Secure Mobile Attendance System with Face Recognition and Edge Computing

BACKEND IMPLEMENTATION OF MOBILE FACIAL RECOGNITION

THESIS

RESEARCH PROJECT

by

Yowen 2301902390

BINUS INTERNATIONAL

BINUS UNIVERSITY

JAKARTA

2023

Secure Mobile Attendance System with Face Recognition and Edge Computing

BACKEND IMPLEMENTATION OF MOBILE FACIAL RECOGNITION

THESIS

Proposed as a requirement for obtaining

Sarjana degree at

Program Computer Science

Education Level Strata-1 (Sarjana/Bachelor)

by

Yowen 2301902390

BINUS INTERNATIONAL

BINUS UNIVERSITY

JAKARTA

2023

Secure Mobile Attendance System with Face Recognition and Edge Computing

BACKEND IMPLEMENTATION OF MOBILE FACIAL RECOGNITION

THESIS

Prepared by:

Yowen

2301902390

Approved by:

Supervisor	Co-Supervisor
Jude Joseph Lamug Martinez	**Ardimas Andi Purwita**
D4017	**D6405**

BINUS UNIVERSITY

JAKARTA

2023

PERNYATAAN

STATEMENT

Dengan ini, saya/kami,

With this, I/We,

Name (*Name*): Yowen

NIM (*Student ID*): 2301902390

Judul Tesis (*Thesis Title*): Backend Implementation of Mobile Facial Recognition

Memberikan kepada Universitas Bina Nusantara hak non-eksklusif untuk menyimpan, memperbanyak, dan menyebarluaskan tesis saya/kami, secara keseluruhan atau hanya sebagian atau hanya ringkasannya saja, dalam bentuk format tercetak atau elektronik.
Hereby grant to my/our school, Bina Nusantara University, the non-exclusive right to archive, reproduce, and distribute my/our thesis, in whole or in part, whether in the form of a printed or electronic format.

Menyatakan bahwa saya/kami, akan mempertahankan hak exclusive saya/kami, untuk menggunakan seluruh atau sebagian isi tesis saya/kami, guna pengembangkan karya di masa depan, misalnya dalam bentuk artikel, buku, perangkat lunak, ataupun sistem informasi.
I/We acknowledge that I/we retain exclusive rights of my/our thesis by using all or part of it in a future work or output, such as an article, a book, software, or information system.

Catatan: Pernyataan ini dibuat dalam 2 (dua) bahasa, Indonesia dan Inggris, dan apabila terdapat perbedaan penafsiran, maka yang berlaku adalah versi Bahasa Indonesia.
Note: This Statement is made in 2 (two) languages, Indonesian and English, and in the case of a different interpretation, the Indonesian version shall prevail.

Jakarta, 30/06/2023

Yowen

2301902390

BINUS INTERNATIONAL

BINUS UNIVERSITY

Major Computer Science

Sarjana Komputer Thesis

Semester 8 - Year 2023

Secure Mobile Attendance System with Face Recognition and Edge Computing

BACKEND IMPLEMENTATION OF MOBILE FACIAL RECOGNITION

Yowen 2301902390

ABSTRACT

Attendance systems are systems to keep track of attendances of individuals in a company or schools. These systems are usually in the form of biometric scanners such as fingerprint scanners or RFID cards. This thesis focuses on the development of a secure mobile based facial recognition attendance system by employing an edge computing approach. More specifically, the comparison of different facial recognition models and their suitability in an android application without sending confidential data such as images of the user's faces into a separate server. The final model chosen will be taking into account the model size, accuracy, and speed. This model will be used to correctly identify a user in the android application strictly for attendance purposes. This thesis compares the performance of FaceNet, VGGFace, and MobileFaceNet for facial recognition. This thesis will also focus on designing and implementing a backend database system in order to facilitate the mobile based attendance application. The results shows that the MobileFaceNet model was the best choice for this application as it is significantly smaller in size compared to the other two models while still having high accuracy. The model will then be integrated to the android application and the required classes and functions to facilitate this will be developed in Kotlin.

Key words

RFID, android, edge computing, FaceNet, VGGFace, MobileFaceNet, database, Kotlin

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

The first chapter, the introduction chapter, introduces the readers to the project developed by the author and the author's team. It includes the background and motivation of the project, the scopes set for the project, alongside the objectives and aims for the project. The introduction chapter will also provide a brief explanation regarding the structure of the thesis as well as some basic information on what to expect from the remaining chapters.

1.1 Background

In the old days, the use of paper based records as a form of an attendance system was widespread throughout the world. This system was prevalent in schools and in the workplace in order to keep track of a student's/employee's attendance on a daily basis. However, as the years go by, new inventions and breakthroughs are found in a myriad of fields all across the globe with technology being one of the biggest winners [1]. With inventions such as the Internet, it became possible to link a piece of machinery to another in a series of intertwined computer networks. This is essentially the concept of the Internet of Things (IoT), which is a system of connected computing devices both in the form of digital and mechanical machines capable of transferring data between one another in a network [2].

Back to attendance systems, as technology literacy improved throughout society, long are the days where teachers would keep track of their student's attendance in a thick record book. Nowadays, modern attendance systems make use of technology for efficiency and ease of use. This is because paper-based attendance systems require much more manual labor as well as being much more prone to human error such as employees inputting incorrect date and time [3]. On top of that,

paper-based attendance records need to be stored physically thus leading to regular maintenance. This is why there was a need to digitize this sector. A common digital method for keeping track of attendance is in the form of a fingerprint scanner. This works because in general, every living human being has a fingerprint unique to them [4].

Another common modern method for keeping attendance is the use of an RFID card scanner. By issuing everyone a personal RFID card, companies and educational institutions can make use of an RFID card scanner to keep track of daily attendance. This works because everyone in this system will be issued a unique RFID card. Lastly, another common form of keeping attendance is through the use of mobile applications. This way, everyone will be assigned to their own account within the application which they will use for their daily attendance [5].

However, after further contemplation, all of the aforementioned methods come with their own set of problems. When it comes to finger print scanners, the first problem that comes to mind is hygiene [6]. A fingerprint scanner works by tapping your finger, usually your thumb, on a scanner in a piece of hardware. This means that your thumb will be subjected to countless amounts of bacteria on a daily basis. This is because the use of fingerprint scanners can potentially lead to the transmission of microorganisms, including pathogens, from animate sources to the surfaces of the scanners. If the surfaces are not regularly disinfected, these pathogens can remain a continuous source of transmission and pose a risk to the health of users. In particular, the COVID-19 virus, SARS-CoV-2, can remain active and infectious on surfaces for several days [6]. The simple solution is to wash one’s hands after taking attendance. However, you cannot assume that every individual is willing to maintain this basic level of hygiene. This problem is also further exacerbated with our current state of the world. Another problem with the use of fingerprint scanners is the need to purchase hardware. This means that companies will need to invest some additional overhead cost before they can make use of this system.

Next, when it comes to RFID card scanners, the first problem that comes to mind is anonymity. Anonymity can be good a lot of the time, but not when it comes to attendance systems. Anonymity here refers to who actually did the attendance. RFID card scanners are capable of deducing who an ID card is referring to but not who was the one actually performing the tapping. This way, it is possible someone could cheat the system by having someone else do their attendance for them while they are elsewhere and should instead be marked as absent. This phenomenon is known as 'buddy punching' and refers to when an employee's attendance is done by another employee to cheat the system [7]. The problem with buddy punching is when it turns into a habit. This in turn usually leads into a verbal warning, followed by a written warning, and eventually, in some cases, dismissal of said employee [7]. Another problem with this method is, like the previous method, requires the purchase of additional hardware to integrate. Companies will have to purchase both an RFID card scanner and the RFID card itself for each of the employees before they can start using this system of attendance. Lastly, a common problem with this method is due to human error. It is very likely that someone will somehow lose their RFID card. This can be a hassle as the company will then have to purchase a new card to be issued to said person.

Lastly, when it comes to mobile based attendance systems, the main problem is related to malicious users spoofing the system [8]. Spoofing refers to the act of falsifying or imitating something with the intention of deceiving or misleading others. Mobile based attendance systems track the validity of a user's attendance by monitoring the user's current GPS location. This makes it possible for the user to cheat the system by spoofing their GPS [8].

In this project, the author's team will be developing a new secure mobile attendance system in order to combat the flaws of the previously mentioned methods. After considering the disadvantages of all the aforementioned methods, the author's team has deduced that it is much more feasible to tackle the flaws of this system as opposed to the other methods by adding additional security checks to prevent user spoofing. These additional prevention methods will be

discussed further in chapter 3. The scope and role of the author will be explained in further detail in section 1.2.2.

1.2 Scope

This section defines the scope of the project as a whole as well as the different responsibilities of the author's team members throughout the project. It will also go further into detail regarding the author's personal scope.

1.2.1 Group Scope

This project is a mobile attendance application which makes use of facial recognition to ensure the user is physically there onsite. This solution aims to counter the different problems found on the aforementioned attendance system methods. In addition, to enhance data confidentiality and privacy, the application incorporates an innovative approach known as edge computing. Edge computing enables the processing and analysis of data at the edge of the network, closer to the source of data generation, rather than relying solely on a central server. This ensures high data confidentiality and privacy as no images of the user used in the facial recognition will be stored nor sent to the server. Thus, in the case of a perpetrator intercepting the data sent, they will not find any meaningful information. Once the facial recognition system has verified the user, only their attendance will be sent to the database.

The following table explains in further detail the roles of each member throughout the project.

Table 1.1: Role distribution

Name	Role
Yowen	- Performing evaluation in performance based on size, speed, and accuracy of different face detection and face recognition algorithms - Implementing face authenticity detection feature into android application - Developing android application functions and classes that handles integration of chosen model into the android application - Designing and implementing a database system for the application
Bently Edyson	- Usability testing on people's behavior on application - Unit testing on each important function and classes - Integration testing on ai face recognition

	- End to end testing to test whether when user use it, will it find any bug or error
Bryan Putra Suandi	- Designing user diagram, flow, wireframe, and mockup for the android mobile application - Developing the front end of the android mobile application - Implementing a network-restriction user access feature to prevent users from using the application outside of the intended range or fake GPS

1.2.2 Individual Scope

In this project, the author's responsibilities are to:

- Evaluate the performance of different face detection and face recognition models on a dataset based on model size, inference time, and F1 score.

- Integrate best model according to evaluation results while taking into account specified group aims criteria
- Implement a way to ensure a given face is not an image to prevent image spoofing
- Implement a database system for the android application

1.3 Aims and Benefits

This section goes into detail the different aims and benefits the author and his team hopes to attain with this project.

1.3.1 Group Aims

The aim of this project is to develop an attendance system with both ease of use as well as data privacy. The following are the group metrics set for the project:

- User usability acceptance rate of 70% or higher [9]
- Model inference time of 1 second or lower [10]
- Model size of 25MB or lower [11]

According to J. Sauro [9], the average usability acceptance rate is 68% and anything below that is considered as below average. Hence, the author's team has decided a baseline goal of 70% for the project. Maximum model inference time is decided based on a study from Google [10], which shows that mobile website visitors will leave a page if it takes longer than 3 seconds to load. Although this does not directly translate into the context of a mobile application, it gives an insight as to how long mobile users are willing to wait for a good user experience. As for model size, the average top android application size across categories as of 2022 is 60MB [11]. This serves as a baseline for the estimated finished application size of the project. Since the face recognition model

will be bundled within the application, the author's team has decided to allocate a maximum of roughly 50% of the total application size for the model itself.

1.3.2 Individual Aims

The individual aims of the author can be summarized as to:

- Provide a method to evaluate the performance of 2 face detection and 3 face recognition models that meets the group aims criteria for an edge computing use case
- Design and implement database system for the android application based on client criterias and needs

1.3.3 Group Benefits

The main benefits of this project can be summarized into the following points:

- Provide a modern attendance system with ease of use for the users
- Offline edge computing approach means user facial information will not be sent to a separate cloud server
- No additional hardware necessary for system to function compared to other methods
- Digital data means no need for physical storage and regular maintenance

1.3.4 Individual Benefits

The benefits of the author's individual aims for this project can be summarized into the following points:

- The evaluation method proposed ensures an objective selection process based on testing results for face detection and face recognition models and can help guide other projects with a similar need
- The database will be able to handle all the client's criteria and needs as well as being seamlessly integrated into the android application

1.4 Structure

This section explains the different chapters found within this thesis.

1.4.1 Chapter 1: Introduction

This chapter serves as an introductory chapter. It contains the background, scope, as well as the aims and benefits of the project.

1.4.2 Chapter 2: Theoretical Foundation

This chapter explains in detail the theories in which the author makes use of to achieve the author's aim for this project.

1.4.3 Chapter 3: Problem Analysis

This chapter explores the problem in further detail and explains why the author has chosen this implementation for the project. It also discusses some other works related to the author's project.

1.4.4 Chapter 4: Solution Design

This chapter explains how the author plans to achieve the set individual aims. It explains the methods and steps to achieve the author's personal scope and aims for the project.

1.4.5 Chapter 5: Implementation

This chapter will show the results obtained from the different methods from the previous chapter. It also goes into detail about the backend database implementation developed by the author.

1.4.6 Chapter 6: Discussion

This chapter will mainly discuss the results from the different models tested and evaluate their benefits and drawbacks.

1.4.7 Chapter 7: Conclusion and Recommendations

This chapter acts as a conclusion for the entire thesis. It will also provide several recommendations on how the project can be further improved in the future.

CHAPTER 2

THEORETICAL FOUNDATION

2.1 Attendance

Attendance refers to the presence of an individual or availablity within a particular event, gathering, or activity. Attendance is commonly used to measure and track how often an individual is present in settings such as schools, workplaces, or meetings. Attendance is typically recorded to monitor and ensure the reliability, punctuality, and accountability of individuals. It can be measured either by physically marking one's presence, signing in on a register, using electronic systems, or through other means of verification.

2.1.1 Attendance Systems

Attendance systems are tools or mechanisms used to keep track of the attendance of individuals, usually employees or students in an organization or educational institution [12]. It is used to keep track of the presence, absence as well as tardiness or early arrivals of each individual in the system [12]. This information is then stored and can later be used in the future such as calculating total work hours of a certain employee.

There are a variety of ways to implement an attendance system. One of the most common forms of attendance system is in the form of biometric scanners such as fingerprint scanners. A fingerprint scanner is a piece of technology that identifies the identity of a person by analyzing their unique fingerprint [13]. Another common form of taking attendance is in the form of RFID

cards [14]. This form of attendance management works by issuing a unique RFID card to each employee or student. This unique RFID card will then be used with an RFID card reader to identify the identity of each individual.

