

Ministry of high education

Culture and science city at Oct 6

The High institute of Computer Science & information systems



المعهد العالي لعلوم الحاسب ونظم المعلومات

Graduation Project:

Smart Virtual University: Web-based E-Learning System

Prepared by

- | | |
|-------|---------------------------------|
| 42001 | 1- زياد فؤاد |
| 42002 | 2- ماهيتاب |
| 42003 | 3- |
| 41003 | 4- |
| 41004 | 5- محمود خالد |
| 41020 | 6- هاني محمد |
| 41025 | 7- إسماعيل جمعة كمال ابو القاسم |
| 41049 | 8- اسلام احمد رجب عبد الرحمن |

Assistant:

Eng. : Ahmed Tamam

Supervised by:

Dr : Mohamed Turki

Project No : 5

Academic Year : 2021 – 2020

Acknowledgement

First of all, we must thank Dr. Mohamed Turki and Eng. Ahmed Tamam for their efforts with us through project preparation period and their continues supervision with us until the project is completed properly. Second, we thank the institute specifically Dr. Samy the dean of the institute, and all of the teaching staff on their efforts with us during the study period and wishing them success in the upcoming future.

Abstract

Distance education is a modern way of learning that allows students who are prevented by various circumstances and reasons from accessing the university campus to learn. The meeting between faculty members and students takes place by relying on modern technology via the Internet, multimedia users of audio and video interaction.

Distance education is not a type of luxury or laziness to attend classes. One of the most important goals of distance education is that it gives the right to everyone to learn anywhere in the world. In many cases, many students face challenges that often prevent them from completing the stages of education, especially university. However, distance learning gives them an opportunity to be able to pursue education and pursue their dreams.

Distance education has many advantages, whether for students or for the faculty, as it enables them to gain practical experience in dealing with various technological means, saving the time required to go to classes and return from them, the possibility of attending lessons at any time according to the student's daily schedule, as lessons are usually available. For students on the Internet so that they can read it at any time.

Because of the massive spread of smartphones, nearly 90% of people use smartphones, so what prevents them from educating.

Project Objectives & Summery

- 1- Any student can access the virtual study platform provided that they have a computer and a stable internet connection. No matter what their commute looks like, they can log into a virtual classroom and start learning.
- 2- It allows students to see many courses and register for more than one course.
- 3- Increase student interaction with the content and enable them to view and add comments to specific course content.
- 4- It also allows students to take exams after each course.
- 5- Maintain all data in a secure database and provide multiple methods to the user.

Table of Contents

1- Introduction	22
1.1- Purpose	22
1.2- Benefits	22
1.3- Glossary	22
2- Requirements	23
2.1- User Requirements	23
2.1.1- Functional Requirements	23
2.1.1.1- University	23
2.1.1.2- Instructor	24
2.1.1.3- Student	24
2.1.2- Non-Functional Requirements	24
2.1.2.1- Performance Requirements	25
2.1.2.2- Security Requirements	25
2.1.2.3- Reliability Requirements	25
2.1.2.4- Usability Requirements	26
2.1.2.5- Availability Requirements	26
3- System architecture diagram.....	27
Figure 3 Smart Virtual University System architecture diagram	27
4- UML.....	28
4.1- Goals of UML.....	28
4.2- Data Flow Diagram (DFD).....	28
4.2.1- Context diagram (DFD level 0).....	28
Figure 4.2.1 Smart Virtual University Context Diagram.....	28
4.2.2- Data flow diagram (DFD)	29
Figure 4.2.2 Smart Virtual University DFD Diagram	29
4.3- Use case	29
4.3.1- Actors	29
Figure 4.3 Smart Virtual University use case	30
4.3.2- Use case descriptions	31
Table 1. Presents the Register use case description	31
Table 2. Presents the Verify Registered Students use case description.	31

Table 3. Presents the Add Accounts use case description.	31
Table 4. Presents Provide Technical Support use case description.....	32
Table 5. Presents Add Courses use case description.....	32
Table 6. Presents Follow Attendance use case description.....	32
Table 7. Presents Add Course Plan use case description.	32
Table 8. Presents Add Events use case description.....	33
Table 9. Presents Make Communication use case description.....	33
Table 10. Presents Marks Verification use case description.....	33
Table 11. Presents Register for course use case description.	33
Table 12. Presents Received Notifications use case description.....	34
Table 13. Presents Access Courses use case description.	34
Table 14. Presents Upload Work use case description.....	34
4.4- Activity diagram	35
Figure 4.4 Smart Virtual University Activity Diagram	35
4.5- Sequence Diagram.....	36
Figure 4.5 Smart Virtual University Sequence Diagram	36
4.6- Class Diagram.....	37
Figure 4.6 Smart Virtual University Class Diagram.....	37
Appendix A: Other Desirable Functional Requirements.....	38
1- Instructor	38
2- Student	38

List of Figures

Figure 2.1.1.....	17
Figure 2.1.2.....	18
Figure 2.1.3.....	19
Figure 3.1.1.....	22
Figure 3.1.2.....	22
Figure 3.2.1.....	23
Figure 3.2.2.....	23
Figure 3.2.3.....	24
Figure 3.2.4.....	24
Figure 3.2.5.....	24
Figure 3.2.6.....	24
Figure 3.3.1.....	25
Figure 3.3.2.....	25
Figure 3.3.3.....	25
Figure 3.4.1.....	26
Figure 3.4.2.....	26
Figure 3.4.3.....	26
Figure 3.5.....	27
Figure 3.6.....	28

Chapter 1: Introduction



1.1 An Overview

Mobile app:

A mobile application, also referred to as a mobile app or simply an app, is a computer program or software application designed to run on a mobile device such as a phone, tablet, or watch. Apps were originally intended for productivity assistance such as email, calendar, and contact databases, but the public demand for apps caused rapid expansion into other areas such as mobile games, factory automation, GPS and location-based services, order-tracking, and ticket purchases, so that there are now millions of apps available. Apps are generally downloaded from application distribution platforms which are operated by the owner of the mobile operating system, such as the App Store (iOS) or Google Play Store. Some apps are free, and others have a price, with the profit being split between the application's creator and the distribution platform. Mobile applications often stand in contrast to desktop applications which are designed to run on desktop computers, and web applications which run in mobile web browsers rather than directly on the mobile device.

Most mobile devices are sold with several apps bundled as pre-installed software, such as a web browser, email client, calendar, mapping program, and an app for buying music, other media, or more apps. Some pre-installed apps can be removed by an ordinary uninstall process, thus leaving more storage space for desired ones. Where the software does not allow this, some devices can be rooted to eliminate the undesired apps.

Apps that are not preinstalled are usually available through distribution platforms called app stores. They began appearing in 2008 and are typically operated by the owner of the mobile operating system, such as the Apple App Store, Google Play, Windows Phone Store, and BlackBerry App World. However, there are independent app stores, such as Cydia, GetJar and F-Droid. Some apps are free, while others must be bought.

Usually, they are downloaded from the platform to a target device, but sometimes they can be downloaded to laptops or desktop computers. For apps with a price, generally a percentage, 20-30%, goes to the distribution provider (such as iTunes), and the rest goes to the producer of the app. The same app can, therefore, cost a different price depending on the mobile platform.

Apps can also be installed manually, for example by running an Android application package on Android devices.

Android:

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It was unveiled in 2007, with the first commercial Android device launched in September 2008.

It is free and open source software; its source code is known as Android Open Source Project (AOSP) which is primarily licensed under the Apache License. However most Android devices ship with additional proprietary software pre-installed, most notably Google Mobile Services (GMS) which includes core apps such as Google Chrome, the digital distribution platform Google Play and associated Google Play Services development platform. About 70 percent of Android smartphones run Google's ecosystem; competing Android ecosystems and forks include Fire OS (developed by Amazon) or LineageOS. However the "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use Android branding.

The source code has been used to develop variants of Android on a range of other electronics, such as game consoles, digital cameras, PCs and others, each with a specialized user interface. Some well-known derivatives include Android TV for televisions and Wear OS for wearables, both developed by Google. Software packages on Android, which use the APK format, are generally distributed through proprietary application stores like Google Play Store or Samsung Galaxy Store, or open source platforms like Aptoide or F-Droid.

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of March 2020, the Google Play Store features over 2.9 million apps. The current stable version is Android 10, released on September 3, 2019.

Applications ("apps"), which extend the functionality of devices, are written using the Android software development kit (SDK) and, often, the Java programming language. Java may be combined with C/C++, together with a choice of non-default runtimes that allow better C++ support. The Go programming language is also supported, although with a limited set of application programming interfaces (API). In May 2017, Google announced support for Android app development in the Kotlin programming language.

