Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____
(рівень вищої освіти)

Дослідження методів аналізу обліку відвідування занять студентами
(тема)

Виконав: студент 2 курсу, групи ІПЗм-17-2__
спеціальності 121- Інженерія програмного забезпечення_____
(код і повна назва спеціальності)
Освітньо-наукової програми
Інженерія програмного забезпечення
(повна назва освітньої програми)

_____ Гонтар В.О._____
(прізвище, ініціали)

Керівник _____проф. Четвериков Г. Г._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук_____

Кафедра програмної інженерії_____

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність _121– Інженерія програмного забезпечення_____
(код і повна назва)

Освітньо-наукової програми Інженерія програмного забезпечення__
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _____Гонтарю Владиславу Олександровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів перевірки присутності студентів затверджена наказом по університету від "___" _____20 ___ р № _____
заповнюється вручну після отримання наказу

2. Термін подання студентом роботи до екзаменаційної комісії
від "___" _____20 ___ р

3. Вихідні дані до роботи *методи перевірки присутності студентів, пояснювальна записка, програмна система для первірки присутності студентів з мобільним пристроєм. Використовувати ОС Windows, середовище об'єктно-орієнтованого проектування*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд методів перевірки присутності студентів, використання мобільних телефонів при проектуванні, реалізація програмної системи*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *діаграма послідовностей, діаграма прецедентів, діаграма класів, діаграма розгортання, схеми архітектури системи, структура даних, інтерфейс системи, слайди презентації*_____

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	Четвериков Г.Г.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз проблемної області		
2	Розробка моделі предметної галузі		
3	Розробка структури зберігання даних даних		
4	Створення коду програми		
5	Тестування і налагодження програми		
6	Підготовка пояснювальної записки		
7	Підготовка презентації та доповіді		
8	Попередній захист		
9	Нормоконтроль, рецензування		
10	Занесення диплома в електронний архів		
11	Допуск до захисту у зав. кафедри		

Дата видачі завдання - - *2019р*.

Студент _____
(підпис)

Керівник роботи _____ __проф. Четвериков Г.Г_____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 72 с., 21 рис., 20 джерел.

Метою роботи є проектування та розробка програмної системи для перевірки присутності студентів.

Методи розробки базуються на технології Java, протоколу передачі даних HTTP, фреймворку Spring для створення веб-застосунків на платформі Java EE, MongoDB базі даних, середовищі розробки Intellij IDEA Community Edition.

Результатом роботи є програмна система, що складається з серверу, веб-клієнта та мобільного пристрою, що допомагає перевіряти пристуність студента, яка дозволить вчителям не витрчати зайвий час на перевірку присутності студентів через автоматизацію цього процесу.

JAVA, MAVEN, СТУДЕНТ, ПЕРЕВІРКА ПРИСУТНОСТІ, SPRING BOOT, ПРОГРАМНА СИСТЕМА, FACE RECOGNITION.

Explanatory note: 72 p., 21 fig., 20 sources.

The purpose of the work is to design and develop a software system to check the presence of students.

The methods of development technologies are based on Java, data transfer protocol HTTP, Spring framework for building web applications on the platform Java EE, MongoDB database, development environment Intellij IDEA Community Edition.

The result of the work is a software system consisting of a server, a web client and a mobile device, which helps to verify the student's attendance, which will allow teachers not to spoil the time to check the presence of students through the automation of this process.

JAVA, MAVEN, STUDENT, CHECK ATTENDANCE, SPRING BOOT, SOFTWARE SYSTEM, FACE RECOGNITION.

ЗМІСТ

Вступ 8
1 АНАЛІЗ ПРОБЛЕМНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ 10
1.1 Концепція перевірки присутності студентів 10
1.2 Аналіз існуючих реалізацій перевірки присутності студентів……12
1.3 Автоматизований метод перевірки присутності студентів.....18
1.4 Постановка задачі.....20
2 Модель ЕЛЕКТРОННОЇ ПЕРЕВІРКИ СТУДЕНТІВ 22
2.1 Комп'ютерний зір 22
2.2 Розпізнання обличчя……27
2.3 Класифікатори……..30
2.3.1 Подібність……31
2.3.2 Вірогідність……32
2.4 Підхід нейронної мережі……35
2.4.1 Нейронні мережі з фільтрами Габора……35
2.4.2 Нечіткі нейронні мережі……36
3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ 38
3.1 UML проектування програмної системи 38
3.2 Архітектура програмної системи 42
4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 44
4.1 Вибір засобів розробки..... 44
4.2 Опис програмної системи 47
4.2.1 Сервер 48
4.2.2 Веб-клієнт 50
4.2.3 Мобільний додаток..... 52.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ..... 58
Висновки..... 61
ПЕРЕЛІК ПОСИЛАНЬ..... 62
ДОДАТОК А Слайди презентації 64
ДОДАТОК Д Електронні матеріали (CD) 72

ВСТУП

У кожному фундаменті підтримка відвідуваності має важливе значення для перевірки продуктивності студентів. Коли в навчальному закладі так багато студентів, буде важче відзначати відвідуваність кожного студента. І це вимагає багато часу. Традиційним методом маркування відвідуваності є система ручного обслуговування, в якій відвідуваність записується і підтримується вручну. Ця система стикається з проблемою втрати часу, а також ускладнюється, коли сила більше. І можна легко маніпулювати. Всякий раз, коли ми повинні вимірювати продуктивність студентів, пошук і розрахунок середньої кількості відвідуваних студентів є складним завданням. Тому ми потребуємо автоматизованої системи маркування та підтримки відвідуваності. Так що ми зможемо легко підтримувати дійсний і правильний запис відвідуваності.

Автоматизація системи відвідуваності надає певні переваги факультету, зменшуючи адміністративне навантаження на персонал. Ця система економить час і також використовується для цілей безпеки. Використовуючи цю систему, ми можемо запобігти фальшивій відвідуваності та проксі-серверам. Пропуск занять без знань персоналу стане важким для студентів шляхом впровадження цієї системи. Багато систем управління відвідуваністю впроваджуються на ринок, таких як система RFID, системи карток перфорації, карткові системи, біометричні системи, які включають аналіз відбитків пальців, аналіз діафрагми тощо. Тому нам потрібна система, яка б відзначала відвідуваність без будь-якого втручання людини. Таким чином, ми запровадимо ефективну систему, яка автоматично відзначатиме відвідуваність студентів, розпізнаючи їхні обличчя, тобто автоматизовану систему відвідування, використовуючи розпізнавання облич.

Автоматизована система відвідуваності, що використовує розпізнавання облич, зазвичай складається з етапів збору зображень, розробки баз даних, розпізнавання облич, попередньої обробки, вилучення об'єктів і класифікації, після

чого виконується етап пост-обробки. По-перше, зображення обличчя кожного студента повинні подаватися в систему і зберігатися в базі даних. Потім відвідуваність студентів записується за допомогою камери, прикріпленої в класі, у відповідному місці, звідки фотокамера може зафіксувати весь клас, що постійно фіксує зображення студентів, виявляє обличчя на зображеннях і порівнює виявлені обличчя з бази даних і відзначити відвідуваність. У певному методі фотокамера фіксується в точці входу в клас, щоб отримати зображення студентів, коли вони входять у клас. За допомогою цієї системи, ми можемо заощадити багато часу, ніж система ручного обслуговування і легко підтримувати запис.

Ключовим фактором підвищення якості освіти є те, що студенти регулярно відвідують заняття. Традиційно студенти стимулюються відвідувати заняття з використанням точок відвідування, які в кінці семестру становлять частину кінцевої оцінки студентів. Проте традиційно це створює додаткові зусилля від вчителя, який повинен правильно відзначити відвідування студентів, що в той же час витрачає значну кількість часу на процес навчання. Крім того, він може стати набагато складнішим, якщо доводиться мати справу з великими групами студентів. Цей документ представляє нову систему автоматичного маркування керування відвідуваністю без будь-якого втручання в регулярний навчальний процес. Система може бути використана також під час екзаменаційних сесій або інших видів навчальної діяльності, де обов'язкове відвідування. Ця система виключає класичну ідентифікацію студентів, наприклад, називаючи імена студентів, або перевіряючи відповідні ідентифікаційні картки, які можуть не тільки заважати навчальному процесу, але також можуть бути стресовими для студентів під час сесій.

1 АНАЛІЗ ПРОБЛЕМНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Концепція перевірки присутності студентів

Управління відвідуваністю студентів є критичним змістом адміністрації студентів шкіл / коледжів / університетів [1]. Традиційний сценарій відстеження відвідуваності студентів у класі здійснюється шляхом зобов'язання студентів фізично відзначати аркуш відвідуваності, що йде навколо класу, а Викладач курсу виступає з промовою. Наприклад, викладач курсу з великим класом може помітити, що турбуватися про те, що лист відвідуваності проходить по класу, а фізичне маркування відвідування студентами є гнітючими і, без сумніву, займають їх від навчання та отримання повної уваги від студентів [2].

Основна увага в цій роботі присвячена застосуванню смартфону в системі відвідуваності. Сьогодні смартфон є дуже поширеним для всіх вчителів, щоб вони могли легко брати участь і обробляти відвідуваність, де вона потребує. Основна перевага цієї системи полягає в тому, що викладач може отримати обчислений відсоток, може надрукувати копію з інформацією про відвідуваність деталей, може зберігати дані в базі даних телефону, а також зберігати дані у віддаленій базі даних сервера, що гарантує, що інформація ніколи не втратить, може використовувати дані, які потрібні.

У деяких університетах професори та лекції беруть участь, викликаючи імена та прізвища студентів, а потім маркуючи їх, а в інших - вчителі проходять навколо аркуша паперу, просячи студентів підписатися на листі присутніх поруч з прізвищами. Обидві практики мають свої недоліки. У першому випадку, якщо на одному уроці відвідують численні групи, перевірка всіх цих учнів на ім'я та прізвище може зайняти більше 10 хвилин з кожного уроку; у другому випадку друзі відсутніх учнів можуть підписати для них поруч зі своїми іменами та прізвищами. Ці практики ставлять професорів університетів, лекції та їхні інститути до значних недоліків, коли йдеться про відвідування. Щоб виправити ці систематичні

недоліки, ми вирішили використовувати мобільні стільникові телефони на платформі Android для оновлення цієї послуги. Кожен професор має свій власний стільниковий телефон, і кожен професор завантажує це мобільне додаток у свій мобільний телефон і додає конкретні дані, пов'язані з заняттями та курсами, які він / вона навчає.

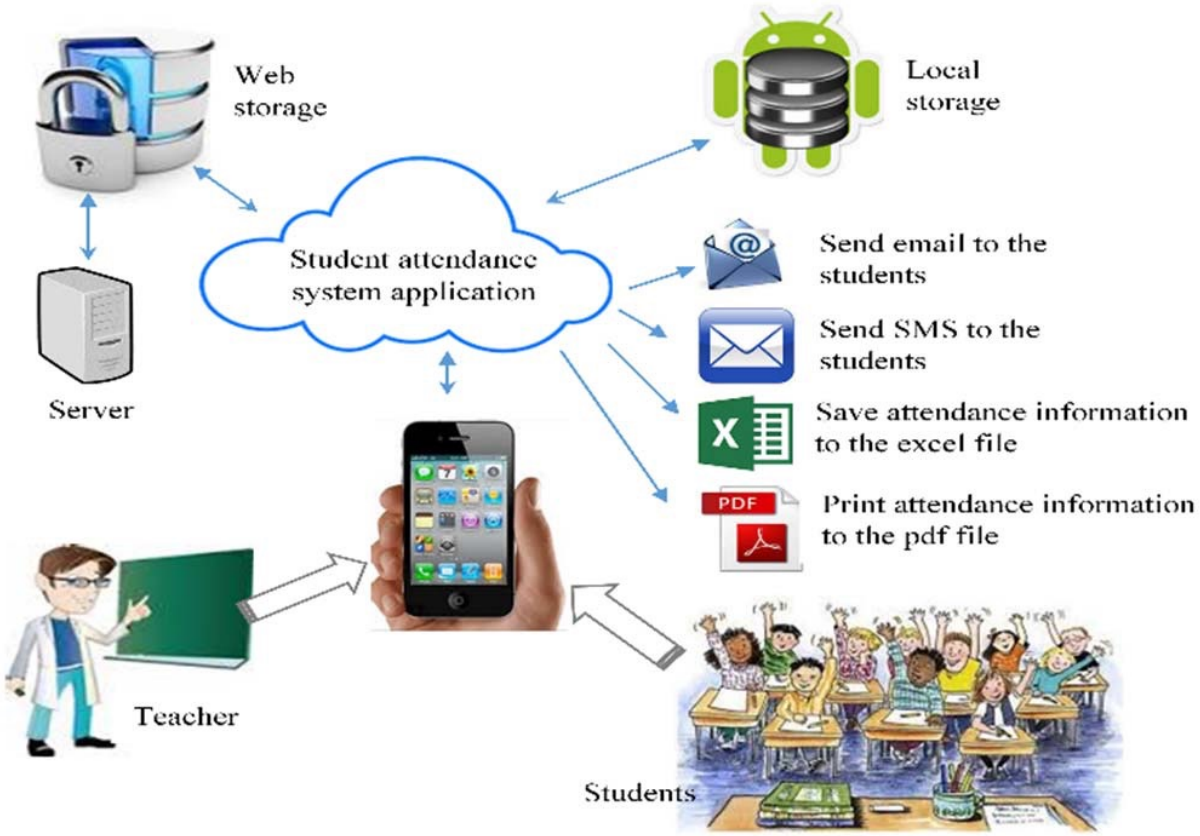

Рисунок 1.1 – Схематична робота додатку

База даних характеризується як збір даних. База даних використовується для зберігання даних, зібраних мобільним додатком. На додаток до додаткових функцій для клієнтів, онлайн-платформа може керувати записом про відвідуваність студентів, запитуючи базу даних. Це включає в себе комп'ютеризовані операції, наприклад, стиснення відвідуваності конкретного студента шляхом визначення показника відвідуваності для унікального курсу. Веб-сервер надає кошти бази даних MySQL. Сервер є віддаленим сховищем записи відвідуваності. Тільки мобільний додаток може спілкуватися з веб-сервером. Додаток може оновлювати

базу даних MySQL і отримувати дані з веб-сервера. У разі втрати даних користувач може відновити з веб-сервера базу даних MySQL. Викладач може змінювати дані учня, але учневі дозволяється вносити зміни в його інформацію.

Відвідуваність кожного студента по кожному курсу встановлюється окремо, виходячи з передумови перебування в класі. Якщо час перебування в класі пов'язано з необхідним часом, то відвідуваність позначається як «присутній». Відвідуваність учнів, які не відвідували заняття протягом навчальної години, позначається як «відсутній». Немає шансів на дублювання запису в системі. Розрахунок відсотків виконується автоматично за допомогою мобільного додатку для кожного учня на кожному з зареєстрованих курсів, щоб перевірити його / її успішність для участі в іспиті. Якщо розрахований відсоток менше необхідного відсотка, студент може бути виключений з університету.

У разі низького відсотка, електронний лист відправляється опікуну студента, включаючи інформацію про відсотки, а також попередження. Таким чином, батьки учнів будуть автоматично отримувати інформацію про прогрес своєї дитини. Для відправки служби коротких повідомлень (SMS) використовується мобільний додаток. Всякий раз, коли учень отримує низький відсоток, його / її опікуну відправляється SMS-повідомлення, щоб повідомити йому про прогрес їхньої дитини в разі, якщо він не може перевірити свою електронну пошту.

1.2 Аналіз існуючих реалізацій перевірки присутності студентів

Університет в центральному Ланкаширі (UCLAN) контролює відвідуваність, оскільки дослідження показали, що регулярне відвідування та академічні досягнення тісно пов'язані, тому важливо знати, що студент регулярно відвідує заняття. Слідкування за відвідуваністю студентів, може допомогти визначити тих, яким потрібна підтримка на ранній стадії, і вжити заходів, щоб допомогти їм продовжити навчання. Моніторинг відвідуваності студентів є обов'язковим для всіх

студентів. Університет очікує, що студент буде відвідувати всі свої обов'язкові сеанси. Кожен студент має спеціальну картку UCLan, яка використовується для реєстрації відвідуваності лекції. Для того, щоб вважатися присутнім, кожен студент має прикласти картку UCLan на передню панель електронного кард-рідера, який знаходиться у кожному класі (Рис. 1.2). Уся ця процедура записує не тільки присутність, але і час зчитування, тобто, на скільки хвилин студент міг запізнитися, що допомагає формувати статистику по кожному студентові не тільки, як присутній або відсутній, але й можливість відслідковувати кількість спізнень, регулярність та інше. це записує вашу присутність у момент зчитування її. Другий важливий момент – це немає ніякого зриву учбового процесу і відволікання вчителя на кожного студента, який спізнився. З нього знімається головна відповідальність перевірки присутності студентів і кожен викладач має змогу концентруватися на одній лише справі – давати лекційний матеріал.

Звісно бувають випадки, що студент загубив або забув картку вдома, Університет в Ланканширі дає можливість кожному вичтелеві доступ до системи SAM, яка з’єднана с карт-рідером у кожній аудиторії, що допоможе вручну додати учня у систему присутності після заняття.

