

Cab Transaction Using Facial Recognition and Matching Engine

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Engineering

by

Kshitij Shukla (EU2152053)

Rohan Chavan (EU1152082)

Saniket Patil (EU2152014)

Under the guidance of

Mr. Vivian Lobo

(Assistant Professor)



Department of Computer Engineering

**St. John College of Engineering and
Management
University of Mumbai**

2019-2020

CERTIFICATE

This is to certify that the project entitled **“Cab Transaction using Facial recognition and Matching Engine”** is a bonafide work of

“Kshitij Shukla” (EU2152053)

“Rohan Chavan” (EU1152082)

“Saniket Patil” (EU2152014)

Submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Computer Engineering”**.

Mr. Vivian Lobo

Project Guide

Dr. Rahul Khokale
Head of Department

Dr. G. V. Mulgund
Principal

Project Report Approval for B. E.

This project report entitled **Cab Transaction Using Facial Recognition and Matching Engine** by *Kshitij Shukla , Rohan Chavan, Saniket Patil* is approved for the degree of *Bachelor of Engineering* in *Computer Engineering* from *University of Mumbai*.

Examiners

1.-----

2.-----

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature

Kshitij Shukla (2152053)

Signature

Rohan Chavan (1152082)

Signature

Saniket Patil (2152014)

Date:

Abstract

After booking a cab, at the end of the ride one can make the transactions with either cash, card, UPI, E-wallet etc. Sometimes these methods are not hassle free as they cause inconvenience to the customer. Therefore, a new system has been proposed for transactions, where face scan method will be used. The system in the cab will calculate the fare based on the distance and then it will authenticate the riders face and transact via connected central database. Face recognition begins with extracting the coordinates of features such as width of mouth, width of eyes, pupil, and compare the result with the measurements stored in the database and return the closest record (facial metrics). The main purpose of this research is to investigate different types of face recognition algorithms like Eigen face and Fisherface and Local Binary Patterns Histograms Face Recognizer. The open CV provides these recognition algorithms. This is done by comparing the receiver operating characteristics curve to implement in the given Transaction using Facial Recognition. In addition, it is noted that Local Binary Patterns Histograms Face Recognizer delivers better results than the Fisherface algorithms; If the user's input image matched with the trained dataset image then the User Profile and Transaction details gets loaded, and the subsequent trip details gets stored in the User Profile database. The database is connected to frame web server.

Table of Contents

	Abstract	v
	List of Figures	viii
Chapter 1	Introduction	3
	1.1 Motivation and Problem Statement	4
	1.2 Objectives	4
	1.3 Scope	4
Chapter 2	Review of Literature	5
	2.1 Face Recognition Techniques to Differentiate Similar Faces and Twin Faces	6
	2.2 Secure Authentication for Mobile Banking Using Facial Recognition	7
	2.3 Biometric Face Recognition Payment System	9
Chapter 3	Requirement Analysis	10
	3.1 Non Functional Requirements	10
	3.1.1 Hardware Requirement	11
	3.1.2 Software Requirement	11
Chapter 4	Design	12
	4.1 Use Case Diagram for Cab Transaction using facial recognition	12
	4.2 Level 0 Data Flow Diagram for Cab Transaction using facial recognition	13
	4.2 Level 1 Data Flow Diagram for Cab Transaction using facial recognition	13
	4.2 Level 2 Data Flow Diagram for Cab Transaction using facial recognition	14
	4.3 Sequence Diagram for Cab Transaction using facial recognition	15
	4.4 Activity Diagram for Cab Transaction using facial recognition	16
Chapter 5	Report on the Present Investigation	17
	5.1 Proposed System	17
	5.1.1 System Architecture	18
	5.2 Implementation	19
	5.2.1 Eigen faces face recognizer	19
	5.2.2 Fisherface face recognizer	20

	5.2.3 Local binary patterns histograms (LBPH) Face Recognizer	21
	5.2.4 Required Modules	23
	5.2.5 Prepare training data	24
	5.3 Data Preparation for Face Recognition	24
	5.4 Microservice Architecture	26
	5.5 Django	27
	5.6 REST API	27
	5.7 Django REST Framework	29
	5.8 Bottle Server	30
	5.9 Postman HTTP Server	31
Chapter 6	Results and Discussions	33
	6.1 Registration and Login	34
Chapter 7	Conclusion and Future Scope	40
	7.1 Conclusion	41
	7.2 Future Scope	41
	References	42
Appendix	Technologies Used and External Libraries	44
	Publication	46
	Acknowledgement	47

List of Figures

Figure No.	Figure Name	Page No.
2.1	System Architecture for Face Recognition Technology	6
2.2	Flow of proposed system	8
2.3	Biometric face recognition payment system	9
4.1	Use Case for Cab Transaction using facial recognition	12
4.2	Level 0 Data Flow Diagram for Cab transaction using facial recognition	13
4.3	Level 1 Data Flow Diagram for Cab transaction using facial recognition	13
4.4	Level 2 Data Flow Diagram for Cab transaction using facial recognition	14
4.5	Sequence Diagram for Cab transaction using facial recognition	15
4.6	Activity Diagram for Cab transaction using facial recognition	16
5.1	System Architecture	18
5.2	Image showing the variance extracted from a list of faces	20
5.3	Image of principal components using Fisherface algorithm	21
5.4	LBPH Face recognizer Process	22
5.5	LBPH Histogram	22
5.6	ROC curve between Fisherface and LBPH	23
5.7	Directory structure tree for training data	24
5.8	Data preparations for face recognition	25
5.9	Data preparations for face recognition	25
5.10	Microservice architecture	26
5.11	Django framework architecture	27
5.12	REST API architecture	29
5.13	Django REST framework	29
5.14	Bottle server architecture	30
5.15	Postman http client architecture	31
6.1	Registration Screen	34
6.2	Login Screen	34

6.3	Working architecture	35
6.4	Django administration login	36
6.5	Django Admin Dashboard	36
6.6	Logged in User Dashboard	37
6.7	Frame Capturing	37
6.8	Dataset of 60 images stored in user database	37
6.9	Trip/fare calculation dashboard	38

Chapter 1

Introduction

Current system of Payment in cabs for global markets like India are two phased, i.e., either physical or digital. The physical payment system means paying with Cash and Coins whereas digital payment method includes debit cards, Internet banking and most popular recently is paying with e-wallets and UPI. With the popularity in India of mobile payment platforms such as Paytm and PhonePe, QR codes can be found almost anytime, anywhere in Indian daily life. From luxury shopping centres to street vendors, consumers can make payments easily by scanning a QR code with their smartphones. Issue arises in both type of payment methods, you could not have the change in cash, maybe you forgot the money, or in digital payment methods, problems such as the vendor isn't accepting a particular wallet, or your mobile has no Internet connectivity. The awkwardness of such situation is a thing of past with the system we propose. We propose a new method of making transactions in a cab, i.e., Payment via Facial Recognition. The said system will have the ability to make a transaction from your bank with just your face, thus eliminating a need of carrying Cash, Cards, Mobile phones, etc. All you need is your face.

1.1 Motivation

The unavailability of a payment system which is both convenient and available at all times without the need to carry all payment methods such as cash, cards and mobile has motivated us to create a new payment system which eliminates the need of traditional payment methods.

1.2 Problem Statement

In today's world it's not convenient to have multiple payment methods all ready with you at all times and have different vendors accept different payment methods. On top of that a consumer forgetting to carry means of payment such as cards, cash is a real bother. Hence a payment system has to be made which eliminates the need of carrying any modes of payment such as cash, cards, mobiles with internet connection.

1.3 Objectives

The objectives are as follows:

- To design a real-time face recognition system for cabs transactions.
- To design a system where face can be used for logging into a profile.
- To design a system for cabs where face detection system can also be used to pay along with card or cash.
- To design a system where once you load your profile system can initiate a transaction without any card.
- To design a secure system for cab transaction using facial recognition technique.

1.4 Scope

Improving facial recognition accuracy using as less user images as possible. Security of the payment gateway engine, easy user interface.

Chapter 2

Review of Literature

Literature survey may also be known as a literature review. This segment consists of an existing and established theory and research in the report range. Literature review is a type of review article. A literature review is a scholarly paper, which includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews are basis for research in nearly every academics fields. . A narrow-scope literature review may be included as part of a peer-reviewed journal article presenting new research, serving to situate the current study within the body of the relevant literature and to provide context for the reader. In such a case, the review usually precedes the methodology and results sections of the work.

2.1 Face Recognition Techniques to Differentiate Similar Faces and Twin Faces.

In computer technology image based on identical twin, face recognition technology is challenging task. Traditional facial recognition system exhibit poor performance in differentiating identical twins and similar person under practical conditions. The following methods for differentiate identical twins. Traditionally lot of manual experiments were performed to identify twins and also to recognize their features with difference, and many more systems were existed to show differences in twins by using finger prints, voice and iris as part of pattern recognition. In existing methods, many techniques are used for twin's identification like finger print, voice and iris recognition [1].

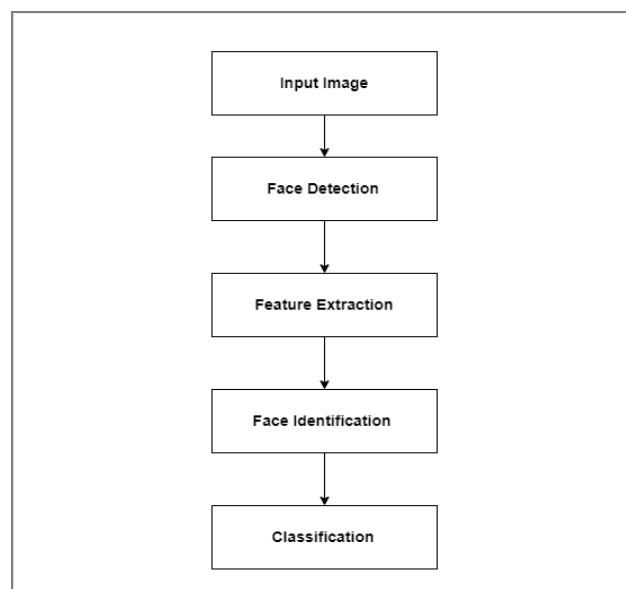


Figure 2.1: System Architecture for Face Recognition Technology.

The process of finger print identification is used to identify unique person in industry or organizations. In Figure 2.1 the method proposes a scan image taken from the person and compare with database for identification. The iris recognition also similar method to finger prints identification [1].

This study concludes to identify the twins and similar faces using Gabor filter and Multi-scale Fast Radial symmetry transform. Gabor filter is used to differentiate when faces are not similar. But multi-scale Fast Radial Symmetry Transform technique is used to differentiating identical twins and similar faces using facial aspects. This method gives good performance compare than Gabor filter method [1].

