLogins – один працівник має тільки один логін і навпаки. Це зв’язок один до одного (1:1);
- зв’язок між таблицями workers та accessLogs – один працівник може мати багато логів доступу. Тому в даному випадку використовується зв’язок один до багатьох (1:N);
- зв’язок між таблицями workers та fakeLogs – один працівник може мати мати багато фейкових логів доступу. Це зв’язок один до одного (1:N);

2.3.4 Проектування фізичної моделі бази даних

Відповідно до опису даталогічної моделі бази даних розроблено фізичну модель бази даних, яка реалізує збереження структурованої інформації про працівників, облікові записи, події доступу та верифікацію особи. Фізична модель визначає конкретні типи даних для кожного поля, первинні та зовнішні ключі, обмеження на значення та унікальність, а також логічні зв’язки між таблицями.

Для кожної сутності реалізовано таблицю, яка забезпечує структурну цілісність даних. Таблиці фізичної моделі даних показані у вигляді SQL-коду на

рисунках 2.6 - 2.10.

```
CREATE TABLE userRoles (
    roleID SERIAL PRIMARY KEY,
    roleName VARCHAR(50) NOT NULL UNIQUE
                         CHECK (roleName IN ('administrator','manager','user'))
);
```

Рисунок 2.6 – Фізична модель сутності userRoles

```
CREATE TABLE workers (
    workerID SERIAL PRIMARY KEY,
    workerSurname VARCHAR(255) NOT NULL,
    workerName VARCHAR(255) NOT NULL,
    cardID SMALLINT NOT NULL UNIQUE,
    roleID INT NOT NULL REFERENCES userRoles(roleID) ON DELETE SET NULL,
    photoPath TEXT NOT NULL

);
```

Рисунок 2.7 – Фізична модель сутності workers

```
CREATE TABLE userLogins (
    loginID SERIAL PRIMARY KEY,
    workerID INT NOT NULL REFERENCES workers(workerID) ON DELETE CASCADE,
    login VARCHAR(255) NOT NULL UNIQUE,
    passwordHash TEXT NOT NULL
);
```

Рисунок 2.8 – Фізична модель сутності userLogins

```
CREATE TABLE accessLogs(
    logID SERIAL PRIMARY KEY,
    workersID INT NOT NULL REFERENCES workers(workerID) ON DELETE CASCADE,
    timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    direction SMALLINT NOT NULL CHECK (direction IN (0, 1))
);
```

Рисунок 2.9 – Фізична модель сутності accessLogs

```
CREATE TABLE fakeLogs (
    fakeLogID SERIAL PRIMARY KEY,
    cardID SMALLINT(32) NOT NULL,
    workerName VARCHAR(255),
    timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    photoPath TEXT
);
```

Рисунок 2.10 – Фізична модель сутності fakeLogs

2.3.5 Програмна реалізація бази даних

На основі розробленої інфологічної та фізичної моделей реалізовано програмну імплементацію бази даних для системи обліку робочого часу з використанням СКБД PostgreSQL [28].

Для реалізації таблиць використано чистий SQL, що дозволило максимально точно задати структуру кожної сутності, описати ключі, обмеження, а також визначити типи даних відповідно до логіки функціонування інформаційної системи [28]. Кожна таблиця містить атрибути, які були визначені на етапах інфологічного проєктування та даталогічного моделювання, і відповідає структурі, поданій у фізичній моделі.

На рисунку 2.11 наведено SQL-код створення таблиці userRoles, яка зберігає перелік допустимих ролей у системі. У структурі таблиці передбачено поле roleID як первинний ключ, а поле roleName — як обов'язкове (NOT NULL), унікальне (UNIQUE) та обмежене допустимими значеннями за допомогою конструкції CHECK.

```
CREATE TABLE userRoles (
    roleID SERIAL PRIMARY KEY,
    roleName VARCHAR(50) NOT NULL UNIQUE
                         CHECK (roleName IN ('administrator','manager','user'))
);
```

Рисунок 2.11 – SQL-код створення таблиці userRoles

На рисунку 2.12 показано фрагмент коду створення таблиці workers, у якій зберігаються персональні дані працівників, їхні картки доступу, фото та ролі. Зовнішній ключ roleID забезпечує логічний зв'язок з таблицею userRoles.

```
CREATE TABLE workers (
    workerID SERIAL PRIMARY KEY,
    workerSurname VARCHAR(255) NOT NULL,
    workerName VARCHAR(255) NOT NULL,
    cardID SMALLINT NOT NULL UNIQUE,
    roleID INT NOT NULL REFERENCES userRoles(roleID) ON DELETE SET NULL,
    photoPath TEXT NOT NULL

);
```

Рисунок 2.12 – SQL-код створення таблиці workers

Таблиця accessLogs, представлена на рисунку 2.13, використовується для зберігання подій входу та виходу. Поле direction обмежене значеннями 0 (вихід) або 1 (вхід) за допомогою CHECK. Поле timestamp автоматично фіксує дату й час події за замовчуванням.

```
CREATE TABLE accessLogs(
    logID SERIAL PRIMARY KEY,
    workersID INT NOT NULL REFERENCES workers(workerID) ON DELETE CASCADE,
    timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    direction SMALLINT NOT NULL CHECK (direction IN (0, 1))
);
```

Рисунок 2.13 – SQL-код створення таблиці accessLogs

На рисунку 2.14 подано SQL-запит створення таблиці fakeLogs, яка зберігає інформацію про невдалі спроби ідентифікації. Це дає змогу вести облік фальсифікацій.

```
CREATE TABLE fakeLogs (
    fakeLogID SERIAL PRIMARY KEY,
    cardID SMALLINT(32) NOT NULL,
    workerName VARCHAR(255),
    timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    photoPath TEXT
);
```

Рисунок 2.14 – SQL-код створення таблиці fakeLogs

Виконана програмна реалізація бази даних дозволяє забезпечити збереження всіх важливих елементів інформаційної моделі системи: обліковихданих працівників, історії входів/виходів, журналів верифікації облич, а також доступу до цих даних у режимі реального часу через серверну частину. База даних побудована із врахуванням розширюваності та подальшої інтеграці з веб-інтерфейсом адміністратора й аналітичним модулем.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

3.1 Реалізація програмного забезпечення

На основі аналізу архітектури модуля обліку робочого часу, що описана у попередньому розділі, можемо провести налаштування параметрів для апаратного та програмного забезпечення та реалізувати модуль.

Першим етапом при реалізації модуля є побудова апаратної частини, яка складається з двох взаємодіючих мікроконтролерів — Arduino Uno та Arduino Mega 2560 [26]. Таке рішення дозволило розподілити обчислювальне навантаження та усунути апаратні конфлікти між пристроями, що працюють через SPI-протокол. Отже зберемо першу частину апаратного модуля, яка складається з Arduino UNO, RFID-зчитувача MFRC522 та LCD дисплея. На рисунку 3.1 показано виводи мікроконтроллера Arduino UNO.

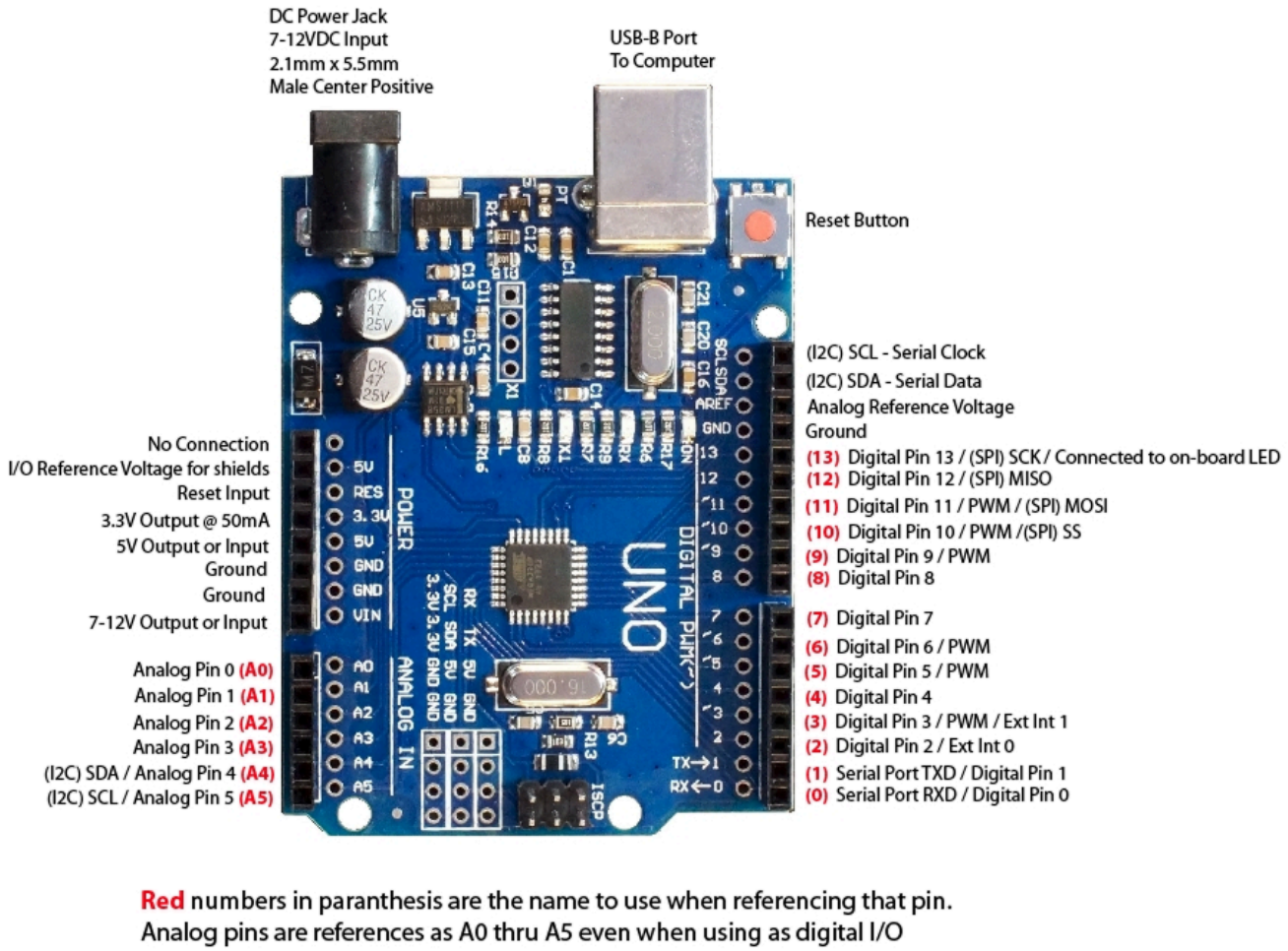

Рисунок 3.1 – Виводи Arduino UNO

Відповідно до пінауту представленого на рисунку 3.1 буде у таблиці 3.1 показано під'єднання до LCD дисплея за допомогою I2C та RFID-зчитувача MFRC522 у таблиці 3.2.

Таблиця 3.1 – Підключення виводів до LCD дисплея

Номер виводу UNO	Номер вивіду на LCD	Призначення
5 V	VCC	Живлення модуля
GND	GND	Мінус живлення
A4	SDA	Пін для даних I2C
A5	SDL	Пін для тактів I2C

Таблиця 3.2 – Підключення виводів до MFRC522

Номер виводу UNO	Номер вивіду на MFRC522	Призначення
3.3 V	VCC	Живлення модуля
GND	GND	Мінус живлення
Пін 9	RST	Скидання модуля
Пін 10	CS(SS)	Вибір підпорядкованого пристрою SPI
Пін 11	MOSI	Передача даних від Arduino
Пін 12	MISO	Прийом даних на Arduino
Пін 13	SCK	Серійний тактовий сигнал SPI

Друга частина апаратного модуля складається з наступних компонентів Arduino Mega, годинника реального часу DS3231 та microSD card зчитувача. На рисунку 3.2 показано виводи мікроконтроллера Arduino MEGA 2560.

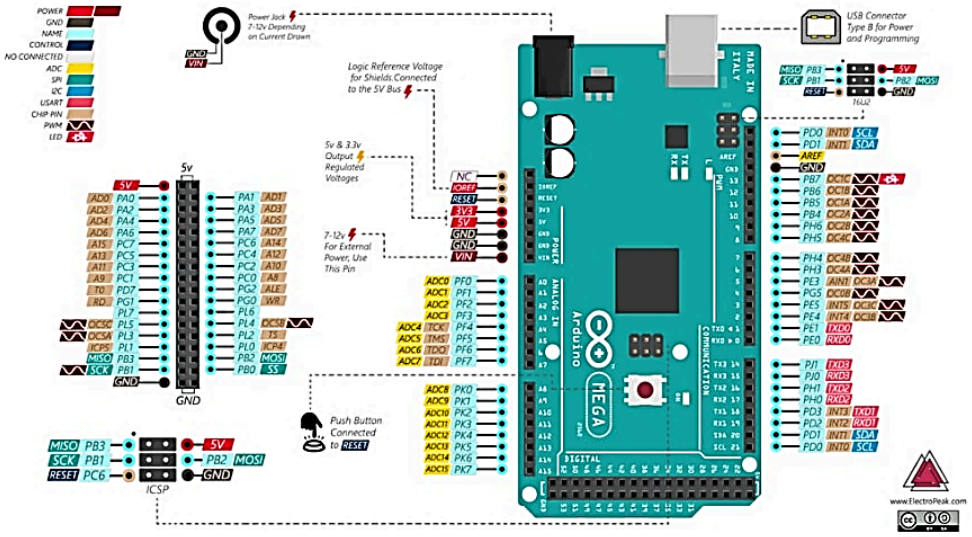

Рисунок 3.2 – Виводи Arduino MEGA 2560

Відповідно до пінауту представленого на рисунку 3.2 буде у таблиці 3.3 показано під'єднання до DS3231 та microSD card зчитувача у таблиці 3.4 .