The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin; in December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android application development.

Other development tools are available, including a native development kit (NDK) for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers, and various cross platform mobile web applications frameworks. In January 2014, Google unveiled an framework based on Apache Cordova for porting Chrome HTML 5 web applications to Android, wrapped in a native application shell. Additionally, Firebase was acquired by Google in 2014 that provides helpful tools for app and web developers.

Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK (Android application package) file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software. Google Play Store allows users to browse, download and update applications published by Google and third-party developers; as of July 2013, there are more than one million applications available for Android in Play Store. As of July 2013, 50 billion applications have been installed. Some carriers offer direct carrier billing for Google Play application purchases, where the cost of the application is added to the user's monthly bill. As of May 2017, there are over one billion active users a month for Gmail, Android, Chrome, Google Play and Maps.

Due to the open nature of Android, a number of third-party application marketplaces also exist for Android, either to provide a substitute for devices that are not allowed to ship with Google Play Store, provide applications that cannot be offered on Google Play Store due to policy violations, or for other reasons. Examples of these third-party stores have included the Amazon Appstore, GetJar, and SlideMe. F-Droid, another alternative marketplace, seeks to only provide applications that are distributed under free and open source licenses.

- **Why the android ?**

There are so many reasons you should choose Android platform for mobile application development.

- **Zero/negligible development cost:**

The development tools like Android SDK, JDK, and Eclipse IDE etc. are free to download for the android mobile application development. Also Google charge a small fee \$25, to distribute your mobile app on the Android Market.

- **Open Source:**

The Android OS is an open-source platform based on the Linux kernel and multiple open-source libraries. In this way developers are free to contribute or extend the platform as necessary for building mobile apps which run on Android devices.

- **Multi-Platform Support:**

In market, there are a wide range of hardware devices powered by the Android OS, including many different phones and tablet. Even development of android mobile apps can occur on Windows, Mac OS or Linux.

- **Multi-Carrier Support:**

World wide a large number of telecom carriers like Airtel, Vodafone, Idea Cellular, AT&T Mobility, BSNL etc. are supporting Android powered phones.

- **Open Distribution Model:**

Android Market place (Google Play store) has very few restrictions on the content or functionality of an android app. So the developer can distribute theirs app through Google Play store and as well other distribution channels like Amazon's app store.

Website:

A website can be defined as a collection of several webpages that are all related to each other and can be accessed by visiting a homepage, using a browser like Internet Explorer, Mozilla, Google Chrome or Opera. In this tutorial, we will explain the concept of website development, from the simplest to the most advanced. It will help novice users to learn all about websites and how they are designed and maintained. At the same time, this tutorial has enough material to help even system administrators to broaden their knowledge about websites.

-Each website has its own URL which is a unique global address called domain name. A URL comprises of –

- The protocol used to access the website
- The subdomain which by default is www.
- The domain name; domain names are normally chosen to have a meaning.
- The suffix name which can be .com, .info, .net, .biz, or country specific. For detailed information
- The directories or in simple words, a folder in the server that holds this website.

Why Do We Need Websites?

Websites primarily act as a bridge between one who wants to share information and those who want to consume it. If you are running a business, then it is almost imperative for you to have a website to broadcast your offerings and reach out to potential clients at a global stage.

The following points explain why it is important to have a website –

- A website is an online brochure where you can advertise your business offers.
- It gives you a platform to reach out to a far-and-wide global customer base.
- If you are a blogger, you have the possibility to influence your readers.
- You can show all your ideas and publish them on a website.
- If you have a business idea, then you don't have to wait. You can straightaway open an online shop and sell your products or services online. An added advantage is that the online shop will be open 24/7 for your clients, throughout the year.
- You can communicate with your customers, giving them an opportunity to express themselves.
- You can provide valuable customer support by having a trouble-ticket system.
- If you have an official website with a domain, then you can have your personalized email. For example, **info@tutorialspoint.com** (it is much better than **florjan.llapi@yahoo.com**).

How to Setup a Website ?

A website is composed of several elements and while setting up a website, you would have to take care of each of them.

- To set up a website and make it live, you should first purchase a hosting plan.
- Select a domain name for this website.
- Point the DNS records to the server or the hosting provider.
- Develop the content that you want to publish on the website.
- Check if you need to purchase a public certificate and install it.
- Publish the webpage on the Internet.

In the subsequent chapters of this tutorial, we will discuss each of these steps in detail.

skills required to set up a website

The skills required to set up a website can vary from very basic to the most advanced. If you are going to set up a professional website for a global audience, then you should have the following set of skills or you would have to hire a group of people to do this job for you.

- **Content Experts**

Content experts supply the content that is to be published on the website. They design the content as per the requirement of the target audience and then, edit and polish the content before it gets published.

Content experts normally rely on the expertise of the site designer and the webmaster. Note that the content can be text, data, images, audio or links.

- Website Designer

A web designer is a technical person who designs and maintains the Graphical User Interface (GUI) of the website. For example, where the buttons should be placed, how the images are to be displayed, etc.

- Graphic Designers

Graphic designers develop image files that are to be included in the website. These professionals have a keen understanding of developing suitable graphics for the web environment.

- Web Developers

Web developers create the program codes to manipulate the supplied content, based on the site design established by the website designer. A web programmer should use a set of programming languages to compile specific functions that the webpages should do in the background. Here is a set of important programming languages that a web programmer must be good at –

- HTML / XHTML – These are the markup languages which you will use to build your website. A web programmer must have a good understanding of HTML and XML.
- PHP – It is a popular programming language to develop webpages.

1.2 Advantages & Disadvantages

1.3 Project Challenges

Chapter 2: Software Requirements Specification.

1- Introduction

The following section provides an overview of the derived Software Requirements Specification (SRS) for the subject Smart Virtual University Web Site. To begin with, the purpose of the document is identified the Functional requirements and the non-functional requirements, Subsequently, Benefits of the system. to conclude, a complete document overview is provided to facilitate increased readercomprehension and navigation.

1.1- Purpose

The purpose of this SRS is to outline both the functional and non- functional requirements of the subject Smart Virtual University Web Site. Functional requirements are listed first, according to their relationship to the University, Instructors, and Students. The non-functional requirements that pertain to safety, security, the interface, human engineering, qualification, operation, maintenance and performance are subsequently presented. The functional requirements have been specified using a natural language description.

1.2- Benefits

This software system will be an online academic portal for any university wishing to manage their academic needs online. More specifically to design and develop a simple and intuitive system which shall cater the academic needs of any institute, the system allows instructors to create, deliver, and manage web-based components for courses. It can be used to add online elements to a traditional course, or to develop completely online courses with few or no face-to-face meetings.

1.3- Glossary

2- Requirements

The following section presents the complete set of User Requirements and System Requirements with functional and non-functional requirements identified for the subject Smart Virtual University Web Site. Functional requirements are listed first, according to their relationship to the University, Instructors, and Students. The non- functional requirements that pertain to safety, security, the interface, human engineering, qualification, operation, maintenance and performance are subsequently presented. The functional requirements have been specified using a natural language description.

2.1- User Requirements

The following section presents the complete set of User Requirements and System Requirements with functional and non-functional requirements identified for the subject Smart Virtual University Web Site. Functional requirements are listed first, according to their relationship to the University, Instructors, and Students. The non- functional requirements that pertain to safety, security, the interface, human engineering, qualification, operation, maintenance and performance are subsequently presented. The functional requirements have been specified using a natural language description.

2.1.1- Functional Requirements

This subsection presents the identified functional requirements for the subject Smart Virtual University Website

2.1.1.1- University

- A University shall be able to register on the platform.
- A University shall be able to add accounts for instructors.
- A University shall be able to verify registered students.
- A University shall be able to provide technical support
-

2.1.1.2- Instructor

- An Instructor shall add and upload courses contents.
- An Instructor shall add a lesson plan.
- An Instructor shall be able to communicate with students in audio, video, chatting, and by sending messages.
- An Instructor shall be able to follow the attendance of the student.
- An Instructors shall add events (Quizzes, Assignments, Meetings, etc.) using a calendar.
- An Instructor shall be able to view and grade students work.

2.1.1.3- Student

- A Student shall be able to register to their courses.
- A Student shall be able to access courses contents whenever contents become available.
- A Student shall be able to view instructor's lesson plan
- A Student shall be able to receive notifications about and view new events.
- A Student shall be able to communicate with instructor in audio, video, chatting and messages.
- A Student shall be able to be notified about work need to be assigned.
- A Student shall be able to upload and assign his work.
- A Student shall be able to view his grades.

2.1.2- Non-Functional Requirements

This subsection presents the identified non-functional requirements for the subject Smart Virtual University Website. The subcategories of non- functional requirements given are Performance, Security, Reliability, Usability and Availability.