2.2 Face Detection

Face detection is the process of identifying and locating human faces in a digital image or recording [15]. It is a computer technique powered by artificial intelligence with a wide range of applications. There are a variety of methods in which face detection algorithms can detect a face in a given image.

Feature based face detection methods work by attempting to extract the facial features of a face [16]. This method first involves the training of a classifier to identify between facial and non-facial regions [16]. The purpose of this is to find patterns in features that even we, as humans, might overlook when classifying whether something is a face or not [16].

Knowledge based face detection methods work by enforcing a set of rules constructed by the individual who created it regarding their understanding of what is or is not a face [16]. For instance, the rules created may specify that a face should include eyes, a nose, and a mouth at specific intervals along a predetermined scale. However, there is a major drawback regarding this method. It is quite challenging to create a suitable set of rules [16]. If the rules are too narrow, the system may produce a significant number of false negatives. If the rules are too wide, the system may produce a big number of false positives.

Template matching face detection methods work with the use of parameterized or pre-defined templates in order to discover or recognize faces in input images [16]. The system evaluates how

well the input photographs and the templates agree. The template may show, for instance, a human face divided into sections for the nose, mouth, eyes, and facial contours. In addition, a facial model may merely consist of edges, in which case the edge detection method might be applied. Although the implementation of this method is simple, it is insufficient for face recognition due to it having difficulties in detecting faces with variation of lighting and poses [16].

Appearance based face detection methods work by training a machine learning model on what a face should look like by giving it a collection of faces as training photos [16]. This method is generally considered to be much more accurate than others with the drawback being that the rules defined by the model are too intricate and complex for humans to interpret [16].

2.2.1 Haar Cascade

Haar cascade classifiers are machine learning based approaches developed by Paul Viola and Michale Jones in which a cascade function is trained using a large number of both positive and negative pictures [17]. After training, it can be used to detect objects within a given image making it suitable for face detection purposes.

Firstly, the algorithm is given a large amount of positive images, in this case being images that contain a face, as well as negative images, which are images not containing a face. Next, a set of haar features is defined in order to help finding the edges or lines in an image [18].

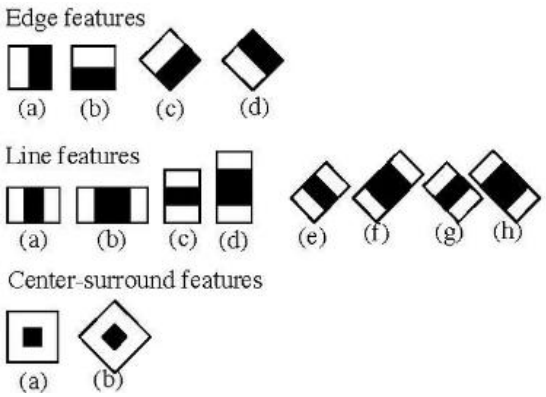

Figure 2.1: Haar features

These haar features help the algorithm detect the presence of an edge or a line in an image. For example, it can detect an edge feature if the difference in sum of all the pixels on the lighter side and sum of all the pixels on the darker side is close to one [18]. These haar features continuously traverse the entire image in order to locate all the different types of haar features in an image. Once all the haar features are extracted from an image, facial features and patterns such as eyebrows and lips can be detected due to the difference in pixels between the two regions [18].

2.2.2 Google MLKit

The Google MLKit is a mobile SDK developed by Google that is capable of bringing Google's on-device machine learning models into an Android or IOS application [19]. Google ML Kit enables developers to quickly add machine learning capabilities to their projects. It offers a collection of pre-built models for typical machine learning applications like text recognition, face detection, barcode scanning, and picture labeling.

The Google MLKit face detection algorithm is based on BlazeFace which is a lightweight face detector tailor made for mobile based GPUs [20]. This makes it capable of running at high FPS on

flagship mobile devices. The Google MLKit face detector is capable of detecting facial landmarks, facial contours, facial boundaries, as well as probabilities such as the smiling probability of a given face [21].

2.3 Face Recognition

Face recognition is a method of recognizing or verifying a person's identity using their face [22]. Individuals may be recognized using facial recognition technology in real-time or in still images and videos. Face recognition is a form of biometric authentication.

Face recognition works by firstly detecting the presence of a face in an image or video [23]. Once a face is detected, the data is preprocessed and its pixels normalized before being fed into a face recognition model. The face recognition model will then output a digitalized facial representation of the input face known as face embeddings [23]. The face embeddings can then be used to calculate whether 2 images of a face belong to the same person or different person.

2.3.1 Pixel Normalization

In computer vision, pixel normalization is a form of image processing that converts the pixel values to a specific scale or range in order to standardize and normalize the values [24]. Pixel normalization is done in order to speed up the speed in which the model takes to train as well as to increase its accuracy [24]. Common methods for performing pixel normalization involve dividing each pixel value with the maximum value it can take, that is 255 for 8 bit images, 4095 for 12 bit images, and 65535 for 16 bit images [24].

2.3.2 Cosine Similarity

Cosine similarity is a metric used to measure the similarity between two different vectors [25]. Cosine similarity refers to the cosine angle between two vectors. It is calculated by dividing the dot product of the two vectors by the product of their magnitudes. The formula to calculate the cosine similarity between two vectors is [26]:

$$similarity(A, B) = cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$

Where:

- θ is the angle between the two vectors
- A and B represents the two different vectors
- $A \cdot B$ is the dot product between the two vectors
- $||A||$ represents the L2 norm or magnitude of a given vector

Cosine similarity results lie between -1 and 1. 1 Indicates that two different vectors are identical, 0 indicates that the two vectors are orthogonal while -1 indicates that the two vectors are diametrically opposed [26].

2.4 Machine Learning

Machine learning is a field of study in artificial intelligence that involves the development of models that are able to compute and make predictions or decisions based on a given input data without being explicitly programmed [27]. Machine learning focuses on training models to recognize patterns in data and utilizing that data to carry out specific tasks. Machine learning

algorithms can generally be divided into two different categories, supervised machine learning, and unsupervised machine learning.

2.4.1 Supervised Machine Learning

Supervised machine learning involves providing the model with a set of labeled examples [28]. For example, a model trained to recognize between spam and non spam emails would have been trained using data marked as either spam or non spam. The aim of this method is to train a model capable of finding patterns in data in order to map the input to the expected output [29]. It is generally used for regression and classification problems [29].

2.4.2 Unsupervised Machine Learning

Unsupervised machine learning leans more towards ‘learning by yourself’. In unsupervised machine learning, the model is given unlabeled data and is expected to discover patterns and relationships in the data without additional guidance [29]. The aim of this method is to train a model capable of clustering input data into an unknown amount of classes. The amount of classes is unknown as it is not possible to predetermine the amount of classes the final model will be able to classify the input data into.

2.5 Neural Networks

Neural networks are a part of machine learning and the foundation of deep learning algorithms [30]. It is structured and modeled based on the way a human brain works, that is mimicking the ways in which biological neurons send signals to one another. Neural networks are composed of interconnected nodes each with their own weights and values. Neural networks learn by refining

the weights and values of each node during training. A simple neural network consists of three main components [31].

The first layer, the input layer, is where preprocessed training data is fed into the algorithm [31]. Next, it passes the data received onto the next layer, the hidden layer. The hidden layer acts as the intermediate layer in a neural network between the input layer and the output layer. It contains one or more layers within it. In general, the larger the amount of layers in a hidden layer, the more complex a neural network algorithm becomes [32]. Thus, the amount of layers in a hidden layer varies depending on the problem trying to be solved by the algorithm. Finally, the output of the final layer in a hidden layer gets passed onto the output layer. The output layer is the final layer in a neural network and is usually in the form of a prediction or classification [33]. Typically, the neurons in the output layer use a particular activation function that is suitable for the task at hand. For example, the output neurons may make use of the sigmoid activation function for a binary classification task [33]. This in turn yields a number between 0 and 1 that refers to the chance of being in the positive class. On the other hand, when dealing with multi-class classification problems, a softmax activation function may instead be used in the output layer with each neuron indicating the likelihood of belonging to a specific class [33].

2.5.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) is a type of neural network widely used to process images and video data [34]. It involves the use of convolutional layers which performs a convolutional operation on the input data. Convolutional operations involve the use of a filter over the input data to compute the dot product between the filter and the area of the input currently being analyzed. A feature map that highlights significant spatial patterns in the input data is created by repeating this procedure across the full input dataset [34].

CNNs generally have additional layers in addition to the convolutional layer, such as pooling layers and fully connected layers. The feature maps created by the convolutional layer are

downsampled using pooling layers, which reduces the amount of data and increases the network's effectiveness [34]. The final output of the network is calculated using fully connected layers, usually by making use of a softmax activation function to the output of the last layer [34].

2.6 Model evaluation

Model evaluation refers to the process of analyzing the performance of a model based on the task it was designed for. It is an important step as it determines how well a model performs and whether it will be suitable for deployment. The metrics used during model evaluation differs based on the specific task being performed [35]. This means that evaluation of a binary classification model will be different to that of a regression model. For example, binary classification models may use metrics such as accuracy, precision, recall and F1 score to evaluate performance while regression models may instead use mean squared error (MSE) or mean absolute error (MAE) during evaluation [35].

2.6.1 Accuracy

Accuracy is a metric used for evaluating models tasked for classification purposes. It is a simple metric that denotes the proportion of correctly classified instances out of all the instances in a given dataset [36]. The formula for accuracy is the following [36]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- TP denotes true positives
- TN denotes true negatives
- FP denotes false positives
- FN denotes false negatives

Accuracy is a simple metric that can be used to measure a model's performance. However, its main drawback is that it does not take into account the balance of data in the given dataset. For example, a model tasked with classifying between a dog and a cat will still achieve an accuracy of 90% even if it predicts dog for all data when given an unbalanced dataset composed of 90 dogs and 10 cats.

2.6.2 Precision and Recall

Precision is a metric used for evaluating models tasked for classification purposes. It is a metric used to calculate the ratio of true positives predicted by the model over the total positive predictions a model made [37]. On the other hand, recall is a metric used to calculate the proportion of true positives that was identified correctly. The formula for recall and precision is the following [37]:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Where:

- TP denotes true positives
- FP denotes false positives
- FN denotes false negatives

Precision and recall are usually in a tug of war situation [37]. For example, a model tasked with classifying whether an email is a spam or not might have a higher precision if the threshold for whether an email is a spam or not is increased. This however leads to lower recall as more emails that are spam will be misclassified as normal email. Conversely, if the threshold is lowered, the model's precision will drop and recall will increase due to it mislabeling more normal email as spam emails.

2.6.3 F1 Score

F1 score is a metric that takes into account both precision and recall values to evaluate a model. It is a harmonic mean of precision and recall and provides a balanced evaluation of the model's performance. The formula to calculate a model's F1 score is the following [38]:

$$F1\ score = 2 * \frac{(precision * recall)}{(precision + recall)}$$

The F1 score offers a technique to find a balance between precision and recall trade-offs and offers a single statistic for contrasting various approaches. A model with a high F1 score will have high recall and precision, thus being able to properly identify positive occurrences while avoiding false positives and false negatives [38].

2.7 Database

A database is a collection of organized structured data stored in a computer system. It is accessed and manipulated through the use of a database management system (DBMS) [39]. In a database, data is organized into multiple tables each of which containing columns relating to the table. For example, a database for a school might have tables to store each student's information as well as tables to store the different types of classes present in the school. Through the use of a DBMS, the school can then create, update and query information into the database. An example would be querying students whose last name starts with the letter 'B'.

2.7.1 Firestore

Firestore is a flexible and scalable cloud based NoSQL document database provided by Google Cloud Platform [40]. It is designed to store and manage data for web, mobile, and server

applications. In firestore, data is stored in a document oriented model with fields of key-value pairs. Documents can then be grouped into collections providing hierarchical structure for data organization. Due to its flexible nature, individual documents in the same collection can have different fields with varying data types. Firestore offers real-time data synchronization, automatic scaling, and a powerful querying mechanism, making it suitable for a wide range of applications.

2.7.2 Entity Relationship Diagram (ERD)

Entity relationship diagrams (ERD) are visual representations of the different tables and relations present in a database system [41]. It contains standardized symbols and notations to depict the layout of a database. Relationships between tables can also be differentiated with the use of crow foot notations. The following figure shows examples of crow foot notations [41]:

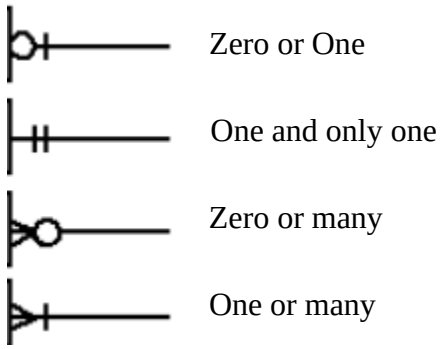

Figure 2.2 : Crow foot notations

ERDs are frequently used in the early phases of database design to help understand the database's structure and the relationships between the different entities and characteristics. Before the database is put into use, they may be utilized to find any possible problems or inconsistencies in the data model.

2.8 Use Case Diagram

Use case diagrams are graphical representations that illustrate a user's capabilities when interacting with a system or application. In use case diagrams, use cases or processes are represented by ellipses while the actors are represented as stick figures. Arrows are used to represent the relationship between one use case to another. There are two main types of relationship in a use case diagram, that is, includes and extends [42].

A relationship is deemed to be 'include' when a use case is considered as part of another use case. For example, when logging in to an application, the application will have to verify the credentials of the user. This verification process and logging in process counts as an include relationship as the verification process counts as part of the logging in process as a whole.

On the other hand, a relationship is deemed to be 'extend' when a use case is considered as an optional part of another use case. For example, using the scenario from above, when logging in to an application, an error popup can occur if the credential is invalid or if there are other problems such as poor network connection. This error pop up process and logging in process counts as an extend relationship as the error popup process will depend on the result of the logging in process and not occur every time a user runs the login process.