Таблиця 3.3 – Підключення виводів до DS3231

Номер виводу MEGA 2560	Номер вивіду на DS3231	Призначення
3.3 V	VCC	Живлення модуля
GND	GND	Мінус живлення
Пін 20	SDA	Пін для даних I2C
Пін 21	SCL	Пін для тактів I2C

Таблиця 3.4 – Підключення виводів до microSD card зчитувача

Номер виводу MEGA 2560	Номер вивіду на microSD card зчитувача	Призначення
3.3 V	VCC	Живлення модуля
GND	GND	Мінус живлення
Пін 50	MISO	Прийом даних на Arduino
Пін 51	MOSI	Передача даних від Arduino до модуля
Пін 52	SCK	Серійний тактовий сигнал SPI
Пін 53	SS	OUTPUT

Комунікація між модулями відбувається за допомогою передачі повідомлень за допомогою асинхронного послідовного інтерфейсу UART через Serial1 (піни TX18 і RX19) на MEGA 2560 та Serial (піни RX2 і TX3) на UNO.

Основна логіка для першої частини апаратного модуля, реалізованого на базі Arduino UNO, полягає у зчитуванні інформації з RFID-картки та передачі унікального коду працівника на центральний контролер через UART-зв'язок. Код реалізовано у функціях loop, showOnLCD та timeOnMEGAClock, а обробка карти у відповідному фрагменті функції loop. Лістинг програмного забезпечення подано в додатку А.

Функція loop відповідає за повний цикл зчитування RFID-картки, передачу UID через UART і відображення результату на LCD. Вона виконує такі етапи:

- ініціалізація карти та автентифікація сектора за допомогою бібліотеки MFRC522 виконується перевірка наявності нової картки та автентифікація з використанням заданого ключа {0x47, 0x61, 0x6C, 0x50, 0x52, 0x41} (custom key для секторів 1 та 2);
- зчитування даних з блоків 4 та 5 де записані ім'я працівника (16 байт) та унікальний код (16 байт). Дані значення обробляються та конвертуються у текстові рядки nameEmp і codeEmp;
- передача коду працівника через UART на Arduino Mega через обгорнутий формат <code> та очікування відповіді від Mega;
- обробка відповіді та виведення на LCD за допомогою функції timeOnMEGAClock, і якщо дані коректні, на дисплеї виводяться ім'я, код працівника, напрям руху та час. У випадку невдалої перевірки, виводиться повідомлення ACCESS DENIED;
- функція showOnLCD форматує та виводить дані на екран, із динамічним розбиттям відповіді на напрям і час. LCD очищається, вмикається підсвітка, після чого дані виводяться у дві строки;

Таким чином, логіка модуля забезпечує надійне зчитування RFID, передачу даних, зворотну відповідь та інтерфейсне відображення результату.

Основна програмна логіка апаратного модуля на Arduino Mega реалізована у

функціях loop, saveToSD, requestCardIDSync, recvWithStartEndMarkers та sendTimeToUno, а також у допоміжних структурах обробки EEPROM, SD-карти та модуля реального часу (RTC DS3231). Лістинг коду наведено у додатку Б.

Центральна функція loop керує обробкою вхідного коду з UNO, перевіркою його в EEPROM, збереженням логів на SD-картку та керуванням замком. Основні етапи:

- синхронізація cardID із сервера;
- зчитування коду з UNO за допомогою recvWithStartEndMarkers() обробляються повідомлення від Arduino UNO у форматі <code>;
- перевірка дозволу на доступ, якщо cardID наявний у EEPROM, виконується зміна стану входу/виходу (чергування), запис події в SD-картку (saveToSD), і відправка часу та напрямку (Entry/Exit) назад на UNO;
- Якщо картку не знайдено повертається повідомлення <NOCARD> назад до UNO. Такі події не записуються у журнал.

Другим етапом є реалізація програмної частини сервера, який виконує розпізнавання обличчя. Даний етап реалізації також поділений на декілька частин.

Першою частиною є налаштування середовища, яке є початковим етапом запуску серверу, який забезпечує стабільну роботу всіх підмодулів.

На рисунку 3.3 зображено етапи які виконано при налаштуванні середовища.

Налаштування середовища	
1	Встановлення шляхів до директорій з фото працівників і логів
2	Імпорт та ініціалізація глобальних змінних
3	Налаштування з'єднання з UART

Рисунок 3.3 – Основні етапи при налаштуванні середовища

Ця частина створює основу для роботи всіх інших функціональних блоків від обробки UART-команд до логування подій. У разі порушення цього етапу — система не зможе коректно обробляти події доступу.

Другою частиною, яку виконано для реалізації сервера, є налаштування

комунікації з апаратною частиною. На рисунку 3.4 зображено етапи які виконано у ході виконання даної частини.

Налаштування комунікації з апаратною частиною	
1	Встановлення з'єднання з Arduino MEGA 2560 через configPort()
2	Початкова і періодична синхронізація cardID через UART за допомогою sendCardID() та activeCardID()
3	Віідновлення з'єднання при помилках за допомогою `reconnectPort()`

Рисунок 3.4 – Основні етапи при налаштуванні комунікації з апаратною частиною

Модуль зв’язку з Arduino виконує автоматичне підключення через функцію configPort(), періодичну синхронізацію списку cardID через sendCardID() (використовуючи дані з activeCardID()), а також забезпечує повторне з'єднання у разі втрати з’єднання (reconnectPort()). Передача даних здійснюється у форматі <ADD|cardID> з затримкою для уникнення втрат.

Третьою частиною, яку виконано для запуску серверу, є обробка подій доступу. На рисунку 3.5 зображено етапи які виконано у ході виконання даної частини.

Обробка подій доступу	
1	Зчитування UART-повідомлень у форматі `dbString=...`
2	Декодування отриманих даних за допомогою decodeDBString()
3	Перевірка на існування cardID в локальному словнику imageWorkers

Рисунок 3.5 – Основні етапи при обробці подій доступу

Четвертою, і фінальною, є частина верифікація особи та логування. На рисунку 3.6 зображено етапи які виконано у ході виконання даної частини. Лістинг коду наведено у додатку В.

Верифікація особи та логування	
1	Відкриття камери, захоплення зображення
2	Розпізнавання обличчя (порівняння з еталонним кодуванням)
3	Логування результатів у базу: `accessLogs` або `fakeLogs`
4	береження фото при фейкових спробах (`saveFrame`)

Рисунок 3.6 – Основні етапи при верифікації особи та логуванні

Третім і фінальним етапом реалізації модуля для обліку робочого часу з використанням IoT технологій є реалізація веб сервера, який застосовується для роботи з веб сторінкою, яка візуалізує дані.

Для реалізації цієї частини системи обліку робочого часу було побудовано масштабовану RESTful-архітектуру на базі Node.js з використанням фреймворку Express. Уся логіка поділена на модулі, що забезпечують чіткий розподіл відповідальностей, легкість обслуговування та зручність масштабування. Архітектура зображена на рисунку 3.7.

Рисунок 3.7 – Архітектура API-сервера модуля обліку робочого часу

Ініціалізація сервера полягає у створенні HTTP-додатку на основі Express, що забезпечує обробку запитів за допомогою ключових middleware. До середовища інтегруються засоби для аналізу вхідних JSON-запитів, обробки

cookie-файлів, керування крос-доменними запитами, а також прийому й збереження зображень працівників через Multer. Крім того, налаштовано доступ до директорії зі статичними файлами для відображення фото.

Налаштування середовища охоплює структурування файлової системи сервера де виділяється окрема директорія для фотографій працівників та тимчасових файлів завантаження. Завдяки використанню path і fs, при завантаженні нових зображень автоматично перевіряється наявність директорій, генеруються унікальні назви, і файли зберігаються в цільовому розміщенні. Це забезпечує стабільну роботу навіть при великій кількості одночасних завантажень.

База даних реалізована на основі PostgreSQL із використанням пулу з'єднань бібліотеки pg. Сервер взаємодіє з кількома таблицями: workers, userLogins, userRoles, accessLogs і fakeLogs. Всі запити реалізовано асинхронно з перевіркою унікальності критичних полів (наприклад, cardID та login), а також з підтримкою CRUD-операцій і агрегацій для обробки робочих годин та історії відвідувань.

Маршрутизація API структурована на логічні блоки відповідно до функціонального призначення: аутентифікація, керування працівниками, облік робочого часу та обробка фейкових логів. Кожен блок представлений відповідними REST-кінцевими точками, які отримують, створюють, оновлюють чи видаляють інформацію з бази.

Управління файлами виконується безпосередньо на файловій системі сервера. Усі фото, пов'язані з працівниками, зберігаються у централізованій папці. При оновленні працівника старе зображення видаляється, а нове перейменовується з урахуванням прізвища, імені та часової мітки. Це мінімізує ризик конфліктів файлів і забезпечує чисту файлову структуру без застарілих зображень.

Асинхронна обробка та масштабування забезпечується завдяки використанню конструкцій async/await та ефективного пулу з'єднань до бази даних. Всі запити до PostgreSQL виконуються з обробкою винятків, що дає стабільність та дозволяє серверу працювати навіть за умов високого навантаження [27-29].

3.2. Інтерфейс користувача

Для створення інтерфейсу користувача використано React.js, який дозволяє створювати інтерактивні фронтенд-інтерфейси за допомогою компонентної підходу. Даний підхід забезпечує можливість створення модульного і швидко оновлюваного UI залежно від етапу обробки[9]. Також дана бібліотека дозволяє зручно інтегровуватися з найпоширенішими типами API.

Інтерфейс Time According System побудований у темній кольоровій гамі з акцентами на сині та червоні кольори, які орієнтовані на корпоративне використання з зручним візуальним сприйняттям дій та фокусом на зручність.

На рисунку 3.8 зображно екран входу в систему Time According System, реалізований у мінімалістичному темному стилі з акцентом на елементи управління.

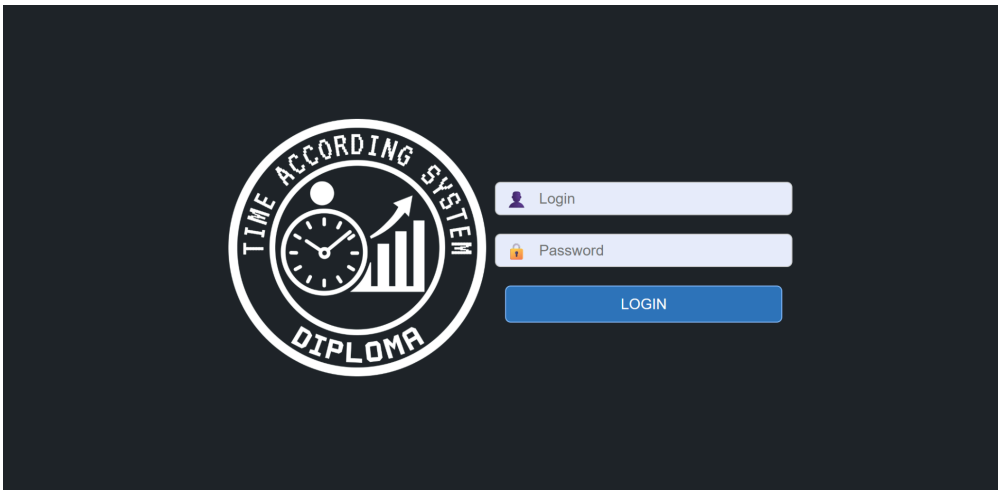

Рисунок 3.8 – Вигляд сторінки входу в обліковий запис

Інтерфейс даної сторінки ліворуч містить логотип , стилізований як кругла емблема з годинником, графіком та написом "Time According System – Diploma", що демонструє академічний характер розробки. Емблема виконана у білому кольорі, що надає їй виразності на темному тлі.

Праворуч від логотипа розташовано форму входу, що складається з:

– двох вхідних полів з іконками користувача та замка для введення

логіна й пароля;

– кнопку входу синього кольору з білим написом “LOGIN”;

Усі елементи мають заокруглені кути та світлу заливку з тінню, що створює об’ємний ефект на темному фоні. Візуальне розділення контенту, чітка типографіка та лаконічний функціонал інтерфейсу відповідають сучасним принципам UX-дизайну.

На рисунку 3.9 зображено інтерфейс сторінки адміністратора системи, який виконано з чітким поділом на функціональні блоки.

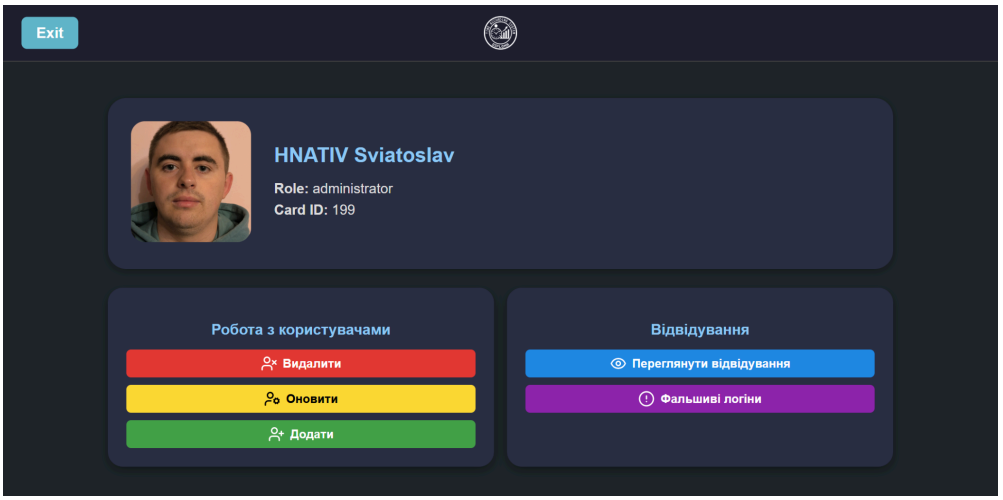

Рисунок 3.9 – Дизайн сторінки адміністратора

У верхній частині розташовано кнопку "Exit" світло-блакитного кольору, що виконує функцію виходу з системи, та логотип, який є візуальним маркером системи.

Центральна область екрану структурована на дві ключові частини. Перша частина — це профіль адміністратора, де відображено фотографію користувача, його ім’я, роль та card ID. Друга частина містить функціональні панелі для керування системою. Панель "Робота з користувачами" має три кнопки: видалення, оновленн і додавання нового користувача. Кожна кнопка має власний колір, білий шрифт і заокруглені краї для зручності взаємодії.

У секції "Відвідування" розміщено кнопку для перегляду журналу відвідування, та кнопку для перегляду фальшивих логінів, кожна з відповідною піктограмою.

На рисунку 3.10 зображено сторінку для видалення працівників перегляду списку працівників, оформлений у єдиному темному стилі веб застосунку.