2.1.2.1- Performance Requirements

- The database shall be able to accommodate a minimum of 10,000 records of students.
- The software shall support use of multiple users at a time.
- There are no other specific performance requirements that will affect development.

2.1.2.2- Security Requirements

- Utilize certain cryptographic techniques
- Keep specific log or history data sets
- Restrict communications between some areas of the program
- Check data integrity for critical variables
- Later version of the software will incorporate encryption techniques in the user/license authentication process.
- The software will include an error tracking log that will help the user understand what error occurred when the application crashed along with suggestions on how to prevent the error from occurring again.
- Communication needs to be restricted when the application is validating the user or license.

2.1.2.3- Reliability Requirements

- All data storage for user variables will be committed to the database at the time of entry.
- Data corruption is prevented by applying the possible backup procedures and techniques.

2.1.2.4- Usability Requirements

- A logical interface is essential to an easy-to-use system, speeding up common tasks.

2.1.2.5- Availability Requirements

- All cached data will be rebuilt during every startup.
- There is no recovery of user data if it is lost.
- Default values of system data will be assigned when necessary.

3- System architecture diagram

As shown in figure (3) that depicts a system that used to abstract the software system's overall outline and build constraints, relations, and boundaries between components. It provides a complete view of the physical deployment of the evolution roadmap of the software system, and focuses on the structure along with the technological requirements, external services, servers, and databases.

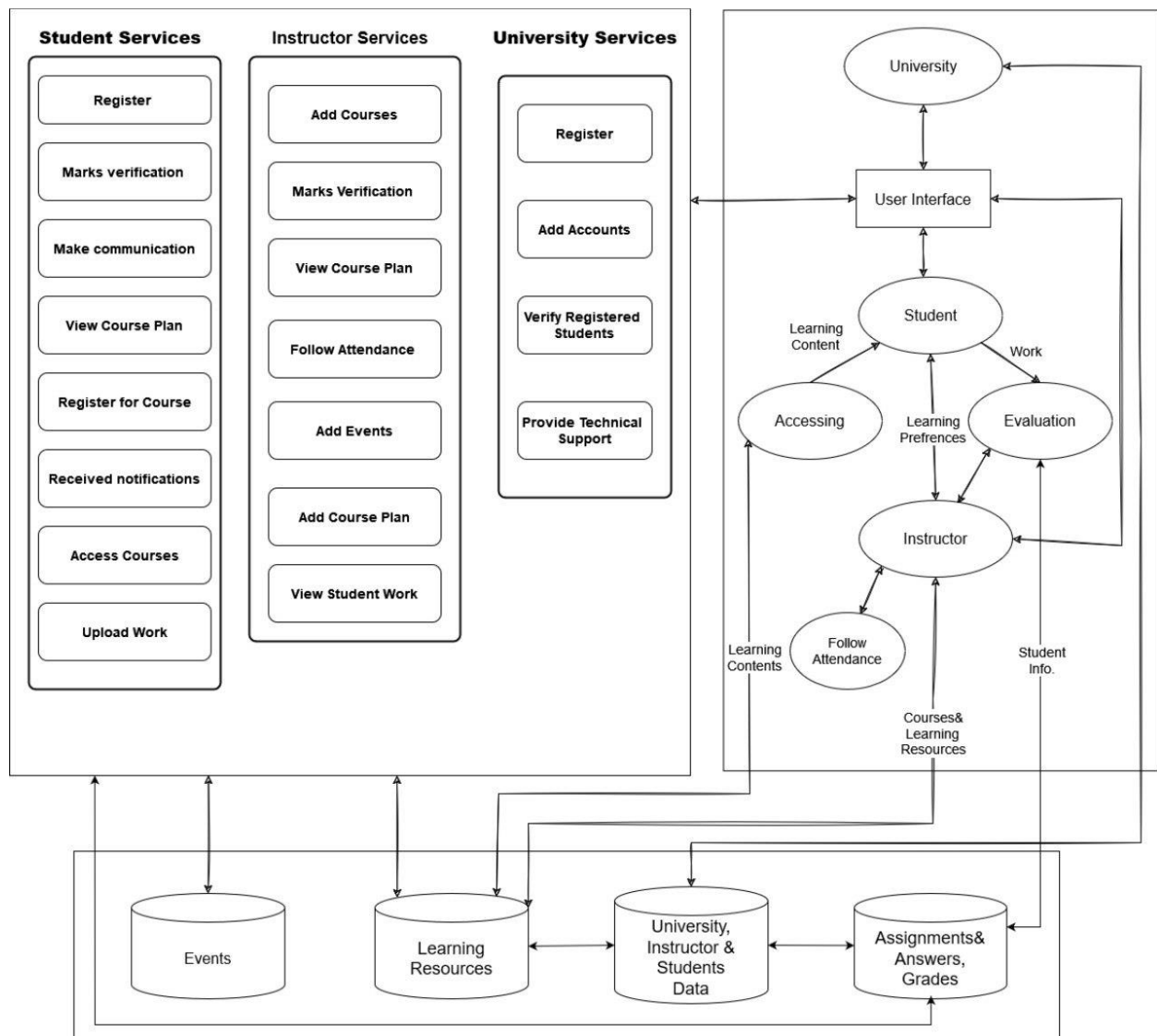


Figure 3 Smart Virtual University System architecture diagram

4- UML

4.1- Goals of UML

The UML was invented primarily to address the challenges faced in the design and architecture of complex systems. The basic objectives or goals behind UML modeling are (James, Unified Modeling Language Reference):

- Define an easy to use and visual modeling language for modeling a system's structure
- Provide extensibility
- Be language and platform independent so that it can be used for modeling a system
- irrespective of the language and platform in which the system is designed and implemented
- Provide support for Object Orientation, design and apply frameworks and patterns.

4.2- Data Flow Diagram (DFD)

4.2.1- Context diagram (DFD level 0)

This is the Zero Level DFD of E-learning Platform, where we have elaborated the high-level process of E-learning. It's a basic overview of the full E-learning platform or process being analyzed or modeled.

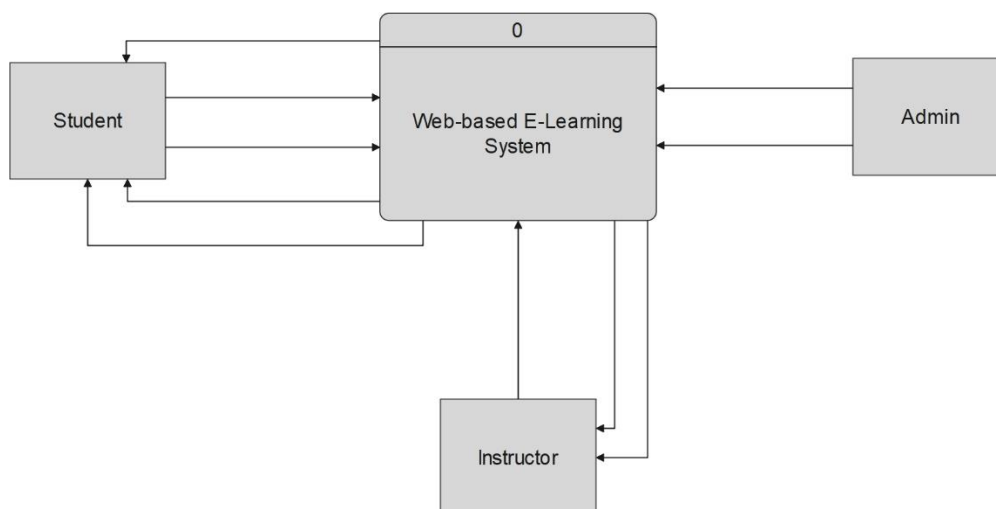


Figure 4.2.1 Smart Virtual University Context Diagram

4.2.2- Data flow diagram (DFD)

DFD of e-learning platform shows how the system is consisted sub-systems (processes), each of which works with one or more of the data flows to or from another agent, and which together provide all of the features of the e-learning system as an entire. It also identifies inside data stores of college student, subject, test, specialties, task that must be present, DFD provides a more detailed large of pieces of the level 0 DFD.

