# VIGNAN’S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH (Deemed to be UNIVERSITY)

**VADLAMUDI – 522 213, GUNTUR DIST, ANDHRA PRADESH, INDIA.**



**CERTIFICATE**

This is to certify that the Internship Report entitled **“ ”** that is being submitted by **Name(151FAXXXX)** in partial fulfilment for the award of B. Tech degree in Information Technology at Vignan’s Foundation for Science, Technology and Research, Deemed to be University, is a record of bonafide work carried out by them at **“Name of the organization”** under the supervision of **“Guide name at company”** under the co-guidance of the following faculty member of Department of IT.

# Project Guide Head of the Department

**DECLARATION**

I hereby declare that the project entitled “**XXXXXX**” submitted to the **DEPARTMENT OF INFORMATION TECHNOLOGY**. This REPORT is our original work and the project has not formed the basis for the award of any degree, associate-ship, and fellowship or any other similar titles and no part of it has been published or sent for publication at the time of submission.

By xxxxxx(151FA0xxxx) xxxxxx(151FA0xxxx)

Date: 20-April 2019.

# ACKNOWLEDGMENT

Internship program is a golden opportunity for learning and self-development. Consider myself very lucky and honoured to have so many wonderful people lead me through in the completion of this project.

I express my gratitude towards the Managing Director of **XXXXXX,** for permitting me to undertake the **Internship Program** in their extreme company and for the help and cooperation throughout the course of my Internship Program.

We feel it our responsibility to thank “Name of the guide” under whose valuable guidance that the project came out successfully after each stage, and also it is our responsibility to extend our thanks to **Dr. S. Thiruselvan, Department Internship Coordinator,** for extending his support towards the Internship Program in **“name of the organizationXXX”**, in **Hyderabad**.

It is a great pleasure for me to express my sincere thanks to **Prof. K. V. Krishna Kishore, HOD, IT** of VFSTR Deemed to be University**,** for providing me an opportunity to do my Internship Program.

We extend our wholehearted gratitude to all our faculty members of Department of Information Technology who helped us in our academics throughout course.

Finally, we wish to express thanks to our family members for the love and affection overseas and forbearance and cheerful depositions, which are vital for sustaining effort, required for completing this work.

With Sincere regards,

xxxxxx(151FA0xxxx) xxxxxx(151FA0xxxx)

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE OF CONTENTS** | | | |
|  |  | ***Title*** | ***Page No.*** |
|  | **TABLE OF CONTENTS** | | 04 |
| **1** | **INTRODUCTION** | |  |
|  | 1.1 Project Idea | | 12 |
|  | 1.2 | Overview | 13 |
|  | 1.3 | Modules | 14 |
| **2** | **SOFTWARE REQUIREMENT SPECIFICATION** | | |
|  | 2.1 Introduction | | 16 |
|  | 2.2 Functional Requirements | | 17 |
|  | 2.3 Non Functional Requirements | | 19 |
|  | 2.4 | Hardware Requirements | 22 |
|  | 2. 5 Software Requirements | | 22 |
| **3** | **ANALYSIS AND DESIGN PHASE** | |  |
|  | 3.1 Introduction | | 25 |
|  | 3.2 Class Diagram | | 25 |
|  | 3.3 | Object Diagram | 26 |
|  | 3.4 | Activity Diagram | 27 |
|  | 3.5 | Sequence Diagram | 28 |
|  | 3.6Collaboration Diagram | | 29 |
|  | 3.7State chart Diagram | | 30 |
| 3.8 Component Diagram | | | 31 |
|  | 3.9Deployment Diagram | | 31 |
|  | 3.10 Data Design | | 32 |
|  | 3.11 Conclusion | | 35 |

|  |  |
| --- | --- |
| **4 SYSTEM LOW LEVEL DESIGN** |  |
| 4.1 Modules of the project | 37 |
| 4.2 Objectives | 37 |
| 4.3 Project overview | 38 |
| **5 IMPLEMENTATION** |  |
| 5.1 Same Code | 41 |
| 5.2 Screen Captures | 52 |
| 5.3 Firebase Analytics | 59 |
| **6 TESTING** |  |
| 6.1 Software Testing | 63 |
| 6.2 Black Box Testing | 63 |
| 6.3White Box Testing | 63 |
| 6.4 Performance Testing | 63 |
| 6.5Load Testing | 63 |
| 6.6Manual Testing | 63 |
| 6.7Firebase Test Lab | 67 |
| **7 RESULTS AND CHALLENGES** |  |
| 7.1 Results | 70 |
| 7.2Challenges | 70 |
| **8 CONCLUSION AND FUTURE WORK** |  |
| 8.1 Conclusion | 72 |
| 8.2Scope for future work | 72 |
| 8.3Limitations | 72 |
| **BIBILOGRAPHY** | 73 |

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGENO** |
| 2.1 | Use Case Diagram | 21 |
| 3.2 | Class Diagram | 25 |
| 3.3 | Object Diagram | 26 |
| 3.4 | Activity Diagram | 27 |
| 3.5 | Sequence diagram | 28 |
| 3.6 | Collaboration diagram | 29 |
| 3.7 | State Chart diagram | 30 |
| 3.8 | Component diagram | 30 |
| 3.9 | Deployment diagram | 31 |
| 5.2.1 | User login screen | 52 |
| 5.2.2 | Forgot password page | 52 |
| 5.2.3 | User sign up page | 53 |
| 5.2.4 | User home screen | 53 |
| 5.2.5 | Service page | 54 |
| 5.2.6 | Booking page | 54 |
| 5.2.7 | All bookings page | 55 |
| 5.2.8 | Referral page | 55 |
| 5.2.9 | User profile page | 56 |
| 5.2.10 | Service provider login page | 56 |
| 5.2.11 | Service provider signup page | 57 |
| 5.2.12 | Service provider bookings | 57 |
| 5.2.13 | Reference page | 58 |
| 5.2.14 | Service provider profile page | 58 |
| 5.3.1 | Firebase Authentication | 59 |
| 5.3.2 | Password reset mail template | 59 |
| 5.3.3 | Firebase Database | 60 |
| 5.3.4 | Active sessions | 60 |
| 5.3.5 | User engagement | 61 |

|  |  |  |
| --- | --- | --- |
| 6.1 | Test case for empty login | 64 |
| 6.2 | Test case for wrong login | 65 |
| 6.3 | Test case for signup fail | 66 |
| 6.4 | Test case for user signup fail | 67 |
| 6.7 | Firebase test lab clusters | 68 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| Table 3.10.1 | SQLite Database | 32 |
| Table 3.10.2 | List of tables in database | 32 |
| Table 3.10.3 | Admin table | 32 |
| Table 3.10.4 | Users table | 33 |
| Table 3.10.5 | Service providers table | 33 |
| Table 3.10.6 | Bookings table | 34 |
| Table 3.10.7 | Services table | 34 |
| Table 6:1 | Test case for empty login fields | 64 |
| Table 6:2 | Test case for wrong login fields | 64 |
| Table 6:3 | Test case for signup fail | 65 |
| Table 6:4 | Test case for User signup fail | 67 |

**ACRONYMS & ABBREVIATIONS**

|  |  |
| --- | --- |
| * **HTML:** | Hyper Text Markup Language. |
| * **XML:** | Extensible Markup Language. |
| * **IDE:** | Integrated Development Environment |
| * **PHP:** | Hyper Text Preprocessor |
| * **RDBMS:** | Relational Database Management System. |
| * **GUI:** | Graphical User Interface |
| * **HTTP:** | Hyper Text Transfer Protocol |
| * **API:** | Application Programming Interface |
| * **E-R:** | Entity-Relationship |
| * **UML:** | Unified Modeling Language |
| * **OOAD:** | Object-Oriented Analysis & Design. |

**INTERNSHIP SUMMARY**

**Location:**Hyderabad

**Center:** “Name of the organization”

**Duration:**

**Date of start:**

**Date of submission:**

**Title of project:**

**Team Members:**

xxxxxx(151FA0xxxx) xxxxxx(151FA0xxxx) **Name of the guide:**

**Name of Faculty guide:** VFSTR University. **Project Area: DRIVER DROWSINESS**

Driver drowsiness **Abstract:**

* Drowsy driving is one of the major causes of road accidents and death. Hence, detection of driver’s fatigue and its indication is an active research area. Most of the conventional methods are either vehicle based, or behavioural based or physiological based. Few methods are intrusive and distract the driver, some require expensive sensors and data handling. Therefore, in this study, a low cost, real time driver’s drowsiness detection system is developed with acceptable accuracy. In the developed system, a webcam records the video and driver’s face is detected in each frame employing image processing techniques. Facial landmarks on the detected face are pointed and subsequently the eye aspect ratio, mouth opening ratio and nose length ratio are computed and depending on their values, drowsiness is detected based on developed adaptive thresholding. Machine learning algorithms have been implemented as well in an offline manner..