Якщо кабінет, в якому перебуває студент, не має пристрою для читання електронних карток, викладач так само може записати студента до SAM після занять. Майте на увазі, що система SAM має надійну функціональність звітності, яка дозволяє спільно використовувати дані про відвідуваність з науковим керівником кожного студента. Університет розуміє, що час від часу є вагома причина, чому студент не можете відвідувати клас, і в цьому випадку учень повинен зв'язатися зі своїм центром підтримки, щоб повідомити їм, щоб система SAM була відповідно оновлена.

Університет в Честері (University of Chester) розробив додаток, який має скачати кожен студент університету для того, щоб помічати себе, як присутній учень. Студенти отримають кнопку "перевірити" на своєму екрані протягом перших 30 хвилин кожної сесії.

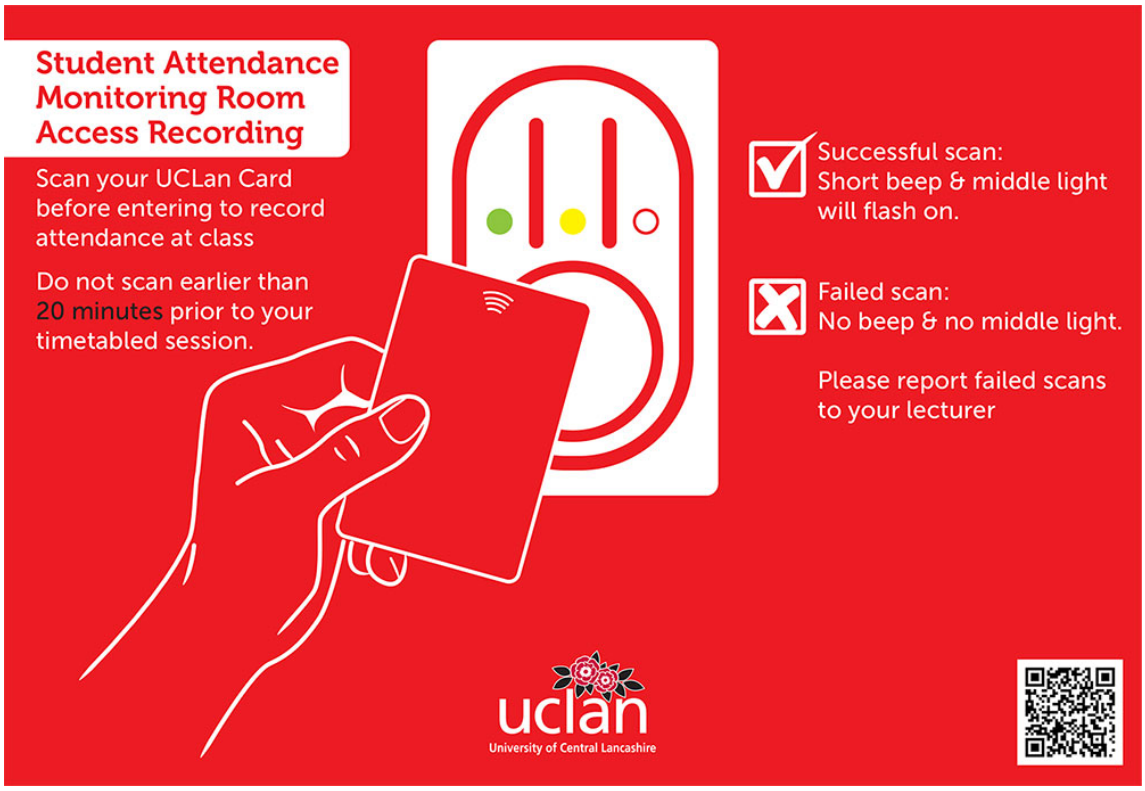

Рисунок 1.2 – Перевірка присутності студента карткою і кард-рідером вбудованим у кожному класі

Університет в Честері (University of Chester) розробив додаток, який має скачати кожен студент університету для того, щоб помічати себе, як присутній учень. Студенти отримають кнопку "перевірити" на своєму екрані протягом перших 30 хвилин кожної сесії. Це буде показано, лише якщо студент перебуває в кімнаті, в якій відбувається заняття у розкладі, щоб бути учасником сесії. Ви просто натискаєте кнопку і підтверджуєте, що ви там. Репетитори нагадують про реєстрацію на початку кожної сесії. Також викладач може входити в систему і відмічати кожного студента вручну. Система використовує GPS, щоб перевірити місцезнаходження учня та місцезнаходження кабінету і якщо студент знаходиться у кабінеті, він має право додати себе, як присутній.

Університет імені Адамса (Adams State University) в Америці та декілька інших навчальних закладах, використовують систему Presence. Кожен учень має студентську картку, яка і використовується для перевірки присутності студента, але це робиться іншим способом, більш ефективним. Викладач має додаток на

своєму телефоні, який може працювати на будь-якому телефоні. Також величезною перевагою використання мобільних пристроїв є те, що не потрібно підключатися до мережі Wi-Fi. Можна використовувати додаток використовуючи WiFi, LTE, 3G або будь-який інший план передачі даних, який, можливо. До телефону підключається з'ємний, невеликий кард-рідер за допомогою якого здійснюється перевірка присутності (Рис. 2.2). Коли виконується ідентифікація, Presence виконує те, що ми називаємо «без торкання пальцем», тобто не потрібно, щоб хто-небудь чіпав або перевіряв щось на екрані пристрою, щоб змусити працювати. Ви можете буквально перевіряти присутність без затримок. Присутність вбудована безпосередньо в програмне забезпечення, тому, коли карта проведена, вона підключається до платформи Presence, перевіряє дані і перевіряє студента протягом секунди. Іншими словами, Presence прискорює реєстрацію подій, надаючи в реальному часі дані про присутність студентів.

Університет технології та економіки Будапешта (Budapest University of Technology and Economics) розробили систему, яка дозволяє використовувати NFC технологію для перевірки. Студенти мають завантажити додаток і згенерувати свою електронну студентську картку і прикладати до телефона викладача на якому також встановлено додаток. Для такої операції необхідно піднести свій телефон до телефона викладача і інформація буде обміняна. Звісно ж це допомагає прискорити перевірку, але все одно телефони студентів мають мати NFC технологію, що розповсюджується, але не є всесвітньо відомою. Багато телефонів, особливо старих, не мають такого функціоналу.

Прагнучи поліпшити систему реєстрації відвідуваності, дослідники працювали над поліпшенням під іншим кутом зору. Деякі системи засновані на комп'ютерах, які можуть бути онлайн або автономно. Таких систем дуже багато і методів також.

Перевірка присутності студента використовуючи додаток, який має мати викладач.

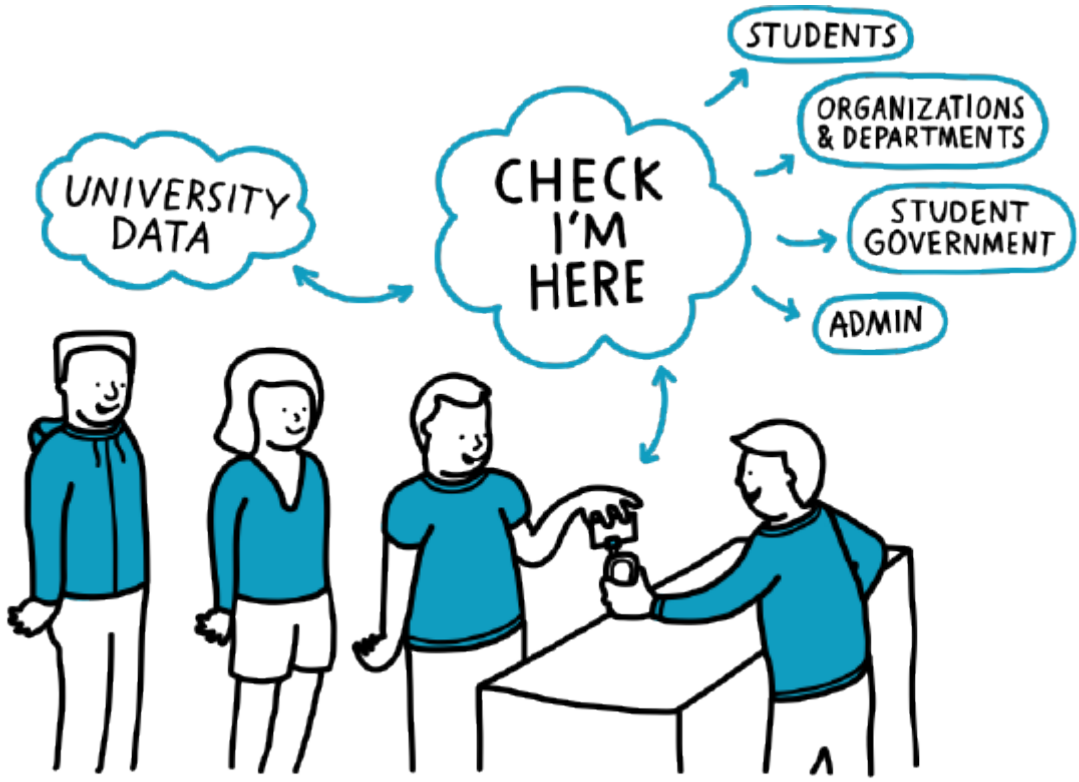

Рисунок 2.3 – Перевірка присутності студента завдяки з’ємному кард-рідеру та додатку

Прагнучи поліпшити систему реєстрації відвідуваності, дослідники працювали над поліпшенням під іншим кутом зору. Деякі системи засновані на комп'ютерах, які можуть бути онлайн або автономно. Таких систем дуже багато і методів також.

Перевірка присутності студента використовуючи додаток, який має мати викладач. На кожній парі, відкриваючи додаток, викладачу показується список студентів, які повинні бути на занятті. Роблячи перекличку, викладач відмічає учня у своєму телефоні, що дозволяє автоматично переносити усі данні одразу до сервера і формувати звіти про присутність та відсутність. Також у деяких додатках дозволялося автоматично повідомляти батьків с приводу прогулів або спізнень. Цей метод не дуже відрізняється від бумажної перевірки, але дозволяє автоматизувати деякі частини цього процесу. Викладачу не буде потрібно переносити дані вдруге у систему після всіх занять, а також повідомляти батьків, якщо студент прогулює класи.

Наступний метод перевірки відвідуваності студентів, дозволяє кожному студенту самому контролювати свою присутність. Кожен учень має загрузити додаток, який має усі позиції, у координатній плоскості, лекційних кабінетів, де проводяться заняття. Це означає, що у кожного студента, який потрапляє у клас, на заняття, з'являється кнопка яка дозволяє зареєструватись на заняття, якщо позиція телефона визначена і відповідає радіусу лекційної в якій проходить заняття. Якщо користувач додатком не буде знаходитися на уроці, у нього не з'явиться кнопка реєстрації і він буде відмічений у системі, як відсутній.

Незважаючи на те, що в обох методах також є втрата часу, все ж це поліпшення ручного процесу, оскільки дані про відвідуваність можна безпечно зберігати і легко створювати звіти.

Наступний метод зв'язаний с використанням студентської квиток. У кожного студента навчального закладу є картка на якій міститься унікальний барт-код. Цей барт-код можна зчитувати, щоб дізнатися кому саме належить він. Викладач має завантажити додаток, а також підключити до телефону з'ємний карт-рідер. Це дозволяє створити лінію з студентів і перевіряти кожного. Для цього достатньо вибрати правильний предмет в якому відобразиться список студентів, які мають перебувати на цій парі і проводити через карт-рідер кожну картку студента. Як наголошуюсь університети, які використовують такий метод для перевірки, кожне зчитування займає від 2-3 секунд. Але все одно для цього треба виділяти достатньо багато часу, ще треба закупляти такі карт-рідери для перевірки, що не є доступною опцією для багатьох вузів.

Наступний метод для перевірки присутності студентів теж використовує картки, але це спеціальні картки які видаються усім студентам вузу. Ця система працює іншим чином. Кожен кабінет на вході має карт-рідер. При вході в аудиторію кожен учень має прикласти картку до нього перед тим, як увійти. Фіксує час прикладання, щоб помічати запізнення. Викладач взагалі не причетний до перевірки відвідуваності студентами лекцій, що скорочує час. Але звісно ж, студент міг відсканувати картку і не піти на заняття. Це не спрацює тому, що такий карт-рідер знаходиться і в кабінеті і кожен має відсканувати картку при виході

також. Це дуже розумний і практичний метод для перевірки відвідуваності студентів, але він дуже дорогий. Треба обладнати усі кабінети такими сканерами, купити картки студентам і роздати.

В останньому методі перевірки присутності студентів використовується додаток, який має загрузити кожен студент, викладач також має додаток вчителя, який дозволяє йому відмічати студентів у ручному порядку, якщо телефон студента розрядився, або був загублений. Основна ідея цього методу в тому, що на 5-10 секунд буде показуватися QR-код, на великому екрані, що дозволить кожному студенту його побачити. Потрібно відкрити додаток, зареєструватись та включити сканування QR-коду, яка є внутрішньою функцією програми. Після цього данні будуть адресовані до серверу, які будуть містити інформацію QR-коду та локації учня. Це убереже систему від обману, бо користувачі могли б сфотографувати і відправити QR-код до своїх друзів. Але без присутності у навчальному закладі, нічого не вийде. Викладач має тільки включити QR-код і через 10 секунд виключити. Тобто вся присутність студентів може перевіритись за 30 секунд. Цей метод і буде розроблятися в ході цієї дипломної роботи.

1.3 Автоматизований метод перевірки присутності студентів

З технічною та цифровою епохою, важливу роль відіграє технологія розпізнавання облич, яка полягає в процесі виявлення облич та перевірці їх ідентифікації за допомогою методів обробки зображень у комп'ютерному зорі. Автоматизація системи відвідуваності, що використовує розпізнавання облич, усуває більшість недоліків, які створюють системи ручної реєстрації, включаючи втрату продуктивного часу класу, легку маніпуляцію записами відвідуваності, проксі-відвідування та небезпечну систему. У даній роботі запропоновано надійну та безпечну систему для автоматизації системи відвідуваності шляхом інтеграції технології розпізнавання облич за допомогою класифікаторів Хаар-каскад для

виявлення облич та алгоритму машинного навчання лінійної двійкової моделі гістограми для ідентифікації обличчя. Відвідуваність автоматично оновлюється у журналі відвідуваності в базі даних записів про відвідуваність. Запропонована система в даній роботі також вирішує питання, які виникають при побудові системи розпізнавання облич, деякі з яких - час отримання інформації, властивості зображення, такі як розмір, якість і налаштування інтенсивності, а також орієнтації облич, використовуючи достатній і точний набір даних навчання, методи попередньої обробки та класифікатори Хаара-каскаду. Інша проблема з системами розпізнавання облич виникає під час розпізнавання двох граней з подібними ознаками, де ймовірність виникнення помилкового результату є відносно високою. Проблема вирішується запропонованою системою шляхом інтеграції модуля гендерної класифікації з використанням моделі лінійної дискримінації (LDA), яка виводить стать студентів, що знаходяться в класі, щоб дозволити перевірку результатів модуля розпізнавання облич. Запропонована система даної роботи була зроблена економічною і портативною за допомогою мобільного додатку, а також високонадійна допомога в модулі гендерної класифікації. Система забезпечує технічно та економічно обгрунтовані рішення для подолання багатьох завдань, що виникають внаслідок ручних систем обслуговування і технологій розпізнавання облич.

Ця система пропонує побудувати автоматизовану систему відвідуваності з використанням розпізнавання облич на основі обробки зображень. Вона продовжується, спочатку створюючи базу даних особливостей обличчя всіх зарахованих студентів і зберігаючи шаблони. Різні набори даних використовуються для підготовки системи відвідуваності для узгодження і розпізнавання образів. Лінійний бінарний розпізнавач OpenCV підготовлений для розпізнавання облич. Проведення відвідуваності відбувається за допомогою камери, встановленої в класі, у вигляді фотографій, узятих відразу всіх учнів, і далі використовується для виявлення обличчя. Перетворюючи захоплене зображення на шкалу сірого кольору, вона обробляється за допомогою класифікатора Хаара для виявлення облич кожного студента на зображенні. Алгоритм лінійної двійкової схеми

гістограми (LBPH) використовується для порівняння разом з розпізнаванням обличчя кожного студента, використовуючи навчений набір даних, що зберігається в базі даних. Нарешті, у базі даних відзначено відвідуваність усіх присутніх студентів. Автори запропонували використовувати сервер для обробки даних. Різні набори даних використовуються для підготовки системи відвідуваності для узгодження і розпізнавання образів.

1.4 Постановка задачі

Відповідно до аналізу предметної галузі створення перевірки присутності студентів на лекціях є доволі пріоритетним напрямом діяльності, отже аналіз методів перевірки відвідуваності в сфері сучасного освіти є актуальним.

Таким чином, необхідно вирішити наступні задачі:

– провести аналіз предметної галузі, що пов’язаний з дослідженням використання, реалізації та результатів використання різноманітних методів перевірки присутності студентів;

– дослідити існуючі методи перевірки присутності і вибрати найбільш актуальний;

– спроектувати класифікаційну модель сегментів покупців;

– спроектувати правила підрахунку релевантності елементу щодо кожного сегменту;

– розробити базу даних для зберігання інформації відповідно до розробленої моделі;

– виконати програмну реалізацію сервісу з можливістю фотографування студентів і відправки фотографії на серверну обробку.