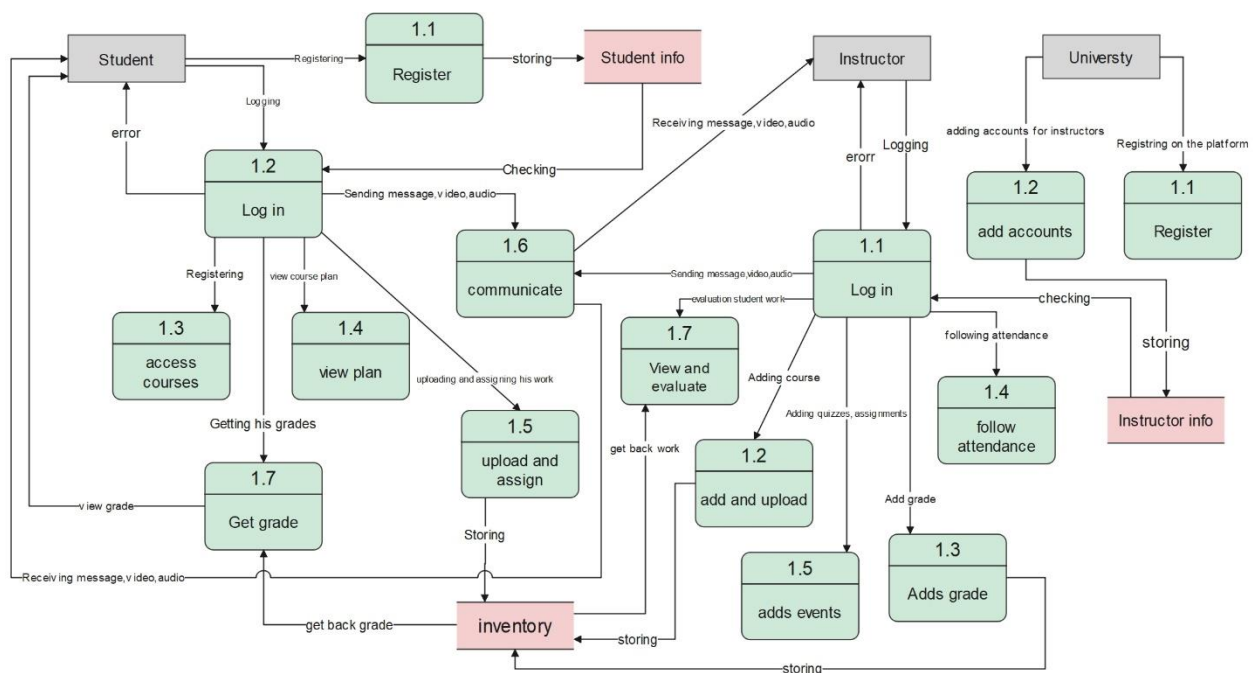


Figure 4.2.2 Smart Virtual University DFD Diagram

4.3- Use case

This subsection extends upon the functional requirements given in Section 2.1 through the presentation of detailed use cases. To facilitate an unambiguous and clear view of how the end-users interact with the subject Smart Virtual University, the actors (end-users) involved in the use cases, a use case diagram and detailed use case descriptions are provided.

4.3.1- Actors

There are three actors in the Smart Virtual University platform, University, Instructor, and Student.

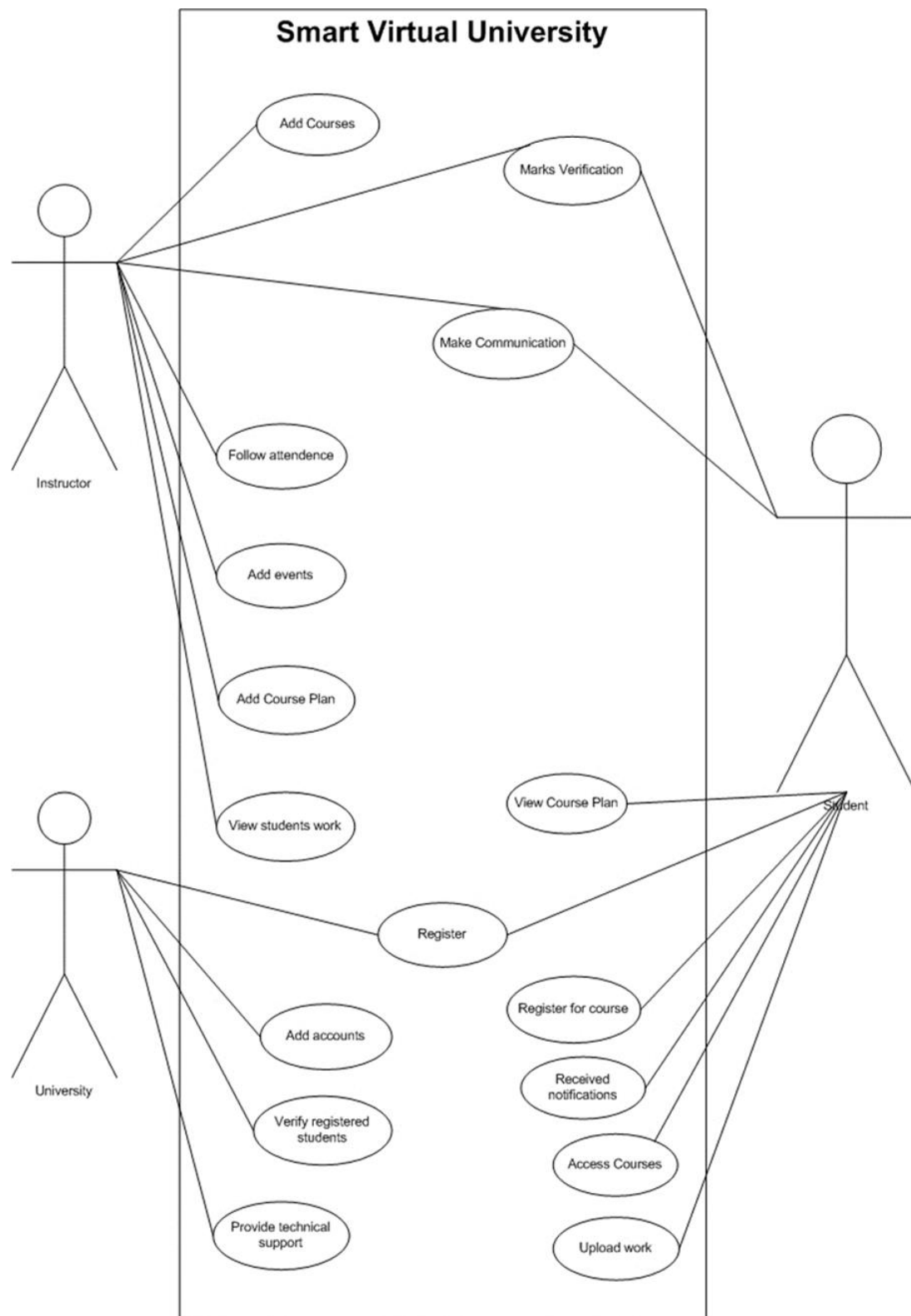


Figure 4.3 Smart Virtual University use case

4.3.2- Use case descriptions

Table 1. Presents the Register use case description

Use Case	Register.
Actors	University, Student.
Description	<ul style="list-style-type: none">- University must register at the platform to be able to use services provided- Student must register at the platform to access services provided by the platform.
Data	<ul style="list-style-type: none">- University: University Name, Password, E-mail, Phone Number- Student: E-mail, Password, Phone Number, National ID, University ID (Only Students).
Response	<ul style="list-style-type: none">- University and Instructor registration process confirmed- Registered students have to be confirmed by University before they can access the platform

Table 2. Presents the Verify Registered Students use case description.

Use Case	Verify Registered Students.
Actors	University.
Description	After the registration of students, university have to make sure that registered students are already on their databases.
Data	Students' information.
Response	Accepting or rejecting a student

Table 3. Presents the Add Accounts use case description.

Use Case	Add Accounts.
Actors	University.
Description	University can create accounts for its instructors.
Data	Instructor Full Name, E-mail, Password, Phone Number.
Response	Instructor has been added.

Table 4. Presents Provide Technical Support use case description.

Use Case	Provide Technical Support.
Actors	University.
Description	University could provide technical support in case there is a need for passwordresetting or any inquiry, etc.
Data	
Response	Type of technical support

Table 5. Presents Add Courses use case description.

Use Case	Add Courses.
Actors	Instructor.
Description	Instructor has the ability to add his courses by adding descriptions and uploading hiscourse contents even text files or video files.
Data	Description that may be provided by instructor, course files.
Response	Added New Course.

Table 6. Presents Follow Attendance use case description.

Use Case	Follow Attendance.
Actors	Instructor.
Description	Instructor has the ability to follow students' attendance.
Data	Student Name, Date Time.
Response	Student attendance log.

Table 7. Presents Add Course Plan use case description.

Use Case	Add Course Plan.
Actors	Instructor.
Description	Instructor has the ability to add a time plan for his courses.
Data	Date, Description for every date time
Response	Plan of course has been added.

Table 8. Presents Add Events use case description.