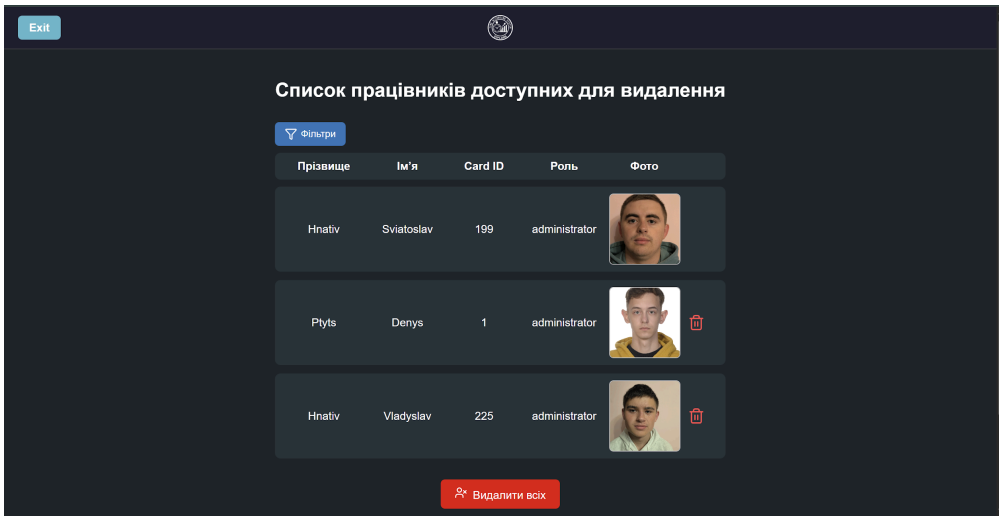

Рисунок 3.10 – Сторінка видалення працівників

Під шапкою яку описано вище наведено заголовок “Список працівників доступних для видалення”, набраний великим білим шрифтом для акцентування уваги на основному призначені сторінки. Знизу під заголовком розміщено кнопку “Фільтри”, оформлену в синьому кольорі, яка відкриває панель фільтрації за заданими параметрами.

Основна частина контенту представлена у вигляді таблиці працівників. Для кожного з них виводяться такі атрибути, як прізвище, ім’я, card ID, роль та фотографія. Кожен запис реалізовано окремим блоком з темним фоном, плавними заокругленнями та наступним розміщенням елементів. Фото знаходиться праворуч, а текстова інформація вирівняна ліворуч по центру блоку. Імена та ролі подано білим шрифтом. Кожен працівник має іконку видалення у вигляді піктограми кошика.

У нижній частині розміщена велика червона кнопка “Видалити всіх”, що візуально вирізняється серед інших елементів як критична дія, призначена для одночасного очищення списку.

На рисунку 3.11 представлено сторінку оновлення даних працівників.

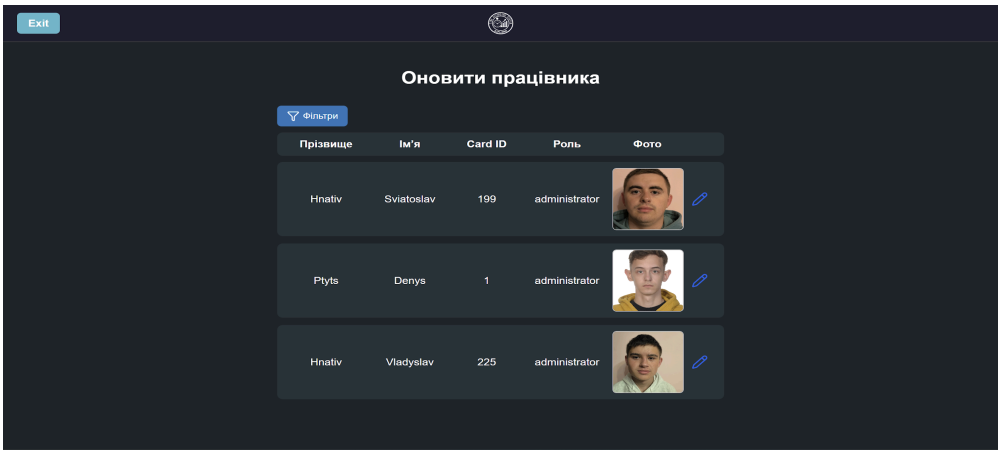

Рисунок 3.11 - Сторінка оновлення працівників

Сторінка організована за схожою структурою як і попередня та містить структуровану таблицю записів як і на сторінці видаленя.

У останній колонці передбачено іконку олівця, яка відкриває модальне вікно для редагування відповідного запису. Іконка має чітке контрастне обрамлення, що підвищує її помітність і спрощує взаємодію користувача з елементом.

На рисунку 3.12 представлено модальне вікно, яке дозволяє оновити інформацію про працівника.

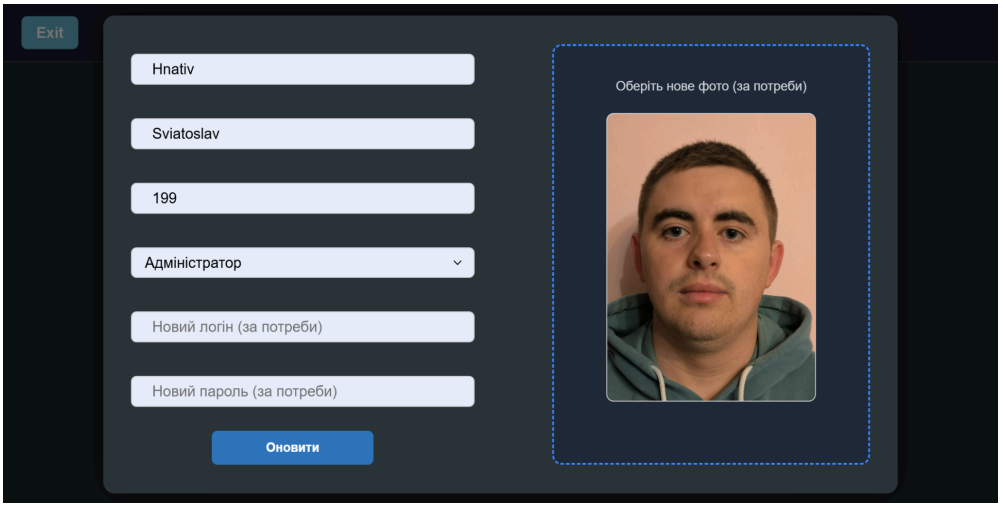

Рисунок 3.12 - Сторінка оновлення працівників

Сторінка розроблена з використанням темно-синього фону, тінню та заокругленими кутами. З допомогою підходу вікно виокремлюється з-поміж решти інтерфейсу й утримує увагу користувача на активному вмісті. Вміст

сторінки має чіткий візуальний поділ функціональних зон.

У лівій частині розміщено форму для оновлення даних працівника. Вона містить текстові поля для введення прізвища, імені, card ID, логіна та пароля, а також випадаючий список для вибору ролі (наприклад, “Адміністратор, Менеджер, Користувач”). Поля мають світлий фон, чіткі межі та достатній простір для комфортного введення інформації. Внизу форми розміщена синя кнопка “Оновити”, яка виконує функцію збереження змін.

Праву частину вікна відведено під область з допомогою якої відбуваться завантаження зображення. Зображення обрамлене пунктирною синьою рамкою. Рамка виконує роль маркера для завантаження або заміни фото. Надпис “Оберіть нове фото (за потреби)” підказує користувачеві можливість оновлення зображення. Натискання на область відкриває інтерфейс вибору файлу.

Загалом, модальне вікно забезпечує зрозумілу структуру, зручну навігацію та адаптацію до різних типів пристроїв, що робить процес редагування інтуїтивно доступним.

На рисунку 3.13 зображено інтерфейс сторінки додавання нового працівника, який виконаний з чітким поділом на зони введення даних і завантаження зображення. Поділ на зони аналогічний до модального вікна при оновленні працівника.

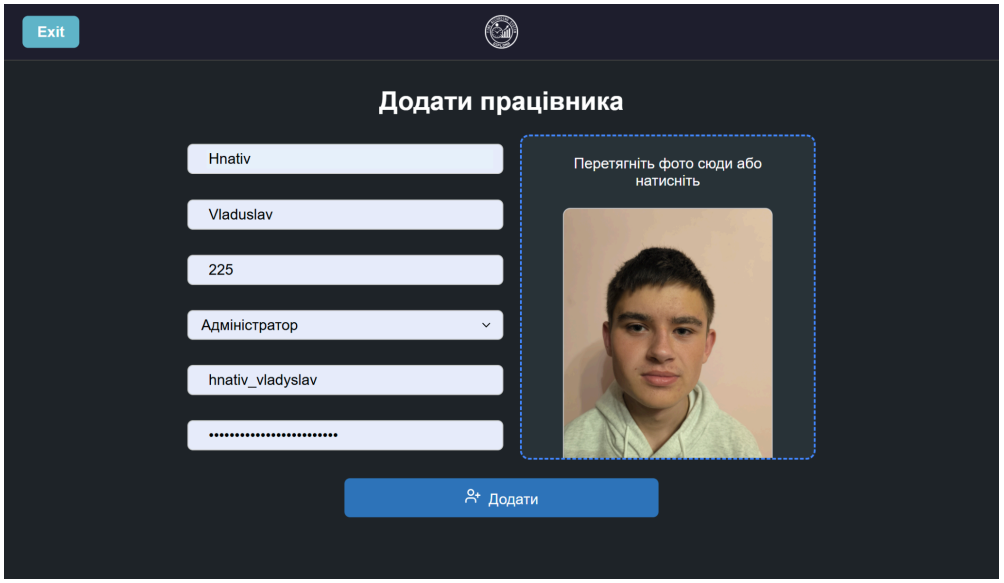

Рисунок 3.13 - Сторінка додавання працівників

На рисунку 3.14 представлено сторінку зведеної таблиці обліку відпрацьованого часу та нарахувань по всіх працівниках, що є адміністративною функцією модуля . Дизайн виконано у темній кольоровій гамі з чіткою сіткою даних, що забезпечує легке сприйняття інформації та підтримує загальний стиль.

Основний блок інтерфейсу представлено у вигляді таблиці, де кожен рядок відповідає одному працівникові. Таблиця містить колонки з аватаркою працівника, його прізвищем та ім’ям, card ID, кількістю відпрацьованих годин за поточний день, тиждень і місяць, а також розрахованою сумою оплати в гривнях. Завершує кожен рядок колонка дії, де є іконка перегляду у формі ока, яка відкриває деталізовану інформацію про працівника.

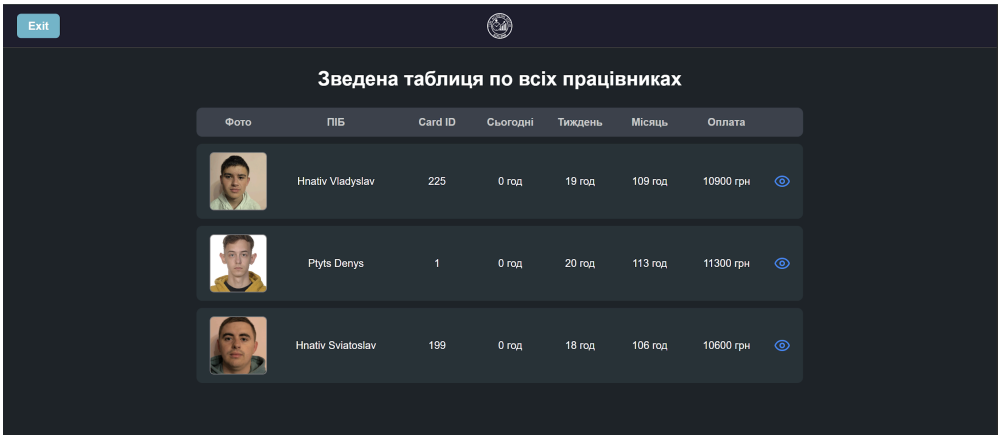

Рисунок 3.14 - Сторінка додавання працівників

На рисунку 3.15 представлено модальне вікно статистики по днях та місяцях для конкретного працівника, яке є частиною аналітичної функціональності.

У верхній частині вікна розміщено заголовок "Статистика по днях: [ім’я працівника]", вирівняний по центру і набраний білим шрифтом, згідно з стилем сторінки. Поруч із заголовком знаходиться чекбокс, що дозволяє обмежити перегляд графіка лише поточним календарним місяцем — це дає змогу швидко зосередитись на актуальній аналітиці.

Центральний блок містить лінійний графік, побудований у біло-синіх тонах, з маркерами для кожного значення. По горизонтальній осі відображено дати, по вертикальній — кількість відпрацьованих годин. Графік демонструє щоденну динаміку роботи обраного працівника, що дозволяє оцінити ритм відвідувань.

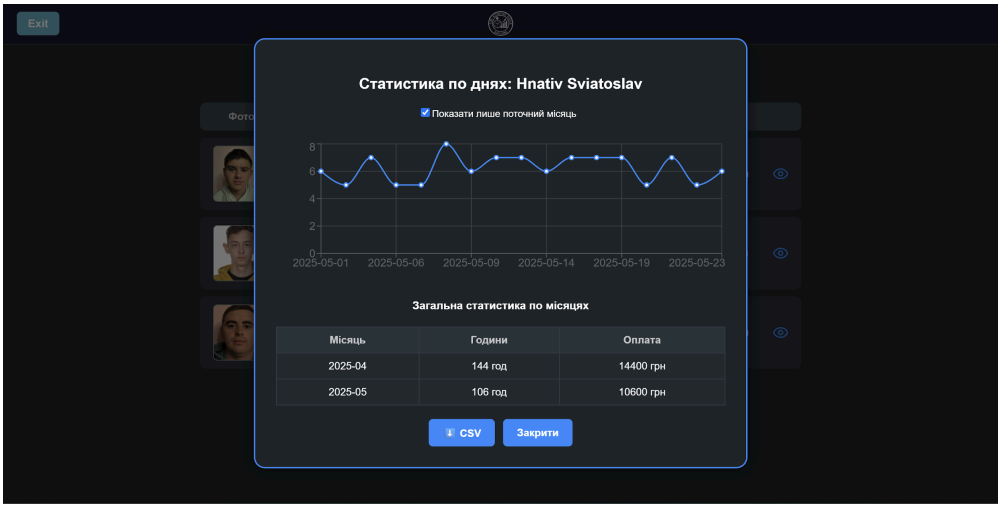

Рисунок 3.15 - Модальне вікно аналітики даних

У нижній частині вікна розміщено таблицю з узагальненою статистикою по місяцях. Вона містить три основні колонки — назви місяців, кількість відпрацьованих годин і відповідну суму оплати. Форматування таблиці чітке і впорядковане, що сприяє зручному перегляду зведених показників.

