



EYES UP

CPIT 499 Final Report

By

Ahlam Abuatallah	1706758
May Alrefaee	1705094
Mshaer alzubaidi	1708032

Department of Information Technology
Faculty of Computing and Information Technology
King Abdulaziz University
Jeddah – Saudi Arabia
[Fall 2021]

DECLARATION by AUTHORS

“I/we certify that this work has not been accepted in substance for any degree, and is not concurrently being submitted for any degree other than that of BS Information Technology being studied at King Abdulaziz University, Jeddah. I/we also declare that this work is the result of my/our own findings and investigations except where otherwise identified by references and that I/we have not plagiarized another’s work”.

[Signature]



Ahlam AbuAtallah

[Signature]



May Alrefaee

[Signature]



Mshaer Alzubidi

Abstract

Getting behind the wheels of a car when tired or drowsy can have serious, even fatal consequences. Drowsy driving caused by lack of sleep and driving at times of the day when you should be sleeping.

Many people who do not get enough hours of sleep are prone to danger because fatigue can slow down your thought process, response time and can affect your vision and health risks.

Every driver fear is getting into an accident and putting their loved ones in a bad situation, missing signs or exits and find themselves in a different city or get distracted from the road which often result in accidents and serious injuries, and the worst fear of all is getting sleepy while driving and start drifting from the lane or being unable to keep your eyes open.

Our application will provide a safer driving experience for all drivers. Eyes-Up will use the facial recognition technology to detect eyes, if users closed their eyes alerts will go off to notify the driver, if there was no responding Eyes-Up can call emergency contact.

Many more features are in the application like give you navigation to the nearest café to buy a coffee and add to do list to remember your stuff while traveling.

This report includes the project proposal, literature review, analysis of collected data and requirement specifications, application implementation and testing and list of recommendations and future work.

TABLE OF CONTENTS

Abstract	III
List of Figures	VIII
List of Tables	X
Chapter I: Introduction.....	1
1.1 Project Background:.....	1
1.2 Problem Definition:	1
1.3 Project Objectives:	1
1.4 Proposed Solution:	2
1.4.1 Architecture Diagram.....	2
1.5 Project Scope	2
1.5.1 Target Users	3
1.5.2 Methodology	3
1.6 Project Feasibility study:.....	4
1.6.1 Completion Criteria	4
1.6.2 Success Criteria.....	5
1.6.3 Deliverables	5
1.7 Project Schedule.....	5
1.8 Project Tools	7
1.8.1 Software	7
1.8.2 Hardware.....	7
1.9 Conclusion	7
Chapter II: Literature Review	8
2.1 Introduction.....	8
2.2 Similar Projects Review:.....	8
2.3 Dareb project Similarities:	9
2.4 Dareb project Differences:	9
2.5 Similar Projects Comparison:	10
2.6 Creativity and originality	11

Chapter III: Analysis and Design.....	12
3.1 Elicitation Techniques	12
3.2 Questionnaire analysis	12
3.3 Requirements Specification	13
3.3.1 Functional requirements:	13
3.3.1.1 User requirements:	13
3.3.1.2 System requirements:.....	13
3.3.2 Non-functional requirements:	14
3.4 Use Case Diagram.....	14
3.4.1 Use Case Elaboration:.....	15
3.5 Traceability Matrix:	19
Chapter IV: Methodology	20
4.1 Methodology	20
4.2 Design Tools	20
4.3 Activity diagram	21
4.4 Entity Relation diagram	22
4.5 Class diagram.....	23
Chapter V: Implementation.....	24
5.1 Introduction.....	24
5.2 Implementation tools and languages.....	24
5.3 Website Implementation	25
5.3.1 Sign-Up	25
5.3.2 Log-In	26
5.3.3 Emergency Contact.....	27
5.3.4 Drowsiness detection	27
5.3.5 Play Alarm	30
5.3.6 Nearest Café Locations	31
5.3.7 Users Profile.....	33
5.4 Database.....	34
5.4.1 Connection	34
5.4.2 Database Functions	34

Store User Function	34
Update user function	35
Update User Password	35
Get User By phone.....	36
Check if user exist.....	36
5.4.3 Tables Creation and data insertion.....	37
Sign Up a new user	37
Log in.....	38
5.5 Difficulties	38
5.6 System Objectives.....	39
5.7 Conclusion	39
Chapter VI: Testing.....	40
6.1 Introduction.....	40
6.2 Testing Description.....	40
6.2.1 Preparations.....	40
6.2.2 Goals	40
6.2.3 Target users:.....	40
6.2.4 Test Environment:.....	40
6.3 Functionalities.....	41
6.4 Testing Methods	42
6.4.1 System Testing.....	42
6.4.2 Usability testing	42
6.4.2.1 Usability testing plan	43
6.5 Testing Result	44
6.5.1 Unit Testing	44
6.5.2 System Testing.....	45
6.5.3 Usability testing	46
6.4 Conclusion	48
Chapter VII: Results and Discussion	49
7.1 Introduction.....	49
7.2 System Interface	49

7.2.1 Sign-up/Login Interfaces.....	49
7.2.2 User Profile interface	52
7.2.3 Nearest Café Location.....	52
7.2.4 Options Page	53
7.2.5 The Face Detecting Interface	54
7.2.6 To Do List Interface.....	55
7.3 Achieved Objectives	57
7.4 Work Limitation	57
7.5 Conclusion	57
Chapter VIII: Conclusion and Future Work	58
8.1 Conclusion	58
8.2 Future Work	58
References.....	59
Appendix A - Questionnaire	60
Appendix B - Work Distribution	64

List of Figures

Figure 1: Architecture Diagram	2
Figure 2: Agile Methodology Diagram.....	3
Figure 3: Pre-Planning Schedule	5
Figure 4: Planning Schedule	5
Figure 5: Analysis Schedule	6
Figure 6: Design Schedule	6
Figure 7: The Future Work	6
Figure 8 User requirements from questionnaire	12
Figure 9 Use case Diagram	14
Figure 10 Agile methodology	20
Figure 11 Activity Diagram	21
Figure 12 Entity Relation Diagram.....	22
Figure 13 Class Diagram	23
Figure 14 Sign-up code	25
Figure 15 username validation.....	25
Figure 16 password validation	25
Figure 17 phone validation	26
Figure 18 address validation	26
Figure 19 birthdate validation.....	26
Figure 20 login form	26
Figure 21 emergency contact.....	27
Figure 22 detecting drowsiness.....	27
Figure 23 set up camera	28
Figure 24 open front camera	28
Figure 25 detecting eyes	29
Figure 26 alarm start	29
Figure 27 first alarm.....	30
Figure 28 second alarm.....	30
Figure 29 third alarm and calling emergency contact.....	31

Figure 30 Nearest cafe location	31
Figure 31 Nearest cafe around user	32
Figure 32 Google Map Permission	32
Figure 33 Log-out	33
Figure 34 Log-out 2	33
Figure 35 Database Connection	34
Figure 36 Store User Function	34
Figure 37 Update user information	35
Figure 38 Update User password	35
Figure 39 Find User by primary key	36
Figure 40 Check if user exist	36
Figure 41 database table	37
Figure 42 Sign-up a new user	37
Figure 43 Log-in users	38
Figure 44 Sign up task results	46
Figure 45 Sign in task results	46
Figure 46 Option menu task results	47
Figure 47 face detecting task results	47
Figure 48 checking nearby cafes task result	48
Figure 49 add to list task result	48
Figure 50 Sign-up/Login interface	49
Figure 51 Sign-up page	50
Figure 52 Login Form Interface	51
Figure 53 Verify Login by phone number	51
Figure 54 User profile	52
Figure 55 Nearest Cafe location	52
Figure 56 Allow emergency contact	53
Figure 57 Choose alarm type	53
Figure 58 Detecting	54
Figure 59 3 rd Alarm	54
Figure 60 2 nd Alarm	54

Figure 61 To do list.....	55
Figure 62 add new note.....	55
Figure 63 note added.....	56
Figure 64 done with note	56
Figure 65 question about age	60
Figure 66 Do you own a car?.....	60
Figure 67 do you get drowsy when you drive for hours?	61
Figure 68 have you ever felt unsafe because you were drowsy?.....	61
Figure 69 the reasons for not pulling over?	62
Figure 70 what do you do to overcome drowsiness?.....	62
Figure 71 drowsiness detection application.....	63
Figure 72 how to warn you from drowsiness.....	63

List of Tables

Table 1: Eyes-Up Project Feasibility Study.....	4
Table 2: Similar Projects to Eyes-Up.....	8
Table 3 Dareb Project	9
Table 4: Comparison between Eyes-up and similar projects.....	10
Table 5 Login use case.....	15
Table 6 Verify login use case.....	15
Table 7 Registration use case.....	16
Table 8 Get destination use case	16
Table 9 Set dialog alarm use case	17
Table 10 Face feature use case.....	17
Table 11 Assistance use case	18
Table 12 Traceability matrix.....	19
Table 13 Difficulties in implementation	39
Table 14 system objectives	39
Table15 The path to accomplish each task	41
Table 16 Test scenario	42

Table 17 Criteria Of Measuring Usability According to Nielsen(2003)	43
Table 18 check password test case.....	44
Table 19 check phone number test case.....	45
Table 20 System testing results.....	45
Table 21 Group work disturpution.....	64

Chapter I: Introduction

1.1 Project Background:

Drowsy driving is a risky activity that cause accidents and injures to thousands of people every year. drivers can be more manageable if provided with enough knowledge and motivations drivers can improve their actions. Our aim is to develop a detailed and strategic plan to avoid drowsy driving accidents and crashes.[1]

1.2 Problem Definition:

The lack of sleep has a major impact on safety, health, and quality of life. While the solution is to get more sleep which is easy, it is difficult to attract public attention when society's values rarely balance enough sleep. Many people find it hard to get the sleep they need in a modern 24/7 day-to-day system with an emphasis on work, increasing transport and expanding channels of communication and entertainment. One of the major problems that sleep involves is the overall attitude towards sleepiness while driving, which requires an effective approach to the problem.[1]

1.3 Project Objectives:

Our project aim is making the driving experience safer for all people by achieving the following objectives:

- **Safety:** Providing a safer driving experience with less accidents.
- **Assistance:** Assist the driver when needed to focus while driving.
- **Interface:** Creating a simple and easy to use interface for easy communication
- **Communication:** Communicate with the driver whenever drowsiness is detected on the driver's face

1.4 Proposed Solution:

The group proposed to implement and develop software application intendent to develop a safer driving experience by using facial and speech-recognition technologies which will alert the user and inform him whenever they fully shut their eyes or yawn more than once.

1.4.1 Architecture Diagram

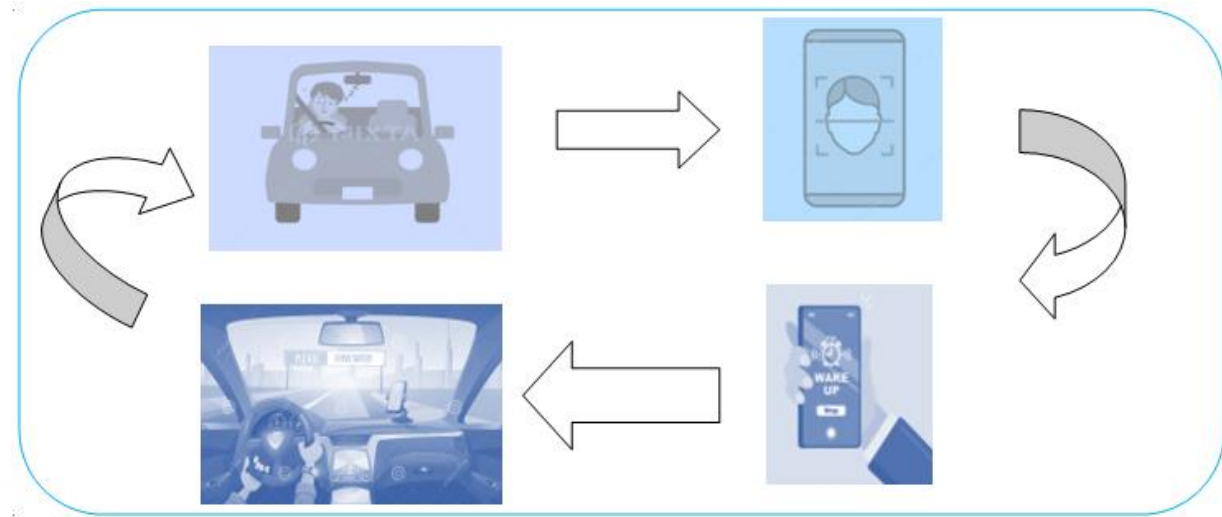


Figure 1: Architecture Diagram

1.5 Project Scope

The Eyes-Up application main purpose is giving the driver a safe driving experience without too many complications. The application main used technology is facial recognition and voice recognition.

The facial recognition is used to detect driver's eyes when close and when yawning. multiple alerts will go off to notify the driver by flashlights, alarm sounds and nudges.

The driver can use the sound recognition technology to get assistant without taking their eyes off the road, by speaking directly to the application to play music or give navigation to nearest café.

The Eyes-Up application can also sync with your smart watch, to get all information needed to make the user experience easier and to get the heart rate for detecting drowsiness.

1.5.1 Target Users

Our main target users are the car drivers especially those who work long hours or traveling by car, they get easily distracted and sleepy and might risk their lives.

1.5.2 Methodology

In our system, we will use the Agile Methodology as a software development Life cycle approach. Since agile methodology promotes continuous iteration of development and testing throughout the software development lifecycle of the project.

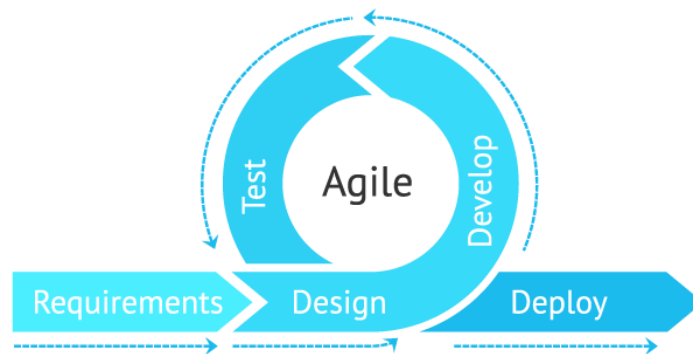


Figure 2: Agile Methodology Diagram

1.6 Project Feasibility study:

Serial no.	Feasibility concern	Status	Reasons
1	Economical	satisfied	The cost of our application is not expensive comparing to other similar applications. In fact, it will be low price. Users will only need to have an Android mobile to install the application.
2	Technical	satisfied	The system will need to detect the facial recognition and the voice recognition. The system needs a lot of work and tools: <ul style="list-style-type: none">• Python• Database• Speech recognition• Face recognition• IOT
3	Operational	satisfied	The application will be a great assistant for the drivers. The system could be used whenever the driver feels drowsy, it will help the driver stay more alert while driving. The user of this application will be able to ask the application to guide them to the nearest café through effortlessly.

Table 1: Eyes-Up Project Feasibility Study

1.6.1 Completion Criteria

If all the functions in the Eyes-Up application are developed and implemented successfully and the drivers who are the main target been able to interact with the application successfully. Then the Eyes-Up application can be considered as completed.

1.6.2 Success Criteria

Eyes-Up can be considered succeed if all the objectives mentioned above have been accomplished and achieved. Especially the safety of the target users which is the main goal of this project.

1.6.3 Deliverables

At the end of our project, we will have an application that will be used by any user who want to be safe while driving and feeling drowsy to help them stay awake and lower the risks.