Для організації перевірки наявності студентів необхідно реалізувати:

– систему в якій буде збережені усі дані студентів кожного учбового закладу;

– сукупність понять та відносин, які розподіляють студента між різними учбовими закладами, які він може відвідувати;

– створити абстрактну модель для ефективного підрахунку кількості відсутніх і присутніх у реальному часі;

– модель має бути розширюваною, тобто реалізувати можливість внесення та додавання інших класифікаційних сегментів за їх потребою;

– знайти метод або бібліотеку, яка змогла за фото визначати і вдмічати студентів, які присутні;

– використовувати мобільний телефон з камерою у отриманні і відправки даних на сервер для подальшої обробки;

– знайти метод хешування обличь для швидкої машинної обробки.

2 МОДЕЛЬ ЕЛЕКТРОННОЇ ПЕРЕВІРКИ СТУДЕНТІВ

2.1 Комп’ютерний зір

Як люди, ми сприймаємо тривимірну структуру навколишнього світу з очевидною легкістю. Подумайте, наскільки яскравим є тривимірне сприйняття, коли ви дивитеся на вазу з квітами на столі поруч з вами. Ви можете визначити форму і напівпрозорість кожного пелюстка через тонкі візерунки світла і затінення, які грають по всій поверхні і легко відривають кожну квітку від фону сцени. Дивлячись на обрамлений груповий портрет, ви можете легко роздивитися (і називати) всіх людей на картинці і навіть здогадуватися про їхні емоції від зовнішнього вигляду обличчя. Перцептивні психологи витратили десятиліття, намагаючись зрозуміти, як працює візуальна система, і, навіть якщо вони можуть розробити оптичні ілюзії, щоб роздратувати деякі свої принципи, повне рішення цієї головоломки залишається невловимою [1].

Дослідники з комп'ютерного зору паралельно розробляли математичні прийоми для відновлення тривимірної форми і вигляду об'єктів у зображенні. Тепер у нас є надійні методи для точного обчислення часткової 3D моделі середовища з тисяч частково перекритих фотографій (рис. 2.1a). Враховуючи достатню кількість переглядів певного об'єкта або фасаду, ми можемо створити точні щільні 3D-моделі поверхні, що використовують стереопорівняння (рис. 2.1b). Ми можемо відслідковувати людину, яка рухається на складному фоні (рис. 2.1c). Ми можемо навіть з помірним успіхом спробувати знайти і назвати всіх людей на фотографії, використовуючи комбінацію обличчя, одягу, виявлення та розпізнавання волосся (рис. 2.1d). Однак, незважаючи на всі ці досягнення, мрія мати комп'ютер, який інтерпретує зображення на тому ж рівні, що й дворічний (наприклад, підрахунок всіх тварин на картинці) залишається невловимим. Чому так важко бачити? Частково це відбувається тому, що бачення є зворотною задачею, в якій ми прагнемо відновити деякі невідомі з огляду на недостатню інформацію, щоб повністю визначити рішення. Тому ми маємо вдаватися до

фізико-імовірнісних моделей, щоб розрізняти потенційні рішення. Однак, моделювання візуального світу у всій його багатої складності набагато складніше, ніж, скажімо, моделювання голосового тракту, який виробляє звукові звуки.

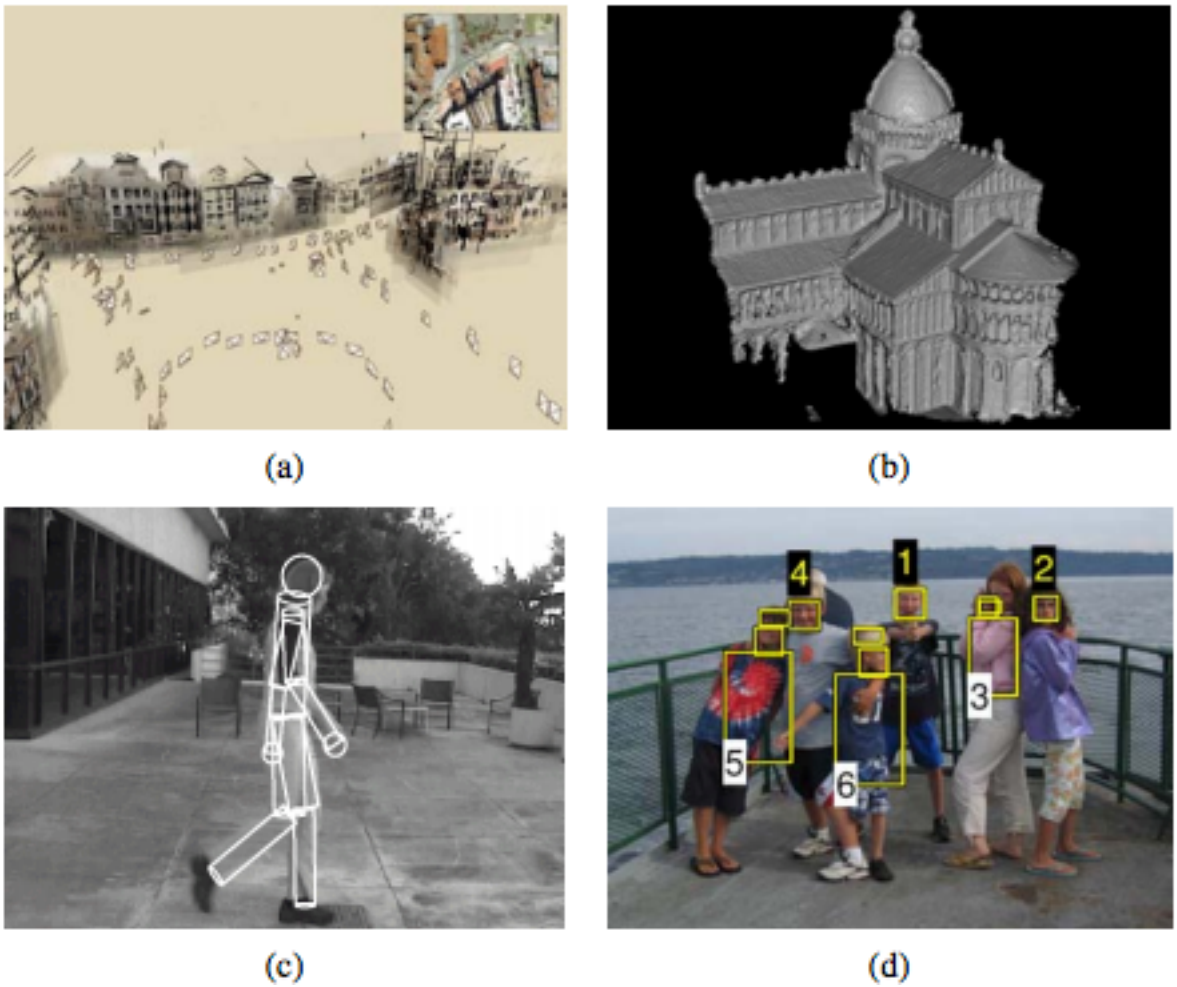

Рисунок 2.1 - Деякі приклади алгоритмів та додатків комп'ютерного зору.

Передні моделі, які ми використовуємо в комп'ютерному зорі, зазвичай розробляються у фізиці (радіометрія, оптика, дизайн датчиків) і в комп'ютерній графіці. Обидва ці поля демонструють, як об'єкти рухаються і живуть, як світло відбивається від їх поверхонь, розсіюється атмосферою, переломлюється через об'єктиви камери (або людські очі), і, нарешті, проектується на плоску (або вигнуту) площину зображення. Хоча комп'ютерна графіка ще не є досконалою (жоден повністю комп'ютерний фільм з людськими персонажами ще не досяг успіху при перетині звичної долини2). Домени, такі як передача нерухомої сцени,

що складається з повсякденних об'єктів або анімуючих вимерлих істот, таких як динозаври, ілюзія реальності є досконалою.

У комп'ютерному зорі ми намагаємося зробити інверсію, тобто описати світ, який ми бачимо на одному або декількох зображеннях, і реконструювати його властивості, такі як форма, освітлення і розподіл кольорів. Дивно, що люди і тварини роблять це так без зусиль, в той час як алгоритми комп'ютерного зору так схильні до помилок. Люди, які не працювали на місцях, часто недооцінюють складність проблеми. Це помилкове сприйняття того, що зір повинен бути легким, починається з перших днів штучного інтелекту, коли спочатку вважалося, що когнітивні (логічні докази і планування) частини інтелекту є суттєво складнішими, ніж компоненти сприйняття.

Доброю новиною є те, що комп'ютерне зір сьогодні використовується в широкому спектрі реальних додатків, які включають:

- Оптичне розпізнавання символів (OCR): читання рукописних поштових індексів на літерах (рис. 2.2a) і автоматичне розпізнавання номерних знаків (ANPR);
- Перевірка машини: швидка перевірка деталей для забезпечення якості з використанням стереооглядача з спеціалізованим підсвічуванням для вимірювання допусків на крилах літаків або автозапчастин (Малюнок 2.2b) або шукають дефекти в сталевих виливках з використанням рентгенівського зору;
- Роздрібна торгівля: розпізнавання об'єктів для автоматизованих смуг перевірки (мал. 2.2c);
- Побудова 3D-моделі (фотограмметрія): повністю автоматизована побудова 3D-моделей з аерофотознімків, що використовуються в таких системах, як Bing Maps;
- Медична візуалізація: реєстрація доопераційної та внутрішньоопераційної зйомки (рис. 2.2d) або проведення довгострокових досліджень морфології мозку людей, коли вони старіють;

- Автомобільна безпека: виявлення несподіваних перешкод, таких як пішоходи на вулиці, в умовах, де не працюють активні методи зору, такі як радар або лідар добре (Малюнок 2.2e);
- Переміщення збігу: об'єднання зображень, створених комп'ютером (CGI), з кадрами живої дії відстеження точок функції у вихідному відео, щоб оцінити рух і форму 3D-камери навколишнього середовища. Такі методи широко використовуються в Голлівуді що відокремлює справжніх людей від андроїд роботів і комп'ютерних анімованих людей) обмежені. Вони також вимагають використання точного матування, щоб вставити нові елементи між елементами переднього і заднього планів;
- Захоплення руху (mocap): використання ретрорефлекторних маркерів, що розглядаються з різних камер або інших методів бачення, для захоплення акторів для комп'ютерної анімації;
- Спостереження: моніторинг зловмисників, аналіз дорожнього руху (Рисунок 2.2f) та моніторинг басейнів для жертв утоплення;
- Розпізнавання відбитків пальців та біометричні дані: для автоматичного аутентифікації доступу, а також для судових застосувань.

Отже, краще подумати про проблему, яка існує, до відповідних методів, а не для того, щоб захопити першу техніку, про яку ви, можливо, чули. Такий відхід від проблем до рішень є типовим інженерним підходом до вивчення зору та його відображення власний досвід у цій галузі. По-перше, я придумав детальне визначення проблеми і вирішив, з якими проблемами стоїть проблема. Потім я намагаюся з'ясувати, які методи, як відомо, працюють, реалізовувати деякі з них, оцінювати їх роботу, і нарешті зробіть вибір. Для того, щоб цей процес працював, важливо мати реалістичні дані тесту, як синтетичні, які можуть бути використані для перевірки правильності та аналізу чутливості до шуму, так і даних реального світу, типових для способу використання системи [2].

Вона також приймає науковий підхід до основних проблем зору. Тут я намагаюся придумати найкращі моделі фізики системи: як створюється сцена, як взаємодіє світло зі сценою і атмосферними ефектами, і як працюють датчики,

включаючи джерела шуму і невизначеності. Завдання полягає в тому, щоб спробувати інвертувати процес придбання, щоб отримати найкращий опис сцени.

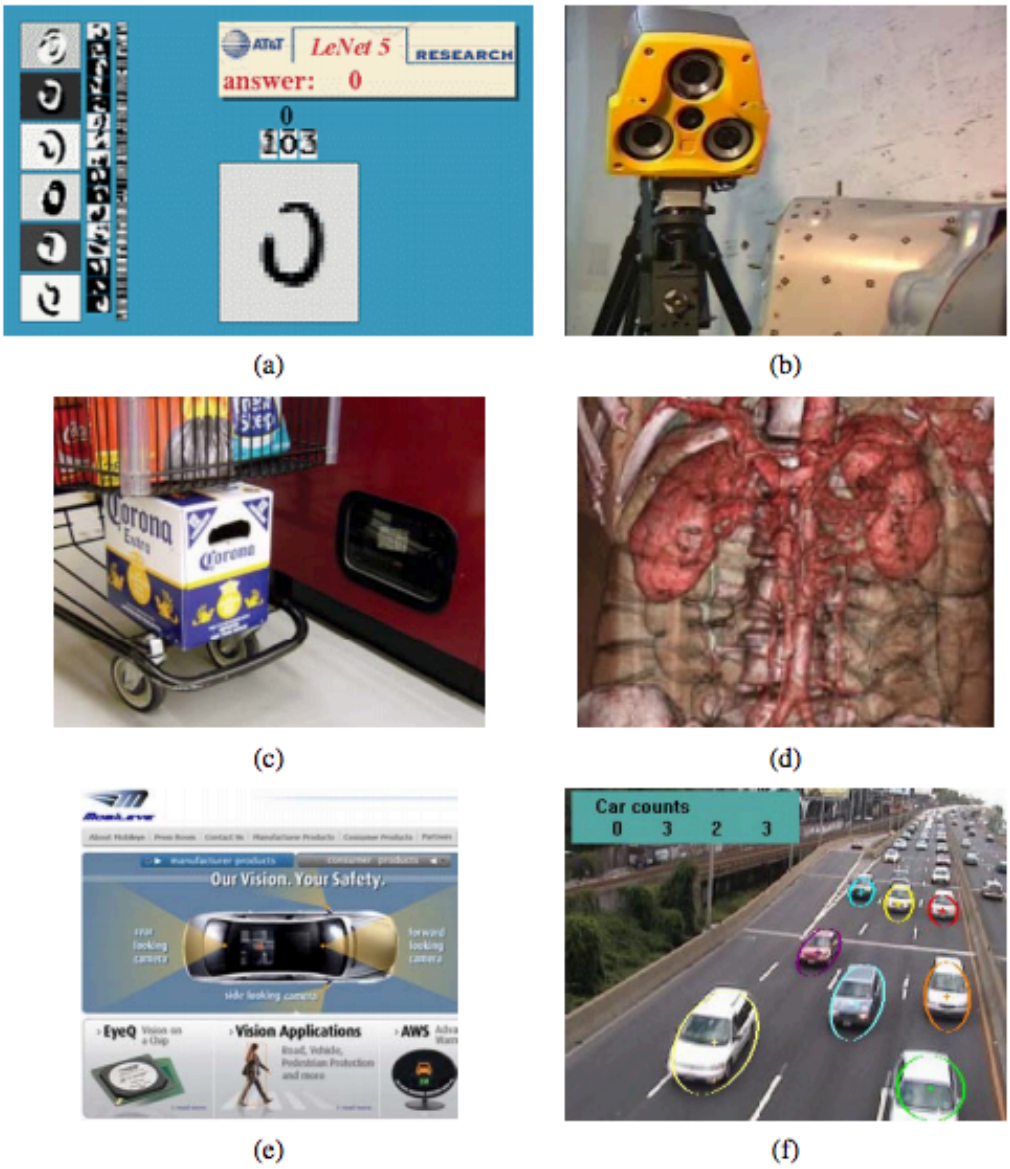

Рисунок 2.2 - Деякі приклади алгоритмів та додатків комп'ютерного зору.

Там, де це доречно, розподіл вірогідності використовують для моделювання сцени та процесу отримання шумного зображення. Асоціація попередніх розподілів з невідомими часто буває називається байєсовським моделюваннямю Можна пов'язувати функцію ризику або втрату з неправильною оцінкою відповіді та налаштувати алгоритму виведення для мінімізації очікуваного ризику. За допомогою статистичних методів допомагає зібрати багато навчальних даних, з

яких можна вивчати ймовірнісні моделі. Нарешті, статистичні підходи дозволяють використовувати перевірені методи висновку для оцінки найкращої відповіді (або розподілу відповідей) та кількісної оцінки невизначеності отриманих оцінок.

Оскільки стільки комп'ютерного зору передбачає вирішення зворотних завдань або оцінку невідомих величин, у моїй книзі також велика увага приділяється алгоритмам, особливо тим, що добре відомі на практиці. Для багатьох проблем із зором дуже легко придумати математичний опис проблеми, яка або не відповідає реальним реальним умовам, або не піддається стабільному оцінюванню невідомих. Нам потрібні алгоритми, які є надійними до шуму та відхиленням від наших моделей і є досить ефективними з точки зору ресурсів часу та простору. У цій книзі, я звертаюся до цих питань докладно, використовуючи байєсовські методики, де це можливо, щоб забезпечити надійність і ефективність пошук, мінімізація та алгоритми розв'язання лінійних систем для забезпечення ефективності. Більшість алгоритмів, описаних у цій книзі, знаходяться на високому рівні, в основному, це перелік кроків, які потрібно заповнити студентами, або прочитавши більш детальну інформацію в інших місцях. Насправді, багато хто алгоритми накреслені у вправах.