2.2 Secure Authentication for Mobile Banking Using Facial Recognition.

In the past decades, banking was done inside the banking hall which was tasking to both the customers and the bankers. The long queues, paper-based data and even the time taken to perform even the smallest transaction can be an uphill task. This has now been a thing of the past since the advent of the internet and mobile phones. The number of online banking users has increased in Nigeria and indeed the world; this has led to many experts in mobile banking software and mobile phone technology to research new and convenient methods for customers to perform banking transactions remotely via their mobile phones. Mobile banking is also known as mobile phone bank. It is referred to as the using mobile phone for banking related business. Security has become a primary concern in order to provide protected mobile transaction between the clients and the bank servers. Secure authentication of client information depends on some fundamental security approaches which will not jeopardize the client sensitive information. This has led to different researches ranging from single-factor authentication, two-way authentication, and multifactor authentication. Bearing in mind the cost of providing these services to clients, most banks are weary of balancing profit making and security. In Nigeria today, most mobile banking applications use the single-factor authentication which consist of the username and password. Secure mobile banking will build confidence in customers knowing that their information is secure and they can carry out secure transactions without fear of man-in-the-middle attacks. Though the issue of theft strongly depends on how a client protects his/her mobile phone device from third parties [2].

Before access will be granted, the user will have to take a facial photograph to have access to his/her account, the geometry of the face, distance of the eyes and the nose is compared. This photograph will be compared with the photograph in the bank server and the NCC server for verification, if it passes the verification, access will be granted, otherwise it will be denied. In the event of unauthorized access, a security alert message will be sent to the bank [2].

On the program end, the security is multi-factored. A username and password level, a facial recognition level and a secret question and answer level. Users are limited to five trials after which access is denied. The response time for a complete transaction is seven minutes maximum putting other limiting factors into consideration; the false acceptance rate is 3%.the implication of false acceptance rate is given by elements on image background and facial defects [2].

The advantages of this system include;

- i. Secure and transaction
- ii. Cost effective
- iii. Transaction can be done anywhere remotely (with availability of mobile network)

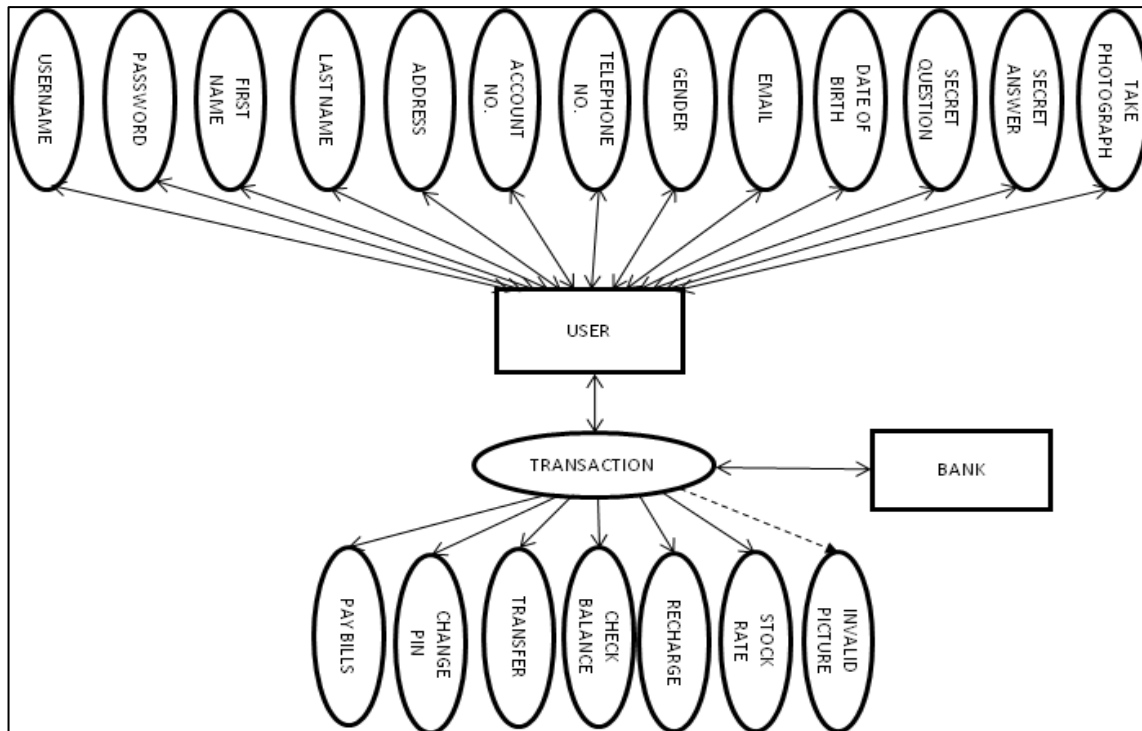


Figure.2.2: Flow of proposed system [2]

In Fig. 2.2 two dependable databases are also used to authenticate genuine users; these databases are the NCC database and the issuing bank database. In an advent of facial defection, users are advised to see their bank information technology operators.

When the security is trusted, it will build customer satisfaction and discourage the use of cash. The number of mobile phone users increases by the day and the success of the security on mobile banking will encourage new users to adopt the trend. Introducing this level of authentication using facial recognition on users' account to authenticate from the Nigeria Communication Commission's database and the facilitating bank's database, will no doubt contribute to mitigate mobile banking fraud. In a bid to make the Nigerian economy cashless, attention should be focused on security [2].

2.3 Biometric Face Recognition Payment System

Use of payment cards in various places such as shopping, restaurants, lodges and online payment for booking hotels, movie tickets, flight and train tickets etc are increasing day by day. Therefore, the problem is that a person has to carry payment cards along with him and keep the cards secure to use it all the time. This also lacked security. In the present work the biometricface recognition payments is used in all kinds of payments. Thus, it avoids the need to memorize different passwords [3].

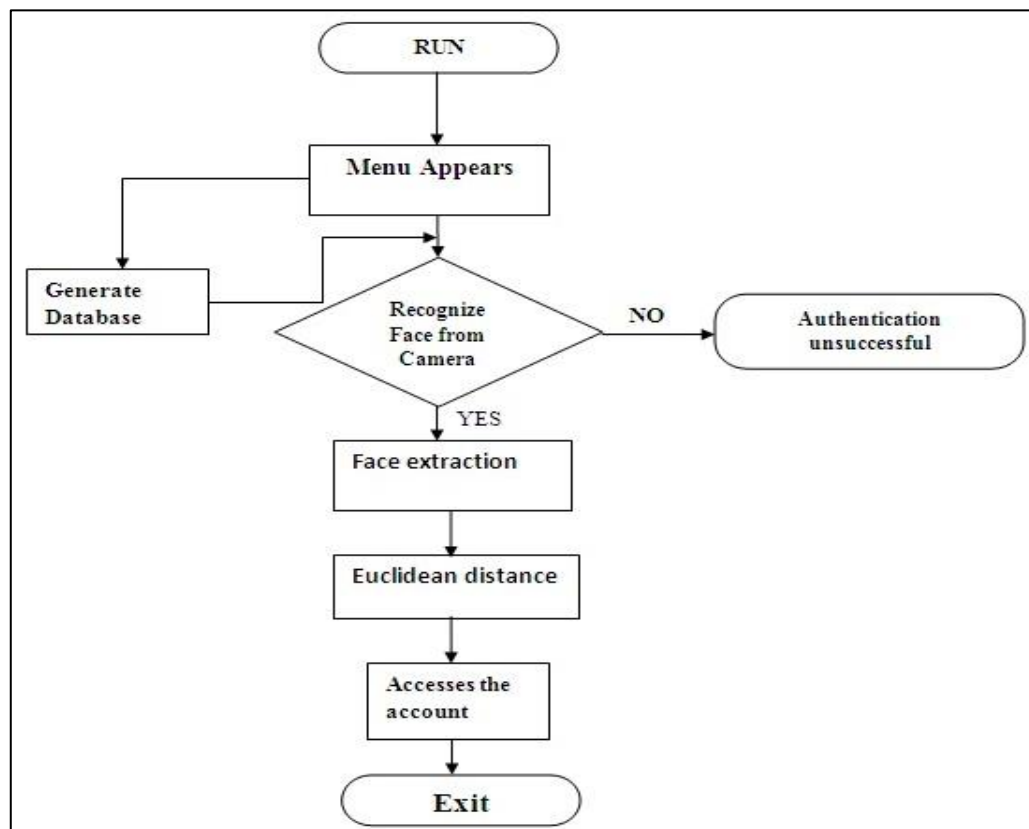


Figure 2.3:Biometric face recognition payment system [3].

Figure 2.3 shows how face recognition payment system is safe, secure and even easy to use. It is reliable and more efficient compared to other payment technologies. A general design of online payment system using face recognition is proposed. The methods adopted for face recognition are by finding the Eigen faces and Euclidean distance [3].

In this literature survey, the face recognition is used in all kinds of payments. For any online payments, the user need not use debit or credit card. A person need not carry card and remember the password for the transaction. Face recognition system is being proposed for all transactions. This is found to be more safe, secure and even easy to use [3].

Chapter 3

Requirement Analysis

Requirement Analysis is also called as the requirement engineering. In system engineering and software engineering requirement engineering is used. Requirement analysis mainly focuses on the task that determines the conditions or the needs of a particular project.

Requirement analysis is the most important phase of any project as it can be responsible for the success or the failure of the project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. Requirement analysis is a long and a tiring process, it is very much important to take stakeholders into consideration. Requirement analysis can only be successful if proper communication is done. A person with good communication skills only will be able understand what exactly the user or the stakeholder wants to be included in the project.

3.1 Non – Functional Requirements

As the name suggest these are the requirements that are not directly interacted with specific Functions delivered by the system.

- **Functionality:** This software will deliver on the functional requirements.
- **Flexibility:** It provides the users to draw the character easily.
- **Learn ability:** The software is very easy to use and reduces the learning work.

3.1.1 Hardware Requirements

- Internet Connectivity.
- Web Camera
- Storage Space: Minimum 32 GB free; Recommended 64 GB or more
- Memory (RAM): Minimum 4 GB; Recommended 8 GB or above

3.1.2 Software Requirements

- Python 3.x
- OpenCV
- Django
- Django Rest Framework
- Bottle Server
- Postman
- Sqlite Database

Chapter 4

Design

4.1 Use Case for Cab Transaction using facial recognition

Here we have used this use case diagram to represent user's interaction with the system. The following Use Case diagram shows the relationship between the user and the different use cases in which the user is involved. The use cases have been depicted by ellipses.

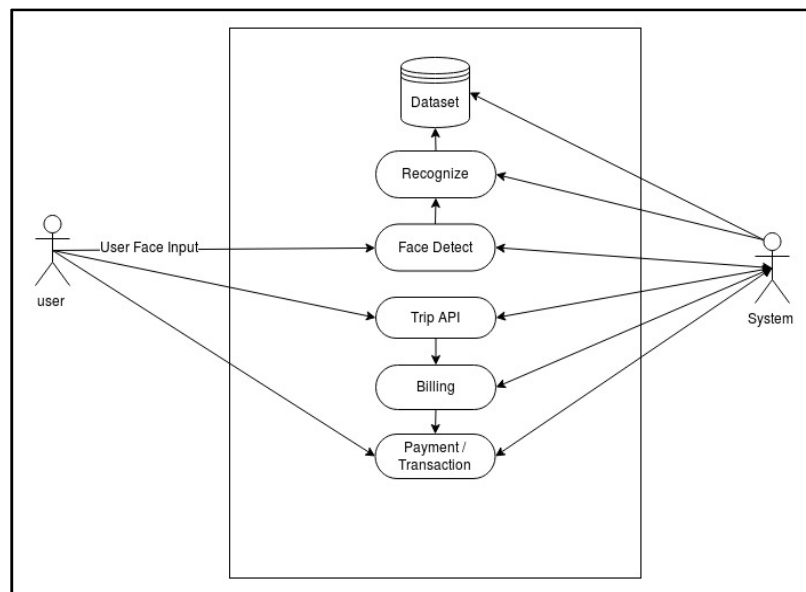


Figure 4.1: Use Case for Cab Transaction using facial recognition

Fig 4.1 explains the overall use case diagrams which are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities. They also help identify any internal or external factors that may influence the system and should be taken into consideration.