The figure below illustrates the different symbols commonly present in a use case diagram:

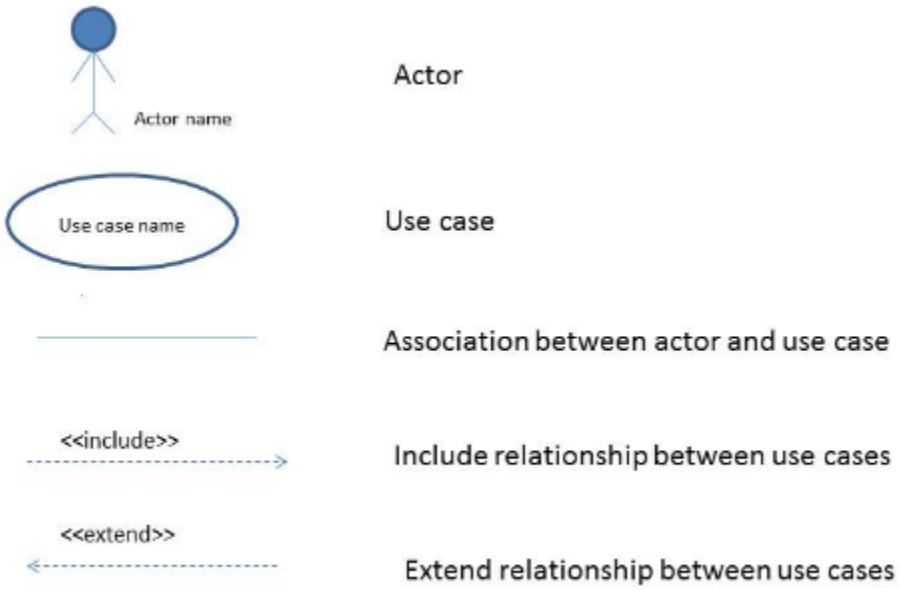

Figure 2.3 : Use case diagram

CHAPTER 3

PROBLEM ANALYSIS

This chapter explores the problem in further detail and explains why the author has chosen this implementation for the project. It also discusses some other works related to the author's project.

3.1 Problem Statement

Based on the assessment of the different methods for attendance systems, the mobile application method seems to have the least drawbacks. This is why the author and his team have decided to choose this as the basis of their project. The edge computing approach is taken to ensure high data confidentiality as no user pictures will be sent. Hence, the main problem the author has to address throughout this project is the transition of running the face recognition model from a high computing power PC to a lower computing power mobile device.

There are several factors the author has to take into account due to the transition mentioned above. Firstly is regarding model size. In general, the larger an application size is, the less likely a user is going to be willing to download said application [43]. This can be due to various reasons such as bigger storage space requirements or simply being discouraged by long download times. Because the author and his team plans to run the model on the mobile device itself, the model that the author selects for this project will be bundled within the application and thus greatly impacts the size of the application. This is why it is important for the author to ensure the model chosen for this project is not too large and is reasonable.

The other factor the author has to consider due to this transition is the reliability of the model's output. Due to a reduction in the desired model size for this project, the reliability of the model will be impacted [44]. Model accuracy and complexity is a common trade off to consider. Reducing a model's size means reducing its complexity. This in turn leads to better performance on lower computing power devices. However, this can also cause a reduction in model accuracy. This is why it is important to ensure the model chosen for this application can be both relatively small in size as well as maintaining high accuracy.

As of now, there are no clear guidelines for choosing a face recognition model to be used in an edge computing approach for mobile application. Hence, there are currently no third party services that provide ready to use API for edge computing mobile based face recognitions. Thus, the author has to develop the necessary android classes and functions to facilitate these needs.

As for the database, several features must be taken into account when designing the structure. On top of facilitating attendance, the application will also facilitate leave requests. The leave request feature is used by employees to request a leave for a specified amount of days. There are two types of leave requests that need to be considered. The first type is paid time off (PTO) while the latter is known as 'permissions'. Permissions typed leave requests are used to accommodate occasions whereby said employee is unable to attend work due to health conditions. In addition to leave requests, another feature the database will have to accommodate is the correction requests. Correction requests are requests issued by employees to correct faulty attendance. There are several reasons an employee would create a correction request. An example of this is on the occasion whereby an employee forgets to tap in or tap out on a specified date. Another example is if an employee would like to change the date of their leave request. Both correction and leave requests will need to be approved by an administrator in the system before taking into effect. This means that each request can either be in a pending, approved, or rejected state.

On top of the base features mentioned above, there are several other additional features requested by the client to facilitate their needs that will be taken into account when designing the database. Firstly, in order to prevent employees from cheating the system by performing attendance at home, the client has requested that the attendance process can only be done under the company's wifi connection. Next, for the leave request feature, the client has also requested there to be a limit to the amount of leave request quota for a given user. More specifically, each employee will gain 12 PTO quota per year and their total PTO cannot exceed 24. Only employees that have worked for over 6 months can request PTOs and a maximum of 3 PTO can be used in a given month. As for permissions, every employee is allotted a total of 12 permission days a year. The client also requested a tolerance work time feature. Each employee is given a total of 7 hours of tolerance work time every month. This tolerance work time will be deducted every time an employee goes home early or is late for work. In cases where an employee is late, they can compensate for the tolerance work time for a maximum of 30 minutes for that day. More information regarding this interview with the client can be found in the appendix section.

3.2 Related Works

The works discussed in this section will be based on the author's personal scope for this project, that is regarding face recognition models as well as its implementations. The face recognition model is necessary to facilitate the face detection feature for users to tap in and the implementation of said model is important to ensure said model is viable and functioning in an android application.

3.2.1 FaceNet

FaceNet is a deep learning model that was developed by Google engineers in 2015 [45]. It can extract face characteristics from images and transform them into high-dimensional vector representations, also referred to as embeddings. A big collection of facial pictures was used to train the model's convolutional neural network (CNN) design and create a mapping between the input

image and its associated embedding. When used for different tasks like facial identification, verification, clustering, and categorization, the resulting embeddings can be used. FaceNet is more resilient than conventional face recognition systems due in part to its capacity to manage variations in posture, lighting, and facial expression [45]. The model accomplishes this by employing a triplet loss function during training, which promotes the face embeddings of the same individual to be farther apart from each other in the embedding space and closer to one another [45]. By doing this, it is made sure that the embeddings both record and are discriminative enough to differentiate between individual facial characteristics.

FaceNet is extensively used in many different uses, including security systems, social media websites, and mobile devices. It is a well-liked option for facial identification jobs due to its high accuracy and robustness.

3.2.2 VGG-Face

Visual Geometry Group Face(VGG-Face) is a VGG16 based neural network developed by researchers in the University of Oxford that is fine tuned with facial images. VGG neural networks are one of the most popular deep neural network models for image recognition tasks [46]. The number 16 in VGG16 represents the amount of deep neural network layers in the model. It is a relatively large deep neural network model consisting of over 130 million parameters [47]. The image classification nature of these models makes it suitable to be used for facial recognition purposes.

VGG-Face takes an input image of size 224x224 pixels. After performing pixel normalization, the model outputs a numerical facial representation of the given image, also known as the face embeddings of said image. The VGG-Face model was capable of achieving an accuracy of 98.78% on the popular LFW dataset [48].

3.2.3 MobileFaceNet

MobileFaceNet is a set of highly efficient convolutional neural network models tailored specifically for mobile and embedded device use. It achieves this by making use of bottleneck layers as a building block of its architecture as well as a fast downsampling strategy [49]. Due to its mobile nature, MobileFaceNet only contains a little under a million parameters in total and is capable of achieving high accuracy with a relatively small size [49].

One of the key advantages of MobileFaceNet is its efficiency and speed, which makes it ideal for deployment on resource-constrained devices such as smartphones and embedded systems. The model can run in real-time on mobile devices with minimal computational resources, making it a practical solution for face recognition applications that require fast and accurate performance.

All related works researched can be summarized with the following table:

Table 3.1: Summary of related works

	FaceNet	VGGFace	MobileFaceNet
Created year	2015	2017	2018
Input dimensions	160x160x3	224x224x3	112x112x3
Output dimensions	1x128	1x2048	1x192
LFW accuracy	99.63%	98.78%	99.55%

3.3 Proposed Solution

In order to choose a face detection and face recognition model that will be best suited for a low computing environment, the author plans to focus on 3 evaluation metrics with varying priorities. Going in order of priority, the first and most important metric is model size. Due to the edge computing nature of the application, the chosen face recognition model will have to be bundled with the application itself. This is because the face recognition process will be performed on the user's end device and not on a cloud server. The second metric the author chooses is the model's F1 score. The F1 score will accurately determine how good the model is at doing its job, that is detecting whether a given face matches the user or not. The last metric chosen for evaluation is the model's inference time. The reason inference time was chosen as a metric is for the same reason model size is chosen; that is due to the edge computing approach of the application. Since the model will be running on a low computing environment, it is important to ensure the face recognition process can be done with low latency which will improve the user's experience as a whole.

To choose the most suitable model for an edge computing based android application, the author proposes the following weights for each evaluation metric:

Table 3.2: Metric weights

Metric Name	Weight
Model F1 score	30%
Model Size	50%
Inference Time	20%

Using the selected metrics, the author proposes the following formula in order to calculate the score of each model:

$$S = (F1\,score * W_{F1\,score}) + ((1 - \frac{Size}{100}) * W_{size}) + ((1 - \frac{Inference\ Time}{100}) * W_{inf\ time})$$

The reason the author chooses to propose this formula instead of simply dividing the value of each metric by the total number of metrics, in this example being 3, is to accommodate the aforementioned varying weights the author has assigned to each metric. In addition, this custom proposed formula provides a method to normalize the model's size and inference time to a range of 0 to 1.

3.3.1 Application Database

In order to facilitate all the functionality of the application along with the additional criterias from the client, a database will be needed. All of the client specific information such as the company's wifi SSID, maximum amount of PTOs, company tolerance work time and others as described in section 3.1 will need to be stored in a table in the database to ensure the client will have the freedom to alter these values at a later date. The database proposed for this project will be the Google Firestore database which is a document oriented NoSQL database. The reason Firestore is chosen for this project is due to its fast and readily available implementation with Android. Using firestore, no additional middleware is needed as the database can be seamlessly connected with the Android project with the help of the online Firestore console. This makes it possible to quickly develop and test out features during development.

CHAPTER 4

SOLUTION DESIGN

This chapter explains how the author plans to achieve the set individual aims. It explains the methods and steps to achieve the author's personal scope and aims for the project.

4.1 Solution Workflow

In order for the face recognition to work, the face embeddings of the user needs to be stored somewhere as a reference embedding to compare with unknown faces. This means that users will need to register their faces first through the application. After registering, the face embedding generated will be stored in the application's internal directory. This prevents the user from accessing the file through the use of a file explorer application. The embeddings (not the image) will also be sent to the database. This is needed so that users will not have to reregister their face again every time the app gets deleted or in the case the user changes their phone device.

After successfully registering the user's face, users will need to pass the facial recognition in order to tap in. This is done with the use of a live camera feed in the application. A face detecting algorithm will need to be run through each frame of the live camera feed in order to extract faces from the video. Once a face is detected, a liveness detection is performed on the face to ensure the face's authenticity and to prevent spoofing of the facial recognition system through the use of an image of the user. If the face passes the liveness detection, the face gets preprocessed and only then will the facial recognition model run on the preprocessed face. The full workflow is illustrated on the figure below:

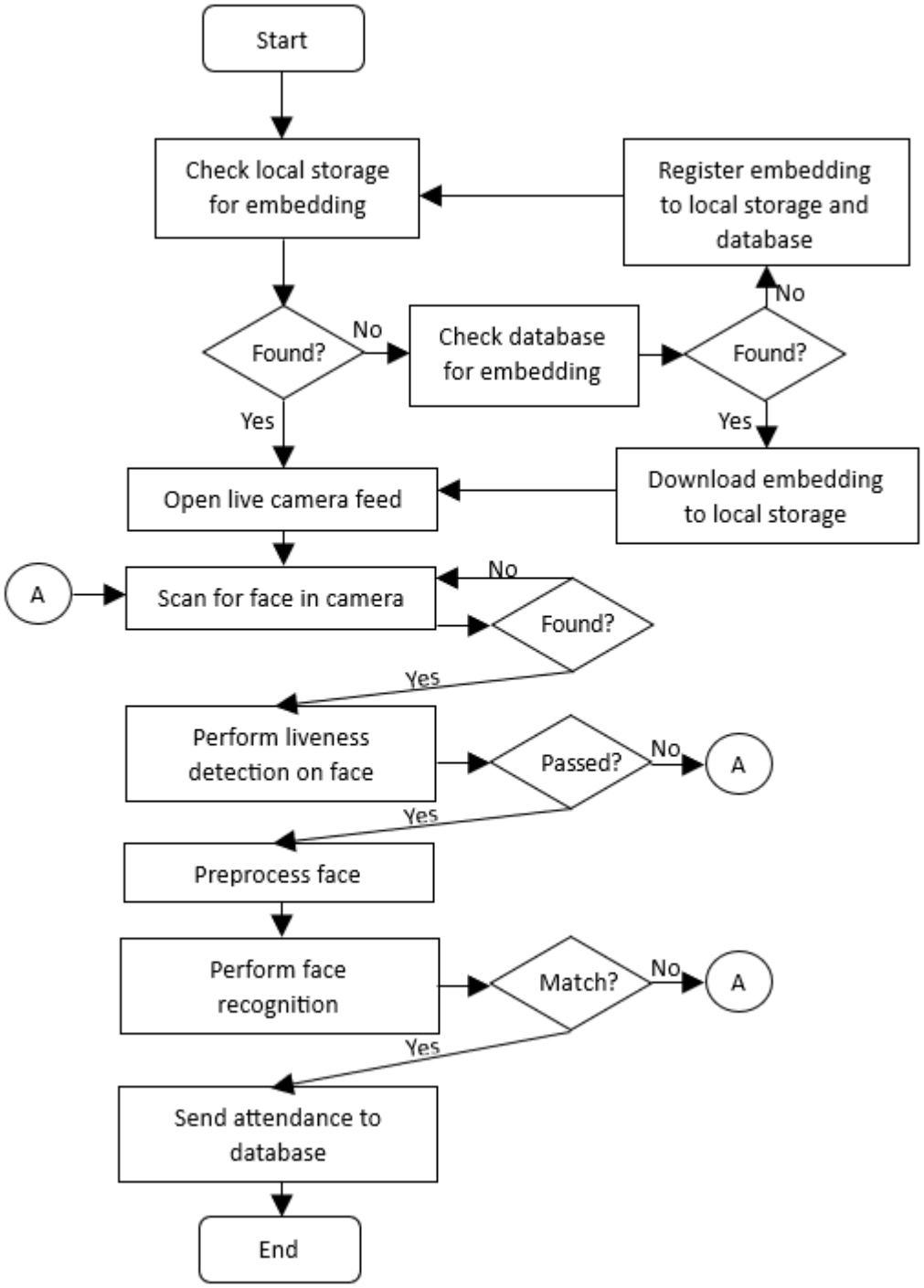

Figure 4.1 : Solution workflow

4.2 Face Detection and Face Recognition Evaluation

In order to determine which face detection and face recognition models will be most suitable for the project, a comparison and evaluation of the performance of each model will have to be conducted. For facial detection, two different algorithms will be tested for this project, Haar Cascade and Google MLKit; As for face recognition, three different models will be tested, VGGFace, MobileFaceNet, and FaceNet.