1.7 Project Schedule

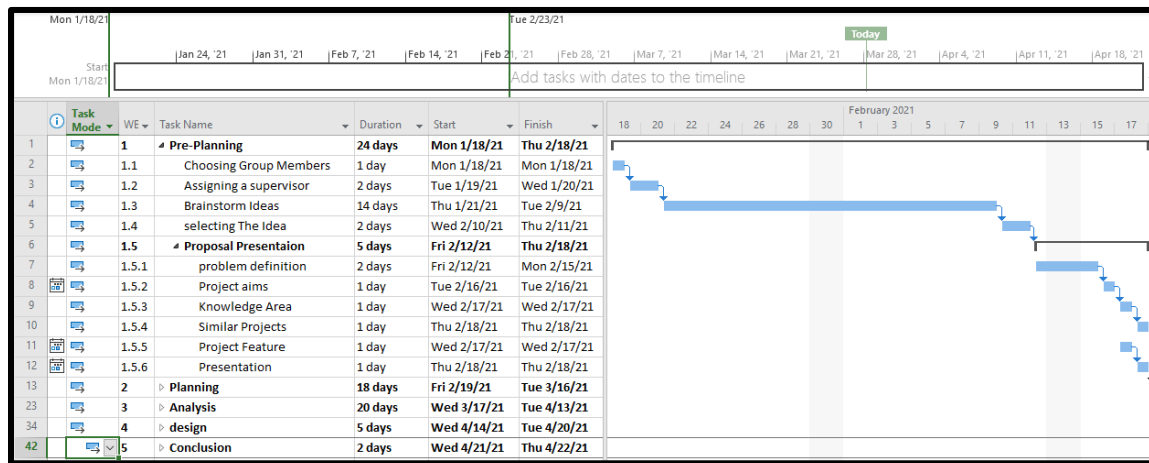


Figure 3: Pre-Planning Schedule

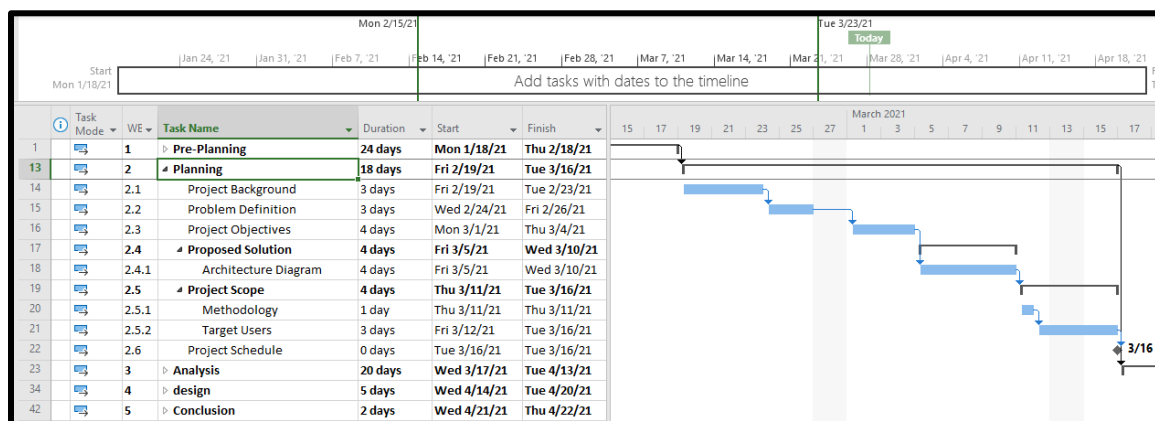


Figure 4: Planning Schedule

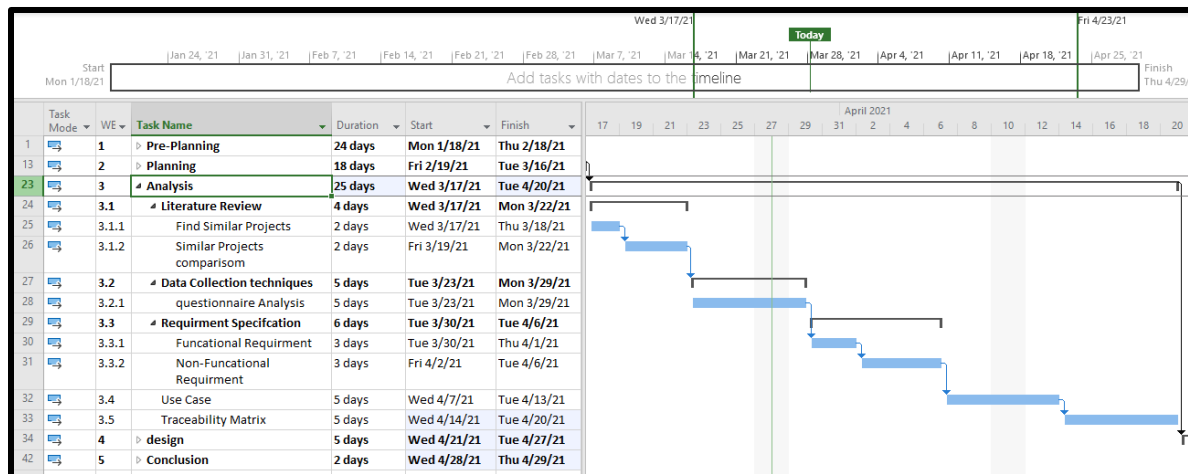


Figure 5: Analysis Schedule

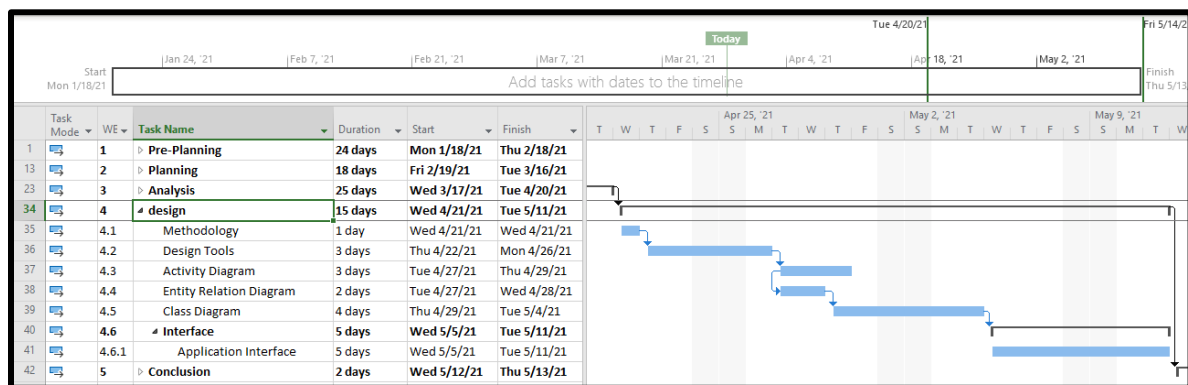


Figure 6: Design Schedule

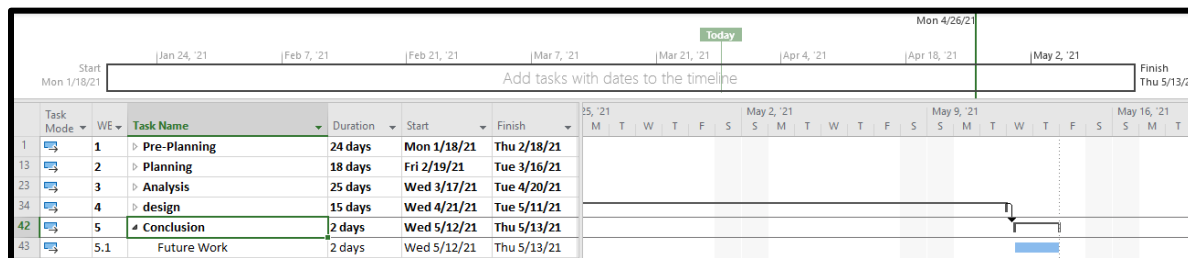


Figure 7: The Future Work

1.8 Project Tools

1.8.1 Software

In our project, these are the software programs that are going to be used:

- Adobe XD tool to design interfaces.
- Python Programming language.
- Microsoft Project to create the project plan.
- Database (mySql).
- Android Studio.
- Creatly tools to draw Use Case, Class diagram, Sequence Diagram, and ERD

1.8.2 Hardware

The main hardware device is a Smartphone that supports the application. And once you open the application the camera will detect the driver's movement and the microphone will detect the driver's voice.

1.9 Conclusion

Finally, to conclude, this chapter introduced the general idea of our project and proposed solution, and included the background, project definition, project objective, project scope, feasibility study and finally we organized the project phases into tasks and identified the tools needed.

Chapter II: Literature Review

2.1 Introduction

In this chapter, we will review four similar systems to our proposed application that aim to help in pandemic situations and provide a brief description of their differences from our project, the features they support and their limitations. Additionally, we are going to highlight our system's functionality and all the technologies that we will be using in developing the application.

2.2 Similar Projects Review:

APPLICATIONS	FEATURE	FLAWS
Awake	Awake is designed to help drivers fight drowsy driving. This app will alert the driver when he/she starts showing signs of drowsy driving and alerts the user with an alarm on their iPhone and/or a combo of vibration and alarm from their Apple Watch.[1]	The alarm is activated once the driver blink for short seconds
Drive awake	Just activate the application and place your phone at a position it can detect your eyes. If your eyes get sleepy on you, the Café Amazon Drive Awake Application wakes you up and routes you to the nearest Café Amazon branch for a cup of coffee before getting back safely on the road. [2]	<ul style="list-style-type: none">• Dose not works in Saudi Arabia.• It only has a few features which is face detection and guidance to nearest cafe.
New nujible	Use Nujible on your Apple Watch to get a subtle, periodic nudge that reminds you to stay awake and focused! Nujible is for those who need a tool to periodically redirect their overactive mind from persistent distraction. It is ideal for people who have trouble concentrating or staying alert, are easily distracted, or have ADHD issues. Use Nujible to help remind you to: - Pay attention in class - Stay alert during meetings	<ul style="list-style-type: none">• Selecting the period is limited only 2 or 4 minutes.• The alarm work for total 20 minutes

Table 2: Similar Projects to Eyes-Up

2.3 Dareb project Similarities:

Dareb is a similar project a group of students did the last year.

EyesUp	Darb
Mobile application	
Face detection	
Audio and visual alerts	
Location detection	
Provide User with different preferences to get rid of drowsiness	

2.4 Dareb project Differences:

EyesUp	Darb
Using mobile Camera for face detection	Using Wireless camera for face detection
Show user the nearest cafe	Show user to the nearest station
Voice recognition to provide user with assistance	Day and night vision detection
Customize dialog alarm	Reports analytical information about driver's accidents
Detect hear rate by connection to a smart watch	Telling real stories about accidents caused by drowsy drivers to increase the awareness
Alarm to notify the user	Providing general guidelines about how to avoid drowsy driving.
Detect hear rate by connection to a smart watch	
Call 911 if an emergency accrues	

Table 3 Dareb Project

2.5 Similar Projects Comparison:

Features / Applications	Awake	Drive Awake	Nujible	Dareb	EyesUP
Mobile application	✓	✓	✓	✓	✓
Face detection	✓	✓		✓	✓
Alarm	✓			✓	✓
nudges			✓	✓	✓
Show nearest cafe		✓			✓
notification				✓	✓
Voice recognition					✓
Customize dialog					✓
Find heart rate					✓
Call 911					✓
Multiple alerts				✓	✓

Table 4: Comparison between Eyes-up and similar projects

2.6 Creativity and originality

Because our application is required there are multiple similar applications in the market. This application differs from similar others since its perfected all three previous applications together and added some new features.

Face recognition in “Awake” was activated even when the driver blinks, our application will be more accurate and will wait for 4 seconds before it is activated, it will also show the driver to the nearest café when drowsiness is detected and since the “Drive Awake” application does not work in Saudi Arabia our application will be perfect.

Our application will also detect the heart rate of the driver in case of any heart attacks it will contact the ambulance. You can customize your own dialog alarm and set it so when the application detects your drowsiness. It will also send you multiple notifications and asks you if you need any other assistance in case you want to hear some radio or music or call someone.

Furthermore, it will be able to contact 911 in case of emergency.

Chapter III: Analysis and Design

3.1 Elicitation Techniques

One main technique was used to collect data and to understand user requirements, so a questionnaire has been distributed.

3.2 Questionnaire analysis

For the data collection process, we used the questionnaire method and sent it to drivers in different ages.

It is divided into three sections. The first part was concerned with determining personal information about the participant's, which stated that 46.5% of the participants are more than 30 years old and more than 60% own a car. The second part was focusing in whether the user has problems with drowsy driving on long road trips which stated that 62% has. The last part attempted to conclude what the user wants from the application and what they can propose to improve the application in long road trips as shown in Figure 8.

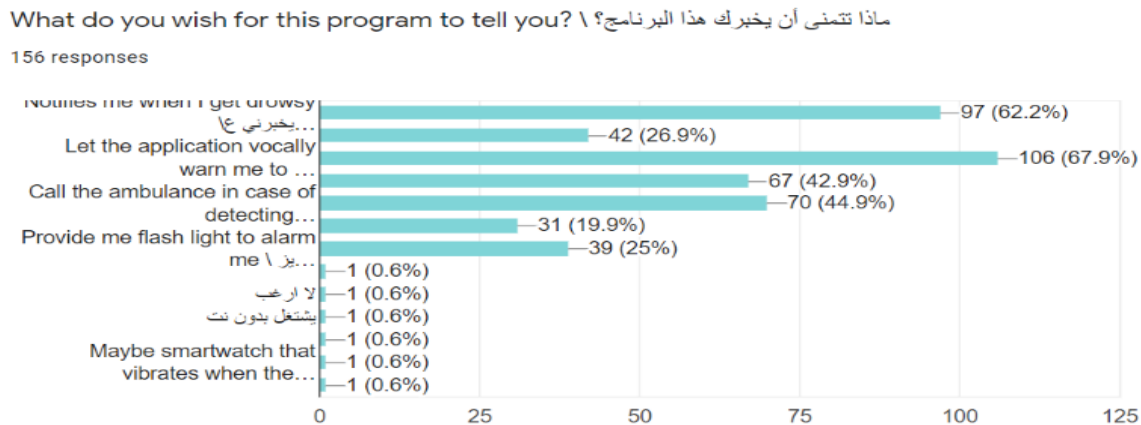


Figure 8 User requirements from questionnaire

3.3 Requirements Specification

3.3.1 Functional requirements:

3.3.1.1 User requirements:

- 1- User shall sign in using their phone number and password.
- 2- User shall sync their smart watch with the system.
- 3- User shall signup using phone number, email, national ID.
- 4- User shall choose his own dialogue alarm type.
- 5- User shall position the phone Infront of them.

3.3.1.2 System requirements:

- 1- System shall present the welcome (sign-in/ sign-up) page to the user.
- 2- System shall verify phone number by text SMS.
- 3- System shall ask the user to give permission to use the camera.
- 4- System shall detect the user face features.
- 5- System shall ask the user to position the phone in front of them.
- 6- System shall be active the whole ride as long as the phone is Infront of the user face.
- 7- System shall detect the drowsiness of the user:
 - a. System shall change the interface theme to a bright blue color.
 - b. System shall provide the nearest café location using google map API.
 - c. System shall ask the user to assist them.
 - d. System shall activate the dialogue alarm.
- 8- System shall collect heart rate data from the smart watch.
 - a. System shall contact 911 when heart rate is abnormal.