**Signature of Student Signature of Faculty Guide**

**Date: Date:**

# PROFILE OF THE COMPANY

**About ECIL**

**Electronics Corporation of India Limited (ECIL)** is a Government of India Enterprise under the Department of Atomic Energy, established on April 11, 1967 by A. S. Rao at Hyderabad, to create a strong indigenous base in electronics. ECIL is a multi-product, multi- disciplinaryorganization with focus on indigenous Nuclear energy, space and Defense sectors. ECIL also has a strong presence in indigenous Electronic Security, Communications, Networking and e-governance domains. ECIL has committed partnerships with nuclear energy establishments of India, particularly Bhabha Atomic Research Center (BARC), Nuclear Power Corporation of India Limited (NPCIL) and Indira Gandhi Centre for Atomic Research (IGCAR). ECIL also actively supports other strategic sectors such as indigenous Defense (Defense Research and Development Organization(DRDO)), Space (Department of Space (India))Civil Aviation, Information and Broadcasting, Telecommunications, Insurance, Banking, Police and Para-military Forces, Oil and Gas, Power, Space Education, Health, Agriculture, Steel and Coal. ECIL is credited with producing the first indigenous digital computers, TDC 312 and TDC 316, solid state TV, control and instrumentation for nuclear power plants and first earth station antenna of India.

**Company address:**

A.S. Rao Nagar, ECIL, Hyderabad

***CHAPTER - 1***

***INTRODUCTION***

*The chapter gives brief introduction of the project.*

## CHAPTER 1 INTRODUCTION

occurring in road accidents. The truck drivers who drive for continuous long hours (especially at night), bus drivers of long istance route or overnight buses are more susceptible to this problem. Driver drowsiness is an overcast nightmare to passengers in every country. Every year, a large number of injuries and deaths occur due to fatigue related road accidents. Hence, detection of driver’s fatigue and its indication is an active area of research due to its immense practical applicability. The basic drowsiness detection system has three blocks/modules; acquisition system, processing system and warning system. Here, the video of the driver’s frontal face is captured in acquisition system and transferred to the processing block where it is processed online to detect drowsiness. If drowsiness is detected, a warning or alarm is send to the driver

from the warning system. Generally, the methods to detect drowsy drivers are classified in three types; vehicle based, behavioural based and physiological based. In vehicle based method, a number of metrics like steering wheel movement, accelerator or brake pattern, vehicle speed, lateral acceleration, deviations from lane position etc. are monitored continuously. Detection of any abnormal change in these values is considered as driver drowsiness. This is a nonintrusive measurement as the sensors are not attached on the driver. In behavioural based method [1- 7], the visual behavior of the driver i.e., eye blinking, eye closing, yawn, head bending etc. are analyzed to detect drowsiness.

This is also nonintrusive measurement as simple camera is used to detect these features. In physiological based method [8,9], the physiological signals like Electrocardiogram (ECG), Electooculogram (EOG), Electroencephalogram (EEG), heartbeat, pulse rate etc. are monitored and from these metrics, drowsiness or fatigue level is detected. This is intrusive measurement as the sensors are attached on the driver which will distract the driver. Depending on the sensors used in the system, system cost as well as size will increase. However, inclusion of more parameters/features will increase the accuracy of the system to a certain extent. These factors motivate us to develop a low-cost, real time driver’s drowsiness detection system with acceptable accuracy. Hence, we have proposed a webcam based system to detect driver’s fatigue from the face image only using image processing and machine learning techniques to make the system low-cost as well as portable.

***CHAPTER - 2***

***SOFTWARE REQUIREMENT***

***SPECIFICATION***

*Gives the details of platform specifications, Hardware,and Software specifications.*

## CHAPTER 2 REQUIREMENT ANALYSIS

This chapter provides the details of the project’s need based survey, system requirements, Hardware Requirements, Software Requirements, and System Requirements.

**Project Scope :-**

a system in which density of traffic is measured by comparing captured image with real time traffic information against the image of the empty road as reference image is proposed. Each lane will have a minimum amount of green signal duration allocated. According to the percentage of matching allocated traffic light duration can be controlled

**Existing System :-**

Traffic congestion is one of the major modern-day crisis in every big city in the world. Previously different techniques had been proposed, such as infra-red light sensor, induction loop etc. to acquire traffic date which had their fair share of demerits. In recent years, image processing has shown promising outcomes in acquiring real time traffic information using CCTV footage installed along the traffic light. Different approaches have been proposed to glean traffic data. Some of them count total number of pixels [3], some of the work calculate number of vehicles [4-6].These methods have shown promising results in collecting traffic data. However, calculating the number of vehicles may give false results if the intra vehicular spacing is very small (two vehicles close to each other may be counted as one) and it may not count rickshaw or auto-rickshaw as vehicles which are the quotidian means of traffic especially in South-Asian countries.

**Dis-advantages :-**

* Traffic congestion is one of the headack. Here using infra-red light sensor to detect traffic.

**Proposed System :-**

In this paper, a system in which density of traffic is measured by comparing captured image with real time traffic information against the image of the empty road as reference image is proposed. Each lane will have a minimum amount of green signal duration allocated. According to the percentage of matching allocated traffic light duration can be controlled.

**Advantages :-**

* minimum amount of green signal duration allocated. According to the percentage of matching allocated traffic light duration can be controlled

**Functional requirements :-**

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform.

**Non –Functional Requirements :-**

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

**Hardware Requirements :-**

* Operating System supported by

1. Windows 7

2. Windows XP

3 . Windows 8

* Processor – Pentium IV or higher
* RAM -- 256 MB
* Space on Hard Disk -- Minimum 512 MB

**Software Requirements :-**

* For developing the Application

1. Python

2. Django

3. Mysql

4. Mysqlclient

5. WampServer 2.4

* Technologies and Languages used to Develop

-- Python

### HOME PAGE:-

* XML
* JAVA

### REGISTRATIONPAGE:-

* XML
* JAVA

### LOGIN PAGE:-

* XML
* JAVA

### BOOKINGS PAGE

* XML
* JAVA

### REFERRALS PAGE

* XML
* JAVA

### PROFILE PAGE

* XML
* JAVA

### PREFERENCE PAGE

* XML
* JAVA

### SP LOGIN PAGE

* XML
* JAVA

### SP SIGN UP PAGE

* XML
* JAVA

### SP BOOKING PAGE

* XML
* JAVA

***CHAPTER - 3***

***ANALYSIS & DESIGN***

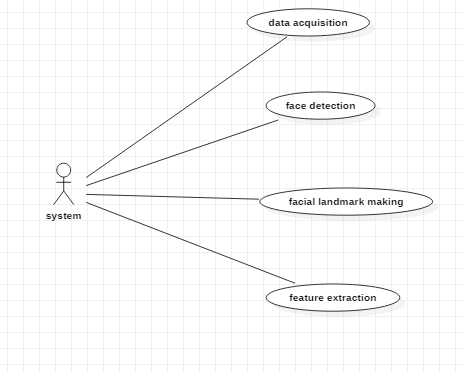
*This chapter gives the details of the system and data design.*

### CHAPTER 3 DESIGN PHASE

**INTRODUCTION**

This chapter provides the design phase of the Application. To design the project, we use the UML diagrams. The Unified Modelling Language (UML) is a general- purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

### 3.1 USE CASE DIAGRAM



**Fig 2.1 Use case Diagram**

The use case diagram is used to represent all the functional use cases that are involved in the project.

The above diagram represents the main two **actors** in the project, they are

* + - User

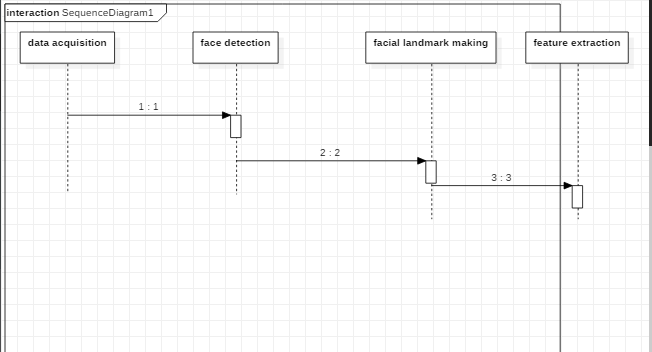
### CLASS DIAGRAM

**Fig 3.2 class diagram**

The above mentioned class diagram represents the Chatbot system workflow model. This diagram has class models with class names as

* + - User
    - Home screen

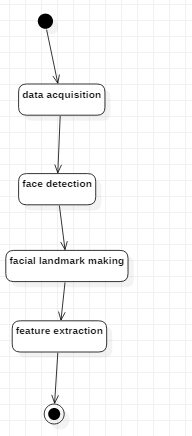
### SEQUENCE DIAGRAM



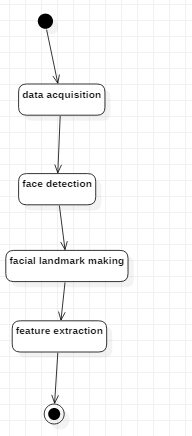
**Fig 3.5 sequence diagram**

The above diagram represents the sequence of flow of actions in the system.