4.2.1 Dataset Gathering

To ensure a fair evaluation for each face detection and face recognition combination, all testing will be done with the same dataset. Because the goal of the face recognition algorithm is to determine whether a specified face belongs to the given user or not, the dataset will consist of two labels. The first label, unknown faces, represents a no match. It refers to images of random people that are not the user. The second label, known faces, represents a match. It refers to images of the user himself/herself. The dataset for this project consists of a balanced 116 known and 115 unknown faces each for a total of 231 images. The reason why one additional known image is needed will be explained in the paragraph below. All images in the dataset are gathered manually. For known faces, group member Bently Edyson volunteered to take several pictures of himself throughout the day over the course of a week. As for unknown faces, images have been gathered with the help from varying friends and family of both genders with their consent.

Because the face recognition model that will be tested are all pretrained models, no model training needs to be done. All three models will output embeddings which will be used to compare whether a given input face matches the user or not through the use of cosine similarity whereby a cosine similarity score of 1 indicates that two images are identical. This means that one image labeled known faces will need to be used to simulate the user registering into the application for the first time. This image will then be used to act as a benchmark to determine whether a given face belongs to the user or not. This is the reason why one additional known face is required in the dataset in

order to maintain a 1:1 balanced dataset of 115 images each for known and unknown faces. The remaining 230 images will then be used to test and evaluate the performance of each face detection and recognition model.

4.2.2 Face Detection Evaluation

Face detection can be considered as a data preprocessing step for the face recognition model. This is because the task of the face detection algorithm is to locate the presence of a face from a given image and crop it. Thus, the result of this face detection algorithm will greatly impact the performance of the face recognition model as it acts as the raw input data in which pixel normalization will be performed on for the face recognition model to accept as input. For this reason, the author will be evaluating the performance of each face detection algorithm by comparing the F1 score for each face recognition model when using said face detection algorithm. For the Haar Cascade algorithm, evaluation can be done directly through Python with the help of the cv2 module. As for the Google MLKit, because it is a mobile SDK developed for mobile use, the author will have to develop a custom simple android application to integrate said algorithm. This custom application will be used to output the result of the face detection and cropping of the Google MLKit algorithm on the given dataset. The end result of all this is two separate preprocessed dataset. The first being the original 231 images after being cropped with the Haar Cascade algorithm, the latter being the original 231 images after being cropped with the Google MLKit algorithm with the help of a custom android application.

4.2.3 Face Recognition Evaluation

For this evaluation, the author will be taking into account the model's F1 score, inference time, as well as the model's size in order to generate a final model score using the formula proposed in section 3.3. As previously mentioned, one image from the dataset labeled as known faces will be used as a benchmark image to classify whether a given image matches the user or not. The same

benchmark image will be used across all testing performed to ensure a fair comparison between model performance. Once the testing results for all the different face recognition and face detection models are complete, the model score for each combination of face detection and face recognition can be calculated using the formula on section 3.3. This model score can then be used to determine the most suitable combination of face detection and face recognition model to be used for the application.

4.2.4 Evaluation Workflow

The entire evaluation workflow for both face detection and face recognition is illustrated on the figure below.

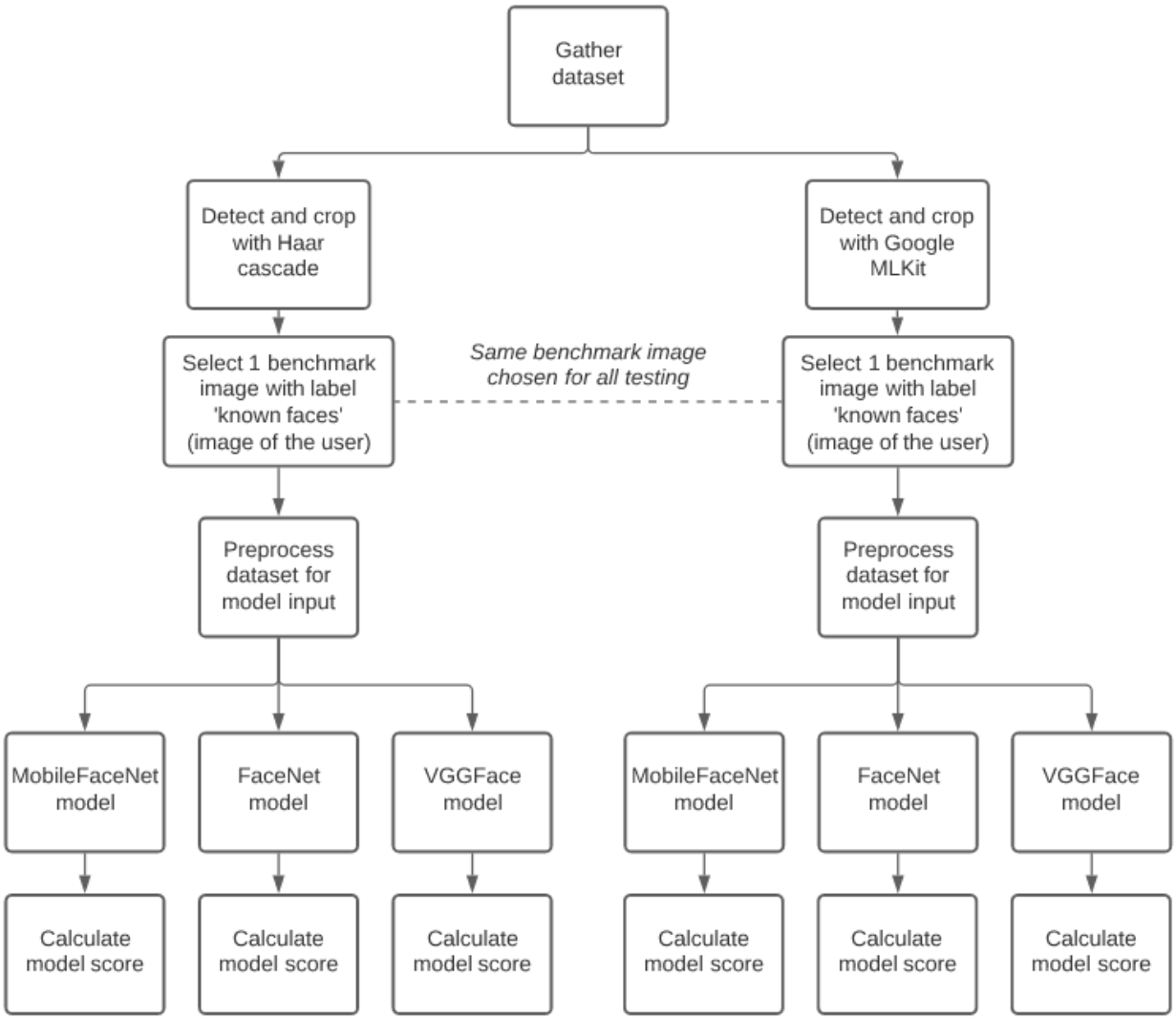

Figure 4.2 : Evaluation workflow

Firstly, gather a dataset of images with label ‘known’ and ‘unknown’ faces. Known faces refer to faces of the actual user while unknown faces refer to faces of random people that do not belong to the user. Next, feed the dataset through both the Haar Cascade face detection method and Google MLKit in order to detect and crop the faces of each image in the dataset. Isolate one cropped image labeled ‘known face’ to act as a benchmark image for testing. This benchmark image represents the user’s initial registration into the application. Next, preprocess the dataset to prepare it for the face recognition models. After all three facial recognition models have been tested, the model score

for each model will be calculated with the formula proposed on section 3.3 in order to evaluate which model will be the best suited for the application.

4.3 Face Recognition Pipeline

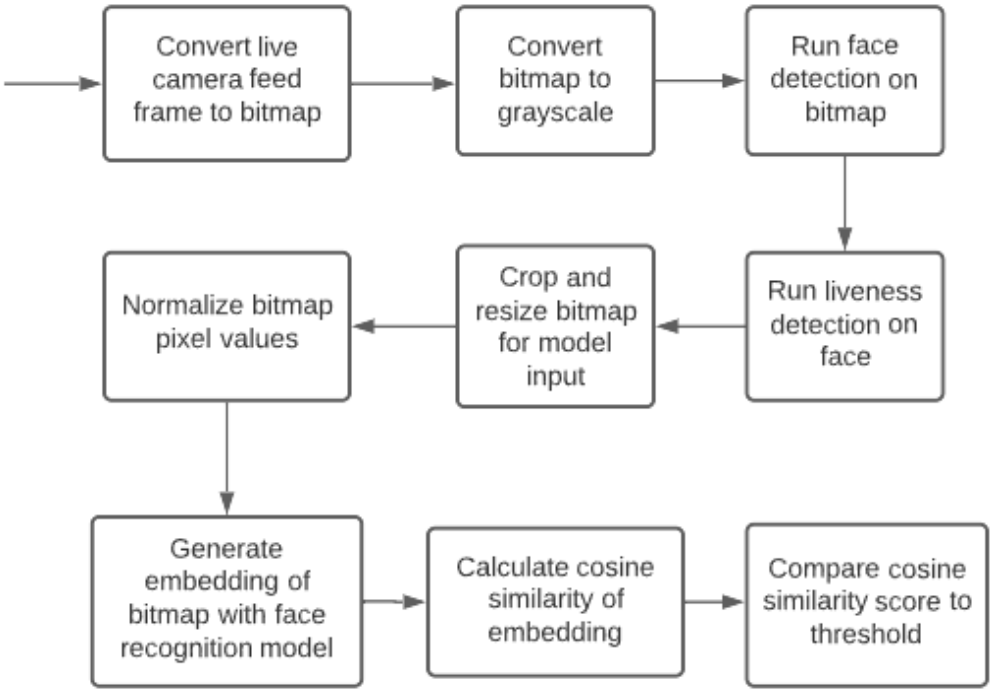

Figure 4.3 : Face recognition pipeline

The face recognition pipeline starts with receiving an input frame from the live camera feed. This frame is then converted into a bitmap to obtain the pixel values which is what the model expects as its input. The bitmap is then converted into grayscale. The face detection algorithm will be run on the grayscale bitmap to detect faces. This is followed by running liveness detection on the detected faces. If the face passes the liveness detection, the bitmap will be cropped and resized to prepare it for model input. Before being fed into the model, the bitmap's values will be normalized

to improve model accuracy. Facial embeddings are then generated with the face recognition model using the normalized bitmap data. Lastly, cosine similarity score will be calculated between the facial embedding generated and the known facial embedding of the user stored locally. This cosine similarity score will then be compared to the threshold to determine whether the user passes the face recognition or not.

4.4 Database Design

The database design for the application is illustrated below. The design was made taking into account all the features the application will have.

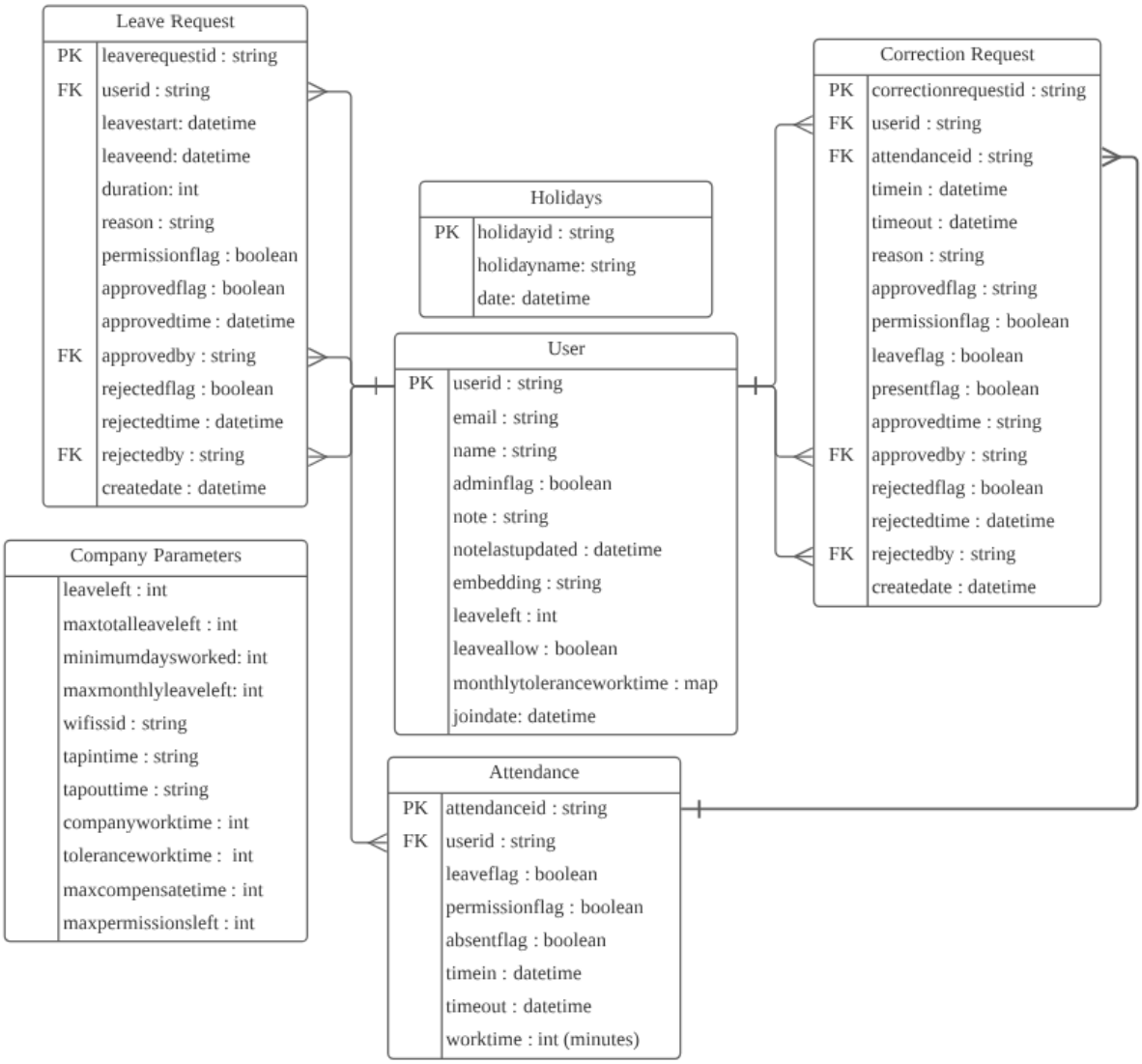

Figure 4.4 : Database ERD

The user collection is used to store every user in the system. It also contains user specific data needed to meet the client's criteria such as leaveleft, leaveallow, and monthlytoleranceworktime. The complete explanation for each field's purpose can be found on table 4.1 below.