3.3.2 Non-functional requirements:

1. Usability: The user interface shall be convenient and easy to use.
2. Availability: The system shall be able to function 24 hours per ride.
3. Performance: The system shall have fast performance and the response time shall be short.
4. Supportability: The system shall support external application to link it with (Google Map and Smart watch).
5. Security: The system shall encrypt the user's data when it stores it.

3.4 Use Case Diagram

The use case is a model that represents the actor's interactions with the system to explain Eyes-Up functionalities and to check the completion of each requirement.

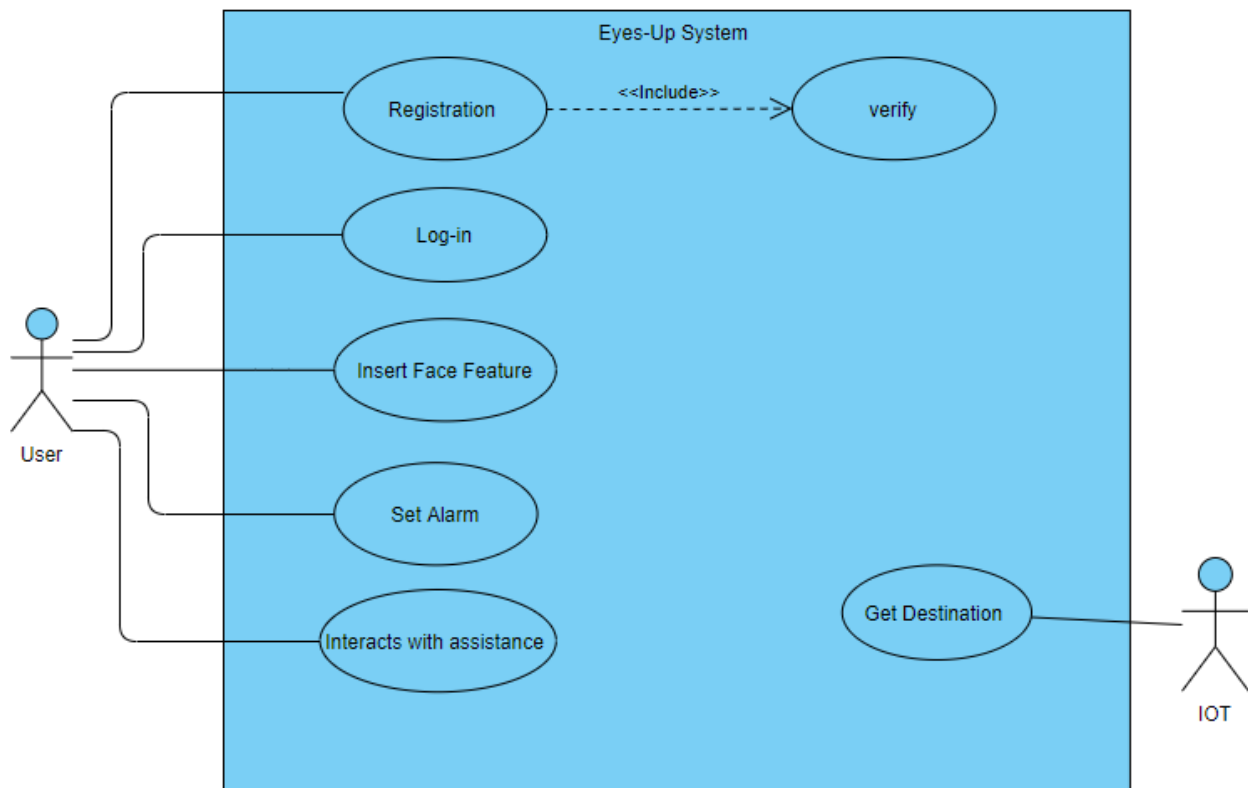


Figure 9 Use case Diagram

3.4.1 Use Case Elaboration:

Use case (UC-1)	login
Primary Actor	user
Use case Overview	Login in into Eyes up
precondition	Registration
Postconditions	User can use the application
Basic flow	1-application display login page. 2-the user enter their registered phone number and password. 3-display the home page.
Exceptions	1 -the user enters invalid phone number or. Password: display error message. 2-user unregistered: Display error message.

Table 5 Login use case.

Use case (UC-2)	Verify Login
Primary Actor	user
Use case Overview	Verify the user account
precondition	login
Postconditions	User can use the application
Basic flow	1- application display login page. 2- the user enters their registered phone number and password. 3- send a verification message. 4- display the home page.
Exceptions	1 – the user does not receive a verification message.

Table 6 Verify login use case.

Use case (UC-3)	Registration
Primary Actor	user
Use case Overview	Register into the application
precondition	Eyes-Up application must be open
Postconditions	User can use the application
Basic flow	<ol style="list-style-type: none"> 1- Press registration button username and fill in information. 2- The application verifies the user's phone number and password. 3- The application verifies the user's phone password. 4- The web application opens home page.
Exceptions	<ol style="list-style-type: none"> 1- Reset phone number if already been used. 2- Reset the password if input is invalid.

Table 7 Registration use case

Use case (UC-4)	Get destination
Primary Actor	user
Use case Overview	System will provide the destination of nearest cafe
precondition	Users get drowsy
Postconditions	Destination of the nearest cafe
Basic flow	<ol style="list-style-type: none"> 1- user will position phone in front of him. 2- System will detect drowsiness. 3- System will provide the nearest cafés location. 4- User will choose the preferred café.
Exceptions	<ol style="list-style-type: none"> 1- The café is not registered in google map. 2- No information is displayed in this area.

Table 8 Get destination use case

Use case (UC-5)	Set Dialog alarm
Primary Actor	user
Use case Overview	Customize alarm dialog.
precondition	Eyes-Up application must be open and logged in.
Postconditions	User can use the application
Basic flow	<ol style="list-style-type: none"> 1- Press dialog alarm icon. 2- Set the specific dialog the user wants the alarm to say. 3- Save the desired alarm. 4- The alarm will be active when the user gets drowsy.
Exceptions	<ol style="list-style-type: none"> 1- Reset the dialog alarm phrase.

Table 9 Set dialog alarm use case

Use case (UC-6)	Face feature
Primary Actor	user
Use case Overview	Get face feature
precondition	Open camera
Postconditions	User can set his feature in the application
Basic flow	<ol style="list-style-type: none"> 1- User open the camera. 2- User take picture with his eyes closed and opened. 3- application check if picture is valid. 4- application create the picture. 5- the application inserts the picture in the database.
Exceptions	<ol style="list-style-type: none"> 1- The application check if the picture is invalid then the user must retake the picture.

Table 10 Face feature use case.

Use case (UC-7)	Assistance
Primary Actor	user
Use case Overview	Get assistance
precondition	The phone must be set Infront of the user
Postconditions	User will get assistance from the application
Basic flow	<ol style="list-style-type: none"> 1- User open the application. 2- Application start detecting face by camera. 3- Application will insert the face feature as a picture. 4- The application will search the picture in the database. 5- the application will get the face feature information feature from the database. 6- If the face feature is drowsy then application will ask the user for assistance 7- The user will accept assistance and the application will help.
Exceptions	<ol style="list-style-type: none"> 1- If the face feature is not drowsy the application will continue detecting face feature. 2- The user will not accept assistance.

Table 11 Assistance use case

3.5 Traceability Matrix:

Traceability Matrix maps the requirements of Eyes-Up system with each use case.

Functional	Use case						
Functional Requirement	UC - 1	UC - 2	UC - 3	UC - 4	UC - 5	UC-6	UC-7
User Requirement							
1	✓						
2							
3		✓					
4			✓				
5				✓	✓	✓	✓
System Requirement							
1	✓						
2		✓			✓		
3						✓	
4			✓	✓		✓	✓
5			✓	✓		✓	✓
6			✓	✓		✓	✓
7						✓	✓
7.a							✓
7.b				✓			✓
7.c			✓				✓
7.d			✓				✓
8							✓
8.a							

Table 12 Traceability matrix

Chapter IV: Methodology

4.1 Methodology

Since our requirements and system functionalities appeared to require continuous updates as the situation changes, we chose Agile as our methodology in this project to reply to the changeover following our plan.

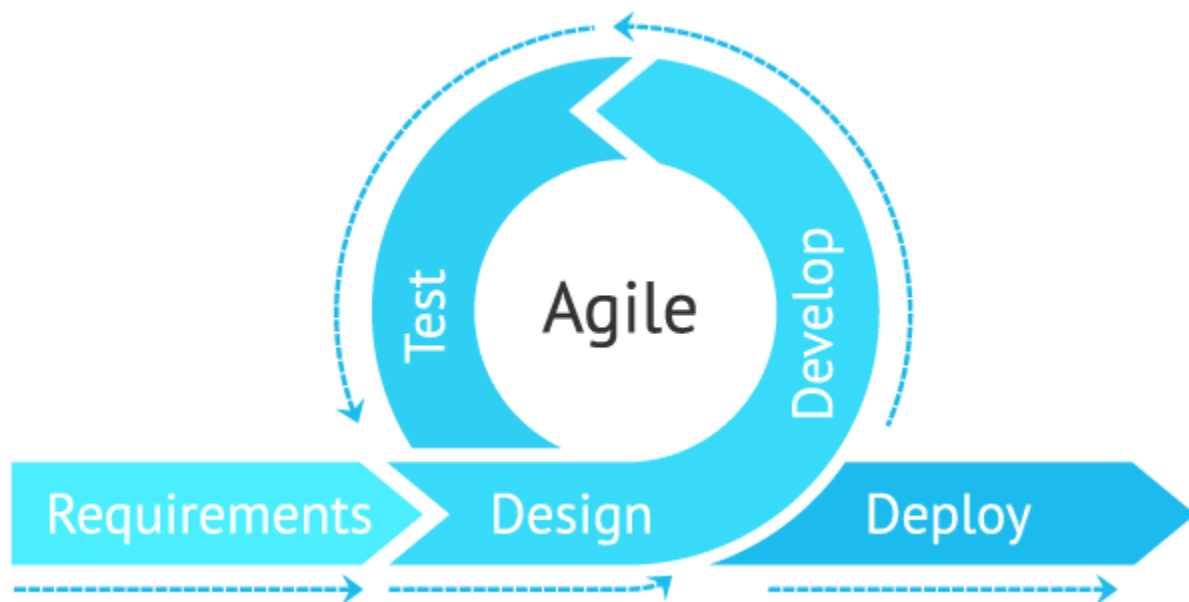


Figure 10 Agile methodology

4.2 Design Tools

StarUML: is a software modeler aimed to support agile and concise modelling.

Creately: an online diagramming and collaboration

4.3 Activity diagram

The activity model represents a series of activities and the flow from one activity to another that each of the actors do to get a certain result or to use a feature.

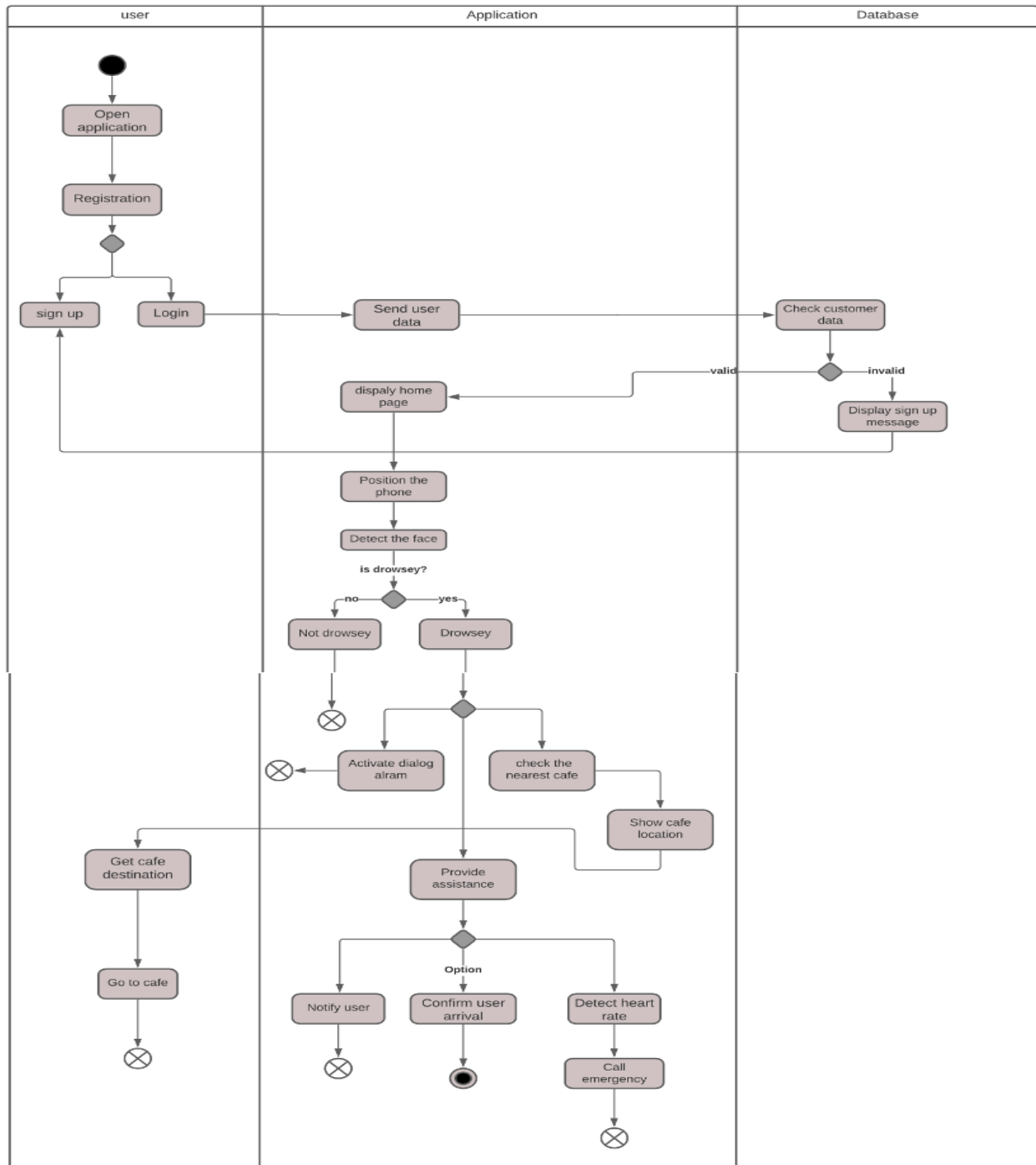


Figure 11 Activity Diagram

4.4 Entity Relation diagram

The ER diagram shows the entities stored in our system database and the relationships between them.