2.2 Розпізнавання обличчя

Багато дослідників намагалися зрозуміти, як люди розпізнають обличчя, більшість з яких виникає, коли виникла проблема автоматичного розпізнавання обличчя, шукаючи натхнення дизайну. Здається важливо зрозуміти, як ми робимо це завдання, як ми сприймаємо людей. Потім ці знання можуть бути застосовані в автоматичних системах розпізнавання осіб. Однак, багато алгоритмів не використовують цю інформацію, використовуючи лише математичні інструменти.

Сьогодні деякі програми розпізнавання облич не потребують виявлення облич. У деяких випадках зображення облич, що зберігаються в базах даних, вже

нормалізуються. Існує стандартний формат введення зображення, тому немає необхідності в кроці виявлення. Прикладом цього може бути кримінальна база даних. Там правоохоронні органи зберігають обличчя людей з кримінальним повідомленням. Якщо є новий суб'єкт і у поліції є його паспортна фотографія, виявлення обличчя не є необхідним. Тим не менш, звичайний вхідний образ систем комп'ютерного зору не є придатним. Вони можуть містити багато елементів або облич. У цих випадках виявлення обличчя є обов'язковим. Це також неминуче, якщо ми хочемо розробити автоматизовану систему відстеження обличчя. Наприклад, системи відеоспостереження намагаються включати виявлення облич, відстеження та розпізнавання. Отже, доцільно припустити виявлення облич як частину більш широкої проблеми розпізнавання облич.

Виявлення обличчя має вирішувати декілька відомих проблем. Вони зазвичай присутні в зображеннях, знятих у неконтрольованих середовищах, таких як відеоспостереження. Ці проблеми можна пояснити деякими факторами:

- Варіація пози. Ідеальним сценарієм для виявлення обличчя буде той, у якому були задіяні лише фронтальні зображення. Але, як зазначалося, це дуже малоймовірно в загальних неконтрольованих умовах. Більш того, продуктивність алгоритмів виявлення облич сильно падає, коли існують великі варіації. Це серйозна проблема дослідження. Різноманітність пози може статися через рух об'єкта або кут камери.
- Особливості оклюзії. Наявність елементів, таких як бороди, келихи або капелюхи, створює високу мінливість. Обличчя також можуть бути частково покриті об'єктами або іншими особами.
- Вираз обличчя. Особливості обличчя також сильно відрізняються через різні жести на обличчі.
- Умови зображення. Різні камери та умови навколишнього середовища можуть впливати на якість зображення, впливаючи на зовнішній вигляд обличчя.

Є деякі проблеми, тісно пов'язані з виявленням облич, крім вилучення ознак і класифікації облич. Наприклад, розташування облич - це спрощений спосіб

виявлення обличчя. Його метою є визначення місця розташування обличчя на зображенні, де є тільки одне обличчя. Ми можемо диференціювати між виявленням обличчя та розташуванням обличчя, оскільки останнє є спрощеним поблемом першого. Методи, такі як розміщення меж голови, спочатку використовувалися на цьому сценарії, а потім експортувалися до більш складних проблем. Виявлення функцій обличчя стосується виявлення та локалізації деяких релевантних функцій, таких як ніс, брови, губи, вуха тощо. Є багато літератури на цю тему, яка обговорюється пізніше. Іншою проблемою, яка іноді є наслідком виявлення обличчя, є відстеження обличчя. Мета багатьох систем полягає не лише у виявленні обличчя, але й у змозі знайти це обличчя в реальному часі. Ще один приклад - система відеоспостереження [3].

Алгоритми розпізнавання облич звичайно поділяють спільні кроки. По-перше, виконується деяке зменшення розміру даних для досягнення допустимого часу відгуку. Деяка попередня обробка також може бути зроблена для адаптації вхідного зображення до передумов алгоритму. Потім деякі алгоритми аналізують зображення як є, а деякі інші намагаються витягти певні відповідні області обличчя. Наступний етап, як правило, передбачає вилучення особливостей обличчя або вимірювань. Потім вони будуть зважуватися, оцінюватися або порівнюватися, щоб вирішити, чи є обличчя і де він знаходиться. Нарешті, деякі алгоритми мають рутину навчання, і вони включають нові дані в свої моделі. Таким чином, виявлення облич є проблемою двох класів, де ми повинні вирішити, чи є обличчя або немає на картинці. Такий підхід можна розглядати як спрощену проблему розпізнавання облич. Розпізнавання обличчя має класифікувати дане обличчя, і існує стільки ж класів, скільки кандидатів. Отже, багато методів виявлення облич дуже схожі на алгоритми розпізнавання облич. Іншими словами, методи розпізнавання облич часто використовуються для розпізнавання облич.

Багато систем розпізнавання облич мають відеопослідовність як вхідні дані. Ці системи можуть вимагати здатності не тільки виявляти, але й відстежувати обличчя. Відстеження обличчя є по суті проблемою оцінки руху. Відстеження обличчя може бути виконано з використанням багатьох різних методів, наприклад,

відстеження голови, відстеження об'єктів, відстеження на основі зображень, відстеження на основі моделі. Це різні способи класифікації цих алгоритмів:

- Відстеження голови / відстеження індивідуальних функцій. Голову можна відстежувати як цілісну сутність, або певні особливості відстежуватись індивідуально.
- 2D / 3D. Двовимірні системи відстежують обличчя і виводять простір зображення, де розташоване обличчя. Тривимірні системи, з іншого боку, виконують 3D-моделювання обличчя. Такий підхід дозволяє оцінити варіації позі або орієнтації.

Основний процес відстеження обличчя прагне знайти зображення на зображенні. Потім він повинен обчислити відмінності між кадрами, щоб оновити розташування обличчя. Існує багато проблем, з якими необхідно зіткнутися: часткові оклюзії, зміни освітленості, швидкість обчислень і деформації обличчя. Одним з прикладів алгоритму відстеження обличчя може бути той, який запропонував Баек. Вектор стану особи включає в себе центральне положення, розмір прямокутника, що містить особу, середній колір області особи і їх перші похідні. Нові особи кандидатів оцінюються оцінювачем Калмана. У режимі відстеження, якщо обличчя не є новим, обличчя з попереднього кадру використовується як шаблон. Положення особи оцінюється оцінювачем Калмана, а область обличчя шукається навколо алгоритмом SSD, використовуючи згаданий шаблон. Коли SSD знаходить область, кольорова інформація вбудовується в оцінку Калмана, щоб точно обмежити область обличчя.

2.3 Класифікатори

Згідно з Jain, Duin і Mao, існують три поняття, які є ключовими у побудові класифікатора - подібність, вірогідність і межі рішення. З цієї точки зору ми презентуємо класифікатори. Потім вектор стану цього обличчя оновлюється.

Результат виявився надійним, коли деякі обличчя перекривалися або коли відбувалися зміни кольору.

2.3.1 Подібність

Такий підхід є інтуїтивним і простим. Подібні шаблони повинні належати до одного класу. Цей підхід використовувався в алгоритмах розпізнавання облич, реалізованих пізніше. Ідея полягає в тому, щоб встановити метрику, яка визначає подібність і представлення зразків одного класу. Наприклад, метрикою може бути евклідова відстань. Представлення класу може бути середнім вектором всіх моделей, що належать до цього класу. З цим параметром може використовуватися правило прийняття рішень 1-NN. Показники класифікації зазвичай добре. Такий підхід подібний алгоритму кластеризації k-means в безнавчанні. Є й інші методи, які можна використовувати. Наприклад, векторне квантування, навчальне векторне квантування або самоорганізаційні карти. Іншим прикладом такого підходу є відповідність шаблону. Дослідження класифікують алгоритм розпізнавання облич на основі різних критеріїв. Деякі публікації визначали відповідність шаблонів як вид або категорію алгоритмів розпізнавання облич [20]. Проте ми можемо побачити відповідність шаблону просто як інший метод класифікації, де немечені зразки порівнюються зі збереженими шаблонами.

2.3.2 Вірогідність

Деякі класифікатори будуються на основі ймовірнісного підходу. Правило прийняття рішення Баєса часто використовується. Правило може бути змінено з урахуванням різних факторів, які можуть призвести до відсутності класифікації.

Правила байєсівського рішення можуть дати Оптимальний класифікатор, і помилка Баєса може бути кращим критерієм для оцінки особливостей. Тому апостеріорні функції ймовірності можуть бути оптимальними. Існують різні байєсовські підходи. Одним з них є визначення правила прийняття плагіонера (MAP).

$$p(Z|\omega_i)P(\omega_i) = \max\{p(Z|\omega_j)P(\omega_j)\}\, Z \in \omega_i \quad (1)$$

де ω_i - класи облич і Z - зображення в зменшеному просторі PCA.

Тоді повинні бути змодельовані щільності всередині класу (функції щільності ймовірності), тобто

$$p(Z|\omega_i) = \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(Z - M_i)^t \sum_i^{-1}(Z - M_i) \quad (2)$$

де Σ_i і M_i - коваріаційні і середні матриці класу ω_i.

Коваріаційні матриці ідентичні і діагональні, отримуючи компоненти за допомогою дисперсії вибірки в підпростір PCA. Інший варіант полягає в тому, щоб отримати коваріаційну матрицю класу, діагоналізуючи матрицю розкиду класу, використовуючи сингулярне розкладання (SVD).

Існують і інші підходи до байєсівських класифікаторів, запропонованих на альтернативі MAP - максимальна ймовірність (ML). Вони запропонували не-евклідову міру подібності міри, а також два класи варіацій зображення обличчя: відмінності між зображеннями від однієї особи (,I, міжособистісні) та варіації між різними індивідуумами E. Вони визначають відповідність МЛ подібності, як

$$S' = p(\Delta|\omega_i) = \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\{-\frac{1}{2}\| i_j - i_k \|^2\} \quad (3)$$

де Δ - вектор різниці між зразками, i_j і i_k.

Які зберігаються як вектор з коефіцієнтами міжособистісного підпростору. Ідея полягає в тому, щоб попередньо обробити ці зображення в автономному режимі, тому алгоритм набагато швидше, коли виконує розпізнавання облич.

Проблема комбінації класифікатора може бути визначена як проблема знаходження комбінованої функції, що приймає N-розмірні вектори оцінки з класифікаторами і виведення N підсумків класифікації. Для об'єднання класифікаторів у розпізнаванні обличчя може бути декілька причин:

- У конструктора є кілька класифікаторів, кожен з яких розроблений з іншим підходом. Наприклад, може бути класифікатор, призначений для розпізнавання облич за допомогою шаблонів брів. Ми могли б об'єднати його з іншим класифікатором, який використовує інший підхід до розпізнавання. Це може призвести до кращої ефективності розпізнавання.
- Можуть бути різні навчальні набори, зібрані в різних умовах і представляють різні особливості. Кожен навчальний набір може бути добре підходить для певного класифікатора. Ці класифікатори можна об'єднати.
- Один навчальний набір може показувати різні результати при використанні різних класифікаторів. Для досягнення найкращих результатів можна використовувати комбінацію класифікаторів.
- Деякі класифікатори відрізняються за своєю продуктивністю в залежності від певних ініціалізацій. Замість вибору одного класифікатора, ми можемо об'єднати деякі з них.

Існують різні комбіновані схеми. Вони можуть відрізнятися один від одного за своїми архітектурами і вибором комбайнера. Комбінатор в розпізнаванні образів зазвичай використовує фіксовану кількість класифікаторів. Це дозволяє використовувати переваги кожного класифікатора. Загальний підхід полягає в тому, щоб розробити певну функцію, яка зважує результат кожного класифікатора. Потім повинна існувати межа рішення для прийняття рішення на основі цієї функції. Комбіновані методи також можуть бути згруповані на основі стадії, на якій вони працюють. Комбінатор може працювати на рівні функцій. Особливості всіх

класифікаторів об'єднуються для формування нового вектора ознак. Потім робиться нова класифікація. Інший варіант - працювати на рівні балів, як було зазначено раніше. Такий підхід відокремлює класифікаційні знання і комбінатор. Цей тип комбінаторів є популярним завдяки тому рівню абстракції. Однак, комбінатори можуть бути різними залежно від характеру виходу класифікатора. Вихідні дані можуть бути простим класом або групою класів (абстрактний інформаційний рівень). Іншим більш точним виходом може бути упорядкований список класів-кандидатів (рівень рангу). Класифікатор міг би мати більш інформативний висновок, включивши до кожного класу певний показник ваги або довіри (рівень вимірювання). Якщо комбінація включає дуже спеціалізовані класифікатори, кожен з них зазвичай має а різний вихід. Об'єднання різних масштабів виробництва та заходів довіри може бути складною проблемою. Однак, вони матимуть аналогічний висновок, якщо всі класифікатори використовують одну і ту ж архітектуру.

Комбінатори можуть бути згруповані в три категорії відповідно до їх архітектури:

- Паралельно. Всі класифікатори виконуються самостійно. Потім застосовується комбайнер.
- Послідовний. Класифікатори працюють один за одним. Кожен класифікатор полірує попередні результати.
- Ієрархічна. Класифікатори об'єднані в деревоподібну структуру.

Комбінаторні функції можуть бути дуже простими рудними комплексами. Низька комбінація складності може вимагати навчання лише однієї функції, вхід якої є оцінками одного класу. Найвища складність може бути досягнута шляхом визначення декількох функцій, по одному для кожного класу. Вони приймають за параметри всі оцінки. Таким чином, для комбінації використовується більше інформації. Більш складні класифікатори можуть потенційно забезпечити кращі результати. Рівень складності обмежений кількістю навчальних вибірок і обчислювальним часом. Тому дуже важливо вибрати рівень складності, який

найкраще відповідає цим вимогам обмеження. Деякі комбінатори також можуть бути навчальними. Комбінатори, які можна тренувати, можуть привести до кращих результатів за рахунок необхідності додаткових навчальних даних

2.4 Підхід нейронної мережі

Штучні нейронні мережі є популярним інструментом у розпізнаванні облич. Вони використовувалися для розпізнавання і класифікації образів. Кохонен був першим, хто продемонстрував, що нейронну мережу можна використовувати для розпізнавання вирівняних і нормалізованих граней. З того часу було запропоновано багато різних методів, заснованих на нейронній мережі. Деякі з цих методів використовують нейронні мережі тільки для класифікації. Один з підходів полягає у використанні нейронних мереж, заснованих на прийнятті рішень, які класифікують попередньо оброблені та суб-вибіркові зображення облич [4].

2.4.1 Нейронні мережі з фільтрами Габора

Алгоритм досягає розпізнавання осіб шляхом реалізації багатошаровий персептрон з алгоритмом зворотного поширення. По-перше, існує етап попередньої обробки. Кожне зображення нормалізується з точки зору контрасту і освітленості. Шум зменшується за допомогою фільтра «згладженість». Вона працює шляхом застосування нечіткого членства до сусідніх пікселів цільового пікселя. Він використовує значення медіани як членство в 1 значенні, а також знижує екстремальні значення, використовуючи середній фільтр і середній фільтр.

Потім кожне зображення обробляється через фільтр Габора. Фільтр представлений як комплексний синусоїдальний сигнал, модульований функцією ядра Гаусса. Фільтр Габора має п'ять параметрів орієнтації і три просторові частоти, тому є 15 вейвлетів Габора. Архітектура нейронної мережі проілюстрована на малюнку 2.4.

Для кожного зображення обличчя виходи - це 15 зображень Габора, які записують зміни, виміряні фільтрами Габора. Перший шар отримує функції Gabor. Кількість вузлів дорівнює розмірності вектора ознак, що містить функції Габора. Вихід мережі - це кількість зображень, які система повинна розпізнати. Навчання мережі, алгоритм зворотного поширення, здійснюється за такою процедурою:

- Ініціалізація ваг і порогових значень.
- Ітераційний процес до виконання умови завершення:
    1) Активуйте, застосовуючи вхідні та потрібні виходи. Розрахуйте фактичні виходи нейронів у прихованому та вихідному шарах, використовуючи функцію активації сигмоїди.
    2) Оновити ваги, поширюючи назад помилки.
    3) Збільшення значення ітерації.

Незважаючи на те, що основна мета алгоритмів полягає в тому, щоб зіткнутися з варіаціями освітлення, вона показує корисне застосування нейронної мережі для розпізнавання облич. Це може бути корисно з деякими вдосконаленнями для того, щоб мати справу з проблемами позі та оклюзії.

2.4.2 Нечіткі нейронні мережі

Іншим підходом є впровадження нечіткої математики в нейронні мережі для розпізнавання облич. Bhattacharjee et al. розроблена в 2009 р. система розпізнавання осіб з використанням нечіткого багатошарового персептрона (MLP). Ідея цього

підходу полягає в тому, щоб захопити поверхні рішень в нелінійних різноманіттях, завдання, яке простий MLP навряд чи може завершити.

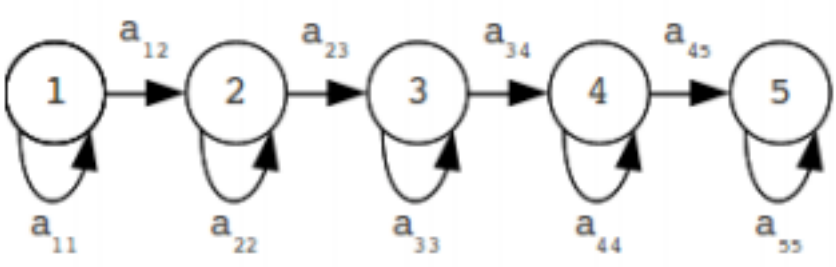

Рисунок 2.4 - нейронні мережі з фільтрами Габора

Вектори ознак отримані за допомогою вейвлет-перетворень Габора. Потім вихідні вектори, отримані на цьому етапі, повинні бути розмитими. Цей процес простий. Чим більше векторний ознака підходить до вектора середнього класу, тим вищим є нечітке значення. Коли різниця між обома векторами зростає, нечітке значення наближається до 0.

