Галицький коледж імені В’ячеслава Чорновола
відділення комп’ютерних та видавничих технологій
циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач відділенням
комп’ютерних та видавничих
технологій
Чубей О.О. /_____/
підпис
«___» _____ 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту
освітньо-кваліфікаційного рівня «молодший спеціаліст»
зі спеціальності 122 «Комп’ютерні науки»

на тему: «Система автоматизованого обліку присутності працівників коледжу в приміщенні»

Студент групи К-47 Чекановський А.Б. _____
(підпис)

Керівник проєкту Павлюс В.П. _____
(підпис)

Консультанти:

з техніко-економічного обґрунтування Меленчук Л.І. _____
(підпис)

нормоконтролер Гавришків Н.Г. _____
(підпис)

Тернопіль – 2021

Галицький коледж імені В’ячеслава Чорновола
відділення комп’ютерних та видавничих технологій
циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ
Завідувач відділенням
комп’ютерних та видавничих
технологій
Чубей О.О. /_____/
підпис
«___» _____ 2020 р.

ЗАВДАННЯ

на дипломне проєктування
на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»
студенту Чекановському Андрію Богдановичу

(прізвище, ім’я та по-батькові студента)

1. Тема проєкту «Система автоматизованого обліку присутності працівників коледжу в приміщенні»
   затверджено наказом по коледжу від “___” _____ 202_ р., №___
2. Термін здачі студентом завершеного проєкту “___” _____ 202_ р.
3. Вихідні дані до проєкту _____
_____
_____
4. Перелік питань, які повинні бути розроблені в проєкті:

а) основна частина _____
_____

б) техніко-економічне обґрунтування _____
_____
_____
_____

5. Перелік графічного матеріалу _____
_____

6. Консультанти проєкту: _____–

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	(вчена ступінь, звання П.І.Б. консультанта)		

КАЛЕНДАРНИЙ ПЛАН
дипломного проєктування

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми, ознайомлення з вимогами до дипломного проєктування	16.11.20	30.11.20
2.	Огляд типових рішень та написання відповідного розділу ПЗ	01.12.20	26.01.21
3.	Дослідження технологій реалізації та написання відповідного розділу ПЗ	27.01.21	15.02.21
4.	Розробка функціональних вимог до проєкту та робота над структурою програмного продукту. Написання відповідного розділу ПЗ	15.02.21	02.03.21
5.	Встановлення та налаштування середовища реалізації та написання відповідного розділу ПЗ	02.03.21	16.03.21
6.	Проєктування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	16.03.21	16.04.21
7.	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	17.04.21	03.05.21
8.	Доопрацювання модулів	03.05.21	17.05.21
9.	Опрацювання економічного розділу дипломного проєкту та оформлення спеціального розділу	17.05.21	18.06.21
10.	Тестування та налагодження програмного продукту та написання відповідного розділу ПЗ	18.05.21	04.06.21
11.	Робота над оформленням пояснювальної записки	04.06.21	11.06.21
12.	Попередній захист дипломного проєкту, доопрацювання	11.06.21	
13.	Підготовка до захисту дипломного проєкту	18.06.21	23.06.21
14.	Захист дипломного проєкту	24.06.21	24.06.21

7. Дата видачі “___” _____ 2020р. Керівник _____/

Завдання прийняв до виконання _____/

Реферат

Система автоматизованого обліку присутності працівників коледжу в приміщенні. Дипломний проєкт. Чекановський Андрій Богданович. Галицький коледж імені В’ячеслава Чорновола, відділення комп’ютерних та видавничих технологій. Спеціальність 122 «Комп’ютерні науки». ГК, 2021. Сторінок – 101, рисунків – 53, додатків – 3.

Дипломна робота присвячена дослідженню сучасних технологій розпізнавання облич, розробці системи обліку відвідуваності з технологією розпізнавання в реальному часі.

У першому розділі дипломної роботи було проведено аналіз предметної області, розглянуто декілька існуючих рішень та описано постановку завдання.

У другому розділі здійснено формалізацію вимог до системи, а саме – вимог до програмного продукту, інтерфейсу користувача та бази даних. Виконано проєктування системи та її складових, а також описано алгоритм її функціонування.

Третій розділ розкриває технології та засоби реалізації, містить опис розробки і тестування основних функцій системи та інтерфейсу.

В четвертому розділі проводиться техніко-економічне обґрунтування.

Об’єкт дослідження – процес розпізнавання облич в реальному часі, засоби моніторингу відвідуваності.

Метою проєкту є реалізація системи автоматизованого обліку присутності працівників коледжу в приміщенні, тобто системи, яка дозволить здійснювати фіксацію працівника на робочому місці.

Результатом роботи над проєктом є програмне забезпечення фіксації та обліку відвідуваності працівників.

РОЗПІЗНАВАННЯ, ВІДВІДУВАНІСТЬ, ОБЛІК, ПРИСУТНІСТЬ, ФІКСАЦІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ОБЛИЧЧЯ, БІОМЕТРИЧНІ ТЕХНОЛОГІЇ, СИСТЕМА, ОБРОБКА, ВИЛУЧЕННЯ ОЗНАК.

Abstract

Automated indoor attendance accounting system for college employees. Diploma project. Chekanovskiy Andriy. Galytskyy College Named After V. Chornovil, Department of Computer and publishing technologies. Specialty 122 "Computer Science". GC, 2021. Pages – 101, figures – 53, appendixes – 3.

The paper is devoted to the study of modern facial recognition technologies, the development of a real-time recognition system.

In the first chapter of the paper, the subject area was analyzed, several existing solutions were considered, and the problem statement was described.

The second chapter formalizes the requirements for the system, namely, the requirements for the software product, user interface and database. The system and its components are designed, and the algorithm of its functioning is described.

The third section covers technologies and implementation tools, describes the implementation and testing of the main functions of the system and interface.

In the fourth chapter, a feasibility study is conducted.

The object of research is the real-time facial recognition process and attendance monitoring tools.

The goal of the project is to implement a system for automated recording of the presence of college employees at the working place.

The result of the project is software for recording and accounting employee attendance.

RECOGNITION, ATTENDANCE, ACCOUNTING, PRESENCE, RECORDING, SOFTWARE, FACES, BIOMETRIC TECHNOLOGIES, SYSTEM, PROCESSING, FEATURE EXTRACTION.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ – програмне забезпечення

VIP – very important person

RFID – radio frequency identification

БД – база даних

ПЗ – програмне забезпечення

HOG – histogram of oriented gradients

ERD – entity relationship model

МН – машинне навчання

ШІ – штучний інтелект

GUI – graphical user interface

PHP – hypertext preprocessor

DOM – document object model

AJAX – asynchronous javascript and xml

СУБД – система управління базами даних

ПК – персональний комп'ютер

ІС – інформаційна система

SQL – structured query language

CSS – cascading style sheets

HTML – hypertext markup language

BLOB – binary large object

ВСТУП

Системи, які інтегрують у собі біометричні технологій, в тому числі і розпізнавання обличчя, вагомо спрощують взаємодію людини з комп'ютером чи його програмним забезпечення. Це спричинено тим, що з їх допомогою можна пришвидшити значну кількість операцій пов'язаних з процесами ідентифікації.

Сучасні інновації та технології призвели до чималих успіхів, які допомагають різним сферам рости, розвиватися та досягати вищого рівня продуктивності. Одним із типів технологій, який все більше знаходить своє застосування є біометричні технології. Це, як правило, пов'язано з ідентифікацією людини на основі певного аспекту її біології. Однією з перших біометричних технологій є розпізнавання відбитків пальців, що значною мірою почало застосовуватись в криміналістиці. Іншою технологією, яка набуває все більшої тенденції у всьому світі є система розпізнавання облич. Тисячі корпорацій та урядових організацій використовують її через високий рівень безпеки, надійності та зручності.

Ця технологія значно покращує процес нагляду та спостереження за кимось. Найпростішим застосуванням подібної системи, підключеної до мережі камер, може бути автоматичне відстежування людини під час її переміщення та виходу із будівлі, навіть якщо про неї не відомо жодної іншої інформації. У більш складному варіанті, система розпізнавання обличчя, що підтримується великою базою із збереженими даними про певних осіб, може дозволити поліції визначити зацікавлену особу в місті через мережеві камери.

Значна кількість ДТП, яка спричиняє смерть або травму, пов'язана з перевтомою водіїв. Втрата контролю над вантажним автомобілем є однією з найбільших небезпек на дорогах. Розпізнавання обличчя може це змінити. Якщо рухи та поведінка водія вказують на його перевтому, система, наприклад, може подати звуковий сигнал, автоматично вимкнути круїз-контроль, поступово сповільнюючи транспортний засіб.

Можливість негайної ідентифікації VIP-гостей дозволить співробітникам установи отримати інформацію про відвідувача перед тим, як гість піддійте до стійки реєстрації. Загалом, ця можливість дає змогу покращити обслуговування VIP-гостя за допомогою персоналізації послуг. База даних ідентифікації клієнтів може визначати, яким особам надається знижка чи спеціальна пропозиція.

Управління відвідуванням персоналу є ключовим у продуктивності підприємства чи організації. Для вирішення цієї проблеми використовуються різноманітні інтелектуальні системи автоматичного управління відвідуванням. Автоматизована система для фіксації та обліку присутності на основі технологій розпізнавання обличчя стане тим рішенням, яке дозволить перейти на новий етап в управлінні персоналом.

Завдяки своїй простоті, зрозумілості та масштабованості дана система відповідатиме основним вимогам інформаційного суспільства. Це зможе покращити робочий процес та забезпечити кращий його контроль. З даною метою була поставлена задача реалізувати систему автоматизованого обліку присутності працівників.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Обґрунтування доцільності створення системи

Система розпізнавання облич надає величезні переваги в порівнянні із технологією відбитків пальців чи долоні. Основною з них є те, що вона працює на відстані і не потребує прямої взаємодії з людиною. Система автоматизованого розпізнавання може бути використана в різних цілях. Серед найпопулярніших можна виділити такі:

- виявлення, затримання та переслідування терористів та злочинців;
- розблокування телефона;
- визначення ознак втоми водія та запобігання аварій;
- здійснення паспортного контролю в аеропортах і на інших пунктах контролю доступу;
- вхід на заходи VIP-гостей;
- аналіз симптомів пацієнтів по обличчю;
- зменшення часу реєстрації;
- облік робочого часу та відвідуваність робітників.

Біометрія швидко стала популярним способом відстеження відвідуваності робочого часу для багатьох провідних компаній світу. Біометрична система надзвичайно корисна як зі сторони ведення записів відвідування, так і для того, щоб допомогти персоналу відчувати себе в безпеці. Оскільки вона опирається на ті особисті характеристики, які варіюються у різних людей, будь-яка особа не буде мати можливості автентифікуватись в системі. Біометричні характеристики не можуть бути дубльованими, це запобігає підробленню та втручанню колег, що працюють в одній компанії до журналу відвідування.

Ручна практика системи відвідуваності забирає багато часу і вимагає повного особистого моніторингу, в той час як біометрична відвідуваність економить час співробітників, знижує накладні витрати на персонал і надає точні трудові дані в систему заробітної плати для ефективного управління бізнес-операціями і тим самим підвищує продуктивність усередині організації. Більше

того, система біометричного відвідування принесе прозорість та покращить культуру роботи в системі.

Біометрична система відвідуваності стає все більш корисною в секторі освіти через обмеження і неточності наявних методів ведення обліку відвідуваності. Традиційні методи фіксації відвідуваності, як студентів так і працівників, такі як перекличка або відвідуваність на основі записної книжки, застаріли і забирають багато часу. Традиційна система відвідуваності схильна до неточностей і помилок.

Біометрична система на основі розпізнавання облич має ряд переваг над ручною:

– Легка у використанні. Все, що потрібно — це декілька кліків та розташувати обличчя в полі зору камери, щоб ідентифікувати особу і зареєструвати час прибуття або відходу.

– Точність. Немає можливості шахрайства або фальсифікації даних фізіологічних характеристик, захоплених програмним забезпеченням.

– Запобігання присутності довірених осіб. Проблема присутності довірених осіб переслідує установи протягом десятиліть. За допомогою ручних систем відвідуваності, як студенти так і працівники могли вплинути на когось реєструватися на них під своїм іменем. Однак, система заснована на біометрії є надійною і захищеною від такого обману.

– Економія часу. Програмне забезпечення для відвідування з підтримкою біометрії – це можливість значно зекономити час. До того ж, записи відвідуваності можуть зберігатись не у масивних журналах, а у електронних таблицях чи у певній базі даних. З програмним забезпеченням, яке дозволить автоматизовано здійснювати відмітки відвідуваності, можливість розширення функціоналу є необмеженою, починаючи від різних типів графіків, діаграм до інформаційних панелей, миттєвої генерації консолідованих звітів.

1.2 Огляд існуючих рішень

1.2.1 Біометричнний термінал SpeedFace-V5L

Zkteco – це підприємство, що спеціалізується на виробництві та розробці біометричних і RFID-технологій в індустрії безпеки. Основна увага приділяється біометрії, такій як розпізнавання обличчя, розпізнавання долонних вен, відбитків пальців та райдужної оболонки ока, що застосовується для обліку робочого часу, систем контролю доступу, відеоспостереження та багатьох інших областей.

SpeedFace-V5L – це високошвидкісний біометричний термінал контролю доступу з розпізнаванням осіб (рисунок 1.1).

Рисунок 1.1 – Біометричний термінал ZKTeco SpeedFace-V5L

Серія SpeedFace-V5L – це повністю модернізована версія терміналу розпізнавання осіб SpeedFace-V5L Visible Light, з потужними алгоритмами розпізнавання і новітніми технологіями комп'ютерного зору. Термінал підтримує як перевірку обличчя, відбитка пальця так і долоні. Він містить достатньо великий об’єм пам’яті, здійснює надшвидке розпізнавання, а також демонструє високі показники безпеки у всіх аспектах.

Функціональні особливості:

– мультиверифікація (сканування обличчя, відбиток пальця, за геометрією руки, доступ паролем);

– камера для розпізнавання облич в режимі реального часу;

– 5-дюймовий екран;

– розпізнавання обличчя під кутом до 30 градусів;

– розпізнавання облич у масці зі створенням 3D-моделі;

– виявлення підвищеної температури.

Даний продукт має велику кількість переваг. Серед них можна визначити такі: високий рівень безпеки, що включає наявність антиспуфінгового алгоритму, великий об’єм пам’яті, наявність інтерфейсу, висока швидкість розпізнавання (до 1 с.), дистанція розпізнавання (до 3 м.), функція вимірювання температури, зручна навігація. До недоліків можна віднести високу ціну на термінал.

Технічні характеристики:

– дисплей: TFT 5" сенсорний;

– пам'ять шаблонів долоні: 3000;

– пам'ять відбитків пальців: 6000;

– пам'ять шаблонів облич: 6000;

– журнал подій: 200 000;

– операційна система: Linux;

– апаратна частина: двоядерний процесор (900МГц), пам'ять 512MB RAM, камера 2MP, настроювана яскравість LED-підсвітки;

– інтерфейси зв'язку: TCP/IP;

– швидкість розпізнавання обличчя: ≤ 1 с;

– алгоритми: ZKFinger V10.0, ZKFace V5.8, ZKPalm V12.0;

– джерело живлення: 12В/3А;

– температура експлуатації: -10°C ~ 44°C;

– вологість: 10% ~ 95%;

– розміри: ширина – 92мм, довжина – 220мм, товщина – 22мм. [1]

Дані з терміналу можна вигружати в Excel, що дає змогу здійснити інтеграцію з 1С чи іншою системою.

Пристрої ZKTeco мають власні веб-застосунки для управління відвідуваністю і часом, які забезпечують постійне підключення до автономних push-комунікаційних пристроїв по Ethernet/3G/Wi-Fi/GPRS. Серед них можна виділити BioTime 8.0 (рисунок 1.2). Він розрахований на самообслуговування співробітників за допомогою мобільного додатку і веб-браузера, а також працює як приватна хмара.

За допомогою веб-браузера адміністратори можуть отримати доступ до BioTime 8.0 будь-де. Він легко обробляє декілька сотень пристроїв та тисячі транзакції співробітників.

BioTime 8.0 постачається з інтуїтивно зрозумілим користувальницьким інтерфейсом, з можливістю управління розкладом, його зміною і графіком, забезпечений функцією генерування звітів про відвідуваність.

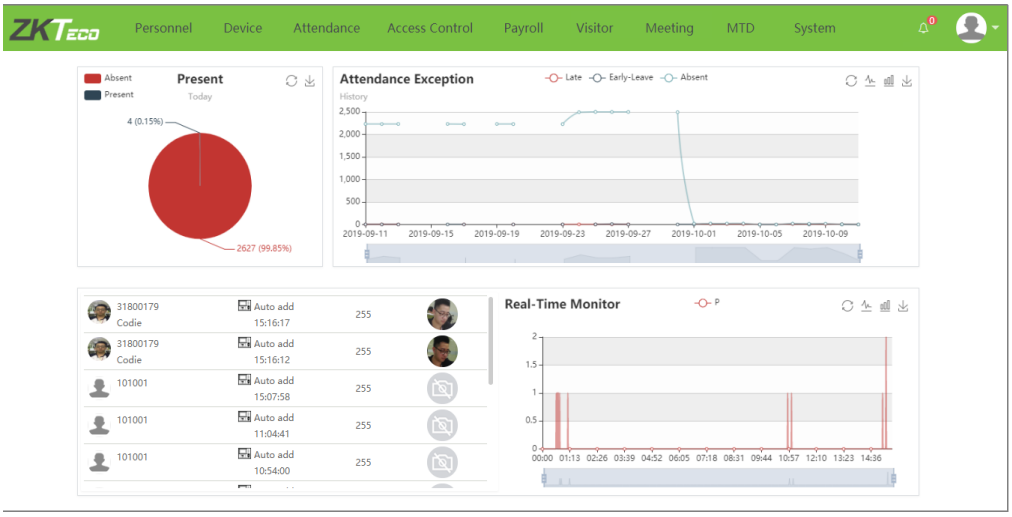


Рисунок 1.2 – Програмне забезпечення BioTime 8.0

1.2.2 Термінал Face Pass 7

Anviz є світовим лідером в області інтелектуальної безпеки, включаючи біометрію, RFID і відеоспостереження. Anviz займається створенням біометричних систем контролю доступу, систем обліку робочого часу та відвідуваності, а також розробкою сучасних камер відеоспостереження.

Anviz Face Pass 7 – біометричний термінал для систем контролю доступу та обліку робочого часу з функцією розпізнавання осіб (рисунок 1.3).

Рисунок 1.3 – Біометричний термінал Anviz Face Pass 7

Він оснащений новою архітектурою глибокого навчання ШІ та інфрачервоною технологією розпізнавання в реальному часі. FacePass 7 має унікальний алгоритм Anviz BioNANO для високоточного, стабільного та швидкого розпізнавання Він працює у будь-яких умовах освітлення: від яскравого освітлення до повної темряви.

Переваги:

– система обліку робочого часу співробітників з розпізнаванням по обличчю та за допомогою RFID-карток або паролю;

– ідентифікація користувача менш ніж за 0,5 секунди;

– дві скануючі камери забезпечують максимально точну ідентифікацію;

– забезпечення стабільної роботи пристрою як в приміщеннях з поганою освітленістю, так і в повній темряві за рахунок інфрачервоного підсвічування;

– ємність на 3 000 користувачів, журнал - на 100 тис. записів подій;

– голосовий супровід дій користувача, що забезпечує зручне користування терміналом;

– пряме управління замком;

– USB-flash для завантаження та вивантаження даних;

– TCP/IP для управлінням пристроєм і обміну даними по мережі Internet;

– сигнал тривоги при розтині корпусу системи;
– вбудований web-сервер;
– сучасний дизайн.

Технічні характеристики:

– процесор: двоядерний, 1ГГц;
– алгоритм: BioNANO;
– кількість скануючих камер: 2;
– дисплей: 3,2 " кольоровий, сенсорний;
– роздільна здатність дисплея: 240 х 320;
– кількість користувачів в пам'яті пристрою: 3 тис.;
– ємність журналу подій: 100 тис.;
– метод ідентифікації: особа, пароль, RFID-карти;
– час ідентифікації: менше 0,5 сек;
– рівень помилкових підтверджень: менше 0,1%;
– інтерфейси: TCP / IP, RS485, USB Host, Wi-Fi;
– модуль читання карт: EM-Marine 125кГц;
– сектор захоплення зображення: по горизонталі: ±20°, по вертикалі ±20°;
– вбудований web-сервер;
– відсоток розпізнавання: більше 99%;
– відстань для ідентифікації користувача: від 30 до 80 см;
– голосовий супровід подій;
– живлення: 12В;
– режим збереження енергії;
– габарити: 124мм х 155мм х 92 мм. [2]

Управляти Anviz FacePass 7 можна віддалено через мережу за допомогою програмного забезпечення – CrossChex (рисунок 1.4).

Це інтелектуальна система управління пристроями контролю доступу та обліку робочого часу, яка застосовується до всіх засобів Anviz з контролю доступу та відвідування робочого часу.

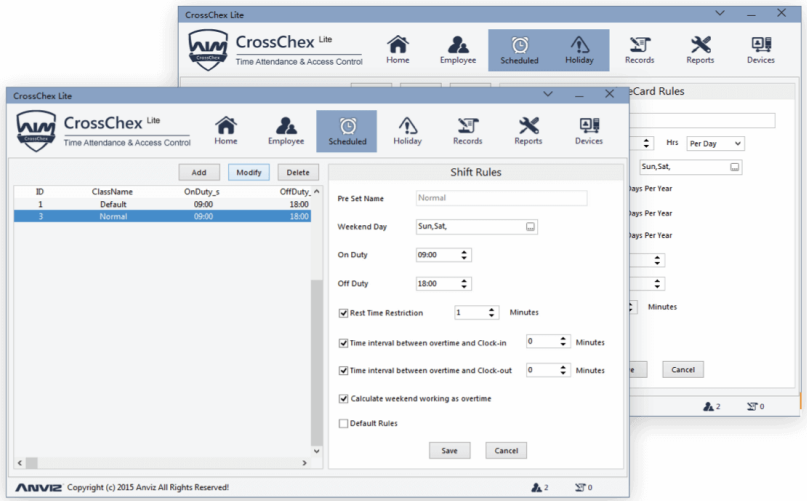


Рисунок 1.4 – Програмне забезпечення CrossChex

Інтерактивний дизайн забезпечує простоту системи в експлуатації, потужний набір функціоналу надає можливість реалізовувати управління відділом, персоналом, керувати зміною графіку, управляти доступом, експортувати різні звіти про відвідування і робочий часу, задовольняючи цим різні вимоги щодо обліку відвідуваності та контролю доступу. CrossChex застосовується до всіх пристроїв контролю відвідуваності та доступу Anviz.

Користувачі можуть отримувати доступ до своїх облікових даних та керувати ними через хмару і навіть завантажувати своє посвідчення особи зі свого мобільного телефону. Хмарний сервіс Anviz також дозволяє створювати звіти про відвідуваність, а також виконувати ключові функції управління, використовуючи новий, інтуїтивно зрозумілий та спрощений інтерфейс користувача. Як і у будь-якій хмарній службі, ці функції можна виконувати з будь-якої точки світу, якщо доступне мережне підключення.

1.2.3 Біометричний термінал Hikvision MinMoe

Hangzhou Hikvision Digital Technology Corporation (часто скорочується до Hikvision) є китайським, частково державним, виробником та постачальником обладнання відеоспостереження для цивільних та військових цілей.

Термінал MinMoe (рисунок 1.5) використовується для того, щоб дати людям простіший спосіб увійти в будівлю, зафіксовувати відвідуваність і одночасно

перевірити температуру і носіння маски без необхідності торкатися до терміналу, а шляхом застосвання розпізнавання осіб і термографічних технології.