Футер модального вікна містить дві кнопки: "CSV", яка дозволяє експортувати статистику у форматі CSV-файлу, та "Закрити" , для завершення перегляду. Обидві кнопки мають гарну видимість і ергономічне розташування.

На рисунку 3.16 представлено інтерфейс сторінки фальшивих логінів, який є складовою безпеки. Ця сторінка призначена для відображення випадків фальсифікації входу та несанкціонованої аутентифікації користувачів, виявлених через механізм розпізнавання обличчя.

У верхній частині сторінки розміщено хеадер, який описаний вище, а трохи нижче — заголовок "Фальшиві логіни", набраний великим білим шрифтом, який чітко відображає призначення сторінки.

Основний зміст сторінки представлено у вигляді таблиці, яка містить ключову інформацію про кожен зафіксований інцидент. Для кожного запису наведено прізвище та ім’я працівника, унікальний card ID, дату виявлення невідповідності, точний час події, а також фотографію особи, зафіксованої камерою. Всі записи подані в окремих рядках з темним фоном і плавно заокругленими краями, що сприяє візуальній впорядкованості даних. Зображення

особи відображається праворуч у формі аватарки, що дає змогу швидко оцінити ситуацію та підтвердити невідповідність.

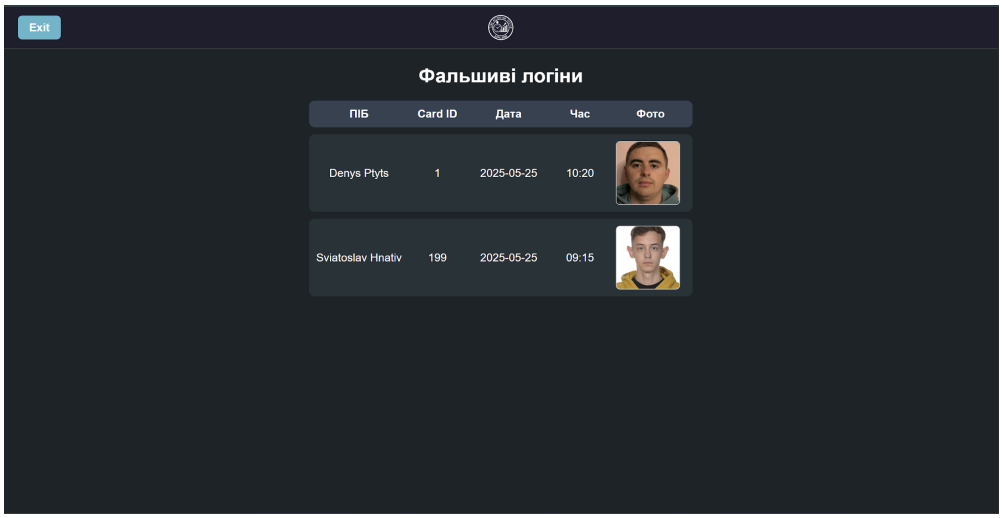

Рисунок 3.16 - Модальне вікно аналітики даних

Інтерфейс користувача веб застосунку, для модуля обліку робочого часу з використанням IoT технологій, демонструє високий рівень структурованості, функціональності та відповідності сучасним принципам UX/UI дизайну. Використання фреймворку React.js дозволило створити гнучкий і модульний інтерфейс із швидким реагуванням на дії користувача та зручною взаємодією з серверною частиною. Темна кольорова гама, візуальні акценти, інтуїтивно зрозуміла навігація та логічне групування елементів забезпечують комфортну роботу адміністратора із завданнями керування працівниками, аналітикою відпрацьованого часу та журналами подій.

Важливим є також вбудований функціонал роботи з фото та зведеною статистикою, що дозволяє ефективно виявляти спроби фальсифікацій і здійснювати повноцінне управління персоналом у режимі реального часу. Таким чином, реалізований інтерфейс користувача повністю відповідає завданням системи обліку робочого часу та сприяє підвищенню продуктивності адміністрування.

3.3 Тестування

Весь функціонал модуля для обліку робочого часу було перевірено на наступних сценаріях тестування: зчитування картки; фейковий доступ; розпізнавання обличчя; синхронізація cardID; запис у SD-карту та базу даних; обробка помилкових вхідних рядків; робота при втраті зв'язку з портом; перегляд статистики у веб-інтерфейсі; додавання, оновлення та видалення працівника.

Тестування за сценарієм «зчитування валідної картки»:

- мета – перевірити, чи система розпізнає картку з дозволеним cardID;
- вхідні дані – фізичне прикладання картки до RFID-зчитувача, яка присутня у EEPROM і БД;
- очікуваний результат – на LCD виводиться ім’я працівника та час, запис фіксується на SD-карті та у PostgreSQL;
- фактичний результат – дані відображено та збережено коректно;
- статус – пройдено.

Тестування за сценарієм «зчитування невалідної картки»:

- мета – перевірити обробку невідомої картки, яка відсутня у БД та EEPROM;
- вхідні дані – випадковий cardID;
- очікуваний результат – на LCD повідомлення "ACCESS DENIED CODE:", відсутній запис у логах;
- фактичний результат – система правильно відхилила доступ;
- статус – пройдено.

Тестування за сценарієм «успішне розпізнавання обличчя»:

- мета – перевірити верифікацію працівника після зчитування картки;
- вхідні дані – збережене обличчя працівника у БД, камера відкривається на 10 секунд;
- очікуваний результат – дані вносяться у таблицю accessLogs, обличчя підтверджене;
- фактичний результат – користувач успішно верифікований;
- статус – пройдено.

Тестування за сценарієм «фейковий вхід іншим обличчям»:

- мета – перевірити реакцію системи на неавтентичне обличчя;
- вхідні дані – чужа особа прикладає картку іншого працівника;
- очікуваний результат – розпізнавання не відбувається, фото зберігається у fakeLogs;
- фактичний результат – фейковий вхід зафіксовано, обличчя розпізнано як інше;
- статус – пройдено.

Тестування за сценарієм «запис до SD-карти»:

- мета – перевірити, чи дані доступу зберігаються локально;
- вхідні дані – вхід працівника з карткою та верифікацією;
- очікуваний результат – у файлі логів SD-карти створено новий рядок з UID, часом і напрямком;
- фактичний результат – запис збережено, доступний для зчитування;
- статус – пройдено.

Тестування за сценарієм «втрата з’єднання з сервером»:

- мета – перевірити надійність роботи у разі обриву з'єднання з Arduino;
- вхідні дані – емуляція розриву TX/RX кабелю або фізичне вимкнення порту;
- очікуваний результат – програма автоматично перепідключається, з’являється повідомлення;
- фактичний результат – з’єднання успішно відновлено після обриву;
- статус – пройдено.

Тестування за сценарієм «додавання нового працівника»:

- мета – перевірити створення запису через веб-інтерфейс;
- вхідні дані – ім’я, прізвище, cardID, логін, пароль, фото, роль;
- очікуваний результат – дані збережені у таблицях workers та userLogins, фото додано у папку;
- фактичний результат – новий працівник з’явився в системі, cardID передано на Mega;

– статус – пройдено.

Тестування за сценарієм «оновлення працівника»:

– мета – перевірити зміну ПІБ, картки, фото або ролі;

– вхідні дані – PUT-запит через веб-фронтенд з новими даними;

– очікуваний результат – БД оновлена, фото замінено, старе видалено;

– фактичний результат – оновлення відбулося коректно;

– статус – пройдено.

Тестування за сценарієм «видалення працівника»:

– мета – перевірити повне очищення даних працівника;

– вхідні дані – cardID, переданий у DELETE-запит;

– очікуваний результат – записи з таблиці видалені, фото стерте з диску;

– фактичний результат – працівника видалено, cardID більше неактивний;

– статус – пройдено.

Тестування за сценарієм «перегляд денних, місячних годин»:

– мета – перевірити функціональність зведення годин;

– вхідні дані – перехід на сторінку /workedhours-summary-all;

– очікуваний результат – відображаються фото, ПІБ, годинник та сума до оплати;

– фактичний результат – таблиця виводиться коректно, дані відповідають записам у accessLogs;

– статус – пройдено.

Тестування за сценарієм «перегляд фейкових логів»:

– мета – перевірити, чи коректно зберігаються та відображаються фейкові спроби доступу;

– вхідні дані – перехід на сторінку /fake-logs;

– очікуваний результат – список з cardID, ім’ям, датою, часом та фото порушника;

– фактичний результат – всі фейкові логи доступні, зображення коректно завантажуються;

– статус – пройдено.

Узагальнені результати тестування можна побачити у додатках Г–Р та таблиці 3.5.

Таблиця 3.5 – Узагальнені результати тестування

	Сценарій тестування	Фактичний результат	Статус
1	Зчитування валідної картки	Виведено ім’я, запис у SD та БД	Пройдено
2	Зчитування невалідної картки	Повідомлення "Немає такого ID", запис відсутній	Пройдено
3	Успішне розпізнавання обличчя	Обличчя розпізнано, запис у accessLogs	Пройдено
4	Фейковий вхід іншим обличчям	Збережено фото, запис у fakeLogs	Пройдено
5	Запис до SD-карти	Новий рядок із UID, часом і напрямком	Пройдено
6	Втрата з’єднання з портом	Автоматичне перепідключення успішне	Пройдено
7	Додавання працівника	Працівника створено, дані збережено у БД	Пройдено
8	Оновлення працівника	Дані оновлено, фото замінено, cardID змінено	Пройдено
9	Видалення працівника	Дані видалено з БД, фото стерте з файлової системи	Пройдено
10	Перегляд годин у зведенні	Показано коректні значення годин по днях, місяцях та оплату	Пройдено
11	Перегляд фейкових логів	Відображено фото, cardID, ПІБ, дата і час фіксації	Пройдено

Отже, тестування системи обліку робочого часу підтвердило її працездатність і відповідність функціональним вимогам. Успішно виконані сценарії зчитування карток, верифікації обличчя, фіксації подій у базі даних та на SD-карті, синхронізації cardID з сервером, обробки фейкових входів і взаємодії з веб-інтерфейсом. Система коректно обробляє вхід і вихід працівників, розпізнає

українські імена, забезпечує захист від підробок і дозволяє перегляд статистики у вигляді денних, тижневих та місячних годин з відповідною оплатою. Час реакції при зчитуванні й розпізнаванні обличчя становить до 10 секунд. Система готова до промислового використання для автоматизації обліку робочого часу на підприємствах.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було:

1. Проведено аналіз предметної області, сучасних технологій обліку часу, охоплено рішення з біометричною верифікацією, RFID-ідентифікацією та цифровими системами. Виділено основні переваги та недоліки кожного з підходів, що дозволило обґрунтувати доцільність поєднання кількох технологій у єдиній системі.

2. Сформульовано вимоги до модуля обліку часу на основі мікроконтролерів, модулів RFID, розпізнавання обличчя та веб-інтерфейсу адміністрування.

3. Розроблено архітектуру IoT-модуля, що складається з апаратної частини на базі Arduino UNO та MEGA 2560, RFID-зчитувача, модуля реального часу та SD-карти. Забезпечено передачу даних на сервер за допомогою послідовного інтерфейсу та реалізовано перевірку особи через систему розпізнавання обличчя.

4. Створено веб-інтерфейс з на базі React і Express.js, для управління працівниками перегляду логів і аналітики відпрацьованого часу.

5. Проведено тестування модуля, яке включало сценарії з валідними та невалідними картками, обробку фальсифікацій та відновлення після втрати з’єднання. За результатами тестування встановлено, що модуль є точним, надійним та зручним у використанні, а реалізовані функції відповідають заявленим вимогам. Результати апробації були представлені на науково-практичній конференції (додаток С).

Розроблений модуль успішно вирішує основні проблеми, характерні для традиційних систем обліку робочого часу, та може бути інтегрований у виробничі процеси підприємств для підвищення дисципліни, прозорості обліку та зниження впливу людського фактору.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Miorandi D., Sicari S., De Pellegrini F., Chlamtac I. Internet of Things: Vision, Applications and Research Challenges. Ad Hoc Networks. 2012. Vol. 10, No. 7. P. 1497–1516.

2. ISO/IEC 30141:2018. Internet of Things – Reference Architecture. Geneva: ISO, 2018. 38 p.

3. IBM. What is the Internet of Things (IoT)?. URL: https://www.ibm.com/think/topics/internet-of-things (дата звернення: 15.03.2025).

4. Jaloudi S. microIoTSCADA: A Tool to Monitor and Control Renewable Energy-Based Systems Using MODBUS. Proc. 4th Interdisciplinary Conf. on Electrics and Computer (INTCEC). Chicago, IL, USA, 2024.

5. Mourtzis D., Angelopoulos J., Panopoulos N. Design and development of an IoT enabled platform for remote monitoring and predictive maintenance of industrial equipment. Procedia Manufacturing. 2021. Vol. 54. P. 166–171.

6. Yang Y., Zhong M., Yao H., Yu F., Fu X., Postolach O. Internet of Things for smart ports: Technologies and challenges. IEEE Instrumentation & Measurement Magazine. 2018. Vol. 21, No. 1. P. 34–40.

7. Ishaq K., Bibi S. IoT Based Smart Attendance System Using RFID: A Systematic Literature Review. International Journal of Advanced Computer Science and Applications. 2023.

8. Makigod G.M. et al. IoT Based Automated Attendance System Using Facial Recognition. IJERT. 2024.

9. Patil B.H. et al. IoT Based Face Recognition System for Attendance Monitoring: A Review. Procedia Computer Science. 2023.

10. Touzene A., Wassim A., Larabi S. An Embedded Intelligent System for Attendance Monitoring. International Journal of Computer Applications. 2024.

11. Oluyemi T.T. et al. Development of an Attendance Management System Using Facial Recognition Technology. Journal of Emerging Technologies. 2024.

12. Al Imran M. et al. RFID-Based Smart Locks and Attendance Tracking in Academia. Sensors. 2024. Vol. 24, No. 3.

13. Nguyen-Tat B.T. et al. Automating Attendance using Computer Vision on Jetson Nano. arXiv preprint arXiv:2402.12345. 2024.