Вибрана нейронна мережа являє собою MLP з використанням зворотного поширення. Для кожного класу є мережа. Таким чином, це двокласна проблема класифікації. Фаззифікація нейронної мережі ґрунтується на наступній ідеї: моделі, чий клас менш визначений, повинні мати меншу роль у регулюванні ваг.

$$\varphi_i = 0.5 + \frac{e^{c(d_j - d_i)/d} - e^{-c}}{2(e^{c} - e^{-c})} \tag{5}$$

де d_i - відстань вектора від середнього класу i. Константа c контролює швидкість, з якою нечітке членство зменшується до 0,5.

$$\varphi_i = 1 - \varphi_i(x_k) \tag{6}$$

де x_k у оновлення ваги дає $|\varphi_i(x_k) - 2(x_k)|m$, де m - постійна. решта процесу - за звичайною процедурою MLP.

Результати алгоритму показують коефіцієнт помилок 2,125 з використанням бази даних ORL [5].

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 UML проектування програмної системи

UML [6] - це уніфікований графічний мова моделювання для опису, візуалізації, проектування та документування ГО систем. UML покликаний підтримувати процес моделювання ПС на основі ГО підходу, організовувати взаємозв'язок концептуальних і програмних понять, відображати проблеми масштабування складних систем.

Моделі на UML використовуються на всіх етапах життєвого циклу ПС, починаючи з бізнес-аналізу і закінчуючи супроводом системи [7]. Різні організації можуть застосовувати UML на свій розсуд в залежності від своїх проблемних областей і використовуваних технологій.

Для проектування програмної системи було створено декілька діаграм UML, які будуть розглянуті нижче.

На діаграмі послідовностей (див. рис. 3.1) зображена система в цілому на вищому рівні абстракції та взаємодія основних компонентів – клієнту, сервісу, IoT або мобільний телефон та слою доступу до даних. Користувач контактує з View та передає через нього дані, які опрацьовує шар сервісів, та у випадку успішного виконання валідації працює з базою даних, або повертає повідомлення про те, що валідація не була пройдена. Якщо дані були надіслані до бази даних, то після обробки вони повернуться через усі етапи до користувача. У той же час IoT або мобільній телефон також може контактувати з сервісами.

Діаграма прецедентів (див. рис. 3.2) має двох акторів – Викладач та Студент. Викладач – людина, яка увійшла у сервіс під своїм акаунтом, який був створений університетом. Після аутентифікації він має можливість фотографувати присутніх студентів і відправляти результат на перевірку на сервер, самостійно відмічати студентів, переглядати статистику присутності, налаштовувати правила та отримувати сповіщення про результат.

Студент – також володіє акаунтом тільки після реєстрації університетом у своїй базі. Студент может перевіряти свою присутність на уроках, а також передивлятися загальну статистику відвідувань. Також у розпорядженні студента є можливість сфотографувати себе у кабінеті с локацією, що може стати гарним аргументом у випадку, якщо тебе не відмітили автоматично, а також викладач не зміг цього зробити своєчасно.

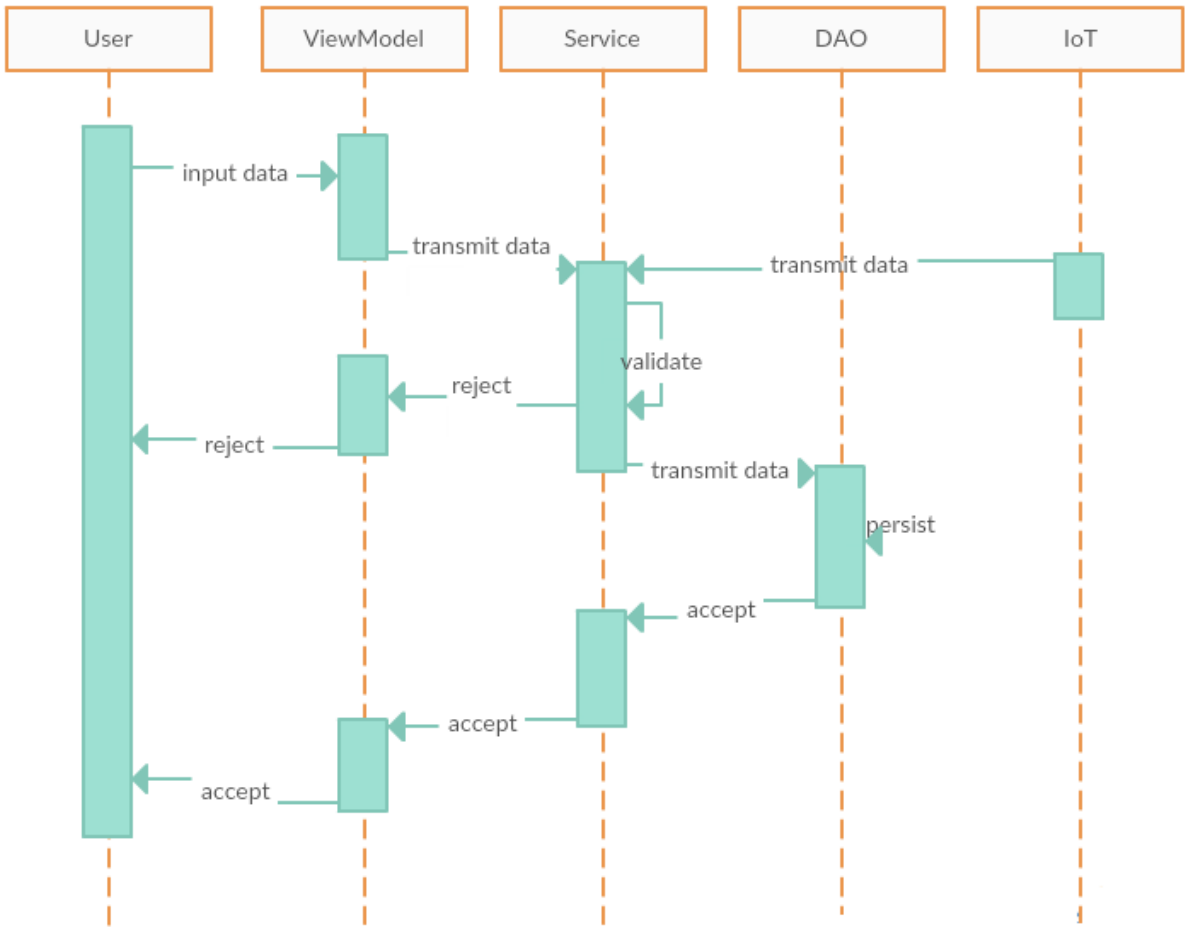

Рисунок 3.1 – Діаграма послідовностей

Діаграма класів (див. рис. 3.3) включає в себе класи Student, як модель студента; Teacher, як модель вчителя, який має право редагувати присутність студента, якщо якимось чином студент не був відмічений системою; Course, завдяки якому буде отримана інформація про курс, у кожного вчителя може бути безліч курсів, яким він навчає; UserCourseList, як модель яка об'єднує студентів і

курси, які вони мають обов’язкого відвідувати та параметр присутності, який буде автоматично опрацьваний, завдяки розпізнаванню по лицю; Admin, як модель, де він відповідає за створенняб курсів, вчителів, студентів.

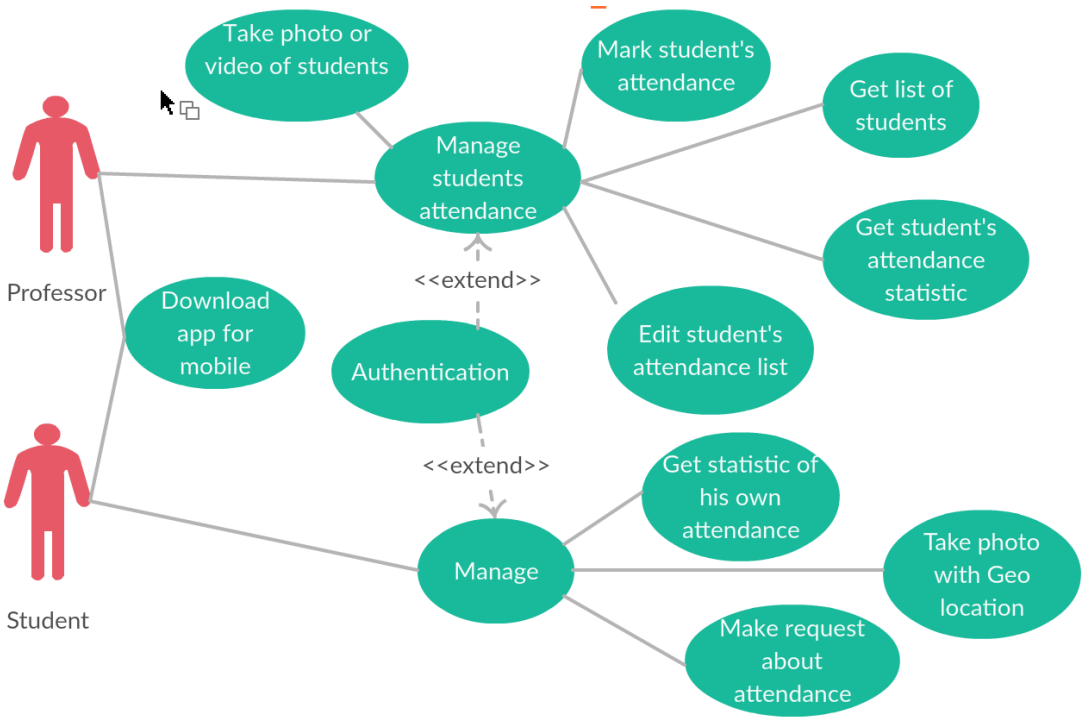

Рисунок 3.2 – Діаграма прецедентів

FaceNet, як модель яка буде використовуватися у парі с таблицею студентів і буде зберігати код у вигляді строки, що буде описувати обличчя студента з різних ракурсів для більш точного опрацювання відвідуваності студентів. CourseStatistic, як модель, яка відповідає і зберігає дані після обробки фрагменту відео або фотографії показуючи, скільки студентів відсутні і скільки присутні у реальному часі. На діаграмі розглянуто лише класи-моделі області, в системі також присутні класи та інтерфейси для взаємодії з базою даних.

Діаграма розгортання (див. рис. 3.4) використовується для виявлення загальної форми і топології розподіленої програмної системи і має зображення розміщення компонентів по різних вузлах системи. Як можна зрозуміти, представлена програмна система розділена на декілька компонентів. Основними складова є віртуальна машина AWS EC2, на якій розгорнуто веб-додаток. Він

володіє RESTful API, завдяки чому взаємодіють веб-клієнт, мобільний пристрій викладача та студента.

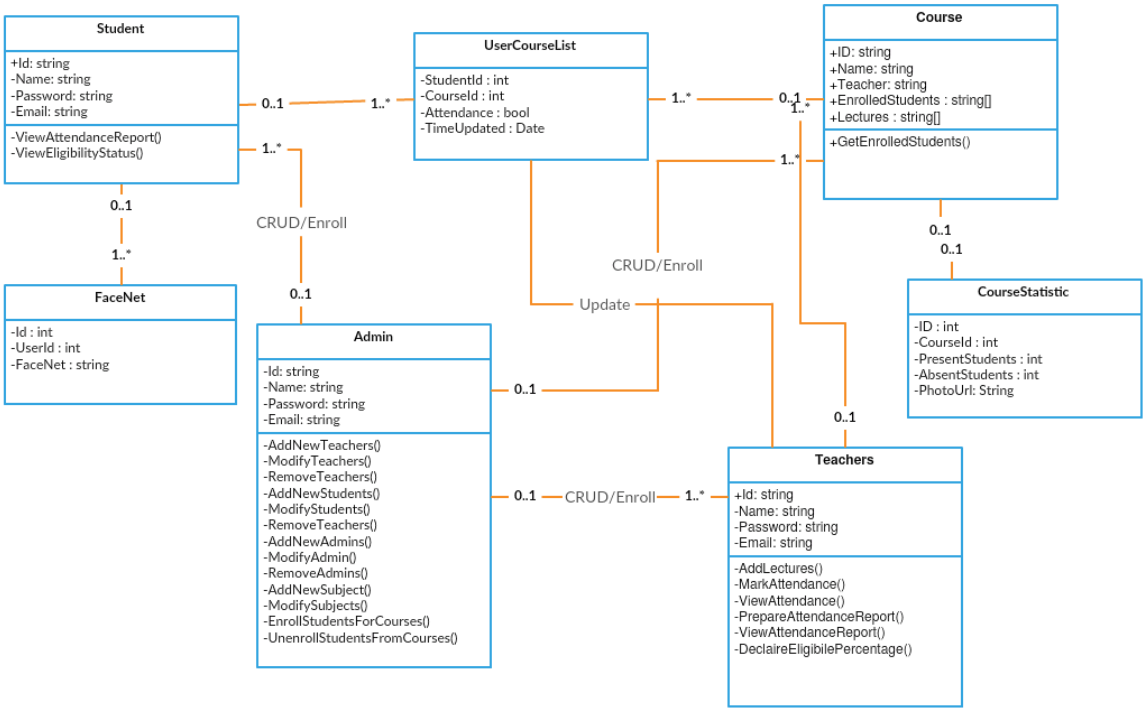

Рисунок 3.3 – Діаграма класів

Сервер взаємодіє з базою даних, яка знаходиться у хмарному середовищі. Веб-додаток також перебуває у хмарному середовищі (AWS S3), яке налаштовано на static website hosting. Мобільний додаток та веб клієнт взаємодіють з сервером через протокол HTTP. Завдяки такій роботі досягається швидка взаємодія між викладачем і перевіркою пристуності студента у реальному часі без затримок. AWS дозволяє створити архітектру в якій не буде проблем с доступомб навіть якщо один із серверів перестане парцювати і не буде відповідати на RESTful API реквести.

Діаграма розгортання (див. рис. 3.4) використовується для виявлення загальної форми і топології розподіленої програмної системи і має зображення розміщення компонентів по різних вузлах системи. Як можна зрозуміти, представлена програмна система розділена на декілька компонентів. Основними складова є віртуальна машина AWS EC2, на якій розгорнуто веб-додаток.

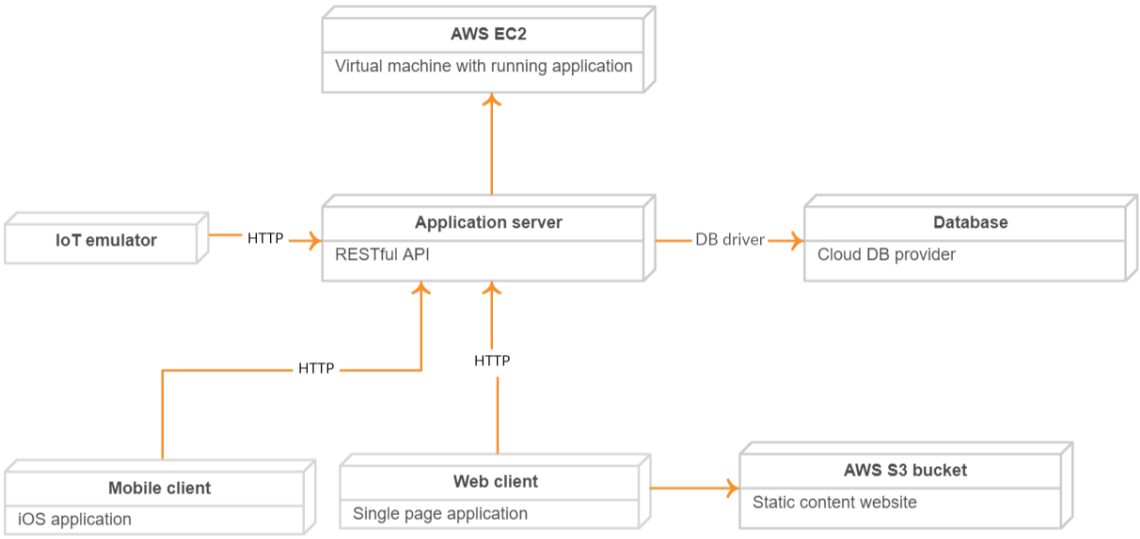

Рисунок 3.4 – Діаграма розгортання

Розроблені діаграми допомагають зрозуміти принцип роботи усієї системи, уявити процеси, що виконуються під час роботи з нею та прискорити подальшу розробку проекту.

3.2 Архітектура програмної системи

Для реалізації сервісу було вирішено вибрати клієнт-серверну архітектуру. Ця архітектура є одним із шаблонів програмного забезпечення та є найпопулярнішою концепцією у створенні розподілених мережевих систем і передбачає взаємодію та обмін даними між ними. Архітектура клієнт-сервер - це архітектура виробників і споживачів, де сервер виступає, як виробник і клієнт, як споживач. Сервер розміщує і надає клієнтові послуги найвищого класу на вимогу. Ці послуги включають зберігання, доступ, доступ до принтера, спільний доступ до файлів та або прямий доступ до необробленої обчислювальної потужності сервера. Клієнт-серверна архітектура працює тоді, коли клієнтський девайс посилає запит ресурсу або процесу на сервері через мережеве з'єднання, яке обов'язково

обробляється і доставляється клієнту. Сервер може обробляти запити декількох клієнтів одночасно. Також і один клієнт може бути підключений до декількох серверів одночасно, кожен з яких надає різний набір послуг. У простій формі Інтернет також побудований на клієнт-серверній архітектурі, де веб-сервери обслуговують багато користувачів з даними веб-сайту одночасно. На рисунку 3.5 наведене схематичне зображення архітектури системи.