Use Case	Add Events.
Actors	Instructor.
Description	Instructor has ability to add events for (Quizzes, Assignments, Meetings, etc.) using a calendar by which the instructor set the event time and adding notes.
Data	Date time, Event type, Notes
Response	Event Added Successfully.

Table 9. Presents Make Communication use case description.

Use Case	Make Communication
Actors	Student, Instructor.
Description	Student and Instructor has the ability to communicate with each other through Audio, Video and Messages.
Data	
Response	

Table 10. Presents Marks Verification use case description.

Use Case	Marks Verification
Actors	Instructor, Student
Description	After the upload of student work, instructor has the ability to grade this work and the student have the ability to view his grade.
Data	Student work, Grade
Response	The grade of student work.

Table 11. Presents Register for course use case description.

Use Case	Register for course
Actors	Student
Description	Student can register for course
Data	Student Name, Year, ID
Response	Course Registered.

Table 12. Presents Received Notifications use case description.

Use Case	Received Notifications
Actors	Student
Description	Student can receive notifications about any event occur.
Data	
Response	Notifications of new events.

Table 13. Presents Access Courses use case description.

Use Case	Access Courses
Actors	Student
Description	Student can view and access registered courses on the platform.
Data	
Response	Registered Courses

Table 14. Presents Upload Work use case description.

Use Case	Upload Work.
Actors	Student
Description	Student can upload his work to instructor
Data	Uploaded files
Response	Upload status.

4.4- Activity diagram

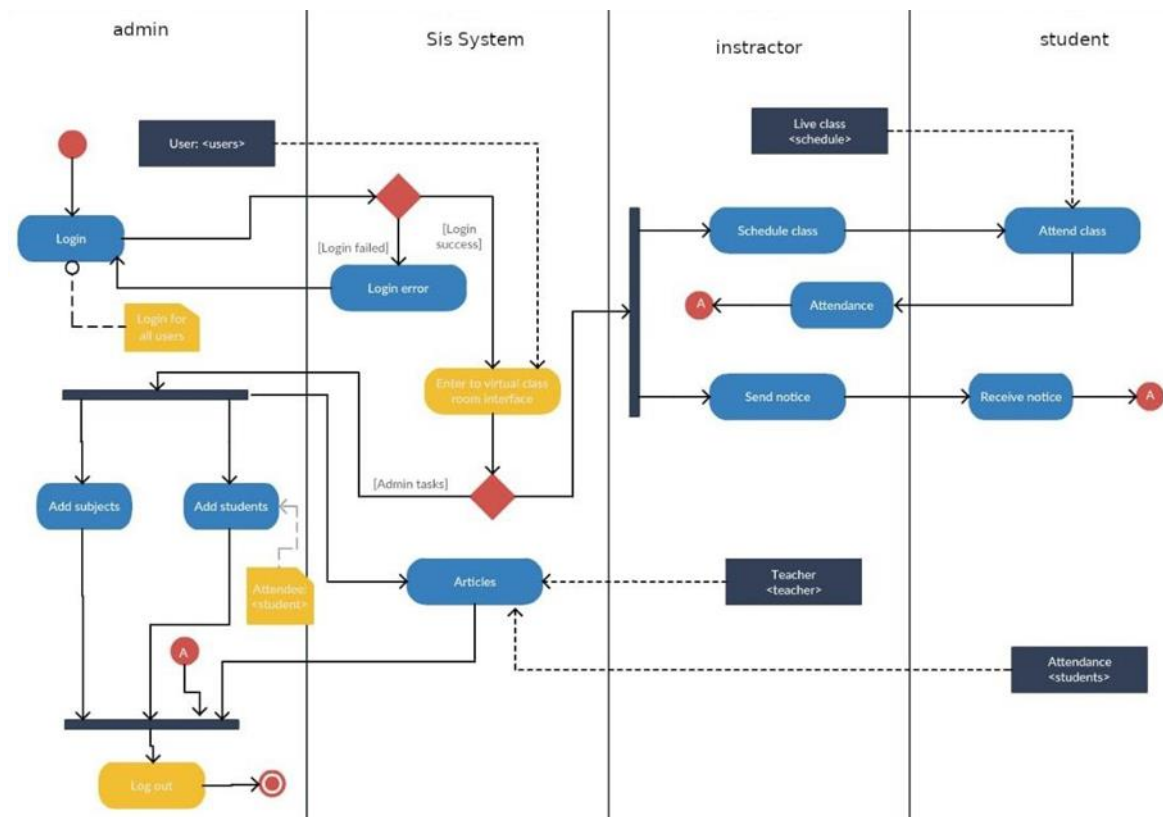


Figure 4.4 Smart Virtual University Activity Diagram

4.5- Sequence Diagram

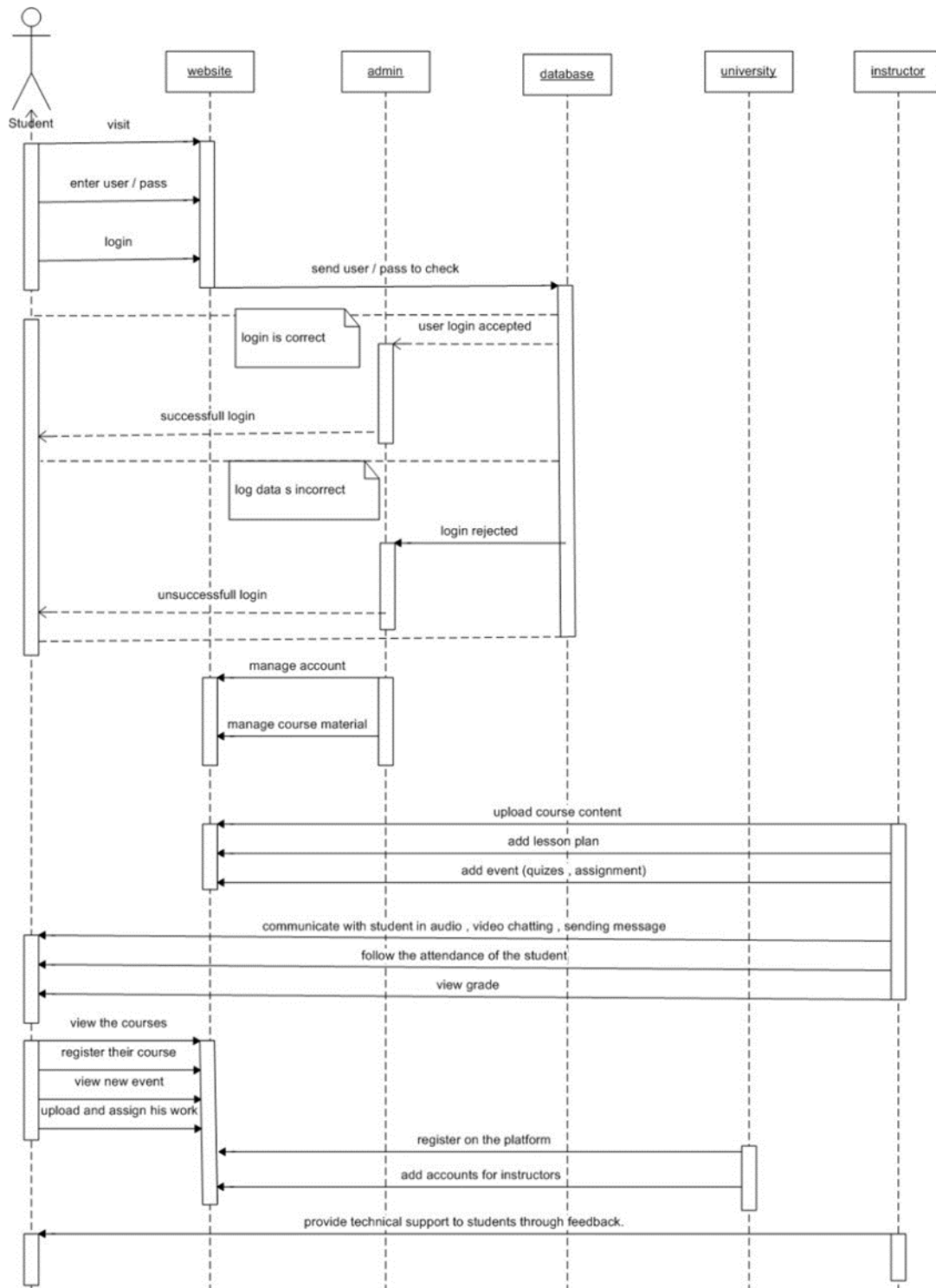


Figure 4.5 Smart Virtual University Sequence Diagram

4.6- Class Diagram

As shown in figure (4.6) E-learning platform Class Diagram describes the structure of a E-learning platform classes, their attributes, operations (or methods), and the relationships among objects.