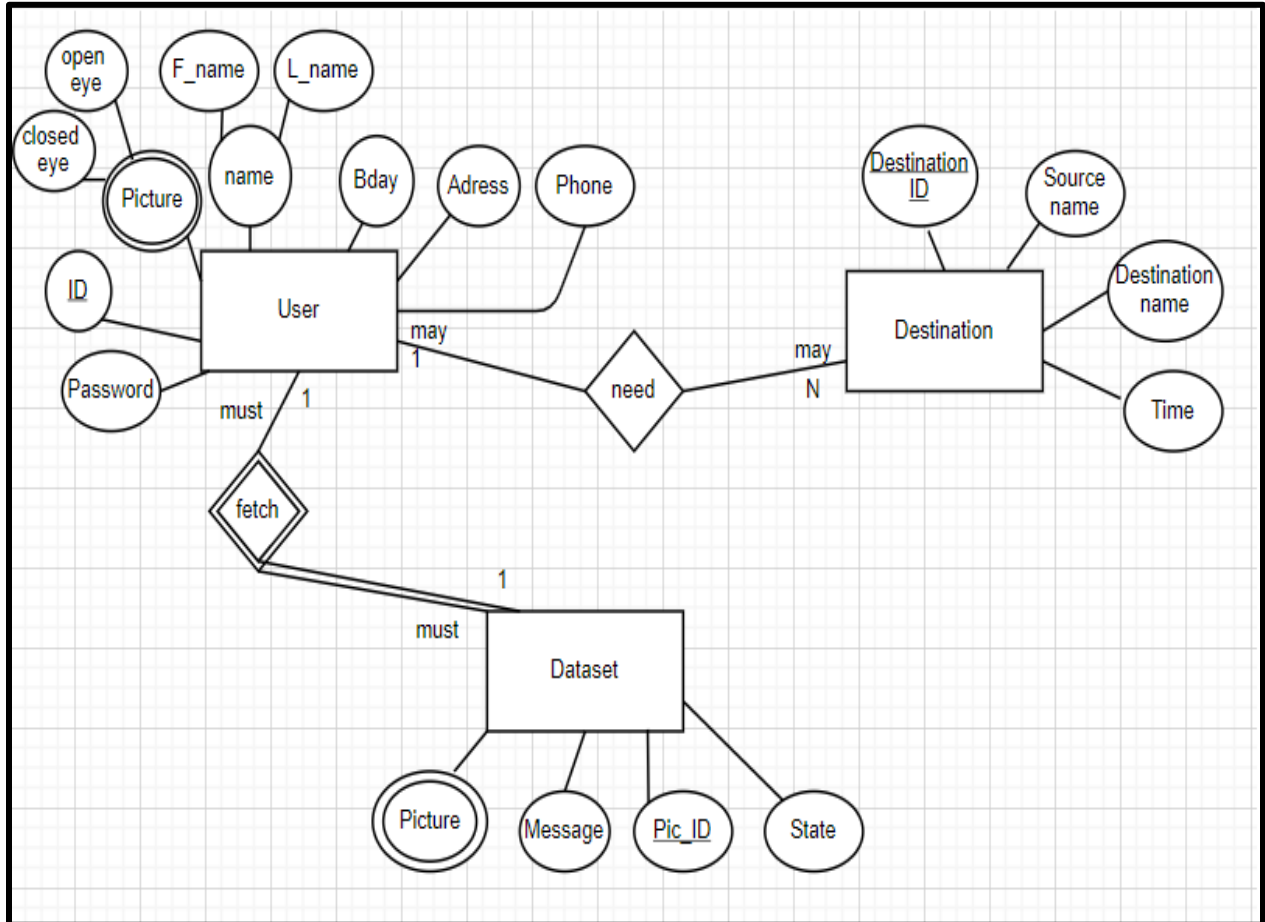


Figure 12 Entity Relation Diagram

4.5 Class diagram

Class diagram is a static model to represent our system classes and the relationships between them.

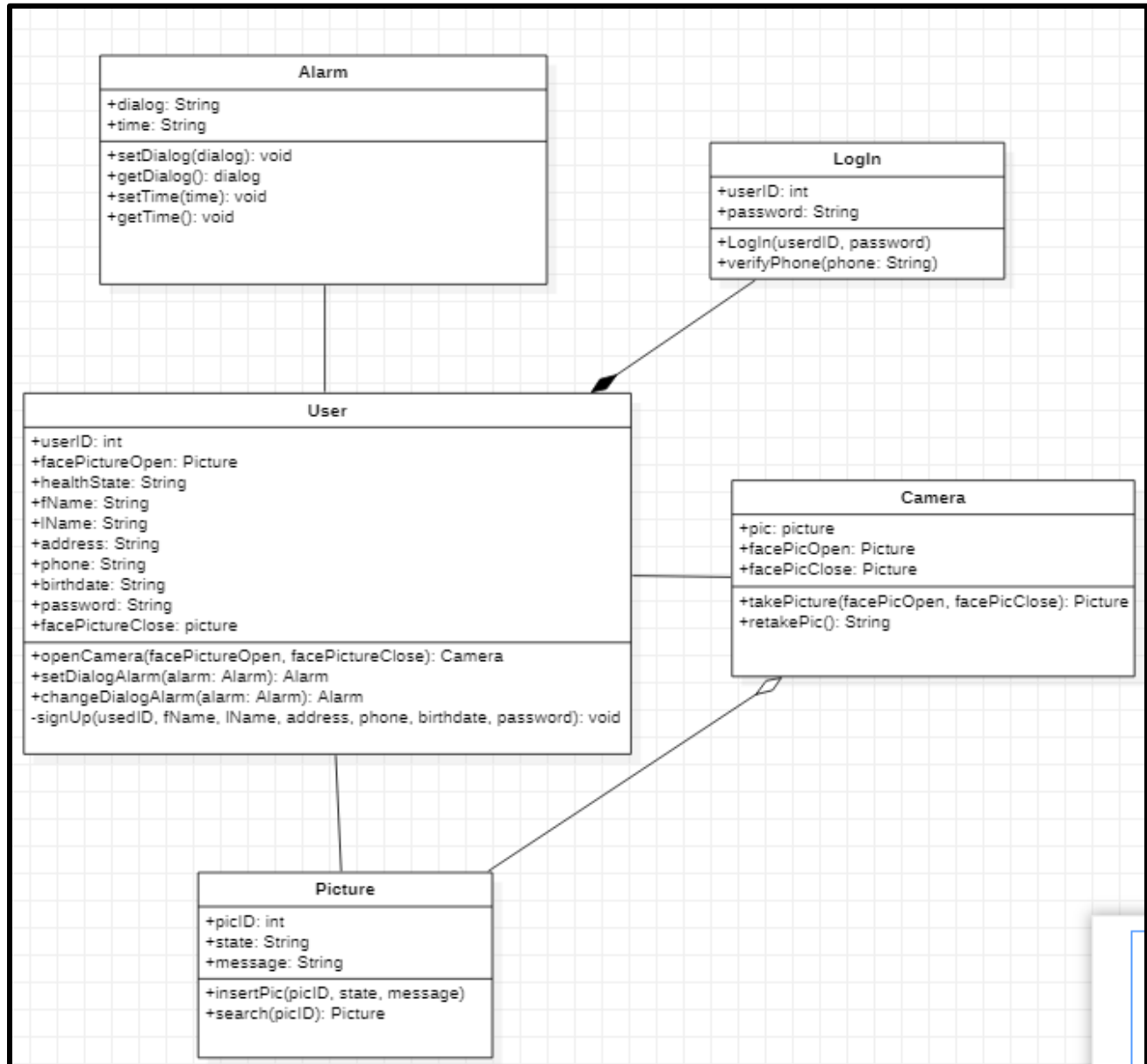


Figure 13 Class Diagram

Chapter V: Implementation

5.1 Introduction

This chapter will cover the implementation of the main functions of the system. Furthermore, we will present the problems and difficulties that we encountered during the Implementation.

5.2 Implementation tools and languages

- phpMyAdmin is a free and open-source administration tool prepared to handle the administration of MySQL over the web. It was used to build a system database
- PHP (Hypertext Preprocessor Language) is processing and dealing with a web server by its interpreter. It was used to build system back end.
- Android Studio: The fastest developer tools for building market-leading apps and accelerating performance. With an intelligent code editor, flexible build system, Realtime profilers and emulators. Code with confidence. Building without limits. Eliminate tiresome tasks. Build rich experiences.
- Google Cloud Vision API allows developers to easily integrate vision detection features within applications, including image labeling, face and landmark detection, optical character recognition (OCR), and tagging of explicit content.
- The XML is a flexible way to create information formats and electronically share structured data via the public internet, as well as via corporate networks.
- Java is a programming language and computing platform
- JSON is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects

5.3 Website Implementation

5.3.1 Sign-Up

Users can create their own account by filling the sign-up form as shown in the Figure 14

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_sign_up);  
  
    mNameET = findViewById(R.id.user_name);  
    mPasswordET = findViewById(R.id.password);  
    mPhoneET = findViewById(R.id.phone);  
    mAddressET = findViewById(R.id.address);  
    mBirthDateBtn = findViewById(R.id.birth_date);  
  
    mSignUpBtn = findViewById(R.id.sign_up_btn);  
  
    mBirthDateBtn.setOnClickListener(v -> openCalender());  
  
    mSignUpBtn.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            if(verifyInput()){  
                mSignUpBtn.setEnabled(false);  
                registerUser();  
            }  
        }  
    });  
}
```

Figure 14 Sign-up code

The processing of the user request must go through a set of condition to validate his input before creating a record in the database. The first step is to validate user's name as shown in Figure 15

```
//checking if username is empty  
if (TextUtils.isEmpty(userName)) {  
    Toast.makeText(context, "please enter userName!", Toast.LENGTH_SHORT).show();  
    mSignUpBtn.setEnabled(true);  
    return false;  
}
```

Figure 15 username validation

The second step is to validate user's password as shown in Figure 16

```
//checking if password is empty  
if (TextUtils.isEmpty(pass)) {  
    Toast.makeText(context, "please enter your password!", Toast.LENGTH_SHORT).show();  
    mSignUpBtn.setEnabled(true);  
    return false;  
}
```

Figure 16 password validation

The third step is to validate user's Phone as shown in Figure 17

```
//checking if password is empty
if (TextUtils.isEmpty(phone)) {
    Toast.makeText(context: this, text: "please enter your phone number!", Toast.LENGTH_SHORT).show();
    mSignUpBtn.setEnabled(true);
    return false;
}
```

Figure 17 phone validation

The fourth step is to validate user's address as shown in Figure 18

```
//checking if address is empty
if (TextUtils.isEmpty(address)) {
    Toast.makeText(context: this, text: "please enter your address!", Toast.LENGTH_SHORT).show();
    mSignUpBtn.setEnabled(true);
    return false;
}
```

Figure 18 address validation

The last step is to validate user's birthdate as shown in Figure 19

```
if (TextUtils.isEmpty(birthDate)) {
    Toast.makeText(context: this, text: "please enter your birthDate", Toast.LENGTH_SHORT).show();
    mSignUpBtn.setEnabled(true);
    return false;
}
```

Figure 19 birthdate validation

5.3.2 Log-In

In order to login to the system the user need to fill the form in the login page as shown in

Figure 20

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_login);

mPasswordET = findViewById(R.id.password);
mPhoneET = findViewById(R.id.phone);

mLogin = findViewById(R.id.log_in_btn);

mLogin.setOnClickListener(v -> {
    if(verifyInput()){
        mLogin.setEnabled(false);
        userLogin();
    }
})
```

Figure 20 login form

5.3.3 Emergency Contact

The user can enter an emergency contact that the application can contact if the user is not responding Figure 21.

```
String nearbyApiKey = "FlxIYPnQ5rhflcpoorSJLpd3JjT1oKx9";
url += "key=" + nearbyApiKey + "&lat=" + lat + "&lon=" + lon + "&categorySet=" + Constant.CAFE;

AndroidNetworking.get(url)
    .setPriority(Priority.LOW)
    .build()
    .getAsJSONObject(new JSONObjectRequestListener() {
        @Override
        public void onResponse(JSONObject response) {
            try {
                NearbyPlacesJsonFormatter obj = new NearbyPlacesJsonFormatter(response);
                cafesAround = obj.getCafes();

                if(!cafesAround.isEmpty()){
                    for (Cafe cafe:cafesAround){
                        LatLng cafeLocation = new LatLng(cafe.getLat(), cafe.getLon());
                        mMap.addMarker(new MarkerOptions()
                            .position(cafeLocation)
                            .title(cafe.getName()))
                            .setIcon(BitmapFromVector(getContext(), R.drawable.cafe));
                    }
                }
            }
        }
    });
```

Figure 21 emergency contact

5.3.4 Drowsiness detection

Our main function in the Eyes-Up Application is the face recognition, we used it in detecting the user's drowsiness. Figure 22 shows how we started it

```
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    if (ContextCompat.checkSelfPermission(requireContext(), Manifest.permission.CAMERA) == PackageManager.PERMISSION_DENIED) {
        ActivityCompat.requestPermissions(requireActivity(), new String[] {Manifest.permission.CAMERA}, 1);
    }else {
        setUpCamera(view);

        safeToTakeAPicture = true;

        // And From your main() method or any other method
        timer = new Timer();
        timer.schedule(new TaskPic(), delay: 1000, period: 3000);
    }
}
```

Figure 22 detecting drowsiness

The first step is when the user clicks the detecting button the application will set up the camera
Figure 23.

```
public void setUpCamera(View view){
    mCamera = openFrontFacingCamera();

    // Create our Preview view and set it as the content of our activity.
    mPreview = new CameraPreview(getContext(), mCamera);
    FrameLayout preview = view.findViewById(R.id.camera_preview);
    preview.addView(mPreview);

    mCamera.setDisplayOrientation(90);
}
```

Figure 23 set up camera

Second step the is opening the front facing camera Figure 24.

```
private Camera openFrontFacingCamera() {
    int cameraCount = 0;
    Camera cam = null;
    Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
    cameraCount = Camera.getNumberOfCameras();
    for ( int camIdx = 0; camIdx < cameraCount; camIdx++ ) {
        Camera.getCameraInfo( camIdx, cameraInfo );
        if ( cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT ) {
            try {
                cam = Camera.open( camIdx );
            } catch (RuntimeException e) {
                Log.e(TAG, "msg: " + e.getLocalizedMessage());
            }
        }
    }
    return cam;
}
```

Figure 24 open front camera

Third step is starting to detect the user's left and right eyes and check if it is closed more then 40% Figure 25.

```
if (face.getRightEyeOpenProbability() != null) {  
    float rightEyeOpenProb = face.getRightEyeOpenProbability();  
    if(rightEyeOpenProb < 0.4 && rightEyeOpenProb>-1){  
        rightEye = false;  
    }else if(rightEyeOpenProb > 0.4){  
        rightEye = true;  
    }  
}  
  
if (face.getLeftEyeOpenProbability() != null) {  
    float leftEyeOpenProb = face.getLeftEyeOpenProbability();  
    if(leftEyeOpenProb < 0.4 && leftEyeOpenProb>-1){  
        leftEye = false;  
    }else if(leftEyeOpenProb > 0.4){  
        leftEye = true;  
    }  
}
```

Figure 25 detecting eyes

If both eyes are closed more than 40% an alarm will start Figure 26

```
if(!rightEye && !leftEye){  
    addToClosed();  
    playAlarm(eyesClosedCount);  
}else{  
    resetClosedCount();  
    playSound(false);  
}
```

Figure 26 alarm start

5.3.5 Play Alarm

When the application detects the drowsiness of the user they will alert them multiple time before starting the final alarm or calling for emergency

The first alarm will go off after 9 seconds of closing both eyes Figure 27.

```
case 3:
    String firstAlarm = "sir please focus !";
    t1.speak(firstAlarm, TextToSpeech.QUEUE_FLUSH, params: null);
    break;
```

Figure 27 first alarm

Second alarm will go off after 21 seconds of not opening both eyes from the first alarm, and a confirmation dialog will appear to the user and ask them to confirm they are awake by saying “I AM AWAKE”. Figure 28.

```
(alarmType){

    Context ctx = requireContext();
    LayoutInflater factory = LayoutInflater.from(ctx);
    final View view = factory.inflate(R.layout.confirmation_dialog, root: null);
    final AlertDialog cancelSoundDialog = new AlertDialog.Builder(ctx).create();
    cancelSoundDialog.setView(view);
    cancelSoundDialog.setCanceledOnTouchOutside(false);

    TextView yes = view.findViewById(R.id.yes_btn);
    TextView no = view.findViewById(R.id.no_btn);

    playSound(true);
    CountDownTimer countDownTimer = new CountDownTimer( millisInFuture: 10000, countDownInterval: 1000) {

        public void onTick(long millisUntilFinished) {

        }

        public void onFinish() {
            playSound(false);
            cancelSoundDialog.dismiss();
            t1.speak( text: "sir I am listening please confirm that you are awake", TextToSpeech.QUEUE_FLUSH,
            startListening());
        }
    };
}
```