Table 4.1 : User collection explanation

Column name	Data type	Explanation

userid (PK)	string	Unique id for each user
email	string	User company email for logging in
name	string	Name of the user
adminflag	boolean	Indicates whether a specified user is an administrator or not
note	string	Personal note for each user
notelastupdated	datetime	Denotes the last time each user's personal note is updated
embedding	string	Facial embedding data of user
leaveleft	int	Indicates the amount of leave request days left a user has
leaveallow	boolean	Denotes whether the user is allowed to create leave request or not
monthlytoleranceworktime	map	Keeps track of the user's remaining tolerance work time throughout the year
joindate	datetime	Date of user registration

The attendance collection is used to store every daily attendance made by the users in the system. It also contains flags to denote the type of each attendance in the form of leaveflag, permissionflag, and absentflag. The complete explanation for each field's purpose can be found on table 4.2 below.

Table 4.2 : Attendance collection explanation

Column name	Data type	Explanation
attendanceid (PK)	string	Unique id for each attendance
userid (FK)	string	Denotes the user performing the attendance
leaveflag	boolean	Indicates whether the current attendance was

		a leave request or not
permissionflag	boolean	Indicates whether the current attendance was a permission or not
absentflag	boolean	Indicates whether the current attendance was an absent or not
timein	datetime	Time of user tapping in
timeout	datetime	Time of user tapping out
worktime	int	The total work time in minutes of the current attendance

The leave request collection is used to store every leave request created by the user in the system. It contains permissionflag to denote whether the request is a PTO or a permission as well as approvedflag and rejectedflag to denote the current status of the request (pending, approved, rejected). The complete explanation for each field's purpose can be found on table 4.3 below.

Table 4.3 : Leave request collection explanation

Column name	Data type	Explanation
leaverequestid (PK)	string	Unique id for each leave request
userid (FK)	string	Denotes the user requesting the leave request
leavestart	datetime	Proposed starting date of leave request
leaveend	datetime	Proposed ending date of leave request
duration	int	Duration of leave request in days
reason	string	The reason the user requested the leave request
permissionflag	boolean	Indicates whether the leave request was a permission or not

approvedflag	boolean	Indicates whether a leave request has been approved or not
approvedtime	datetime	Denotes the time the administrator rejected the request
approvedby (FK)	string	Denotes the administrator who approved the request
rejectedflag	boolean	Indicates whether a leave request has been rejected or not
rejectedtime	datetime	Denotes the time the administrator rejected the request
rejectedby (FK)	string	Denotes the administrator who rejected the request
createdate	datetime	Time of leave request creation

The correction request collection is used to store every correction request created by the user in the system. It contains the attendanceid to keep track of which attendance the specified correction request is for as well as approvedflag and rejectedflag to denote the current status of the request (pending, approved, rejected). The complete explanation for each field's purpose can be found on table 4.4 below.

Table 4.4 : Correction request collection explanation

Column name	Data type	Explanation
correctionrequestid (PK)	string	Unique id for each correction request
userid (FK)	string	Denotes the user requesting the correction request
attendanceid (FK)	string	Denotes the attendance for the proposed correction request

timein	datetime	Proposed new tap in time for the specified attendance
timeout	datetime	Proposed new tap out time for the specified attendance
reason	string	The reason the user requested the correction request
leaveflag	boolean	Used when correction request issued to change absent attendance to leave
permissionflag	boolean	Used when correction request issued to change absent attendance to permission
presentflag	boolean	Used when correction request issued to change absent attendance to present
approvedflag	boolean	Indicates whether a correction request has been approved or not
approvedtime	datetime	Denotes the time the administrator rejected the request
approvedby (FK)	string	Denotes the administrator who approved the request
rejectedflag	boolean	Indicates whether a correction request has been rejected or not
rejectedtime	datetime	Denotes the time the administrator rejected the request
rejectedby (FK)	string	Denotes the administrator who rejected the request
createdate	datetime	Time of correction request creation

The holidays collection is used to store every holiday created by the admin in the system. By default, several common holidays such as Christmas eve and the first day of the year will be added, however, administrators can add additional holidays into the system. The complete explanation for each field's purpose can be found on table 4.5 below.

Table 4.5 : Holidays collection explanation

Column name	Data type	Explanation
holidayid (PK)	string	Unique id for each holiday entry
holidayname	string	Name of holiday
date	datetime	Date of holiday

The company parameter collection is used to store all the information needed to meet the client's criterias based on section 3.1. This collection will only contain a single document which will house all the fields needed. The complete explanation for each field's purpose can be found on table 4.6 below.

Table 4.6 : Company parameters collection explanation

Column name	Data type	Explanation
leaveleft	int	Denotes the amount of PTOs distributed to eligible users at the end of the year
maxtotalleaveleft	int	Denotes the maximum amount of PTO each user can have
minimumdaysworked	int	Denotes the minimum amount of days a user has to be eligible for PTOs
maxmonthlyleaveleft	int	Denotes the maximum amount of PTOs a user can use each month

wifissid	string	Wifi ssid of the office to limit attendance to onsite only
tapintime	string	Denotes the company's starting work hour
tapouttime	string	Denotes the end of the company's work hour
companyworktime	int	Duration in minutes of company work hour
toleranceworktime	int	Total duration in minutes of tolerance work time each user is given every month
maxcompensatetime	int	Maximum duration in minutes each user can compensate for daily if they are late
maxpermissionsleft	int	Maximum amount of permissions each user is given every year

4.5 Use Case Diagram

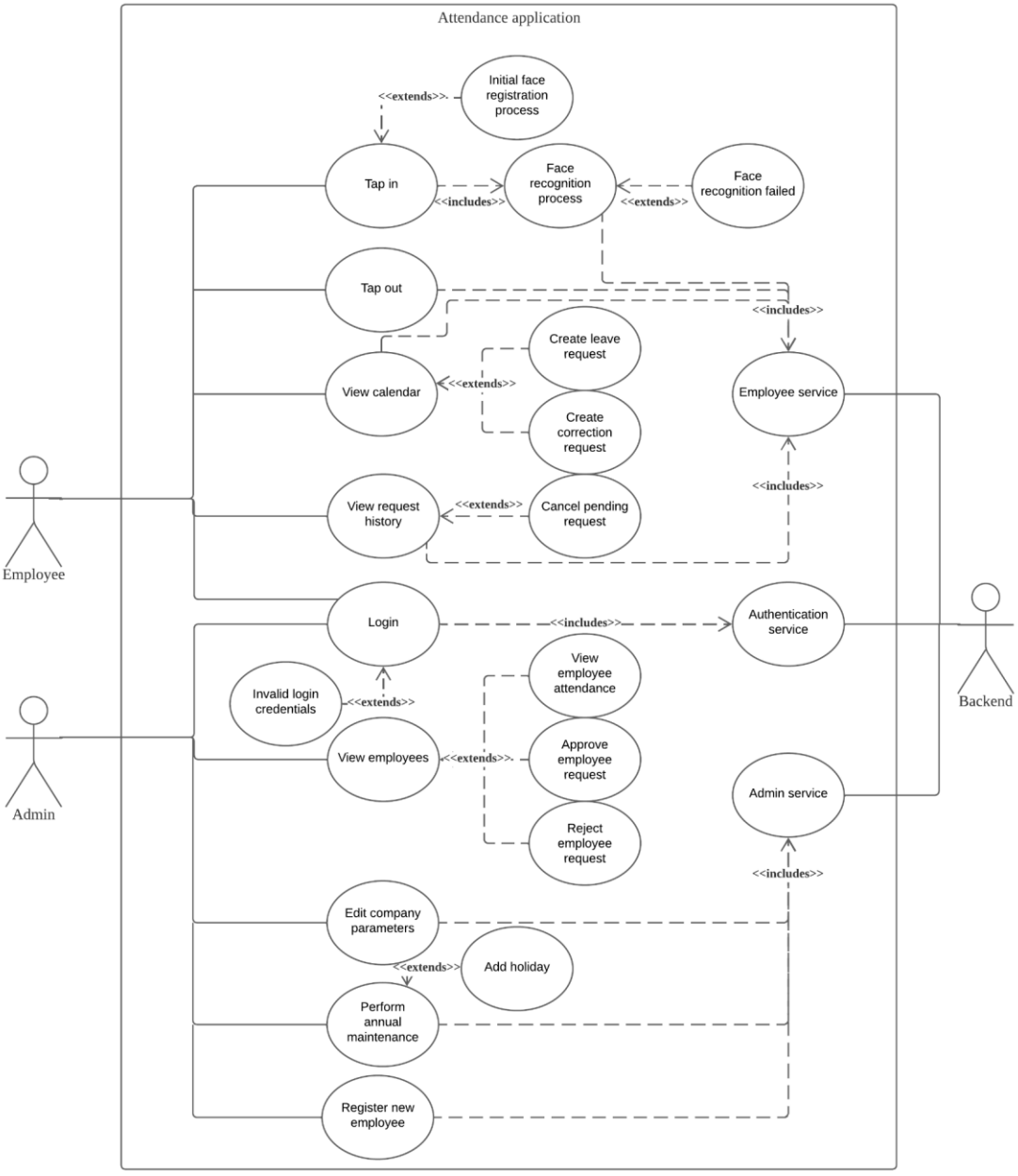

Figure 4.5 : Use case diagram

Figure 4.5 above illustrates the complete use case diagram for the application. The use case diagram for the application contains 3 actors, employees, administrators, and the backend. The first actor, the employee, has access to 5 processes. These processes are:

- Tap in
- Tap out
- View calendar
- View request history
- Login

The administrator actor also contains 5 different processes. These processes are:

- Login
- View employees
- Edit company parameters
- Perform annual maintenance
- Register new employee

The annual maintenance process available to administrators has several purposes. As the name implies, it is meant to be run once at the beginning of the year. This maintenance will be done through a button accessible by administrators in the application. The purpose of this maintenance is to:

- Distribute PTO quota to every eligible employees (employees who have worked more than 6 months)
- Setting the dates of national holidays for the year
- Resetting monthly tolerance work time for every user (7 hours)

CHAPTER 5

IMPLEMENTATION

This chapter goes into detail about the results of the author's testing; it also discusses further regarding how the implementation of the workflow discussed in chapter 4 is done in the backend

5.1 Cosine Similarity Tuning

As previously mentioned, the metric used to determine whether a given face matches the user or not is the cosine similarity. The cosine similarity score is a value in the range of -1 to 1 which represents how similar a given face is to the user. A score of 1 indicates that the two images are the same, 0 indicates no correlation between the two images and -1 indicates the two images are diametrically opposed. Hence, it is important to find the best cosine similarity threshold for each face recognition model to attain the highest F1 score. To find the best cosine similarity for a given model, the author ran the evaluation workflow on section 4.2.4 with varying cosine similarity values starting from 0 up to 1 in increments of 0.01. Negative cosine similarity values were not tested as the goal was to compare the similarity between two given faces. The result of each model's tuning can be found on the table and figures below.

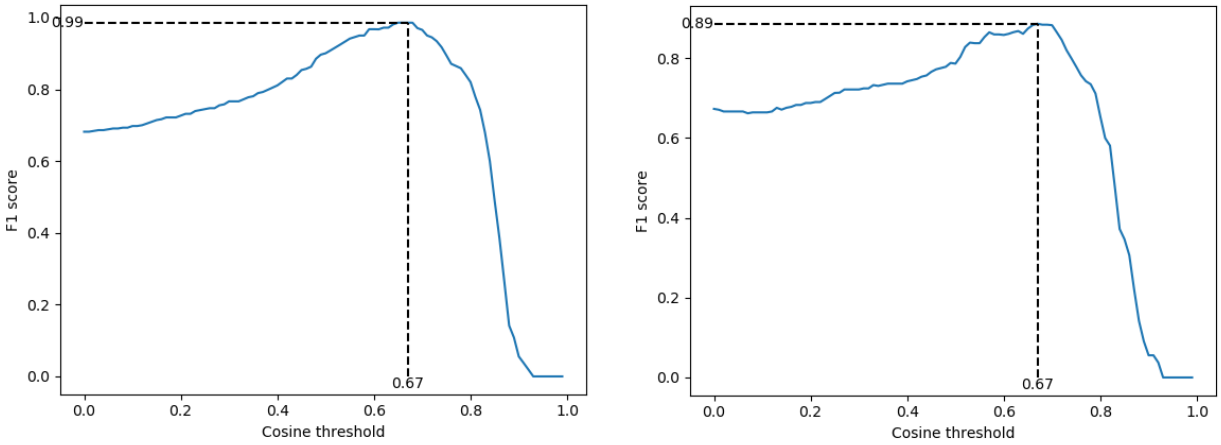

Figure 5.1 : FaceNet cosine similarity thresholds (MLKit left, Haar cascade right)