Рисунок 1.5 – Термінал MinMoe

Основні характеристики:

- протокол зв'язку RS485;
- вбудований кард-рідер;
- допоміжний зчитувач Wiegand;
- розпізнавання облич: 1 канал;
- ємність журналу подій: 100тис.;
- кількість користувачів в пам'яті пристрою: 3тис.;
- відстань розпізнавання осіб 0,3-1,8м;
- швидкість розпізнавання осіб: < 0,2 мс;
- дисплей та вбудована камера;
- мережеве з'єднання TCP / IP;
- джерело живлення: 12В постійного струму;
- аварійний сигнал про ненормальну температуру. [3]

1.3 Постановка завдання

Ручне ведення обліку присутності достатньо клопіткий процес, складний для подальшої обробки та не відображає дані у зручному для користувача форматі

– у вигляді графіків, таблиць, діаграм чи звітів. В сучасному технологічному світі є безліч можливостей і способів реалізувати автоматизовану систему, яка забезпечить ефективніший та більш зручний облік присутності.

Електронна система для управління відвідуваністю повинна бути простою та зрозумілою для користувача і водночас забезпечити всю функціональність, необхідною для обліку та фіксації відвідуваності та робочого часу.

Система повинна бути реалізована у вигляді веб-додатку та програми, яка здійснюватиме розпізнавання обличчя і позначатиме присутність працівника. Веб-застосунок забезпечуватиме реєстрацію користувача, відображення особистого кабінету, дасть можливість переглянути уже наявні записи.

Розширений функціонал повинен бути наданий адміністратору сайту. Тобто, потрібно реалізувати розподіл користувачів та їх можливостей. В адміністратора повинні бути можливості маніпуляції з даними, а саме: додавання користувача, редагування його інформації, зміна записів відвідуваності і т.д.

Основними задачами дипломної роботи є:

– проаналізувати технології та засоби розробки системи;

– проаналізувати алгоритм розпізнавання, на якому буде базуватись система;

– виконати реалізацію алгоритму детекції і розпізнавання облич;

– розробити веб-інтерфейс;

– дослідити ефективність та протестувати роботу системи;

– дослідити економічну складову даного рішення.

Ключовим аспектом та першорядною особливістю системи має стати технологія розпізнавання обличчя.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Формалізація вимог до системи

Специфікація вимог до програмного забезпечення – це розгорнутий опис поведінки системи, що перебуває в розробці. До специфікація належить сукупність прецедентів, що описують всі взаємодії, які користувачі мають з програмним забезпеченням.

Прецеденти – це технологія документування потенційних вимог до програмної системи. Кожен прецедент надає один чи більше сценаріїв. Головна ціль такої діаграми – продемонструвати, яким чином система взаємодіє з користувачем або іншою системою для досягнення конкретної мети. Прецеденти також відомі як функціональні вимоги, які формуються відповідно до технічних платформ, операційних систем, апаратного забезпечення системи.

Система обліку відвідуваності працівників коледжу містить у собі двох головних користувачів (акторів) – «Адміністратор» та «Працівник». Діаграма варіантів використання зображена на рисунку 2.1.

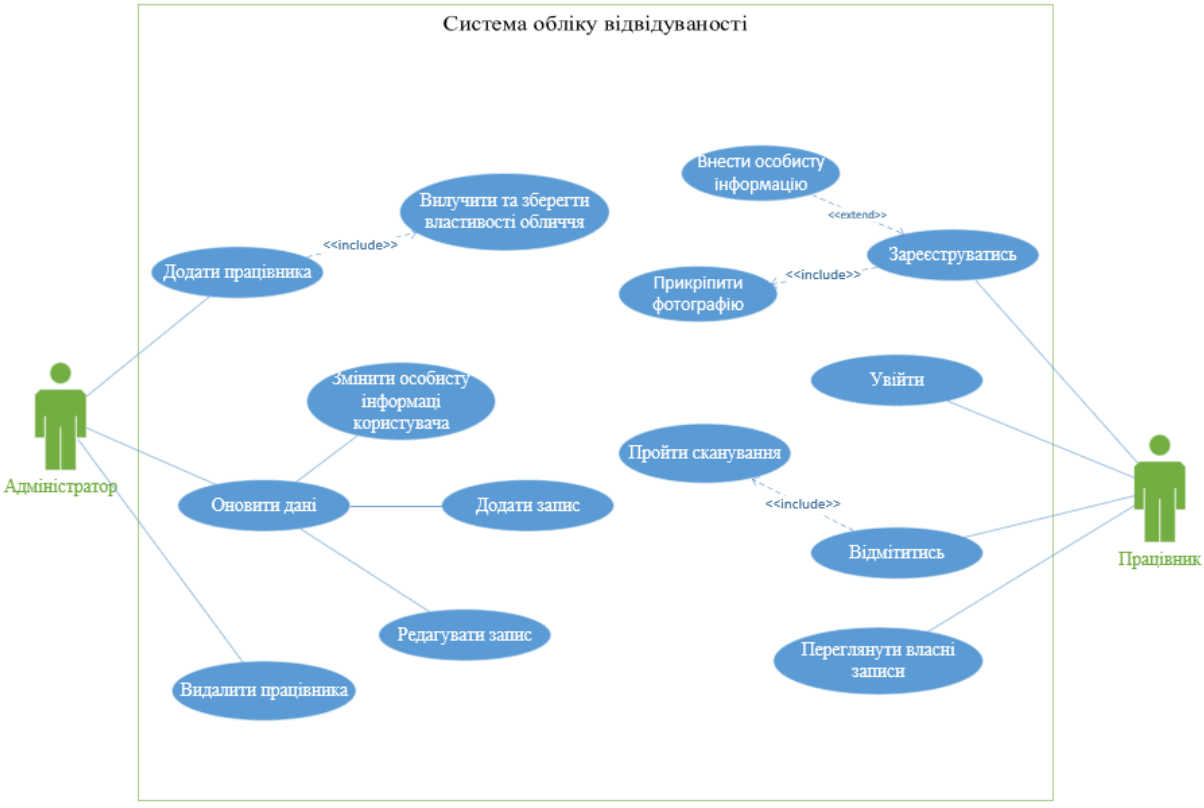


Рисунок 2.1 – Діаграма варіантів використання

До основних функцій Адміністратора належать:

– додати користувача (тобто, він може реєструвати працівника);

– оновити дані відвідуваності (редагування та видалення записів);

– редагувати особисті дані користувача;

– видалити користувача;

– можливість обробки обличчя та додавання даних в БД.

Функціями Працівника є:

– реєстрація;

– вхід;

– здійснення відмітки;

– внесення персональних даних;

– перегляд особистої відвідуваності

– обробка обличчя, додавання даних в базу даних.

2.1.1 Вимоги до програмного продукту

Кожне програмне забезпечення має свої характеристики та особливості. Проте, серед них існують такі, що є практично невід'ємними, а саме:

– Надійність. Забезпечення надійності передбачає отримання відповідей на наступні дві групи питань: чи здатний програмний продукт задовольнити висунуті вимоги до нього; якщо це програма, то чи досягається необхідна точність; при функціонуванні в реальних умовах чи продовжує програма працювати правильно з поточними даними, чи істотно відрізняються від тестових; як багато буде виявлено прихованих помилок після атестації програми як працездатної; яка ймовірність того, що результати будуть містити не виявлені помилки.

– Доступність. Програмний продукт має властивість доступності, якщо він допускає селективне використання окремих його компонентів.

– Модифікованість. ПЗ має структуру, що дозволяє легко вносити необхідні зміни.

– Комунікативність. Програмний продукт має давати можливість легко описувати вхідні дані і видає інформацію, форма і зміст якої прості для розуміння і несуть корисні відомості.

– Структурованість. Взаємопов'язані частини організовані в єдине ціле певним чином. Структурованість програми може мати в своїй основі найрізноманітніші причини. Наприклад, вона може бути розроблена відповідно до спеціальних стандартів або вона може бути написана з використанням мови структурного програмування, чи відображатиме в своїй структурі процес поступового еволюційного розвитку на основі цілеспрямованих і систематизованих змін.

– Ефективність. Програмний продукт має властивість ефективності, якщо він виконує необхідні функції без зайвих витрат ресурсів і часу. Термін «ресурси» тут розуміється в широкому сенсі: це може бути оперативна пам'ять, загальна кількість виконуваних команд на одну ітерацію розв'язуваної задачі, зовнішня пам'ять, пропускна здатність каналу і т. д.

2.1.2 Вимоги до інтерфейсу користувача

Користувальницький інтерфейс (графічний інтерфейс користувача) – це комплекс засобів для взаємодії користувача з технічною системою (в тому числі з програмними додатками, мультимедійним даними).

У поняття інтерфейсу користувача комп'ютерної системи входять наступні складові:

– графічне середовище – зображення на екрані;

– набір керуючих елементів призначених для інтерфейсу і їх розташування на екрані;

– технології взаємодії користувача з системою.

Керуючі елементи інтерфейсу користувача – це графічні елементи (кнопки, списки, діалогові вікна тощо), які дозволяють здійснювати будь-які дії з комп'ютерною системою (наприклад, заповнювати поля, вибирати пункти і властивості об'єктів).

Основні вимоги до інтерфейсу користувача:

– функціональність (відповідність завданням користувача);
– відповідність технології;
– швидка адаптованість користувача в системі;
– задоволення користувача;
– забезпечення захисту від людських помилок;
– зрозумілість і логічність.

2.1.3 Вимоги до бази даних

Проєктування бази даних – це процес створення проєкту бази даних, призначеної для підтримки функціонування об'єкта і сприяє досягненню його цілей. Воно являє собою трудомісткий процес, що вимагає спільних зусиль аналітиків, проєктувальників і користувачів. При проєктуванні бази даних необхідно враховувати той факт, що база даних повинна задовольняти комплексу вимог. Ці вимоги наступні:

– цілісність даних;
– багаторазове використання даних;
– швидкий пошук і отримання інформації за запитами користувачів;
– простота оновлення даних;
– зменшення надлишковості даних;
– захист від несанкціонованого доступу, спотворення чи знищення.

2.2 Проєктування структури системи

В загальному структуру даної системи можна поділити на дві частини. Оскільки користувач повинен взаємодіяти якимось чином із системою, тобто через певний інтерфейс, тому до першої частини буде відноситися веб-застосунок. Друга частина – це модуль розпізнавання та фіксації присутності.

Веб-застосунок включатиме у себе значну кількість модулів. До основних з яких належать: модуль реєстрації, авторизації, перегляду та редагування даних. Дані функціональні блоки напряму взаємодіють із базою даних. Модуль

реєстрації чи авторизації (що є наслідком реєстрації) є точкою входу користувача в систему.

Модуль позначення відвідуваності є доцільним використовувати лише зареєстрованим в системі користувачам. Щоб відзначити свою присутність необхідно натиснути на відповідну кнопку і розмістити своє обличчя в полі зору камери. На рисунку 2.2 продемонстровано процес входу в систему та здійснення відмітки за допомогою методології моделювання IDEF3.

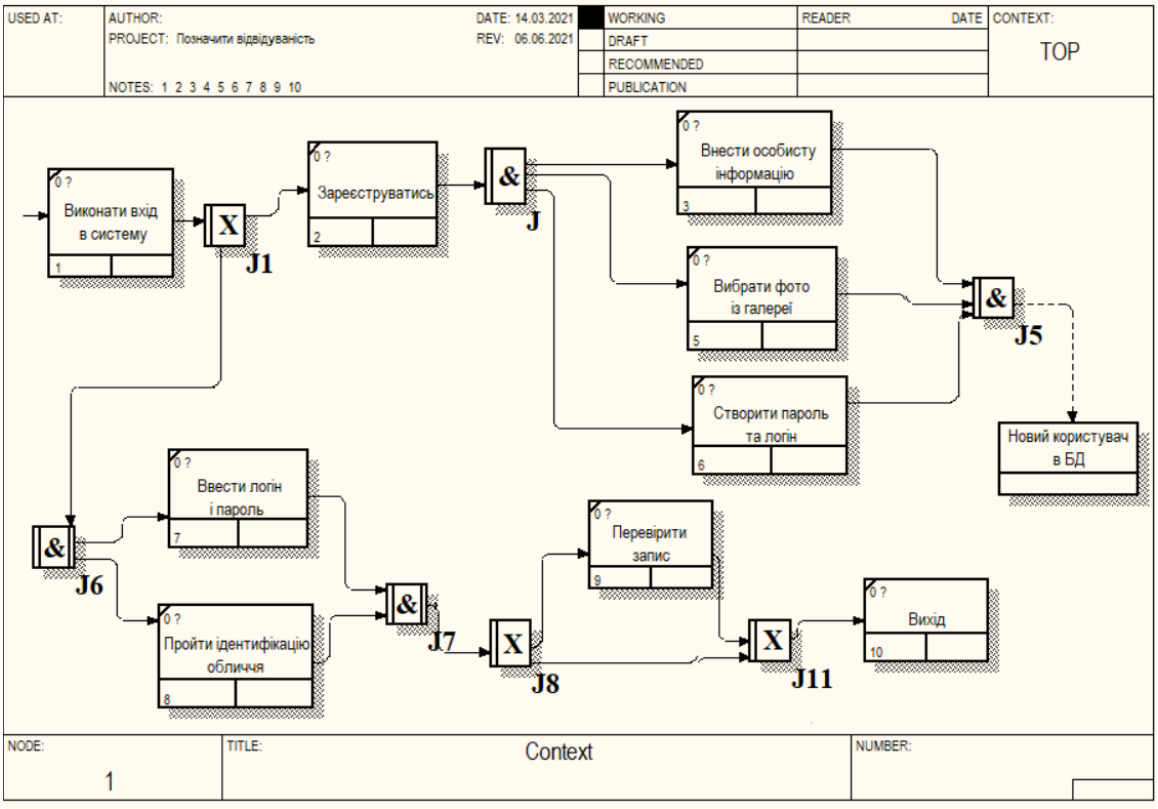


Рисунок 2.2 – Процес позначення присутності

Також користувач матиме можливість переглянути дані відвідуваності. Подання цієї інформація буде у вигляді структурованих таблиць та інших візуальних елементів. Цей модуль буде доступним як для користувача так і для адміністратора.

Як уже було сказано, до другої частини належать модулі розпізнавання та ідентифікації обличчя. Їх можна поділити на:

– Модуль визначення розташування обличчя (на базі HOG-алгоритму). HOG (histogram of oriented gradients) – це дескриптор ознак, який

використовується в комп'ютерному зорі і обробці зображень з метою виявлення об'єктів. В основі даного дескриптора лежить алгоритм визначення вектора затемненості певного квадрата пікселів (градієнт). Необхідно визначити наскільки темним являється даний піксель в порівнянні із тими, які його оточують. Пройшовшись так по кожному пікселі зображення та визначивши градієнти ми отримаємо нове зображення - базову структуру обличчя. Для підтвердження наявності обличчя виконують його порівняння із HOG-шаблоном, що згенерований із облич багатьох людей. [4]

– Модуль вилучення даних з зображення – 128 вимірів рис обличчя. Для ідентифікації конкретної особи по обличчю необхідно отримати ті дані, які будуть відрізняти його від інших. Виконується обчислення вилучення за різними параметрами обличчя. Пропускаємо наше зображення через даний модуль і отримуємо 128 унікальних значень.

– Порівняння облич. Він виконує порівняння 128 значень поточного обличчя із тими, які збережені в базі даних. На виході ми дізнаємось, хто з відомих нам людей найбільше похожий на дану особу.

2.3 Проєктування користувацького інтерфейсу

Хороший користувацький інтерфейс є важливим аспектом для будь-якого проєкту. Він дає можливість користувачам взаємодіяти з системою, чітко окреслювати її функціонал та доступні операції.

Дизайн інтерфейсу користувача – це основні будівельні блоки того, як сайт чи веб-застосунок налаштований і функціонує. Інтерфейс може складатись з багатьох елементів. Деякі з них, що беруть участь у розробці інтерфейсу користувача, включаючи елементи введення, навігаційні компоненти, інформаційні компоненти, контейнери, кнопки, списки, перемикачі, значки та багато іншого.

Інтерфейсом системи стануть вікна python-програм та веб-застосунок. Основна їх ціль – ведення ефективного обліку відвідуваності, фіксація

присутності працівника, забезпечення зручного моніторингу відвідування персоналу.

Вікна програм на Python мають простий інтерфейс – містять декілька кнопок та полів для введення.

Основними сторінками сайту будуть:

– сторінка з формою реєстрації/входу;

– сторінка профілю – головна сторінка користувача, містить його особисті дані;

– інформаційна сторінка з даними присутності працівника - можна переглянути записи відвідувань;

– сторінка із переліком усіх працівників – доступ до неї матиме адміністратор, який зможе переглядати, редагувати та вилучати працівників із системи;

– сторінка з інформацією про відвідуваність працівників – також сторінка для адміністратора, надаватиме функції для редагування записів фіксацій присутності.

На рисунку 2.3 продемонстровано макет веб-додатку.

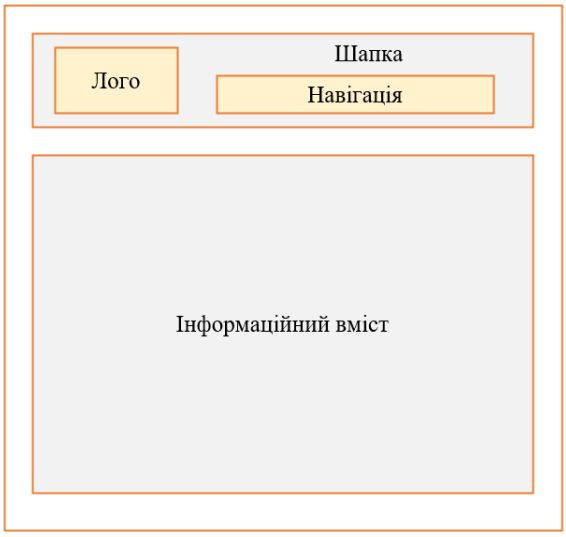


Рисунок 2.3 – Макет веб-додатку

Базовими елементами сторінки є логотип, навігація та інформаційний вміст.

2.4 Проєктування бази даних

Основна роль бази даних полягає в збереженні інформації та її подальшого застосування при розробці додатку. Веб-сайти з реєстрацією, дискусійні форуми та сайти роздрібної торгівлі є прикладами веб-додатків, які залежать від надійного компонента бази даних.

Додатки баз даних використовуються для пошуку, сортування, фільтрації та подання інформації на основі запитів користувачів. Бази даних також можуть містити код для виконання математичних і статистичних обчислень даних для підтримки запитів, що відправляються з веб-браузерів.

Бази даних надають і обмежують доступ до даних на основі таких критеріїв, як ім'я користувача, пароль, регіон або номер облікового запису. Бази даних також забезпечують цілісність даних, гарантуючи, що дані збираються і представляються в узгодженому форматі.

Динамічний веб-сайт відображає оновлену інформацію на веб-сторінках, коли база даних редагується хостом або коли користувачі відправляють інформацію за допомогою веб-форм. Програмний код, шляхом виконання SQL-запитів, може маніпулювати цими даними.

Опишемо предметну область: працівники навчального закладу, присутність яких повинна бути зафіксована. Дані із записами відвідуваності включають: ідентифікатор працівника (для подальшого отримання його ПІП чи інших даних), дату, статус прибуття, відходу, час фіксації. Список працівників містить їх прізвище, ім’я, по батькові, фото та пароль входу в систему. Додатковими даними працівника є: номер телефону, стать, посада, а також зображення та дані визначені із обличчя – розташування обличчя, його кодування.

Опис сутностей, атрибутів і зв’язків між даними представляє модель даних. При створенні моделі даних використовують метод семантичного моделювання. Суть даної техніки полягає у визначенні сенсу даних по відношенню з іншими даними. В якості інструмента семантичного моделювання використовуються ER-діаграми (Entity-Relationship).

Дану нотацію запропонував тайванський інформатик П.Чен. Діаграма «сутність-зв'язок» розроблена для візуального представлення моделей даних створюваної системи. ERD включає певний набір позначень – різні символи і з'єднувачі, які візуалізують дві важливі інформації: основні сутності в області системи і взаємозв'язку між цими сутностями [5].

Ключовими поняттями нотації є «сутність» і «зв'язок». Під «сутністю» розуміється довільна множина реальних або абстрактних об'єктів. Цей термін часто використовується замість «таблиця», що є одним і тим же. Зв’язки описують відношення, що може бути створене між таблицями. У ER-моделях сутність відображається у вигляді закругленого прямокутника з її іменем зверху і атрибутами, перерахованими в тілі форми сутності. Для даної системи ER-діаграма зображена на рисунку 2.4.

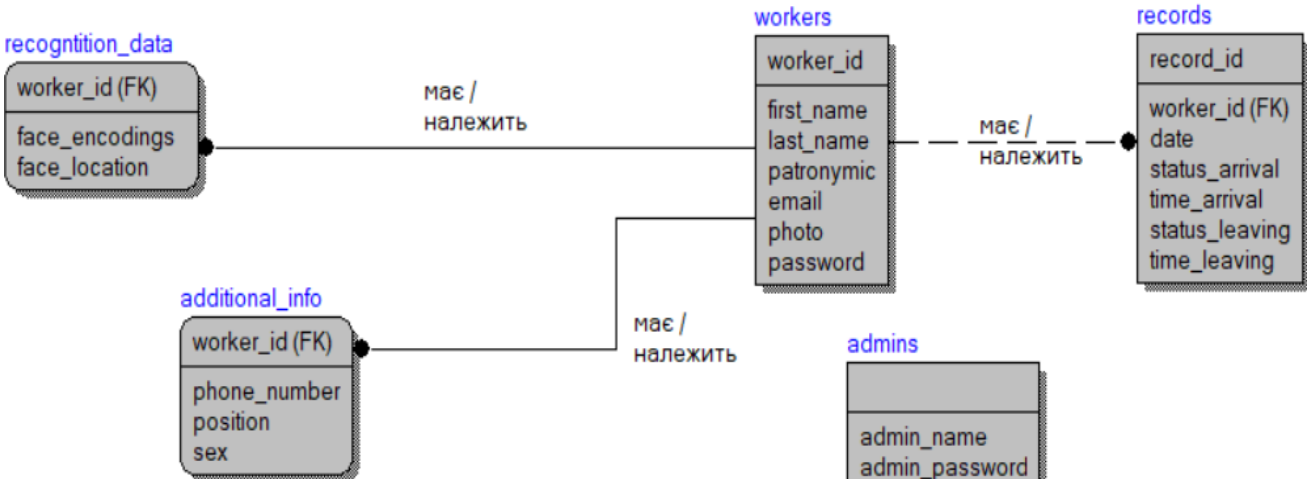


Рисунок 2.4 – ER-діаграма системи

Таблиця records (таблиця 2.1) призначена для збереження кінцевих даних роботи системи – записи відвідуваності. Вона включає поле worker_id, що дозволяє ідентифікувати працівника, date – час відмічання, status_arrival та status_leaving – статус приходу та відходу, time_arrival і time_leaving – час прибуття та відходу.

Таблиця 2.1 – Структура таблиці records

Атрибути сутності	Обмеження атрибутів			
	Тип даних	Значення Null	Значення за замовчуванням	Ключ
record_id	int(11)	false	-	P
worker_id	int(11)	false	-	F
date	date	false	-	-
status_arrival	tinyint(1)	false	0	-
time_arrival	time	false	-	-
status_leaving	tinyint(1)	false	0	-
time_leaving	time	false	-	-

Таблиця workers, структура якої подана у таблиці 2.2, призначена для збереження основної інформації про працівника.

Таблиця 2.2 – Структура таблиці workers

Атрибути сутності	Обмеження атрибутів			
	Тип даних	Значення Null	Значення за замовчуванням	Ключ
worker_id	int(11)	false	-	P
first_name	varchar(100)	false	-	-
last_name	varchar(100)	true	-	-
patronymic	varchar(100)	true	-	-
email	varchar(100)	false	-	-
photo	mediumblob	false	-	-
password	varchar(300)	false	-	-

Таблиця recognition_data (таблиця 2.3) використовується для збереження унікальних параметрів обличчя та його координат на зображенні.