Figure 28 second alarm

The last alarm will call the emergency contact the user has inserted in the application if there is no response. Figure 29.

```
case 10:
    if(prefManager.isContactNumberSet()){
        Context ctx = requireContext();
        LayoutInflater factory = LayoutInflater.from(ctx);
        final View view = factory.inflate(R.layout.confirmation_dialog, root: null);
        final AlertDialog callConfirmationDialog = new AlertDialog.Builder(ctx).create();
        callConfirmationDialog.setView(view);
        callConfirmationDialog.setCanceledOnTouchOutside(false);

        TextView message = view.findViewById(R.id.message);
        TextView yes = view.findViewById(R.id.yes_btn);

        CountDownTimer countDownTimer = new CountDownTimer( millisInFuture: 10000, countDownInterval: 1000 ) {

            public void onTick(long millisUntilFinished) {
                message.setText("calling the emergency contact number in " + millisUntilFinished);
            }

            public void onFinish() { callAFriend(contactNumber); }
        }.start();

        yes.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(countDownTimer != null){
                    countDownTimer.cancel();
                }
            }
        });
    }
}
```

Figure 29 third alarm and calling emergency contact

5.3.6 Nearest Café Locations

Also one of the application's main feature is giving them the nearest café location from them to sober up and rest first the application will get the users application. Figure 30.

```
public void getLocation(){
    gpsTracker = new GpsTracker(getActivity());
    if(gpsTracker.canGetLocation()){
        while (gpsTracker.getLatitude() == 0 || gpsTracker.getLongitude() == 0){
            gpsTracker.getLocation();
        }
        mMyLatitude = gpsTracker.getLatitude();
        mMyLongitude = gpsTracker.getLongitude();
    }else{
        gpsTracker.showSettingsAlert();
    }
}
```

Figure 30 Nearest cafe location

The second step the application will start finding the nearest cafes around the user's location. Figure 31.

```
String url = Constant.MAPS_API_REQUEST_NEARBY_CAFES;
String nearbyApiKey = "FlxIYPnQ5rhflcpoorSJLpd3JjT1oKx9";
url += "key=" + nearbyApiKey + "&lat=" + lat + "&lon=" + lon + "&categorySet=" + Constant.CAFES;

AndroidNetworking.get(url)
    .setPriority(Priority.LOW)
    .build()
    .getAsJSONObject(new JSONObjectRequestListener() {
        @Override
        public void onResponse(JSONObject response) {
            try {
                NearbyPlacesJsonFormatter obj = new NearbyPlacesJsonFormatter(response);
                cafesAround = obj.getCafes();

                if(!cafesAround.isEmpty()){
                    for (Cafe cafe:cafesAround){
                        LatLng cafeLocation = new LatLng(cafe.getLat(), cafe.getLon());
                        mMap.addMarker(new MarkerOptions()
                            .position(cafeLocation)
                            .title(cafe.getName()))
                            .setIcon(BitmapFromVector(getContext(), R.drawable.cafe));
                    }
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
```

Figure 31 Nearest cafe around user

If user selected one of the cafes that appeared in the map the application will ask them if they want to see directions to this location. Figure 32.

```
Context ctx = requireContext();
LayoutInflater factory = LayoutInflater.from(ctx);
final View view = factory.inflate(R.layout.delete_confirmation_dialog, root: null);
final AlertDialog openMapsDialog = new AlertDialog.Builder(ctx).create();
openMapsDialog.setView(view);

TextView yes = view.findViewById(R.id.yes_btn);
TextView no = view.findViewById(R.id.no_btn);
TextView message = view.findViewById(R.id.message);

message.setText("do you want to open google maps to show directions to" + " " + m.getTitle());

yes.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
            Uri.parse("http://maps.google.com/maps?daddr="+m.getPosition().latitude+"-"+m.getPosition().longitude));
        startActivity(intent);
        openMapsDialog.dismiss();
    }
});
```

Figure 32 Google Map Permission

5.3.7 Users Profile

The user can see and edit his account information in the Eye-Up application and if user is done with the application, they can log out of it. Figure 33 and 34.

```
super.onCreate(view, savedInstanceState);

mLogoutBtn = view.findViewById(R.id.logout);

mUserNameTV = view.findViewById(R.id.user_name_text_view);
mPhoneTV = view.findViewById(R.id.phone_text_view);
mAddressTV = view.findViewById(R.id.address_text_view);

User user = SharedPrefManager.getInstance(requireContext()).getUserData();

mUserNameTV.setText(user.getUserName());
mPhoneTV.setText(user.getPhone());
mAddressTV.setText(user.getAddress());

mLogoutBtn.setOnClickListener(v -> {
    logOut();
});
```

Figure 33 Log-out

```
public void logOut(){
    Context ctx = requireContext();
    SharedPrefManager.getInstance(ctx).logout();
    PackageManager packageManager = ctx.getPackageManager();
    Intent intent = packageManager.getLaunchIntentForPackage(ctx.getPackageName());
    ComponentName componentName = intent.getComponent();
    Intent mainIntent = Intent.makeRestartActivityTask(componentName);
    startActivity(mainIntent);
    Runtime.getRuntime().exit(status: 0);
}
```

Figure 34 Log-out 2

5.4 Database

5.4.1 Connection

To export and import data, we need to connect the website with the database on a server. The database for the system has been linked, as shown in Figure 35.

```
<?php

class DB_Connect {
    private $conn;

    // Connecting to database
    public function connect() {
        require_once 'Config.php';

        // Connecting to mysql database
        $this->conn = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_DATABASE);

        // return database handler
        return $this->conn;
    }
}
```

Figure 35 Database Connection

5.4.2 Database Functions

We stored all the functions that we will need in an external file so it will be easy to fix and call. Figure 36.

Store User Function

```
public function storeUser(
    $username,
    $password,
    $phone,
    $birthdate,
    $address)
{
    $password = md5($password);
    $stmt = $this->conn->prepare("INSERT INTO `users` (`id`, `username`, `password`, `phone`, `birthdate`, `status`, `address`)
    $stmt->bind_param("sssss", $username,
        $password,
        $phone,
        $birthdate,
        $address);
    $result = $stmt->execute();
    $stmt->close();

    // check for successful store
    if ($result) {
        $stmt = $this->conn->prepare("SELECT * FROM users WHERE phone = ?");
        $stmt->bind_param("s", $phone);
        $stmt->execute();
        $user = $stmt->get_result()->fetch_assoc();
        $stmt->close();

        return $user;
    } else {
        return false;
    }
}
```

Figure 36 Store User Function

Update user function

We allowed the users to update their account information. Figure 37.

```
public function updateUser($user_id,
                          $username,
                          $birthdate,
                          $status)
{
    if ($username != -1) {
        $sql = "UPDATE users SET username='" . $username . "' where id=" . $user_id . " ";
        $this->conn->query($sql);
    }

    if ($birthdate != -1) {
        $sql = "UPDATE users SET birthdate='" . $birthdate . "' where id=" . $user_id . " ";
        $this->conn->query($sql);
    }

    if ($status != -1) {
        $sql = "UPDATE users SET status='" . $status . "' where id=" . $user_id . " ";
        $this->conn->query($sql);
    }

    $stmt = $this->conn->prepare("SELECT * FROM users WHERE id = " . $user_id);
    $stmt->execute();
    $user = $stmt->get_result()->fetch_assoc();
    $stmt->close();
    return $user;
}
```

Figure 37 Update user information

Update User Password

We allowed users to update their own passwords. Figure 38.

```
public function updateUserPassword(
    $phone, $password)
{
    $password = md5($password);
    $sql = "UPDATE users SET password='" . $password . "' where phone='" . $phone . "' ";
    $this->conn->query($sql);

    $stmt = $this->conn->prepare("SELECT * FROM users WHERE phone ='" . $phone . "'");
    $stmt->execute();
    $user = $stmt->get_result()->fetch_assoc();
    $stmt->close();
    return $user;
}
```

Figure 38 Update User password

Get User By phone

Since phone number is the primary key, we call all users by their phone number. Figure 39.

```
public function getUserByEmailAndPassword($phone, $password)
{
    $password = md5($password);

    $stmt = $this->conn->prepare("SELECT * FROM `users` WHERE `phone`='" . $phone . "' and `password`='" . $password . "' and status=1;");

    if ($stmt->execute()) {
        $user = $stmt->get_result()->fetch_assoc();
        $stmt->close();

        return $user;
    } else {
        return false;
    }
}
```

Figure 39 Find User by primary key

Check if user exist

When user wants to log in the database check first if they are already exist or not by their phone number. Figure 40.

```
public function isUserExisted($phone)
{
    $stmt = $this->conn->prepare("SELECT * from users WHERE phone = ?");

    $stmt->bind_param("s", $phone);

    $stmt->execute();

    $stmt->store_result();

    if ($stmt->num_rows > 0) {
        // user existed
        $stmt->close();
        return true;
    } else {
        // user not existed
        $stmt->close();
        return false;
    }
}
```

Figure 40 Check if user exist

5.4.3 Tables Creation and data insertion

As shown in Figure 41, the system database contains one table which will be storing all user's information called users

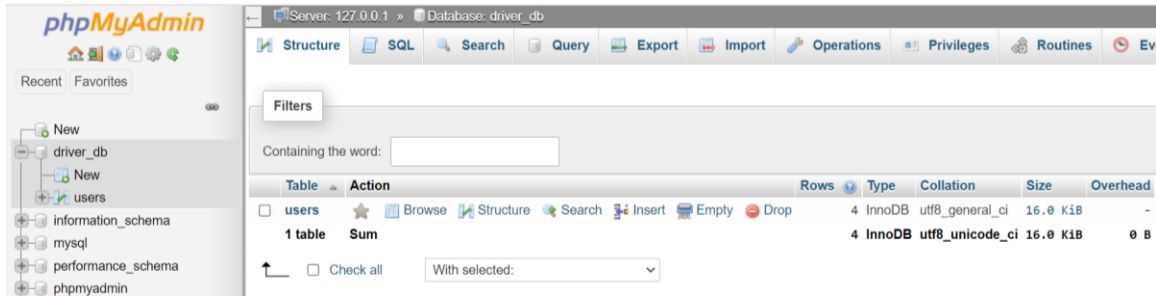


Figure 41 database table

Sign Up a new user

```
if ($type == 'register') {
    $username = $_GET['username'];
    $password = $_GET['password'];
    $phone = $_GET['phone'];
    $birthdate = $_GET['birthdate'];
    $address = isset($_GET['address']) ? $_GET['address'] : '';

    if (strlen($username) < 2) {
        $response["error"] = TRUE;
        $response["msg"] = "Enter valid name";
        echo json_encode($response);
        die();
    }

    if (strlen($phone) < 8) {
        $response["error"] = TRUE;
        $response["msg"] = $phone . " is not a valid phone";
        echo json_encode($response);
        die();
    } else {

        if ($db->isUserExisted($phone)) {
            $response["error"] = TRUE;
            $response["msg"] = "Phone used before!";
            echo json_encode($response);
            die();
        }
    }

    if (strlen($password) < 8) {
        $response["error"] = TRUE;
        $response["msg"] = "Password should be more than 8 letters";
        echo json_encode($response);
        die();
    }
}
```

Figure 42 Sign-up a new user

Log in

```
(($type == 'login')) {
    $phone = $_GET['phone'];
    $password = $_GET['password'];

    if (strlen($phone) < 10) {
        $response["error"] = TRUE;
        $response["msg"] = $phone . " is not a valid phone";
        echo json_encode($response);
        die();
    }

    if (strlen($password) < 8) {
        $response["error"] = TRUE;
        $response["msg"] = "Password should be more than 8 letters";
        echo json_encode($response);
        die();
    }

    $user = $db->getUserByEmailAndPassword($phone, $password);

    if ($user != false) {
        $response["error"] = FALSE;
        $response["msg"] = "Login successfully";
        $response["data"]["id"] = $user["id"];
        $response["data"]["username"] = $user["username"];
        $response["data"]["phone"] = $user["phone"];
        $response["data"]["birthdate"] = $user["birthdate"];
        $response["data"]["address"] = $user["address"];

        echo json_encode($response);
        die();
    } else {
        --
    }
}
```

Figure 43 Log-in users

5.5 Difficulties

Table 13 shows the problems and difficulties that we encountered during the implementation

Table 13 Difficulties in implementation

Problem Description	Solution
Face Detection	
When the user starts the face detection the camera automatically opens the back facing camera.	We added a code that will force open the front camera instead.
When the user starts the face detection the camera automatically rotates the users face.	We added a code that will rotate the user's camera 90 degree.

5.6 System Objectives

Table 14 shows the objectives that have been achieved.

Table 14 system objectives

Objective	Achieved
Safety: Providing a safer driving experience with less accidents.	yes
Assistance: Assist the driver when needed to focus while driving	yes
Interface: Creating a simple and easy to use interface for easy communication	yes
Communication: Communicate with the driver whenever drowsiness is detected on the driver's face	yes

5.7 Conclusion

In this chapter, the implementation of the Eyes-Up system has been discussed with all the important details of the tools and technology used to develop and build the application along with the database and the interfaces

Chapter VI: Testing

6.1 Introduction

There are many different levels of testing to evaluate and assist the implementation of the system. three types of system testing is performed on the Eyes-Up application. System testing, Unit testing and useability testing are chosen in this chapter. The goal of system testing is to ensure that the integrated system is working as expected and the useability testing refers to how easy users find it to accomplish specific function

6.2 Testing Description

6.2.1 Preparations

Before testing our application, we identified the goals and objectives of the testing phase. We also determined our target users, application functionalities, test cases and measurement metrics that we will test during this phase.

6.2.2 Goals

The main objective of our test is to ensure that our application performs all required functions correctly. In addition, Testing in General is to find bugs in our program.

6.2.3 Target users:

Car drivers: especially those traveling in long road and feeling sleepy.

6.2.4 Test Environment:

The Eyes-up application testing conducted at King Abdul-Aziz University in FCIT College. Test date starts from Sunday 7 November 2021 to Thursday 2 December 2021.

6.3 Functionalities

Eyes-Up application provides many functionalities for the drivers. The following are the list of tasks to be tested by them.

- Task 1: Sign Up
- Task 2: Sign In
- Task 3: User account
- Task 4: Check nearby cafés
- Task 5: Allow emergency contact
- Task 6: choose Alarm type
- Task 7: Add to do list
- Task 8: Start Detecting
- Task 9: Sign Out

The following table demonstrates the correct navigation path to accomplish each task.

The path to accomplish each task

Task No.	Task Name	Right Path
1	Sign Up	Main Menu → Sign Up
2	Sign In	Main Menu → Sign In
3	User Account	Sign In → Profile
4	Check nearby cafés	Sign In → Check nearby cafes
5	Allow emergency contact	Sign In → Option menu → Allow Emergency contact
6	choose Alarm type	Sign In → Option menu → Change Alarm type
7	Add to do list	Sign In → Notes Menu → Add New Notes
8	Start Detecting	Sign In → Detecting Menu
9	Sign Out	Sign In → Profile menu → Sign Out

6.4 Testing Methods

6.4.1 System Testing

System testing tests the application, and it is done manually by trying different test scenarios and observing the application results. The following table (Table 16) shows several different test scenarios.