4.2 Data Flow Diagram for Cab Transaction using facial recognition

A Data Flow Diagram (DFD) is a graphical portrayal of the "stream" of information through a data framework, displaying its procedure angles. A DFD is frequently utilized as a fundamental advance to make an outline of the framework without really expounding, which can later be explained.

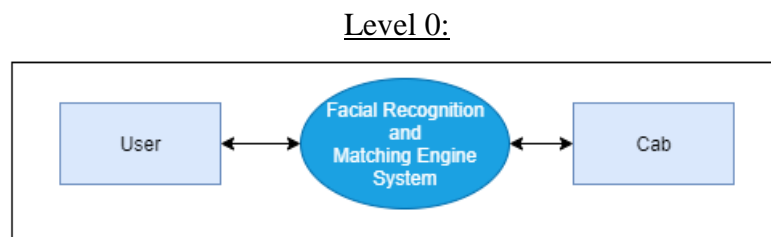


Figure 4.2 Level 0 DFD for Cab Transaction using Facial Recognition and Matching Engine.

There will be broadly 2 entities that will be interacting with the FRaME. Users from web client for registration and Image capturing and Clients side from Cabs for making transactions.

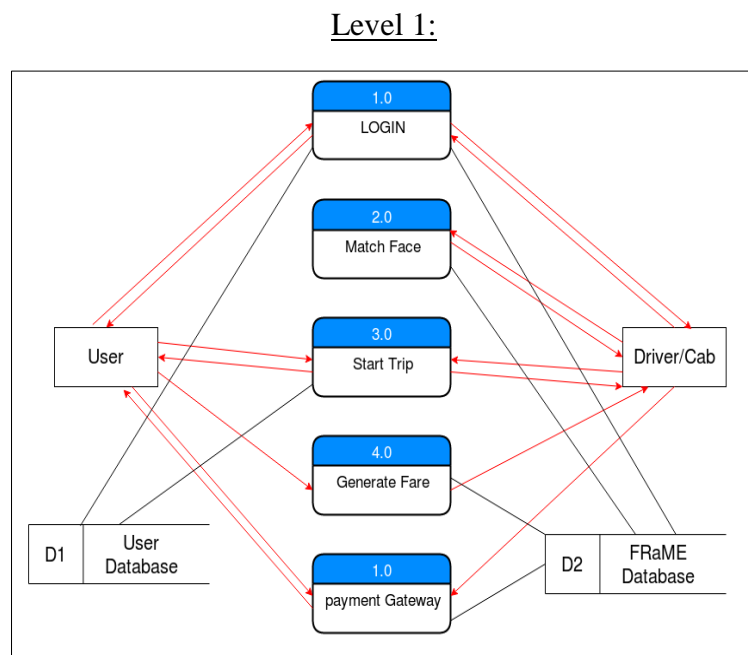


Figure 4.3 Level 1 DFD for Cab Transactions using Facial Recognition and Matching

Figure 4.3 depicts the level 1 DFD for our system. It portrays what modules of the system user interacts while signing up and which modules of the system user interacts which making a transaction in the cab.

In level 2 DFD how the individual data shows how the is flow happening in the system from one module to another.

The following are the data flows.

Level 2:

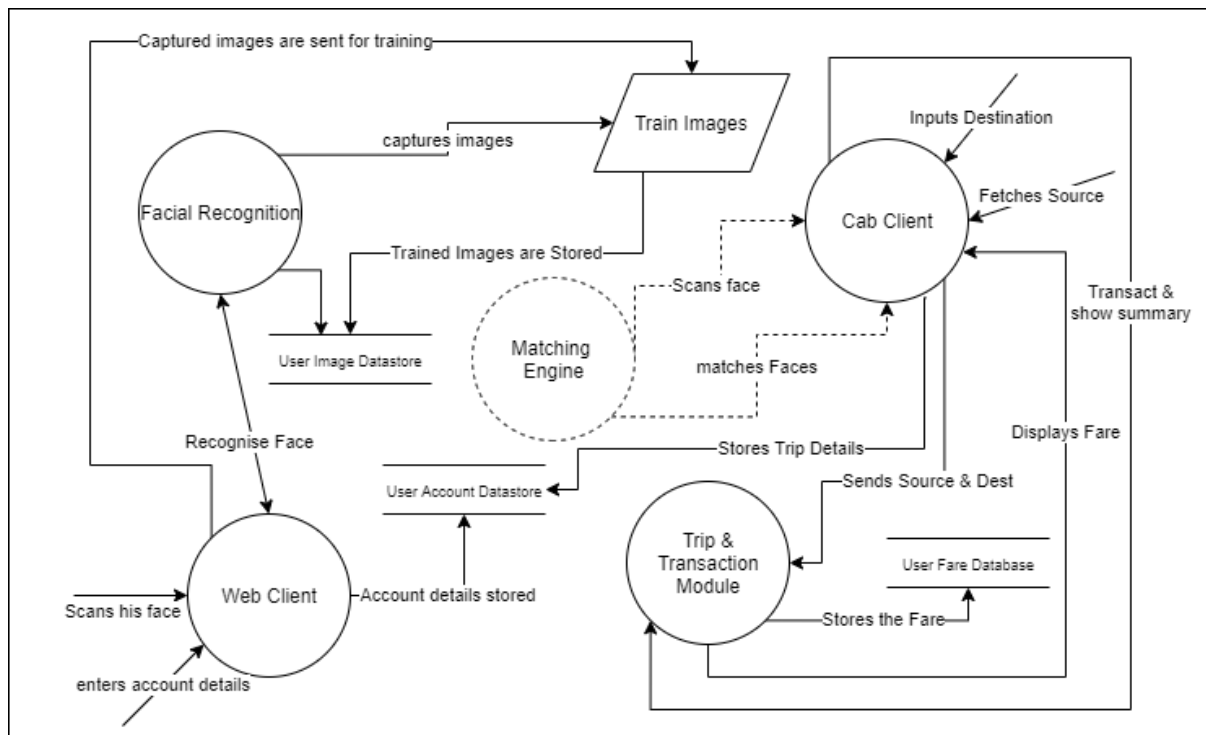


Figure 4.4 Level 2 DFD for Cab Transaction using Facial Recognition and Matching.

The modules show in the above figure is Facial Recognition, Web client, Matching engine, Cab Client, Training module, Trip module. The data stores are User Image data store where all the captured user images will be store, User account data store where all the user details such as username, email id is stored, User Fare data store, where all the fares of the user is stored.

4.3 Sequence Diagram for Cab Transaction using facial recognition

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place.

System can also use the terms event diagrams or event scenarios to refer to a sequence diagram.

Sequence diagrams describe how and in what order the objects in a system function.

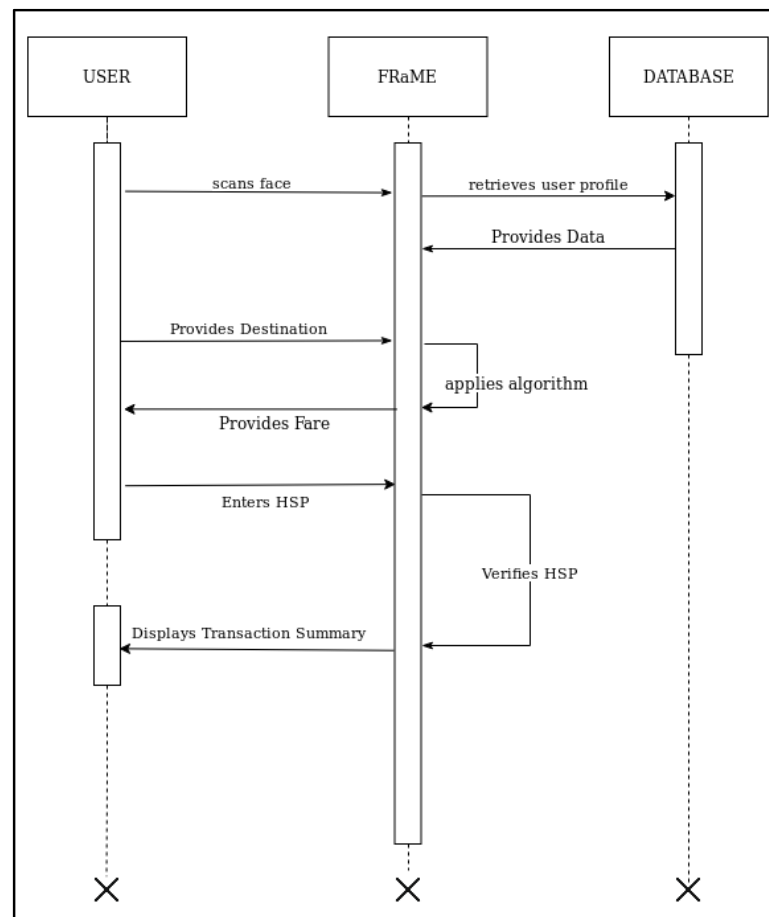


Fig 4.5 Sequence Diagram for Cab Transaction using facial recognition

In Fig 4.5 sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. The dataset used by facial recognition also uses the same database in a specific user directory structure. The proposed system consists of two parts which are face recognition and matching engine. The Authenticator is one of the most important part of the system, it is the bridge between the web-app and the frame engine, it identifies whether an particular user already exists or not.

4.4 Activity Diagram for Cab Transaction using facial recognition

Activity diagram is a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.

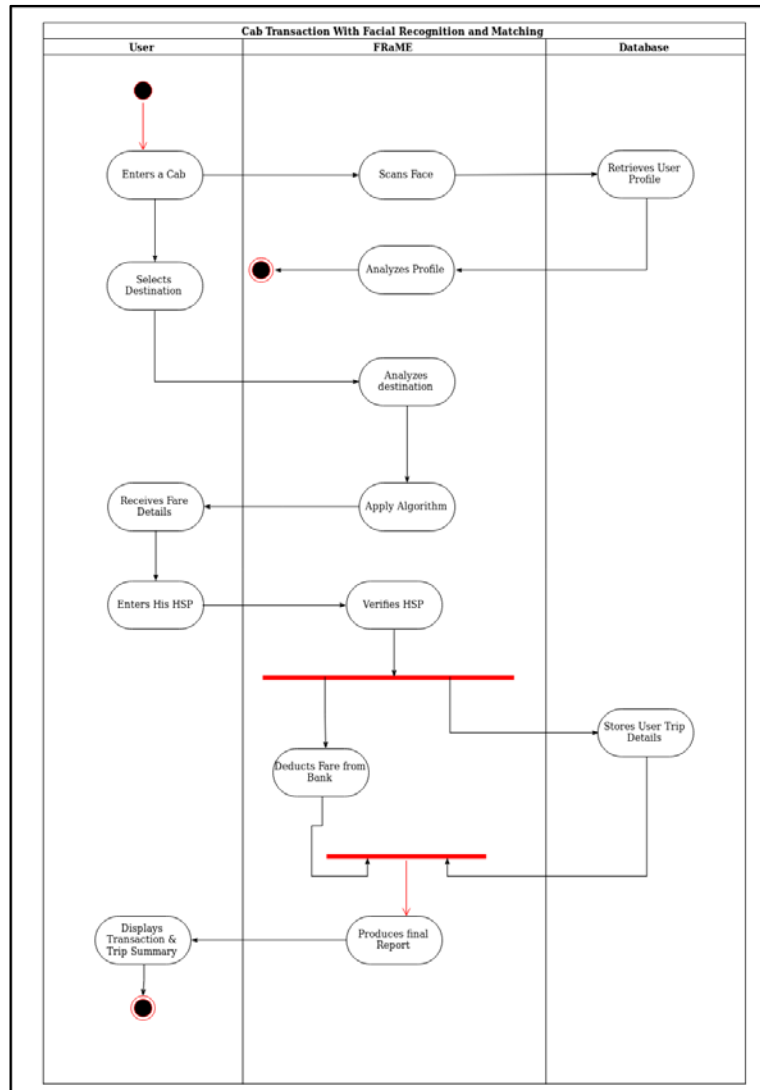


Fig 4.6 Activity Diagram for Cab Transaction using facial recognition

The fig 4.6 shows the activity diagram of Cab Transaction using Facial Recognition and Matching Engine in a sequence. The flow is sequential at most points but also forked at points like verifying the high security password where there are parallel operations such as deducting the fare from bank as well as storing the user trip details. At the same time both operations join together into a single sequence to produce a final report.