Таблиця 2.3 – Структура таблиці recognition_data

Атрибути сутності	Обмеження атрибутів			
	Тип даних	Значення Null	Значення за замовчуванням	Ключ
worker_id	int(11)	false	-	F
face_encodings	longtext	true	-	-
face_location	varchat(50)	true	-	-

Additional_info – таблиця, яка виступає доповненням для таблиці workers. Її основне призначення – зберігати додаткову інформацію про працівника. Структура даної таблиці подана в таблиці 2.4.

Таблиця 2.4 – Структура таблиці additional_info

Атрибути сутності	Обмеження атрибутів			
	Тип даних	Значення Null	Значення за замовчуванням	Ключ
worker_id	int(11)	false	-	F
phone_number	varchar(20)	true	-	-
position	varchar(100)	true	-	-
sex	varchar(20)	true	-	-

Таблиця admins – створена для збереження даних адміністратора. Її структура подана у таблиці 2.5.

Таблиця 2.5 – Структура таблиці admins

Атрибути сутності	Обмеження атрибутів			
	Тип даних	Значення Null	Значення за замовчуванням	Ключ
admin_name	varchar(100)	false	-	-
admin_password	varchar(300)	false	-	-

Для відображення предметної області створюваної системи існує концептуальна модель. Вона зображена на рисунку 2.5.

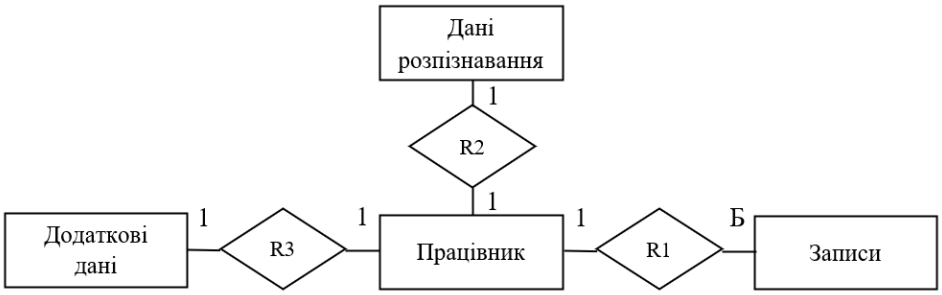


Рисунок 2.5 – Концептуальна інформаційна модель

Для кращого розуміння утворених сутностей, в таблиці 2.6 детально описано зв'язки між сутностями.

Таблиця 2.6 – Опис зв’язків між сутностями

Зв’язки між сутностями	Назва та код зв’язку	Тип зв’язку	Зміст зв’язку
Працівник – Записи	«Має» R1	1:Б	Один працівник має багато записів відвідуваності; багато записів відвідуваності належать одному працівнику
Працівник – Дані розпізнавання	«Має» R2	1:1	Один працівник має одні власні дані розпізнавання; дані розпізнавання належать тільки одному працівнику
Працівник – Додаткові дані	«Має» R3	1:1	Один працівник має одні власні додаткові дані; додаткові дані належать тільки одному працівнику

2.5 Алгоритм функціонування системи

Алгоритм функціонування системи залежить від кінцевої цілі користувача. Розглянемо основні варіанти використання застосунку користувачем:

– зареєструватись/увійти. Щоб здійснити реєстрацію/вхід потрібно зайти на веб-застосунок і заповнити усі реєстраційні поля чи поля для входу;

– відмітитись – ключова функція системи. Для того, щоб скористатись нею необхідно на запустити програму mark-presence.exe та натиснути на кнопку «Прибув» (чи «Покидаю») та направити своє обличчя на камеру. Ця процедура запустить алгоритм ідентифікації особи. Демонстрація цього алгоритму подана на рисунку 2.6;

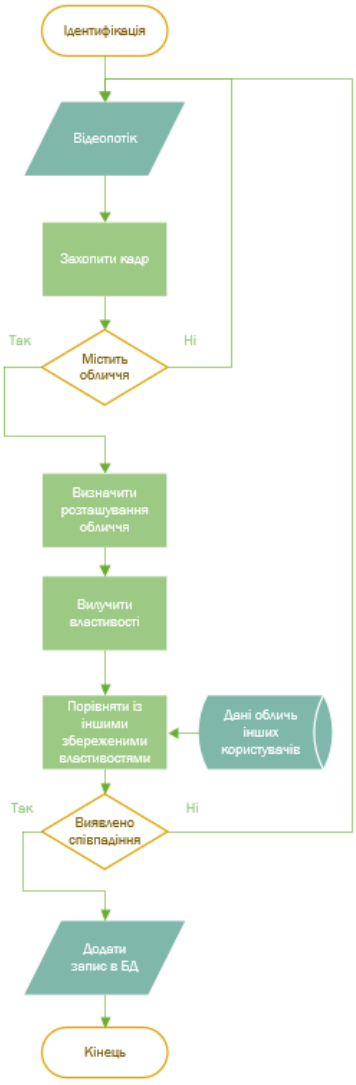


Рисунок 2.6 – Алгоритмічна модель ідентифікації

– перегляд особистих записів. Щоб здійснити огляд особистих записів потрібно увійти в систему та вибрати відповідний пункт меню;

– додати, редагувати чи видалити дані – базові функції адміністратора. Для доступу до цієї панелі необхідно мати спеціальний логін та пароль, за допомогою якого вхід відбуватиметься не як для простого користувача та, натисненням відповідних кнопок, виконувати певні операції.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Опис технологій та засобів реалізації.

Розробка систем, пов’язаних із розпізнаванням обличчям можна реалізувати на багатьох мовах програмування. Коли ми говоримо про розпізнавання, особлива увага приділяється мовам C++ і Python. Кожна з них має доступні бібліотеки, що допоможуть виконати вищенаведені завдання. Говорячи про переваги та недоліки, то C++ виділяється високою швидкістю обробки даних, що напряму впливає на швидкість. Проте, для початківців в даній області C++ є достатньо складною мовою. Абсолютно ніщо не зрівняється з бібліотечною екосистемою Python. Існує безліч рішень, які легко інтегруються в проєкт. Крім того, значна частка бібліотек оптимізована так, що знаходиться практично на одному рівні по швидкості з аналогом на мові C++, а відмінна система типів Python виконає для вас значний обсяг низькорівневої роботи, що значно полегшить завдання.

За останні роки Python не покидає список найпопулярніших мов програмування. Це мова програмування загального призначення і високого рівня. Python можна використовувати для розробки настільних графічних додатків, веб-сайтів чи веб-додатків. Зрозумілі правила синтаксису мови програмування також полегшують читання коду та підтримку додатків [6]. Існує ряд переваг Python, до найважливіших з яких можна віднести наступні:

– Читання та обслуговування коду. Синтаксис Python ясний і лаконічний. Мова розроблена таким чином, що читається та розуміється на рівні з англійською мовою. Це значно полегшує розуміння коду. Python також вимагає менше рядків коду для вирішення задачі у порівнянні з такими мовами, як C++ або Java. Маючи код, який легко розуміється і в якому легко орієнтуватись, ви можна зменшити обсяг роботи для підтримки і розширення бази коду.

– Можливість застосування безлічі парадигм програмування. Як і інші сучасні мови програмування, Python підтримує кілька парадигм програмування. Він повністю підтримує об'єктно-орієнтоване та структуроване програмування. Крім того, його мовні функції підтримують різні концепції у функціональному та

аспектно-орієнтованому програмуванні. У той же час Python також має систему динамічних типів і автоматичне управління пам'яттю.

– Наявність інструментів та фреймворків з відкритим кодом. Ще одна сильна сторона Python полягає в тому, що в ньому є багато рішень, які спрощують процес розробки.

– Мобільність та інтерактивність. Python може бути легко вбудований в широкий спектр додатків, навіть тих, які використовують різні мови програмування. Можна легко виправити нові модулі та розширити основний словниковий запас Python.

– Використання в машинному навчанні та технологій, пов’язаних із штучним інтелектом. Машинне навчання і штучний інтелект привертають все більше уваги, тому все більше розробників намагаються включити їх в різні проєкти. Широке коло розробників вважають Python кращою мовою для проєктів у даних галузях. Машинне навчання – це дуже складна процедура, що дозволяє отримувати інформацію з високою швидкістю. Python підтримує безліч інноваційних проєктів, які використовують можливості МН.

Для реалізації рішення на Python було надано перевагу та використано наступні бібліотеки: OpenCV, face_recognition, numpy.

В області штучного інтелекту комп'ютерний зір є одним з найцікавіших і складних завдань. Він діє як проміжна ланка між програмним забезпеченням комп'ютера та тим, що оточує нас. В даний час існують різні бібліотеки для виконання завдань обробки зображень і комп'ютерного бачення. OpenCV – це бібліотека з відкритим кодом, що націлена на вирішення питань комп’ютерного зору. Вона підтримується різними мовами програмування, такими як Python, C++, C та працює на більшості платформ, таких як Windows, Linux і macOS [7].

Комп'ютерне бачення – це процес, за допомогою якого можна зрозуміти зображення та відео, як вони зберігаються та як ми можемо маніпулювати та витягувати дані з них. З допомогою OpenCV можна обробляти зображення і відео, щоб ідентифікувати об'єкти, почерк людини або навіть осіб. Для ідентифікації

малюнка зображення і його різних ознак ми виконуємо математичні операції над ними.

Існує безліч задач, які вирішуються за допомогою OpenCV, деякі з них перераховані нижче:

– розпізнавання осіб;

– автоматизований контроль і спостереження;

– підрахунок кількості людей;

– інтерактивні художні інсталяції;

– виявлення дефектів в процесі виробництва;

– розпізнавання об'єктів;

– аналіз медичних зображень;

– побудова 3D-моделей об'єктів;

– аналіз зображення.

Функціональність OpenCV:

– введення / виведення зображень/відео;

– обробка зображень/відео;

– виявлення об'єктів;

– машинне навчання, кластеризація і т.д.

Розпізнавання осіб – це метод ідентифікації або перевірки особистості людини по його обличчю. Існують різні алгоритми, які можуть розпізнавати обличчя, але їх точність може відрізнятися [8]. Процес розпізнавання поділяється на три основних кроки:

– виявлення обличчя. Найперше завдання, яке ми виконуємо – це виявлення обличчя на зображенні або відео. Тепер, коли ми знаємо точне місце розташування/координати обличчя, ми витягуємо цей фрагмент для подальшої обробки;

– вилучення об'єктів. Нейронна мережа приймає зображення обличчя людини в якості вхідних даних і виводить вектор, який представляє найбільш важливі риси обличчя;

– порівняння обличчя: після обчислення властивостей обличчя осіб ми можемо порівняти з іншими, які у нас є.

Однією із найпростіших бібліотек для реалізації задач з виявленням обличчя, вилученням об'єктів, порівнянням облич є face_recognition. Вона побудована з використанням сучасної бібліотеки, що тісно пов’язана з глибоким навчанням – dlib. Face_recognition розроблена таким чином, щоб якомога полегшити вирішення поставленої розробником задачі. Функціонал даної бібліотеки забезпечує:

– знаходження декількох облич на зображенні;

– ідентифікацію облич;

– відстеження облич в реальному часі;

– знаходження та маніпуляцію особливостями обличчя;

– точність розпізнавання до 99,38%.

NumPy - це також бібліотека Python, що створена для здійснення операцій над великими масивами. Вона містить обширний набір математичних функцій для обробки цих масивів. Також, це одна із бібліотек, що застосовується в OpenCV. Основним завданням NumPy є вирішення питання швидкості обчислення математичних алгоритмів [9]. Так як Python є інтерпретованою мовою, то дані операції працюють повільніше, інколи навіть в декілька разів ніж у таких мовах як C чи Java, які є компільованими. NumPy вирішує цю проблему, оптимізуючи та створюючи багато функцій і операторів для більш ефективного обчислення. Numpy набирає популярність і використовується в ряді складних систем.

Основні відомості про NumPy:

– це числова бібліотека Python з відкритим вихідним кодом;

– NumPy містить багатовимірні масиви і матричну структуру даних;

– може бути використана для виконання ряду математичних операцій з масивами, таких як тригонометричні, статистичні та алгебраїчні процедури;

– містить велику кількість математичних, алгебраїчних і функцій перетворення;

– NumPy – це розширення бібліотек Numeric і Numarray;

– містить генератори випадкових чисел;

– являє собою оболонку навколо бібліотеки, реалізованої в C.

Для створення простого графічного інтерфейсу в проєкті використано Tkinter – пакет для Python, створений для роботи з бібліотекою Tk. Компоненти графічного інтерфейсу користувача (GUI) бібліотеки Tk реалізовані на мові програмування Tcl. Під елементами графічного інтерфейсу користувача маються на увазі кнопки, текстові поля для введення, перемикачі, радіокнопки, перемикачі, списки, вікна, та ін. Через них можна управляти та взаємодіяти з програмою. Віджетами називають сукупність усіх цих елементів інтерфейсу. В Python за допомогою бібліотеки Tkinter можна створювати віджети [10].

Етапи розробки інтерфейсу з GUI наступні:

– імпорт бібліотеки;

– створення головного вікна;

– створення віджетів;

– установка їх властивостей;

– визначення подій;

– визначення обробників подій;

– розташування віджетів на головному вікні;

– відображення головного вікна.

Tkinter надає різні елементи управління, такі як кнопки, мітки і текстові поля, використовувані в додатку з графічним інтерфейсом. Ці елементи керування, зазвичай, називаються віджетами.

До основних віджетів Tkinter належить:

– Button (кнопка) – використовується для створення кнопок.

– Checkbutton (прапорець) – відображення ряду параметрів у вигляді прапорців.

– Entry (поле введення) – відображення однорядкового текстового поля для прийому значень від користувача.

– Frame (рамка) – використовується в якості віджета-контейнера для організації інших віджетів.

– Label (надпис) – надання однорядкового заголовка для інших віджетів.

– Listbox (список) – віджет Listbox використовується для надання користувачеві списку опцій.

– Message (повідомлення) – для відображення текстових полів для прийому значень від користувача.

– Radiobutton (перемикач) – відображення ряду параметрів у вигляді перемикачів, де користувач може вибрати тільки один варіант за раз.

– Text (текст) – використовується для відображення тексту в декількох рядках.

– Toplevel (верхній рівень) – віджет верхнього рівня, використовується для надання окремого контейнера вікна.

PHP є мовою сценаріїв на стороні сервера та використовується для розробки статичних або динамічних веб-сайтів. PHP-код інтерпретується веб-сервером та передається у браузер як HTML-код [11]. Дана мова програмування є чудовим варіантом з багатьох причин:

– швидкий час завантаження – PHP забезпечує високу швидкість завантаження сайту;

– характеризується гнучкістю, ефективністю, простотою та безпекою;

– менш дороге програмне забезпечення – при роботі з PHP більшість інструментів, пов'язаних з розробкою, є програмним забезпеченням з відкритим кодом, тому не потрібно платити за них;

– гнучкість баз даних – PHP гнучка у питанні підключення до бази даних. Дана мова може підключатися до декількох баз даних, але найбільш часто використовується MySQL. MySQL можна також використовувати безкоштовно;

– PHP має дуже хорошу онлайн-документацію з хорошим набором функцій. Це робить мову відносно легкою для вивчення і дуже добре підтримуваною в Інтернеті. Існує безліч форумів і навчальних посібників з різних

методів і проблем PHP, тому зазвичай дуже легко знайти допомогу, якщо вона потрібна [12].

jQuery – це популярна багатофункціональна бібліотека JavaScript. Вона створена з метою полегшити написання коду на JavaScript. Основні можливості бібліотеки:

– вибір DOM: jQuery надає зручні методи пошуку та вилучення елемента елементів DOM шляхом задання різних параметрів, таких як ідентифікатор, ім'я класу, атрибут тегу, вказуючи номер дочірнього елемента і т.д;

– маніпуляція DOM: можна додавати елементи, видаляти, замінювати, задавати їм нове значення чи вкладати в них HTML, виконувати маніпуляції з класами CSS і багато іншого;

– події: за допомогою бібліотеки jQuery, можна використовувати функції, які є аналогічними подіям DOM - click, dblclick, hover, mouseen, mouseleave, toogle, unload, submit, keyup, keydown і т.д;

– AJAX: jQuery також використовується для отримання даних з серверів. Підхід AJAX забезпечує оновлення даних без перезавантаження усієї сторінки;

– кросбраузерність: бібліотека jQuery працює на різних браузерах. Правильна робота модулів jQuery забезпечується у будь-якому браузері, наприклад, Chrome, Opera, Internet Explorer, Mozilla, Safari [13].

Системою управління реляційними базами даних було обрано MySQL. Це вільна, швидка, стійка та одна з найвикористовуваніших СУБД в розробці сайтів. MySQL є швидкою, менш складною в налаштуванні та адмініструванні в порівнянні з іншими, її сервери є багатопотоковими, тому багато користувачів можуть підключитись до неї одночасно. Крім того, інтерфейси програмування доступні для багатьох мов, таких як C, Perl, Java, PHP, Python і Ruby. Також можна отримати доступ до MySQL за допомогою програм, що підтримують ODBC (Open Database Connectivity), протокол зв'язку з базами даних, розроблений корпорацією Майкрософт [14]. Для забезпечення додаткової безпеки MySQL підтримує зашифровані з'єднання з використанням протоколу Secure Sockets Layer. Дана

СУБД має скромний розмір дистрибутива, особливо в порівнянні з величезним об'ємом дискового простору деяких комерційних систем баз даних. Це надійна, відкрита та масштабована СУБД. MySQL може працювати на різних платформах UNIX, Linux, Windows і т.д. MySQL можна встановити на сервер або навіть на ПК [15].

3.2 Реалізація користувацького інтерфейсу та основних функцій системи

Для того, щоб розробляти динамічні сайти потрібно на локальному комп'ютері встановити спеціальну програму-сервер. Такою являється програма XAMPP. Це кросплатформна збірка веб-серверів, яка містить Apache, MySQL, інтерпретатор PHP і велику кількість інших бібліотек. Насамперед потрібно скачати інсталятор. Після його запуску відкриється початкове вікно встановлення. Далі, відображається вікно, яке дозволяє вибрати потрібні компоненти для встановлення (рисунок 3.1).

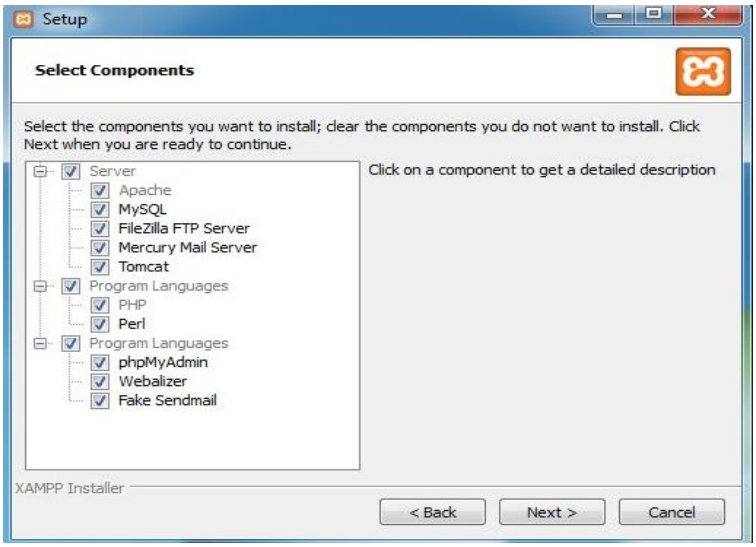


Рисунок 3.1 – Вибір компонентів установки

Також необхідно вказати шлях установки сервера. За замовчуванням сервер ставиться на локальний диск C в папці «xampp» (рисунок 3.2).

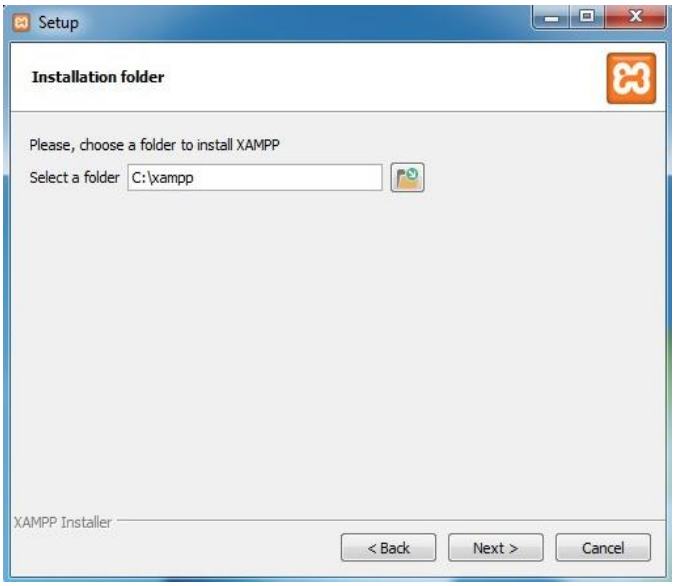


Рисунок 3.2 – Екран вибору шляху установки

Насамкінець, відкривається екран запуску завершення встановлення та запуску серверного додатку.

Реалізація проєкту виконувалась у двох середовищах розробки: Visual Studio Code (VS Code) та PyCharm. Для встановлення VS Code необхідно скачати інсталяційний файл із офіційної веб-сторінки Visual Studio Code (рисунок 3.3).

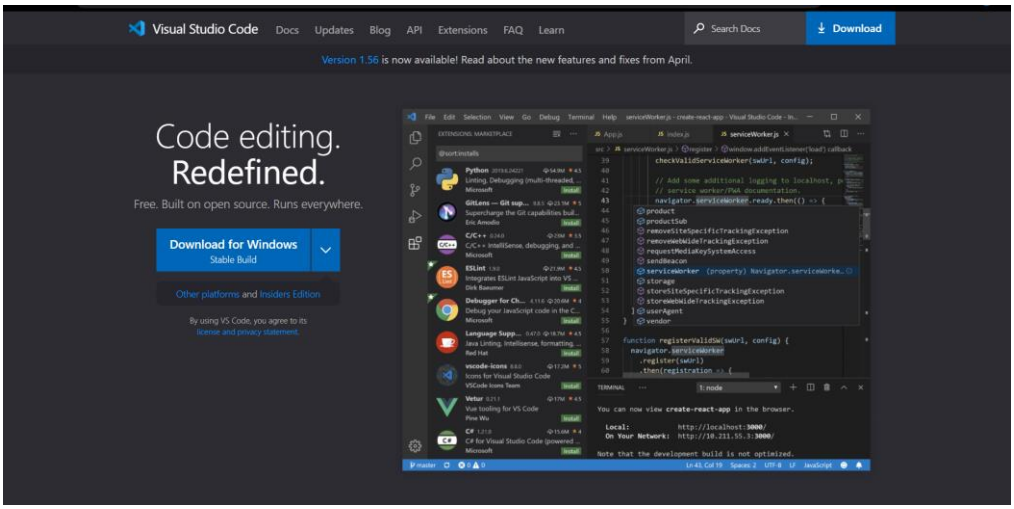


Рисунок 3.3 – Сторінка завантаження VS Code

Запускаємо інсталяційний файл та ставимо редактор коду на власний ПК. Появляється вікно з додатковими опціями установки. Вибираємо такі як зображено на рисунку 3.4 та переходимо далі.