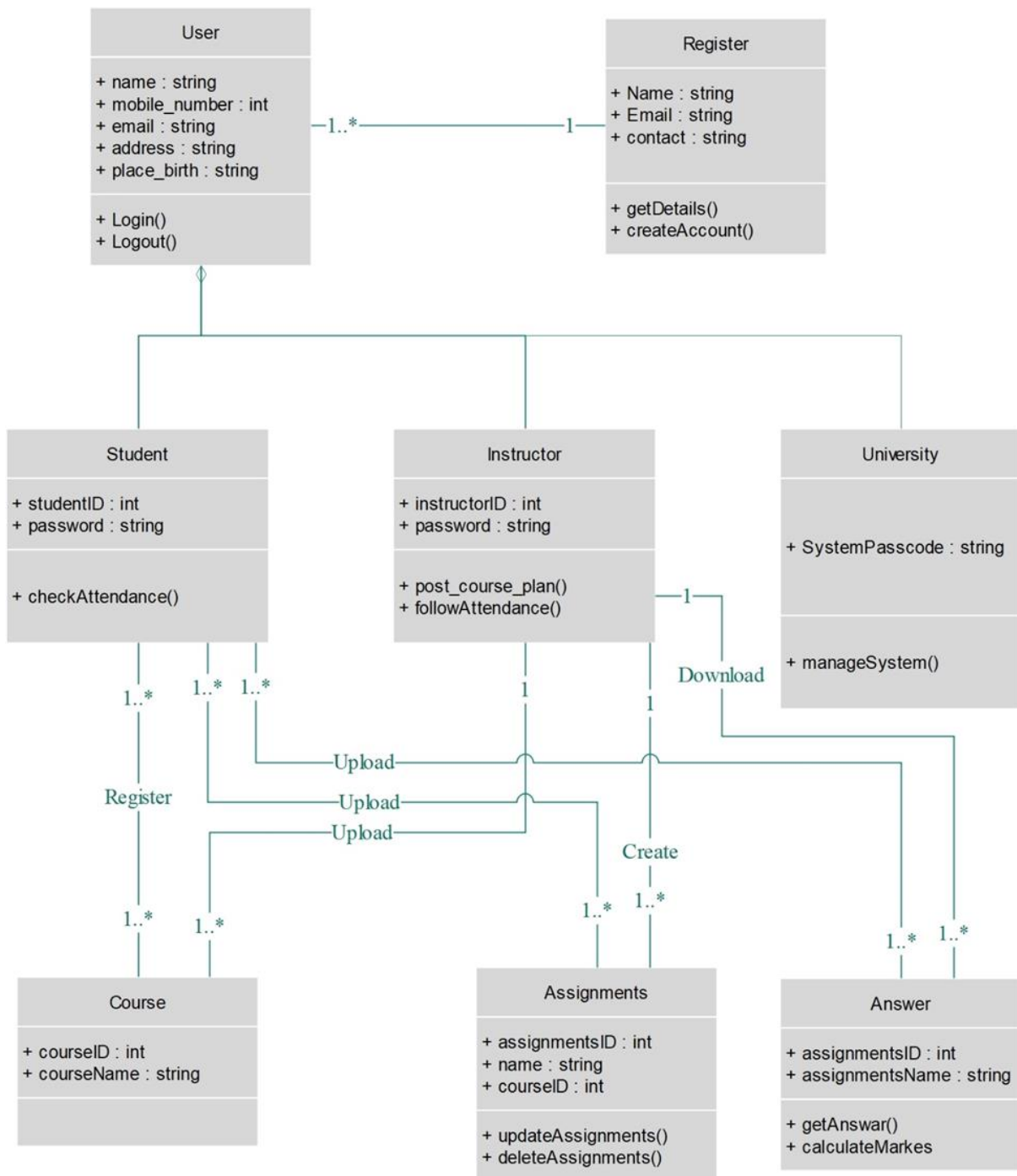


Figure 4.6 Smart Virtual University Class Diagram

Appendix A: Other Desirable Functional Requirements

1- Instructor

- An instructor should view and edit his profile page.
- An instructor should edit courses contents.
- An instructor should view everyone registered in his course.
- An instructor should view and edit course description.
- An instructor should view and set study groups.
- An instructor should be able to manage question banks
- An instructor should be able to add and view comments on specific content of a course.

2- Student

- A Student shall be able to edit his profile page.
- A Student shall be able to view instructor's profile page.
- A Student shall be able to view and add comments to a specific content of a course.
- A Student shall be able to join study groups.
- A Student shall be able to view question banks.

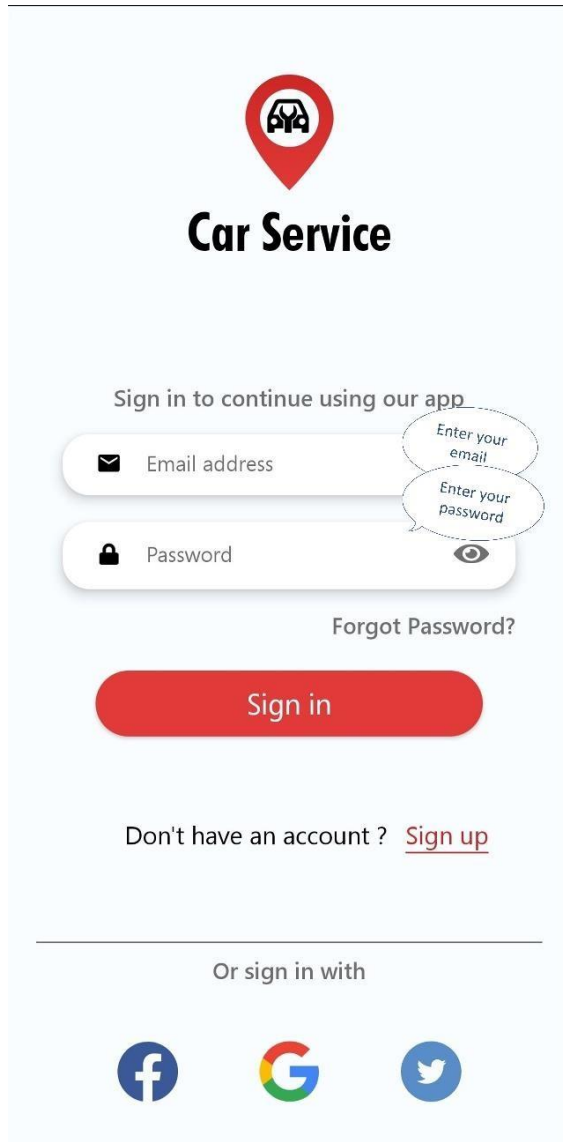
Project tools:

- 1- Android Studio.
- 2- Firebase.
- 3- Adobe After Effect.
- 4- Adobe XD.
- 5- Adobe Illustrator.
- 6- Adobe Photoshop.

Chapter 3: Project implementation

3.1 Registration & Sign in

Sign in



The sign-in form for 'Car Service' features a red location pin icon with a car icon inside. Below the title, it says 'Sign in to continue using our app'. There are two input fields: 'Email address' and 'Password', both with icons (envelope and lock) and placeholder text. A red 'Sign in' button is at the bottom. Below the button, it says 'Don't have an account ? [Sign up](#)'. At the bottom, it says 'Or sign in with' followed by Facebook, Google, and Twitter icons.

Car Service

Sign in to continue using our app

Email address

Password

Forgot Password?