Chapter 5

Report on present investigation

5.1 Proposed System

The proposed system involves automated cashless payment system using facial recognition. This system will work as an API between the payment system and any other application in which face of the user can be used for authentication. Using this system, the user can directly pay fares for the cab rides or hotel stay etc. The initial step of the system is the registration process. During the time of registration, the system will collect the facial information of the user by asking user to upload an video of user's face. The system will then extract the frames from that video which will be used stored in the database for further use. Every user has a specific user directory, in which all the frames collected from the video will be saved. This dataset of images will be used to train our facial recognition model powered by OPENCV, Local Binary Pattern Histogram (LBPH) Algorithms. After the complete registration process, the user can login into the system by using facial recognition, and selects the destination. The system will then allocate a nearby cab to the user, and ensure the pickup for that user. Based on the pickup point and the destination, an invoice will be generated and the amount will be shown to the user. After the completion of the ride, the user will be redirected to the payment engine, where user will be prompted to enter his special pin to complete the transaction.

5.1.1 System Architecture

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.

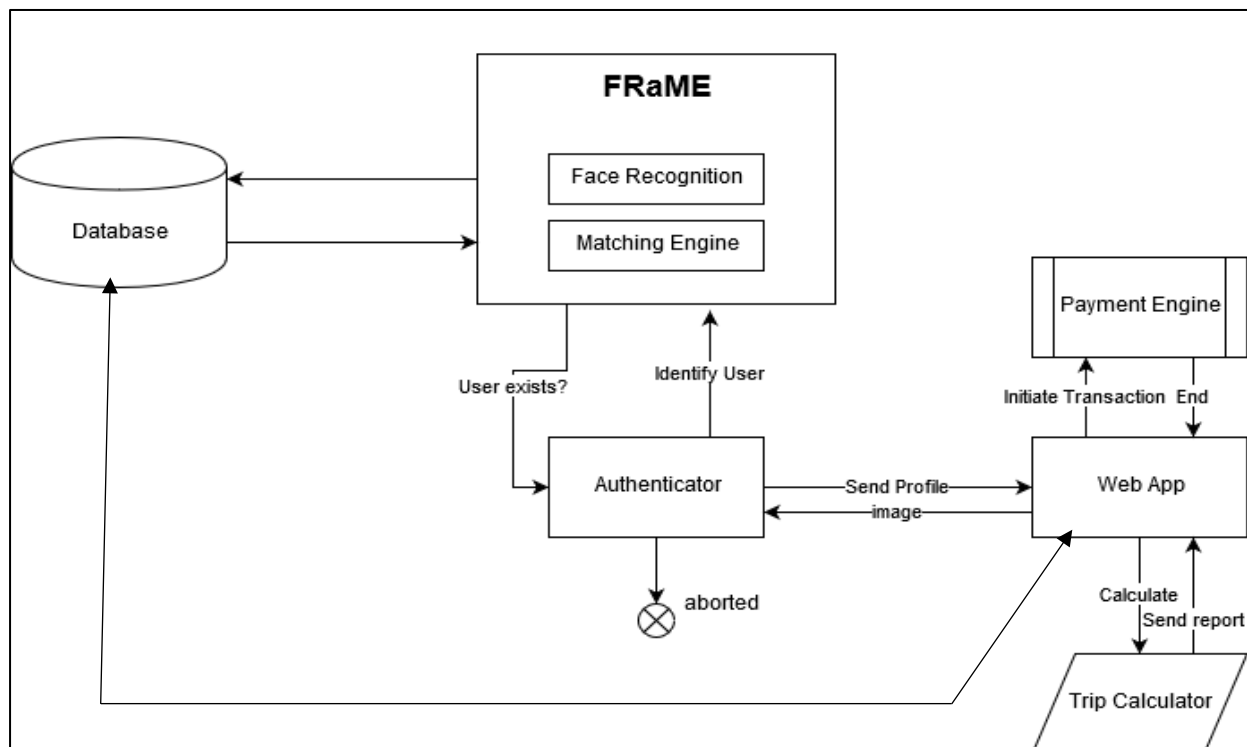


Figure 5.1 System Architecture

Figure 5.1 shows system is a two tier application, First tier is the webapp and second tier is the facial recognition and matching engine. The web app manages the calculation for the distance the cab travels and the payment engine. Authentication for the webapp is done by the face recognition and matching engine which is connected to the database. The dataset used by facial recognition also uses the same database in a specific user directory structure. The facial recognition and matching engine consists of two part which are face recognition and matching engine. The authenticator is one of the most important part of the system, it is the bridge between the webapp and the frame engine, it identifies whether a particular user already exists or not.

5.2 Implementation

There are 4 easy steps to computer coding facial recognition, which are similar to the steps that our brains use for recognizing faces. These steps are:

- 1) Data Gathering: Gather face data (face images in this case) of the persons user want to identify.
- 2) Train the Recognizer: Feed that face data and respective names of each face to the recognizer so that it can learn.
- 3) Recognition: Feed new faces of that people and see if the face recognizer user just trained recognizes them.
- 4) OpenCV has three built-in face recognizers and thanks to its clean coding, user can use any of them just by changing a single line of code. Here are the names of those face recognizers and their OpenCV calls:
 - EigenFaces – `cv2.face.createEigenFaceRecognizer()`
 - FisherFaces – `cv2.face.createFisherFaceRecognizer()`
 - Local Binary Patterns Histograms (LBPH) – `cv2.face.createLBPHFaceRecognizer()`

5.2.1 Eigenfaces face recognizer

This algorithm considers the fact that not all parts of a face are equally important or useful for face recognition. Indeed, when user look at someone, user recognize that person by his distinct features, like the eyes, nose, cheeks or forehead; and how they vary respect to each other. In that sense, user are focusing on the areas of maximum change. For example, from the eyes to the nose there is a significant change, and same applies from the nose to the mouth. When user look at multiple faces, user compare them by looking at these areas, because by catching the maximum variation among faces, they help user differentiate one face from the other. In this way, is how EigenFaces recognizer works. It looks at all the training images of all the people as a whole and tries to extract the components which are relevant and useful and discards the rest. These important features are called principal components.[10]

Principal components, variance, areas of high change and useful features indistinctly as they all mean the same.

EigenFaces recognizer trains itself by extracting principal components, but it also keeps a record of which ones belong to which person. Thus, whenever user introduce a new image to the algorithm, it repeats the same process as follows:

1. Extract the principal components from the new picture.
2. Compare those features with the list of elements stored during training.
3. Find the ones with the best match.
4. Return the 'person' label associated with that best match component.

In simple words, it's a game of matching. However, one thing to note in above image is that EigenFaces algorithm also considers illumination as an important feature. In consequence, lights and shadows are picked up by EigenFaces, which classifies them as representing a face. Face recognition picks up on human things, dominated by shapes and shadows: two eyes, a nose, a mouth. Fig 5.2 Shows the variance of shades extracted from various faces in detail.

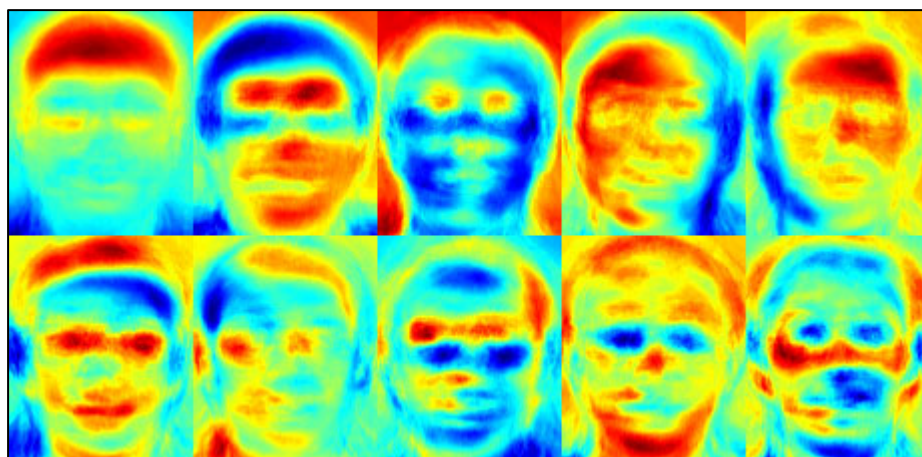


Figure 5.2 : Variance extracted from a list of faces.[10]

5.2.2 Fisherface face recognizer

This algorithm is an improved version of the last one. As system just saw, EigenFaces looks at all the training faces of all the people at once and finds principal components from all of them combined. It doesn't focus on the features that discriminate one individual from another. Instead, it concentrates on the ones that represent all the faces of all the people in the training data, as a whole. Since EigenFaces also finds illumination as a useful component, it will find this variation very relevant for face recognition and may discard the features of the other

people's faces, considering them less useful. In the end, the variance that EigenFaces has extracted represents just one individual's facial features.