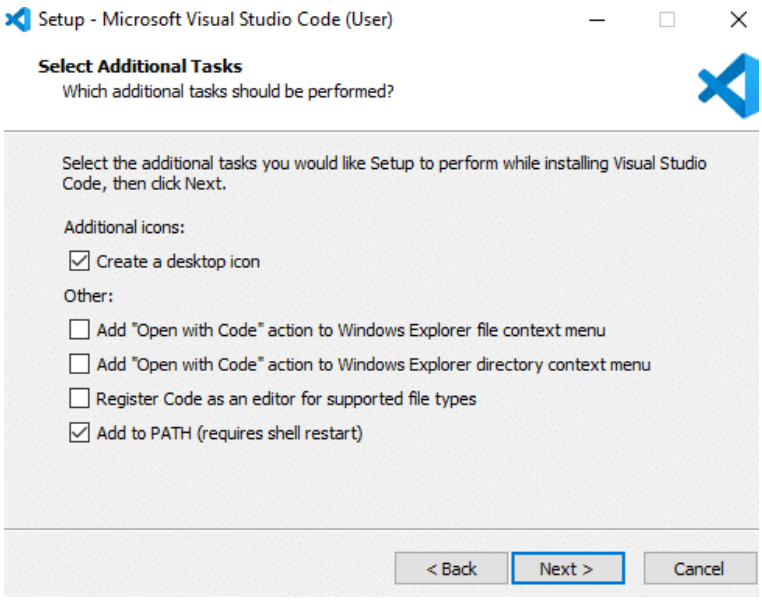


Рисунок 3.4 – Вибір параметрів установки

Після цього натискаємо на кнопку інсталяції. Наприкінці завершуємо установку. Запускається редактор коду (рисунок 3.5) і уже можна переходити до розробки веб-застосунку на PHP.

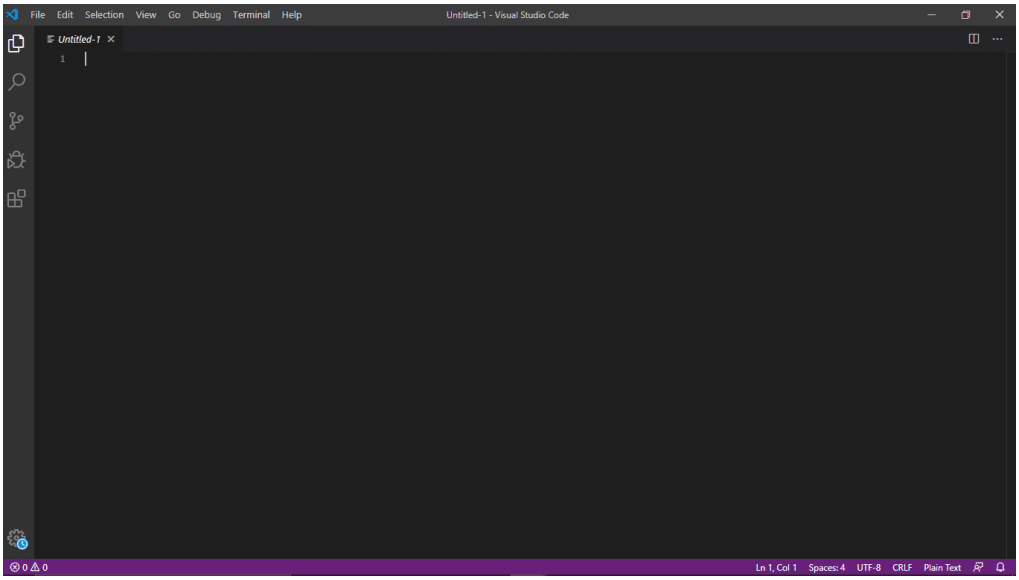


Рисунок 3.5 – Інтерфейс Visual Studio Code

Розробка програм на Python відбувалась у середовищі PyCharm. Для його встановлення необхідно відвідати офіційну сторінку. Потім натиснути кнопку «Download» під секцією «Community».

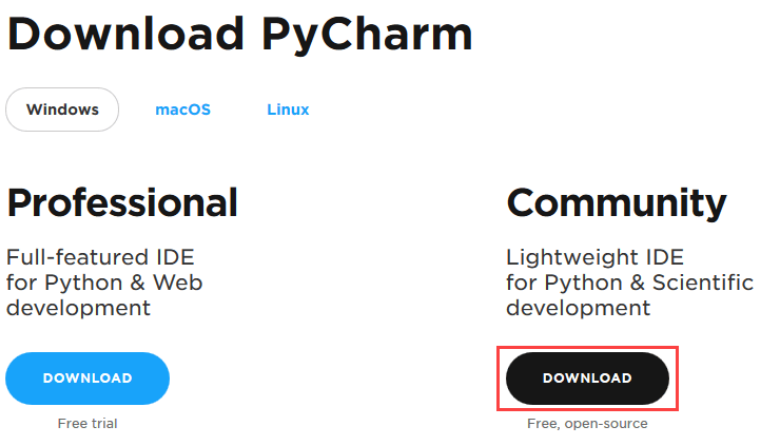


Рисунок 3.6 – Сторінка скачування PyCharm

Коли завантаження завершилось, запускаємо exe-файл інсталяції середовища. Відкриється початкове вікно установки. Переходимо далі та вказуємо шлях встановлення як показано на рисунку 3.7.

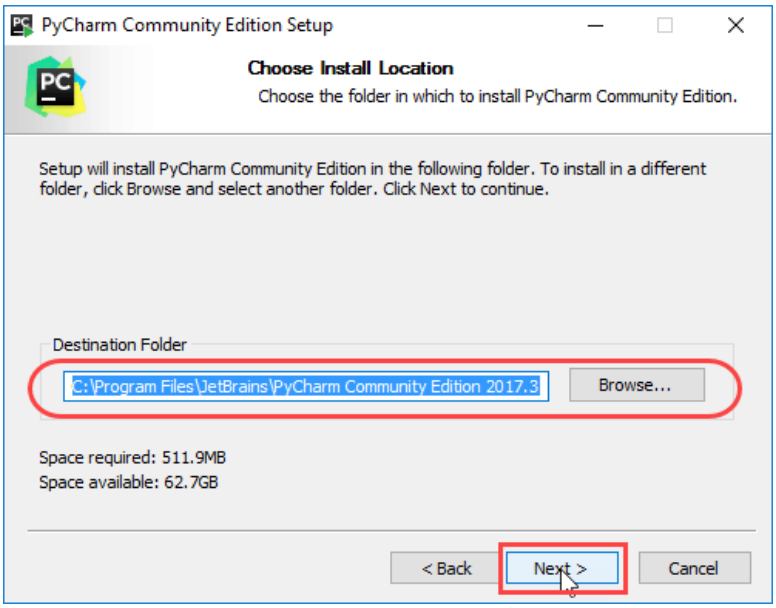


Рисунок 3.7 – Задання шляху інсталяції середовища

Далі установник попросить вказати ім'я для відображення в стартовому меню. За замовчуванням стоїть «JetBrains». Натисніть «Install». Установка почнеться автоматично (рисунок 3.8).

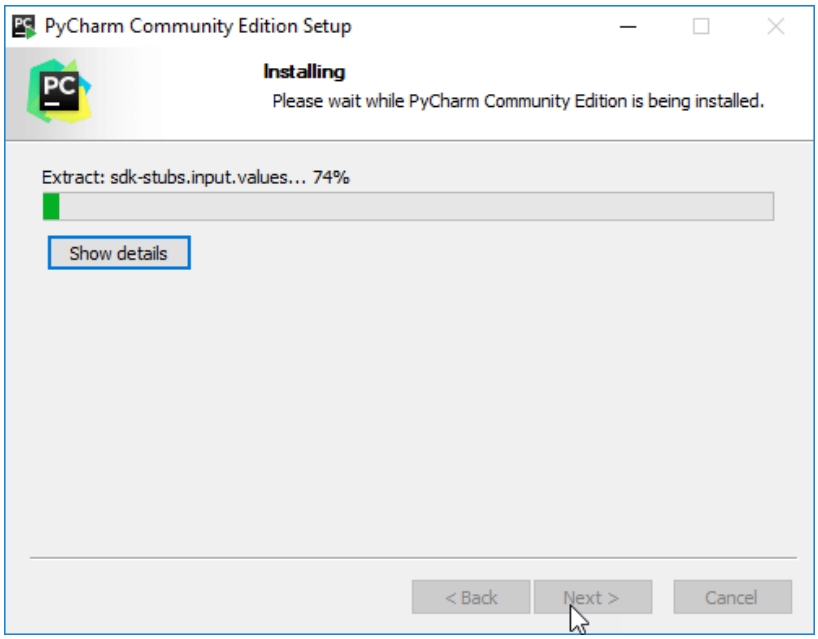


Рисунок 3.8 – Процес встановлення

Після завершення установки поставте галочку навпроти «Run PyCharm Community Edition», а потім «Finish». На рисунку 3.9 зображено даний процес.

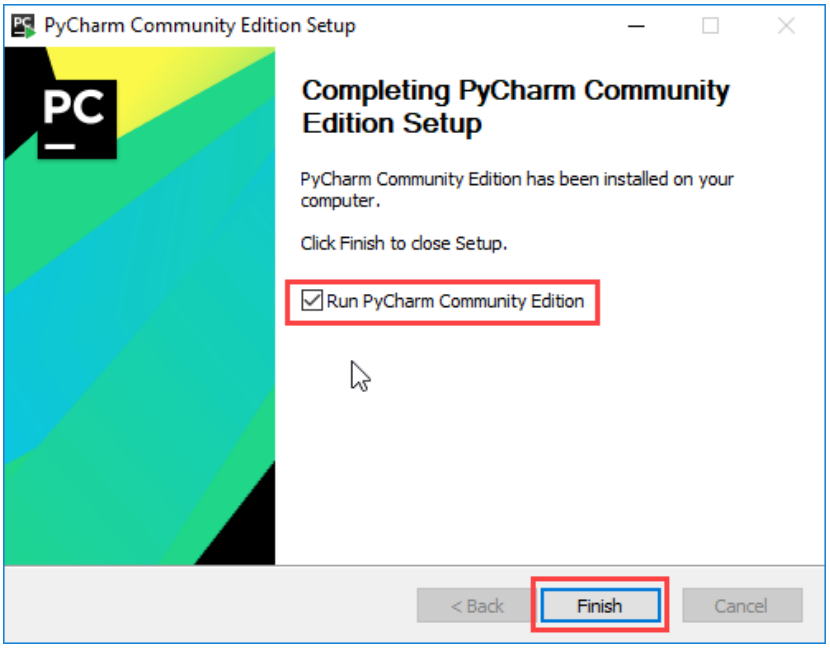


Рисунок 3.9 – Завершення встановлення PyCharm

Після завершення установки відкривається вікно створення проєкту. Натискаємо «Create New Project», як продемонстровано на рисунку 3.10, і переходимо до реалізації проєкту.

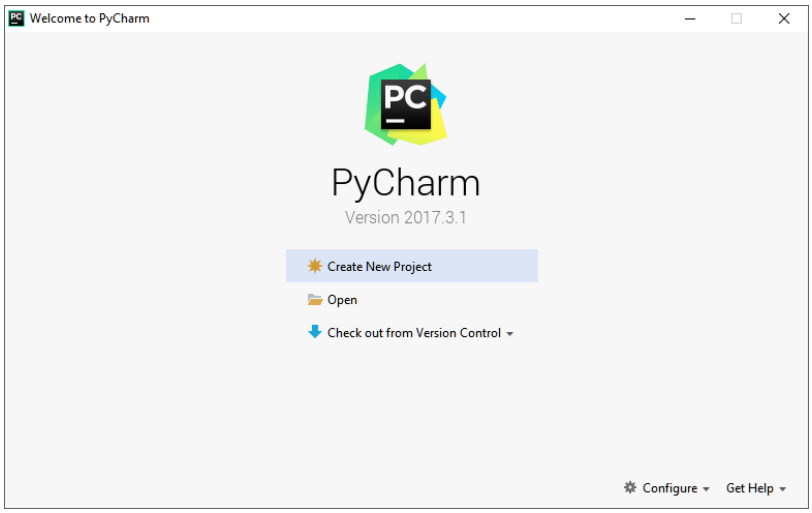


Рисунок 3.10 – Вікно створення проєкту

Також, одним із основних елементів розробленої ІС являється єдина база даних. Створення бази даних та її таблиць здійснювалось за допомогою phpMyAdmin. Щоб потрапити до нього для початку відкриваємо Xampp та запускаємо Apache і Mysql (рисунок 3.11).

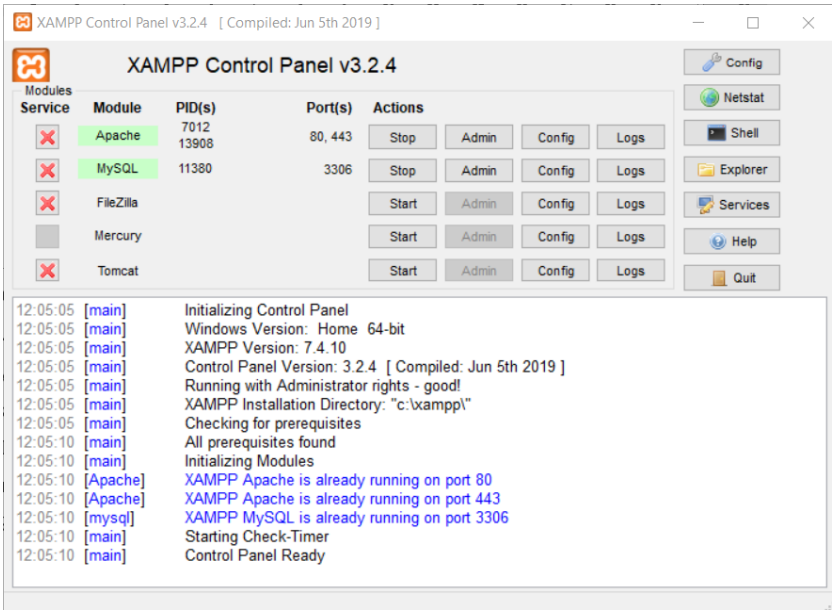


Рисунок 3.11 – Запуск модулів Apache та MySql

Коли локальний сервер запущений, у рядку адреси браузера необхідно вписати наступне посилання: http://localhost/phpmyadmin/. Щоб додати нову БД необхідно на лівій панелі phpMyAdmin натиснути кнопку «Нова». Після цього

з’явиться форма, де необхідно вказати ім’я та тип кодування символів бази даних (рисунок 3.11).

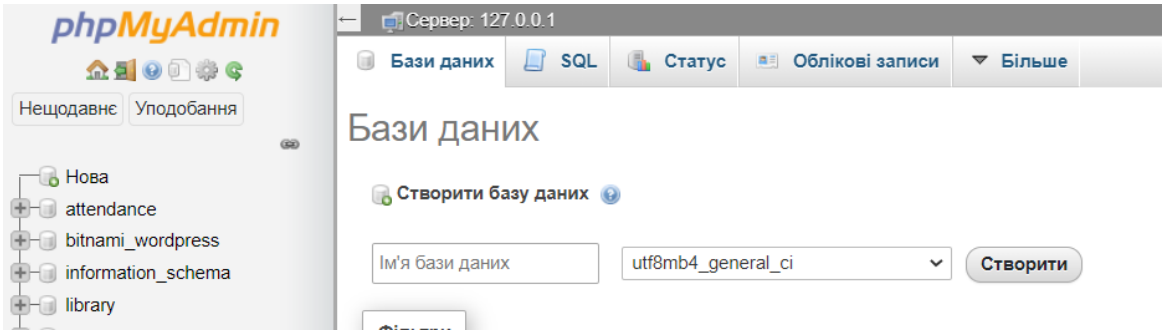


Рисунок 3.12 – Створення бази даних

Для додавання таблиць наявна кнопка «Нова» під назвою БД. Вибираємо її і появляється конструктор таблиць (рисунок 3.13).

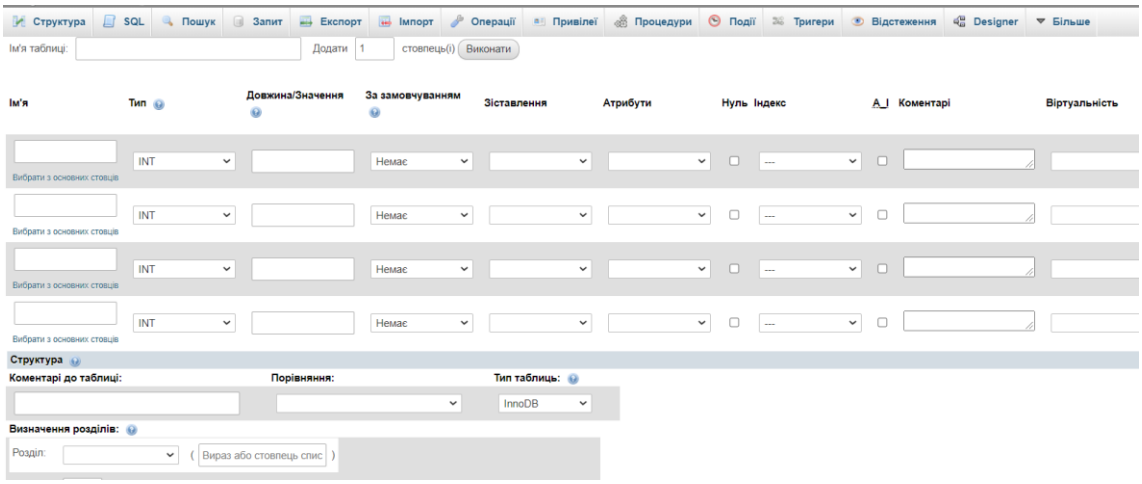


Рисунок 3.13 – Конструктор таблиць

Створюємо базу даних worker-attendance та 5 таблиць: workers, additional_info, recognition_data, records, admins. Вказуємо поля, типи даних для них, довжину, розставляємо індекси та застосовуємо інші параметри. При реалізації таблиць використовувались такі типи полів: int, varchar, date, tinyint, time, json (longtext), mediumblob. Щоб встановити зв’язки між таблицями, на вкладці «Структура» вибираємо поле з індексом таблиці натискаємо кнопку «Вид відносин». У формі, яка відкрилась (рисунок 3.14) задаємо поле іншої таблиці.

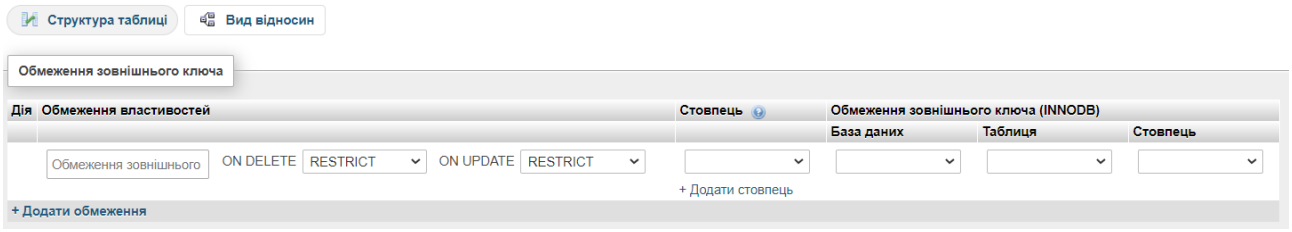


Рисунок 3.14 – Форма налаштування зв’язків

Як результат, отримуємо створену базу даних.

Інші етапи реалізації даного рішення можна розділити на дві частини. Перша частина – створення веб-застосунку. Друга – реалізація програмного забезпечення для обробки зображення, сканування та фіксації присутності.

Однією з найважливіших частин веб-додатку є реєстрація та авторизація користувачів. Розробка модуля реєстрації супроводжувалась наступними етапами:

– створення html-розмітки. Для можливості користувачеві ввести реєстраційні дані була реалізована форма, яка включає у себе 6 полів введення (прізвище, ім’я, по батькові, електронна пошта, пароль та поле його повторного введення), кнопку для вибору файлу, щоб завантажити зображення, та кнопку підтвердження і відправки даних на сервер;

– задання стилів для елементів;

– обробка введених даних. Перевірка коректності введень дозволяє уникнути помилок на наступних етапах розробки;

– відправка даних на сервер та збереження користувача в базі даних.

Після реєстрації користувач перенаправляється на головну сторінку системи – index.php. Вхід в систему відбувається аналогічним чином. Інтерфейс форми входу подано на рисунку 3.15.

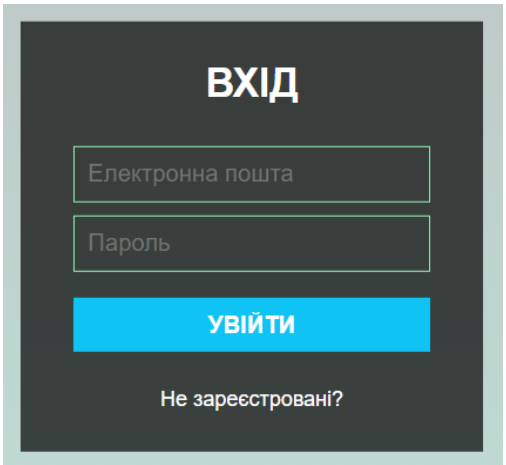


Рисунок 3.15 – Форма входу

Розглянемо реалізацію входу. HTML-розмітка із полями для введення електронної пошти та паролю:

```
<form method="post" class="login-form">
    <h1>Вхід</h1>
    <div class="input-form">
        <input name="worker_email" class="worker-
email" type="email" placeholder = "Електронна пошта">
        <span class="error-worker-email"></span>
        <input name="worker_password" class="password" type="pa
ssword" placeholder="Пароль">
        <span class="error-worker-password"></span>
        <input name="worker_login" class="login-
button" type="submit" value="Увійти">
    </div>
    <a href="registration.php" class="not-
registred">Не зареєстровані? </a>
</form>
```

Завдяки вказаному типу поля як «email» браузер самостійно здійснює перевірку введення на відповідність формату електронної пошти. Також, не менш важливим є вказання типу «password» для поля з паролем, що маскує його введення. При натисненні кнопки з типом submit виконується наступний код, реалізований на jQuery:

```
$('.login-form').on('submit',function(event) {
    event.preventDefault();
    $.ajax({
        url:"check-worker-login.php",
        method: "POST",
```

```
            data: $(this).serialize(),
            dataType:"json",
            success:function(data){
                if(data.success){
                   location.href="<?php echo $baseUrl;?>index.php"
;
                }
                if(data.error){
                    if(data.error_worker_name != ''){
                        $('.error-worker-email').text(data.
error_worker_email);
                    }else{
                        $('.error-worker-email').text('');
                    }
                    if(data.error_worker_password != ''){
                        $('.error-worker-password').text(data.
error_worker_password);
                    }else{
                        $('.error-worker-password').text('');
                    }
                }
            }
        })
    })
```

Вищенаведений код запобігає стандартній поведінці натиснення на кнопку submit та, за допомогою AJAX, відправляє дані з полів в check-worker-login.php, де дані проходять валідацію і повертається масив з інформацією про помилки чи успішне виконання:

```
    if($error == true){
      $output = array(
          'error'                 =>  true,
          'error_worker_email'    =>  $error_worker_email,
          'error_worker_password' =>  $error_worker_password,
      );
   }else{
      $output = array(
          'success'       =>  true,
      );
   }

   echo json_encode($output);
```

Після проходженні реєстрації та здійснення входу в систему відображається сторінка з профілем користувача. Дана сторінка містить верхню частину, яка

розміщена на усіх сторінках сайту - хеадер. Розмітка хеадера включає тег <img/> – для логотипу, <nav></nav> – елементи навігації та <a><a/> – кнопка виходу. Вигляд хеадера сайту подано на рисунку 3.16.

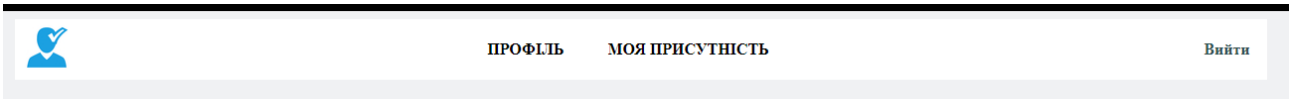


Рисунок 3.16 – Хеадер сайту

У зв'язку з наявністю хеадера на усіх сторінках, процедура підключення до БД була переміщена саме сюди:

```
$connect = new PDO("mysql:host=localhost;dbname=worker-attendance","root","");
$baseUrl = "http://localhost/attendance/";
```

HTML-розмітка хедера подана у лістингу А1, додаток А.