### Activity DIAGRAM



**State chart Diagram:-**

****

## DATA DESIGN

* + 1. **Databases SQLite**

|  |
| --- |
| **Name** |
| Driver Drowsiness |

**Table 3.10.1 SQLite Database**

* + 1. **Tables**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Users | Contains all the registered user details. |
| View Face dedection | All the registered service provider details. |
| Services | Contains all the types of services available. |

**Table 3.10.2 List of Database Tables**

## CONCLUSION

In this paper, a low cost, real time driver drowsiness monitoring system has been proposed based on visual behavior

and machine learning. Here, visual behavior features like eye aspect ratio, mouth opening ratio and nose length ratio are computed from the streaming video, captured by a webcam. An adaptive thresholding technique has been developed to detect driver drowsiness in real time. The developed system works accurately with the generated synthetic data. Subsequently, the feature values are stored and machine learning algorithms have been used for classification. Bayesian classifier, FLDA and SVM have been explored here. It has been observed that FLDA and SVM outperform Bayesian classifier. The sensitivity of FLDA and SVM is 0.896 and 0.956 respectively whereas the specificity is 1 for both. As FLDA and SVM give better accuracy, work will be carried out to implement them in the developed system to do the classification (i.e., drowsiness detection) online. Also, the system will be implemented in hardware to make it portable for car system and pilot study on drivers will be carried out to validate the developed system.

***CHAPTER - 4***

***SYSTEM LOWLEVEL DESIGN***

*This chapter gives an overview of all modules in the project.*

### CHAPTER 4

**SYSTEM LOWLEVEL DESIGN**

This chapter mainly provides the overview on modules of the application, objectives of the project and a detailed project overview.

* 1. **Modules of the Application:**

**Data Acquisition :-**

The video is recorded using webcam (Sony CMU-BR300) and the frames are extracted and processed in a laptop. After extracting the frames, image processing techniques are applied on these 2D images. Presently, synthetic driver data has been generated. The volunteers are asked to look at the webcam with intermittent eye blinking, eye closing, yawning and head bending. The video is captured for 30 minutes duration.

**Face Detection :-**

After extracting the frames, first the human faces are detected. Numerous online face detection algorithms are there. In this study, histogram of oriented gradients (HOG) and linear SVM method [10] is used. In this method, positive samples of descriptors are computed on them. Subsequently, negative samples (samples that do not contain the required object to be detected i.e., human face here) of same size are taken and HOG descriptors are calculated. Usually the number of negative samples is very greater than number of positive samples. After obtaining the features for both the classes, a linear SVM is trained for the classification task. To improve the accuracy of VM, hard negative mining is used. In this method, after training, the classifier is tested on the labeled data and the false positive sample feature values are used again for training

purpose. For the test image, the fixed size window is translated over the image and the classifier computes the output for each window location. Finally, the maximum value output is considered as the detected face and a bounding box is drawn around the face. This non-maximum suppression step removes the redundant and overlapping bounding boxes.

**Facial Landmark marking :-**

After detecting the face, the next task is to find the locations of different facial features like the corners of the eyes and mouth, the tip of the nose and so on. Prior to that, the face images should be normalized in order to reduce the effect of distance from the camera, non-uniform illumination and varying image resolution. Therefore, the face image is resized to a width of 500 pixels and converted to grayscale image. After image normalization, ensemble of regression trees [11] is used to estimate the landmark positions on face from a sparse subset of pixel intensities. In this method, the sum of square error loss is optimized using gradient boosting learning. Different priors are used to find different structures. Using this method, the boundary points of eyes, mouth and the central line of the nose are marked and the number of points for eye, mouth and nose are given in Table I. The facial landmarks are shown in Fig 2. The red points are the detected landmarks for further processing.

**D. Feature Extraction**

After detecting the facial landmarks, the features are computed as described below. Eye aspect ratio (EAR): From the eye corner points, the eye aspect ratio is calculated as the ratio of height and width of the eye as given by

**Classification :-**

After computing all the three features, the next task is to detect drowsiness in the extracted frames. In the beginning, adaptive thresholding is considered for classification. Later, machine learning algorithms are used to classify the data. For computing the threshold values for each feature, it is assumed that initially the driver is in complete awake state. This is called setup phase. In the setup phase, the EAR values for first three hundred (for 10s at 30 fps) frames are recorded. Out 4of these three hundred initial frames containing face, average of 150 maximum values is considered as the hard threshold for EAR. The higher values are considered so that no eye closing instances will be present. If the test value is less than this threshold, then eye closing (i.e., drowsiness) is detected. As the size of eye can vary from person to person, this initial setup for each person will reduce this effect. Similarly, for calculating threshold of MOR, since the mouth may not be open to its maximum in initial frames (setup phase) so the threshold is taken experimentally from the observations. If the test value is greater than this threshold then yawn (i.e., drowsiness) is detected. Head bending feature is used to find the angle made by head with respect to vertical axis in terms of ratio of projected nose lengths. Normally, NLR has values rom 0.9 to 1.1 for normal upright position of head and it increases or decreases when head bends down or up in the state of drowsiness. The average nose length is computed as the average of the nose lengths in the setup phase assuming that no head bending is there. After computing the threshold values, the system is used for testing. The system detects the drowsiness if in a test frame drowsiness is detected for at least one feature. To make this thresholding more realistic, the decision for each frame depends on the last 75 frames. If at least 70 frames (out of those 75) satisfy drowsiness conditions for at least one feature, then the system gives drowsiness detection indication and the alarm.

* 1. **OBJECTIVES OF THE PROJECT**

In this project by monitoring Visual Behaviour of a driver with webcam and machine learning SVM (support vector machine) algorithm we are detecting Drowsiness in a driver. This application will use inbuilt webcam to read pictures of a driver and then using OPENCV SVM algorithm extract facial features from the picture and then check whether driver in picture is blinking his eyes for consecutive 20 frames or yawning mouth then application will alert driver with Drowsiness messages. We are using SVM pre-trained drowsiness model and then using Euclidean distance function we are continuously checking or predicting EYES and MOUTH distance closer to drowsiness, if distance is closer to drowsiness then application will alert driver.

***CHAPTER - 5***

***IMPLEMENTATION***

*The chapter gives the details of the implementation.*

### CHAPTER 5 IMPLEMENTATION

This chapter mainly provides the sample code and implementation of the project.

* 1. **Sample Code**
     1. **XML Code for Bookings page**

*<?***xml version="1.0" encoding="utf-8"***?>*

<**LinearLayoutxmlns:android="**[**http://schemas.android.com/apk/res/android**](http://schemas.android.com/apk/res/android)**" android:layout\_width="match\_parent" android:layout\_height="match\_parent"**

**android:orientation="vertical"**>

<**android.support.v7.widget.LinearLayoutCompat android:layout\_width="match\_parent" android:layout\_height="150dp" android:background="@color/colorTextHint" android:gravity="center" android:orientation="vertical"**>

<**android.support.v7.widget.AppCompatTextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Bookings" android:textSize="20sp"** />

<**android.support.v7.widget.AppCompatTextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginTop="10dp" android:text="@string/text\_hello"** />

<**android.support.v7.widget.AppCompatTextView android:id="@+id/textViewName" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"** />

</**android.support.v7.widget.LinearLayoutCompat**>

<**android.support.v7.widget.AppCompatTextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:paddingBottom="5dp" android:paddingLeft="16dp" android:paddingTop="5dp" android:text="Bookings" android:textColor="@android:color/black"** />

<**ScrollView android:id="@+id/scrollview" android:layout\_width="fill\_parent" android:layout\_height="fill\_parent" android:layout\_below="@+id/spinner1" android:layout\_alignParentBottom="true" android:layout\_alignParentLeft="true"** >

<**LinearLayout android:layout\_width="match\_parent" android:layout\_height="562dp" android:layout\_marginBottom="200dp" android:orientation="vertical"** >

<**ListView**

**android:id="@+id/listView1" android:layout\_width="wrap\_content" android:layout\_height="600dp" android:layout\_marginLeft="0dp"** >

</**ListView**>

</**LinearLayout**>

</**ScrollView**>

</**LinearLayout**>

* + 1. **Java Code for Bookings page**

**package** com.example.spaceimpactor.houser.activities;

**import** java.util.ArrayList; **import** android.os.Bundle; **import** android.app.Activity; **import** android.content.Context; **import** android.content.Intent; **import** android.database.Cursor;

**import** android.database.sqlite.SQLiteDatabase;

**import** android.view.View;

**import** android.view.View.OnClickListener; **import** android.widget.AdapterView; **import** android.widget.ArrayAdapter; **import** android.widget.Button;