System can do it by tuning EigenFaces so that it extracts useful features from the faces of each person separately instead of extracting them from all the faces combined. In this way, even if one person has high illumination changes, it will not affect the other people's features extraction process. Precisely, FisherFaces face recognizer algorithm extracts principal components that differentiate one person from the others. In that sense, an individual's components do not dominate (become more useful) over the others.[10]

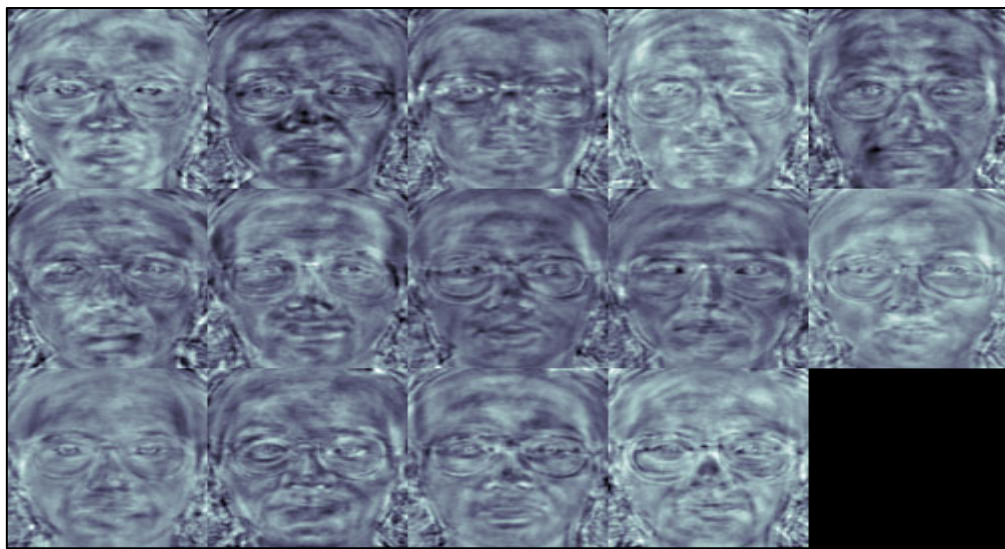


Figure 5.3: principal components using Fisherface algorithm. [10]

Figure 5.3 shows that FisherFaces only prevents features of one person from becoming dominant, but it still considers illumination changes as a useful feature. System know that light variation is not a useful feature to extract as it is not part of the actual face.

5.2.3 Local binary patterns histograms (LBPH) Face Recognizer

The Eigen faces and Fisherface are both affected by light and so cannot guarantee perfect light conditions. LBPH face recognizer is an improvement to overcome this drawback.. Take a 3×3 window and move it across one image. At each move (each local part of the picture), compare the pixel at the centre, with its surrounding pixels. Denote the neighbours with intensity value less than or equal to the centre pixel by 1 and the rest by 0. After user read these 0/1 values under the 3×3 window in a clockwise order, user will have a binary pattern like 11100011 that

is local to a particular area of the picture. When user finish doing this on the whole image, user will have a list of local binary patterns.[10]

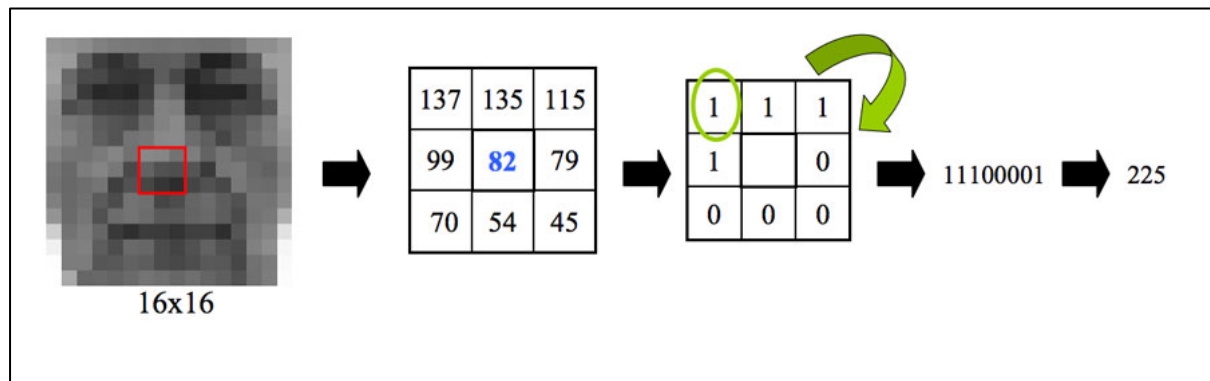


Figure 5.4: LBPH Face recognizer Process [4]

Now, after user get a list of local binary patterns, user convert each one into a decimal number using binary to decimal conversion as shown in figure 5.4 and then user make a histogram of all of those decimal values. Figure 5.5 shows a sample histogram of LBPH.

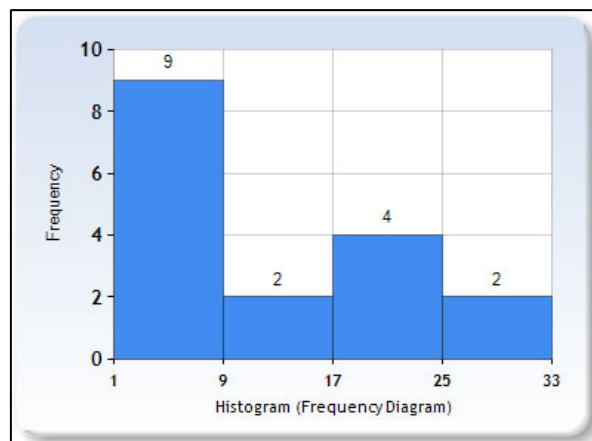


Figure 5.5: LBPH Histogram [10]

In the end, user will have one histogram for each face in the training data set. That means that if there were 100 images in the training data set then LBPH will extract 100 histograms after training and store them for later recognition. Remember, the algorithm also keeps track of which histogram belongs to which person.

Later during recognition, the process is as follows:

1. Feed a new image to the recognizer for face recognition.
2. The recognizer generates a histogram for that new picture.
3. It then compares that histogram with the histograms it already has.

4. Finally, it finds the best match and returns the person label associated with that best match.

Below is a group of faces and their respective local binary patterns images. System see that the LBP faces are not affected by changes in light conditions:

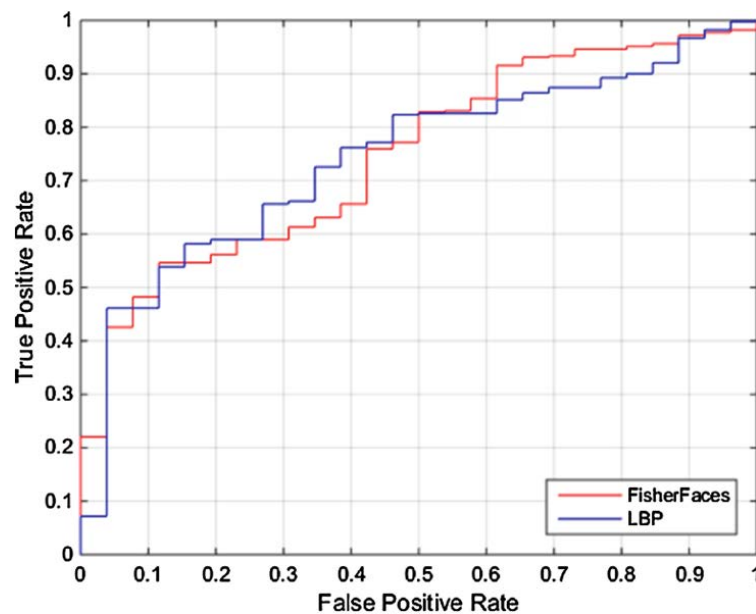


Figure 5.6 ROC curve between Fisherface and LBPH

5.2.4 Required Modules

Import the following modules:

- `cv2`: This is the OpenCV module for Python used for face detection and face recognition.
- `os`: System will use this Python module to read our training directories and file names.
- `numpy`: This module converts Python lists to numpy arrays as OpenCV face recognizer needs them for the face recognition process. [4]

5.2.5 Prepare training data

The premise here is simple: The more images used in training, the better. Being thorough with this principle is important because it is the only way for training a face recognizer so it can learn the different ‘faces’ of the same person; for example: with glasses, without glasses, laughing, sad, happy, crying, with a beard, without a beard, etc. So, our training data consists of total two people with 12 images of each one. All training data is inside the folder: training-data.

This folder contains one subfolder for every individual, named with the format: sLabel (e.g. s1, s2) where the label is the integer assigned to that person. For example, the subfolder called s1 means that it contains images for person 1.

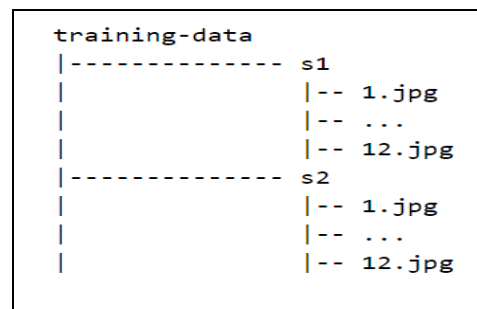


Figure 5.7 Directory structure tree for training data [4]

Figure 5.6 shows the folder test-data contains images that system will use to test our face recognition program after system have trained it successfully. Considering that the OpenCV face recognizer only accepts labels as integers, system need to define a mapping between integer tags and the person’s actual name.[4]

5.3 Data Preparation for Face Recognition

To know which face belongs to which person, OpenCV face recognizer accepts information in a particular format. That means that if there were 100 images in the training data set then LBPH will extract 100 histograms after training and store them for later recognition. It doesn't focus on the features that discriminate one individual from another. Instead, it concentrates on the ones that represent all the faces of all the people in the training data, as a whole. Since EigenFaces also finds illumination as a useful component, it will find this variation very relevant for face recognition and may discard the features of the other people's faces,

considering them less useful. Remember that all the sub folders containing images of a person following the format “sLabel” where Label is an integer representing each person.[4]

In fact, it receives two vectors:

- One is the faces of all the people.
- The second is the integer labels for each face.

For example, if system had two individuals and two images for each one. Then the system will make the file structures as shown in the figure 5.7

PERSON-1	PERSON-2
img1	img1
img2	img2

Figure 5.8 Data preparations for face recognition [4]

After making directories the system will label the images according to the person .The data preparation step will produce following face and label vectors as shown in the figure 5.8

FACES	LABELS
person1_img1_face	1
person1_img2_face	1
person2_img1_face	2
person2_img2_face	2

Figure 5.9 Data preparations for face recognition [4]

In detail, system can further divide this step into the following sub-steps:

1. Read all the sub folders names provided in the folder training-data. In this tutorial; system have folder names:s1, s2.
2. Extract label number. Remember that all the sub folders containing images of a person following the format:sLabel where Label is an integer representing each person. So for example, folder name: s1 means that the person has label 1, s2 means the person's

label is 2, and so on. System will assign the integer extracted in this step to every face detected in the next one.

3. Read all the images of the person, and apply face detection to each one.
4. Add each face to face vectors with the corresponding person label (extracted in above step).[4]

5.4 Microservice architecture

Microservices are a software development technique a variant of the service-oriented architecture (SOA) structural style that arranges an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight. A microservice is not a layer within a monolithic application (example, the web controller, or the backend-for-frontend). Rather it is a self-contained piece of business functionality with clear interfaces, and may, through its own internal components, implement a layered architecture. [5]

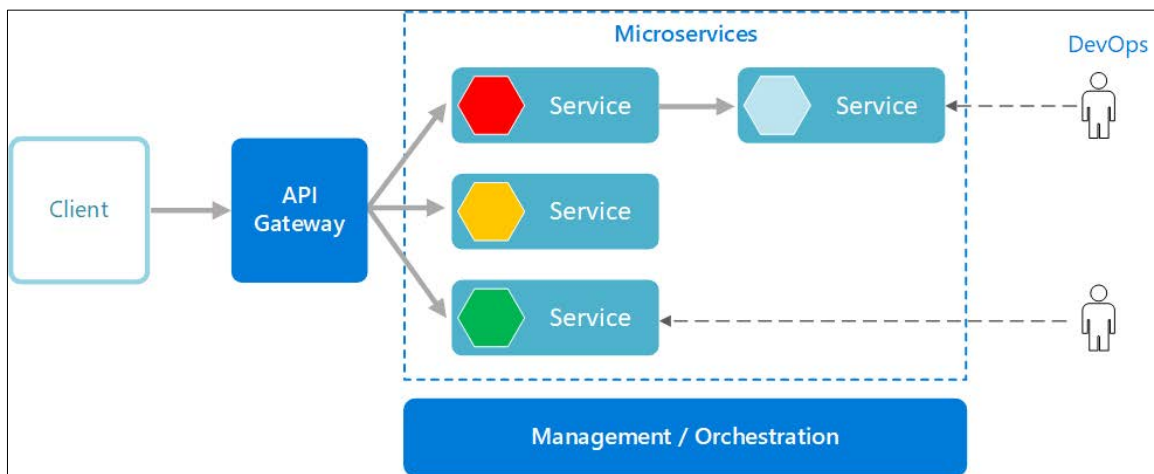


Figure 5.10 Microservice architecture [5]

The main idea behind a microservice architecture is that applications are simpler to build and maintain when broken down into smaller pieces that work seamlessly together. When using microservices, you isolate software functionality into multiple independent modules that are individually responsible for performing precisely defined, standalone tasks. These modules communicate with each other through simple, universally accessible application programming interfaces (APIs) [5].