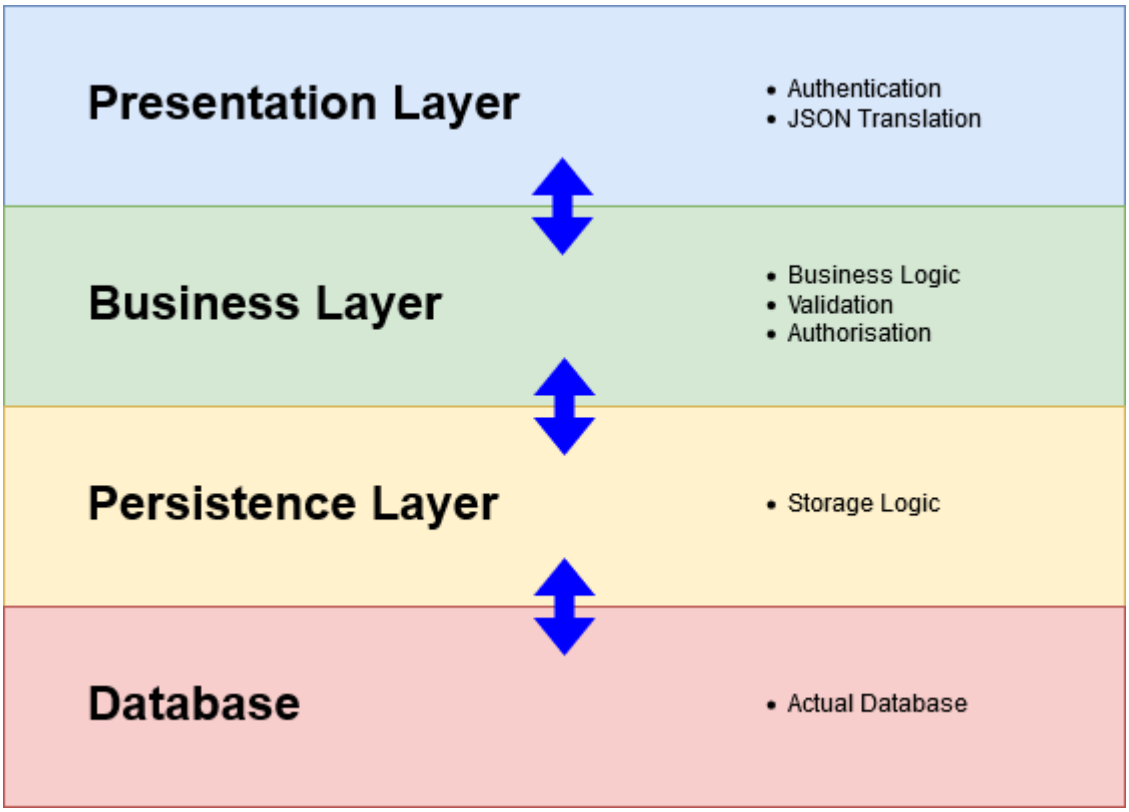

Рисунок 3.5 – Архітектура системи

Запит, що надходить від браузера користувача потрапляє до диспетчеру сервлетів (Presentation layer) [8], який згідно до зазначених в конфігураційних файлах мапінгу направляє запит до правильного і необхідного контролеру. Контролер, якщо це необхідно, викликає сервіси (Business layer), які в свою чергу можуть звертатися до DAO об’єктів (Persistence layer) для отримання доступу до бази даних (Database).

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Вибір засобів розробки

Під час планування та моделювання програмної системи було визначено, що вона має складатися з 3 частин: сервер, які буде відповідати на запити, бази даниих на якій будуть зберігатися дані, до серверу зможуть підключатися клієнти за допомогою REST API. На першому етапі реалізується веб-клієнт та мобільний додаток на платформі Android.

Для розробки серверної частини використовується фреймворк для створення Web-додатків на мові Java з використанням Spring Boot. Java – це орієнтовано-об`єктна мова програмування, яка була випущена Sun Microsystems у 1995 році, як основний елемент платформи Java [9]. Java є платформо-незалежною мовою, тому це досить поширена та популярна мова програмування. Для розробки використовується середовище для програмування Intellij IDEA версії Community Edition [10], яка є безкоштовною для розробки продуктів і має неповний функціонал, але цього буде достатньо. Intellij IDEA є програмним продуктом написанним на мові Java, архітектура якої забезпечує вирішення різних завдань, таких як інтеграцію різних інструментів і мов програмування. Механізми подібної інтеграції дозволяють використовувати Intellij IDEA для програмування розвинених середовищ та систем, звільняють від рутини написання базових елементів, які вже доступні, як шаблони, на користь створення складних, спеціалізованих функцій. Тим самим не тільки вирішується проблема підтримки багатомовних і багатофункціональних середовищ розробки, але і закладається база для спрощенного переходу від одного типу розробки до іншого в процесі їх розвитку. Ця платформа, що буде акумулювати останні досягнення, роблячи їх доступними для розробників конкретних продуктів і усуваючи необхідність докорінної переробки систем [11].

Для виклику серверного API використовуються REST контролери з фреймворку Spring [12], що дозволяє відправляти та отримувати різного формату

дані по протоколу https або http. Серверна частина розгортається на віртуальній машині для збереження коштів та легкої конфігурації інфраструктури [13]. Серед усіх хмарних платформ, найбільш відома і дешева є AWS EC2 [14], яка дозволяє масштабувати систему горизонтально на рівні сервісу, і не потребує великих коштів за послуги. На сьогодні прийнято використовувати REST [15] – (скор. від англ. RepresentationalStateTransfer – «передача репрезентативного стану») – метод взаємодії компонентів розподіленого додатка в мережі Інтернет, при якому виклик методів являє собою звичайний HTTP-запит, а необхідні дані передаються, як параметри цього запиту.

У свою чергу HTTP – це протокол передачі будь-яких даних, що використовується в комп'ютерних мережах. Скорочена назва від Hyper Text Transfer Protocol, протокол передачі гіпер-текстових документів. HTTP – протокол прикладного рівня, схожими на нього є FTP і SMTP. Обмін повідомленнями йде за звичайною схемою «запит-відповідь». Для ідентифікації ресурсів HTTP використовує глобальні URL. На відміну від інших протоколів, HTTP не потребує зберігати свій стан. Відсутній проміжний стан збереження між парами «запит-відповідь». Компоненти, що використовують HTTP, з легкістю можуть самостійно проводити збереження інформації про стан, пов'язаний з останніми запитами та відповідями. Браузер або мобільний додаток, який посилає запити, може відстежувати затримку у відповідях. Сервер може зберігати IP-адреси та заголовки запитів останніх клієнтів. Проте, згідно з протоколом, клієнт та сервер не мають бути обізнаними з попередніми запитами та відповідями, у протоколі не реалізована внутрішня підтримка стану і він не ставить таких вимог до клієнта та сервера.

Для збереження даних такої системи була вибрана MongoDB – одна з найпоширеніших систем керування базами даних. MongoDB [16] — документо-орієнтована система керування базами даних (СКБД) з відкритим кодом, яка не потребує опису схеми таблиць та реляційних властивостей інших баз даних. MongoDB займає нішу між масштабованими і швидкими системами, що оперують даними у форматі ключ-значення, і реляційними СКБД [17], функціональними і

зручними у формуванні запитів. Основна ідея використовування, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [18].

MongoDB розроблена на десятки платформ таких, як ОС UNIX, під Win32 і OS/2 з моменту виходу в Інтернет і являє собою платформу, що швидко розвивається, завдяки великій кількості програмістів, зацікавлених в її розвитку. Найлогічнішим рішенням є вибір хмарного середовища для бази даних - MongoDB Atlas. Облегшення полягає в тому, що це позбавляє необхідності масштабувати базу даних самостійно і перекладає цей аспект на платформу.

Веб-сервіс був створений на базі контейнеру Spring сервлетів і використовує майже усі інструменти Spring Boot Framework.

Spring Boot – масштабована платформа для створення веб-додатків, яка входить до складу платформи Spring. Spring Framework складається з декількох модулів, які надають широкий спектр послуг:

- аспект-орієнтоване програмування: дозволяє реалізувати наскрізні методи;
- інверсія управління: конфігурація компонентів додатків і управління
- життєвими циклами об'єктів Java, здійснюється головним чином через інверсію управління;
- управління транзакціями: об'єднує кілька API, управління транзакціями та координування операцій над Java-об'єктами;
- доступ до даних: робота з реляційною системою управління базами даних на платформі Java з використанням JDBC і об'єктно-реляційних відображень та інструментів з NoSQL баз даних;
- автентифікація і авторизація: регулюються процеси безпеки, які підтримують цілий ряд стандартів, протоколів, інструментів і практик за допомогою Spring Security суб-проекту;
- Model-View-Controller: HTTP сервлети, що забезпечує створення веб-додатків і веб-служб RESTful;
- тестування: підтримка класів для інтеграційних тестів та юніт-тестів.

Платформа Spring – одна із найпопулярніших платформ для розробки додатків з відкритим кодом, призначена для спрощення розробки для J2EE. Вона складається з контейнера, платформи управління компонентами і набору служб для веб-інтерфейсів користувача, збереження стану і транзакцій.

Веб-сервіс був створений на базі контейнеру Spring сервлетів і використовує майже усі інструменти Spring Boot Framework.

Для розробки веб-клієнту використовується JavaScript [19], CSS та html, для розробки мобільного додатку на платформі Android використовується Android Studio та мова Java. Середа розробки, яку необхідно використовувати - це Android Studio, яка розроблена компанією Google. Додаток не має бази даних, усі дані для зберігання відправляються на сервер POST-запитом, а для відображення - GET-запитом.

Четвертою складовою може бути «розумний» пристрій Internet of Things, який надсилає команди на сервер з використанням веб-сокетів, надсилаючи команди сервер їх отримує та повідомляє про це мобільний та веб додатки.

Якщо описувати модель взаємодії усіх компонентів системи, то можна зрозуміти, що головною ланкою між усіма клієнтськими додатками є сервер з REST-сервісом. В усіх чотирьох компонентах системи присутні одні й ті ж класи-моделі для зручності передачі даних.

4.2 Опис програмної системи

Як вже було сказано, вся система складається з трьох компонентів: серверу, веб-клієнту та мобільного додатку. Під час розробки програмної системи однією з цілей було створити найбільш простий інтерфейс у мобільному додатку, шляхом перенесення більшості функціоналу у веб-клієнт. Мобільний додаток має бути максимально спрощеним та зручним у використанні.

4.1.2 Сервер

На сервері реалізований REST-сервіс, за допомогою якого веб-клієнт, мобільний додаток використовують функції системи. Методи, що повертають дані, вертають їх у форматі JSON. REST або передача репрезентативного стану - це архітектурний стиль для забезпечення стандартів між комп'ютерними системами в Інтернеті, що полегшує взаємодію систем між собою. Системи, сумісні з REST, які часто називаються RESTful, характеризуються тим, як вони є без громадянства та відокремлюють питання клієнта та сервера. Наведемо список базових викликів для API:

Для викладача:

- GET /students/{course_id} – вертає список усіх студентів, які записані на цей курс для викладача;
- GET /students/{course_id}/{date}/{time_start} – вертає список студентів з відвідуваністю, на конкретну пару;
- GET /courses/{teacher_id} – повертає список усіх курсів викладача;
- GET /lessons/{course_id} – повертає розклад викладача;
- GET /lessons/{course_id}/today – повертає розклад викладача на сьогодні;
- POST /attendance/{lesson_id}– відсилає фотографію групи до серверу на автоматичну обробку.
- PUT /attendance/{student_id}/{lesson_id} – оновлює інформацію про відвідуваність студента, у випадку ручної зміни викладачем, якщо система не відмітила автоматично;
- GET /statistics/course/{course_id} – повертає статистичні дані студентів і їх присутність і відсутність.

Для студента:

- GET /statistics/course/{course_id}/{student_id} – отримати статистику за відвідуванням конкретного курсу;

- GET /lessons/{lesson_id}/{student_id} – перевірити свою присутність на парі, щоб не прогавити момент, що ти не був відмічений за якихось обставин;
- POST /attendance/student/{student_id}/{lesson_id} – відіслати запит с фотографією своєї присутності у класі, с датою, с гео локацієюДля адміністратора університету:
- POST /teacher– додати вчителя у систему;
- POST /teacher/{teacher_id}/courses – прив’язати курси до викладача;
- PUT /teacher/{teacher_id} – оновити інформацію викладача;
- DELETE /teacher/{teacher_id} – видалити вчителя;
- POST /student – додати студента у систему;
- POST /student/{student_id}/courses – прив’язати курси до студента;
- PUT /student/{student_id} – оновити інформацію студента;
- DELETE /student/{student_id} – видалити студента;
- POST /course – додати курс у систему;
- PUT /course/{course_id} – оновити інформацію про курс;
- DELETE /course/{course_id} – видалити курс;
- POST /lesson – додати урок у систему;
- PUT /lesson/{lesson_id} – оновити інформацію про урок;
- DELETE /lesson/{lesson_id} – видалити урок;
- POST /lesson/{lesson_id}/course/{course_id} – прив’язати урок до курсу.

API надає основний фунціонал, завдяки якому відбувається збір та обробка інформації у системі. Вся ця інформація сберігається у AWS, що забезпечить більш легший контроль данних, також зберегти себе від утрати усіх данних. Також весь сервер с REST буде масшатуватися дуже легко, що дозволить обробляти десятки тисяч запросів, якщо в цьому будет необхідність з боку користувачів такої системи. Все це дозволить бути мобільним і адаптивним до тих ситуацій, які можуть виникнути у системі.

4.2.2 Веб-клієнт

Веб-клієнт має у своєму розпорядженні величезний функціонал, що дозволяє контролювати максимальне усе через веб-інтерфейс. Після авторизації, викладача зустрічає головна сторінка на рисунку 4.1. Де у викладача є можливість перевірити, або змінити деяку інформацію з приводу відвідуваності, курсів, продивитися статистичні дані відвідуваності. Це допомагає викладачеві контролювати свій курс і студентів повнюстю. Звісно ж це скорочує час перевірки присутності і дозволяє розповісти більше матеріалу у відведений час. Звісно у такій системі можливі помилки з визначенням облич студентів і тому ручний метод может не допустити помилок. Від яких впершу чергу будуть страждати студент, які регулярно відвідують заняття.

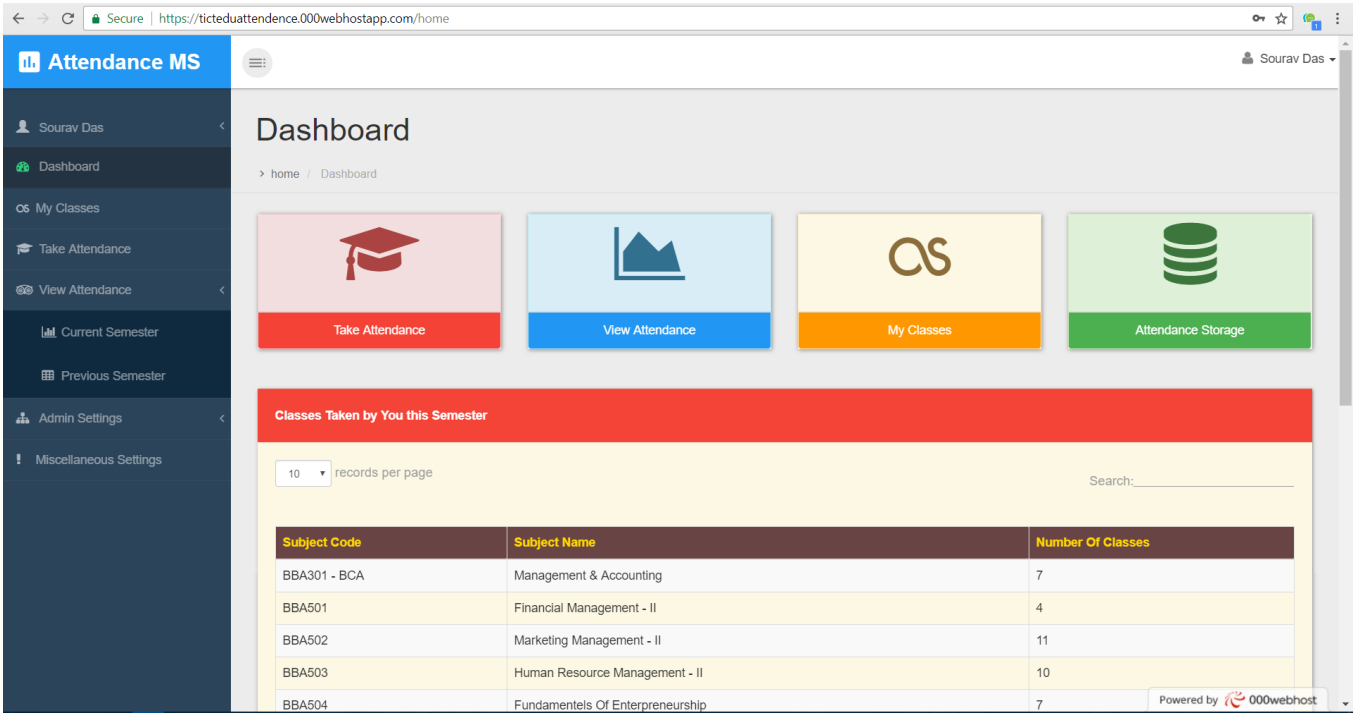

Рис. 4.1 – головний інтерфейс сторінки викладача

Розглянемо ситуацію, коли викладачу треба відмітити присутність або відсутність студента у зв’язку з тим, що система перевірки присутності по

обличчю, дала збій, що може привести до поганих наслідків для тих студентів, які чесно були присутні на заняттях, а система несправедливо їм поставить табличку “відсутній”. Для цього було розроблено адмін панель, де викладач має право власноруч відмітити присутність або відсутність студента, якщо у цьому є необхідність на рисунку 4.2.