**import** android.widget.EditText; **import** android.widget.ImageView; **import** android.widget.ListView; **import** android.widget.Spinner; **import** android.widget.TextView;

**import** android.widget.AdapterView.OnItemClickListener;

**import** android.widget.Toast;

**import** com.example.spaceimpactor.houser.R;

**public class** Bookings **extends** Activity{ Spinner **sp**;

ImageView**out**; TextView**aaa**; SQLiteDatabase**db**; ListView**l**; EditText**t1**;

ArrayList<String>**list1**; ArrayAdapter**adapter**;

Button **sub**;

String **lmb**,**lser**,**lem**;

@Override

**protected void** onCreate(Bundle savedInstanceState) { **super**.onCreate(savedInstanceState); setContentView(R.layout.***activity\_bookings***);

**final** GlobalClassglobalvariabel=(GlobalClass)getApplicationContext(); **aaa**=(TextView)findViewById(R.id.***textViewName***); **aaa**.setText(globalvariabel.GetUsername().toString());

**db**=openOrCreateDatabase(**"ServiceProvider"**, Context.***MODE\_PRIVATE***, **null**); **l** = (ListView) findViewById(R.id.***listView1***);

**final** ArrayList<String> list = **new** ArrayList<String>();

**list1** = **new** ArrayList<String>();

Cursor res = **db**.rawQuery(**"SELECT \* FROM book where uid='"** + **aaa**.getText() + **"'"**, **null**); **if** (res.getCount() != 0) {

**while** (res.moveToNext()) {

list.add(**"Name: "** + res.getString(1) + **"\nMobile No: "** + res.getString(2) + **"\n"** + **"Service Type: "** + res.getString(3) + **"\n"** + **"Email: "** + res.getString(4)+ **"\n"** + **"Booking Date: "** + res.getString(5));

**list1**.add(res.getString(1));

}

}

**adapter** = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_list\_item\_1***, list);

**l**.setAdapter(**adapter**);

}

}

* + 1. **Java Code for Profile Fragment**

**package** com.example.spaceimpactor.houser.fragment;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.database.sqlite.SQLiteDatabase;

**import** android.os.Bundle;

**import** android.support.annotation.NonNull; **import** android.support.v4.app.Fragment; **import** android.text.TextUtils;

**import** android.util.Log;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.support.annotation.Nullable;

**import** android.support.design.widget.Snackbar;

**import** android.support.design.widget.TextInputEditText; **import** android.support.design.widget.TextInputLayout; **import** android.support.v4.widget.NestedScrollView; **import** android.support.v7.app.AppCompatActivity; **import** android.support.v7.widget.AppCompatButton; **import** android.support.v7.widget.AppCompatTextView; **import** android.widget.Button;

**import** android.widget.EditText; **import** android.widget.ProgressBar; **import** android.widget.TextView; **import** android.widget.Toast;

**import** com.google.firebase.auth.FirebaseAuth; **import** com.google.firebase.auth.FirebaseUser; **import** com.google.firebase.database.DataSnapshot; **import** com.google.firebase.database.DatabaseError;

**import** com.google.firebase.database.DatabaseReference; **import** com.google.firebase.database.FirebaseDatabase; **import** com.google.firebase.database.ValueEventListener;

**public class** ProfileFragment**extends** Fragment **implements** View.OnClickListener {

**private** NestedScrollView**nestedScrollView**;

**private** TextInputLayout**textInputLayoutName**; **private** TextInputLayout**textInputLayoutPhone**; **private** TextInputLayout**textInputLayoutEmail**; **private** TextInputLayout**textInputLayoutPassword**;

**private** TextInputLayout**textInputLayoutConfirmPassword**;

**private** TextInputEditText**textInputEditTextName**; **private** TextInputEditText**textInputEditTextPhone**; **private** TextInputEditText**textInputEditTextEmail**; **private** TextInputEditText**textInputEditTextPassword**;

**private** TextInputEditText**textInputEditTextConfirmPassword**;

**private** AppCompatButton**appCompatButtonRegister**;

**private** InputValidation**inputValidation**; **private** DatabaseHelper**databaseHelper**; **private** User **user**;

EditText**id**,**ps**,**em**,**mb**; Button **sub**;

SQLiteDatabase**db**; TextView**aaa**;

**private static final** String ***TAG*** = MainActivity.**class**.getSimpleName();

**private** TextView**txtDetails**;

**private** EditText**inputName**, **inputEmail**; **private** Button **btnSave**;

**private** DatabaseReference**mFirebaseDatabase**;

**private** FirebaseDatabase**mFirebaseInstance**;

**private** String **userId**;

**private** Button **btnChangeEmail**, **btnChangePassword**, **btnSendResetEmail**, **btnRemoveUser**, **changeEmail**, **changePassword**, **sendEmail**, **remove**, **signOut**;

**private** EditText**oldEmail**, **newEmail**, **password**, **newPassword**; **private** ProgressBar**progressBar**;

**private** FirebaseAuth.AuthStateListener**authListener**;

**private** FirebaseAuth**auth**;

@Override

**public** View onCreateView(LayoutInflaterinflater, ViewGroup container, Bundle savedInstanceState) {

*// Inflate the layout for this fragment*

View myView = inflater.inflate(R.layout.***fragment\_profile***, container, **false**);

*// appCompatButtonRegister = (AppCompatButton) myView.findViewById(R.id.appCompatButtonRegister);*

*// appCompatButtonRegister.setOnClickListener(this);*

**return** myView;

}

**private void** createUser(String name, String email) {

*//* ***TODO***

*// In real apps this userId should be fetched*

*// by implementing firebase auth*

**if** (TextUtils.*isEmpty*(**userId**)) {

**userId**= **mFirebaseDatabase**.push().getKey();

}

User user = **new** User();

**mFirebaseDatabase**.child(**userId**).setValue(user); addUserChangeListener();

}

*/\*\**

* *User data change listener*

*\*/*

**private void** addUserChangeListener() {

*// User data change listener* **mFirebaseDatabase**.child(**userId**).addValueEventListener(**new** ValueEventListener() { @Override

**public void** onDataChange(DataSnapshotdataSnapshot) { User user = dataSnapshot.getValue(User.**class**);

*// Check for null*

**if** (user == **null**) {

Log.*e*(***TAG***, **"User data is null!"**); **return**;

}

**inputEmail**.setText(**""**); **inputName**.setText(**""**);

}

@Override

**public void** onCancelled(DatabaseError error) {

*// Failed to read value*

Log.*e*(***TAG***, **"Failed to read user"**, error.toException());

}

});

}

**private void** updateUser(String name, String email) {

*// updating the user via child nodes*

**if** (!TextUtils.*isEmpty*(name))

**mFirebaseDatabase**.child(**userId**).child(**"name"**).setValue(name);

**if** (!TextUtils.*isEmpty*(email))

**mFirebaseDatabase**.child(**userId**).child(**"email"**).setValue(email);

}

*//sign out method* **public void** signOut() { **auth**.signOut();

}

@Override

**public void** onResume() { **super**.onResume(); **progressBar**.setVisibility(View.***GONE***);

}

@Override

**public void** onStart() {

**super**.onStart(); **auth**.addAuthStateListener(**authListener**);

}

@Override

**public void** onStop() {

**super**.onStop();

**if** (**authListener**!= **null**) {

**auth**.removeAuthStateListener(**authListener**);

}

}

**public void** onViewCreated(@NonNullView view, @NullableBundle savedInstanceState) {

**super**.onViewCreated(view, savedInstanceState);

*// initViews();*

*// initObjects();*

**auth**= FirebaseAuth.*getInstance*();

*//get current user*

**final** FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();

**authListener**= **new** FirebaseAuth.AuthStateListener() { @Override

**public void** onAuthStateChanged(@NonNullFirebaseAuthfirebaseAuth) { FirebaseUser user = firebaseAuth.getCurrentUser();

**if** (user == **null**) {

*// user auth state is changed - user is null*

*// launch login activity*

startActivity(**new** Intent(getActivity(), LoginActivity.**class**));

}

}

};

*// btnChangeEmail = (Button) findViewById(R.id.change\_email\_button);*

*//btnChangePassword = (Button) findViewById(R.id.change\_password\_button);* **btnSendResetEmail**= (Button) getView().findViewById(R.id.***sending\_pass\_reset\_button***); **btnRemoveUser**= (Button) getView().findViewById(R.id.***remove\_user\_button***);

*// changeEmail = (Button) getView().findViewById(R.id.changeEmail);*

*// changePassword = (Button) getView().findViewById(R.id.changePass);*

**sendEmail**= (Button) getView().findViewById(R.id.***send***); **remove** = (Button) getView().findViewById(R.id.***remove***); **signOut**= (Button) getView().findViewById(R.id.***sign\_out***);