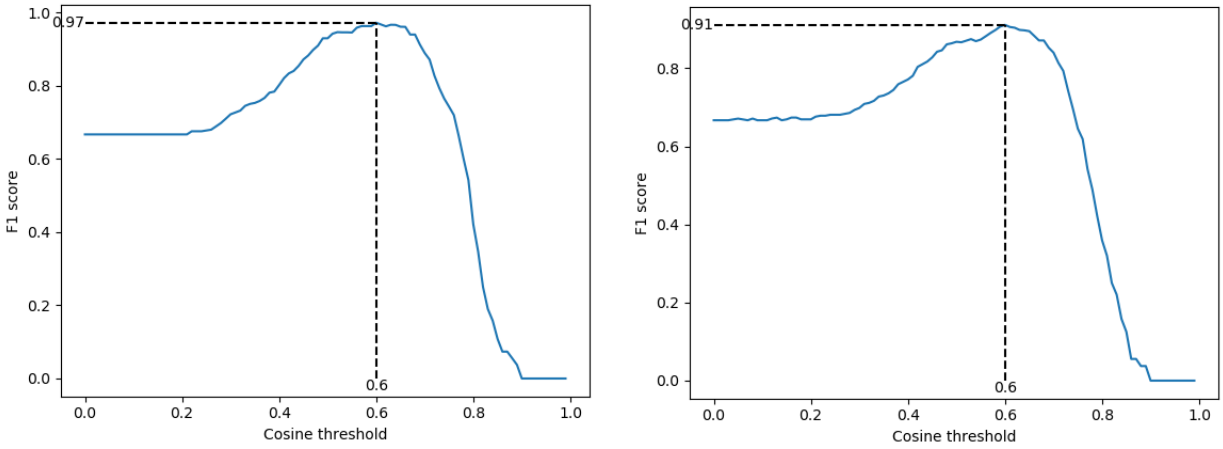

Figure 5.2 : VGGFace cosine similarity thresholds (MLKit left, Haar cascade right)

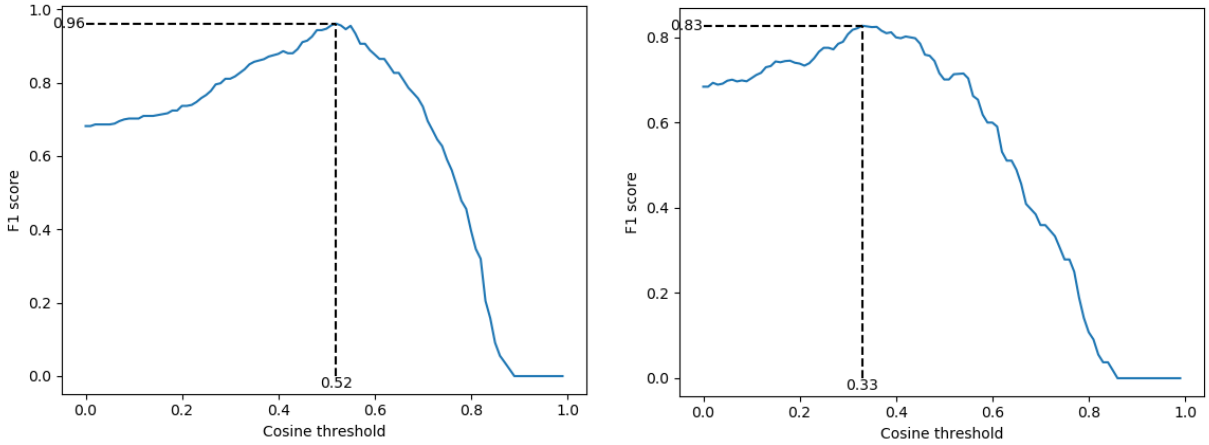

Figure 5.3 : MobileFaceNet cosine similarity thresholds (MLKit left, Haar cascade right)

Table 5.1 : Cosine similarity thresholds

Model name	Best cosine threshold (MLKit)	Best cosine threshold (Haar cascade)
FaceNet	0.67	0.67
VGGFace	0.60	0.60
MobileFaceNet	0.52	0.33

From this testing, we can see that the cosine threshold value for both MLKit and Haar cascade seems to be identical for both FaceNet and VGGFace. This however does not apply for the MobileFaceNet model. The best cosine threshold value of the MobileFaceNet model when paired with Google MLKit is 0.52 while the best cosine threshold value of the MobileFaceNet model when paired with Haar cascade is 0.33 resulting in a 52% difference.

5.2 Pixel Normalization

In order to improve the performance of the model, the pixel values of each facial image needs to be normalized. This counts as a data preprocessing step to prepare the input data for the face recognition model. Firstly, each input image needs to be grayscaled to reduce image noise. After the image has been grayscaled, the next step is to normalize the values. The method to normalize pixel values differs for different facial recognition models. For FaceNet and MobileFaceNet, pixel normalization is performed by subtracting 127.5 and dividing by 128 to each RGB channel in a pixel [49]. The reason the value is subtracted by 127.5 is because the maximum value in an RGB channel is 255. This means that by subtracting it by 127.5, which is exactly half of its value, will in turn result in a value between -127.5 and 127.5. This is followed by dividing by 128 in order to normalize each pixel value to a range of -1 to 1. As for VGGFace, pixel normalization is done by subtracting the value of each RGB channel in a pixel by the mean value of each RGB channel in the dataset [50].

5.3 Model Evaluation Results

The result of the testing and evaluation performed on section 4.2 can be found on the table and figures below.

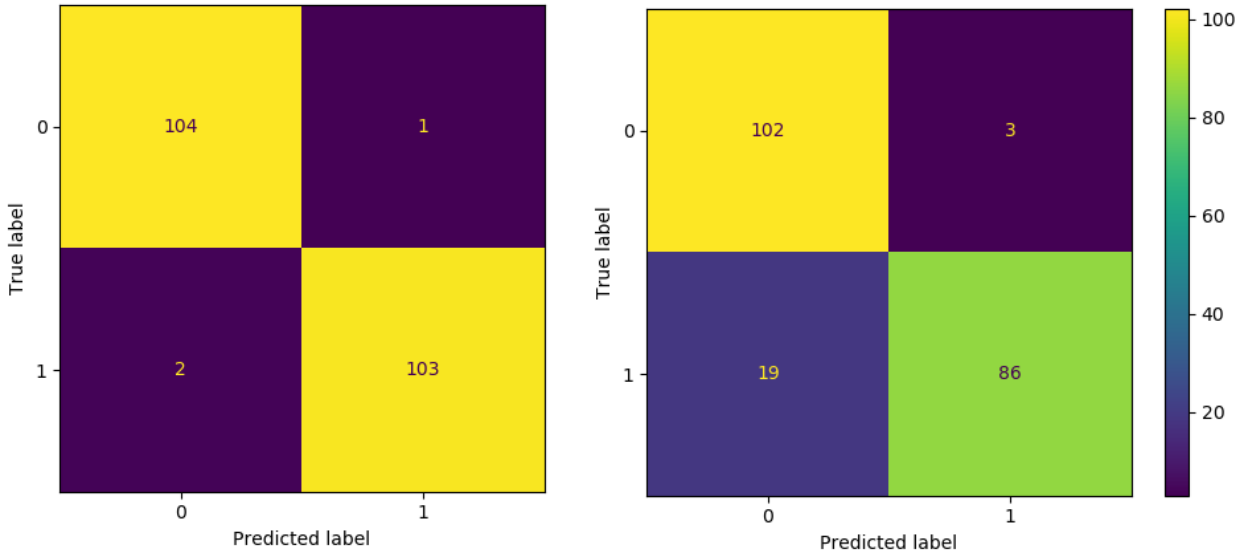

Figure 5.4 : FaceNet confusion matrix (MLKit left, Haar cascade right)

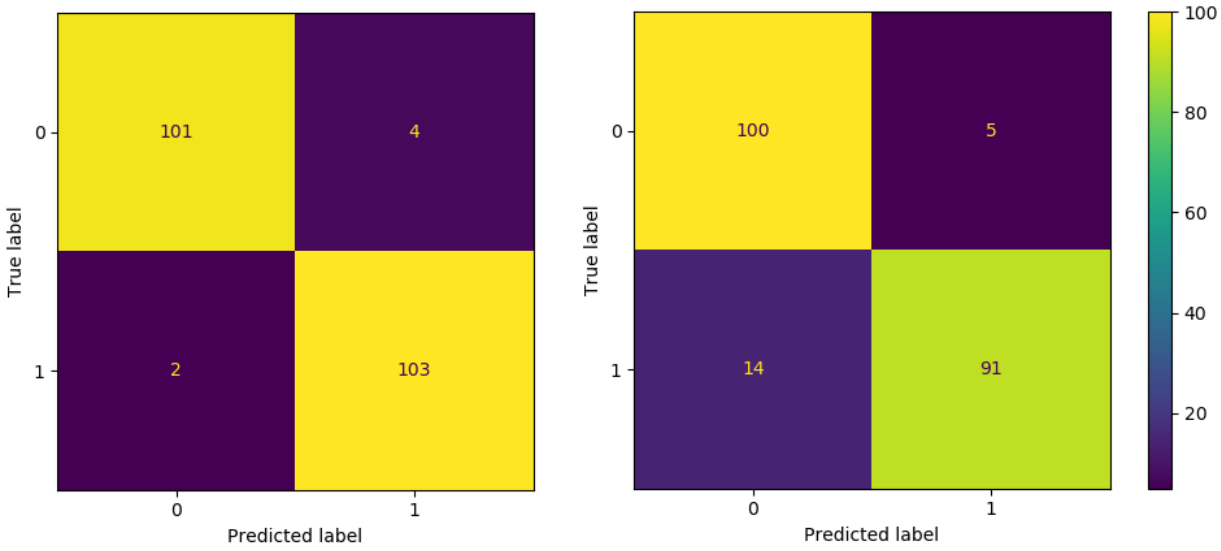

Figure 5.5 : VGGFace confusion matrix (MLKit left, Haar cascade right)

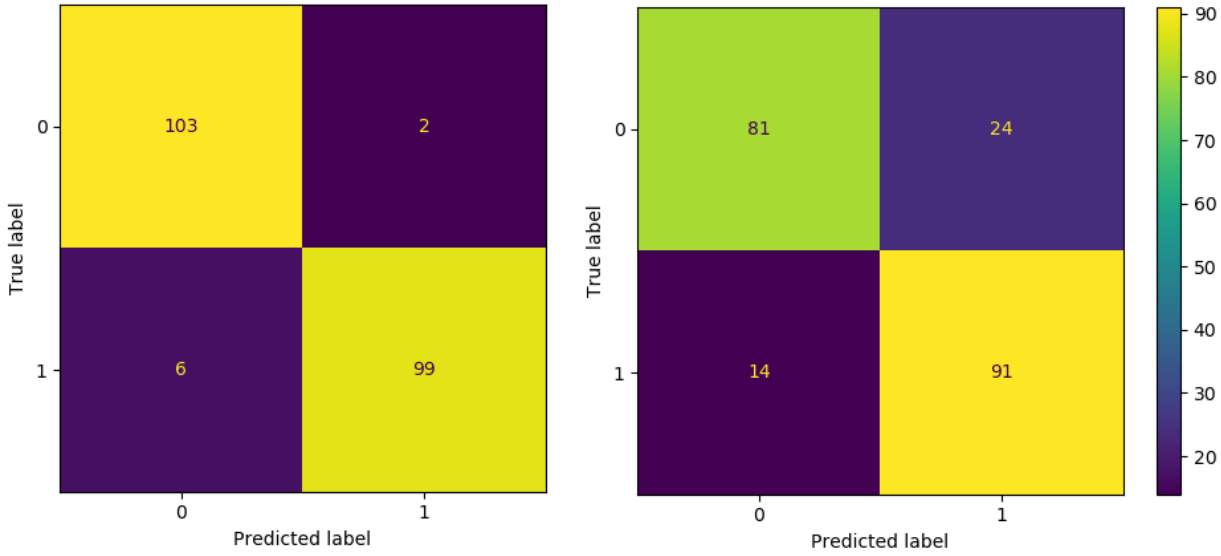

Figure 5.6 : MobileFaceNet confusion matrix (MLKit left, Haar cascade right)

Table 5.2 : Evaluation results

	FaceNet	VGGFace	MobileFaceNet
F1 score (MLKit)	0.99	0.97	0.96
F1 score (Haar cascade)	0.89	0.91	0.83
Size (MB)	23.1	23.5	5.1
Inference time (ms)	68	89	67
Model score	0.7455	0.6955	0.8285

The cosine similarity threshold for each model is individually tuned with the aforementioned method to obtain the highest F1 score for a given model. From the results, we can see that the MLKit method for face detection is better compared to Haar cascade as it results in better model F1 score across the board.

First, the FaceNet model was able to achieve an F1 score of 0.99 when paired with Google MLKit and an F1 score of 0.89 when paired with Haar cascade. There is an 11.2% increase in F1 score for the FaceNet model in favor of Google MLKit. Next, the VGGFace model achieved an F1 score of 0.97 when paired with Google MLKit and an F1 score of 0.91 when paired with Haar cascade. The increase in F1 score is 6.6% in favor of Google MLKit. Lastly, the MobileFaceNet model was able to achieve an F1 score of 0.96 when paired with Google MLKit and an F1 score of 0.83 when paired with Haar cascade. This is the largest increase yet at 15.6% in favor of Google MLKit. Overall, Google MLKit seems to edge out Haar cascade by 11.2% in this testing in terms of model F1 score.

These testing results help the author in determining the appropriate weights for each metric on the formula proposed in section 3.3. The biggest difference between models seems to be the model size. This is the reason why model size has a significantly larger weight as opposed to the other 2 metrics. As for inference time, all 3 models had an inference time significantly less than the targeted maximum of 1 second (1000 ms) from section 1.3.1. This is the reason why inference time has the least significant weight on the formula proposed on section 3.3 as the latency for all 3 models is so small that it can be considered as negligible for the application's use case as it will not be used to process hundreds of faces and instead only for one single image whenever a user taps in for attendance.

Model score is tabulated with the formula proposed on section 3.3 using each model's MLKit F1 score due to its superior performance. By looking at the model score, this evaluation indicates that the best combination of face detection and face recognition for a low computing power environment is the Google MLKit paired with the MobileFaceNet model.

5.4 Liveness Detection

The liveness detection feature in the application is present in the application to prevent users from being able to spoof the facial recognition attendance system with a static image of themselves. This is done by analyzing the user's eye movements. The implementation involved extracting the probabilities of both left and right eye opening from frames in the live camera feed and storing them in an array. This is done with the help of the Google MLKit face detector set to classification mode. The standard deviation of the array was then calculated, and the liveness detection result was based on this standard deviation value. A low standard deviation value indicates the use of a static image as the probabilities for left and right eye opening will remain stationary while a high standard deviation indicates a live user.

5.5 Database Implementation

The database chosen for this project's backend is Google's Firestore. Due to its document-collection structure, every entity on chapter 4.5 represents a collection in the database. Every entry, for example a user in the users table, represents a document. Firestore read and write operations are asynchronous by nature. This ensures that traffic will not be bottlenecked in the case of large amounts of data being queried.

All database related functions in this project are stored in a utility class to be called by the frontend. Data models are also created according to the design on section 4.5 in order to maintain data structure and integrity. Table 5.3 is the list of all the functions within the database utility class.