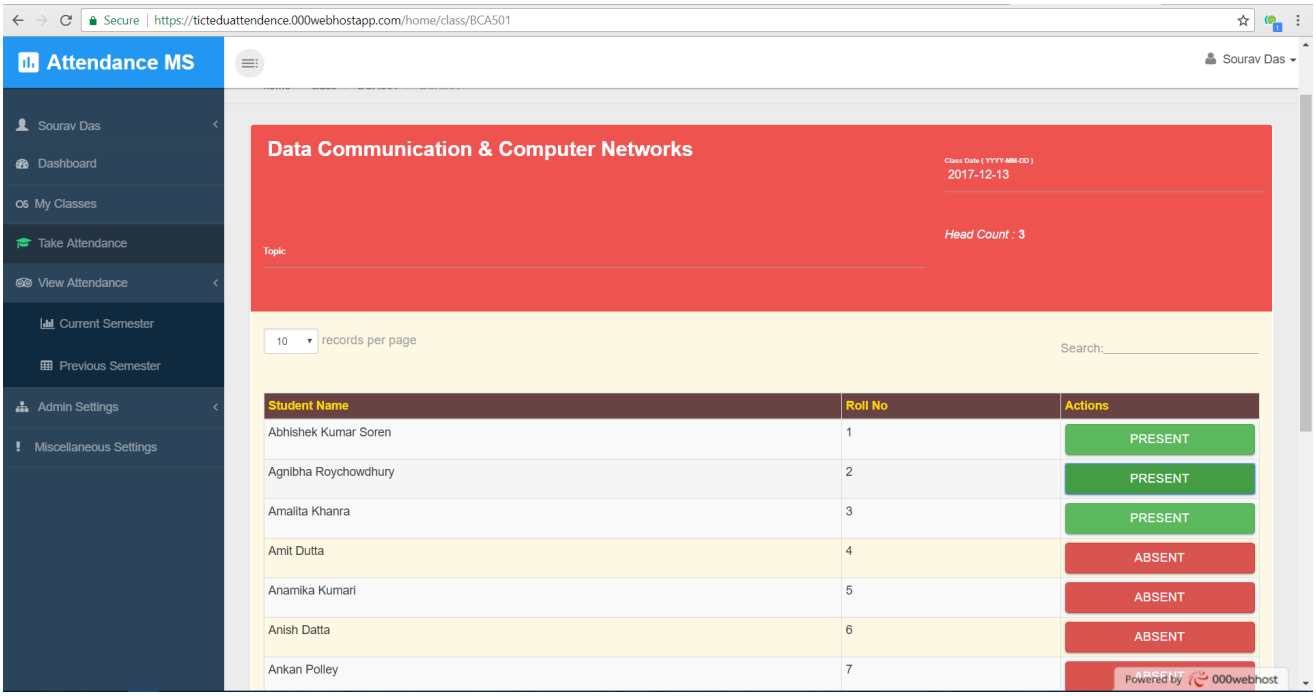

Рис. 4.2 – викладач має право відмічати студентів власноруч

Наступна сторінка, яка має дуже важливу вагу – це сторінка перегляду загальної картини пристуності студентів на конкретному курсі. Це дуже практична сторінка на якій відображається таблиця с даними, такими як: ім’я студента, також кількість класів, які відвідував студент на протязі конкретного часу, а також для повного розуміння, було додано відсоток відвідувань студента від усієї кількості класів, які пройшли за цей час. Тобто, якщо у викладача є якісь умови при яких його курс вважається пройденим, таким як, відсоток присутності студента, то він з легкістю зможе визначити, кто з усього списку студента буде допущений, наприклад, до екзамену, це допоможе контролювати це більш швидше і повідомляти студентів про те, що якщо відвідуваність не збільшиться, то вони просто не здадуть цей курс на рисунку 4.3. Наступний інтерфейс на рисунку 4.4,

який ми розглянемо відноситься до попереднього інтерфейсу. Тому що саме цей інтерфейс показує найдетальнішу інформації з приводу присутності і відсутності учня.

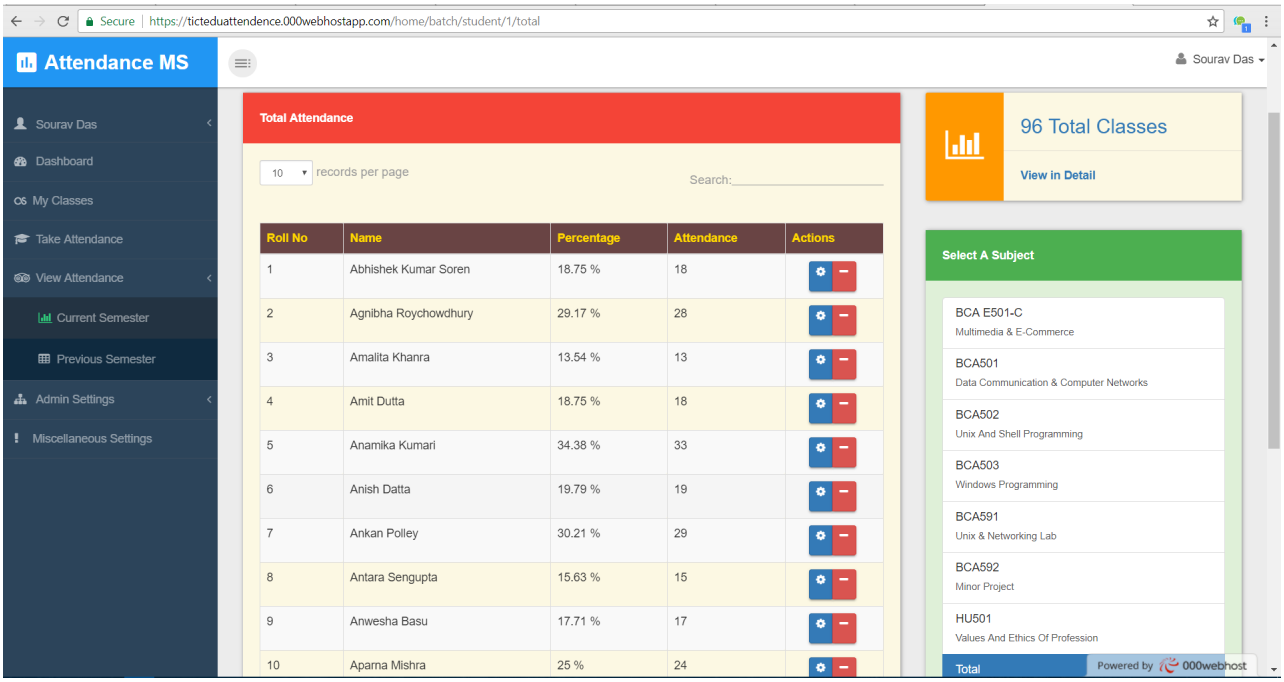

Рис. 4.3 – Інтерфейс загальної статистики присутності студентів на заняттях

Відображається таблиця на якій є ім’я студента, а також дати усіх уроків як колонки, які були у семестрі і напроти кожного прізвища є помітка, на якому саме уроці студента не було. Це допоможе більш детально розібратися у систематичності прогулів і допомогти студентові надолужувати втрачені знання із-зі прогулів.

4.2.3 Мобільний додаток

Для того, щоб почати працювати з мобільним додатком, необхідно для почтаку загрузити та встановити додаток на свій смартфон, у нашому випадку

будуть приклади з Android девайсу. Після того, як викладач або студент відкриють додаток, їх зустріне сторінка авторизації (див. рис. 4.5).

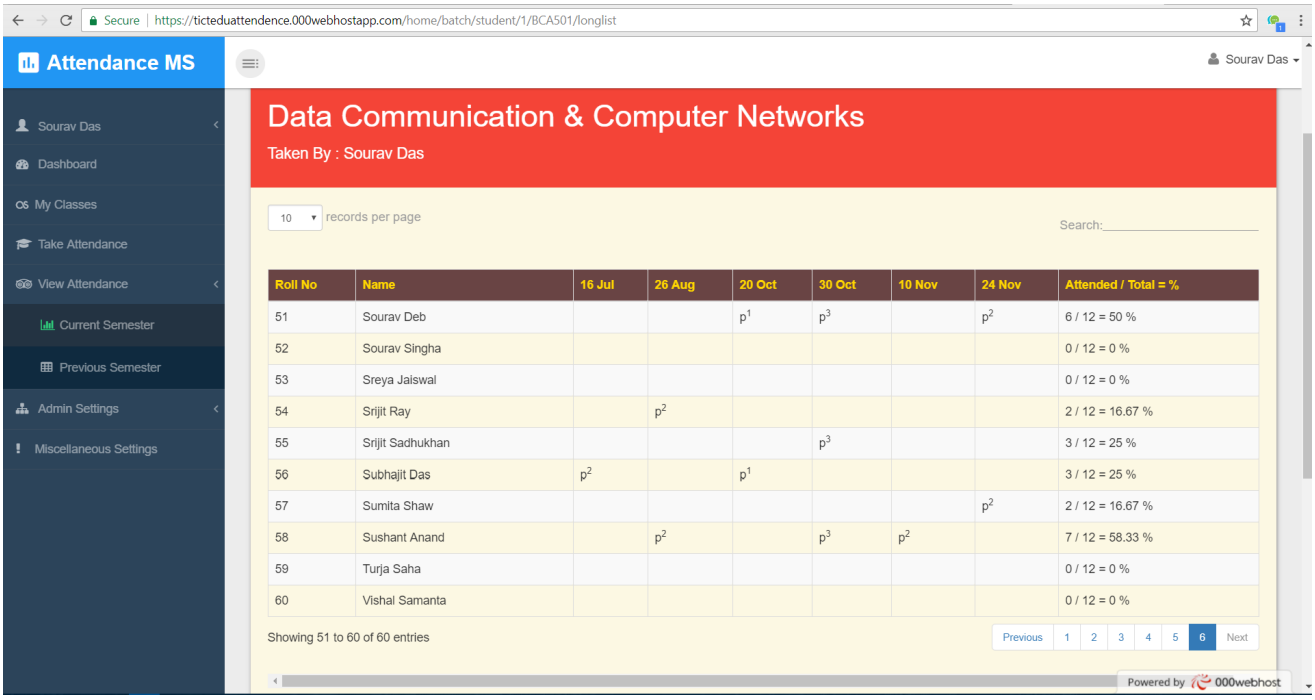

Рис. 4.4 – Інтерфейс детальної інформації з приводу прогулів

Після того, як викладач або студент відкриють додаток, їх зустріне сторінка авторизації (див. рис. 4.5). Для авторизації кожен повинен мати свій емейл та пароль. Неможливо зареєструватися через мобільний додаток, щоб додатком не користувалися люди, які не являються студентом або викладачем. У системі може зареєстровувати тільки готолвний адміністратор навчального закладу, пілся цього він генерує пароль з яким користувачі і можуть заходити у системі під своїм ім’ям. Такий спосіб зменшує ризики псування данних у системі. Звісно кожен користувач після проходження авторизації буде мати право змінити пароль на якийсь інший, якщо сгенерований його не влаштує. Отримати доступ до систему зможуть переглянувши одне з писем у своїй почті, де буде відправлений пароль для входу у систему.

Після авторизації, викладач потравляє на наступний інтерфейс головної сторінки (див. рис. 4.6), звідки він може спокійно почати користуватися системою

у повному обсязі для досягнення основної мети, а саме перевірити присутність студентів.

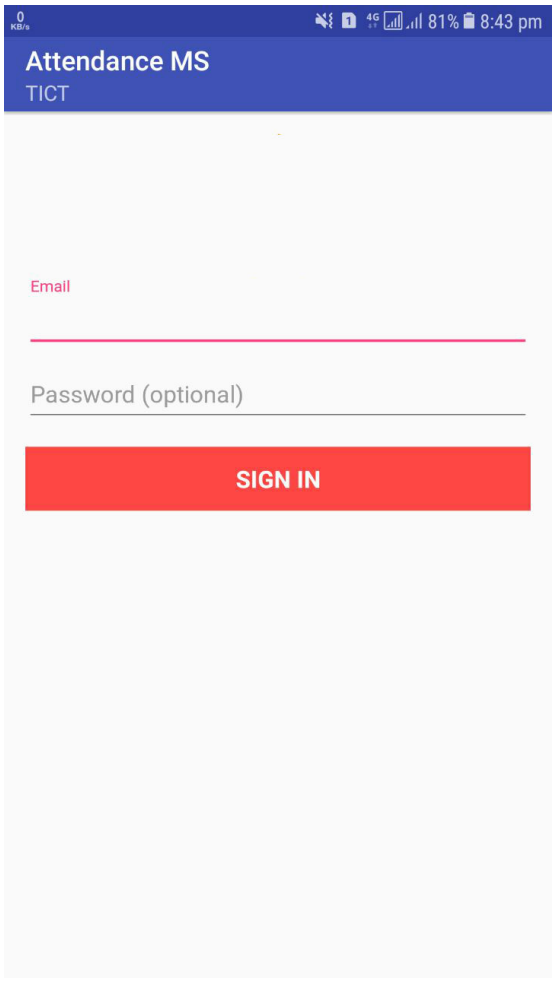

Рис. 4.5 – Інтерфейс авторизації з мобільного додатку

Після того, як він обере поточний курс з поточним заняттям, де буде одразу зрозуміло, скільки студентів мають бути присутніми на занятті і скільки по факту є. Для того, щоб сфотографувати студентів і відправити на сервер, на більш детальну обробку облич студентів, він повинен вибрати пункт “Take Attendance”, автоматично підключиться камера, а потім навести її на студентів, що сфотографувати їх і після цього, фотографія автоматично буде відправлена на сервер на обробку (див. рис. 4.7).

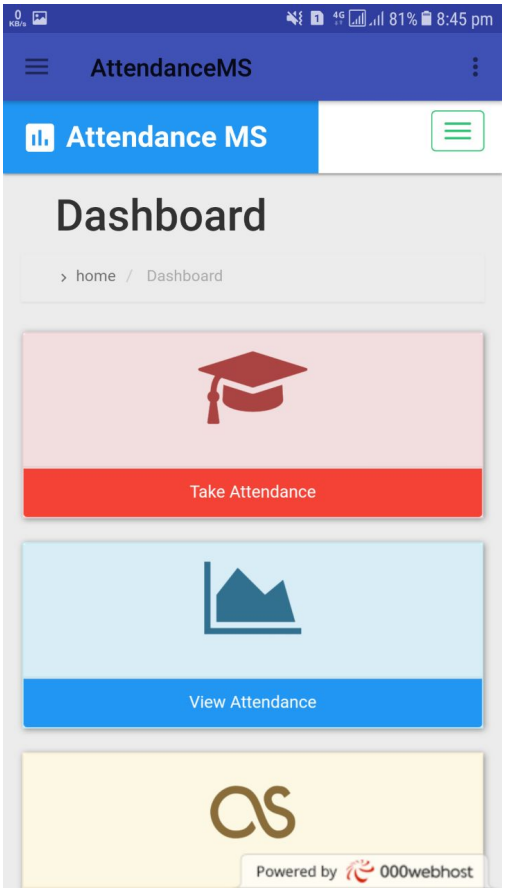

Рисунок 4.6 – Головна сторінка перевірки пристуності і статистики

Звісно ж, додаток намагається виділити обличчя, щоб показати викладачеві, наскільки вдалий кадр був вибраний і скільки людей система не може розпізнати.