Наступна частина профілю розміщує в собі особисте зображення та основну інформацію про користувача. Так як зображення збережене в БД у вигляді бінарного набору даних (тип даних BLOB), його вставлення має специфічний вигляд, а саме: `<img src = "data:image/jpg; charset= utf8; base64, <?php echo base64_encode ($row['photo']); ?> "/>`, де $row['photo'] – зображення із бази даних, а base64_encode – функція кодування, що забезпечує коректну передачу бінарних даних по протоколах. Поля «Номер телефону», «Посада», «Стать» були необов'язковими для введення, тому, їх відсутність враховувалась при виведенні наступним алгоритмом:

```
<?php
    if(!empty($row['position'])){
        echo "<p>"."Посада: "."</p>";
        echo "<p>". $row['position'] ."</p>";
    }
?>
```

Вигляд вищезгаданих елементів продемонстровано на рисунку 3.17.

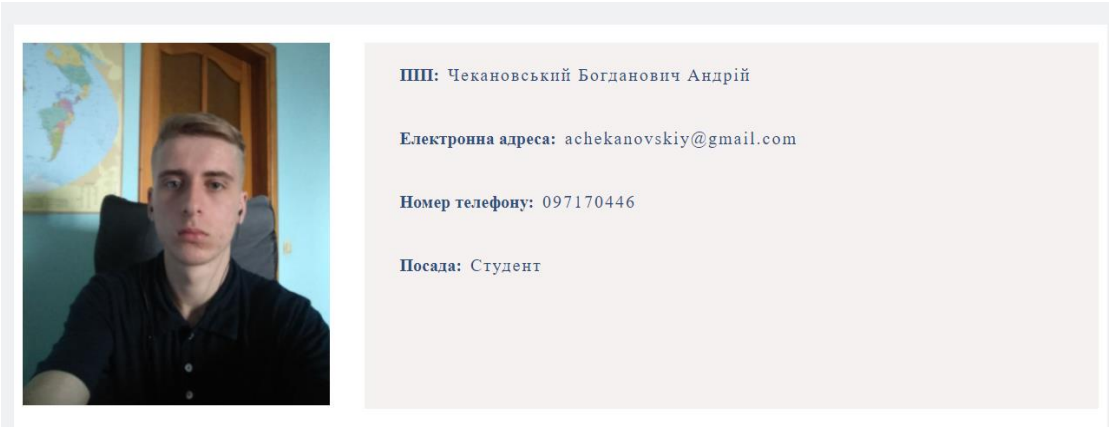


Рисунок 3.17 – Інтерфейс першої частини сторінки «Профіль»

У самому низу сторінки було реалізовано таблицю, що відображає відвідуваність працівника за останні 5 днів (рисунок 3.18).

Н/п	Дата	Прибуття	Час прибуття	Відхід	Час відходу
1	2021-05-04	✓	12:55:09	✓	12:55:13
2	2021-04-04	✓	15:37:57	✗	00:00:00
3	2021-03-04	✗	00:00:00	✗	00:00:00
4	2021-02-04	✓	15:55:05	✗	00:00:00
5	2021-01-04	✓	15:55:27	✗	00:00:00

Рисунок 3.18 – Таблиця відвідуваності користувача

Для отримання вищенаведених даних виконується такий SQL-запит до БД:

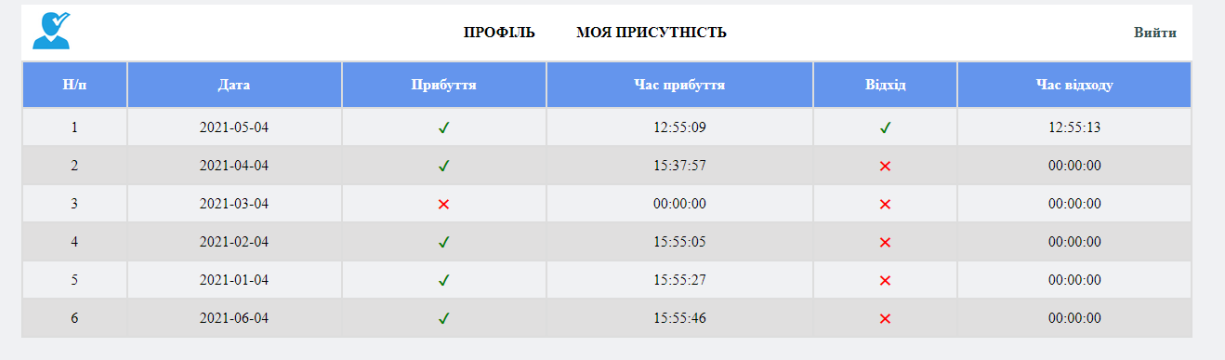
```
$query = "SELECT * FROM workers INNER JOIN additional_info ON workers.worker_id = additional_info.worker_id WHERE workers.worker_id = ".$_SESSION['worker_id'];
```

Для перетворення «1» і «0» в полях «Прибуття» та «Відхід» на символ галочки та хрестика було залучено спеціальні символи HTML, а саме – «✔» та «✖». Їх інтеграція в код відбувається наступним чином:

```
<?php
    if($row['status_arrival']=='1'){
        echo "<span style='color:green;'>&#10004;</span>";
    }else{
        echo "<span style='color:red;'>&#10006;</span>";
    }
?>
```

Код даної сторінки поданий у лістингу А2 додатку А.

Також, для користувача системи є можливість переглянути усі записи з його відвідуваністю. Переходячи на сторінку «Моя присутність», кількість записів уже не обмежена п’ятьма (рисунок 3.19).



Н/п	Дата	Прибуття	Час прибуття	Відхід	Час відходу
1	2021-05-04	✓	12:55:09	✓	12:55:13
2	2021-04-04	✓	15:37:57	✕	00:00:00
3	2021-03-04	✕	00:00:00	✕	00:00:00
4	2021-02-04	✓	15:55:05	✕	00:00:00
5	2021-01-04	✓	15:55:27	✕	00:00:00
6	2021-06-04	✓	15:55:46	✕	00:00:00

Рисунок 3.19 – Інтерфейс сторінки «Моя присутність»

Реалізація даної сторінки наведена у лістингу А3, додатку А.

За вихід із системи відповідає кнопка «Вийти». Вона видаляє всі дані з сесії та перенаправляє на форму входу:

```
<?php
    session_destroy();
    header('location:login.php');
?>
```

Додатковий функціонал, який надає веб-застосунок наданий адміністратору. Логін і пароль такого користувача уже збережений в БД. Вводимо їх на сторінці входу, яка має аналогічний вигляд сторінки входу працівника та потрапляємо на сторінку з усіма працівниками.

До основних можливостей адміністратора належить редагування даних працівників та їх видалення. На головній сторінці розроблена таблиця із 7-ма стовпцями: фото, прізвище, ім'я, по батькові, електронна пошта, редагувати, видалити. В кожен рядок даної таблиці будуть записуватись зареєстровані працівники. Останні два поля є кнопками, прив’язаними за кожним користувачем.

Реалізація їх функціоналу (лістинг А4, додаток А) здійснена за наступним алгоритмом: виконується запит на отримання даних з БД, при виведенні таблиці з

для кожної кнопки формується посилання на php-файл в якому конкатенується змінна з ідентифікатором працівника, а коли кнопка натискається – відкривається інша сторінка, яка використовує ці дані. Наприклад, посилання на видалення та оновлення відомостей про користувача виглядає таким чином:

```
<a href= "update-worker-action.php?edit = <?php echo $row['worker_id']; ?> "class="edit-worker"> <span style= "color: green; "> &#x0270E; </span> </a
<a href= "delete-worker.php?delete = <?php echo $row['worker_id']; ?> "class="delete-worker"> <span style= "color: red;">&#10007; </span> </a>
```

Тобто, дану логіку було реалізовано з використанням методу передачі даних, що називається GET-запитом. Далі скрипт перевіряє чи задана змінна «edit» та «delete», здійснює запит до БД. Для редагування інформації процедура виглядатиме так:

```
if(isset($_GET['edit'])){
    $query = "SELECT * FROM workers INNER JOIN additional_info ON workers.worker_id = additional_info.worker_id WHERE workers.worker_id =".$_GET['edit'];
    $statement = $connect->prepare($query);
    if($statement->execute()){
        $rows_count = $statement->rowCount();
        if ($rows_count > 0){
            $rows = $statement->fetchAll();
            $row = $rows[0]; ?>
    <div class="action-header"> <p class="action-title"> <?php echo $row['last_name'] ." ". $row['first_name']. " ". $row['patronymic']; ?> </p> </div>
?>
```

Код видалення запису ще також передбачає повернення на попередню сторінку, що виконує функція header(«location:»):

```
if(isset($_GET['delete'])){
    $query = "DELETE FROM `workers` WHERE `worker_id`=".$_GET['delete'];
```

```
        $statement = $connect->prepare($query);
        $statement->execute();
        header("location:index.php");
    }
```

Редагування відбувається на новій сторінці. На ній створено усі необхідні поля для зручності внесення змін (лістинг А5, додаток А). Нижче, на рисунку 3.20, продемонстровано інтерфейс сторінки оновлення даних користувачів.

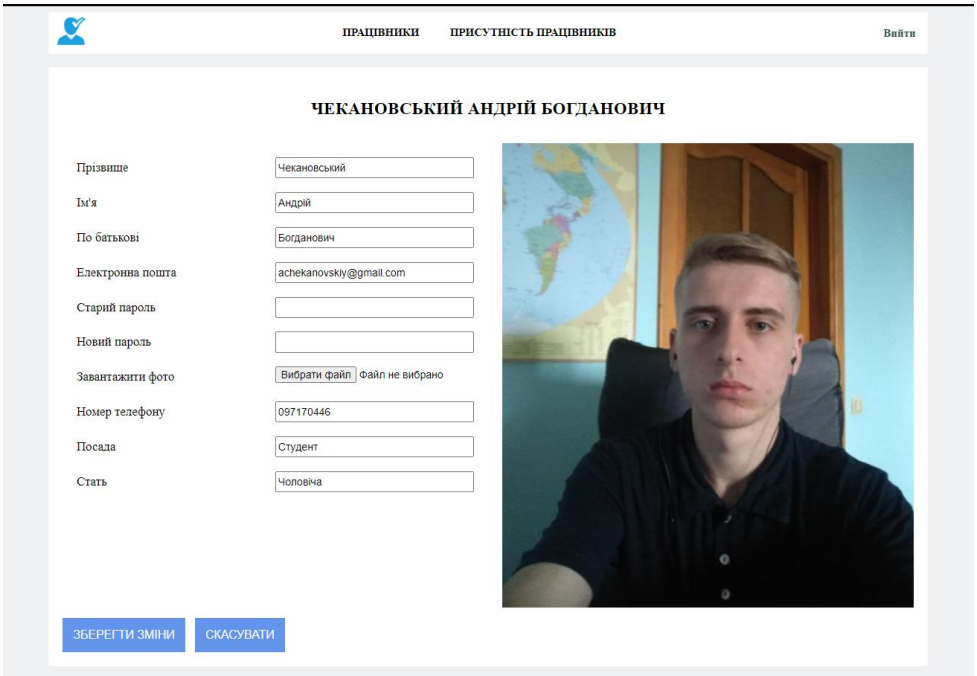


Рисунок 3.20 – Інтерфейс редагування даних

Для збереження змін чи їх скасування було реалізовано відповідні кнопки. Перед занесенням в БД відбувається валідація введень (лістинг А6, додаток А). До прикладу, щоб здійснити перевірку на коректність зображення, спершу відбувається перевірка на наявність вибору файлу, далі з'ясовується допустимий розмір файлу та визначається правильність розширення. Реалізація цього алгоритму:

```
    if ($_FILES['avatar']['name'] === ''){
        header("location:            update-worker-action.php?edit=
$worker_id");
    }else if ($_FILES['avatar']['size'] > 16777214) {
        $error = "Зображення занадто велике";
```

```
        $_SESSION["update_error"] = $error;
        header("location:                                update-worker-
action.php?edit=$worker_id");
    }              else              if              (!preg_match(
'/.*\.jpg$/',$_FILES['avatar']['name'])){
        $error  =  "Файл  некоректний!(доступні  розширення:  jpeg,
png)";
        $_SESSION["update_error"] = $error;
        header("location:      update-worker-action.php?edit      =
$worker_id");
    }else {
        $isImage=true;
    }
```

Наступним, що було реалізовано – сторінка записів відвідувань. Вона також являє собою таблицю, проте її функціонал дещо змінений і вдосконалений. При її реалізації деяку частину коду було написано на jQuery.

Для здійснення сортування перехоплювалась подія натискання на кнопку біля заголовку стовпця «ПІП» (або «Дата»). Потім, завдяки заданню в розмітці атрибуту name та data-order, за допомогою jQuery визначалась назва поля та тип сортування – за зростанням чи спаданням (лістинг А7, додаток А) . Програмний код реалізації сортування таблиці представлено у додатку А, лістинг А8.

Щоб виконати редагування певного запису після натиснення зеленої кнопки виконуються наступні кроки: визначення ідентифікатора запису, виведення розмітки з полями введення, заповнення їх уже наявними даними, блокування кнопки редагування для запобігання повторної генерації розмітки, натиснення на кнопку збереження даних – «Ок».

Інтерфейс даного блоку зображено на рисунку 3.21.

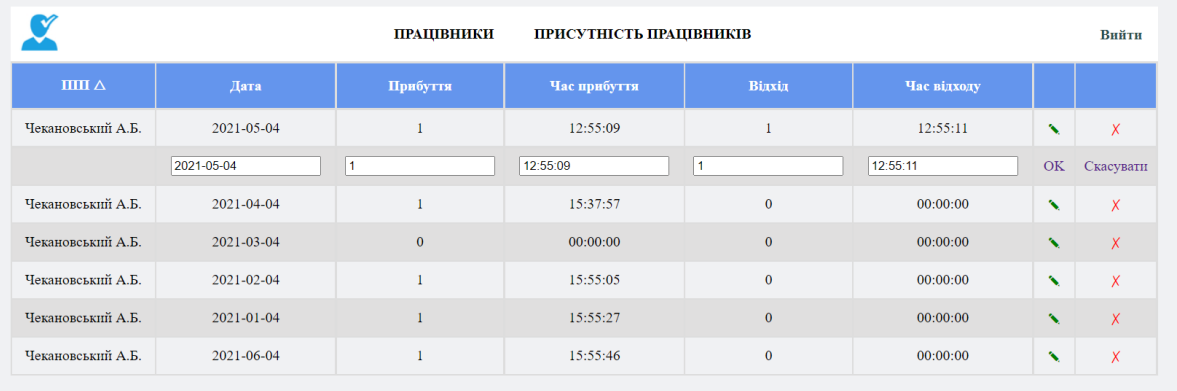


Рисунок 3.21 – Інтерфейс таблиці з даними присутності усіх працівників

Щоб зберегти введені зміни наявна кнопка «Ок», для скасування – кнопка «Скасувати». Програмний код сторінки реалізації обробки події редагування та видалення записів поданий у додатку А, лістинг А9.

Розробку ПЗ на Python було вирішено поділити на дві невеликі програми. Дане рішення прийняте через необхідність розмежування таких функцій як обробка обличчя певного користувача та позначення присутності (що не залежить від того, який користувач запустить дану програму).

Перша програма, яка має назву face-processing.exe та відповідає за виконання наступних функцій:

– відображення інтерфейсу;
– здійснення верифікації користувача за електронною поштою та паролем;
– підключення до бази даних;
– отримання зображення з БД;
– перевірка наявності обчислених даних обличчя поточного користувача;
– виявлення облич на зображенні;
– витягання ознак з обличчя;
– оновлення/додавання даних обличчя в БД.

Інша програма – mark-presence.exe виконує усі операції пов’язані із позначенням присутності користувача. Перелік функцій, які вона надає:

– відображення інтерфейсу;
– отримання даних усіх облич з БД;

– детекція обличчя на кожному кадрі, переданому через камеру;

– витягання ознак обличчя на поточному кадрі;

– визначення відповідності обличчя із тими, що розміщені в БД;

– позначення присутності ідентифікованого працівника.

Інтерфейс програми face-processing.exe складається з декількох вікон. Головне вікно містить 2 поля для введення електронної пошти та паролю. (рисунок 3.22).

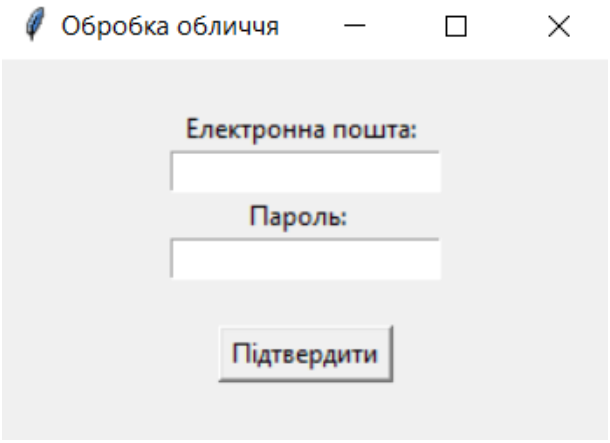


Рисунок 3.22 – Стартова сторінка програми face-processing.exe

Після введення значень в поля відбувається пошук такого користувача в системі. Якщо не знайдено – виконується процедура відображення вікна із відповідним повідомлення. У випадку, коли користувач існує, відбувається перевірка на наявність даних зображення для розпізнавання. Якщо вони існують, то користувач може здійснити закриття вікна, а у випадку зміни зображення в БД, здійснити повторну обробку шляхом натискання кнопки «Оновити дані» (рисунок 3.23).

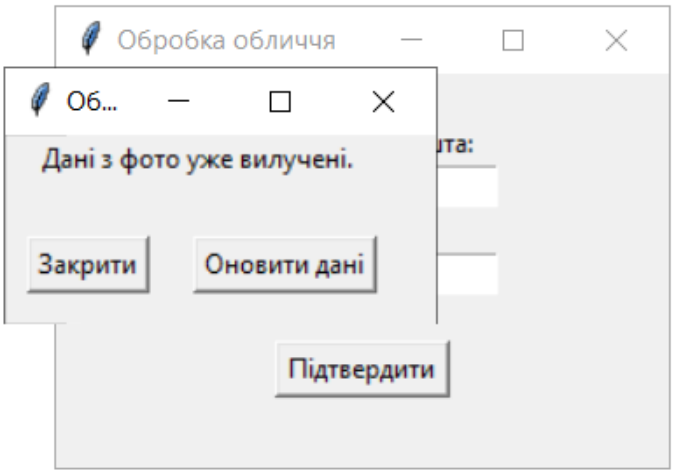


Рисунок 3.23 – Вікно оновлення даних

Основними функціями даної програми є:

– checkUnencodedWorkerFace(workerId) – здійснює перевірку наявності даних розпізнавання в БД. Виконується підрахунок кількості записів, які належать поточному користувачеві. Для цього використовується sql-функція count. Якщо кількість рівна нулю, то виконується відображення вікна з попередженням, що даних немає і необхідно їх обчислити . В іншому випадку – з'являється вікно з кнопкою, за допомогою якої можна оновити дані;

– detectFace(detector, image, inHeight, inWidth) – виконує попередню обробку зображення (змінює розміри, задає колірну модель RGB), пропускає його через детектор та повертає із заданою рамкою навколо обличчя;

– setEncodings() – виконує додавання даних обличчя в БД. Дана функція виконує ряд процедур та функцій, до яких належить: отримання зображення з бази даних для користувача, обчислення кодування обличчя, перетворення вилучених даних із 128-вимірного масиву у json-формат, виконання запиту на збереження json-даних;

– updateEncodings() – оновлює дані обличчя. Функціонує за схожим алгоритмом як setEncodings(), проте виконується запит до БД не на вставлення, а на оновлення даних;

– getImage(workerId, "image.jpg") – функція отримання зображення з БД;

– getEncodingsLocations(imageFromDb) – виконує повернення 128-вимірного масиву даних, вилучених з обличчя та координат розташування обличчя на фото. Розглянемо програмний код даної функції:

а) процедура отримання детектора та визначення розташування обличчя:

```
hogFaceDetector = dlib.get_frontal_face_detector()
outDlibHog, faceBorders=detectFace(hogFaceDetector, image)
```

б) процедура обробки зображення:

```
imageSmall = cv2.resize(image, (0, 0), None, 0.25, 0.25)
imageSmall = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

в) процедура вилучення даних з обличчя:

```
encFaces = []
for eachFace in faceBorders:
    currFaceLocation = [(eachFace[1], eachFace[2], eachFace[3], eachFace[0])]
encCurrFace = face_recognition.face_encodings(imageSmall, known_face_locations=currFaceLocation)
    encFaces.append(encCurrFace[0])
```

Для відображення інтерфейсу даної програми створено такі функції:

– login() – відображає форму входу для ідентифікації працівника;

– login_verify() – виконує перевірку коректності введених даних форми входу;

– user_data_incorrect() – вікно, яке повідомляє, що відповідно до введених даних не існує користувача ;

– encodings_are_empty() – екран з повідомленням про відсутність даних обличчя ;

– encodings_not_empty() – повідомляє про відсутність даних обличчя;

– success(message) – діалогове вікно, що інформує користувача про успішність виконання певного коду;

– error(message) – діалогове вікно, що інформує користувача про виникнення певної помилки під час виконання програми;

– delete_user_data_incorrect_screen() – видаляє екран некоректно введених даних;

– delete_encodings_not_empty_screen() – знищує вікно сповіщення про відсутність даних обличчя.

Реалізація функцій face-processing.exe розміщена у лістинг Б1, додаток Б.

Наступна програма – mark-presence.exe. Вона має простий інтерфейс, який складається з одного вікна. На ньому розміщено 2 кнопки – «Прибув» та «Покидаю» (рисунок 3.24).

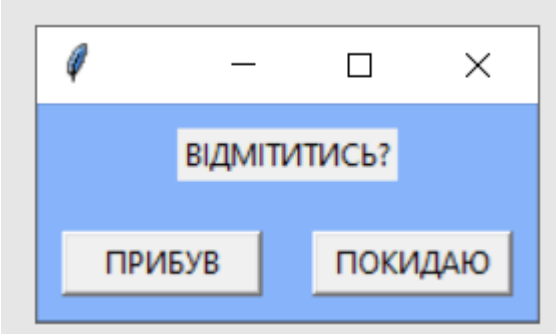


Рисунок 3.24 – Інтерфейс програми mark-presence.exe

Основними функціями даної програми є

– detectFace(detector, image, inHeight, inWidth) – функція детекції обличчя;

– compareEncodings(encodingsList1, encodingsList2) – функція порівняння поточних обличь та вибраних з бази даних;

– getEncodingsFromDb(workerId) – здійснює отримання масиву властивостей обличчя даного користувача із БД;

– realTimeRecognition(workerIds, encodeListKnown) – у реальному часі здійснює детекцію обличчя, виймає його властивості та порівнює з тими, які були їй передані;

– markAction() – відображає інтерфейс програми;

– leave() – функція, яка визначає подію натискання кнопки «Покидаю», викликає realTimeMark-функцію з переданим параметром «leave»;

– arrive() – визначає подію кнопки «Прибув», здійснює виклик функції realTimeMark, передаючи в неї параметр «arrive»;

– realTimeMark(action) – виконує операцію додавання записів в таблицю records бази даних. В залежності від того, який параметр вона отримала («arrive» чи «leave»), дані записуються у відповідні поля. Фрагмент коду даної функції:

```
    if action == "arrive":
        SQLStatement = "INSERT INTO `records` (`worker_id`, `status_arrival`, `date`, `time_arrival`, `status_leaving`, `time_leaving`) VALUES (%s, %s, %s, %s, %s, %s)"
        mycursor.execute(SQLStatement, (foundWorkers[0], '1', currentDay, currentTime, '0', '00:00:00'))
        conn.commit()
    elif action == "leave":
        SQLStatement = "INSERT INTO `records` (`worker_id`, `status_arrival`, `date`, `time_arrival`, `status_leaving`, `time_leaving`) VALUES (%s, %s, %s, %s, %s, %s)"
        mycursor.execute(SQLStatement, (foundWorkers[0], '0', currentDay, '00:00:00', '1', currentTime))
        conn.commit()
```

Реалізація функцій mark-presence.exe розміщена у Додатку Б, Лістинг Б2.

3.3 Тестування інформаційної системи

Оцінка коректності роботи системи та її компонентів є важливим елементом при їх розробці. Простими словами, необхідно протестувати систему – виявити будь-які прогалини, помилки або невідповідність поставленим вимогам.

Здійснимо тестування форми реєстрації. При введенні коректних даних у поля форми система успішно виконує занесення даних та перенаправляє на іншу сторінку. На рисунку 3.25 продемонстровано процедуру введення.

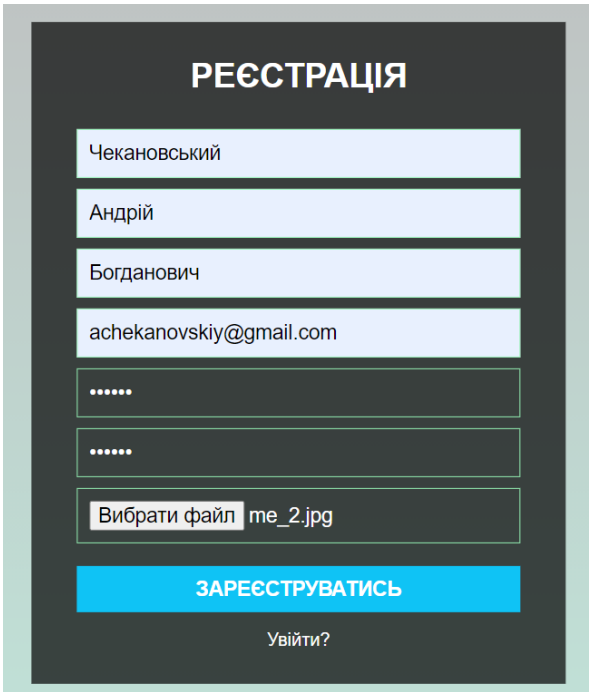


Рисунок 3.25 – Введення реєстраційних даних

Далі виконаємо введення некоректних даних. Наприклад, не вкажемо значення для певного поля (рисунок 3.26).

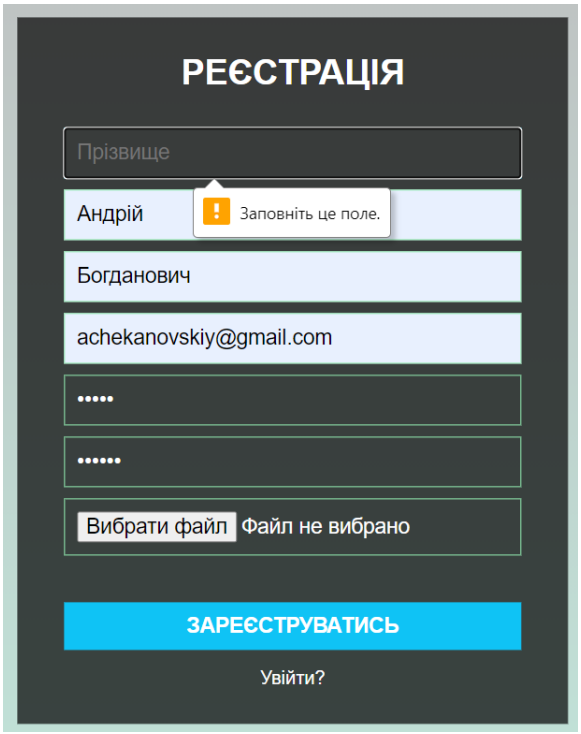


Рисунок 3.26 – Введення некоректних даних

Можна побачити, що з’являється повідомлення про пусте поле. Далі, введемо пароль підтвердження відмінний від заданого. Користувача сповіщає про дану помилку (рисунок 3.27).

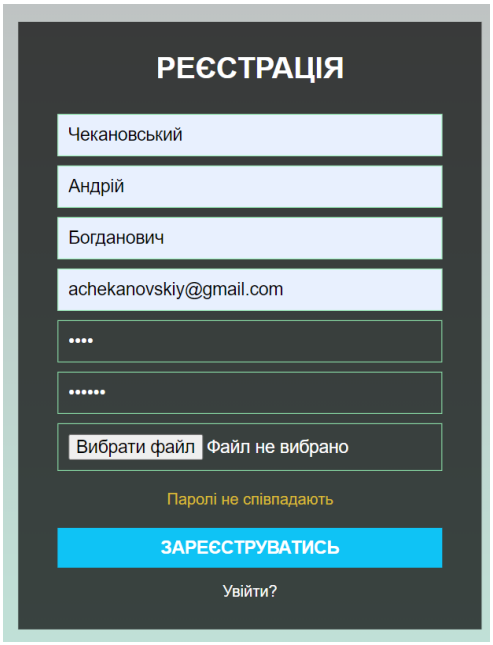


Рисунок 3.27 – Невідповідність паролів

Після дотримання усіх вимог введення, натискаємо кнопку «Зареєструватись». Користувача успішно перенаправляє на головну сторінку – «ПРОФІЛЬ». При переході на сторінку «МОЯ ПРИСУТНІСТЬ» жодних помилок не виявлено, вона відображається теж коректно.

На даний момент, жодних записів користувача немає, тому усі таблиці пусті (рисунок 3.28).

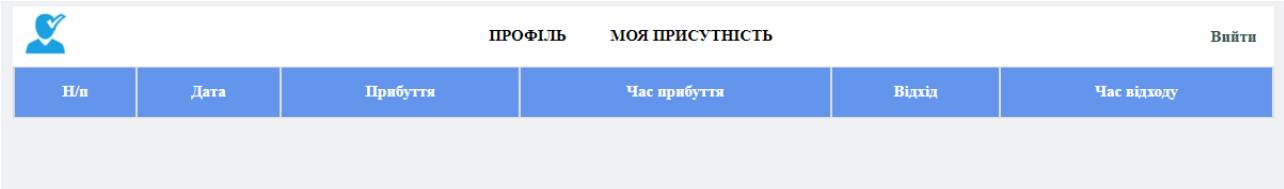


Рисунок 3.28 – Таблиця для зберігання записів

Тепер, перейдемо до тестування програми відмічання працівника. Для початку, запустимо програму обробки зображення face-processing.exe. Нам доступні поля для введення електронної пошти та паролю. Якщо введені дані не

відповідають даним жодного користувача, то відображається вікно з інформацією про це (рисунок 3.29).

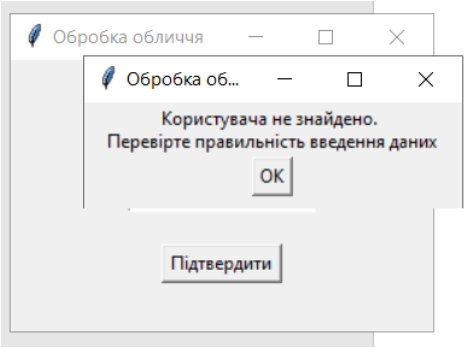


Рисунок 3.29 – Вікно з інформацією про некоректність даних користувача

Нажимаємо кнопку «ОК» або закриваємо програму.

У іншому ж випадку, коли користувач існує, з'являється інше вікно, яке дає можливість обчислити дані із обличчя на зображенні (рисунок 3.30), або оновити їх (рисунок 3.31).

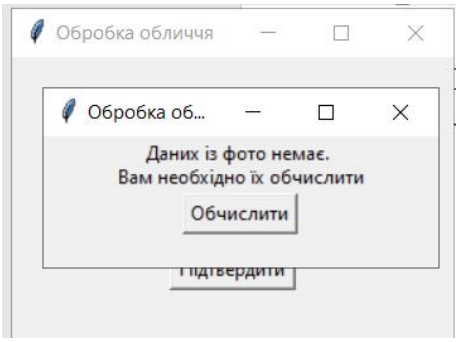


Рисунок 3.30 – Вікно обчислення

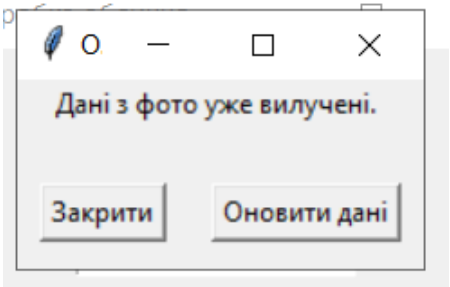


Рисунок 3.31 – Вікно оновлення

Процедура додавання даних обличчя проводиться тільки один раз, або у випадках, коли фото користувача було змінене. Після збереження даних обличчя програма face-presence.exe може розпізнати цю людину. Запускаємо exe-файл та натискаємо на кнопку «ПРИБУВ». Після цього запускається камера та розпізнає працівника (рисунок 3.32)

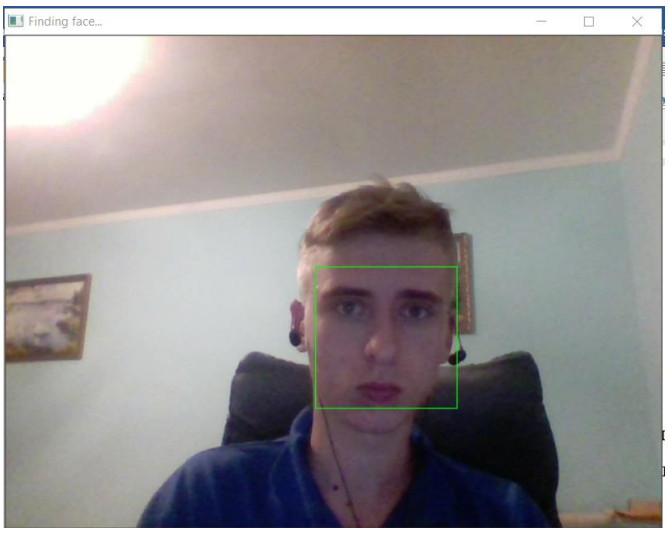


Рисунок 3.32 – Процес розпізнавання

Тобто, щойно було здійснено відмічання працівника про прихід на робоче місце. Перевірити чи насправді це виконалось можна з особистого профіля у веб-застосунку. Сторінка продемонстрована на рисунку 3.33.

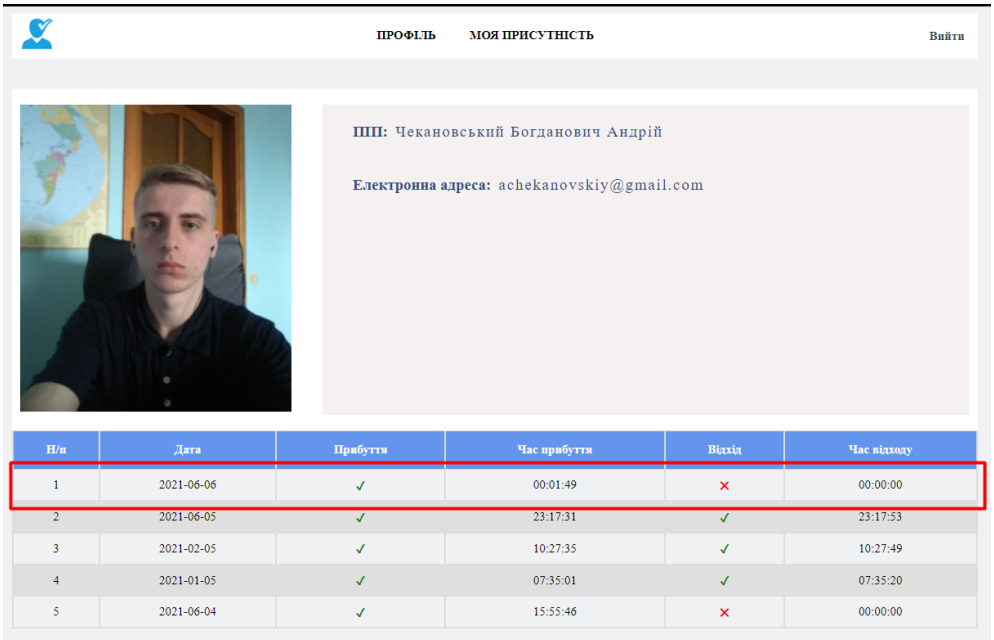


Рисунок 3.33 – Додано новий запис

Значок галочки та хрестика у полях «Прибуття» та «Відхід» інформує про статус прибуття та відходу працівника. Кнопка виходу із системи працює коректно та здійснює перенаправлення на форму входу.

Протестуємо тепер можливості адміністратора. Для цього додаємо до URL-адреси ключове слово «admin» та переходимо на задану сторінку. Відкривається форма входу (рисунок 3.34).

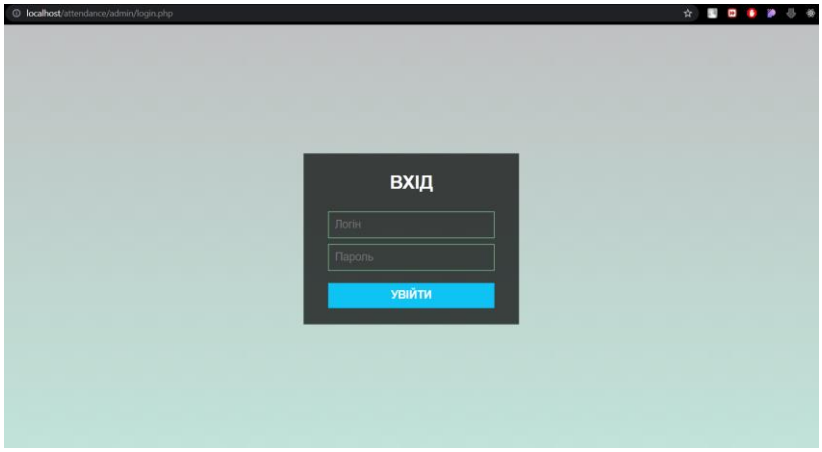


Рисунок 3.34 – Сторінка входу для адміністратора

При відсутності заповнених полів з’являється повідомлення про це під кожним полем (рисунок 3.35).

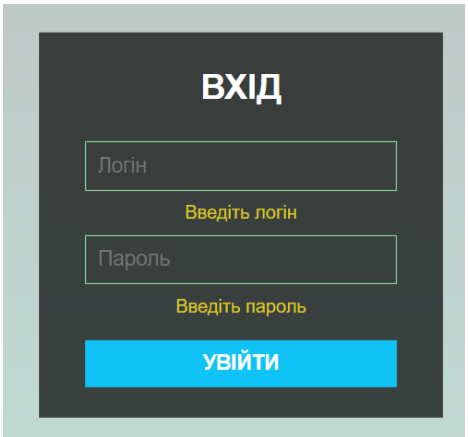


Рисунок 3.35 – Некоректність полів вводу

Вводимо пароль та логін. Нас перенаправляє на головну сторінку. Усі введені попередньо дані відображаються правильно. Тепер протестуємо кнопки редагування та видалення (рисунок 3.36).

Рисунок 3.36 – Кнопки редагування та видалення користувачів

Натиснувши на першу кнопку відкриється сторінка редагування даних (рисунок 3.37).

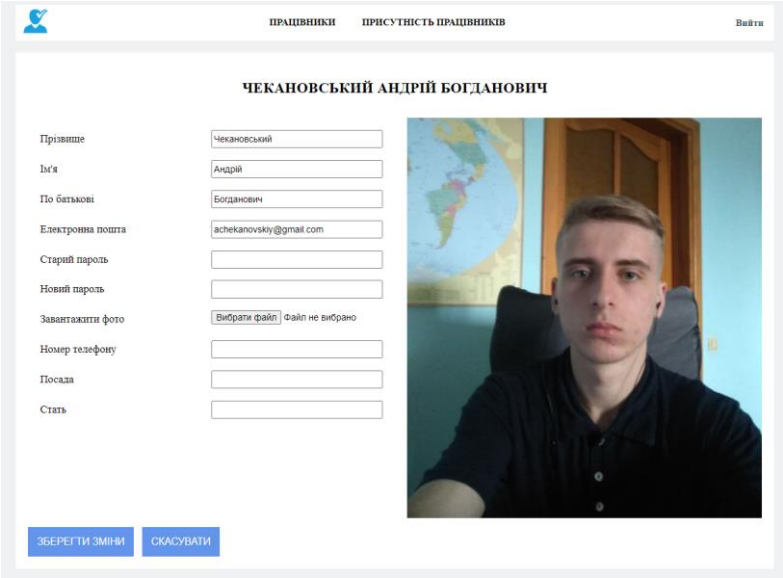


Рисунок 3.37 – Форма редагування даних

На даній формі передбачено: недопустимість пустих полів з основною інформацією, перевірка на коректність електронної адреси, перевірка паролю, неможливість вибрати інший файл, відмінний від файлу з розширенням jpg або png і т.д. На рисунку 3.38 продемонстровано як відображаються помилки.

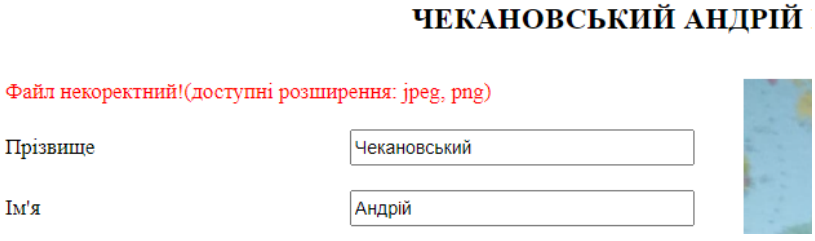


Рисунок 3.38 – Помилка при виборі зображення

Заповнюємо поля форми та натискаємо кнопку «ЗБЕРЕГТИ ЗМІНИ». Усі введені дані вдало зберігаються в БД. Тепер, коли користувач відкриє свій профіль, він побачить уже редаговані дані.

Спробуємо видалити користувача. На головній сторінці адміністратора нажимаємо на хрестик у вибраному рядку таблиці. Сторінка перезавантажиться і, як результат, одного користувача уже не буде в системі (рисунок 3.39).

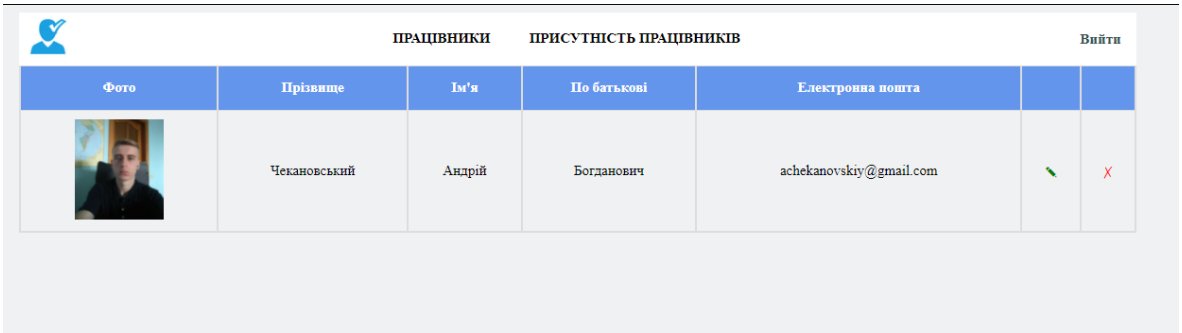


Рисунок 3.39 – Демонстрація роботи кнопки видалення

Для маніпуляції даними записів переходимо на сторінку «Присутність працівників». На ній існують такі ж кнопки редагування та видалення, як і на сторінці «ПРАЦІВНИКИ». Виконуємо виклик форми редагування натисканням кнопки, яка знаходиться в передостанньому стовпці таблиці (рисунок 3.40).

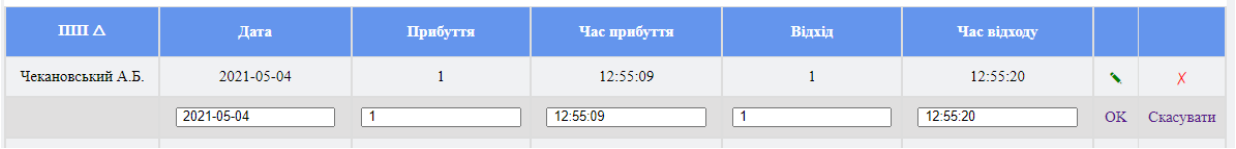


Рисунок 3.40 – Форма редагування записів

Вносимо зміни, наприклад, у полях з часом прибуття/відходу, тиснемо кнопку «Ок». Результат можемо побачити на рисунку 3.41.

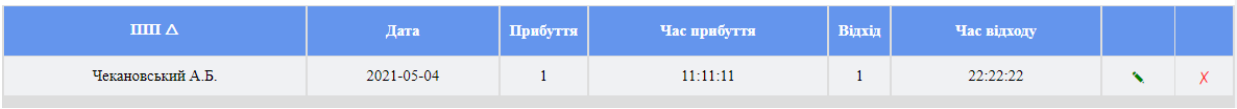
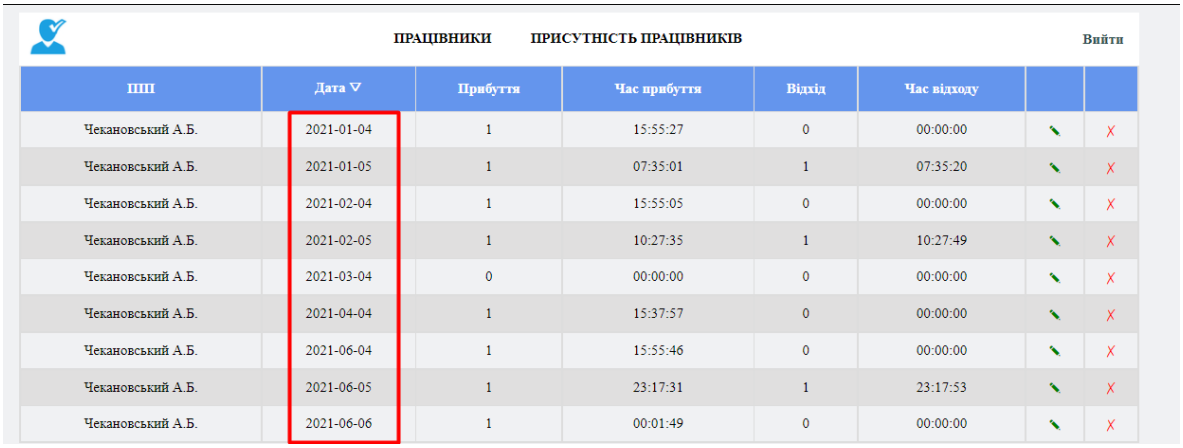


Рисунок 3.41 – Відредагований запис

Також спробуємо видалити запис за допомогою червоного хрестика. Тепер вищенаведеного запису не існує. Для зручності перегляду записів можемо здійснити сортування даних таблиці по полю «Дата». Для цього нажимаємо по заголовку даного поля (рисунок 3.42).



ПІП	Дата ▽	Прибуття	Час прибуття	Відхід	Час відходу		
Чекановський А.Б.	2021-01-04	1	15:55:27	0	00:00:00		
Чекановський А.Б.	2021-01-05	1	07:35:01	1	07:35:20		
Чекановський А.Б.	2021-02-04	1	15:55:05	0	00:00:00		
Чекановський А.Б.	2021-02-05	1	10:27:35	1	10:27:49		
Чекановський А.Б.	2021-03-04	0	00:00:00	0	00:00:00		
Чекановський А.Б.	2021-04-04	1	15:37:57	0	00:00:00		
Чекановський А.Б.	2021-06-04	1	15:55:46	0	00:00:00		
Чекановський А.Б.	2021-06-05	1	23:17:31	1	23:17:53		
Чекановський А.Б.	2021-06-06	1	00:01:49	0	00:00:00		

Рисунок 3.42 – Результат видалення запису та сортування таблиці

Отже, було проведено тестування програмного засобу. В ході тестування усі виявлені недоліки було усунено, як у веб-застосунку, так і на стороні опрацювання і збереження даних за допомогою програм. При тестуванні користувацької та адміністративної частини, було проведено перевірку роботи модуля реєстрації, входу в систему, правильність роботи з базою даних та формами редагування, видалення даних.

При перевірці роботи зовнішніх програм увага зверталась на модулі отримання даних з БД, їх опрацювання, на успішне здійснення відмітки, шляхом розпізнавання обличчя.

В результаті проведеного тестування можна зробити висновок, що система працює коректно та готова до експлуатації.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Аналіз ринку

Функціонал даного рішення є достатнім для зручного та ефективного користування, з мінімальними технічними вимогами до обладнання користувача. Воно знаходить певну спільність з деякими виробами на ринку. Проте, кожне з них має свої особливості та відрізняється широким спектром характеристик, способами інтеграції, методами реалізації та шляхами експлуатації.

На українському ринку є декілька, помітно лідируючих компаній, які займаються розробкою біометричних терміналів, програмного забезпечення відвідуваності та обліку робочого часу. Враховуючи вищесказане, реалізований виріб можна вважати модифікацією уже існуючих. Застосунок вирішить питання пов’язані із управлінням персоналом та вдосконаленням робочого процесу. Він допоможе автоматизувати ведення журналу відвідуваності, забезпечить прозорість та контроль за персоналом, дасть можливість досягти організаційної гнучкості. До того ж, системи обліку присутності використовуються в структурах із значною кількістю працівників, що збільшує шанси системи виходу на ринок. Зацікавленими у таких системах є організації, фірми, підприємства, навчальні заклади і т.д. Потенційним замовником даного виробу являється останній з переліку, а саме – коледж. Оскільки мовою інтерфейсу даного програмного рішення є українська, головним ринком його реалізації виступає український.

В Україні кількість закладів, що використовують автоматизовані системи обліку відвідуваності є незначною, що може стати причиною високого попиту на даний виріб. Варто зазначити, що рівень попиту буде залежати від його конкурентоспроможності, вартості, якості та багатьох інших факторів. При виході товару на ринок, найефективнішим методом його продажу буде електронна торгівля. Це зменшить витрати на організацію, підтримку бізнесу, дасть можливість ефективно та зручно здійснювати операції, буде піддатливий розширенню і глобалізації. Враховуючи, що даний товар є програмним застосунком, це дозволяє обсягу продаж бути необмеженими.

4.2 Розрахунок витрат на проєктування

Розробка будь-якого проєкту без сумнів несе певні затрати: від закупки базових інструментів до витрат на заробітню плату. В таблиці 4.1 продемонстровано кошторис витрат на проєктування.

Табл. 4.1 – Кошторис витрат на проєктування

Витрати	Сума, грн	Обґрунтування
1. Заробітна плата проєктувальників	80661	Загальна зарплата усіх учасників проєкту
2. Відрахування на соціальні потреби	17745,4	Складає 22% від суми заробітних плат
3. Контрагентські роботи і послуги	0	Не проводились
4. Витрати на відрядження	0	Співробітники не були у відрядженні
5. Додаткові прямі витрати	300	Хостинг, домен
6. Прямі витрати	98706,4	Усього прямих витрат
7. Накладні витрати	29611,9	Витрати на електроенергію, приміщення
8. Планові накопичення	25663,6	Витрати на покращення рівня розвитку проекту
9. Кошторисна вартість	153982,01	Сума планових накопичень, накладних витрати і прямих витрат
10. ПДВ	30796,4	Податок на додану вартість складає 20% від кошторисної вартості
11. Розробка загалом	184778,4	Загальна ціна розробки

Розробкою даного прєкту займалось 6 спеціалістів: фронтенд-розробник, бекенд-розробник, дизайнер, пайтон-розробник, тестувальник. Заробітна плата працівників залежить від кількості спеціалістів, розміру посадових окладів та терміну розробки його перебування у розробці.

Мінімальний посадовий оклад / тарифна ставка згідно статті 96 КЗпП на 01 січня календарного року повинна бути встановлена у розмірі, не меншому за прожитковий мінімум для працездатних осіб, тобто, 2270 грн.

Розрахунок зарплати проводиться у таблиці 4.2.

Табл. 4.2 – Розрахунок заробітної плати проєктувальників

N	Посада	Ставка	Відрахування	Кількість		Загальна сума
п/п		грн/міс	грн/міс	чол.	місяців	з/п, грн.
1	Керівник ДП	6800	858	1	4	21896
2	Фронтенд-розробник	18000	3510	1	1	14490
3	Бекенд-розробник	23000	4485	1	1	18515
4	Дизайнер	9000	1775	1	1	7245
5	Пайтон-розробник	14000	2730	1	1	11270
6	Тестувальник	9000	1775	1	1	7245
		Усього зарплати:				80661

Дані заповнення вищенаведених таблиць утворені шляхом обчислення та врахування різних економічних аспектів. До них входить:

– податок на доходи фізичних осіб;

– військовий збір;

– єдиний внесок;

– відрахування;

– виконувані роботи та послуги;

– витрати на відрядження;

– допоміжні прямі витрати.

Усі таблиці обрахунків економічної складової проекту подані у таблицях В1 – В5 додатку В.

4.3 Обґрунтування необхідності розробки

Інтеграція автоматизованих систем, перенесення ручної праці в машинну, здійснення переходу у сферу інженерних та інформаційних технологій – усі дії, які націлені на більш ефективне виконання конкретної роботи. Це дозволяє позбутись рутинних і механічних справ, дає можливість сфокусуватись на виконанні інших задач, здійснення яких неможливе без застосування досвіду та знань працівника.

Системи автоматизованого обліку присутності персоналу демонструють найбільшу ефективність у закладах із значною кількістю кадрів. Відсутність паперів, відсутність ручного ведення журналу та зайвого вкладу людських зусиль усуває додаткове навантаження на виконуючого ці обов’язки.

Система надзвичайно корисна, оскільки забезпечує зручний моніторинг відвідуваності, де користувач може легко отримати дані конкретного працівника чи створити систематичний загальний звіт. Після того, як відвідуваність позначена, зафіксовані дані зберігаються в системі управління відвідуваністю, звідки кожен, хто має права, може переглянути деталі. Ця функція особливо корисна під час пошуку конкретної особи або під час здійснення аналізу. Завдяки збереженню записів у базі даних, на противагу звичайного методу ведення реєстру, вагомо спрощує процес пошуку конкретного запису. Головна перевага цієї системи – вона запроваджує керований та систематизований підхід до ведення обліку відвідуваності.

ВИСНОВКИ

Даний дипломний проєкт присвячений розробці та дослідженню систем обліку на основі технології розпізнавання облич. В результаті дослідження можна зробити висновок, що системи розпізнавання – це ефективний, сучасний та багатогранний інструмент, функціональні можливості якого можуть покращити значну кількість процесів у декілька, або навіть в десятки разів. Існує ряд сфер та задач, де застосування технологій комп'ютерного зору є невід'ємним атрибутом.

З іншої сторони, система автоматизованого обліку присутності працівників є доцільною та актуальною, через те, що в період інформаційних технологій цінуються саме ті рішення, які автоматизують процеси, що раніше вимагали значного людського втручання. Даний проєкт відповідає цій вимозі, а також усім іншим, які були поставлені перед початком розробки.

В процесі виконання дипломного проєкту було проведено аналіз наявних систем-аналогів, здійснено дослідження в галузі розпізнавання облич, вивчено основні аспекти, необхідні для реалізації даного проєкту. Було розглянуто питання вибору засобів та мов реалізації, яким чином взаємодіє веб-інтерфейс з програмним кодом на PHP та jQuery, з базою даних, який набір функціональних можливостей доступний користувачеві і т.д.

Навіть, незважаючи на те, що вона успішно виконує поставлені задачі, існує безліч варіантів розширення як її програмної бази, так і способів та шляхів інтеграції. Даний програмний продукт можна удосконалити та використовувати не тільки у навчальному закладі, що передбачалось при його створенні, але і в інших офісних установах.

В подальшому систему можна урізноманітнити певною кількістю діаграм, графіків, додати можливість формувати звіти та завантажувати їх у вигляді pdf-файлів. Таким чином, розроблена система має великі перспективи для подальшого розгортання у масштабніший проєкт.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Біометричний термінал ZKTeco speedface-v5l. *Secure.ua*: вебсайт. URL: https://secur.ua/ua/biometricheskij-terminal-zkteco-speedface-v5l-ti-s-detekciej-temperatury.html (дата звернення: 07.01.2021).

2. Біометричний зчитувач геометрії обличчя Anviz FacePass Pro. *відеокамери.com.ua:* вебсайт. URL: https://xn--80adgebslrpy8u.com.ua/Anviz_FacePass (дата звернення: 13.05.2021).

3. Face Recognition Terminals. *Hikvision.com:* вебсайт. URL: https://www.hikvision.com/en/products/Access-Control-Products/Face-Recognition-Terminals/ (дата звернення: 15.05.2021).

4. Чекановський А.Б. Комп'ютерний зір і обробка зображень. Гістограма напрямлених градієнтів. Збірник наукових тез: за матеріалами студентських наукових читань. Тернопіль: Навчально-практична майстерня редакційно-видавничих технологій Галицького коледжу імені В'ячеслава Чорновола, 2021р., С. 208-214.

5. Діаграми «сутність-зв'язок» – самовчитель UML. *Waykun*: вебсайт. URL: https://jak.waykun.com/articles/diagrami-sutnist-zv-jazok-samovchitel-uml.html (дата звернення: 25.03.2021).

6. Що таке Python? Дізнайтеся що являє собою мова програмування Python та які її переваги. *Futurenow*: вебсайт. URL: https://futurenow.com.ua/shho-take-python-piton-perevagy-programuvannya-na-python/ (дата звернення: 17.04.2021).

7. OpenCV. *Wikipedia*: вебсайт. URL: https://uk.wikipedia.org/wiki/OpenCV (дата звернення: 18.04.2021).

8. Система розпізнавання облич. *Wikipedia*: вебсайт. URL: https://uk.wikipedia.org/wiki/Система_розпізнавання_облич (дата звернення: 20.04.2021).

9. Що таке NumPy? *uk.photo-555.com*: вебсайт. URL: https://uk.photo-555.com/1909487-what-is-numpy (дата звернення: 22.04.2021).

10. Основні компоненти програми для ОС з графічним інтерфейсом. *nikolay.in.ua*: вебсайт. URL: http://nikolay.in.ua/ distantsijne-navchannya/ 8-klas/ 840-osnovni-komponenti-programi-dlya-os-z-grafichnim-interfejsom (дата звернення: 24.04.2021).

11. PHP – найбільш популярна мова для веб програмування. *chili-web.com.ua:* вебсайт. URL: https://chili-web.com.ua/php-5/ (дата звернення: 26.04.2021).

12. jQuery. *astwellsoft.com:* вебсайт. URL: https://astwellsoft.com/uk/blog/tehnology/jquery.html (дата звернення: 29.04.2021).

13. Що таке JQuery? *yoip.com.ua*: вебсайт. URL: http://yoip.com.ua/shho-take-jquery/ (дата звернення: 03.05.2021).

14. Що таке MYSQL як і де використовують MYSQL: *ruszura.in.ua*: вебсайт. URL: http://ruszura.in.ua/uncategorized/scho-take-mysql-yak-i-de-vykorystovuyut-mysql.html (дата звернення: 06.05.2021).

15. Що таке БД MySql. *uk.photo-555.com:* вебсайт. URL: https://uk.photo-555.com/2686691-what-is-mysql-database (дата звернення: 06.05.2021).

ДОДАТКИ

Додаток А

Програмний код веб-застосунку

Лістинг А1 – Розмітка хеадера сайту