5.5 Django

Django is a Python-based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

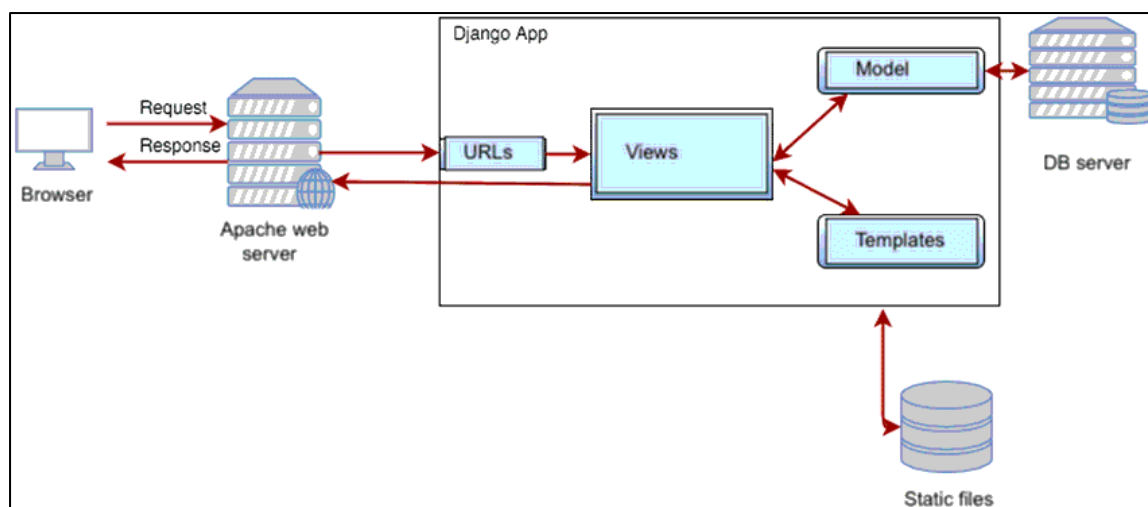


Figure 5.11 Django framework architecture [6]

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Some well-known sites that use Django include the Public Broadcasting Service, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, and Nextdoor. It was used on Pinterest, but later the site moved to a framework built over Flask.

5.6 REST API

REST is acronym for Representational State Transfer. It is architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000 in his famous dissertation. Like any other architectural style, REST also does have its own 6 guiding

constraints which must be satisfied if an interface needs to be referred as RESTful. These principles are listed below.

Guiding Principles of REST

- **Client–server** – By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.
- **Stateless** – Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.
- **Cacheable** – Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.
- **Uniform interface** – By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.
- **Layered system** – The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.
- **Code on demand (optional)** – REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.

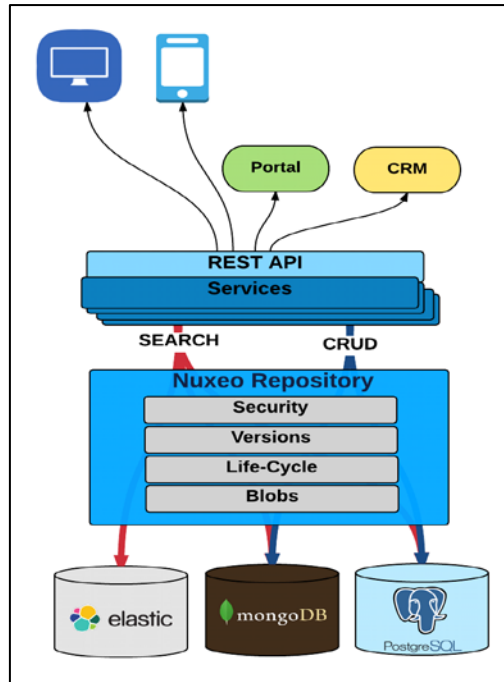


Figure 5.12 REST API architecture [7]

5.7 DJANGO REST FRAMEWORK

Django REST framework is a powerful and flexible toolkit for building Web APIs.

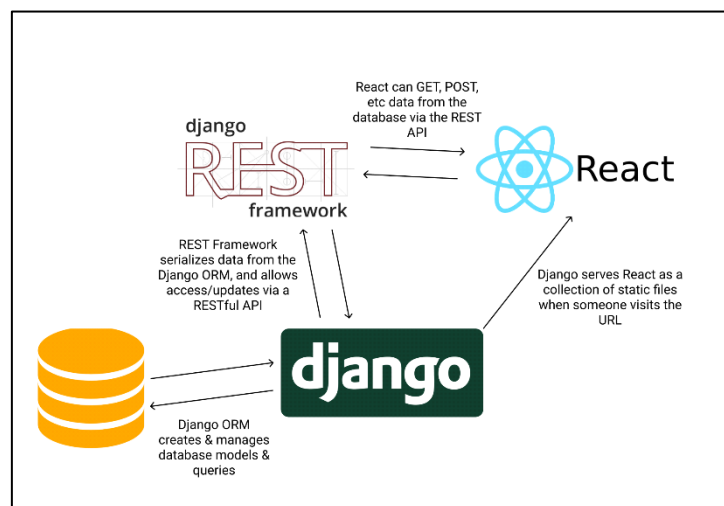


Figure 5.13 Django REST framework [8]

Some reasons you might want to use REST framework:

- The Web browsable API is a huge usability win for your developers.
- Authentication policies including packages for OAuth1a and OAuth2.

- Serialization that supports both ORM and non-ORM data sources.
- Customizable all the way down - just use regular function-based views if you don't need the more powerful features.
- Extensive documentation, and great community support.
- Used and trusted by internationally recognised companies including Mozilla, Red Hat, Heroku, and Eventbrite.

5.8 Bottle server

Bottle is a WSGI micro web-framework for the Python programming language. It is designed to be fast, simple and lightweight, and is distributed as a single file module with no dependencies other than the Python Standard Library. The same module runs with Python 2.7 and 3.x.

It offers request dispatching (routes) with URL parameter support, templates, a built-in web server and adapters for many third-party WSGI/HTTP-server and template engines.

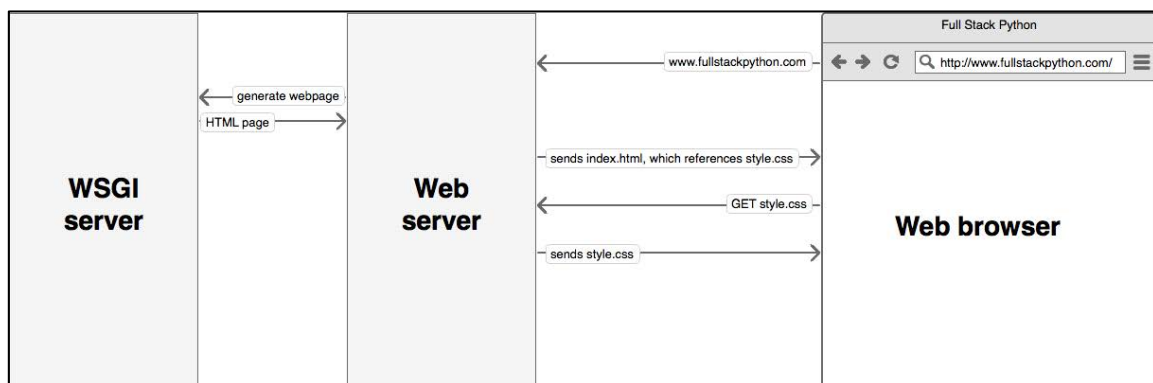


Figure 5.14 Bottle server architecture [9]

It is designed to be lightweight, and to allow development of web applications easily and quickly.

Features

- Single file which runs with both Python 2.7 and 3.x
- Can run as a standalone web server or be used behind ("mounted on") any web server

which supports WSGI

- Built-in template engine called SimpleTemplate Engine
- Support for JSON client data (for REST and JavaScript clients)
- Plugins for popular databases and key/value stores and other features

5.9 Postman http client

Postman is a REST client which makes testing of web services very simple and efficient. It has a user friendly interface which is very intuitive and lets you make HTTP request in no time. It is stuffed with a lot of great features like requests history, authentication, Header presets, which will save you a lot of time and increase your productivity.

Postman is available as a Google Chrome in-browser app.

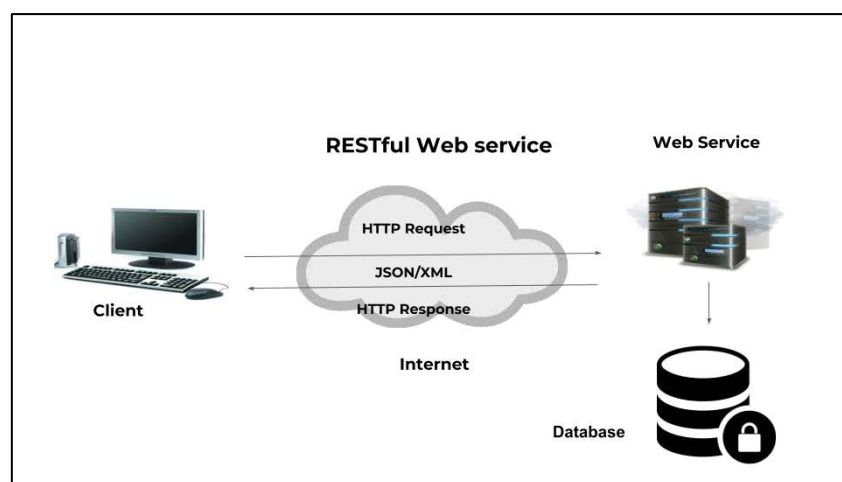


Figure 5.15 Postman http client architecture [11]

Full list of features:

- Powerful request builder - lets you set request type, headers, URL parameters.
- History - saves requests history and lets you group them in collections.
- Authentication - supports three types of authentication: Basic, Digest Auth and OAuth 1.
- Auto complete – dropdowns with suggestions show up for most of the inputs.

- Environments – set environment variables in your requests.
- Save to disk – save HTTP responses to disk.
- Header presets – group headers and their value for later use.

Chapter 6

Results and Discussions

In this chapter the results of the study are presented and discussed with reference to the aim of the study, which was to make a transaction with the help of facial recognition and matching. The system takes images from the user during the registration process and the second to match them during the time of transaction. These aspects were described in the previous chapter that presented the methodology used in the study and implementation for the same.