**oldEmail**= (EditText) getView().findViewById(R.id.***old\_email***); **newEmail**= (EditText) getView().findViewById(R.id.***new\_email***); **password** = (EditText) getView().findViewById(R.id.***password***); **newPassword**= (EditText) getView().findViewById(R.id.***newPassword***);

**oldEmail**.setVisibility(View.***GONE***); **newEmail**.setVisibility(View.***GONE***); **password**.setVisibility(View.***GONE***); **newPassword**.setVisibility(View.***GONE***);

*// changeEmail.setVisibility(View.GONE);*

*// changePassword.setVisibility(View.GONE);* **sendEmail**.setVisibility(View.***GONE***); **remove**.setVisibility(View.***GONE***);

**progressBar**= (ProgressBar) getView().findViewById(R.id.***progressBar***);

**if** (**progressBar**!= **null**) {

**progressBar**.setVisibility(View.***GONE***);

}

**btnSendResetEmail**.setOnClickListener(**new** View.OnClickListener() { @Override

**public void** onClick(View v) { **oldEmail**.setVisibility(View.***VISIBLE***); **newEmail**.setVisibility(View.***GONE***); **password**.setVisibility(View.***GONE***); **newPassword**.setVisibility(View.***GONE***);

*// changeEmail.setVisibility(View.GONE);*

*// changePassword.setVisibility(View.GONE);* **sendEmail**.setVisibility(View.***VISIBLE***); **remove**.setVisibility(View.***GONE***);

}

});

**sendEmail**.setOnClickListener(**new** View.OnClickListener() { @Override

**public void** onClick(View v) {

**progressBar**.setVisibility(View.***VISIBLE***);

**if** (!**oldEmail**.getText().toString().trim().equals(**""**)) {

**auth**.sendPasswordResetEmail(**oldEmail**.getText().toString().trim())

.addOnCompleteListener(**new** OnCompleteListener<Void>() {

@Override

**public void** onComplete(@NonNullTask<Void> task) {

**if** (task.isSuccessful()) {

Toast.*makeText*(getActivity(), **"Reset password email is sent!"**, Toast.***LENGTH\_SHORT***).show();

**progressBar**.setVisibility(View.***GONE***);

} **else** {

Toast.*makeText*(getActivity(), **"Failed to send reset email!"**, Toast.***LENGTH\_SHORT***).show();

**progressBar**.setVisibility(View.***GONE***);

}

}

});

} **else** {

**oldEmail**.setError(**"Enter email"**); **progressBar**.setVisibility(View.***GONE***);

}

}

});

**btnRemoveUser**.setOnClickListener(**new** View.OnClickListener() { @Override

**public void** onClick(View v) {

**progressBar**.setVisibility(View.***VISIBLE***);

**if** (user != **null**) { user.delete()

@Override

.addOnCompleteListener(**new** OnCompleteListener<Void>() {

**public void** onComplete(@NonNullTask<Void> task) {

**if** (task.isSuccessful()) {

Toast.*makeText*(getActivity(), **"Your profile is deleted:( Create a account now!"**, Toast.***LENGTH\_SHORT***).show(); startActivity(**new** Intent(getActivity(), RegisterActivity.**class**));

*// finish();*

**progressBar**.setVisibility(View.***GONE***);

} **else** {

Toast.*makeText*(getActivity(), **"Failed to delete your account!"**, Toast.***LENGTH\_SHORT***).show();

**progressBar**.setVisibility(View.***GONE***);

}

}

});

}

}

});

**signOut**.setOnClickListener(**new** View.OnClickListener() { @Override

**public void** onClick(View v) { signOut();

}

});

*//txtDetails = (TextView) findViewById(R.id.txt\_user);* **inputName**= (EditText) getView().findViewById(R.id.***name***); **inputEmail**= (EditText) getView().findViewById(R.id.***email***); **btnSave**= (Button) getView().findViewById(R.id.***btn\_save***);

**mFirebaseInstance**= FirebaseDatabase.*getInstance*();

*// get reference to 'users' node*

**mFirebaseDatabase**= **mFirebaseInstance**.getReference(**"users"**);

*// store app title to 'app\_title' node*

**mFirebaseInstance**.getReference(**"app\_title"**).setValue(**"Houser"**);

*// app\_title change listener* **mFirebaseInstance**.getReference(**"app\_title"**).addValueEventListener(**new** ValueEventListener() { @Override

**public void** onDataChange(DataSnapshotdataSnapshot) { Log.*e*(***TAG***, **"App title updated"**);

String appTitle = dataSnapshot.getValue(String.**class**);

*// update toolbar title*

*//getSupportActionBar().setTitle(appTitle);*

}

@Override

**public void** onCancelled(DatabaseError error) {

*// Failed to read value*

Log.*e*(***TAG***, **"Failed to read app title value."**, error.toException());

}

});

*// Save / update the user* **btnSave**.setOnClickListener(**new** View.OnClickListener() { @Override

**public void** onClick(View view) {

String name = **inputName**.getText().toString(); String email = **inputEmail**.getText().toString();

*// Check for already existed userId* **if** (TextUtils.*isEmpty*(**userId**)) { createUser(name, email);

} **else** {

updateUser(name, email);

}

}

});

}

@Override

**public void** onClick(View v) { postDataToSQLite();

}

**private void** initViews() {

**nestedScrollView**= (NestedScrollView) getView().findViewById(R.id.***nestedScrollView***);

**textInputLayoutName**= (TextInputLayout) getView().findViewById(R.id.***textInputLayoutName***); **textInputLayoutPhone**= (TextInputLayout) getView().findViewById(R.id.***textInputLayoutPhone***); **textInputLayoutEmail**= (TextInputLayout) getView().findViewById(R.id.***textInputLayoutEmail***); **textInputLayoutPassword**= (TextInputLayout) getView().findViewById(R.id.***textInputLayoutPassword***); **textInputLayoutConfirmPassword**= (TextInputLayout) getView().findViewById(R.id.***textInputLayoutConfirmPassword***);

**textInputEditTextName**= (TextInputEditText) getView().findViewById(R.id.***textInputEditTextName***); **textInputEditTextPhone**= (TextInputEditText) getView().findViewById(R.id.***textInputEditTextPhone***); **textInputEditTextEmail**= (TextInputEditText) getView().findViewById(R.id.***textInputEditTextEmail***); **textInputEditTextPassword**= (TextInputEditText) getView().findViewById(R.id.***textInputEditTextPassword***); **textInputEditTextConfirmPassword**= (TextInputEditText) getView().findViewById(R.id.***textInputEditTextConfirmPassword***);

}

**private void** initObjects() {

**inputValidation**= **new** InputValidation(getActivity()); **databaseHelper**= **new** DatabaseHelper(getActivity()); **user** = **new** User();

}

**private void** postDataToSQLite() {

**if** (!**inputValidation**.isInputEditTextFilled(**textInputEditTextName**, **textInputLayoutName**, getString(R.string.***error\_message\_name***))) {

**return**;

}

**if** (!**inputValidation**.isInputEditTextFilled(**textInputEditTextPhone**, **textInputLayoutPhone**, **"Enter Phone Number"**)) {

**return**;

}

**if** (!**inputValidation**.isInputEditTextFilled(**textInputEditTextEmail**, **textInputLayoutEmail**, getString(R.string.***error\_message\_email***))) {

**return**;

}

**if** (!**inputValidation**.isInputEditTextFilled(**textInputEditTextEmail**, **textInputLayoutEmail**, getString(R.string.***error\_message\_email***))) {

**return**;

}

**if** (!**inputValidation**.isInputEditTextEmail(**textInputEditTextEmail**, **textInputLayoutEmail**, getString(R.string.***error\_message\_email***))) {

**return**;

}

**if** (!**inputValidation**.isInputEditTextPhone(**textInputEditTextPhone**, **textInputLayoutPhone**, getString(R.string.***error\_message\_phone***))) {

**return**;

}

**if** (!**inputValidation**.isInputEditTextFilled(**textInputEditTextPassword**, **textInputLayoutPassword**,

getString(R.string.***error\_message\_password***))) {

**return**;

}

**if** (!**inputValidation**.isInputEditTextMatches(**textInputEditTextPassword**, **textInputEditTextConfirmPassword**, **textInputLayoutConfirmPassword**, getString(R.string.***error\_password\_match***))) {

**return**;

}

**if** (!**databaseHelper**.checkUser(**textInputEditTextEmail**.getText().toString().trim())) {

**user**.setName(**textInputEditTextName**.getText().toString().trim()); **user**.setPhone(**textInputEditTextPhone**.getText().toString().trim()); **user**.setEmail(**textInputEditTextEmail**.getText().toString().trim()); **user**.setPassword(**textInputEditTextPassword**.getText().toString().trim());

**databaseHelper**.updateUser(**user**); emptyInputEditText();

}

}

**private void** emptyInputEditText()

{

**textInputEditTextName**.setText(**null**); **textInputEditTextPhone**.setText(**null**); **textInputEditTextEmail**.setText(**null**); **textInputEditTextPassword**.setText(**null**); **textInputEditTextConfirmPassword**.setText(**null**);