Table 16 Test scenario

Test scenario
login->home -> menu -> note
login-> faces detect -> sound alarm -> nearby cafe
login-> account page->update information -> sign out
login-> face detect -> pop up message alarm -> press the message button
login-> menu -> change alarm type -> pop up message
login-> menu -> change alarm type -> sound alarm

6.4.2 Usability testing

The usefulness of a product can be assessed by two key features – its utility and its usability. Utility refers to a product's ability to do a specific function. Usability refers to how easy users find it to accomplish that specific function. We will test the usability of our application following Nielsen criteria.

- Learnability: How easy to accomplish a task when the user uses the application for the first time?
- Efficiency: How quickly to perform a task after learning the applications?
- Memorability: How easy to reestablish proficiency when the user returns to the application after a period?
- Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- Satisfaction: How pleasant is it to use the application?

Table 17 Criteria Of Measuring Usability According to Nielsen(2003)

Measure	Strongly Agree	Agree	Strongly disagree	Disagree	No opinion
Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?					
Efficiency: Once users have learned the design, how quickly can they perform tasks?					
Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?					
Satisfactory: How pleasant is it to use the design?					

6.4.2.1 Usability testing plan

We will do the usability testing by giving our application to a group of users who have cars and need to travel or drive on long road. They will use our application to give us feedbacks about it, their opinion, satisfaction, and what difficulties they face while using it so we can increase the usability of our system by solve the problems that face the user.

6.5 Testing Result

6.5.1 Unit Testing

```
@Test
public void testcheckPasswod(){
    System.out.println("checkPasswod");
    String password = "1234567Cd";
    User instance = new User();
    boolean expResult = true;
    boolean result = instance.checkPasswod(password);
    assertEquals(expResult, result);
}
```

Table 18 check password test case

Functionality	Input	Expected result	Result
Check Password	More than 8 At least one small letter At least one capital letter No special character	True	Pass
	less than 8 At least one small letter At least one capital letter No special character	False	Pass
	More then 8 No one small letter At least one capital letter No special character	False	Pass
	More then 8 At least one small letter No one capital letter No special character	False	Pass
	More then 8 At least one small letter At least one capital letter with special character	False	Pass

```

@Test
public void testcheckPhone(){
    System.out.println("checkPhone");
    String phone = "1234567890";
    User instance = new User();
    boolean expResult = true;
    boolean result = instance.checkPhone(phone);
    assertEquals(expResult, result);
}

```

Table 19 check phone number test case

Functionality	Input	Expected result	Result
Check phone number	More than 10 No letters	True	Pass
	less than 10 No letters	False	Pass
	More than 10 With letters	False	Pass

6.5.2 System Testing

Table 20 System testing results

Test scenario	Result
login->home -> menu -> note	pass
login-> faces detect -> sound alarm -> nearby cafe	pass
login-> account page->update information -> sign out	pass
login-> face detect -> pop up message alarm -> press the message button	pass
login-> menu -> change alarm type -> pop up message	pass
login-> menu -> change alarm type -> sound alarm	pass

6.5.3 Usability testing

We performed the pre usability testing on drivers to get feedback. We asked users to evaluate the application's functionalities according to Nielsen criteria.

- Pre usability testing results

The following is a description of the evaluation results for each functionality:

- Sign-up

10 out of 10 drivers strongly agreed that the sign up interface design is learnable, and highly efficient. They also agreed that the process of signing up is easy to remember after a period of not using the application, and strongly agreed on the simplicity of the process when recovering from severe errors.

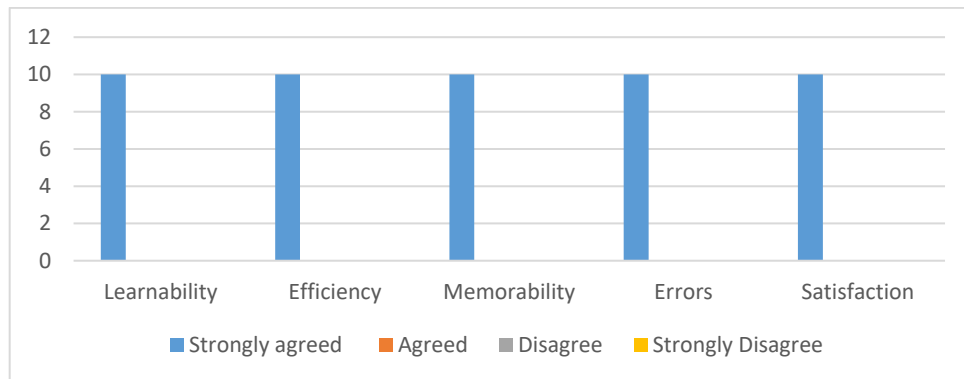


Figure 44 Sign up task results

- Sign-in

10 out of 10 drivers strongly agreed that the sign in interface design is learnable, and highly efficient. They also agreed that the process of signing in is easy to remember after a period of not using the application, and strongly agreed on the simplicity of the process when recovering from severe errors.

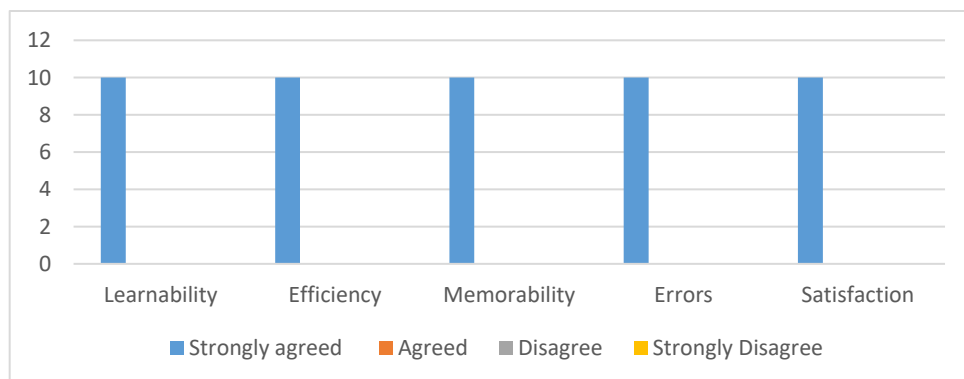


Figure 45 Sign in task results

- Option menu

10 out of 10 drivers strongly agreed that the option menu interface design is learnable, and highly efficient. They also agreed that the process of add the emergency contact and changing the alarm type is easy to remember after a period of not using the application, and strongly agreed on the simplicity of both processes when recovering from severe errors.

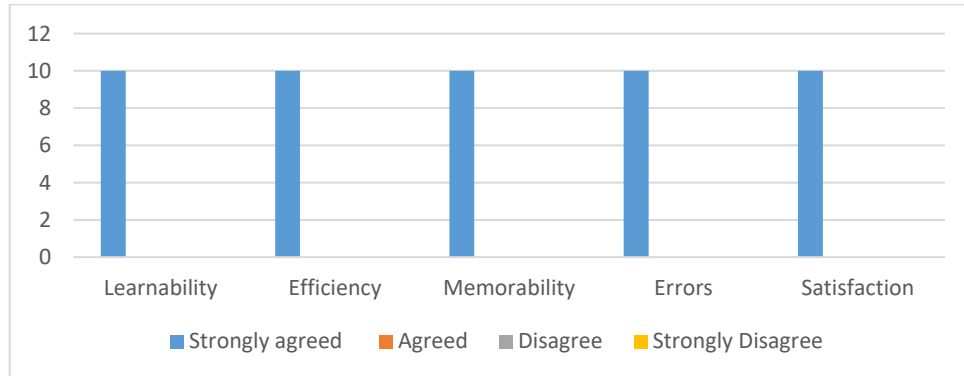


Figure 46 Option menu task results

- Face detecting

7 out of 10 drivers strongly agreed that the face detecting interface design are learnable, and highly efficient but 3 out of 10 drivers agreed. They also strongly agreed that the process of the face detecting is easy to remember after a period of not using the application, and strongly agreed on the simplicity of the process when recovering from severe errors.

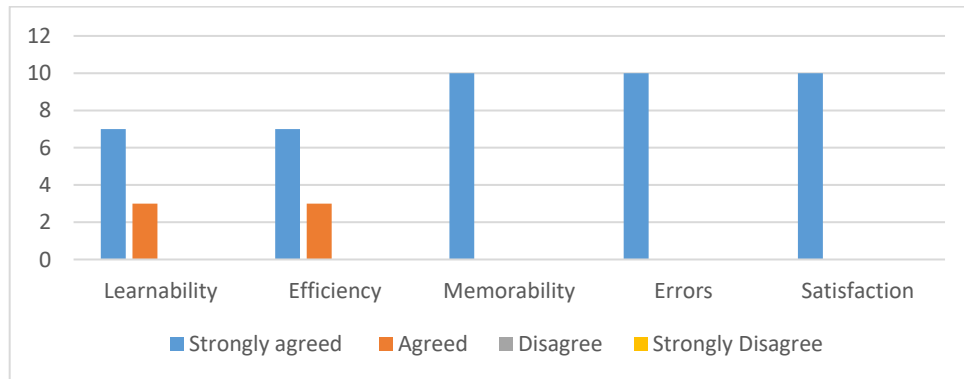


Figure 47 face detecting task results

- Checking nearby cafés

10 out of 10 drivers strongly agreed that checking the nearby cafés interface design are learnable, and highly efficient. They also strongly agreed that the process is easy to remember after a period of not using the application.

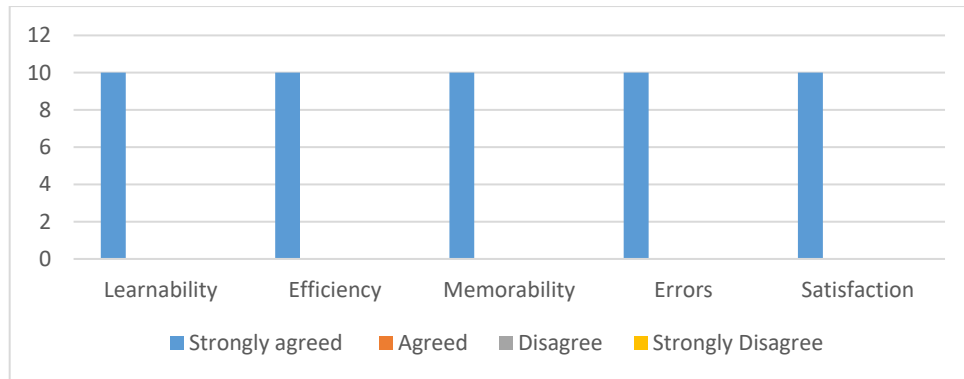


Figure 48 checking nearby cafes task result

- Adding to do list

10 out of 10 drivers strongly agreed that adding to do list interface design are learnable, and highly efficient. They also agreed that the process of adding to do list is easy to remember after a period of not using the application, and strongly agreed on the simplicity of the process when recovering from severe errors.

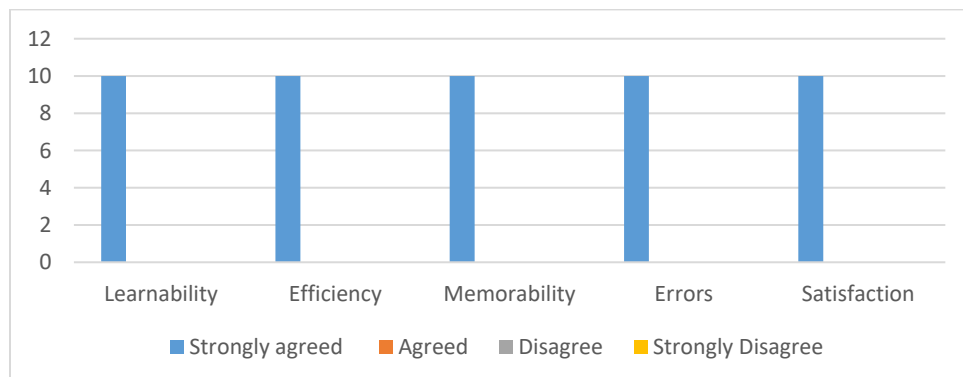


Figure 49 add to list task result

6.4 Conclusion

This chapter described our testing plan include testing preparing and goals. In addition, we described the test cases for unit, system, usability testing and the result for each of them.

Chapter VII: Results and Discussion

7.1 Introduction

In this chapter, we will represent the application interfaces to clarify the system functionalities and to demonstrate how each feature works and what are the interfaces that leads to accomplish the feature goal. We will describe the results with screenshots of the system's interfaces. The objectives that have been achieved and not achieved will also be discussed.

7.2 System Interface

The Eyes-Up application contains five main interfaces: Sign-up/Login page, detecting page, User Profile Page, Cafés nearby and finally the to do list.

7.2.1 Sign-up/Login Interfaces

Once the application installs and opens the Eyes-Up application the first page will appear is the login/sign up page. Figure 44.

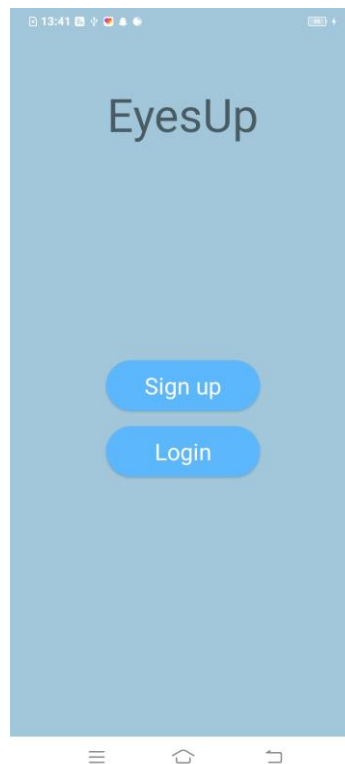


Figure 50 Sign-up/Login interface

Figure 45 shows the sign-up page, it gets displayed once the user presses the sign up button from the previous page. The user inserts all their information.

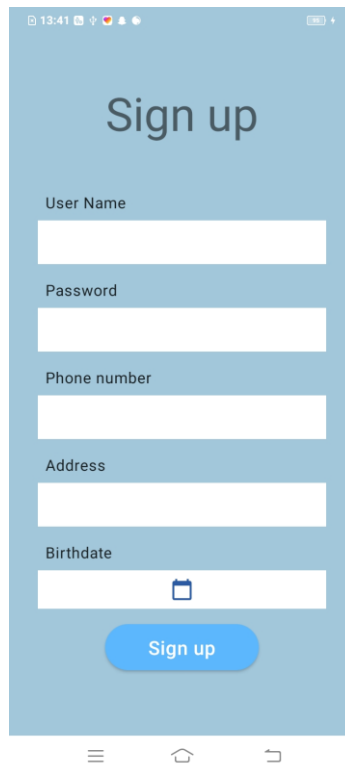
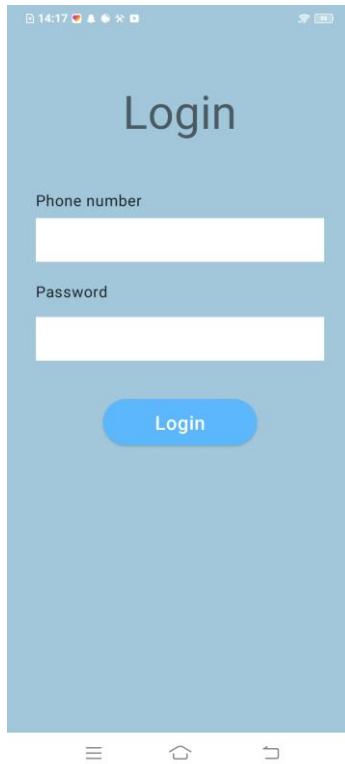


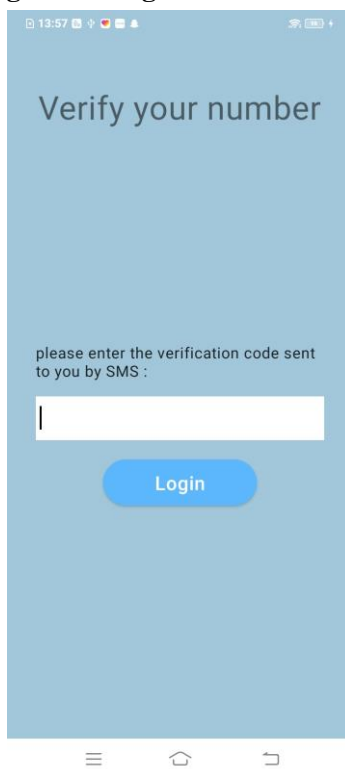
Figure 51 Sign-up page

Figure 46 shows the login page, it gets displayed once the user presses the login button from Figure 45 page. The user enters all their information to log into the application. And then verify by phone number Figure 47.