6.1 Registration and Login

This UI is made in ReactJS. React makes it painless to create interactive UIs. We have designed simple views for each state in our application, and React efficiently updates and renders just the right components when our data changes.

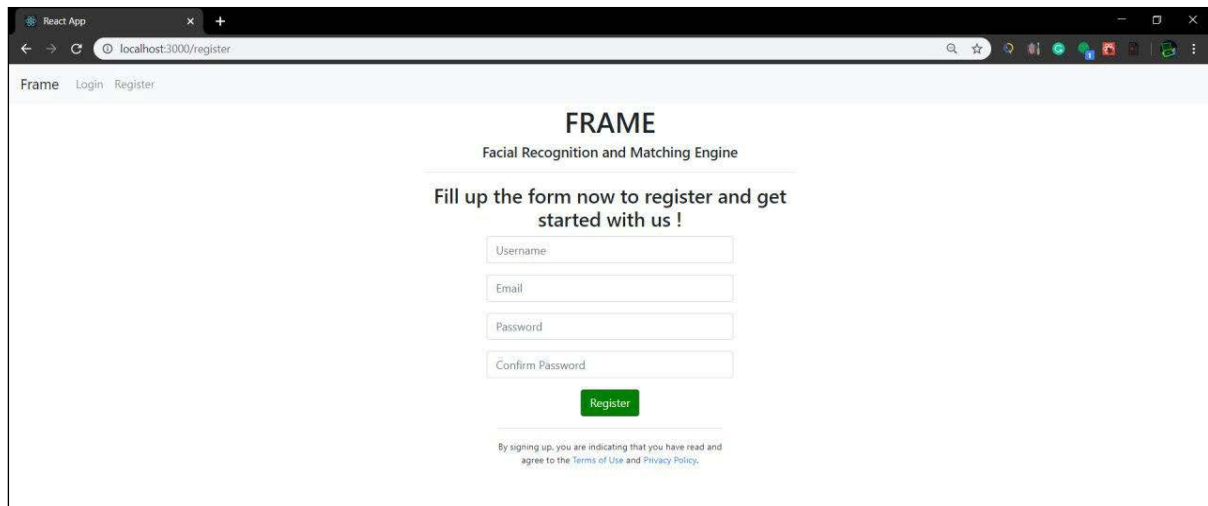
A screenshot of a web browser showing the registration page for 'FRAME Facial Recognition and Matching Engine'. The browser's address bar shows 'localhost:3000/register'. The page has a navigation bar with 'Frame', 'Login', and 'Register' links. The main heading is 'FRAME' with the subtitle 'Facial Recognition and Matching Engine'. Below this, a prompt says 'Fill up the form now to register and get started with us !'. The form contains four input fields: 'Username', 'Email', 'Password', and 'Confirm Password'. A green 'Register' button is positioned below the fields. At the bottom, a small disclaimer states: 'By signing up, you are indicating that you have read and agree to the Terms of Use and Privacy Policy.'

Figure 6.1 Registration Screen

Registration will be done on the above screen, the user is supposed to enter his username, email and password combination. After verification checks such as unique username and similar passwords the registration will be successful if all checks are passed.

Upon successful registration, the user will be redirected to the login page.

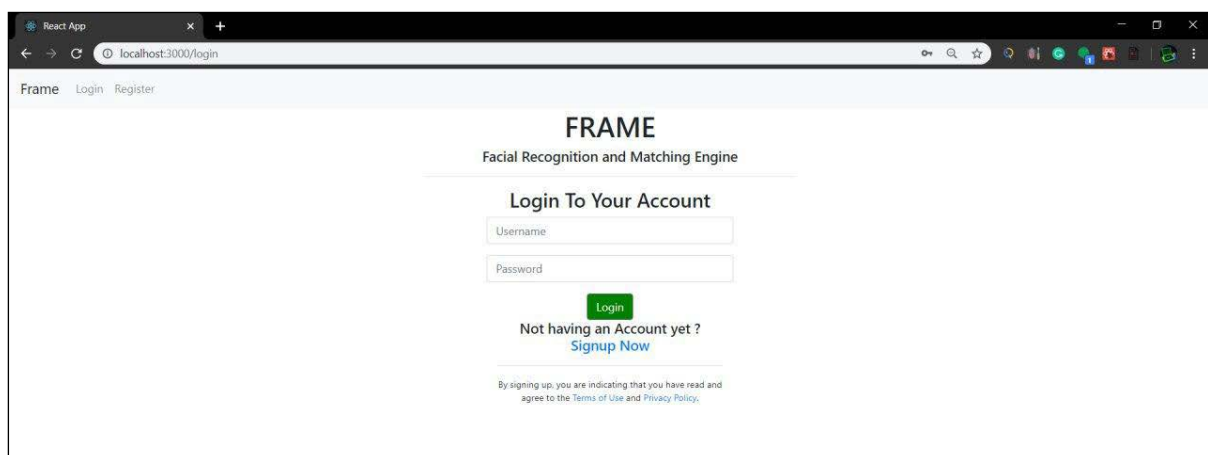
A screenshot of a web browser showing the login page for 'FRAME Facial Recognition and Matching Engine'. The browser's address bar shows 'localhost:3000/login'. The page has a navigation bar with 'Frame', 'Login', and 'Register' links. The main heading is 'FRAME' with the subtitle 'Facial Recognition and Matching Engine'. Below this, the heading 'Login To Your Account' is displayed. The form contains two input fields: 'Username' and 'Password'. A green 'Login' button is positioned below the fields. Below the button, a link says 'Not having an Account yet ? Signup Now'. At the bottom, a small disclaimer states: 'By signing up, you are indicating that you have read and agree to the Terms of Use and Privacy Policy.'

Figure 6.2 Login Screen

The user has to enter his correct combination of username and password which he entered during the time of registration. The information entered in the Registration gets stored in the user info database

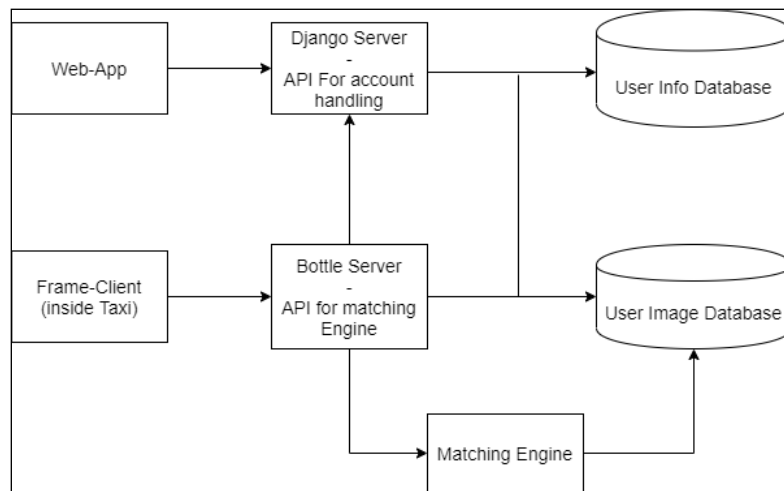


Figure 6.3 Working Architecture

System will be using Django server for all the account handling services such as creating a new user, deleting user, changing password, recovering password , etc. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. Our Main Web-App will be responsible for User Registration. Here 60 user photos will be taken and stored into the User Image Database.

The Django Server will be used for account handling, such as user registration, user login, password updating, data updating, etc. The entire project is based on microservice architecture. Microservices are a software development technique —a variant of the service-oriented architecture (SOA) structural style— that arranges an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight.

Every module will be independent. The bottle server will help the modules communicate with each other. The client will hit the api endpoint of bottle server to access the matching engine inside the cab. The web app will also use bottle server api to store images in the database.

The below figure is the Django admin panel.

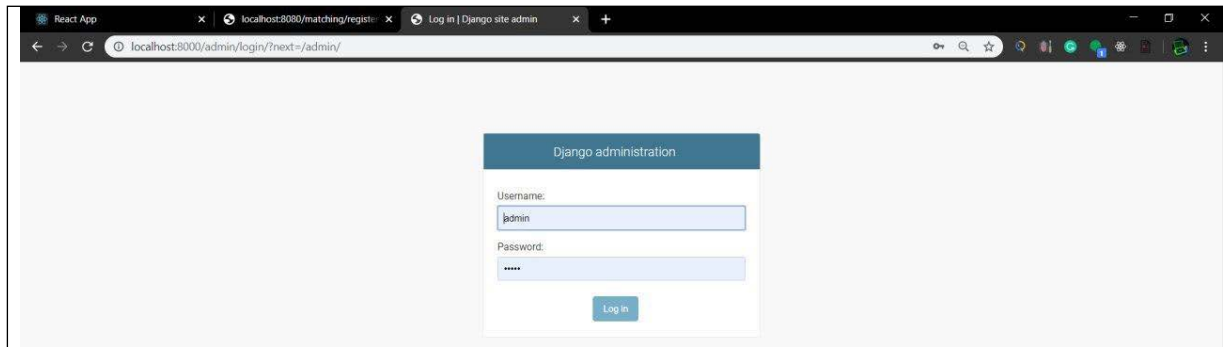


Figure 6.4 Django Administration Login

The above is the login screen for Django Administration. The admin can log in through this.

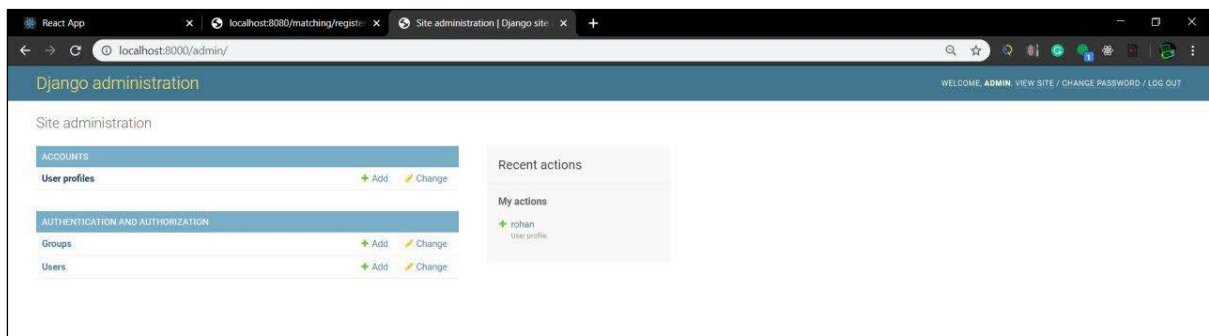


Figure 6.5 Django Admin Dashboard

With the Admin panel the admin has the rights to create, moderate or remove any user.

Upon login the user will be greeted with the below screen from which he can take images for the database. The system takes up to 60 images and stores it in the user image database.