}

}

**5.1.3 XML Code for Profile Fragment**

*<?***xml version="1.0" encoding="utf-8"***?>*

<**android.support.v4.widget.NestedScrollView xmlns:android="**[**http://schemas.android.com/apk/res/android**](http://schemas.android.com/apk/res/android)**" xmlns:tools="**[**http://schemas.android.com/tools**](http://schemas.android.com/tools)**"**

**xmlns:app="**[**http://schemas.android.com/apk/res-auto**](http://schemas.android.com/apk/res-auto)**" android:id="@+id/nestedScrollView" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:paddingBottom="20dp" android:paddingLeft="20dp" android:paddingRight="20dp" android:paddingTop="20dp"**>

<**LinearLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical"**

**android:paddingBottom="@dimen/activity\_vertical\_margin" android:paddingLeft="@dimen/activity\_horizontal\_margin" android:paddingRight="@dimen/activity\_horizontal\_margin" android:paddingTop="0dp" app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"**>

<**TextView android:layout\_width="match\_parent" android:layout\_height="match\_parent"**

**android:text="Update your Profile" android:textColor="@android:color/black" android:textStyle="bold" android:textSize="30sp" android:layout\_marginBottom="70dp" android:gravity="center"**/>

<**EditText android:id="@+id/old\_email" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Confirm Email"**

**android:inputType="textEmailAddress" android:maxLines="1" android:singleLine="true"** />

<**EditText android:id="@+id/new\_email" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="New Email" android:inputType="textEmailAddress" android:maxLines="1" android:singleLine="true"** />

<**EditText android:id="@+id/password" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:focusableInTouchMode="true" android:hint="@string/hint\_password" android:imeActionId="@+id/login"**

**android:imeOptions="actionUnspecified" android:inputType="textPassword" android:maxLines="1" android:singleLine="true" tools:ignore="InvalidImeActionId"** />

<**EditText android:id="@+id/newPassword" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:focusableInTouchMode="true" android:hint="New Password" android:imeActionId="@+id/login" android:imeOptions="actionUnspecified" android:inputType="textPassword" android:maxLines="1" android:singleLine="true" tools:ignore="InvalidImeActionId"** />

<**Button android:id="@+id/send"**

**style="?android:textAppearanceSmall" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginTop="16dp" android:background="@android:color/black" android:text="Send" android:textColor="@android:color/white" android:textStyle="bold" android:layout\_gravity="end"**/>

<**ProgressBar android:id="@+id/progressBar" android:layout\_width="30dp" android:layout\_height="30dp" android:visibility="gone"** />

<**Button android:id="@+id/remove"**

**style="?android:textAppearanceSmall" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginTop="16dp" android:background="@color/colorPrimaryDark" android:text="Remove" android:textColor="@android:color/white" android:textStyle="bold"** />

<**android.support.design.widget.TextInputLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content"**>

<**EditText android:id="@+id/name"**

**android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Name" android:inputType="textCapWords" android:maxLines="1"** />

</**android.support.design.widget.TextInputLayout**>

<**android.support.design.widget.TextInputLayout android:layout\_width="match\_parent" android:layout\_height="wrap\_content"**>

<**EditText android:id="@+id/email"**

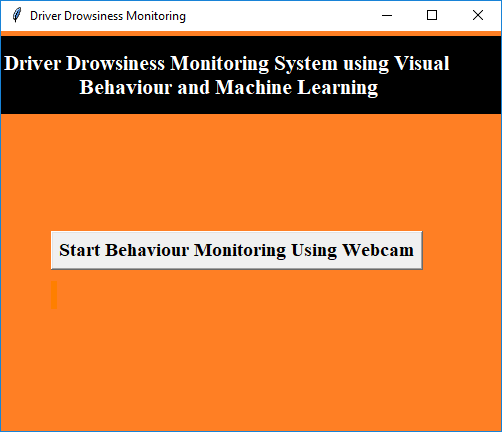
**android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Email" android:inputType="textEmailAddress" android:maxLines="1"** />