Table 5.3 : Database functions

Name	Parameters	Output	Explanation
getUser	userid: String	User	Retrieves the specified user document.
getAllUser	-	List<User>	Retrieves all documents in the user collection.
createUser	user: User	-	Creates a new user document.
checkUserIsAdmin	userid: String	Boolean	Checks whether the specified user is an admin or not. Returns true if the user is an admin.

getAttendance	userid: String? dateStart: Date dateEnd: Date	List<Attendance>	Retrieves all attendance documents from dateStart to dateEnd for the specified user. Retrieves for all user if userid parameter is left null.
getLeaveRequest	userid: String?	List<LeaveRequest>	Retrieves all leave request documents for the specified user. Retrieves for all user if userid parameter is left null.
getCorrectionRequest	userid: String?	List<CorrectionRequest>	Retrieves all correction request documents for the specified user. Retrieves for all user if userid parameter is left null.
getTotalLeaveThisMonth	userid: String	Int	Retrieves the total amount of PTOs for the specified user in the current month.
getTotalPermissionThisYear	userid: String	Int	Retrieves the total amount of permissions for the specified user in the current month.
getCompanyParams	-	CompanyParameter	Retrieves company

			parameters necessary to facilitate other features.
getHolidays	duration: Int?	List<Holidays>	Retrieves all holiday documents for the past days based on duration. Retrieves all holiday documents if the duration parameter is null.
getPendingRequestDates	userid: String	List<Date>	Retrieves all pending leave request and correction request dates for the specified user.
createAttendance	user: User attendance: Attendance	-	Creates a new attendance document.
registerFace	userid: String embs: String	-	Updates embedding data for the specified user.
createLeaveRequest	leavereq: LeaveRequest	-	Creates a new leave request.
createCorrectionRequest	corr: CorrectionRequest	-	Creates a new correction request.
rejectLeaveRequest	leaverequestid: String userid: String	-	Rejects the specified leave request.
rejectCorrectio	correctionrequestid: String	-	Rejects the

nRequest	userid: String		specified correction request.
approveLeaveRequest	leavereq: LeaveRequest user: User	-	Approves the specified leave request.
approveCorrectionRequest	corr: CorrectionRequest user: User	-	Approves the specified correction request.
cancelLeaveRequest	leaverequestid: String	Boolean	Cancels the specified pending leave request. Returns true if cancellation is successful.
cancelCorrectionRequest	correctionrequestid: String	Boolean	Cancels the specified pending correction request. Returns true if cancellation is successful.
checkPendingRequestDuration	userid: String date: Date	Int,Int	Checks the duration, in days, of pending leave and correction request for the specified user on the specified month. Returns a pair of int the former for PTO and the latter for permission.
checkCorrectionRequestExist	attendanceid: String	Boolean	Checks whether a correction request already exists for a specified

			attendance. Returns true if correction request already exists.
checkValidLeaveRequestDate	userid: String date: Date duration: Int	Boolean	Checks whether the dates specified for a leave request overlaps with any other data. Returns true if it does not overlap.
checkValidCorrectionRequestDate	userid: String date: Date	Boolean	Checks whether the new date specified for a correction request overlaps with any other data. Returns true if it does not overlap.
tapOutAttendance	user: User attendance: Attendance	-	Updates specified user's data and attendance for today to mark user tapping out.
checkBackAttendance	user: User	Boolean	Checks whether specified user has any missing absents or missing tap out data for the past 14 days. Returns true if missing tap out data found.
addHolidayManual	holiday: Holiday	-	Adds a holiday into the system. Only accessible by admins.

deleteHoliday Manual	holidayid: String	-	Deletes a holiday from the system. Only accessible by admins.
checkYearlyM aintenanceDon e	-	Boolean	Checks whether admin yearly maintenance has already been done for the current year. Returns true if already done.
adminYearlyM aintenance	holidays: List<Holiday>	-	Yearly maintenance performed by admins. Needed to reset monthlytolerance worktime of employees, distribute leaveleft, and auto inserting several common holidays for the year.
updateCompan yParams	params: CompanyParameter	-	Updates company parameters. Only accessible by admins.

CHAPTER 6

DISCUSSION

This chapter discusses the key results during development and testing within this thesis. It also outlines several interesting findings throughout this thesis.

6.1 Discussion

The results show that different face recognition models will have different performance based on the facial detection methods used. In essence, all three models, FaceNet, MobileFaceNet, and VGGFace were able to perform well for facial recognition purposes as all three models had comparable F1 scores. It is just that when it comes to this thesis' specific use case, that is an edge computing approach, the MobileFaceNet model seems to edge out the competition due to the results shown in section 5.3.

For face detection, the MLKit method seems to outperform the Haar cascade method for all three face recognition models. When looking into the confusion matrix results of both in figure 5.4 to 5.6, the trend seems to be that all three face recognition methods result in a larger false negative amount. This means that in terms of the attendance application, the Haar cascade method is more likely to fail the user's face recognition step even though the face truly belongs to the registered user. An interesting finding from this result is that as seen in figure 5.6, the MobileFaceNet model suffered an increase in false negative amount coupled with a larger increase in false positive amount. This means that the MobileFaceNet model, as opposed to VGGFace and FaceNet, was both predicting more faces that belong to the user as no match as well as predicting more faces that do not belong to the user as a match. This explains why the MobileFaceNet model suffered the largest decrease in F1 score at 15.6% when switching from MLKit to Haar cascade as opposed to 11.2% and 6.6% for FaceNet and VGGFace respectively.

When talking about the face recognition models tested, from the results, it seems to be the case that the FaceNet model seems to be the best performing out of the other models tested due to it resulting in the highest F1 score, especially when paired with the MLKit face detection method. However, when talking about the model that is best suited for an edge computing use case, the MobileFaceNet model edges out due to a higher model score as seen in table 5.2. This is mainly due to the significantly smaller model size of the MobileFaceNet model as the FaceNet model is roughly 4.5x the size of the MobileFaceNet model. An interesting finding is that the VGGFace model seems to be the best performing model when using the Haar cascade method for face detection. However, much like the other two models, the VGGFace model still yielded a better F1 score when paired with the MLKit method. Another interesting finding from the results is that both FaceNet and VGGFace models seem to produce identical best cosine threshold regardless of the two facial detection methods tested. This however does not apply to the MobileFaceNet model as the MLKit method resulted in a higher best cosine similarity threshold compared to Haar cascade. As for inference time, in the context of the attendance application's use case, all three models performed excellently and were able to run with low latency. However, when talking about processing hundreds or even thousands of faces continuously, the MobileFaceNet model will edge out its competition due to its lowest inference time when compared to the other two models.

As for the database, with the help of the web based Firestore console, connection between the application to the database is done seamlessly. Once connected, simply add the necessary dependencies into the project and the database is ready to be used. Because no additional middleware is needed to facilitate the connection, functions stored in the database utility class can be tested directly through code within the application itself. The results can then be checked through the online firestore console. This makes it possible to rapidly develop and test database functions needed to facilitate the processes within the application. In addition, the firestore console provides analytics insights such as total read, write and deletes within the application.

CHAPTER 7

CONCLUSION AND RECOMMENDATION

This chapter acts as a conclusion chapter summarizing the main contents of this thesis. It also provides several recommendations that can be implemented for future works.

7.1 Conclusion

To conclude, the author's aim within this thesis is to provide a method to compare the performance of different face detection and face recognition methods for an edge computing use case. The author chose to use model F1 score, size and inference time as the metrics for comparison and the best performing model will be chosen for the attendance application. The MobileFaceNet model was the best performing model based on the results on table 5.2 and hence deemed to be the most suited model for an edge computing application. This is mainly due to its significantly smaller model size while still maintaining comparable F1 score to the other two models tested.

After integrating the MobileFaceNet model into the attendance application, the author also designed and implemented a backend database to facilitate the application's needs. Additional application features such as paid time off quotas for employees also needed to be taken into account for the database design. The database chosen for the project was Firestore. This is because firestore was able to be readily integrated into the application with the help of an online web console. This makes it possible for the author to rapidly develop and test the database functions needed for the application.

7.2 Recommendations

In the current application, each individual user's monthly tolerance work time as well as PTO quotas are distributed yearly with the help of an annual maintenance button accessible by administrators. The down side of this method is the inherent manual nature which can be considered as a hassle and relies on the administrators not forgetting to perform said maintenance. Perhaps with the use of cloud functions, this maintenance can be automated every year.

Another use case for cloud functions within the application is in the form of absent checking. In the application, absents are inserted into the database by querying the last 14 days of attendance of a specified user. This query happens once every time the user enters the application. If the query returns gaps in dates, the database will automatically insert an attendance with absentflag set to true. The downside of this implementation is that since the trigger to this query is whenever the user enters the application, it is possible that users who do not attend for work for over two weeks and do not open the application at all during those two weeks will have missing attendances. This is where cloud functions can be used to check for absent employees at the end of each day.

The last recommendation has to do with the methodology of the liveness detection feature. The liveness detection feature in the application is employed as a way to ensure users cannot spoof the face recognition with a static image of the user. This is done by analyzing the state of the eyes on each frame of the live camera feed during the tap in process and calculating the standard deviation. The downside to this implementation is that theoretically, it is still possible to spoof the liveness detection by using a video of the user. All the user had to do was record a video of themselves staring into the camera while blinking repeatedly to emulate liveness. Perhaps a classification model that takes video as input can be used to improve the current liveness detection implementation.

References

[1] P. Michael, "Technology statistics: How fast is Tech advancing? [growth charts] 2023," Media Peanut, 2 December 2023. [Online]. Available: https://mediapeanut.com/how-fast-is-technology-growing-statistics-facts/. [Accessed 16 March 2023].

[2] A. S. Gillis, "What is the Internet of things (IoT)?," TechTarget, 4 March 2022. [Online]. Available: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT. [Accessed 19 March 2023].

[3] Camelo HQ, "Problems with Manual Attendance Tracking: Why You Should Change," Camelo HQ, 1 April 2021. [Online]. Available: https://blog.camelohq.com/manual-attendance-vs-software/. [Accessed 5 April 2023].

[4] MedlinePlus, "Are fingerprints determined by genetics?," MedlinePlus, [Online]. Available: https://medlineplus.gov/genetics/understanding/traits/fingerprints/. [Accessed 30 6 2023].

[5] A. Seivani, "7 Systems that are Effective to Track Employee Attendance," clockster, 27 January 2022. [Online]. Available: https://clockster.com/7-systems-that-are-effective-to-track-employee-attendance/. [Accessed 30 June 2023].

[6] K. Okereafor, I. Ekong, I. O. Markson and K. Enwere, "Fingerprint biometric system hygiene and the risk of covid-19 transmission," *JMIR Biomedical Engineering*, vol. 5, no. 1, 2020.

[7] G. Nixon, "What is ‘time theft’ and why are some employers so worked up about it?," CBC, 20 January 2023. [Online]. Available: https://www.cbc.ca/news/business/time-theft-tensions-workplaces-employers-1.6718610. [Accessed 6 June 2023].

[8] E. Sawall, A. Honnef, M. Mohamed, A. A. S. AlQahtani and T. Alshayeb, "COVID-19 Zero-Interaction School Attendance System," *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pp. 1-4, 2021.

[9] J. Sauro, "Measuring usability with the system usability scale (SUS)," MeasuringU, 3 February 2011. [Online]. Available: https://measuringu.com/sus/. [Accessed 5 April 2023].

[10] D. Kirkpatrick, "Google: 53% of mobile users abandon sites that take over 3 seconds to load," Marketing Dive, 12 September 2016. [Online]. Available: https://www.marketingdive.com/news/google-53-of-mobile-users-abandon-sites-that-take-over-3-seconds-to-load/426070/. [Accessed 5 April 2023].

[11] Statista, "Size of top Android apps by category 2022," Statista, 22 March 2022. [Online]. Available: https://www.statista.com/statistics/1296527/size-top-android-apps/. [Accessed 5 April 2023].

[12] peopleHum, "What is attendance management?," peopleHum, [Online]. Available: https://www.peoplehum.com/glossary/attendance-management. [Accessed 5 April 2023].

[13] M. Rouse, "Fingerprint scanner," Techopedia, 20 December 2016. [Online]. Available: https://www.techopedia.com/definition/29808/fingerprint-scanner. [Accessed 5 April 2023].

[14] S. Amsler and S. Shea, "What is RFID and how does it work?," TechTarget, 31 March 2021. [Online]. Available: https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification. [Accessed 5 April 2023].

[15] A. Kilich, "What is face detection and how does it work?," Cameralyze, [Online]. Available: https://www.cameralyze.co/blog/what-is-face-detection-and-how-does-it-work. [Accessed 5 April 2023].

[16] D. Dwivedi, "Face detection for Beginners," Medium, 21 July 2022. [Online]. Available: https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9. [Accessed 5 April 2023].

[17] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137-154, 2004.

[18] G. S. Behera, "Face detection with Haar Cascade," Medium, 29 December 2020. [Online]. Available: https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08. [Accessed 5 April 2023].

[19] Google, "ML Kit," Google, [Online]. Available: https://developers.google.com/ml-kit/guides. [Accessed 5 April 2023].

[20] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran and M. Grundmann, "Blazeface: Sub-millisecond neural face detection on mobile gpus," *arXiv preprint arXiv:1907.05047*, Jul. 2019.

[21] Google, "Face detection," Google, [Online]. Available: https://developers.google.com/ml-kit/vision/face-detection. [Accessed 5 April 2023].

[22] V. Bruce and A. Young, "Understanding face recognition," *British Journal of Psychology*, vol. 77, no. 3, pp. 305-327, Aug. 1986.

[23] Kaspersky, "What is facial recognition - definition and explanation," Kaspersky, 9 February 2022. [Online]. Available: https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition. [Accessed 5 April 2023].

[24] S. Madeleine, "Normalization, zero centering and standardization of CT Images," IMAIOS, 21 January 2022. [Online]. Available: https://www.imaios.com/en/resources/blog/ct-images-normalization-zero-centring-and-standardization. [Accessed 5 April 2023].

[25] A. R. Lahitani, A. E. Permanasari and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," *2016 4th International Conference on Cyber and IT Service Management*, pp. 1-6, doi: 10.1109/CITSM.2016.7577578, 2016.

[26] F. Karabiber, "Cosine similarity," LearnDataSci, [Online]. Available: https://www.learndatasci.com/glossary/cosine-similarity/. [Accessed 5 April 2023].