Sign in

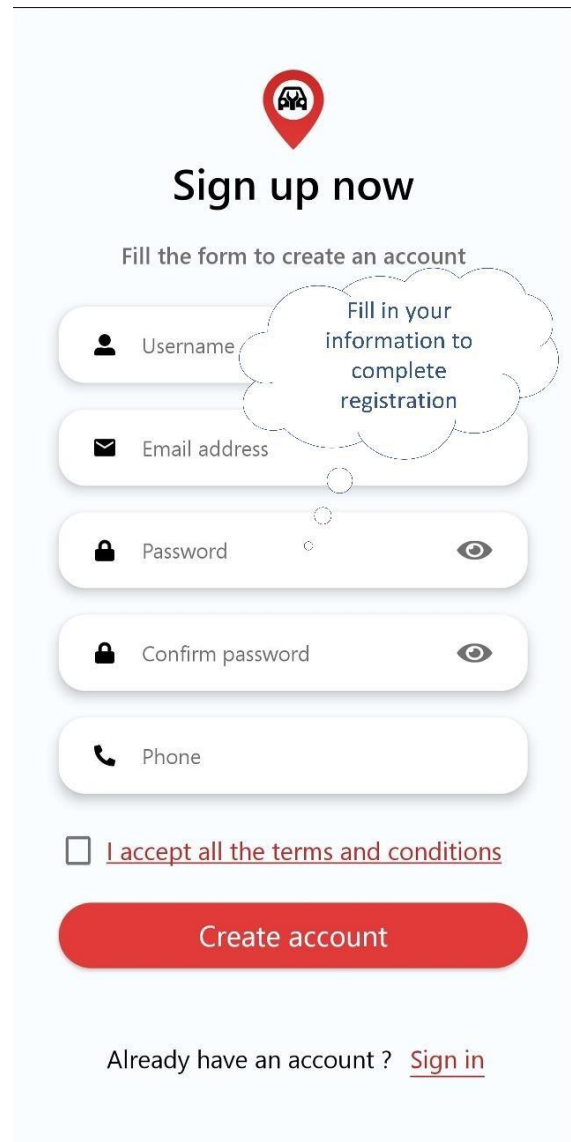
Don't have an account ? [Sign up](#)

Or sign in with

[Facebook](#) [Google](#) [Twitter](#)

Figure 3.1.1

Registration



The registration form for 'Car Service' features a red location pin icon with a car icon inside. Below the title, it says 'Sign up now' and 'Fill the form to create an account'. There are five input fields: 'Username', 'Email address', 'Password', 'Confirm password', and 'Phone'. Each field has an icon (person, envelope, lock, lock, and phone) and placeholder text. A red 'Create account' button is at the bottom. Below the button, it says 'Already have an account ? [Sign in](#)'. At the bottom, it says 'Or sign in with' followed by Facebook, Google, and Twitter icons.

Sign up now

Fill the form to create an account

Username

Email address

Password

Confirm password

Phone

☐ I accept all the terms and conditions

Create account

Already have an account ? [Sign in](#)

Or sign in with

[Facebook](#) [Google](#) [Twitter](#)

Figure 3.1.2

- Enter your email and password to sign in to your profile.
- You can also sign in with Facebook, Gmail or Twitter.
- Or you can create account if you do not have an account.

3.2 Home page & Side bar



Figure 3.2.1

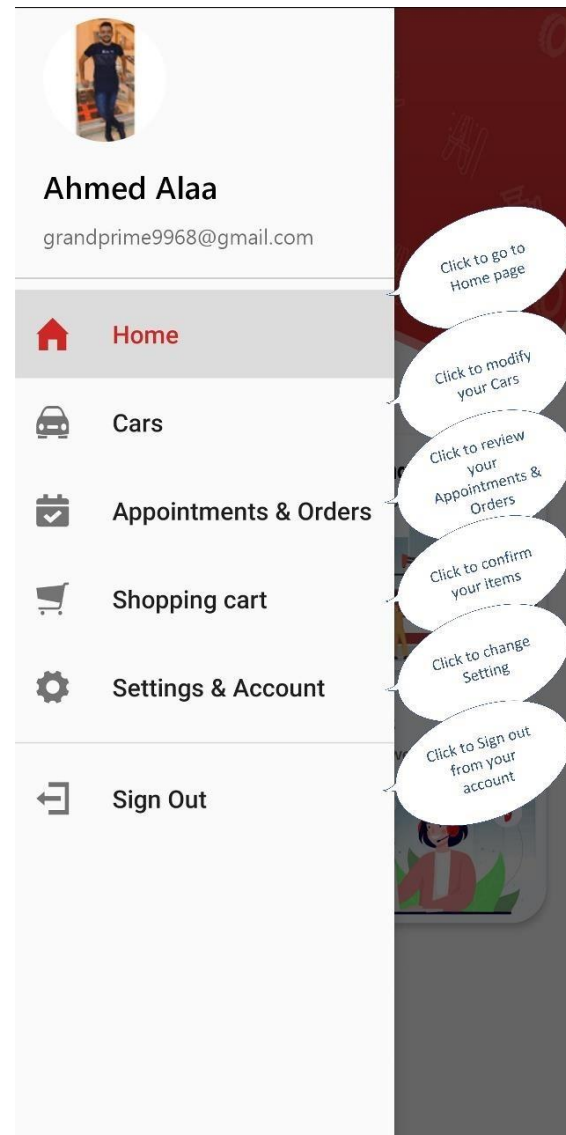


Figure 3.2.2

- After sign in, Home page will appear for the first page.
- Click Side bar for more options.
- Options like: (Cars, Appointments & Orders, Shopping cart, Setting & Account, Sign out).

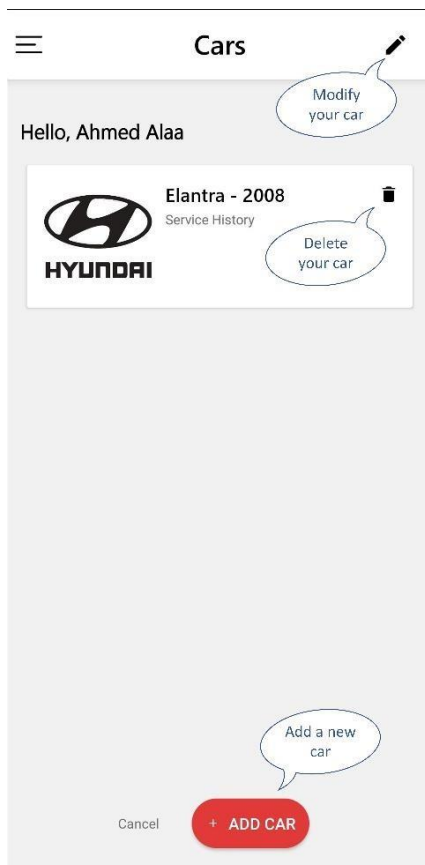


Figure 3.2.3

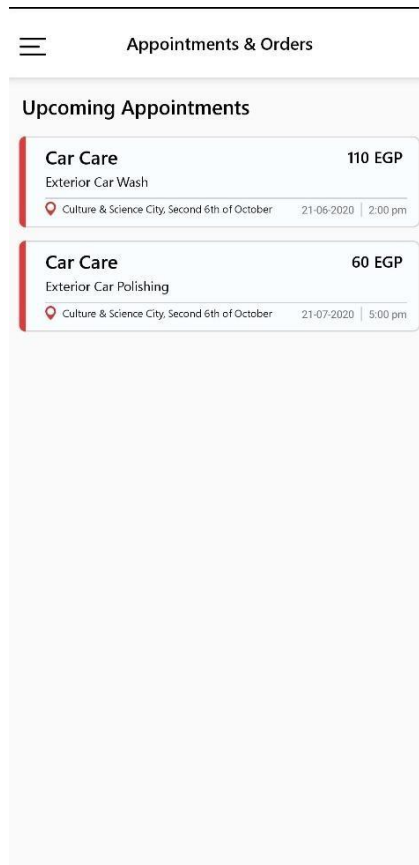


Figure 3.2.4

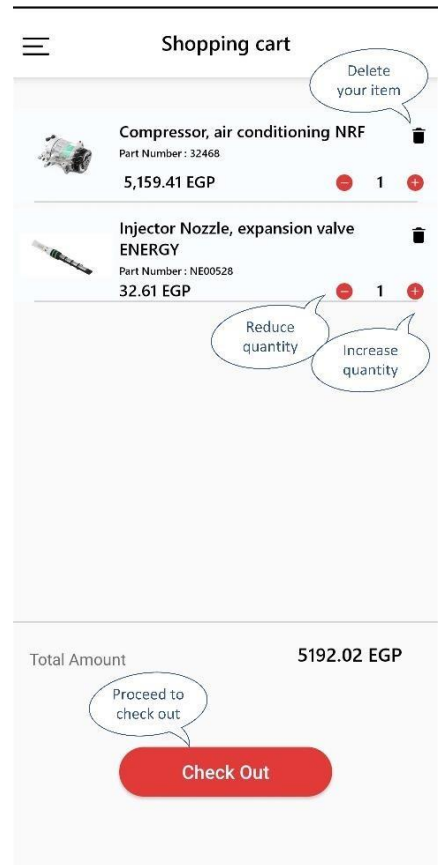


Figure 3.2.5

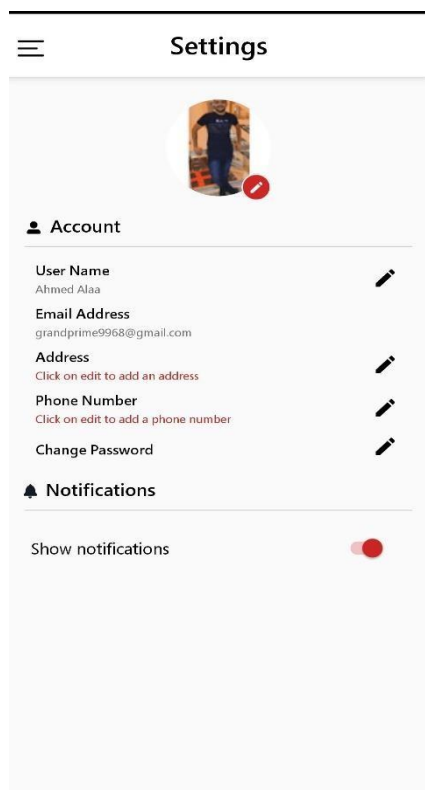


Figure 3.2.6

- Cars page allows you to add or modify your cars.
- You can review your Appointments & Orders by clicking on its button.
- Check your pick up items to confirm your shopping or modify it.
- You can control your account setting by clicking on Setting & Account button.