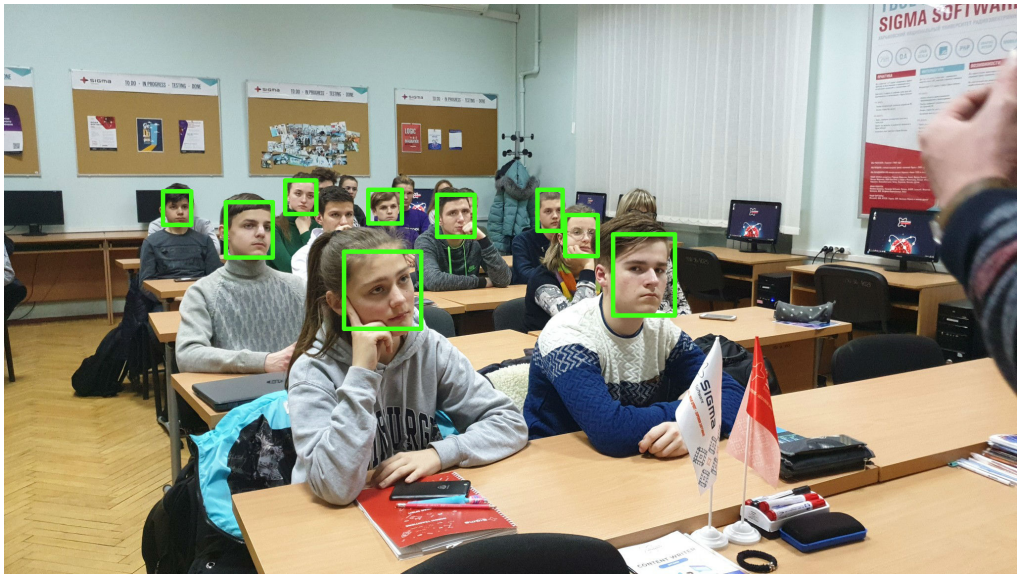

Рис. 4.7 – Камера

Очевидно, що в мобільному додатку не включена бібліотека OpenCV, вона опрацьовує картинки на сервері. В мобільному додатку додана облегшена версія розпізнання облич для викладача.

Після авторизації студент потрапляє на головну сторінку (див. рис. 4.7), де одразує завантажується поточне заняття і інформація був він відмічений чи ні. Якщо цього не трапилося з будь-якої причини, кожен студент має можливість сфотографувати себе через кнопку “Take Attendance”, після цього відкривається фронтальна камера, де студент може сфотографувати себе і автоматично додається на фотографію спеціальні мітки.

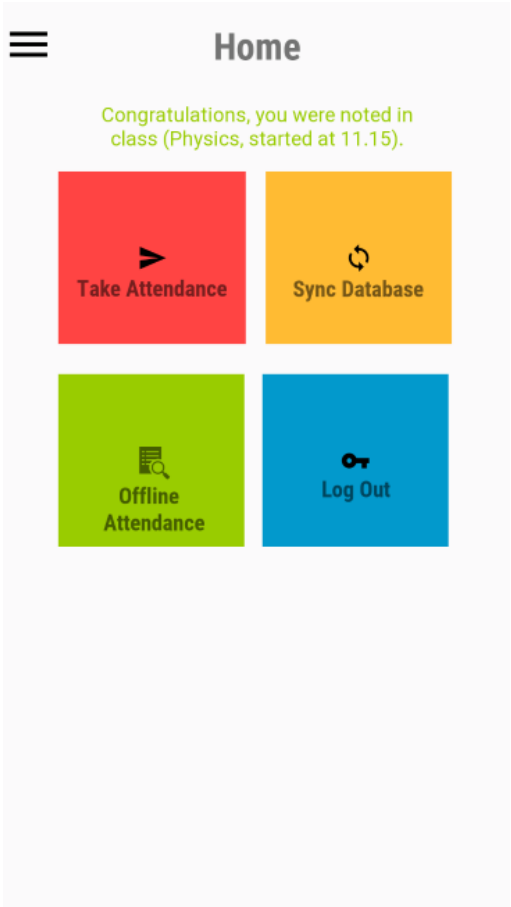

Рисунок 4.7 - Компонент з метриками IoT пристрою

Така фотографія, являє собою доказ студента, що він був присутній на занятті, якщо по будь-яким іншим причинам, вікладач не зміг відмітити

індивідуально у системі і система дала сбій у виявленні такого студента (див. рис. 4.8). Мітки, які включаються у кожну фотографію, дата і час фотографування, гео локація та спеціальний код, який приходить з серверу – відкритий ключ.

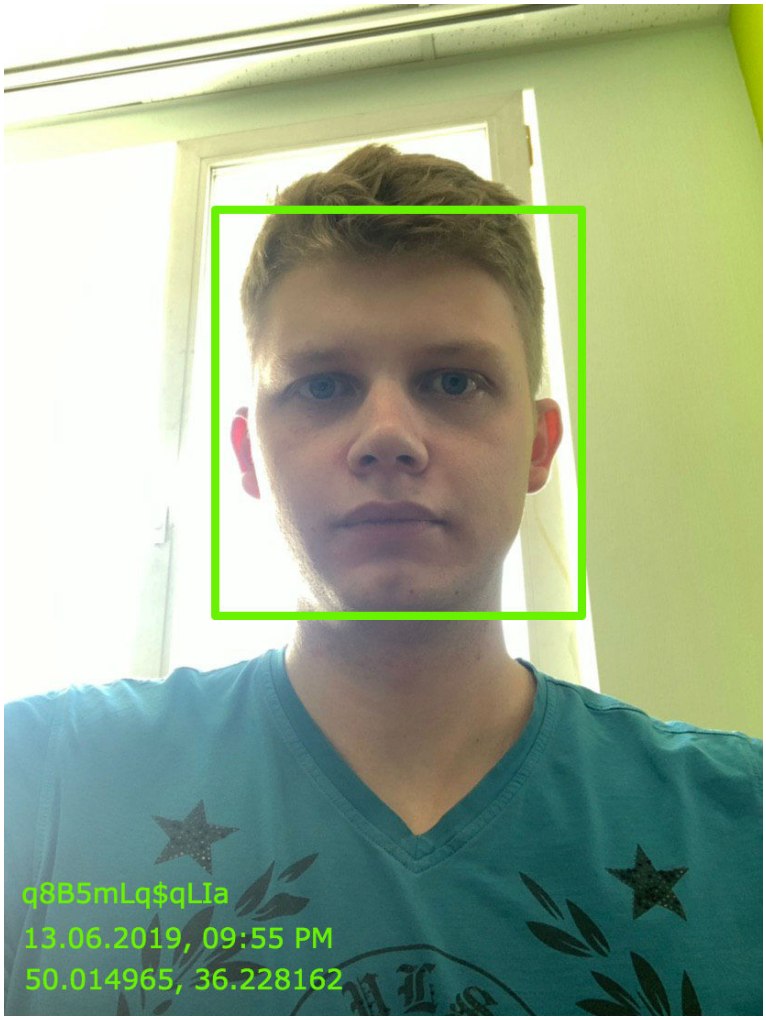

Рисунок 4.8 – Автоматична заявка студента

Тобто це все необхідно, щоб створити онлайн заявку студентові, що є проблема, я не відмічений на занятті, хоча на ньому присутній.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Основною метою будь-якого тестування є перевірка того, що програма відповідає поставленим вимогам, працює коректно навіть у випадку некоректної поведінки користувача та не містить помилок. В ході тестування повинні бути виявлені помилки та невідповідності до вимог. Далі їх необхідно усунути та повторити тестування, щоб пересвідчитися в тому, що виправлення помилок не створило нових проблем. Це регресивне тестування. Усього існує небагато видів тестування, а саме:

- регресивне тестування;
- функціональне тестування;
- тестування зручності використання;
- димове тестування.

Для програмної системи, що розробляється, було проведено усі з перерахованих вище видів тестування. Димове тестування використовується для тестування на явні помилки, що виконується розробником під час написання програмного коду. Більшість тестів, проведених з системою, виконувалися у якості димового тестування (Smoke testing) – на працездатність модулів та явного тестування на помилки. Це поняття димового тестування пішло з інженерного середовища: «При введенні в експлуатацію нового обладнання вважалося, що тестування пройшло вдало, якщо з установки не пішов дим». Такий вид тестування було проведено під час розробки системи. Для виявлення помилок дуже корисним виявився вбудований відладчик інтегрованого середовища розробки Intellij IDEA.

Функціональне тестування полягає в тому, щоб перевірити чи були виконані усі функціональні вимоги і чи правильно вони працюють. Функціональні вимоги до системи були структуровані у вигляді діаграми прецедентів (див. розділ 2). Ці вимоги були перевірені вручну після завершення кодування програмної системи. В ході тестування перевіряється правильність роботи кожної функції.

Наступний вид тестування – це регресійне тестування. Цей вид тестування полягає в тому, щоб після кожної зміни у програмному коді, потрібно перевірити чи не з'явилися нові баги у систему. Найбільш доцільно для цього використовувати автоматичні модульні тести, проте нажаль вони не були реалізовані для даної системи. Натомість регресійне тестування виконувалось вручну після значних змін у програмному коді.

Основна проблема в тому, що число можливих тестів навіть для нескладних програмних компонентів практично нескінченне, стратегія тестування полягає у тому, щоб провести всі можливі тести з урахуванням часу та ресурсів, які є у розпорядженні, а також покрити тестами головну частину функціоналу системи. У розробленій системі є дві частини, що потребують тестування – клієнтська частина та серверна. Бізнес-логіка серверної частини тестувалася шляхом перевірки JSON-відповідей на певні JSON-запити з використанням розширення для браузера Google Chrome – Postman, що зображено на рисунку 5.1.

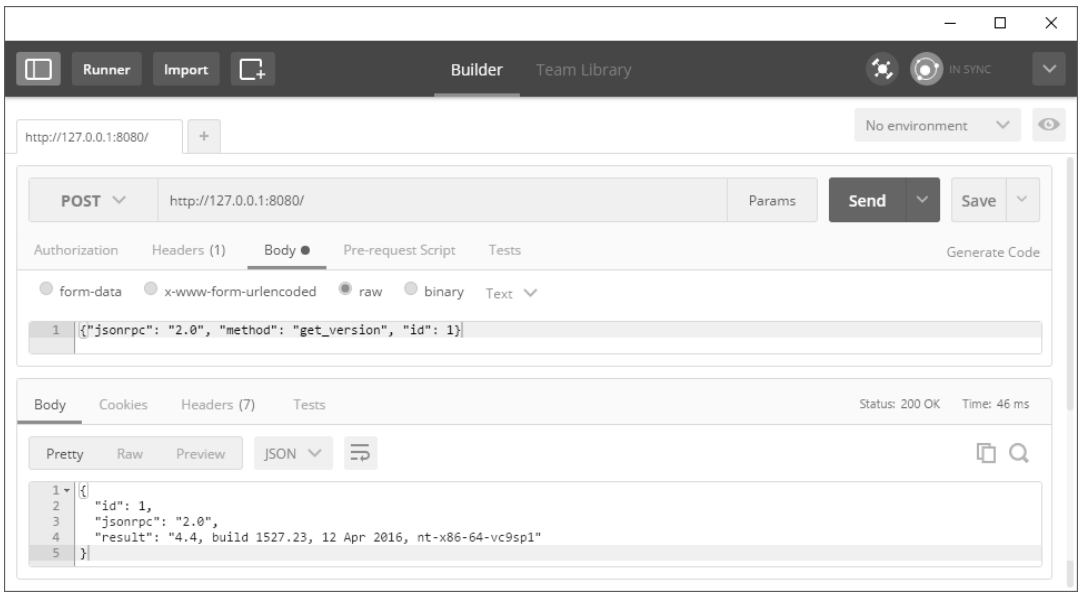

Рисунок 5.1 – Скриншот з програми Postman

Postman дозволяє зберегти структуру та адресу запитів для подальшого використання та групувати їх, наприклад, по типу ресурсу до якого надходять ці запити. Тестування серверної частини за допомогою запитів/відповідей

відноситься до тестування чорної скриньки, тому що при такому тестуванні розглядаються функції програми, а не внутрішня системна робота. Ще тестування серверної частини відноситься до інтеграційного тестування, оскільки тестувалися одразу по декілька компонентів системи разом у взаємодії.

Для тестування окремих дрібних функцій системи було проведено юніт-тестування (модульне тестування) [20]. Модульні тести - це тести, які перевіряють коректність роботи окремої функції або методу. Модульні тести пишуться девелоперами і являються первинною перевіркою того, що внесені зміни не змінили поведінку окремих компонентів системи. За результатами запуску юніт-тестів було покрито 72% строк коду, що є дуже гарним показником.

Також було проведено конфігураційне тестування, у вигляді перевірки працездатності програмної системи для різних видів браузерів. Для перевірки було обрано чотири найпопулярніших браузерів:

- Safari;
- Google Chrome;
- Opera;
- Mozilla Firefox.

В результаті конфігураційного тестування було визначено, що усі компоненти веб-додатку відображалися без помилок та були правильно розташовані щодо одне одного. Увесь функціонал інтерфейсу працював коректно.

В результаті тестування було перевірено, що система відповідає вимогам та поставленому завданню. Виконує всі функції, які вона повинна виконувати. Також програма має зручний та зрозумілий інтерфейс, який дозволяє користувачам виконувати їх задачі швидко та ефективно.

ВИСНОВКИ

У результаті виконання роботи був розроблений програмний продукт, що дозволяє автоматизувати перевірку присутності та відстутності студентів на лекційних і практичних заняттях. Система базується на сервісно-орієнтованій архітектурі, тому складається з окремих частин: сервера, веб-клієнта та мобільного додатку.

Сервер був розроблений з використанням мови програмування Java. Використовувалась середа розробки Intellij IDEA, версії Community Edition. Spring Boot фреймворк, використовувався як набір бібліотек, яка полегшила розробку основної системи. Сервер був розгорнутий у хмарному середовищі AWS EC2. Android на мові Java. База даних розміщується у сервісі MongoDb Atlas.

В результаті виконання роботи було виконано наступні задачі:

- розроблено веб-клієнт з достатнім функціоналом та зручним інтерфейсом;
- розроблено сервер, з сервісами, які надають змогу працювати одночасно різним типам клієнтів з системою;
- розроблено мобільний додаток, основною функцією якого є фотографування групп і відправка на сервер для опрацювання.

В результаті розробки поставлену задачу було цілком виконано. Програмний продукт має зрозумілий інтерфейс для користувачів, не викликає труднощів з відправкою даних на сервер.

Як вже зауважувалося вище, програмна система навіть після реалізації усіх поставлених задач, для комерційного запуску має дороблюватися. Перш за все, бібліотки, які використовувались у розробці, не зможуть покрити усі випадки і кількість помилок буде зростати. Через що така система у рельному житті не зможе працювати у повному обсязі. Для того, щоб така система вийшла у світ, треба дороблювати алгоритми розпізнання облич у реальному часі, та впровадження такої системи у навчальних закладах з мінімальними затратами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Computer Vision [Електронний ресурс]/ Вікіпедія. - URL: https://en.wikipedia.org/wiki/Computer_vision (дата звернення: 20.04.2019).

2. Richard Szeliski. Computer Vision: Algorithms and Applications – New York, 2010 – 979 p.

3. Face Recognition Algorithms [Електроний ресурс] / Digital technologies news. – URL: http://www.ehu.eus/ccwintco/uploads/e/eb/PFC-IonMarques (дата звернення: 12.04.2019).

4. Artificial neural network [Електроний ресурс] / Digital technologies news. – URL: https://en.wikipedia.org/wiki/Artificial_neural_network (дата звернення: 15.04.2019).

5. Face Recognition by Neural Networks based on Gabor filters and Random Projection: – Morocco, Moulay Ismail University, Faculty of Science and Techniques Department of Computer Sciences, 2015 – 56p.

6. Фаулер, М. UML. Основы [Текст] : пер. с англ. А.: Петухов/ М. Фаулер, К. Скотт. - СПб.: Символ, 2017. - 184 с.

7. Мюлер Джордж Р. Проектирование баз данных и UML – Москва: Лори, 2013. – 432 с.

8. Кржиштоф, Ц. Инфраструктура программных проектов: соглашения, идиомы и шаблоны для многократно используемых библиотек [Текст] : пер. с англ. - М.: ООО “И.Д. Вильямс”, 2017. - 416 с.

9. Еккель, Б. Философия Java [Текст] : пер. с англ. Е.: Матвеев/ Б. Еккель. - Спб: Питер, 2016.- 880с.

10. Смирнов, Н. И. JAVA 2 Enterprise. Основы практической разработки распределенных корпоративных приложений [Текст]/ Н.И. Смирнов. – М.: КУДИЦ-ОБРАЗ, 2012. – 240 с.

11. Мартин, Р. Чистый код. Создание, анализ и рефакторинг [Текст] : пер. с англ. Е.: Матвеев/ Р. Мартин. - Питер, 2010. - 464 с.

12. Хо К. Spring 3 для профессионалов – Москва: Вильямс, 2016. – 880 с.

13. James Murty. Programming Amazon Web Services – Washington: O'Reilly, 2009. – 604 p.

14. Michael Wittig, Andreas Wittig. Amazon Web Services in Action – Washington: O'Reilly, 2015. – 424 p.

15. REST API Tutorial [Електроний ресурс] / REST API Tutorial. – URL: http://www.restapitutorial.com/ (дата звернення: 12.12.2018).

16. The MongoDB manual [Електронний ресурс]/ MongoDB docs. - URL: http://docs.mongodb.org/manual/ (дата звернення: 23.11.2018).

17. Крістіан Б., Кінг Г., Грегорі Г. Java Persistence API і Hibernate – Москва: ДМК Пресс, 2017. – 632 с.

18. Кузнецов, С. Д. Основы баз данных [Текст] / С. Д. Кузнецов. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2017. – 484 с.

19. Николенко Д. В. Практические занятия по JavaScript – СПб.: Наука и техника, 2010. – 128 с.

20. Дмитренко В.П. Интернет-тестирование базовых знаний: Учебное пособие - СПб.: Лань П, 2016. - 160 с.