14. Qiang S. Facial and Behavioral Recognition for Smart Attendance. IEEE Access. 2024.

15. Zhao W., Li H. Face Verification in IoT-Driven Attendance Systems. IEEE Internet of Things Journal. 2022.

16. Wang X. et al. Face Recognition for Time Attendance Systems Based on Deep Learning. Journal of Information Security and Applications. 2021.

17. Mishra A., Verma A. IoT Based Time Attendance System Using RFID and Arduino. International Journal for Scientific Research & Development. 2020.

18. Dagar N., Sharma N. RFID Based Employee Attendance System. International Journal of Science and Research. 2021. Vol. 10.

19. Gaur A. et al. Smart Attendance System Using Face Recognition. Procedia Computer Science. 2020. Vol. 167. P. 1410–1417.

20. Khan M. et al. IoT and Time Attendance Systems: Trends and Challenges. IEEE Access. 2024. Vol. 12. P. 27456–27470.

21. Jiang F. et al. Privacy-Aware Access Control in Facial Recognition Systems. IEEE Transactions on Industrial Cyber-Physical Systems. 2023.

22. Suprema. Facestation F2 — мультифункціональний біометричний термінал. URL: https://supremainc.com.ua/suprema-facestation-f2-multymodalnyj-biometrychnyj-terminal (дата звернення: 15.03.2025).

23. ZKTeco Systems. SpeedFace-V5. URL: https://zkteco.systems/ua/product/speedface-v5 (дата звернення: 15.03.2025).

24. ZKTeco Systems. uFace800 facial multi-biometric time attendance and access control terminal. URL: https://zkteco.systems/en/product/uface800 (дата звернення: 15.03.2025).

25. Geitgey A. Face Recognition with Python. URL: https://github.com/ageitgey/face_recognition (дата звернення: 05.03.2025).

26. Banzi M., Shiloh M. Getting Started with Arduino. Sebastopol, CA: Maker Media, 2014. 262 p.

27. Express.js: Web Framework for Node.js. URL: https://expressjs.com/ (дата звернення: 05.03.2025).

28. PostgreSQL: Official Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 05.03.2025).

29. React – Official Documentation. URL: https://react.dev/ (дата звернення: 05.03.2025).

30. TechMagic. Why Use React for Web Development: 10 Reasons to Apply. URL: https://www.techmagic.co/blog/why-we-use-react-js-in-the-development (дата звернення: 15.03.2025).

31. Гнатів С., Осолінський О. Програмний модуль обліку робочого часу з використанням IoT технологій. *Інтелектуальні інформаційні технології в прикладних дослідженнях (IITAR–2025)*: збірник тез доповідей студентської науково-практичної конференції (Тернопіль, 27-29 травня 2025 р.), Тернопіль: ЗУНУ, 2025. С. 278–280.

32. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

33. В. М. Островерхов, Л. І. Біловус, К. З. Возьний, О. О. Луцишин, Г. Л. Монастирський, С. А. Надвиничний, С. В. Питель,С. К. Шандрук., Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

Додаток А

Програмна реалізація алгоритму для зчитування RFID картки

```
#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#define RST_MFRC522 9
#define SS_MFRC522 10
MFRC522 rfid(SS_MFRC522, RST_MFRC522);
MFRC522::MIFARE_Key key;
MFRC522::StatusCode status;
LiquidCrystal_I2C lcd(0x3F, 16, 2);  // LCD адрес
SoftwareSerial MEGASerial(2, 3);     // TX = 2, RX = 3
const byte numChars = 24;
char receivedChars[numChars];
String nameEmp = "";
String codeEmp = "";
String timeLCD = "";
void setup() {
  SPI.begin();
  Serial.begin(9600);
  MEGASerial.begin(9600);
  lcd.init();
  lcd.backlight();
  rfid.PCD_Init();
  rfid.PCD_SetAntennaGain(rfid.RxGain_max);
  rfid.PCD_AntennaOn();
  key.keyByte[0] = 0x47;
  key.keyByte[1] = 0x61;
  key.keyByte[2] = 0x6C;
  key.keyByte[3] = 0x50;
  key.keyByte[4] = 0x52;
  key.keyByte[5] = 0x41;
}
void loop() {
  if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
return;
  status = rfid.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 7,
&key, &(rfid.uid));
  if (status != MFRC522::STATUS_OK) return;
  uint8_t dataBlock[18];
  uint8_t codeBlock[18];
  uint8_t size = sizeof(dataBlock);
  if (rfid.MIFARE_Read(4, dataBlock, &size) == MFRC522::STATUS_OK) {
    nameEmp = "";
    for (int i = 0; i < 16; i++) {
      char c = (char)dataBlock[i];
      if (isPrintable(c)) nameEmp += c;
      else break;
    }
  }
  if (rfid.MIFARE_Read(5, codeBlock, &size) == MFRC522::STATUS_OK) {
    codeEmp = "";
    for (int i = 0; i < 16; i++) {
      char c = (char)codeBlock[i];
      if (isPrintable(c)) codeEmp += c;
      else break;
```

```
    }
    Serial.print(codeEmp);
    MEGASerial.print('<');
    MEGASerial.print(codeEmp);
    MEGASerial.print('>');
  }
  timeLCD = "";
  unsigned long startWaiting = millis();
  while (timeLCD == "" && millis() - startWaiting < 2000) {
    timeOnMEGAClock();
  }
  if (timeLCD == "NOCARD") {
    lcd.clear();
    lcd.backlight();
    lcd.setCursor(0, 0);
    lcd.print(" ACCESS DaNIED");
    lcd.setCursor(0, 1);
    lcd.print("CODE: " + codeEmp);
    delay(3000);
    lcd.noBacklight();
  } else if (timeLCD != "") {
    showOnLCD();
  }
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();
  delay(2000);
}
void showOnLCD() {
  lcd.clear();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print(nameEmp.substring(0, 16));
  int spaceIndex = timeLCD.indexOf(' ');
  String direction = "";
  String timePart = "";
  if (spaceIndex > 0) {
    direction = timeLCD.substring(0, spaceIndex);
    timePart = timeLCD.substring(spaceIndex + 1);
  } else {
    direction = timeLCD;
  }
  lcd.setCursor(0, 1);
  String line2 = codeEmp + "  " + direction + " " + timePart;
  lcd.print(line2.substring(0, 16));
  Serial.print("Code emp: ");
  Serial.println(codeEmp);
  Serial.print("LCD Time: ");
  Serial.println(timeLCD);
  delay(1500);
  lcd.noBacklight();
}
void timeOnMEGAClock() {
  static boolean recvInProgress = false;
  static byte ndx = 0;
  char startMarker = '<';
  char endMarker = '>';
  char rc;
  while (MEGASerial.available() > 0) {
    rc = MEGASerial.read();
    if (recvInProgress) {
```

```
      if (rc != endMarker) {
        receivedChars[ndx++] = rc;
        if (ndx >= numChars) ndx = numChars - 1;
      } else {
        receivedChars[ndx] = '\0';
        timeLCD = String(receivedChars);
        recvInProgress = false;
        ndx = 0;
      }
    } else if (rc == startMarker) {
      recvInProgress = true;
    }
  }
}
```

Додаток Б

Код модуля для Arduino MEGA 2560

```
#include <Wire.h>
#include <EEPROM.h>
#include <SPI.h>
#include <SD.h>
#include <DS3231.h>
#include <LiquidCrystal_I2C.h>
DS3231 myRTC;
LiquidCrystal_I2C lcd(0x3F, 16, 2);
const int sdcard_CS_pin = 53;
const int LOCK_PIN = 3;
#define TOGGLE_BASE_ADDR 500
#define TOGGLE_SLOTS 500
const int lengthOfCode = 4;
const byte numChars = 32;
const uint32_t locktime = 5000;
File DataBase;
bool sdCardError = true;
bool OPEN_DOOR = false;
bool old_in1out0 = false;
bool newData1 = false;
unsigned long curtime = 0;
unsigned long endtime = 0;
unsigned long lastScanTime = 0;
unsigned long lastSyncDataTime = 0;
String codeString = "";
String oldCodeString = "";
String dbString;
String nameOfFile;
String LCD_time = "";
char receivedChars[numChars];
String syncBuffer = "";
bool tempPresence[TOGGLE_BASE_ADDR];
bool collectingSync = false;
bool syncResponseReceived = false;
bool syncWarningShown = false;
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  Wire.begin();
  lcd.init();
  lcd.backlight();
  pinMode(LOCK_PIN, OUTPUT);
  digitalWrite(LOCK_PIN, LOW);
  checkSDCard();
}
void checkSDCard() {
  if (!SD.begin(sdcard_CS_pin)) {
    sdCardError = true;
    Serial.println("SD card error");
  } else {
    sdCardError = false;
    Serial.println("SD_Card is OK!");
  }
}
String strOut2(String InStr) {
```

```
  return (InStr.length() == 1) ? "0" + InStr : InStr;
}
bool isValidCardID(int cardID) {
  if (cardID <= 0 || cardID >= TOGGLE_BASE_ADDR) return false;
  return EEPROM.read(cardID) == 1;
}
void sendTimeToUno(bool isEntry) {
  bool h12Flag, pmFlag;
  String hh = strOut2(String(myRTC.getHour(h12Flag, pmFlag)));
  String mm = strOut2(String(myRTC.getMinute()));
  String direction = isEntry ? "Entry" : "Exit";
  String timeString = "<" + direction + " " + hh + ":" + mm + ">";
  Serial1.println(timeString);
  Serial.print("Send UNO time and direction: ");
  Serial.println(timeString);
}
void recvWithStartEndMarkers() {
  static boolean recvInProgress1 = false;
  static byte ndx1 = 0;
  char startMarker1 = '<';
  char endMarker1 = '>';
  char serialReadChar;
  while (Serial1.available() > 0) {
    serialReadChar = Serial1.read();
    if (recvInProgress1) {
      if (serialReadChar != endMarker1) {
        receivedChars[ndx1++] = serialReadChar;
        if (ndx1 >= numChars) ndx1 = numChars - 1;
      } else {
        receivedChars[ndx1] = '\0';
        recvInProgress1 = false;
        ndx1 = 0;
        newData1 = true;
        Serial.print("Data with UNO: ");
        Serial.println(receivedChars);
      }
    } else if (serialReadChar == startMarker1) {
      recvInProgress1 = true;
    }
  }
}
void saveToSD(bool in1out0) {
  if (sdCardError) {
    checkSDCard();
    Serial.println("Skipping write");
    return;
  }
  LCD_time = "";
  bool century = false;
  bool h12Flag, pmFlag;
  dbString = "";
  String day = strOut2(String(myRTC.getDate()));
  String month = strOut2(String(myRTC.getMonth(century)));
  String year = strOut2(String(myRTC.getYear() % 100));
  nameOfFile = day + month + year + ".txt";
  dbString += day + month + year;
  Serial.print("File to write: ");
  Serial.println(nameOfFile);
  String hour = strOut2(String(myRTC.getHour(h12Flag, pmFlag)));
  String minute = strOut2(String(myRTC.getMinute()));
```

```
  LCD_time = hour + ":" + minute;
  dbString += hour + minute;
  int codeLeng = codeString.length();
  if (codeLeng > lengthOfCode)
    codeString = codeString.substring(codeLeng - lengthOfCode);
  else if (codeLeng < lengthOfCode) {
    String buf = "";
    for (int i = 0; i < (lengthOfCode - codeString.length());i++)buf+=
'0';
    codeString = buf + codeString;
  }
  dbString += codeString;
  dbString += in1out0 ? "1" : "0";
  Serial.print("dbString = ");
  Serial.println(dbString);
  if ((codeString == oldCodeString) && (old_in1out0 == in1out0)) return;
  DataBase = SD.open(nameOfFile, FILE_WRITE);
  if (DataBase) {
    DataBase.println(dbString);
    DataBase.close();
    oldCodeString = codeString;
    old_in1out0 = in1out0;
    Serial.println("Writed");
    sdCardError = false;
  } else {
    Serial.println(" Not OPENED SD.open(nameOfFile, FILE_WRITE)");
    sdCardError = true;
  }
}
void handleSyncMessage(String msg) {
  if (msg.startsWith("ADD|")) {
    int id = msg.substring(4).toInt();
    if (id > 0 && id < TOGGLE_BASE_ADDR) {
      tempPresence[id] = true;
      collectingSync = true;
      lastSyncDataTime = millis();
      syncResponseReceived = true;
      if (EEPROM.read(id) != 1) {
        EEPROM.write(id, 1);
        Serial.print(" ADD ID: ");
        Serial.println(id);
      } else {
        Serial.print("ID is storage: ");
        Serial.println(id);
      }
    }
  }
}
void requestCardIDSync() {
  while (Serial.available() > 0) {
    char c = Serial.read();
    if (c == '<') {
      syncBuffer = "";
    } else if (c == '>') {
      handleSyncMessage(syncBuffer);
      syncBuffer = "";
    } else {
      syncBuffer += c;
    }
  }
```

```
  if (collectingSync && (millis() - lastSyncDataTime > 3000)) {
    if (!syncResponseReceived && !syncWarningShown) {
      Serial.println("Server not answer for REQ_SYNC — card ID not
update.");
      syncWarningShown = true;
    }
    if (syncResponseReceived) {
      Serial.println("Sync is end");
      for (int i = 1; i < TOGGLE_BASE_ADDR; i++) {
        if (EEPROM.read(i) == 1 && !tempPresence[i]) {
          EEPROM.write(i, 0);
          Serial.print("Delete card ID: ");
          Serial.println(i);
        }
      }
    }
    collectingSync = false;
  }
}
void loop() {
  requestCardIDSync();
  recvWithStartEndMarkers();
  if (newData1) {
    codeString = receivedChars;
    int workerCode = codeString.toInt();
    if (!isValidCardID(workerCode)) {
      Serial1.println("<NOCARD>");
      Serial.println(workerCode);
    } else {
      int addr = TOGGLE_BASE_ADDR + (workerCode % TOGGLE_SLOTS);
      byte lastState = EEPROM.read(addr);
      bool currentEventIsInput = !lastState;
      saveToSD(currentEventIsInput);
      digitalWrite(LOCK_PIN, HIGH);
      curtime = millis();
      endtime = curtime + locktime;
      OPEN_DOOR = true;
      EEPROM.write(addr, currentEventIsInput ? 1 : 0);
      oldCodeString = codeString;
      lastScanTime = millis();
      sendTimeToUno(currentEventIsInput);
    }
    newData1 = false;
  }
  curtime = millis();
  if (curtime > endtime) {
    digitalWrite(LOCK_PIN, LOW);
    OPEN_DOOR = false;
  }
}
```