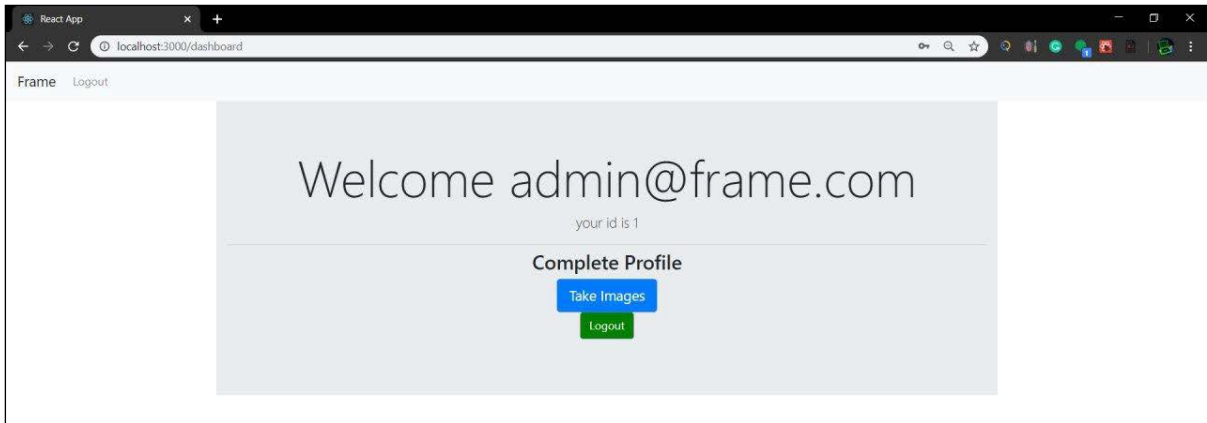


Figure 1.6 Logged in User Dashboard

User will be provided with 2 options, *Take images* and *Logout*. The take images option will open the front camera and take 60 images of the user.

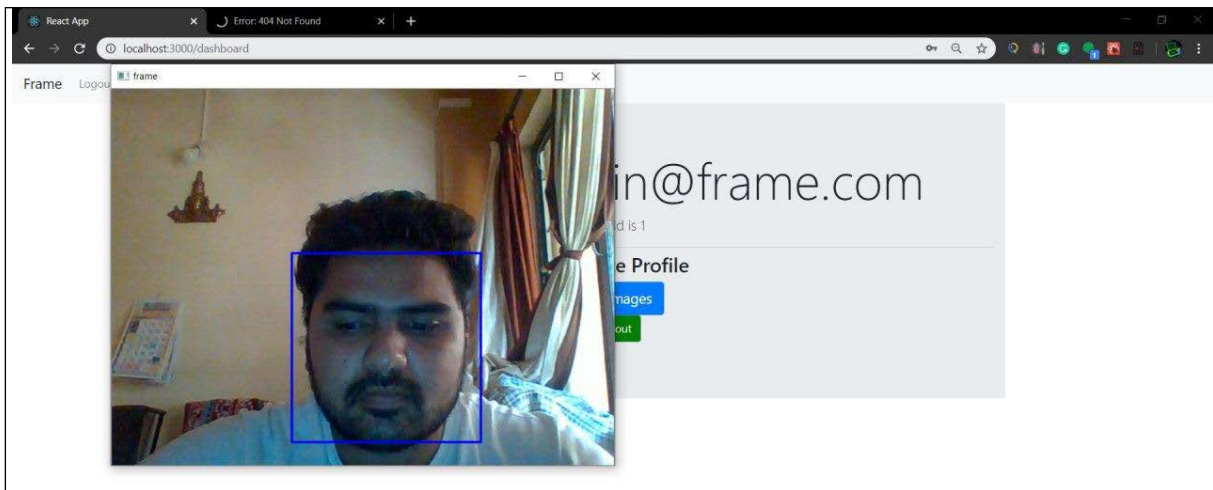


Figure 6.7 Frame Capturing

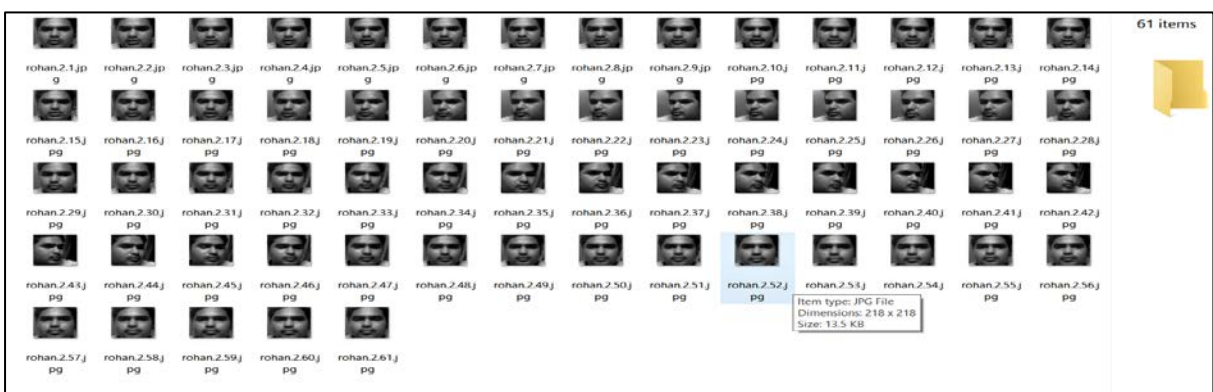


Figure 6.8 Dataset of 60 images stored in the user database

All the images taken will be stored in the user image database via the bottle server.

The Figure no 6.9 shows the dashboard of the user, once face has been successfully scanned and matched. The user will be directed to the below screen where the user see various options. The user will have the option of selecting the source and destination of the journey. Upon selecting, the user will click on calculate fare to get his fare details and make a transaction. After successful transaction the user will get a summary and it will be prompted the details of the journey such as source, destination, distance and lastly fare.

The screenshot shows a web application window titled "Dashboard". Inside, the header says "Welcome to Frame" and "Username :- admin". Below this is a dashed line. The main section is titled "Select the Source and Destination". It contains two input fields, both with the text "virar" and a dropdown arrow. Below these is a "Calculate Fare" button. Another dashed line follows. Below that, it says "Your Fare will be :-". A third dashed line is below that. At the bottom of this section are two buttons: "Start Journey" and "Exit".

Figure 6.9 Trip/fare calculation dashboard

Chapter 7

Conclusion and Future Scope

This section outlines the exploration offering a short review of the considerable number of examinations directed and the outcomes and derivations drawn. It also highlights the contribution of the current study in designing an innovative system to assess and locate the predominant factors in quality assessment process in higher education. Finally, the chapter states limitations and scope of the current study and presents the concluding remarks.

7.1 Conclusion

The venture is a propelled framework which follows microservice design dependent on server customer just as web foundation. All the administrations are interconnected and use REST API for interprocess demand/reaction correspondence. Record the executives is finished utilizing an API worked with Django Rest system, Bootle server goes about as a correspondence unit with the coordinating motor which utilizes opencv LBHP (nearby twofold histogram design) to perceive the pictures which are put away in the client picture database.

7.2 Future Scope

The future Scope of our Facial Recognition and Matching Engine is a versatile equipment pack, that can be arranged and utilized anyplace. This exchanges utilizing facial acknowledgment, in taxis, yet every other assistance, you don't need to convey your telephone/cards/money to a shop, or to travel, or to do any sort of fiscal exchange anyplace. Simply set up a convenient unit which contains a Camera, a touch screen gadget, this framework associated with the web, examine face, load profile, enter pin, and make an exchanges.

References

- [1] R.Prema, Dr. P.Shanmugapriya., et al. "A Face Recognition Techniques for Differentiate Similar Faces and Twin Faces." International conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS 2017).
- [2] Falaye Adeyinka Adesuyi, Osho Oluwafemi, Alabi Isiaq Oludare, Adama Ndako and Amanambu Victor Rick. "Secure Authentication for Mobile Banking Using Facial Recognition." IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727 Volume 10, Issue 3 (Mar. - Apr. 2013), PP 51-59 "www.iosrjournals.org".
- [3] Surekha.R.Gondkar, Saurab. B, Dr. C.S.Mala. "Biometric Face Recognition Payment System". International Journal of Engineering Research & Technology (IJERT). Special Issue – 2018, ISSN: 2278-0181, Volume 6, Issue 13 "Published by www.ijert.org".
- [4] Face Recognition Using OpenCV and Python. [Online] Available: "<https://www.superdatascience.com/blogs/opencv-face-recognition>". [Accessed: September 9,2019].
- [5] Microsrvcies.io 'AboutMicroservices.io' [Online]. Available: "<https://microservices.io/>"
- [6] Django 'Meet Django' [Online]. Available: "<https://www.djangoproject.com/>"
- [7] Rest API 'REST API Tutorial' [Online]. Available: "<https://restfulapi.net/>"
- [8] Django Rest Framework'Django REST framework' [Online]. Available: "<https://www.django-rest-framework.org/>"

- [9] Bottle Server 'Bottle: Python Web Framework' [Online]. Available:"
<https://bottlepy.org/docs/dev/>"
- [10] OpenCV Documentation: 'FaceRecognizer - Face Recognition with OpenCV'
[Online].Available." https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#eigenfaces"
- [11] Postman Http Client 'POSTMAN - HTTP client for testing web services'
[Online].Available: "<https://assist-software.net/downloads/postman-http-client-testing-web-services>"

Appendix

Technologies Used and External Libraries

1. Python 3.x:

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms. The Python interpreter is easily extended with new functions and data types implemented in C or C++. It is also suitable as an extension language for customizable applications.

2. Django

Django is a high-level Python Web framework that encourages rapid development and clean pragmatic design. A Web framework is a set of components that provide a standard way to develop websites fast and easily. Django's primary goal is to ease the creation of complex database-driven websites.

3. OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images

taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.

4. Bottle Server

Bottle is a fast, simple and lightweight WSGI micro web-framework for Python. It is distributed as a single file module and has no dependencies other than the Python Standard Library. The Web Server Gateway Interface (WSGI) is a simple calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language.

5. Sqlite Database

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. The database file format is cross-platform. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be faster than direct filesystem I/O.

6. Postman

Postman is a powerful HTTP client for testing web services. Postman makes it easy to test, develop and document APIs by allowing users to quickly put together both simple and complex HTTP requests. Postman is available as both a Google Chrome Packaged App and a Google Chrome in-browser app. The packaged app version includes advanced features such as OAuth 2.0 support and bulk uploading/importing that are not available in the in-browser version. The in-browser version includes a few features, such as session cookies support, that are not yet available in the packaged app version.

Publication

Sr No.	Paper Title	Author	Publication Details
1	Cab transactions using facial recognition and matching engine	Kshitij Shukla, Rohan Chavan, Saniket Patil	St. John Journal of Engineering Sciences (SJES)

Acknowledgement

First of all we would like to take this opportunity to thank the **Mumbai University** for having projects as a part of the curriculum. We would like to thank our principal sir **Dr. G. V. Mulgund** from **St. John College of Engineering and Management** and would also like to acknowledge our HOD **Mr. Rahul Khokale** for encouraging us to present on the topic “Cab Transactions using Facial Recognition and Matching Engine” on at our department premises for the partial fulfilment of the requirements leading to the award of BE degree.

Most important we would like to acknowledge is our guide **Mr. Vivian Lobo** (Assistant Professor) who supported us throughout this project with utmost cooperation and patience. We are very much thankful to her for sparing her precious time for us and helping us in doing this project. Finally we would like to thank our project coordinators **Mrs. Angelin Florence** and **Mrs. Snehal D’mello** who have helped us in best possible ways in making this project presentable.

Signature
Kshitij Shukla (2152053)

Signature
Rohan Chavan (1152082)

Signature
Saniket Patil (2152014)