</**android.support.design.widget.TextInputLayout**>

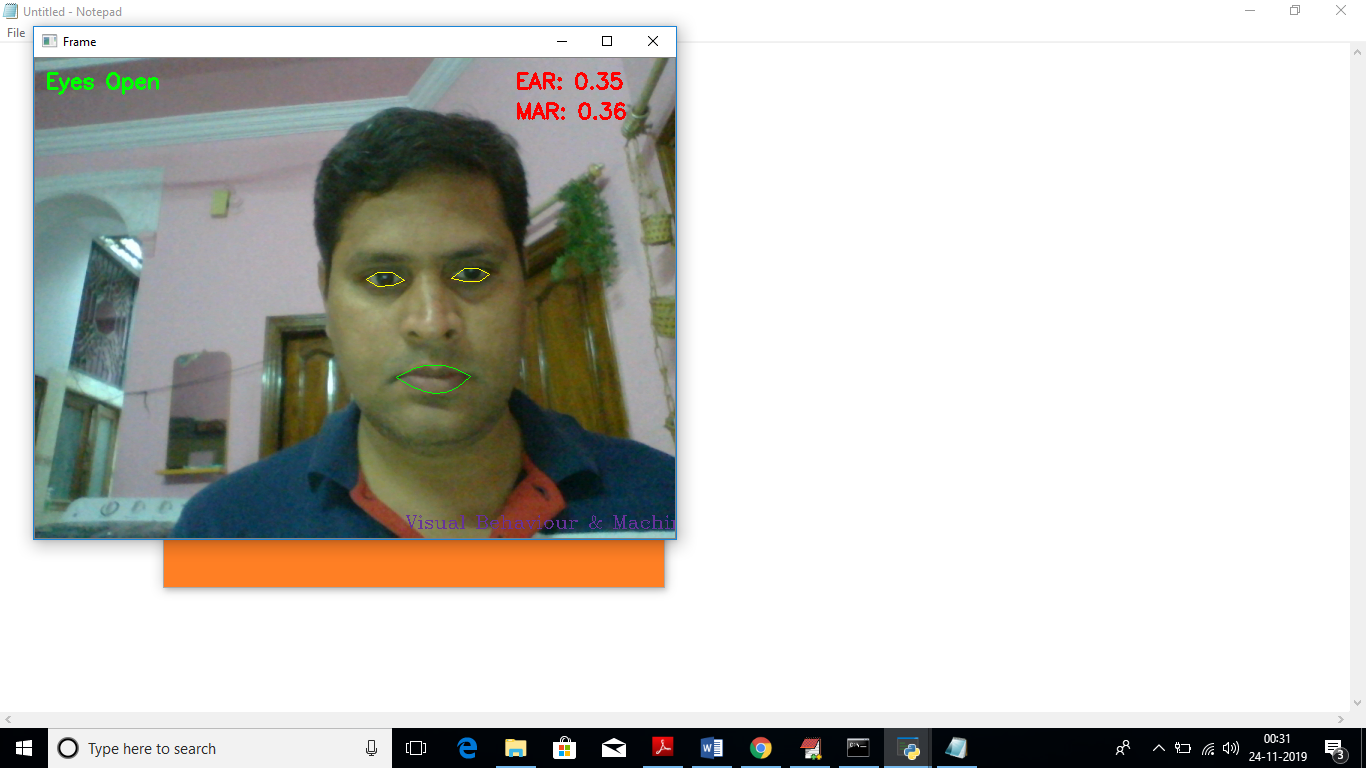
## Screen Captures

* + 1. **User Login Screen:**

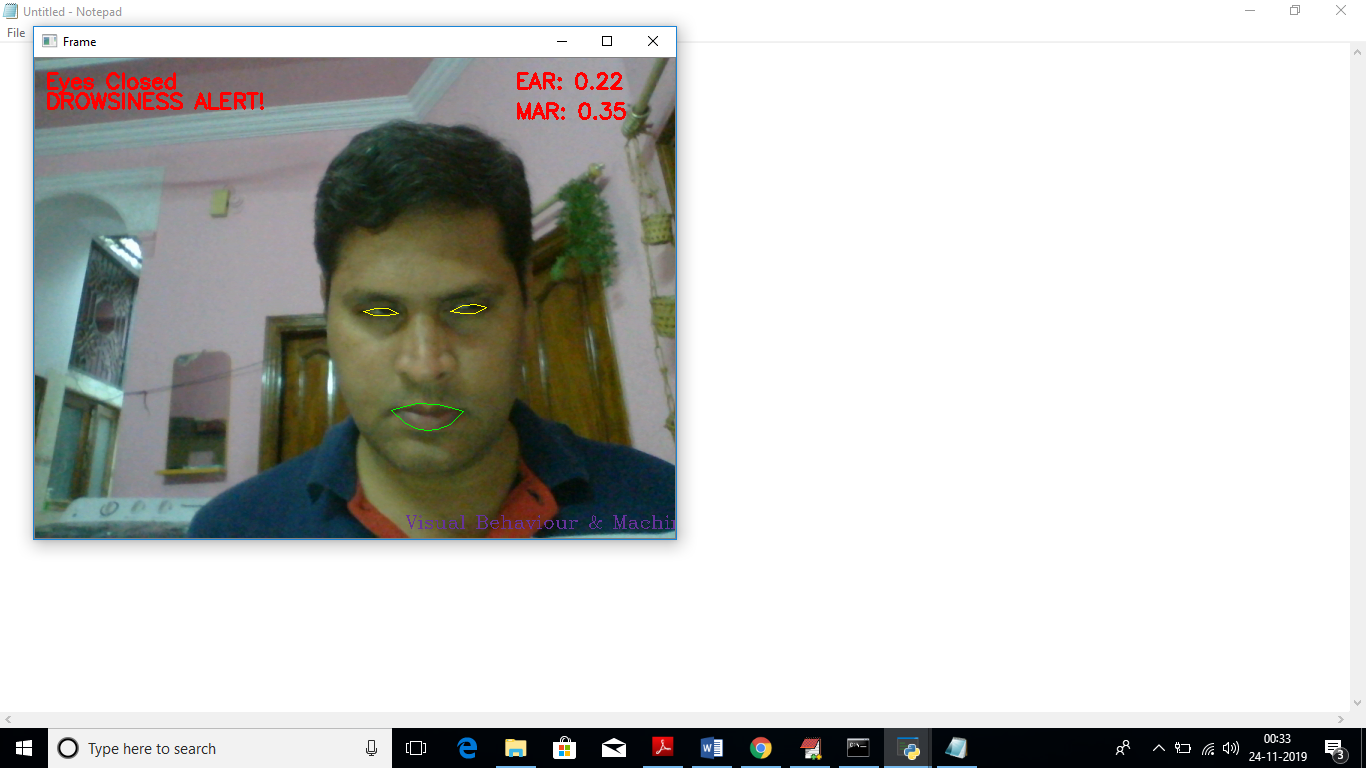
To run this project double click on ‘run.bat’ file to get below screen



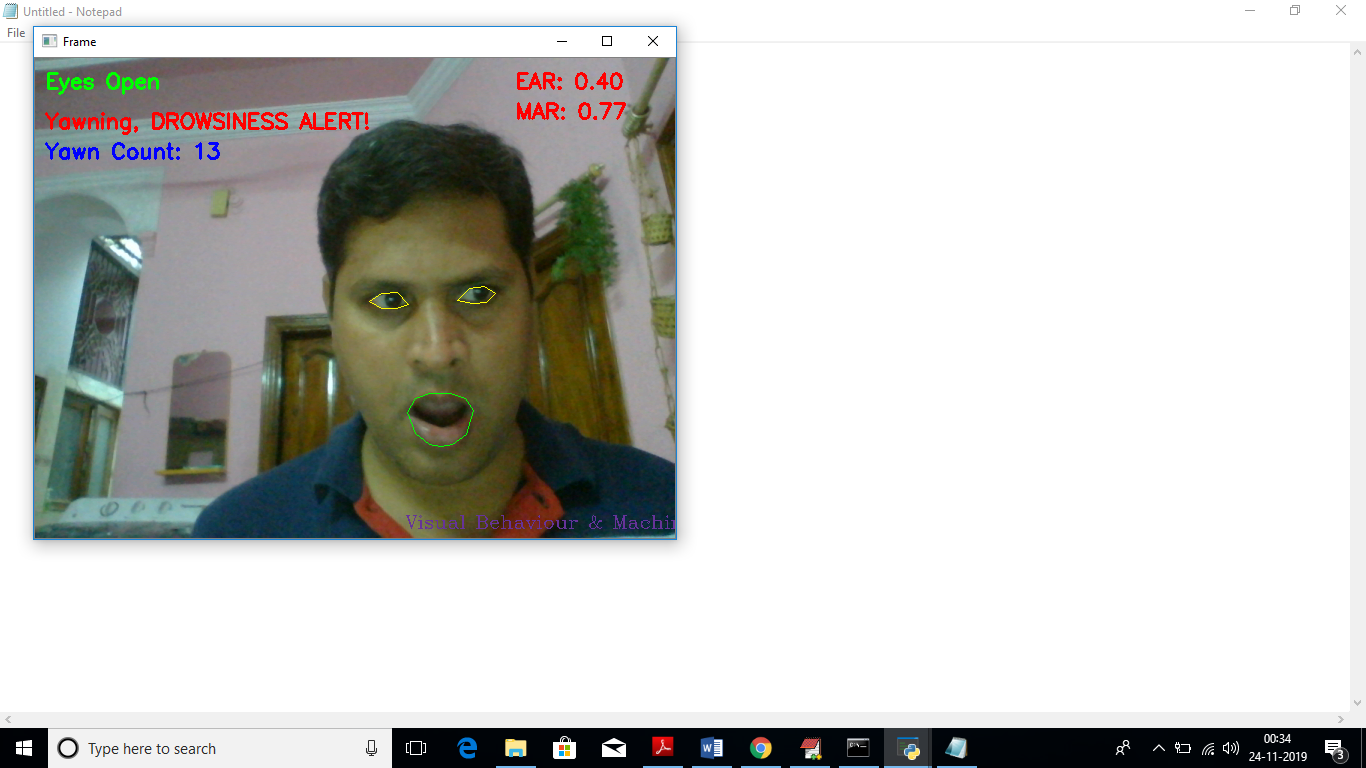
In above screen click on ‘Start Behaviour Monitoring Using Webcam’ button to connect application with webcam, after clicking button will get below screen with webcam streaming

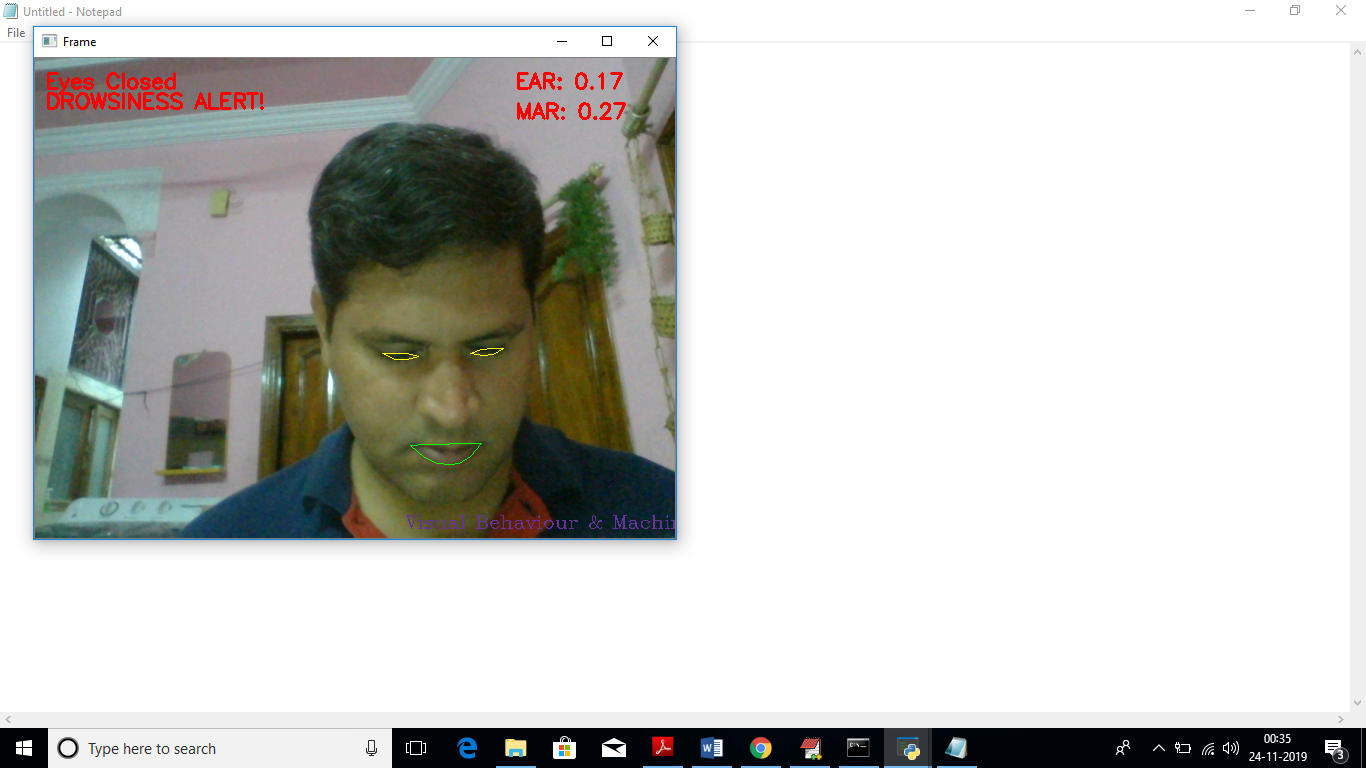


In above screen we can see web cam stream then application monitor all frames to see person eyes are open or not, if closed then will get below message



Similarly if mouth starts yawn then also will get alert message





***CHAPTER - 6***

***TESTING***

*The chapter shows the various test cases.*

# CHAPTER 6

## Software Testing

Software testing is the process of validating and verifying that a software applicationmeets the technical requirements which are involved in its design and development. It is alsoused to uncover any defects/bugs that exist in the application. It assures the quality of thesoftware. There are many types of testing software viz., manual testing, unit testing, black box testing, performance testing, stress testing, regression testing, white box testing etc. Among theseperformance testing and load testing are the most important one for an android application and nextsections deal with some of these types.