Додаток В

Програмна реалізація модуля для розпізнавання обличчя

```
import cv2
import face_recognition
import os
import time
from utils.utils import saveFrame
from datetime import datetime
from config.config import FAKELOGSPHOTOPATH
def recognize_and_compare_face(expected_encoding, expected_name, card_id,
face_data, duration=5, tolerance=0.5):
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
    if not cap.isOpened():
        return False, None, None
    start_time = time.time()
    matched = False
    recognized_name = "Unknown"
    photo_path = None
    while time.time() - start_time < duration:
        ret, frame = cap.read()
        if not ret:
            continue
        # Підвищення чіткості + точніша обробка кольору
        small_frame = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
        rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
        face_locations = face_recognition.face_locations(rgb_small_frame,
model='cnn')
        face_encodings = face_recognition.face_encodings(rgb_small_frame,
face_locations)
        for face_encoding in face_encodings:
            # 1. Порівняння з очікуваним обличчям
            distance =
face_recognition.face_distance([expected_encoding], face_encoding)[0]
            if distance < tolerance:
                matched = True
```

```
                recognized_name = expected_name
                break  # одразу вихід при збігу
            # 2. Якщо не співпало — шукаємо найближчого серед усіх відомих
            closest_name = "Unknown"
            min_distance = 1.0
            for w in face_data.values():
          dist = face_recognition.face_distance([w["encoding"]], face_encoding)[0]
                if dist < min_distance and dist < tolerance:
                    min_distance = dist
                    closest_name = w["name"]
            # 3. Збереження фото, якщо ще не зберігали
            if photo_path is None:
                recognized_name = closest_name
                photo_path = saveFrame(frame, recognized_name, card_id)
        if matched:
            break
    cap.release()
    cv2.destroyAllWindows()
    return matched, recognized_name, photo_path
```

Додаток Г

Результати тестування сценарію «зчитування валідної картки»

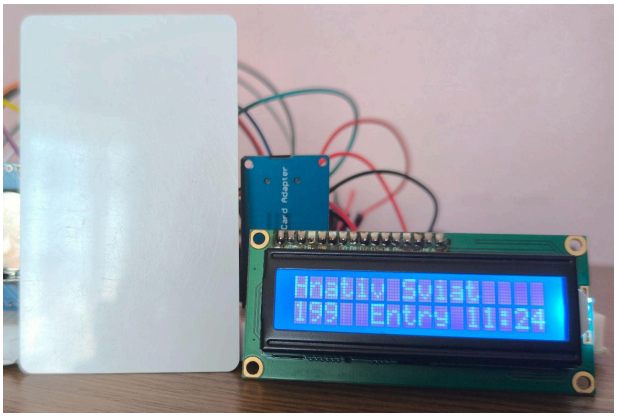

Рисунок Г.1 – Виведення повідомлення на LCD при зчитуванні валідної картки

```
[UART]: Data with UNO: 199
[UART]: File to write: 080625.txt
[UART]: dbString = 080625112401991
Decode: {'date': '2025-06-08', 'time': '11:24', 'cardID': '199', 'direction': 'Entry'}
Face access: Hnativ Sviatoslav
[UART]: Writed
[UART]: Send UNO time and direction: <Entry 11:24>
```

Рисунок Г.2 – Отримані дані на сервері при отриманні даних з валідної картки

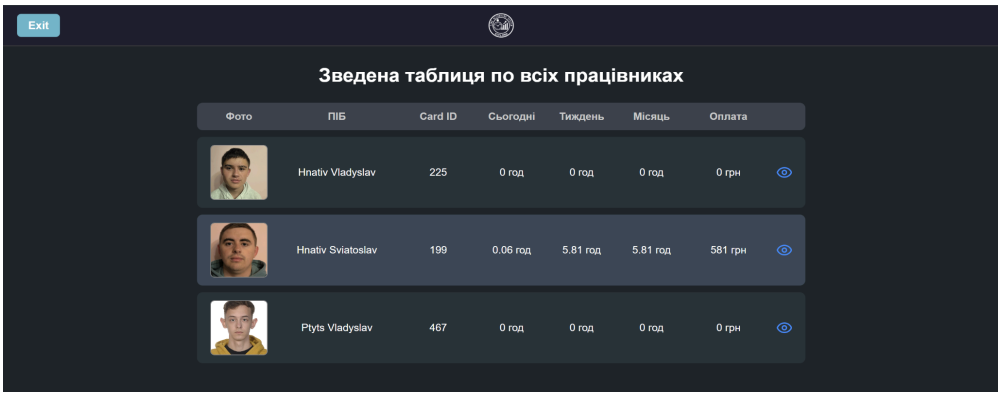

Рисунок Г.3 – Відображення коректно записаних даних з БД у веб інтерфейсі

Додаток Д

Результати тестування сценарію «зчитування невалідної картки»

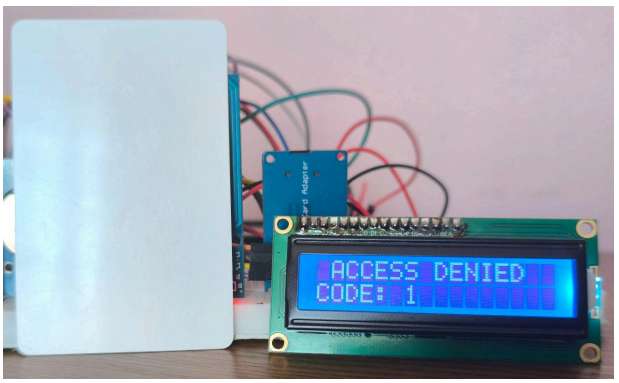

Рисунок Д.1 - Виведення повідомлення «ACCESS DENIED» на LCD при зчитуванні невалідної картки, якої немає у EEPROM або базі даних.

```
[UART]: Data with UNO: 1
[UART]: 1
[UART]: Data with UNO: 1
[UART]: 1
```

Рисунок Д.2 – Отримані дані на сервері при отриманні даних з невалідної картки

Додаток Е

Результати тестування сценарію «успішне розпізнавання обличчя»

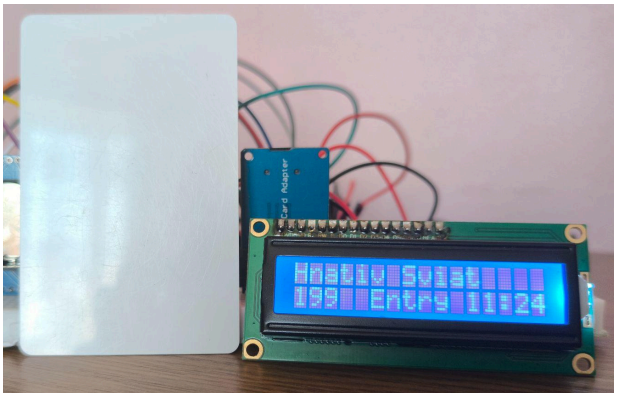

Рисунок Е.1 – Дані картки по якій здійснювався вхід

```
[UART]: Data with UNO: 199
[UART]: File to write: 080625.txt
[UART]: dbString = 080625112401991
Decode: {'date': '2025-06-08', 'time': '11:24', 'cardID': '199', 'direction': 'Entry'}
Face access: Hnativ Sviatoslav
[UART]: Writed
[UART]: Send UNO time and direction: <Entry 11:24>
```

Рисунок Е.2 – Результат розпізнавання обличчя (Face access: Hnativ Sviatoslav)

Додаток Ж

Результати тестування сценарію «фейковий вхід іншим обличчям»

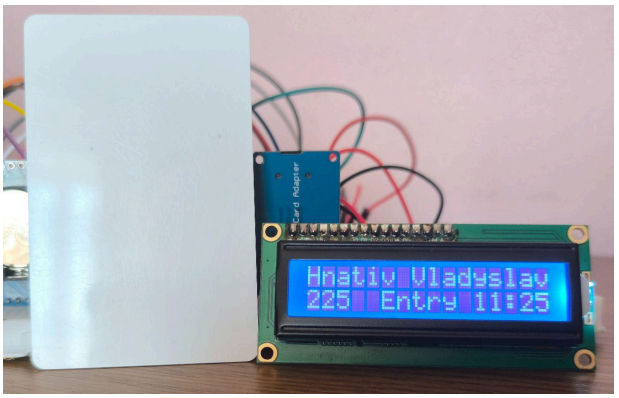

Рисунок Ж.1 – Дані картки по якій здійснювався вхід

```
[UART]: Data with UNO: 225
[UART]: File to write: 080625.txt
[UART]: dbString = 080625112502251
Decode: {'date': '2025-06-08', 'time': '11:25', 'cardID': '225', 'direction': 'Entry'}
Photo save in: D:\Time_According_System\server\data\fake_logs_people\225\Hnativ Sviatoslav_2025-06-08_11-25-57.jpg
Face is not access Hnativ Sviatoslav
[UART]: Writed
[UART]: Send UNO time and direction: <Entry 11:25>
```

Рисунок Ж.2 – Результат розпізнавання обличчя (Face is not access Hnativ Sviatoslav)

Додаток И

Результати тестування сценарію «запис до SD-карти»

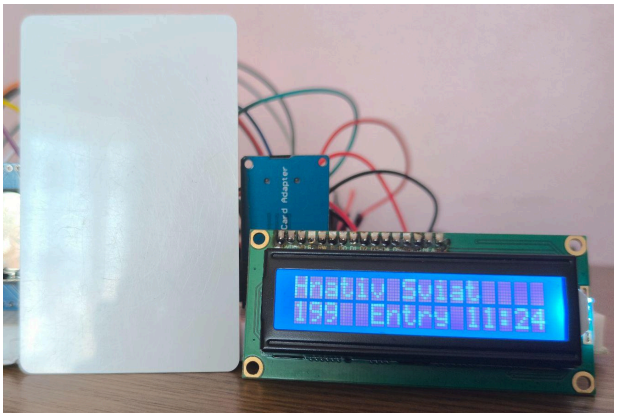

Рисунок И.1 – Дані картки які будуть записані на SD

```
[UART]: Data with UNO: 199
[UART]: File to write: 080625.txt
[UART]: dbString = 080625112401991
Decode: {'date': '2025-06-08', 'time': '11:24', 'cardID': '199', 'direction': 'Entry'}
Face access: Hnativ Sviatoslav
[UART]: Writed
[UART]: Send UNO time and direction: <Entry 11:24>
```

Рисунок И.2 – Відображення на сервері у який файл записані дані (File to write: 080625.txt)

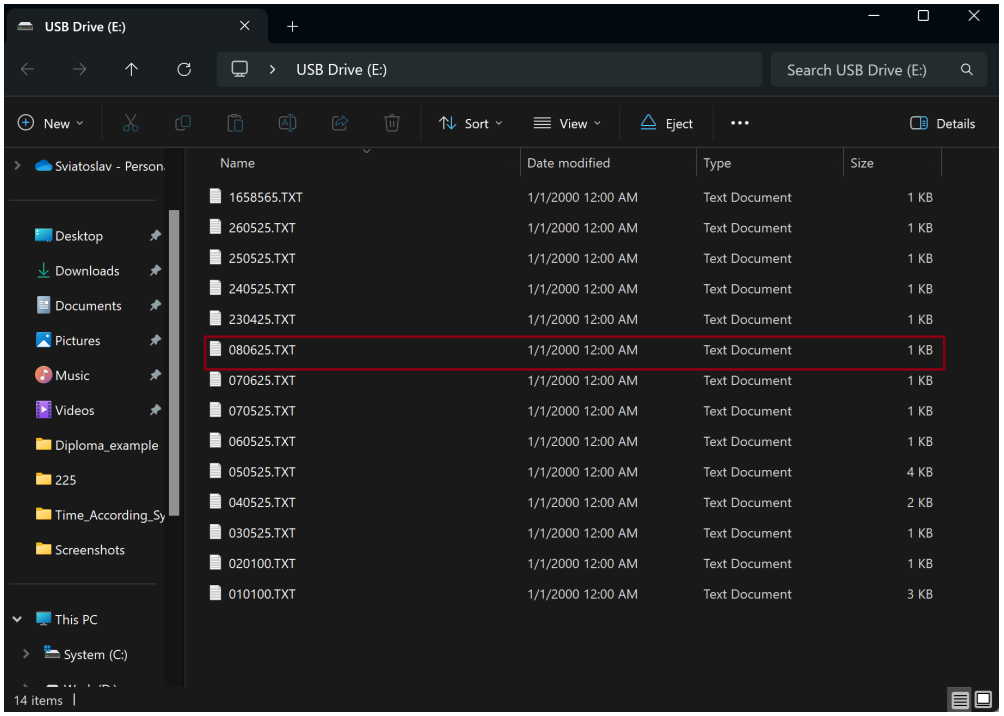

Рисунок И.3 –Файл на флешці куди записані дані

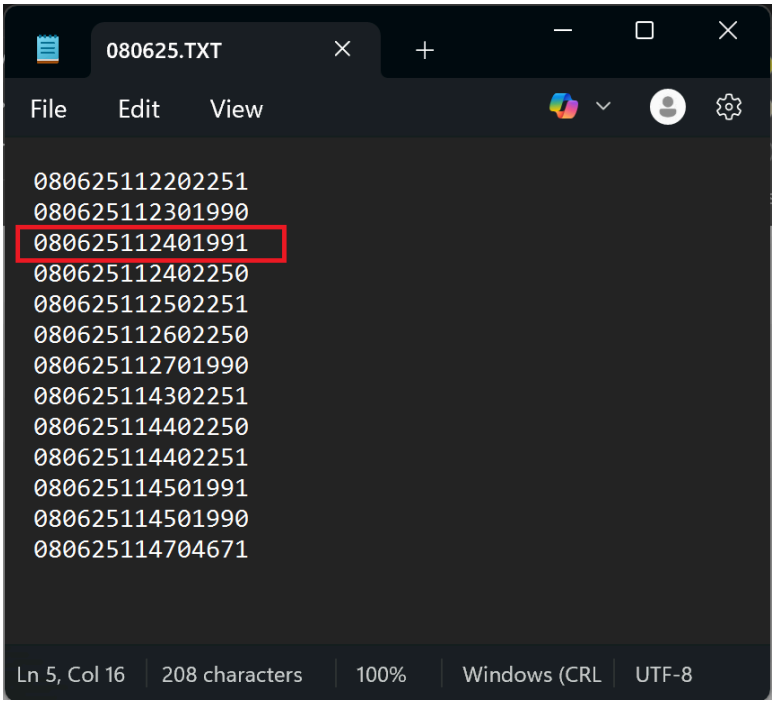

Рисунок И.5 –Записаний рядок у файлі

Додаток К

Результати тестування сценарію «втрата з’єднання з портом»

```
Find port: COM3 — USB-SERIAL CH340 (COM3)
Connected to COM3
```