A mobile application login screen with a light blue background. At the top, the status bar shows the time 14:17 and various icons. The title "Login" is centered in a large, dark font. Below it, there are two white input fields: the first is labeled "Phone number" and the second is labeled "Password". A blue rounded button with the text "Login" is positioned below the password field. At the bottom, there is a white navigation bar with three icons: a hamburger menu, a home icon, and a back arrow.

Figure 52 Login Form Interface



A mobile application verification screen with a light blue background. At the top, the status bar shows the time 13:57 and various icons. The title "Verify your number" is centered in a large, dark font. Below it, there is a text prompt: "please enter the verification code sent to you by SMS :". Underneath this prompt is a white input field with a vertical cursor. A blue rounded button with the text "Login" is positioned below the input field. At the bottom, there is a white navigation bar with three icons: a hamburger menu, a home icon, and a back arrow.

Figure 53 Verify Login by phone number

7.2.2 User Profile interface

The user can easily update any of his credentials from his profile and can navigate between pages easily with the menu navigation at the bottom see Figure 48.

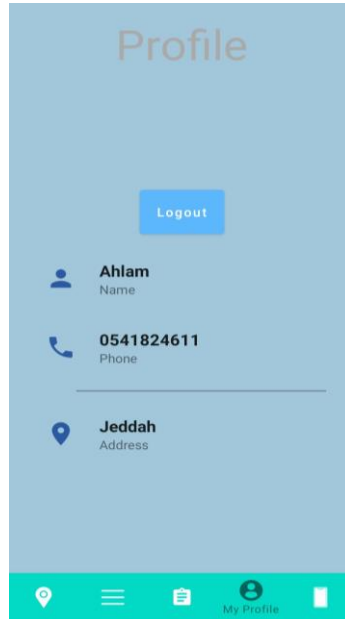


Figure 54 User profile

7.2.3 Nearest Café Location

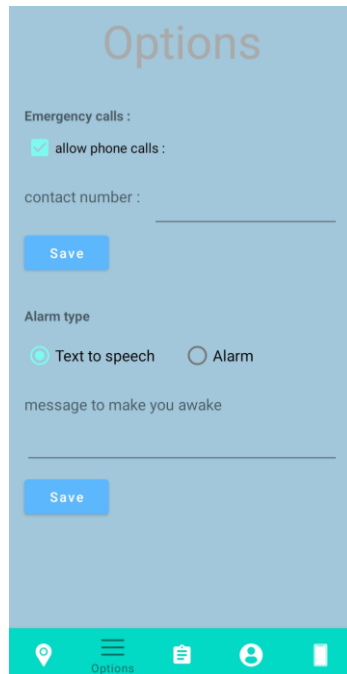
This interface will allow the user to check for the nearest cafés location and can easily find it in the bottom menu navigation. Figure 49.



Figure 55 Nearest Cafe location

7.2.4 Options Page

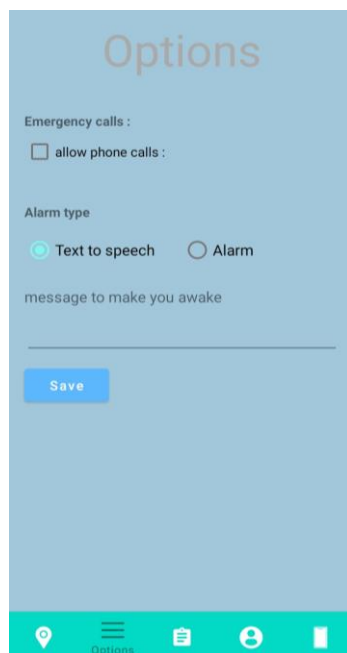
The user can choose if they want to put an emergency contact in the application in case, they did not response the application will call the contact they chose. Figure 50.



The screenshot shows a mobile application interface titled "Options". Under the heading "Emergency calls :", there is a checked checkbox labeled "allow phone calls :". Below this is a text input field for "contact number :". A blue "Save" button is positioned below the input field. Further down, under the heading "Alarm type", there are two radio button options: "Text to speech" (which is selected) and "Alarm". Below these options is another text input field for "message to make you awake". A second blue "Save" button is located below this field. At the bottom of the screen is a teal navigation bar with five icons: a location pin, a hamburger menu, a document, a person, and a battery icon. The word "Options" is centered below the menu icon.

Figure 56 Allow emergency contact

Also, the user can choose what type of alarm they want either the default alarm or they can choose and write a message to make them focus. Figure 51.



This screenshot is identical to the one in Figure 56, showing the "Options" page. However, in this state, the "allow phone calls :" checkbox is unchecked. The "Text to speech" radio button remains selected under the "Alarm type" section. The "message to make you awake" input field and the "Save" buttons are also present.

Figure 57 Choose alarm type

7.2.5 The Face Detecting Interface

Our main feature is the face detection users can easily find it in the bottom navigation menu and once the user clicks on it the face detection will start automatically Figure 52. Once the application detect that the user closed their eyes more than 9 seconds first alarm will start asking them to open their eyes. Second alarm will start after 21 seconds if user did not response asking the user to answer by voice that they are awake Figure 53. last alarm will start after 30 seconds calling the emergency contact Figure 54.



Figure 58 Detecting...

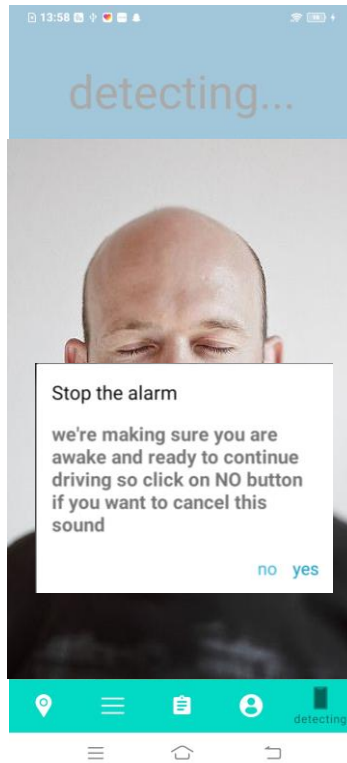


Figure 60 2nd Alarm

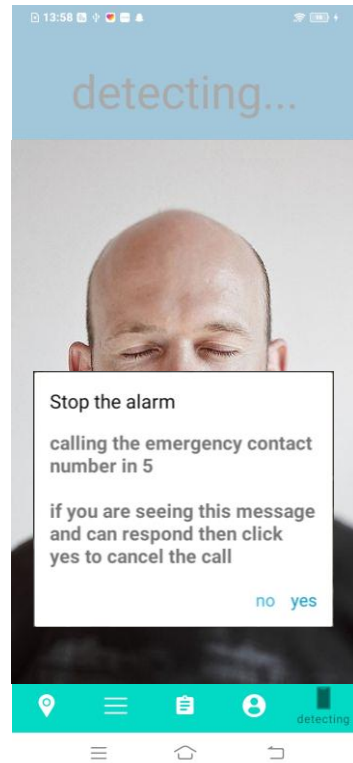


Figure 59 3rd Alarm

7.2.6 To Do List Interface

In this page the user can enter any notes they need to remember to do along the way Figure 55. They can just press the add button and add the note they need and save it Figure 56 and 57, and when they are done with they can just check the box Figure 58.

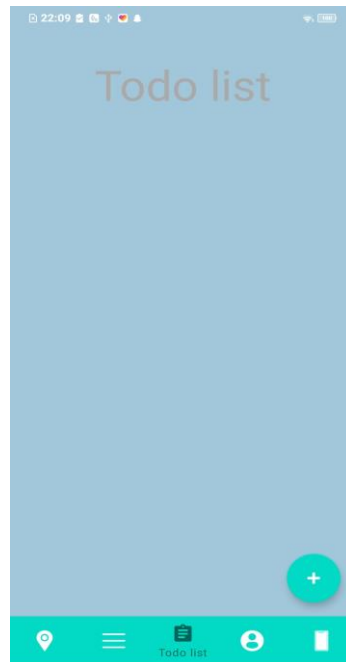


Figure 61 To do list

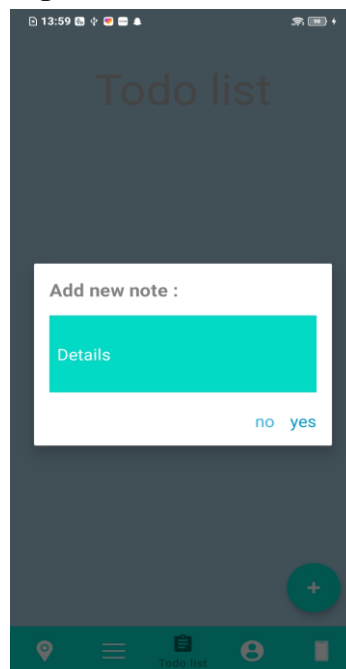


Figure 62 add new note

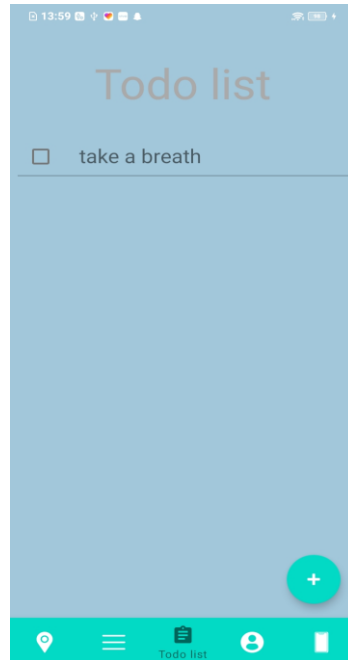


Figure 63 note added

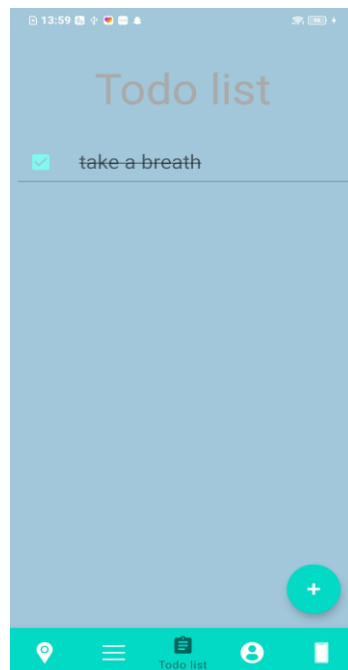


Figure 64 done with note

7.3 Achieved Objectives

- **Safety:** Providing a safer driving experience with less accidents by detecting the drivers face and eyes.
- **Assistance:** Assist the driver when needed to focus while driving by alarming them with loud sounds to get their attention.
- **Interface:** Creating a simple and easy to use interface for easy communication and by a French study the blue color will keep driver more focused.
- **Communication:** Communicate with the driver whenever drowsiness is detected on the driver's face by starting an alarm multilabel time and calling their emergency contact.

7.4 Work Limitation

1. The Login verification: we could not do it because it needs another system included and it was expensive to purchase.

2. The smart watch connecting with the application to calculate the driver's heartbeat to check if they are sleepy, but we could not do it because of the limitation of time.

3. Call 911 we did not force the application to call 911 in case of emergency but will let the driver choose who they want the application to call for them.

7.5 Conclusion

We presented all system interfaces with a brief description for each, the objectives that were achieved, and finally the problems and limitations we have faced.

Chapter VIII: Conclusion and Future Work

8.1 Conclusion

Eyes-Up application is an efficient solution to safer driving experience for all drivers. In this report, we have discussed the aims, objectives and the solution that is to be developed during this project to accomplish these aims, the plan of the project and the used methodology of working on this system. In addition to that, the report also described the analysis method of the collected data through a survey, along with presenting how the system was analyzed using use-case diagram, activity diagram, class diagram and entity-relationship diagram. Lastly, for the final part there were the application interfaces displayed with a brief description to get a full representation of the system features.

8.2 Future Work

In the future work we will work on developing the overall performance of the application such as the interfaces, the timer of capturing the face also capturing the yawing of the user for detecting the drowsiness, we also will work on connecting our application to smart watch to increase the features.

References

- [1] J. Stephen Higgins, PhD, Jeff Michael, EdD, Rory Austin, PhD, Torbjörn Åkerstedt, PhD, Hans P. A. Van Dongen, PhD, Nathaniel Watson, MD, Charles Czeisler, PhD, MD, Allan I. Pack, “Asleep at the Wheel—The Road to Addressing Drowsy Driving”
<https://doi.org/10.1093/sleep/zsx001> Published: 25 January 2017
- [2] “Awake” <https://apps.apple.com/us/app/awake-drowsy-driving/id1493097609>
- [3] “Nujible” <https://apps.apple.com/us/app/nujible-stay-awake-focused/id1311332476>
- [4] “Drive Awake” <https://apps.apple.com/us/app/drive-awake/id598888792>

Appendix A - Questionnaire

The figures below show the main questions introduced in the questionnaire along with the results submitted by users. The results are displayed in a pie chart form with percentages.