[27] J. Bell, "What is machine learning?," *Machine Learning and the City*, pp. 207-216, 2022.

[28] P. Singh, "Supervised machine learning," *Learn PySpark*, pp. 117-159, Sep. 2019.

[29] P. Pancholi and S. Patel, "A brief Introduction of Machine Learning with different Tasks and Applications," *International Journal of Recent Advanced Research*, vol. 6, no. 1, 2019.

[30] IBM, "What are neural networks?," IBM, [Online]. Available: https://www.ibm.com/id-en/topics/neural-networks. [Accessed 5 April 2023].

[31] AWS, "What is a Neural Network," Amazon, [Online]. Available: https://aws.amazon.com/what-is/neural-network/. [Accessed 5 April 2023].

[32] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[33] AndroidKT, "Activation function for output layer in regression, binary, multi-class, and multi-label classification," AndroidKT, 21 January 2023. [Online]. Available: https://androidkt.com/activation-function-for-output-layer-in-regression-binary-multi-class-and-multi-label-classification/. [Accessed 5 April 2023].

[34] S. Saha, "A comprehensive guide to Convolutional Neural Networks - the eli5 way," Medium, 16 November 2022. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed 5 April 2023].

[35] S. Mutuvi, "Introduction to machine learning model evaluation," Medium, 24 September 2021. [Online]. Available: https://heartbeat.comet.ml/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f. [Accessed 5 April 2023].

[36] Google, "Classification: Accuracy," Google, [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy. [Accessed 5 April 2023].

[37] Google, "Classification: Precision and Recall," Google, [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recal. [Accessed 5 April 2023].

[38] J. Korstanje, "The F1 score," Medium, 31 August 2021. [Online]. Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6. [Accessed 5 April 2023].

[39] Oracle, "What is a database?," Oracle, [Online]. Available: https://www.oracle.com/id/database/what-is-database. [Accessed 5 April 2023].

[40] Google, "Firestore documentation | Google Cloud," Google, [Online]. Available: https://cloud.google.com/firestore/docs. [Accessed 12 June 2023].

[41] Visual Paradigm, "What is Entity Relationship Diagram (ERD)?," Visual Paradigm, [Online]. Available: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/. [Accessed 5 April 2023].

[42] Visual Paradigm, "What is Use Case Diagram?," Visual Paradigm, [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/. [Accessed 20 June 2023].

[43] A. Shah, "The smaller, the better: Reducing android app size," Medium, 30 September 2020. [Online]. Available: https://betterprogramming.pub/the-smaller-the-better-reducing-android-app-size-3b063a40ded7. [Accessed 5 April 2023].

[44] AWS, "MLPER-10: Perform a performance trade-off analysis," Amazon, [Online]. Available: Available:https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/mlper-10.html. [Accessed 5 April 2023].

[45] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA,* pp. 815-823, doi: 10.1109/CVPR.2015.7298682, 2015.

[46] G. Boesch, "VGG very deep convolutional networks (vggnet) - what you need to know," viso.ai, 16 March 2023. [Online]. Available: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/. [Accessed 5 April 2023].

[47] Keras, "Keras applications," Keras, [Online]. Available: https://keras.io/api/applications/. [Accessed 5 April 2023].

[48] "Face Verification on Labeled Faces in the Wild," [Online]. Available: https://paperswithcode.com/sota/face-verification-on-labeled-faces-in-the. [Accessed 5 April 2023].

[49] S. Chen, Y. Liu, X. Gao and Z. Han, "MobileFaceNets: Efficient cnns for accurate real-time face verification on mobile devices," *arXiv preprint arXiv:1804.07573,* Apr. 2018.

[50] Q. Cao, L. Shen, W. Xie, O. M. Parkhi and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," *arXiv preprint arXiv:1710.08092v2*, May 2018.

APPENDICES

Client Interview

The author's team conducted an interview with a potential client as part of their research and development process for the mobile attendance application system. This interview aimed to gain valuable insights and feedback from the client to better understand their needs, challenges, and expectations regarding attendance management. The following are the details regarding the interview:

- When: 26 May 2023
- Where: PT. Sentra Solusi Informatika
- Interviewer: Bryan Putra
- Interviewee: Lisna Winda (HRD Manager)

The interview was split into three main parts. The first part involves the introduction of both parties involved in the interview. Bryan introduces himself. The interviewee also introduces herself as Winda and her position in PT. Sentra Solusi Informatika, a company engaged in IT software for Travel, Tour, and Hotel. The next part of the interview involves the main bulk of the discussion. This part is further split into:

- **Understanding the current system**

  Bryan asks Winda to briefly explain the workflow of the current attendance system, which involves using a fingerprint system to collect data in the form of name, date, tap-in time, and tap-out time on a weekly basis and manually inputting it into Excel. They discuss the tolerance for lateness and the penalties for exceeding the allowed lateness. Winda explains that there is a tolerance of 7 hours of lateness per month, and employees are allowed to determine their own tap-in and tap-out times, but there are limits to the arrival and departure times that must be followed.

- **Problems regarding the current system**

  Bryan explores the issues faced by users and administrators of the fingerprint-based attendance system. They discuss challenges such as power outages leading to the inability to record attendance and the manual process of handling absences and finding missing records. Winda mentions that many users forget to tap in or tap out, and when employees are absent, the search and handling process is still done manually.

The final part of the interview involves demoing a prototype of the mobile attendance application being developed by the author and his team and receiving the interviewee's thoughts about the application. The following are the main points of this section:

- The application is beneficial as it helps to remind users to tap in/ tap out in the form of notifications
- The application lacks several features such as the aforementioned tolerance work time feature as well as leave/ permission quota to facilitate their needs
- The interviewee expresses concerns regarding the possibility of employees tapping in/out from outside of the working environment

In conclusion, the interview revealed several client needs. The potential client faces challenges with the current fingerprint-based attendance system, including power outages affecting attendance recording and manual processes for handling absences. As a solution, the proposed mobile application can provide reminders to employees to tap in and tap out through notifications on their smartphones. This will facilitate timely and accurate attendance recording. Additionally, suggested additional features include monitoring lateness tolerance usage during the month and tracking leave, permissions, and sick days. These features will assist in more efficient administration and tracking of employee attendance.

The full interview transcript (in Indonesian) can be found below:

Table A.1 : Interview transcript

Bryan	**Ka Winda**
Selamat siang kak, terima kasih telah menyepatkan waktunya ya untuk bisa wawancara dengan saya, untuk kepentingan skripsi group saya, Sebelum saya mulai, mungkin kaka bisa perkenalkan terlebih dahulu, nama kaka, dan posisi kaka di perusahaan ini	Perkenalkan nama saya Winda, saya sekarang di posisi staff HRD, kita bergerak di bidang IT software untuk Travel, Tour, Hotel.
Maaf kak, nama perusahaannya tadi apa ya?	PT. Sentra Solusi Informatika
Pertanyaan pertama, nah kak, kalo boleh tanya tuh, sekarang di perusahaan kaka ini data absensinya itu pake sistem kyk gimana ya kak ? bisa dijelasin alurnya secara sekilas saja	Kita pake system fingerprint, jadi setiap seminggu sekali saya Tarik data. Terus di situ ada nama, tanggal, jam tap in dan tap outnya, terus saya input ke excel sendiri. Di situ, jadi, keliatan misal ada yang terlambat berapa menit, itu mengurangi toleransi jam yang satu bulan, kita kan ada toleransi keterlambatan 7 jam, jadi setiap 1 bulan gk boleh lebih dari 7 jam.

Berarti dari sini, kaka itu di sistem absensinya itu ada toleransi jam, berarti karyawan itu perbolehkan untuk pakai, dan tap in dan tap outnya terserah mau kapan ya ?	Kita itu jam masuknya kan 8:30 dan pulangnya 17:30. Jadi setiap hari itu, kita hanya bisa membayar keterlambatan, 30 menit, walaupun kita misal dateng terlambat 1 jam, di hari itu, kita hanya bisa bayar 30 menit, dan 30 menitnya itu mengurangi toleransi yang 7 jam itu
Kalo misal masalah, kan kaka pake sistem sidik jari kan ya ?	Iya
Nah, kalo kaka misal dari segi user itu, untuk masalah sistem absen pake mesin jari ini , apakah biasanya ada kendala atau keluh kesal yang sangat sering terjadi gitu ya, di sisi user dan admin tapi di sisi adminnya itu, sebut saja yang bisa di kaitkan dengan usernya, tidak harus se-detail untuk admin saja ? Soalnya kita perspectivenya mau membantu experience user dan admin yang masih bisa di kaitkan dengan user gitu.	Kita kebetulan nih sering mati lampu, nah kalo mati lampu, gk bisa absen. Dan dari sisi admin, kita gk bisa tarik data apapun, karena banyak yang gk bisa absen. Jadi admin harus manual input datanya.
Ada lagi gk selain itu?	Kebanyakan user sering lupa tap in tap out. Terus misal kalao mereka ada pulang cepet atau datang terlambat, mereka sering lupa tap in atau tap out. Misal ada yang gk masuk, contoh 3 hari. Tapi kan dari awal, saya input rekap absennya itu seminggu sekali. Misal dalam seminggu itu ada beberapa orang gk masuk. Itu saya harus cari manual, caranya dengan Tarik data dari fingerptrint itu, saya cariin satu2. User ini 3 hari gk ada, kemana ? Saya paling, bikin keterangan manual, Contoh “ User ini gk masuk, izin atau sakit “ gitu

Tadi kan, sebelumnya kaka udah mencoba sekilas prototype aplikasi kita. Nah, menurut kaka itu, apakah aplikasi kami itu bisa menyelesaikan, masalah2 dan kendala yang kaka barusan sebut? Dari sisi admin dan dari sisi user	Cukup membantu untuk yang sistem tap in dan tap out. Karena itu kan sistem mobile, jadi misal ada yang lupa tap in / tap out kan. User bisa tau, ada muncul warning, di hp mereka masing2, dan mereka segera tap in / tap out nya gitu, mereka bisa pencet sendiri lah tanpa kita manualin dari sisi admin.
Mungkin dari sisi kayak, kalao misal, kan kalian pakai sistem mesin ya ?	Iya
Jadi kan mungkin kaka ngomong tadi seperti, mati lampu atau kendala yang lain, kan jadi tanggung jawab pemilik smartphone.	Iya itu mempermudah. Dan untuk waktunya mungkin bisa di buat general. Kan hp masing2 bisa beda waktunya. Ikutin jam Dunia
Apakah ada fitur2 lain yang mungkin kaka merasa perlu sebagai user dan sebagai admin yang bisa kita pakai yang belum di aplikasi kami? Yang bisa kita implementasikan	Fitur tambahannya mungkin bisa di tambah, yaitu yang kita kan ada toleranasi 1 bulan, 7 jam. Nah, mungkin bisa di tambah fitur itu, jadi kita dari sisi admin, bisa liat user ini, pemakaian selama beberapa hari sudah terpakai berapa jam. Terus ada fitur lagi tambahan mengenai cuti, sisa cuti, saldo cuti. Pemakaian selama bulan ini berapa, terus pemakaian selama satu tahunnya berapa.
Pertanyaan terakhir, kalao menurut kaka itu, kaka kebayang gk, kalao aplikasi kami itu mungkin menonjolkan masalah2 baru kayak contoh, misalkan pakai sistem sidik jari ada + -. Kalao misal datanya kan pasti, datanya gk bisa minta temennya tap innya, menurut kakak itu – apa saja yang bakal bsia muncul kalao kita pake aplikasi hpini, dengan datanya pakai muka, sebagai penanda karyawan masuk?	-nya Mungkin bisa tap in atau tap out di sembarang tempat, di luar jam kerja. Misal kita kan ada visit ke client, kalao misal absensinya by mobile kyk gitu, mereka bisa, abis dari client, nongkrong dulu di luar, terus nunggu waktu jam pulang. Tap in tapoutnya teserah mereka gitu, gk di dalam lingkungan kantor, tidak dalam aktivitas kerja gitu.
Ada lagi gk kak?	Terus kalo ini, registrasinya pakai apa ya?

Kalao kita sih untuk sekarang, jadi kita kebayangnya tuh, setiap kali ada karyawan baru, dari database perusahaannya itu, perusahaanmya nge-generate username sama password untuk masuk ke aplikasi kami. Soalnya kalao kita open to public untuk akses register kan, nanti banyak orang bisa register sembarangan. Nah kita buat khusus untuk perusahaan nih, jadi username sama passwordnya untuk loginnya itu, nanti setelah perusahaan nge hire karyawan, baru di kasih ke karyawan baru ini. Jadi karyawan baru ini tinggal pake username password itu untuk pake login ke aplikasi kami.	Kalao misal login itu. Misalkan ada yang re-sign. Login itu bisa di rename ke orang lain? atau bikin user baru? User ini yang bersangkutan udah resign bisa di timpah?
Tergantung perusahaan, perusahaan itu mau mencatat karyawan ini pernah kerja di sini atau gk. Kalao perusahaan butuh data itu, kita gk timpah, kita buat baru. Cuman yang lama, statusnya resign, tp kalao perusahaan butuh di timpah, bisa	Kalao misal ada yang hari ini ada yang gk ada tap in atao tap out, bisa di kasih warning. Misal, pagi itu kan kita jam 8:30, jam 10 ud di kasih warning, jadi file reportnya, hari ini yang gk masuk ataupun yang belum absen itu siapa aja. Dan tapoutnya di warningin berarti sekitar jam 18:00
Ini kasih warningnya ke usernya ya ?	Kalao bisa, iya
Kalao misal telat masuk, maksimum 1 jam setengah ya, jam 10 berarti ?	Iya
Kalao telat pulangnya harus jam 6 ya ?	Iya
Itu saja yang kepikiran ya ?	Baru itu aja

Mungkin kaka sebagai user nih, kaka merasa keberatan untuk memakai aplikasi ini tuh di sisi apa sih biasanya? Kalao mungkin kaka bisa bayangkan	Kalao dari sisi admin, program kami kan belum keliatan pemakaian per hari, dia, pemakaian toleransinya berapa lama, belum keliatan. Terus belum keliatan juga, untuk pemakaian cuti, izin, sakit, itu biasanya di bedain, belum ada fasilitas itu kan ya, secara detail, perusernya?
Beberapa ud ada sih, tp yang kaka mention belum ada	
Udah itu saja sih kak, pertanyaan wawancara dari saya, terima kasih ya sudah meluangkan waktu untuk menghadiri wawancara ini	Iya sama2, semoga membantu