```
<body>
<header class="header">
    <div class="container">
        <div class="nav">
            <div class="nav-logo">
                <img src="images/logo.jpg" alt="logo">
            </div>
            <nav class="nav-menu">
                <ul class="nav-menu__list">
                    <li class="nav-menu__item">
                        <a class="nav__link"
href="index.php">Профіль</a>
                    </li>
                    <li class="nav-menu__item">
                        <a class="nav__link" href="my-
records.php">Моя присутність</a>
                    </li>
                </ul>
            </nav>
            <div class="user-nav">
                <a href="logout.php" class="user-
nav__link">Вийти</a>
            </div>
        </div>
    </div>
</header>
```

Лістинг А2 – Реалізація сторінки «Профіль»

```
<?php require_once("header.php") ?>
<div class="container">
<div class="profile">
<?php
    $query = "SELECT * FROM workers INNER JOIN additional_info
ON workers.worker_id = additional_info.worker_id WHERE
workers.worker_id =".$_SESSION['worker_id'];
    $statement = $connect->prepare($query);
    if($statement->execute()){
        $rows_count = $statement->rowCount();
        if ($rows_count > 0){
            $rows = $statement->fetchAll();
```

```
            foreach ($rows as $row) { ?>
                <div class="profile-image">
                    <img height='400'
src="<?php echo
base64_encode($row['photo']); ?>"/>
                </div>
                <div class="profile-info">
                    <div class="profile-name">
                        <?php
                            echo "<p> ПІП: </p> <p>";
                            echo $row['last_name']."
".$row['patronymic']." ".$row['first_name'];
                            echo "</p>";
                        ?>
                    </div>
                    <div class="profile-email">
                        <p>Електронна адреса: </p>
                        <p><?php echo $row['email'] ?></p>
                    </div>

                    <div class="profile-number">
                        <?php
                            if(!empty($row['phone_number'])){
                                echo "<p>"."Номер
телефону:"."</p>";
                                echo "<p>".
$row['phone_number'] ."</p>";
                            }
                        ?>
                    </div>
                    <div class="profile-position">
                        <?php
                            if(!empty($row['position'])){
                                echo "<p>"."Посада: "."</p>";
                                echo "<p>". $row['position']
."</p>";
                            }
                        ?>
                            <?php } ?>
                        <?php } ?>
                    <?php } ?>
                    </div>
                </div>
    </div>
<table class="table profile-table">
    <thead>
        <tr>
            <th>Н/п</a></th>
            <th>Дата</a></th>
```

```
                <th>Прибуття</th>
                <th>Час прибуття</th>
                <th>Відхід</th>
                <th>Час відходу</th>
            </tr>
        </thead>
        <tbody>

            <?php
                $query = "SELECT * FROM records INNER JOIN workers
ON records.worker_id = workers.worker_id WHERE
records.worker_id=".$_SESSION['worker_id']." ORDER BY
record_id desc LIMIT 5";
                $statement = $connect->prepare($query);
                $selectedUser = 0;
                if($statement->execute()){
                    $rows_count = $statement->rowCount();
                    if ($rows_count > 0){
                        $rows = $statement->fetchAll();
                        $index = 0;
                        foreach ($rows as $row) { $index += 1;?>
                        <tr>
                            <td> <?php echo $index ?> </td>
                            <td> <?php echo $row['date']; ?> </td>
                            <td> <?php

if($row['status_arrival']=='1'){echo "<span
style='color:green;'>&#10004;</span>"; }
                                    else{echo "<span
style='color:red;'>&#10006;</span>"; }
                                    ?>
                            </td>
                            <td> <?php echo $row['time_arrival'];
?> </td>
                            <td>
                                <?php

if($row['status_leaving']=='1'){echo "<span
style='color:green;'>&#10004;</span>"; }
                                    else{echo "<span
style='color:red;'>&#10006;</span>"; }
                                    ?>
                            </td>
                            <td> <?php echo $row['time_leaving'];
?> </td>
                        </tr>
                        <?php } ?>
                    <?php } ?>
                <?php }; ?>
```

```
    </tbody>
</table>
</div>
```

Лістинг А3 – Реалізація сторінки «Моя присутність»

```
<div class="container">
<div id="records-table">
<table class="table records-table">
    <thead>
        <tr>
            <th>Н/п</a></th>
            <th>Дата</a></th>
            <th>Прибуття</th>
            <th>Час прибуття</th>
            <th>Відхід</th>
            <th>Час відходу</th>
        </tr>
    </thead>
    <tbody>
        <?php
            $query = "SELECT * FROM records
                        INNER JOIN workers ON
records.worker_id = workers.worker_id
                        WHERE
records.worker_id=".$_SESSION['worker_id'];
            $statement = $connect->prepare($query);
            $selectedUser = 0;
            if($statement->execute()){
                $rows_count = $statement->rowCount();
                if ($rows_count > 0){
                    $rows = $statement->fetchAll();
                    $index = 0;
                    foreach ($rows as $row) { $index += 1;?>
                    <tr>
                        <td> <?php echo $index ?> </td>
                        <td> <?php echo $row['date']; ?> </td>
                        <td> <?php

if($row['status_arrival']=='1'){echo "<span
style='color:green;'>&#10004;</span>"; }
                                    else{echo "<span
style='color:red;'>&#10006;</span>"; }
                                    ?>
                        </td>
                        <td> <?php echo $row['time_arrival'];
?> </td>
                        <td>
                            <?php
```

```
if($row['status_leaving']=='1'){echo "<span
style='color:green;'>&#10004;</span>"; }
                                    else{echo "<span
style='color:red;'>&#10006;</span>"; }
                                    ?>
                            </td>
                            <td> <?php echo $row['time_leaving'];
?> </td>
                        </tr>
                        <?php } ?>
                    <?php } ?>
                <?php }; ?>
    </tbody>
</table>
</div>
</div>
```

Лістинг А4 – Програмний код сторінки даних працівників

```
<div class="container">
    <table class="table workers-table">
        <thead>
        <tr>
              <th>Фото</th>
              <th>Прізвище</th>
              <th>Ім'я</th>
              <th>По батькові</th>
              <th>Електронна пошта</th>
              <th></th>
              <th></th>
        </tr>
        </thead>
        <tbody>
           <?php
                $query = "SELECT * FROM workers ";
                $statement = $connect->prepare($query);
                $selectedUser = 0;
                if($statement->execute()){
                 $rows_count = $statement->rowCount();
                    if ($rows_count > 0){
                        $rows = $statement->fetchAll();
                        foreach ($rows as $row) { ?>
                        <tr>
                            <td><img width='100'
src="<?php echo
base64_encode($row['photo']);?>"/></td>
                        <td> <?php echo $row['last_name'];
?> </td>
```

```
                                <td> <?php echo
$row['first_name']; ?> </td>
                                <td> <?php echo
$row['patronymic']; ?> </td>
                                <td> <?php echo $row['email']; ?>
</td>
                                <td><a href="update-worker-
action.php?edit=<?php echo $row['worker_id'];?>" class="edit-
worker"><span style="color: green; ">&#x0270E;</span></a></td>
                                <td><a href="delete-
worker.php?delete=<?php echo $row['worker_id'];?>"
class="delete-worker"><span style="color:
red;">&#10007;</span></a></td>
                            </tr>
                            <?php } ?>
                        <?php } ?>
                    <?php }; ?>
            </tbody>
        </table>
    </div>
```

Лістинг А5 – Реалізація сторінки редагування даних працівника

```
<div class="action">
<div class="container">
<form action="update-worker.php" method="post"
enctype="multipart/form-data">
    <div class="form-action">
        <?php if($_SERVER["REQUEST_METHOD"] == "GET"){
            if(isset($_GET['edit'])){
            $query = "SELECT * FROM workers
                        INNER JOIN additional_info ON
workers.worker_id = additional_info.worker_id
                        WHERE workers.worker_id
=".$_GET['edit'];
            $statement = $connect->prepare($query);
            if($statement->execute()){
                $rows_count = $statement->rowCount();
                if ($rows_count > 0){
                    $rows = $statement->fetchAll();
                    $row = $rows[0]; ?>
                    <div class="action-header">
                        <p class="action-title"> <?php echo
$row['last_name']." ".$row['first_name']."
".$row['patronymic']; ?></p>
                    </div>
                        <div class="action-body">
                            <div class="action-fields">
```

```
                                    <div class="error">
                                        <?php
if(isset($_SESSION['update_error'])){
                                            echo
$_SESSION['update_error'];

unset($_SESSION['update_error']);
                                        }?>
                                    </div>
                                    <input type="hidden"
name="worker_id" value="<?php echo $row['worker_id']; ?>">
                                    <input type="hidden"
name="current_password" value="<?php echo $row['password'];
?>">
                                    <div>
                                        <label class="input-
label">Прізвище</label>
                                        <input name="last_name"
type="text" class="last-name" value="<?php echo
$row['last_name']; ?>">
                                    </div>
                                    <div>
                                        <label class="input-
label">Ім'я</label>
                                        <input name="first_name"
type="text" class="first-name" value="<?php echo
$row['first_name']; ?>">
                                    </div>
                                    <div>
                                        <label class="input-
label">По батькові</label>
                                        <input name="patronymic"
type="text" class="patronymic" value="<?php echo
$row['patronymic']; ?>">
                                    </div>
                                    <div>
                                        <label class="input-
label">Електронна пошта</label>
                                        <input name="email"
type="email" class="email" value="<?php echo $row['email'];
?>">
                                    </div>
                                    <div>
                                        <label class="input-
label">Старий пароль</label>
                                        <input name="old_password"
class="password" type="password">
                                    </div>
                                    <div>
```

```
                                    <label class="input-
label">Новий пароль </label>
                                    <input name="new_password"
class="password" type="password">
                                </div>
                                <div>
                                    <label class="input-
label">Завантажити фото</label>
                                    <input name="avatar"
type="file" class="upload-file">
                                </div>
                                <div>
                                    <label class="input-
label">Номер телефону</label>
                                    <input name="phone_number"
type="text" class="phone-number" value="<?php echo
$row['phone_number']; ?>">
                                </div>
                                <div>
                                    <label class="input-
label">Посада</label>
                                    <input name="position"
type="text" class="position" value="<?php echo
$row['position']; ?>">
                                </div>
                                <div>
                                    <label class="input-
label">Стать</label>
                                    <input name="state"
type="text" class="state" value="<?php echo $row['state'];
?>">
                                </div>
                            </div>
                        </div>
                        <div class="action-worker-image">
                            <img
src="<?php echo
base64_encode($row['photo']);?>"/>
                        </div>
                        <div class="action-footer">
                            <input type="submit"
name="button_action" class="button-action" value="Зберегти
зміни"/>
                            <input type="submit"
name="button_cancel" class="button-close" value="Скасувати">
                        </div>
                        <?php } } ?>
                    </div>
                </form>
```

```
            </div>
        </div>
        <?php }else{ header("location:index.php");
        }
    }else{ header("location:index.php");
} ?>
```

Лістинг 6 – Перевірка валідності відредагованих даних

```
<?php
    include_once("connection.php");
    session_start();
    $_SESSION["update_error"] = "";
?>

<?php
    if (isset($_POST['button_cancel'])){
        header("location: index.php");
    }

    if (isset($_POST['button_action'])){
        $error = "";
        $worker_id = $_POST['worker_id'];
        $first_name = trim($_POST['first_name']);
        $last_name = trim($_POST['last_name']);
        $patronymic = trim($_POST['patronymic']);
        $email =  trim($_POST['email']);
        $current_password = $_POST['current_password'];
        $old_password = $_POST['old_password'];
        $new_password = $_POST['new_password'];

        $image = $_FILES['avatar']['tmp_name'];
        $blob = addslashes(file_get_contents($image));

        $phone_number = trim($_POST['phone_number']);
        $position = trim($_POST['position']);
        $state = trim($_POST['state']);