## Black box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing,and specification-based testing.

## White box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

## Performance Testing

Performance testing is executed to determine how fast a system or sub-system performsunder a particular workload. It can also serve to validate and verify other quality attributes of thesystem such as scalability, reliability and resource usage.

## Load Testing

Load testing is primarily concerned with testing that can continue to operate underspecific load, whether that is large quantities of data or a large number of users.

## Manual Testing

Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test case for Manual Testing are discussed later in this chapter.

|  |  |
| --- | --- |
| **Test Case 1** | |
| Test Case Name | Empty login fields testing |
| Description | In the login screen if the username and password fields are empty |
| Output | Login fails showing an alert box asking to enter username and  password. |

**Table 6:1 Test Case for Empty Login Fields**

**Figure 6-1 Test Case for Empty Login Fields**

|  |  |
| --- | --- |
| **Test Case 2** | |
| Test Case Name | Wrong login fields testing |
| Description | A unique username and password are set by administrator. On entering wrong username or password gives. |
| Output | Login fails showing an alert box username or password  incorrect. |

**Table 6:2 Test Case for Wrong Login Fields**

**Figure 6-2 Test Case for Wrong Login Fields**

|  |  |
| --- | --- |
| **Test Case 3** | |
| Test Case Name | User Signup Fails. |
| Description | User signup need to provide all data. |
| Output | Signup Fails and an alert message appears asking to enter valid email and name. |

**Table 6:3 Test Case for Signup fail**

***CHAPTER - 7***

***RESULTS &CHALLENGES***

*The chapter describes the results and challenges faced in the project.*

# CHAPTER 7

**RESULTS AND CHALLENGES**

## Results

The current android application is developed using Xml, Java, SQL with Firebase connectivity. It can be used by every individual who are in a need of fulfilling their household services.

At the time of submission of my application was capable of doing the following:

* + Displaying thehome screen with different fragments.
  + Authentication of user by using login screen using Firebase.
  + Home screen to display based on user or service provider.
  + After successful login of user, they can choose the service and book a slot of their particular service provider from the displayed list.
  + Add, update, view, delete the user details.
  + After successful login of service provider, they can view all the bookings that are booked by the users and can attend them one by one.
  + Service provider can also set his preferences to not available, if he’s too busy or many users had already booked him.
  + Service provider has the ability to change their particular radius of location for servicing.
  + He can set up to 10 km radius.
  + Logout and end the session.

## Challenges

* + - Understanding the connections of SQLite Database is a tricky part and confusing when dealing with multiple tables within a database.
    - Making exact orientation API design levels was a difficult task as there are many types of devices like desktop, tablet, mobile with varying screen size and resolutions.
    - Implementing synchronization with Firebasewas a challenging task.
    - Learning different technologies and frameworks with little guidance.

***CHAPTER - 8***

***CONCLUSIONS & FUTURE WORK***

*The chapter gives brief conclusion about the project.*

# CHAPTER 8

# CONCLUSION

## Conclusion

* In this paper, a low cost, real time driver drowsiness monitoring system has been proposed based on visual behavior
* and machine learning. Here, visual behavior features like eye aspect ratio, mouth opening ratio and nose length ratio are computed from the streaming video, captured by a webcam. An adaptive thresholding technique has been developed to detect driver drowsiness in real time. The developed system works accurately with the generated synthetic data. Subsequently, the feature values are stored and machine learning algorithms have been used for classification. Bayesian classifier, FLDA and SVM have been explored here. It has been observed that FLDA and SVM outperform Bayesian classifier. The sensitivity of FLDA and SVM is 0.896 and 0.956 respectively whereas the specificity is 1 for both. As FLDA and SVM give better accuracy, work will be carried out to implement them in the developed system to do the classification (i.e., drowsiness detection) online. Also, the system will be implemented in hardware to make it portable for car system and pilot study on drivers will be carried out to validate the developed system.

## .

## Scope for future work

## 

a system in which density of traffic is measured by comparing captured image with real time traffic information against the image of the empty road as reference image is proposed. Each lane will have a minimum amount of green signal duration allocated. According to the percentage of matching allocated traffic light duration can be controlled

## Limitations

**Intelligent Video-Based Drowsy Driver Detection System under Various Illuminations and Embedded Software Implementation**

An intelligent video-based drowsy driver detection system, which is unaffected by various illuminations, is developed in this study. Even if a driver wears glasses, the proposed system detects the drowsy conditions effectively. By a near-infrared-ray (NIR) camera, the proposed system is divided into two cascaded computational procedures: the driver eyes detection and the drowsy driver detection. The average open/closed eyes detection rates without/with glasses are 94% and 78%, respectively, and the accuracy of the drowsy status detection is up to 91%. By implementing on the FPGA-based embedded platform, the processing speed with the 640×480 format video is up to 16 frames per second (fps) after software optimizations

“**Driver Fatigue Detection based on Eye Tracking and Dynamic Template Matching”**

A vision-based real-time driver fatigue detection system is proposed for driving safely. The driver's face is located, from color images captured in a car, by using the characteristic of skin colors. Then, edge detection is used to locate the regions of eyes. In addition to being used as the dynamic templates for eye tracking in the next frame, the obtained eyes' images are also used for fatigue detection in order to generate some warning alarms for driving safety. The system is tested on a Pentium III 550 CPU with 128 MB RAM. The experiment results seem quite encouraging andpromising. The system can reach 20 frames per second for eye tracking, and the average correct rate for eye location and tracking can achieve 99.1% on four test videos. The correct rate for fatigue detection is l00%, but the average precision rate is 88.9% on the test videos.

**“Monitoring Driver Fatigue using Facial Analysis Techniques”**

In this paper, we describe a non-intrusive vision-based system for the detection of driver fatigue. The system uses a color video camera that points directly rewards the driver's face and monitors the driver's eyes in order to detect micro-sleeps (short periods of sleep). The system deals with skin-color information in order to search for the face in the input space. After segmenting the pixels with skin like color, we perform blob processing in order to determine the exact position of the face. We reduce the search space by analyzing the horizontal gradient map of the face, taking into account the knowledge that eye regions in the face present a great change in the horizontal intensity gradient. In order to find and track the location of the pupil, we use gray scale model matching. We also use the same pattern recognition technique to determine whether the eye is open or closed. If the eyes remain closed for an abnormal period of time (5-6 sec), the system draws the conclusion that the person is falling asleep and issues a warning signal.

**“The Steps of Proposed Drowsiness Detection System Design based on Image Processing in Simulator Driving “**

Drowsiness detection has many implications including reducing roads traffic accidents importance. Using image processing techniques is amongst the new and reliable methods in sleepy face. The present pilot study was done to investigate sleepiness and providing images of drivers' face, employing virtual-reality driving simulator. In order to detecting level of sleepiness according to the signal, information related to 25 drivers was recorded with imaging rate of 10 fps. Moreover, on average 3000 frames was analysed for each driver. The frames were investigated by transforming in grey scale space and based on the Cascade and Viola & Jones techniques and the images characteristics were extracted using Binary and Histogram methods. The MPL neural network was applied for analysing data.70% of information related to each driver were inserted to the network of which 15% for test and 15% for validation. In the last stage the accuracy of 93% of the outputs were evaluated. The intelligent detection and usage of various criteria in long-term time frame are of the advantages of the present study, comparing to other researches. This is helpful in early detection of sleepiness and prevents the irrecoverable losses by alarming

# BIBLIOGRAPHY

Code snippets for

any errors <http://stackoverflow.com/>

Android Development Guide htt[ps://www](http://www.udemy.com/android).[udemy.com/android](http://www.udemy.com/android)

Xml and Layout Guide htt[ps://www](http://www.androidhive.com/).[androidhive.com/](http://www.androidhive.com/)

Connecting to Firebase Docs https://firebase.google.com

Software Testing <http://en.wikipedia.org/wiki/Software_testing>

Manual Testing <http://en.wikipedia.org/wiki/Manual_testing>

Performance Testing <http://en.wikipedia.org/wiki/Software_performance_testing>