Рисунок К.1 – Порт до якого спочатку підключена апаратна частина

```
Reconnecting attempt 1/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 2/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 3/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 4/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 5/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 6/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 7/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 8/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 9/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Reconnecting attempt 10/10...
Reconnect failed: 'NoneType' object has no attribute 'port'
Could not reconnect. Searching for new port...
```

Рисунок К.2 – Процес перепідключення до порту

```
Find port: COM5 — USB-SERIAL CH340 (COM5)
Connected to COM5
```

Рисунок К.3 – Результат перепідключення до порту

Додаток Л

Результати тестування сценарію «додавання працівника»

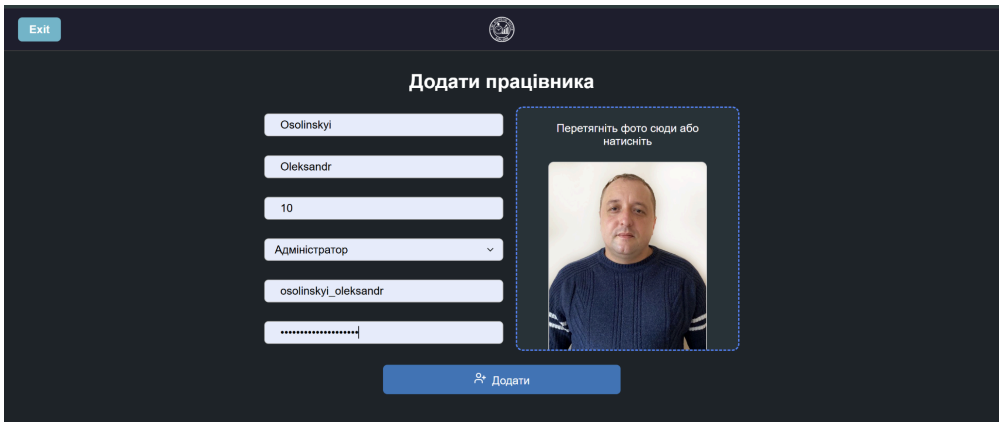

Рисунок Л.1 – Сторінка для введення даних (адміністратор вводить дані користувача у відповідні поля)

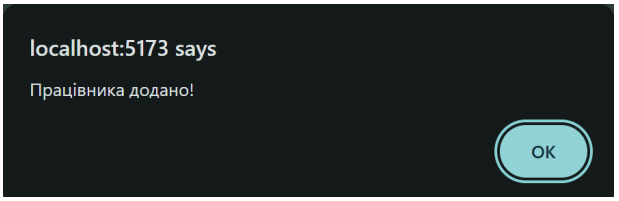

Рисунок Л.2 – Повідомлення про успішне додавання працівника

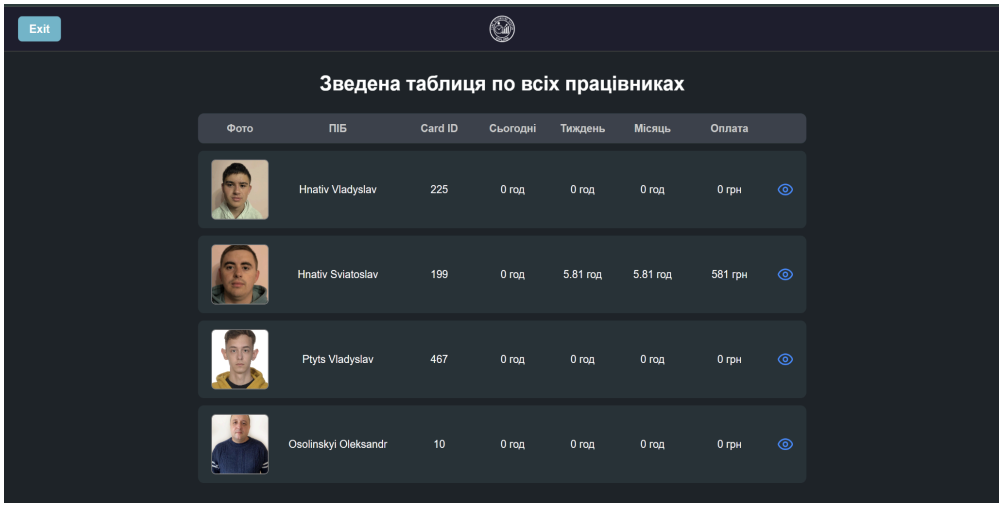

Рисунок Л.3 –Результат успішного додавання працівника

Додаток М

Результати тестування сценарію «оновлення працівника»

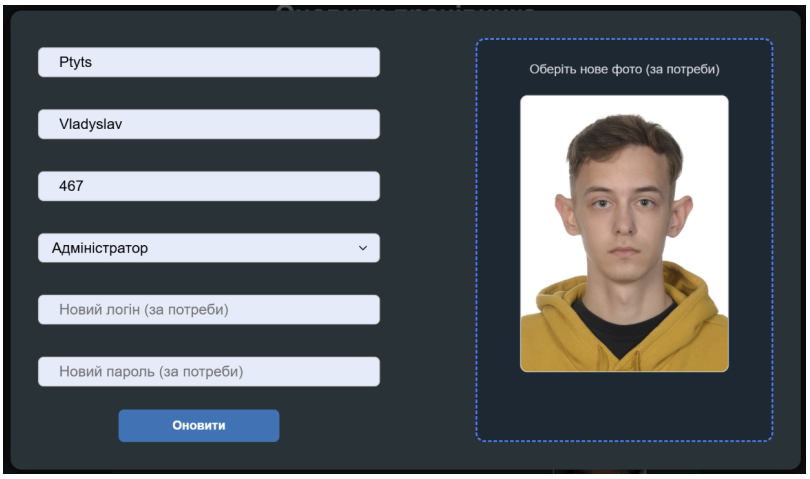

Рисунок М.1 –Дані до оновлення

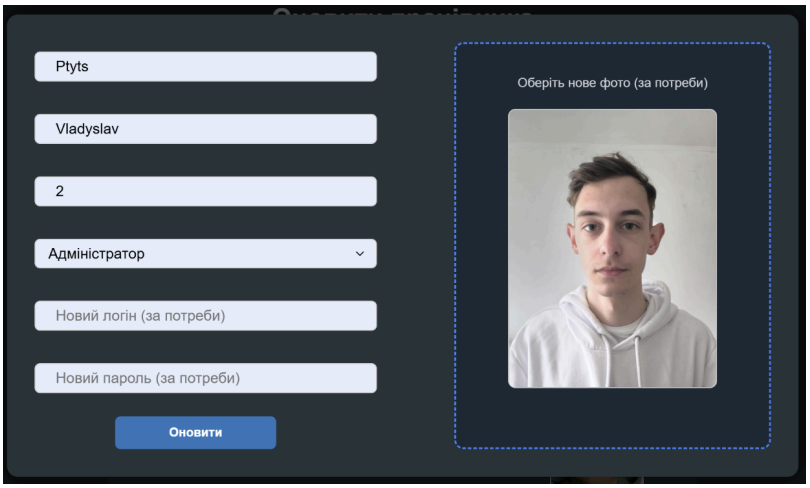

Рисунок М.2 –Нові дані для працівника

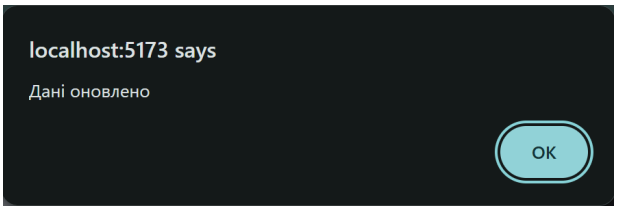

Рисунок М.3 – Повідомлення про успішне оновлення даних працівника

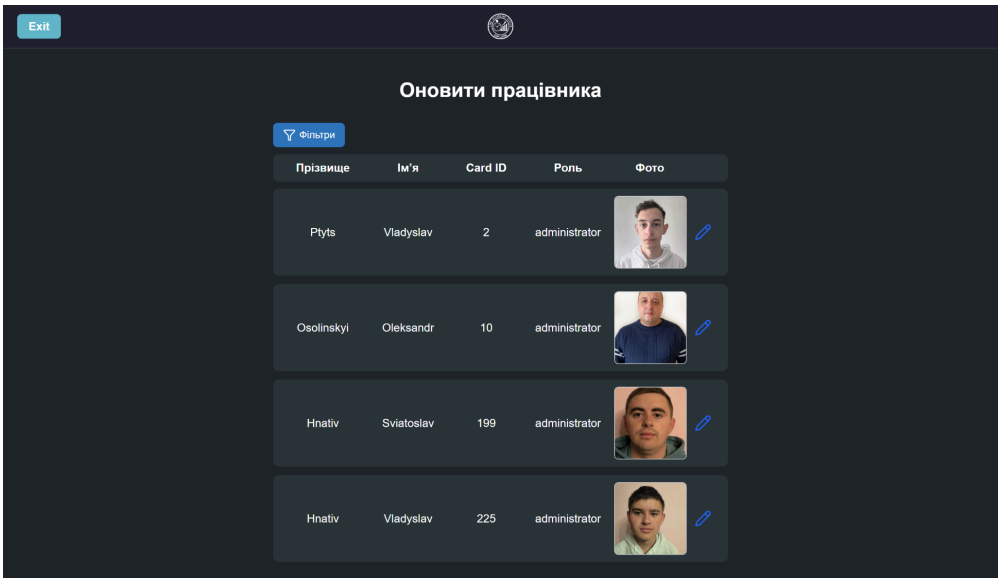

Рисунок М.4 –Результат оновлення даних працівника

Додаток Н

Результати тестування сценарію «видалення працівника»

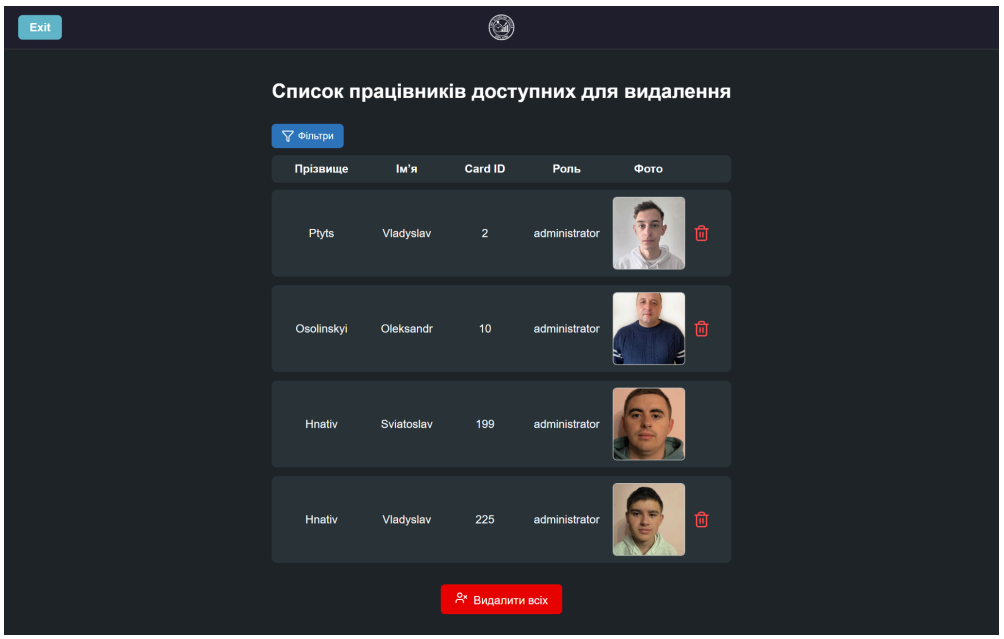

Рисунок Н.1 –Список працівників до видалення

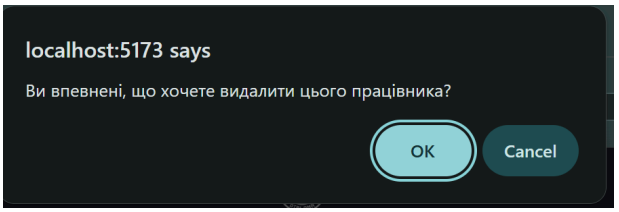

Рисунок Н.2 –Повідомлення для підтвердження видалення працівника

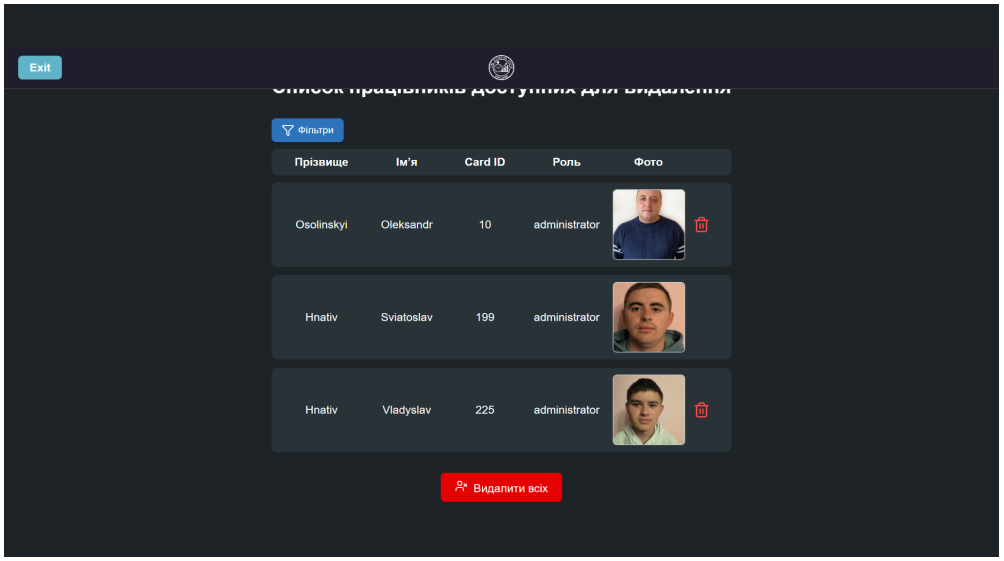

Рисунок Н.3 – Список працівників після видалення

Додаток П

Результати тестування сценарію «перегляд денних, місячних годин»