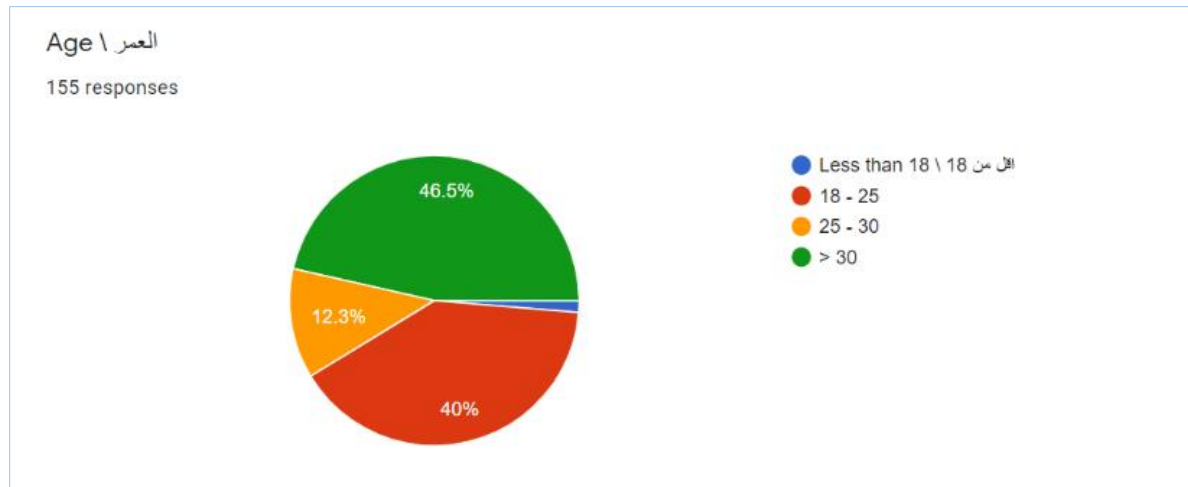


Figure 65 question about age

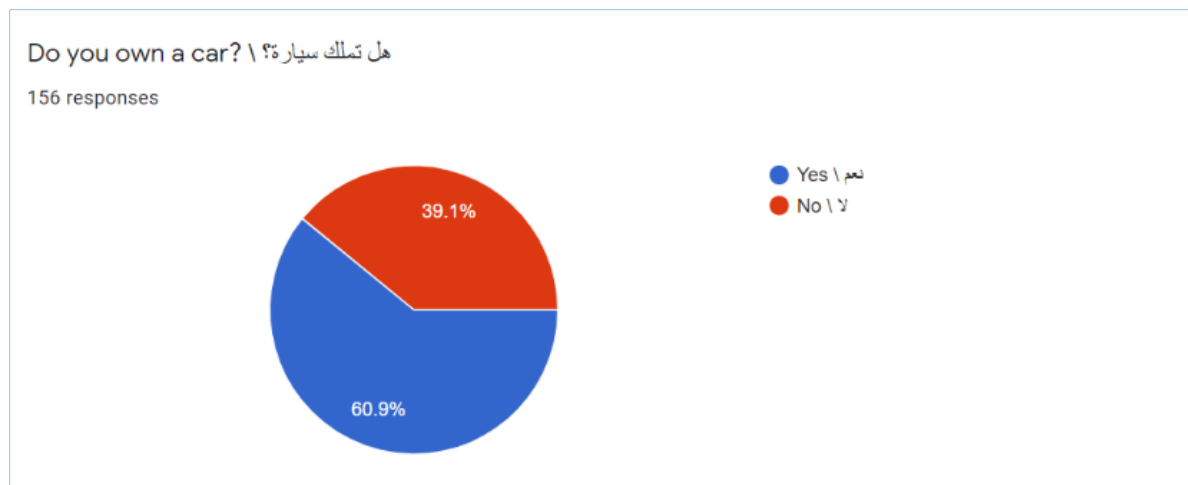


Figure 66 Do you own a car?

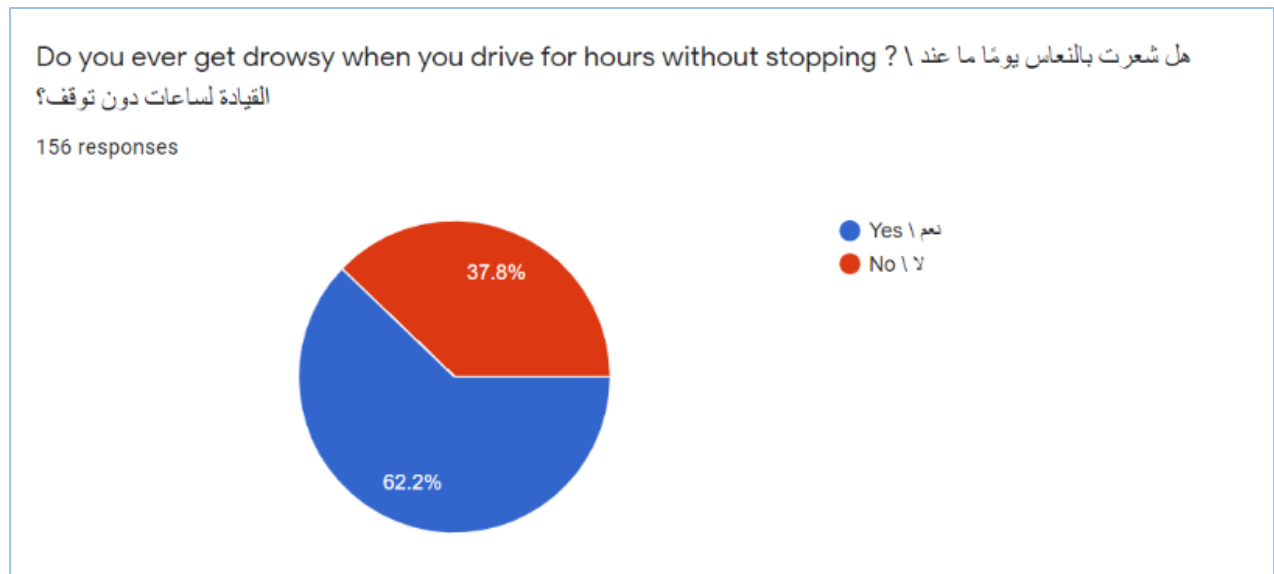


Figure 67 do you get drowsy when you drive for hours?

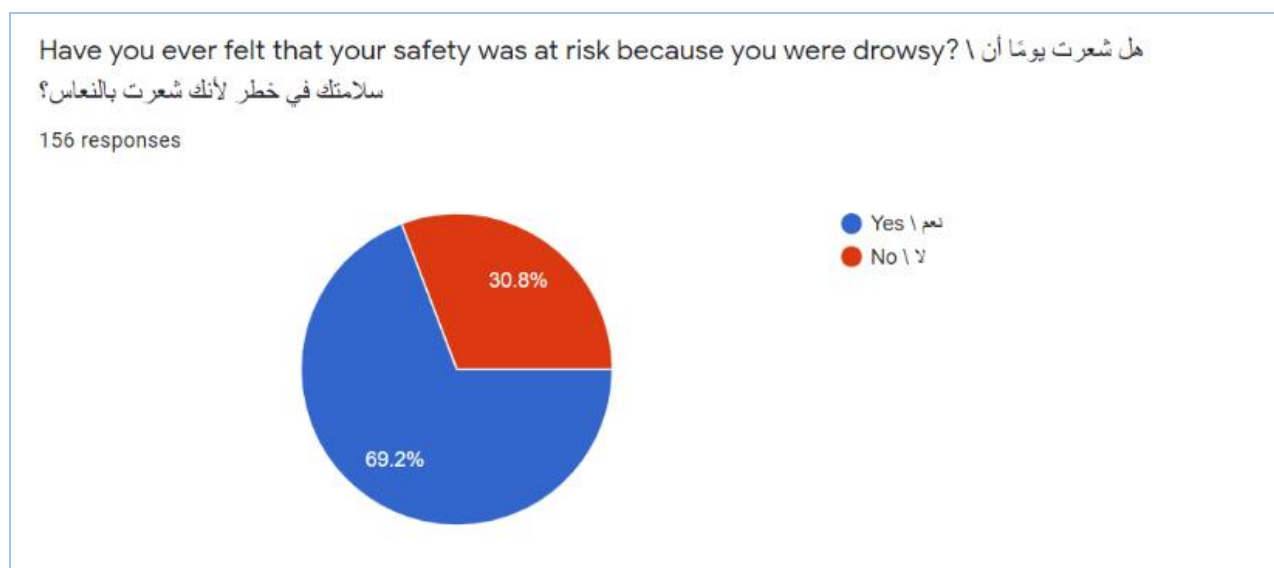


Figure 68 have you ever felt unsafe because you were drowsy?

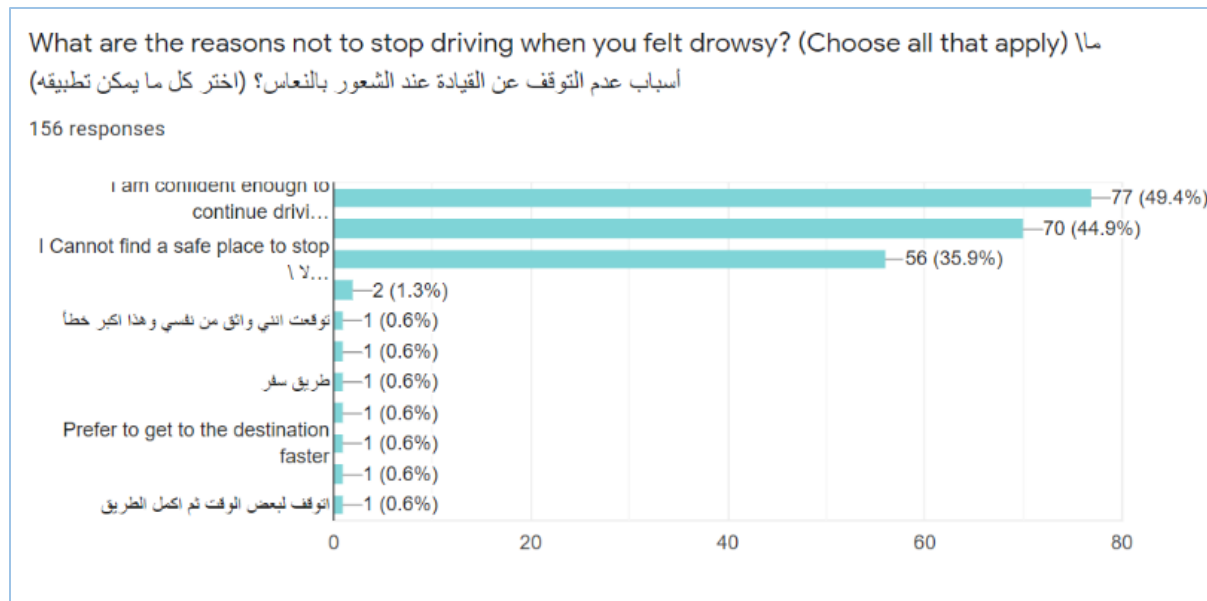


Figure 69 the reasons for not pulling over?

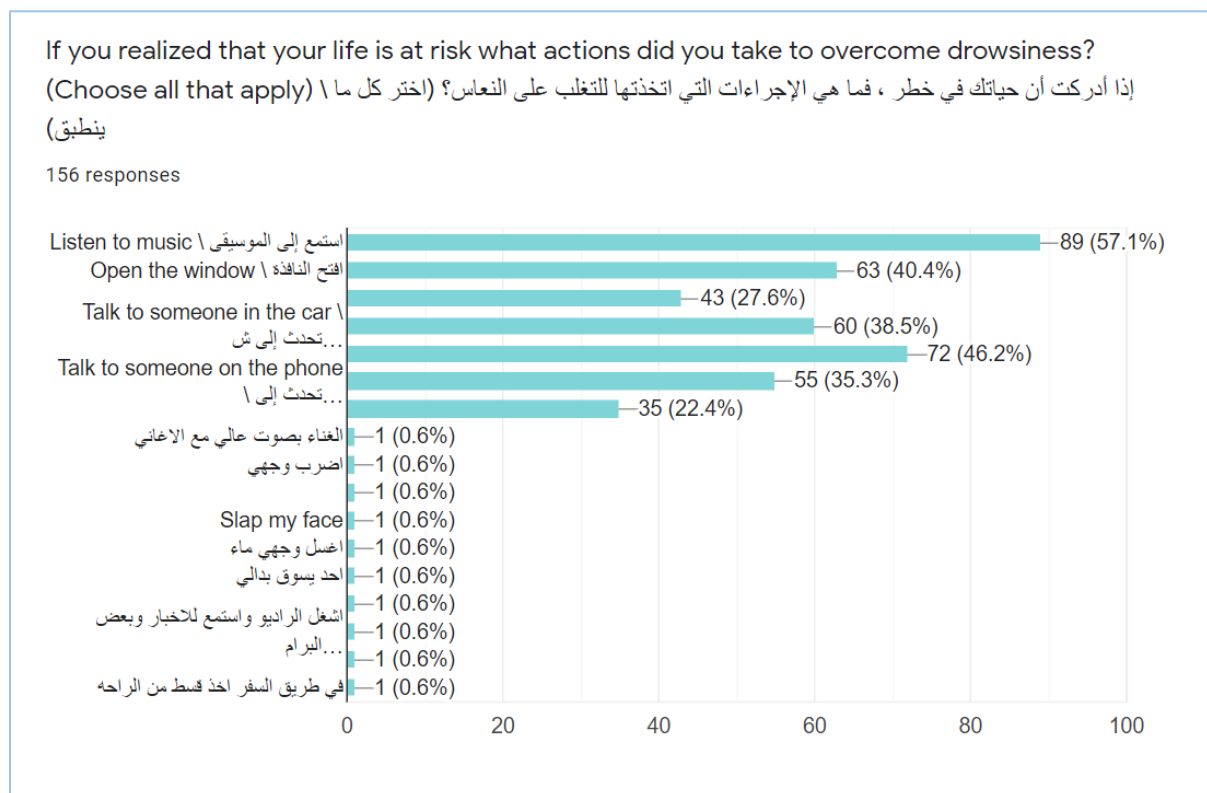


Figure 70 what do you do to overcome drowsiness?

Would you consider using a drowsiness detection application that could wake you up while driving and feeling sleepy? هل تفكر في استخدام تطبيق لاكتشاف النعاس يمكن أن يوقظك أثناء القيادة والشعور بالنعاس؟

156 responses

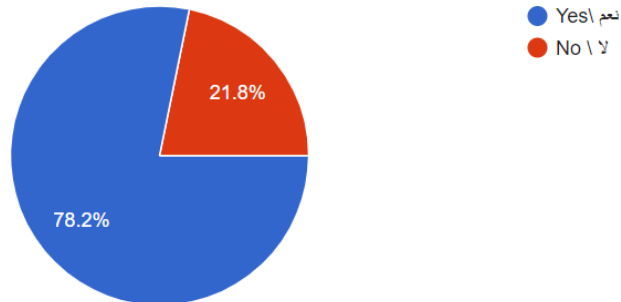


Figure 71 drowsiness detection application

What do you wish for this program to tell you? ماذا تتمنى أن يخبرك هذا البرنامج؟

156 responses

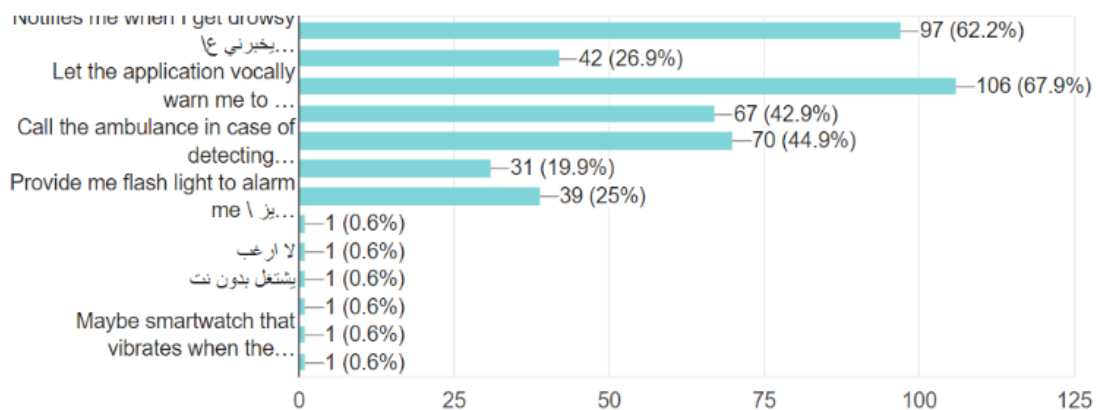


Figure 72 how to warn you from drowsiness

Appendix B - Work Distribution

Table 21 Group work distribution

Chapters:	Team Member:
Chapter 1	All group members participated in documenting and reviewing all chapters of the CPIT-498 report.
Chapter 2	
Chapter 3	
Chapter 4	
Chapter 5	
Chapter 6	
Chapter 7	
Chapter 8	
Chapter 9	