        $isImage = false;
        if (empty($first_name)){
            $error = "Поле \"Ім'я\" не може бути пустим";
            $_SESSION["update_error"] = $error;
            header("location: update-worker-
action.php?edit=$worker_id");
        }
        if (empty($last_name)){
            $error = "Поле \"Прізвище\" не може бути пустим";
            $_SESSION["update_error"] = $error;
```

```
            header("location: update-worker-
action.php?edit=$worker_id");
        }
        if (empty($patronymic)){
            $error = "Поле \"По батькові\" не може бути
пустим";
            $_SESSION["update_error"] = $error;
            header("location: update-worker-
action.php?edit=$worker_id");
        }
        if (empty($email)){
            $error = "Поле \"Електронна пошта\" не може бути
пустим";
            $_SESSION["update_error"] = $error;
            header("location: update-worker-
action.php?edit=$worker_id");
        }
        if (!empty($old_password) && !empty($new_password) ){
            if (md5($old_password) == $current_password){
                $current_password = md5($new_password);
            }else {
                $error = "Паролі не співпадають";
                $_SESSION["update_error"] = $error;
                header("location: update-worker-
action.php?edit=$worker_id");
            }
        }
        if ($_FILES['avatar']['name'] === ''){
            header("location: update-worker-
action.php?edit=$worker_id");
        }else if ($_FILES['avatar']['size'] > 16777214) {
            $error = "Зображення занадто велике";
            $_SESSION["update_error"] = $error;
            header("location: update-worker-
action.php?edit=$worker_id");
        }else if
(!preg_match('/.*\.jpg$/',$_FILES['avatar']['name'])){
            $error = "Файл некоректний!(доступні розширення:
jpeg, png)";
            $_SESSION["update_error"] = $error;
            header("location: update-worker-
action.php?edit=$worker_id");
        }else {
            $isImage = true;
        }
        if($error == ""){
            if ($isImage == true) {
                $query = "UPDATE ` workers ` SET
`first_name`='$first_name', `last_name`='$last_name'
```

```
,`patronymic`='$patronymic', `email`='$email',
`photo`='$blob',`password`='$current_password' WHERE
`worker_id`=$worker_id";
                $statement = $connect->prepare($query);
                $statement->execute();
            }else{
                $query = "UPDATE ` workers ` SET
`first_name`='$first_name', `last_name`='$last_name'
,`patronymic`='$patronymic', `email`='$email',
`password`='$current_password' WHERE `worker_id`= $worker_id";
                $statement = $connect->prepare($query);
                $statement->execute();
            }
            $query = "UPDATE `additional_info` SET
`phone_number`='$phone_number', `position`='$position'
,`state`='$state' WHERE `worker_id`=$worker_id";
            $statement = $connect->prepare($query);
            $statement->execute();
            header("location: update-worker-
action.php?edit=$worker_id");
        }
    }
?>
```

Лістинг А7 – Обробка події сортування за допомогою jQuery

```
$(document).on('click', '.column-sort', function(){
    let column_name = $(this).attr("id");
    let order = $(this).data("sort");
    let arrow = '';

    if(order == 'desc'){
    arrow = '<span> &#9651;</span>';
    }else {
    arrow = '<span> &#9661;</span>';
    }

    $.ajax({
        url:"sort-records-table.php",
        method:"POST",
        data:{column_name:column_name, order:order},
        success:function(data)
        {
                $('#records-table').html(data);
                $('#'+column_name+'').append(arrow);
        },
    })
});
```

Лістинг А8 – Реалізація сортування таблиці

```
<?php
 include_once("connection.php") ;

 $output = '';

 $order = $_POST["order"];
 if($order == 'desc'){
      $order = 'asc';
 } else {
      $order = 'desc';
 }

 $query = "SELECT * FROM records INNER JOIN workers ON
records.worker_id = workers.worker_id ORDER BY
".$_POST["column_name"]." ".$_POST["order"]."";

 $output .= '
        <table class="table records-table">
        <thead>
            <tr>
                <th><a href="#" id="last_name" class="column-
sort" data-sort="'.$order.'">ПІП</a></th>
                <th><a href="#" id="date" class="column-sort"
data-sort="'.$order.'">Дата</a></th>
                <th>Прибуття</th>
                <th>Час прибуття</th>
                <th>Відхід</th>
                <th>Час відходу</th>
                <th></th>
                <th></th>

            </tr>
        <thead>

 ';

 $output .= '<tbody>';

 $statement = $connect->prepare($query);

 if ($statement->execute()){
    if($statement->rowCount() > 0){
        $rows = $statement->fetchAll();
        foreach ($rows as $row) {
            $output .= '
```

```
            <tr>
                <td>' .$row['last_name'] ." ".
substr($row['first_name'], 0,
2).".".substr($row['patronymic'], 0, 2). ".".'</td>
                <td>' .$row['date'].'</td>
                <td>' .$row['status_arrival'].'</td>
                <td>' .$row['time_arrival'].'</td>
                <td>' .$row['status_leaving'].'</td>
                <td>' .$row['time_leaving'].'</td>
                <td><a href="#" class="edit-record" data-
record-id="'.$row['record_id'].'"><span style="color:
green;">&#x0270E;</span></a></td>
                <td><a href="#" class="delete-record" data-
record-id="'.$row['record_id'].'"><span style="color:
red;">&#10007;</span></a></td>
            </tr>
            ';
        }
    }
}
 $output .= '</table>';
 echo $output;
 ?>
```

Лістинг А9 – Обробка події натискання кнопки редагування та видалення запису

```
$(document).on('click', '.edit-record', function(){
    $(this).toggleClass("disabled");
    let record_id = $(this).data("record-id");
    let row_values = $(this).closest("tr").children();

    let edit_row ="";
    edit_row += "<tr>";
    edit_row += "<td></td>";
    edit_row += "<td><input name='date'  type=\"text\"
value=\""+row_values.eq(1).html()+"\"/></td>";
    edit_row += "<td><input name='status_arrival'
type=\"text\" value=\""+row_values.eq(2).html()+"\"/></td>";
    edit_row += "<td><input name='time_arrival' type=\"text\"
value=\""+row_values.eq(3).html()+"\"/></td>";
    edit_row += "<td><input name='status_leaving'
type=\"text\" value=\""+row_values.eq(4).html()+"\"/></td>";
    edit_row += "<td><input name='time_leaving' type=\"text\"
value=\""+row_values.eq(5).html()+"\"/></td>";
    edit_row += "<td><a href=\"#\" class=\"save-edit\" data-
record-id=\""+record_id+"\">OK </a></td>"
```

```
    edit_row += "<td><a href=\"#\" class=\"cancel-edit\" data-
record-id=\""+record_id+"\">Скасувати</a></td>";
    edit_row += "</tr>";

    $(this).closest("tr").after(edit_row);
});

$(document).on('click', '.cancel-edit', function(){
    let record_id = $(this).data("record-id");
    $("a[data-record-
id='"+record_id+"'").toggleClass("disabled");
    $(this).closest("tr").remove();
});
$(document).on('click', '.save-edit', function(){
    let record_id = $(this).data("record-id");
    let row_values = $(this).closest("tr").find( "input" );
        $.ajax({
        url:"edit-record.php",
        method:"POST",
        data:{
            record_id:record_id,
            date:row_values.eq(0).val(),
            status_arrival: row_values.eq(1).val(),
            time_arrival: row_values.eq(2).val(),
            status_leaving: row_values.eq(3).val(),
            time_leaving: row_values.eq(4).val()
        },
        success:function()
        {
                location.href = "<?php echo $baseUrl;?>record-
action.php";
        },
    })
});
```

Додаток Б

Реалізація програм на Python

Лістинг Б1 – Програмний код face-processing

```
def login():
    global login_screen
    global username_entered
    global password_entered
    global username_login_entry
    global password_login_entry

    login_screen = Tk()
    login_screen.title("Обробка обличчя")
    login_screen.geometry("280x180")

    Label(login_screen, text="").pack()

    username_entered = StringVar()
    password_entered = StringVar()

    Label(login_screen, text="Електронна пошта: ").pack()
    username_login_entry = Entry(login_screen,
textvariable=username_entered)
    username_login_entry.pack()
    Label(login_screen, text="Пароль:  ").pack()
    password_login_entry = Entry(login_screen,
textvariable=password_entered, show='*')
    password_login_entry.pack()
    Label(login_screen, text="").pack()
    Button(login_screen, text="Підтвердити", width=10,
height=1, command=login_verify).pack()
    login_screen.mainloop()

def login_verify():
    global workerId

    username = username_entered.get()
    password = password_entered.get()
    username_login_entry.delete(0, END)
    password_login_entry.delete(0, END)

    data = getLoginPasswordList()
    workerId = -1
    res = hashlib.md5(password.encode())
    encryptedPassword = res.hexdigest()
    for d in data:
        if d[1] == username:
```

```
                if d[2] == encryptedPassword:
                    workerId = d[0]
                    break
        if workerId == -1:
            user_data_incorrect()
        else:
            checkUnencodedWorkerFace(workerId)

    def user_data_incorrect():
        global user_data_incorrect_screen
        user_data_incorrect_screen = Toplevel(login_screen)
        user_data_incorrect_screen.title("Обробка обличчя")
        user_data_incorrect_screen.geometry("250x70")
        Label(user_data_incorrect_screen, text="Користувача не
    знайдено. \nПеревірте правильність введення даних").pack()
        Button(user_data_incorrect_screen, text="OK",
    command=delete_user_data_incorrect_screen).pack()

    def delete_user_data_incorrect_screen():
        user_data_incorrect_screen.destroy()

    def delete_encodings_not_empty_screen():
        encodings_not_empty_screen.destroy()

    def encodings_are_empty():
        global encodings_empty_screen
        encodings_empty_screen = Toplevel(login_screen)
        encodings_empty_screen.title("Обробка обличчя")
        encodings_empty_screen.geometry("250x100")
        Label(encodings_empty_screen, text="Даних із фото немає.
    \nВам необхідно їх обчислити").pack()
        Button(encodings_empty_screen, text="Обчислити",
    command=setEncodings).pack()

    def encodings_not_empty():
        global encodings_not_empty_screen
        encodings_not_empty_screen = Toplevel(login_screen)
        encodings_not_empty_screen.title("Обробка обличчя")
        encodings_not_empty_screen.geometry("180x100")
        Label(encodings_not_empty_screen, text="Дані з фото уже
    вилучені. \n").grid(row=0,column=0,columnspan=2)

        Button(encodings_not_empty_screen, text="Закрити",
    command=delete_encodings_not_empty_screen).grid(row=1,column=0
    ,pady=10, padx=10)
        Button(encodings_not_empty_screen, text="Оновити дані",
    command=updateEncodings).grid(row=1,column=1,pady=10, padx=10)

    def success(message):
```

```
    messagebox.showinfo("info", message)

def error(message):
    messagebox.showerror("error", message)

def detectFace(detector, image, inHeight=300, inWidth=0):
    imageDlibHog = image.copy()
    imageHeight = imageDlibHog.shape[0]
    imageWidth = imageDlibHog.shape[1]

    if not inWidth:
        inWidth = int((imageWidth / imageHeight) * inHeight)

    scaleHeight = imageHeight / inHeight
    scaleWidth = imageWidth / inWidth

    imageDlibHogSmall = cv2.resize(imageDlibHog, (inWidth,
inHeight))
    imageDlibHogSmall = cv2.cvtColor(imageDlibHogSmall,
cv2.COLOR_BGR2RGB)

    faceRects = detector(imageDlibHogSmall, 0)
    borders = []
    for faceRect in faceRects:
        cvRect = [
            int(faceRect.left() * scaleWidth),
            int(faceRect.top() * scaleHeight),
            int(faceRect.right() * scaleWidth),
            int(faceRect.bottom() * scaleHeight),
        ]
        borders.append(cvRect)
        cv2.rectangle(
            imageDlibHog,
            (cvRect[0], cvRect[1]),
            (cvRect[2], cvRect[3]),
            (0, 248, 252),
            1
        )
    return imageDlibHog, borders

def getEncodingsLocations(image):
    hogFaceDetector = dlib.get_frontal_face_detector()

    outDlibHog, faceBorders = detectFace(hogFaceDetector,
image)

    imageSmall = cv2.resize(image, (0, 0), None, 0.25, 0.25)
    imageSmall = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
    encFaces = []

    for eachFace in faceBorders:
        currFaceLocation = [(eachFace[1], eachFace[2],
eachFace[3], eachFace[0])]
        encCurrFace =
face_recognition.face_encodings(imageSmall,
known_face_locations=currFaceLocation)
        encFaces.append(encCurrFace[0])
    return encFaces, faceBorders

class NumpyArrayEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return json.JSONEncoder.default(self, obj)

def getImage(worker_id, photo_name):
    try:
        SQLStatement = "SELECT photo FROM workers WHERE
worker_id = %s"
        mycursor.execute(SQLStatement, (worker_id,))
        record = mycursor.fetchall()
        for row in record:
            imageData = row[0]
           error("Помилка зчитування {}".format(error))
with open(photo_name, 'wb') as file:
                file.write(imageData)
            image = cv2.imread(photo_name)
            return image

    except mysql.connector.Error as error:
        print("Помилка зчитування {}".format(error))

def checkUnencodedWorkerFace(workerId):
    try:
        mycursor.execute(f"SELECT count(worker_id) from
recognition_data WHERE worker_id={workerId}")
        result = mycursor.fetchone()

        if result[0] == 0:
            encodings_are_empty()
        else:
            encodings_not_empty()
    except:
        error("Виникла помилка під час пошуку даних")

def setEncodings():
    encodings_empty_screen.destroy()
```

```
    imageFromDb = getImage(workerId, "image.jpg")
    encodings, locations = getEncodingsLocations(imageFromDb)
    try:
        locationList =
f"{locations[0][0]},{locations[0][1]},{locations[0][2]},{locat
ions[0][3]}"
        mycursor = conn.cursor(buffered=True)
        numpyData = {"array": encodings[0]}
        encodedNumpyData = json.dumps(numpyData,
cls=NumpyArrayEncoder)

        SQLStatement = "INSERT INTO recognition_data
(worker_id, face_encodings, face_location) VALUES (%s,%s,%s)"
        mycursor.execute(SQLStatement, (workerId,
encodedNumpyData, locationList))
        conn.commit()
        success("Збережено")
    except:
        error("Не вдалось зберегти. Зображення некоректне")

def updateEncodings():
    encodings_not_empty_screen.destroy()
    imageFromDb = getImage(workerId, "image.jpg")
    encodings, locations = getEncodingsLocations(imageFromDb)
    try:
        locationList =
f"{locations[0][0]},{locations[0][1]},{locations[0][2]},{locat
ions[0][3]}"
        mycursor = conn.cursor(buffered=True)
        numpyData = {"array": encodings[0]}
        encodedNumpyData = json.dumps(numpyData,
cls=NumpyArrayEncoder)
        SQLStatement = "UPDATE recognition_data SET
face_encodings=%s, face_location=%s WHERE worker_id=%s"
        mycursor.execute(SQLStatement, (encodedNumpyData,
locationList, workerId))
        conn.commit()
        success("Оновлено")
    except:
        error("Не вдалось оновити. Зображення некоректне")

def getLoginPasswordList():
    try:
        mycursor = conn.cursor()
        mycursor.execute("SELECT worker_id, email, password
FROM workers")
        rows = mycursor.fetchall()
        return rows
    except Error as e:
```

```
        error(e)

try:
    global conn
    global mycursor

    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="",
        database="worker-attendance"
    )
    mycursor = conn.cursor()
    login()

except Error as e:
    error(e)

finally:
    if conn.is_connected():
        mycursor.close()
        conn.close()
```

Лістинг Б2 – Програмний код mark-presence

```
def detectFace(detector, image, inHeight=300, inWidth=0):
    imageDlibHog = image.copy()
    imageHeight = imageDlibHog.shape[0]
    imageWidth = imageDlibHog.shape[1]

    if not inWidth:
        inWidth = int((imageWidth / imageHeight) * inHeight)

    scaleHeight = imageHeight / inHeight
    scaleWidth = imageWidth / inWidth

    imageDlibHogSmall = cv2.resize(imageDlibHog, (inWidth,
inHeight))
    imageDlibHogSmall = cv2.cvtColor(imageDlibHogSmall,
cv2.COLOR_BGR2RGB)

    faceRects = detector(imageDlibHogSmall, 0)
    borders = []
    for faceRect in faceRects:
        cvRect = [
            int(faceRect.left() * scaleWidth),
            int(faceRect.top() * scaleHeight),
```

```
                int(faceRect.right() * scaleWidth),
                int(faceRect.bottom() * scaleHeight),
            ]
            borders.append(cvRect)
            cv2.rectangle(
                imageDlibHog,
                (cvRect[0], cvRect[1]),
                (cvRect[2], cvRect[3]),
                (0, 255, 0),
                1
            )
        return imageDlibHog, borders

    def compareEncodings(encodingsList1, encodingsList2):
        tolerance = 0.6
        matchesList = []
        for encodings in encodingsList1:
            faceDistance =
    face_recognition.api.face_distance(encodingsList2, encodings)
            faceMatches = list(faceDistance <= tolerance)
            matchesList.append(faceMatches)
        return matchesList

    def getEncodingsFromDb(workerId):
        try:
            mycursor = conn.cursor()

            SQLStatement = f"SELECT face_encodings FROM
    recognition_data WHERE worker_id = {workerId}"
            mycursor.execute(SQLStatement, workerId)

            for row in mycursor:
                strList = json.loads(row[0])
                print(strList)
                strListElement = strList["array"]
                numpyEncodings = [np.array(strListElement)]
                return numpyEncodings

        except Error as e:
            print(e)

    def realTimeRecognition(workerIds, encodeListKnown):
        cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)

        hogFaceDetector = dlib.get_frontal_face_detector()

        while True:
            hasFrame, frame = cap.read()
```

```
        if not hasFrame: break

        outDlibHog, faceBorders = detectFace(hogFaceDetector,
frame)

        imageSmall = cv2.resize(frame, (0, 0), None, 0.25,
0.25)
        imageSmall = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        foundIds = []

        for eachFace in faceBorders:
            facesCurFrame = [(eachFace[1], eachFace[2],
eachFace[3], eachFace[0])]

            encodesCurFrame =
face_recognition.face_encodings(imageSmall,
known_face_locations=facesCurFrame)
            compared = compareEncodings(encodesCurFrame,
encodeListKnown)
            for r in range(len(compared)):
                for c in range(len(compared[r])):
                    if compared[r][c]:
                        foundIds.append(workerIds[c])
            cv2.imshow("Пошук обличчя...", outDlibHog)

            if len(foundIds):

                cv2.waitKey(500)
                cap.release()
                cv2.destroyAllWindows()
                return foundIds
        k = cv2.waitKey(5)
        if k == 27:
            break
    cap.release()
    cv2.destroyAllWindows()
    return []


def success(message):
    messagebox.showinfo("info", message)

def error(message):
    messagebox.showerror("error", message)

def arrive():
    realTimeMark("arrive")
```

```
def leave():
    realTimeMark("leave")

def markAction():
    global mark_screen
    mark_screen = Tk()
    mark_screen['bg'] = '#87b3fa'
    mark_screen.title("")
    label = Label(mark_screen, text="ВІДМІТИТИСЬ?")
    label.grid(row=0,column=0,columnspan=2,pady=10, padx=10)
    b1 = Button(mark_screen, text="ПРИБУВ", width=10,
height=1, command=arrive)
    b2 = Button(mark_screen, text="ПОКИДАЮ", width=10,
height=1, command=leave)
    b1.grid(row=1,column=0,pady=10, padx=10)
    b2.grid(row=1,column=1,pady=10, padx=10)
    mark_screen.mainloop()

def realTimeMark(action):
    encodings = []
    workerIds = []
    try:
        global conn
        global mycursor
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            passwd="",
            database="worker-attendance"
        )
        mycursor = conn.cursor()
        mycursor.execute("SELECT worker_id FROM
recognition_data")
        rows = mycursor.fetchall()

        for row in rows:
            encoding = getEncodingsFromDb(row[0])[0]
            encodings.append(encoding)
            workerIds.append(row[0])
        foundWorkers = realTimeRecognition(workerIds,
encodings)
        now = datetime.now()
        currentDay = now.strftime("%Y-%m-%d")
        currentTime = now.strftime("%H:%M:%S")
        mycursor.execute("SELECT status_arrival,
status_leaving, record_id FROM records WHERE
date='"+currentDay+"' AND
worker_id='"+str(foundWorkers[0])+"'")
        fetchedStatus = mycursor.fetchone()
```

```
        if fetchedStatus is not None:
            isArrived = True if fetchedStatus[0] == 1 else
False
            isLeaved = True if fetchedStatus[1] == 1 else
False
            if action == "arrive":
                if not isArrived:
                    SQLStatement = "UPDATE `records` SET
`status_arrival`= '1', `time_arrival`='%s WHERE
`record_id`=%s"
                    mycursor.execute(SQLStatement,
(currentTime,str(fetchedStatus[2])))
                    conn.commit()
            elif action == "leave":
                if not isLeaved:
                    SQLStatement = "UPDATE `records` SET
`status_leaving`= '1', `time_leaving`=%s WHERE `record_id`=%s"
                    mycursor.execute(SQLStatement,
(currentTime, str(fetchedStatus[2])))
                    conn.commit()
        else:
            SQLStatement = "INSERT INTO `records`
(`worker_id`, `status_arrival`, `date`, `time_arrival`,
`status_leaving`, `time_leaving`) VALUES (%s, %s, %s, %s, %s,
%s)"
            if action == "arrive":
                mycursor.execute(SQLStatement,
(foundWorkers[0], '1', currentDay, currentTime, '0',
'00:00:00'))
                conn.commit()
            elif action == "leave":
                mycursor.execute(SQLStatement,
(foundWorkers[0], '0', currentDay, '00:00:00', '1',
currentTime))
                conn.commit()

    except Error as e:
        error(e)
    finally:
         mycursor.close()
         conn.close()
markAction()
```

Додаток В

Техніко-економічні обчислення

Таблиця В1 – Податок на доходи фізичних осіб

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	6800×18%=1224 грн
Фронтенд-розробник	18000×18%=3240грн
Бекенд-розробник	23000×18%=4140 грн
Дизайнер	9000×18%=1620 грн
Пайтон-розробник	14000×18%=2520 грн
Тестувальлник	9000×18%=1620 грн

Таблиця В2 – Військовий збір

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	6800×1,5%=102 грн
Фронтенд-розробник	18000×1,5%=270грн
Бекенд-розробник	23000×1,5%=345 грн
Дизайнер	9000×1,5%=135 грн
Пайтон-розробник	14000×1,5%=210 грн
Тестувальлник	9000×1,5%=135 грн

Таблиця В3 – Єдиний внесок

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	6800×22%=1496 грн
Фронтенд-розробник	18000×22%=3960 грн
Бекенд-розробник	23000×22%=5060 грн
Дизайнер	9000×22%=1980 грн
Пайтон-розробник	14000×22%=3080 грн
Тестувальлник	9000×22%=1980 грн

Таблиця В4 – Утримання

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	1326 грн (1224 грн+102 грн)
Фронтенд-розробник	3510 грн (3240 грн+270 грн)
Бекенд-розробник	4485 грн (4140 грн+345 грн)
Дизайнер	1755 грн (1620 грн+135 грн)
Пайтон-розробник	2730 грн (2520 грн+210 грн)
Тестувальлник	1755 грн (1620 грн+135 грн)

Таблиця В5 – Виплати працівникам

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	5474 грн (6800 грн-1326 грн)
Фронтенд-розробник	14490 грн (18000 грн-3510 грн)
Бекенд-розробник	18510 грн (23000 грн-4485 грн)
Дизайнер	7245 грн (9000 грн-1755 грн)
Пайтон-розробник	11270 грн (14000 грн-2730 грн)
Тестувальлник	7245 грн (9000 грн-1755 грн)