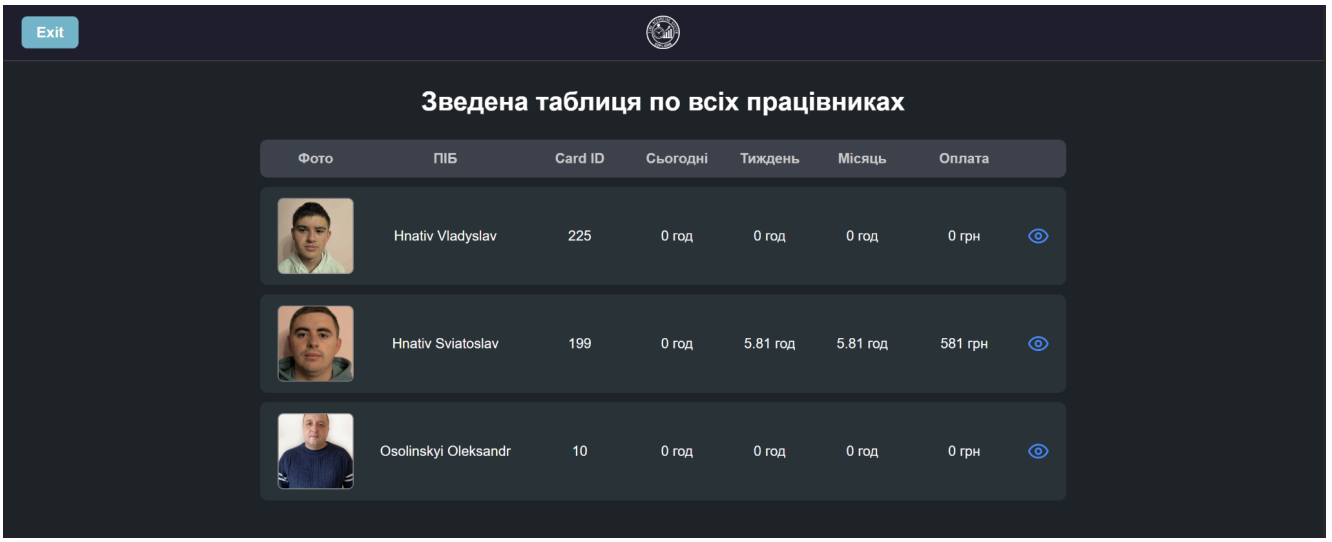

Рисунок П.1 – Сторінка із загальною аналітикою відпрацьованого часу

Рисунок П.2 – Сторінка із загальною аналітикою відпрацьованого часу

Додаток Р

Результати тестування сценарію «перегляд фейкових логів»

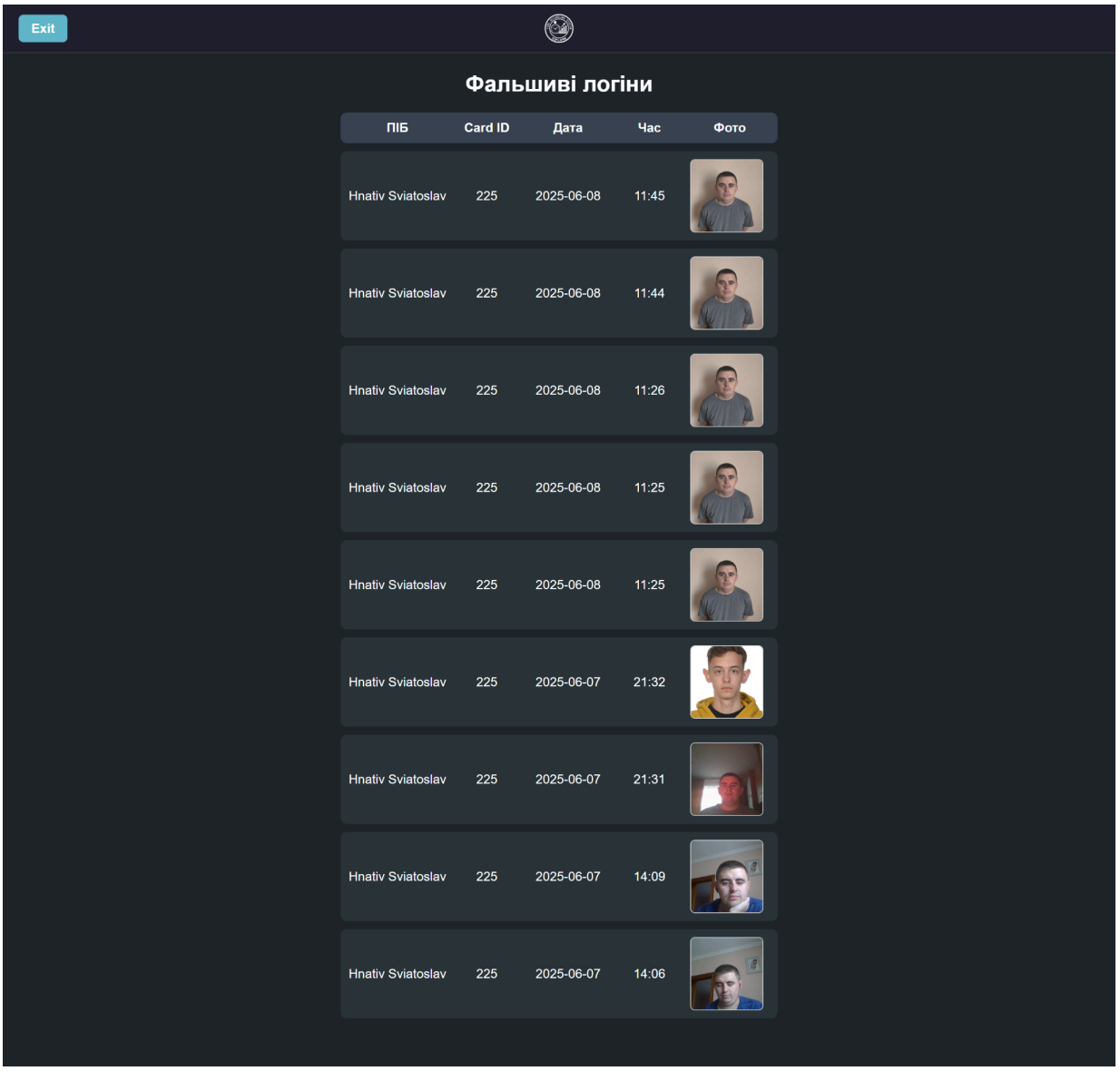

Рисунок Р.1 – Сторінка із для перегляду фейкових логінів

Додаток С

Копія публікації

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(ІІТАR-2025)

27-29 травня 2025 року

Тернопіль

2025

Збірник тез доповідей студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR-2025). (Тернопіль, 27-29 травня 2025 року). Тернопіль: ЗУНУ, 2025. 372 с.

До збірника увійшли тези доповідей учасників студентської науково-практичної конференції "Інтелектуальні інформаційні технології в прикладних дослідженнях" (IITAR – 2025), що відбувалась у рамках AI Week на базі кафедри інформаційно-обчислювальних систем і управління Західноукраїнського національного університету. Мета конференції — об'єднати науковців, освітян, представників IT-бізнесу та здобувачів освіти для обміну досвідом, презентації сучасних досліджень і впровадження інноваційних рішень у сфері комп'ютерних наук, штучного інтелекту та суміжних галузей.

За зміст наукових праць та достовірність наведених фактологічних і статистичних матеріалів відповідальність несуть автори публікацій та їхні наукові керівники. У збірнику зберігається стилістика та орфографія авторів матеріалів.

Склад організаційного комітету конференції

Керівництво оргкомітету

Мирослав Комар, д.т.н., професор, професор кафедри інформаційно-обчислювальних систем і управління

Христина Ліп'яніна-Гончаренко, д.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Надія Васильків, к.т.н., в.о. завідувача кафедри інформаційно-обчислювальних систем і управління

Члени програмного комітету

Василь Коваль, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління, заступник декана факультету комп'ютерних інформаційних технологій

Олександр Осолінський, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Павло Биковий, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Діана Загородня, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Ігор Майків, к.т.н., доцент кафедри інформаційно-обчислювальних систем і управління

Члени оргкомітету

Андрій Івасечко, викладач кафедри інформаційно-обчислювальних систем і управління

Дмитро Дюг, викладач кафедри інформаційно-обчислювальних систем і управління

Христина Юрків, студентка ФКІТ ЗУНУ, технік лабораторії з проблем інформаційних технологій

Мар'яна Соя, студентка ФКІТ ЗУНУ, лаборант кафедри інформаційно-обчислювальних систем і управління

Микола Телька, студент ФКІТ ЗУНУ, лаборант кафедри інформаційно-обчислювальних систем і управління, студентський декан факультету комп'ютерних інформаційних технологій

Гнатів Святослав
студент групи КН-43
s.hnativ@st.wunu.edu.ua
Олександр Осолінський
к.т.н., доцент
oso@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

ПРОГРАМНИЙ МОДУЛЬ ОБЛІКУ РОБОЧОГО ЧАСУ З ВИКОРИСТАННЯМ IoT ТЕХНОЛОГІЙ

Облік робочого часу є одним з ключових елементів управління виробничими процесами. У традиційних системах, що базуються на ручному введенні чи RFID, часто фіксуються недостовірні дані через людський фактор або зловживання — наприклад, передача картки іншій особі [1]. Ці недоліки можна усунути, впровадивши інтелектуальні IoT-рішення з багаторівневою верифікацією особи.

У роботі розроблено вбудовану систему обліку робочого часу з використанням Arduino, RFID-ідентифікації та модуля розпізнавання обличчя. Система побудована на взаємодії контролерів Arduino UNO та MEGA 2560, модулів RTC (DS3231), SD-карти, LCD-дисплея та Python-серверної частини з OpenCV та PostgreSQL. Алгоритм забезпечує попередню перевірку UID у локальній пам'яті (EEPROM), реєстрацію подій навіть у разі втрати зв'язку з сервером, та збереження фото у разі фальсифікації.

Інформація зчитується RFID-зчитувачем та відображається на дисплеї. Дані передаються до MEGA, де фіксуються на SD-карті. Після цього активується процес розпізнавання обличчя через сервер, що порівнює зображення із базою еталонів. Успішна подія фіксується в accessLogs, спроба фальсифікації — у fakeLogs разом із фото.

Систему реалізовано у вигляді модульної архітектури, що включає:

- апаратний рівень: зчитувач MFRC522, модуль реального часу DS3231, LCD-дисплей;
- програмний рівень: обробка подій на Arduino та передача через UART;
- аналітичний рівень: Python-сервер із системою розпізнавання на основі face_recognition, база PostgreSQL, веб-інтерфейс.

На рисунку 1 подано функціональну архітектуру програмного модуля, що охоплює як апаратну, так і програмну частину.

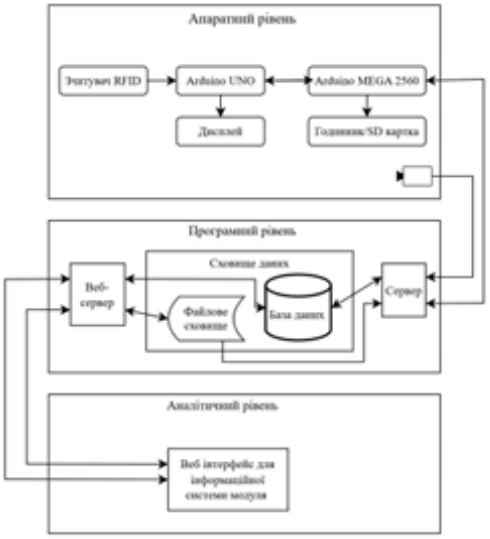

Рисунок 1 — Архітектура IoT-модуля обліку часу з розпізнаванням обличчя

Порівняльний аналіз систем SpeedFace V5 [2], Suprema FaceStation F2 [3] та ZKTeco uFace800 [4] засвідчив, що більшість комерційних рішень залежать від хмарних сервісів, мають високу вартість, обмежену гнучкість та не забезпечують локального збереження подій. Розроблена система вирізняється автономністю, дешевизною, відкритістю до розширення та локальною фіксацією подій, що робить її придатною для підприємств малого та середнього бізнесу.

Наукове та практичне значення полягає у можливості інтеграції модуля в існуючі виробничі процеси для забезпечення точного обліку часу та автоматичного виявлення фальсифікацій. Це відповідає сучасним викликам цифровізації та індустрії 4.0 [5, 6].

Ключові слова: IoT, облік часу, RFID, розпізнавання обличчя, Arduino, веб-інтерфейс, PostgreSQL, автономна система.

Список використаних джерел

1. Hurma. Облік робочого часу: що це таке та як його вести? [Електронний ресурс]. – Режим доступу: https://hurma.work/blog/oblik-robochogo-chasy
2. ZKTeco Systems. SpeedFace-V5. [Електронний ресурс]. – Режим доступу: https://zkteco.systems/ua/product/speedface-v5
3. Suprema. FaceStation F2 — мультифункціональний біометричний термінал. [Електронний ресурс]. – Режим доступу: https://supremainc.com.ua
4. ZKTeco Systems. uFace800 facial multi-biometric terminal. [Електронний ресурс]. – Режим доступу: https://zkteco.systems/en/product/uface800

5. IBM. What is the Internet of Things (IoT)?. [Електронний ресурс]. – Режим доступу: https://www.ibm.com/think/topics/internet-of-things
6. Yan J., Meng Y., Lu L., Li L. Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes and Applications // *IEEE Transactions on Industrial Informatics*. – 2017.
7. Ghosh A., et al. Predictive Maintenance Using Machine Learning: A Review // *Procedia Computer Science*. – 2020. – Vol. 170. – P. 702–707.