3.3 Car Center

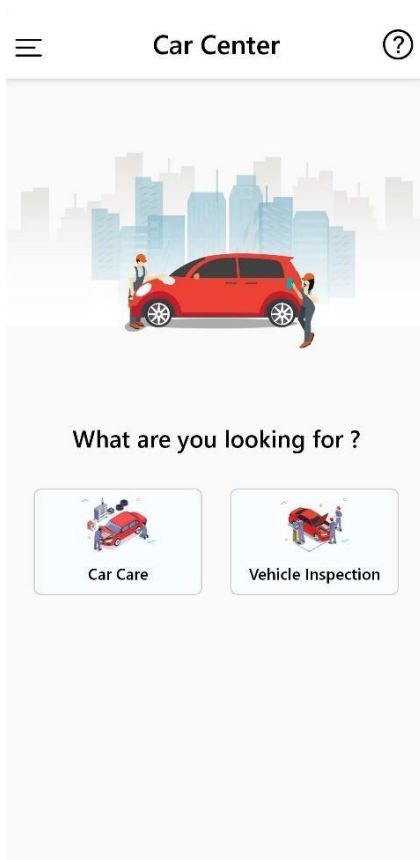


Figure 3.3.1

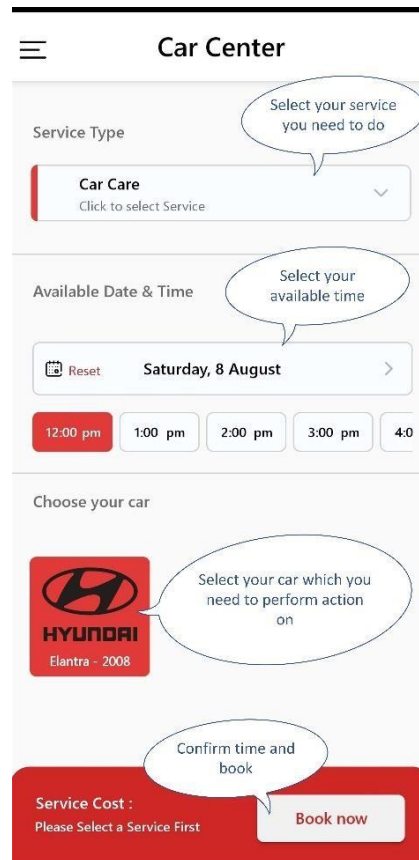


Figure 3.3.2

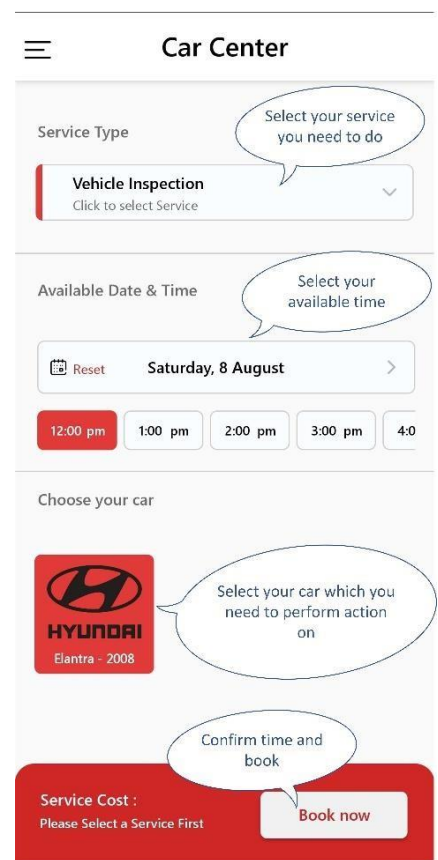


Figure 3.3.3

- In case of go to Car Center, just click on Car Center button and choose between car care or vehicle inspection.
- After decide what to do in car care or vehicle inspection, select service you need, select the available time, and then confirm the book.

3.4 Delivery

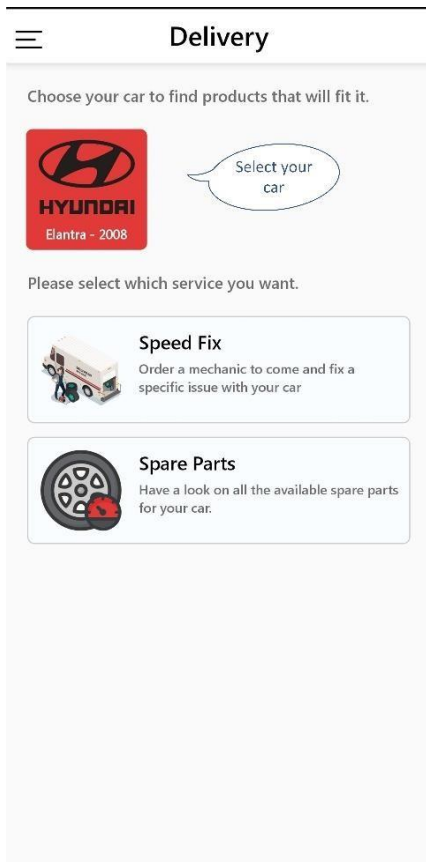


Figure 3.4.1

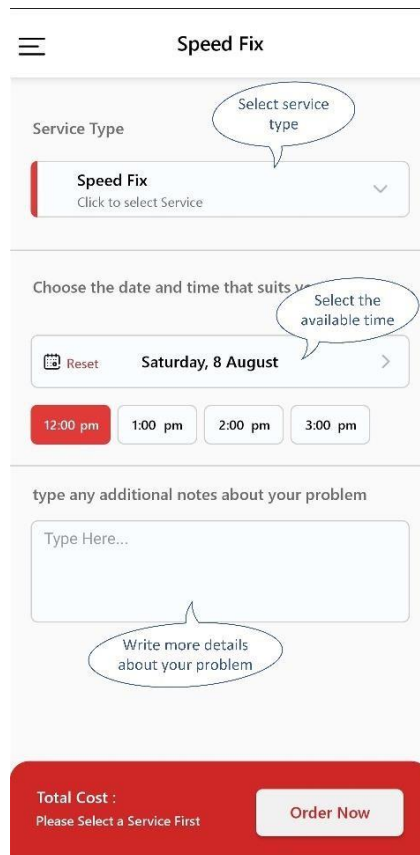


Figure 3.4.2

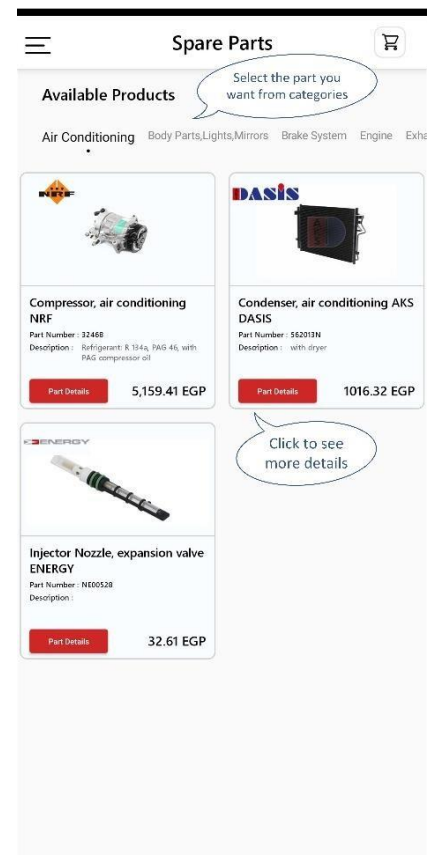


Figure 3.4.3

- Instead of going to car center, you can do speed fix to your car or purchase spare parts.
- Just click Delivery button, select your car, select speed fix, select service type, select the available time and make the order.
- To purchase a part for your car, click on Spare Parts, select the part you need from categories and add to carts.

3.5 Emergency

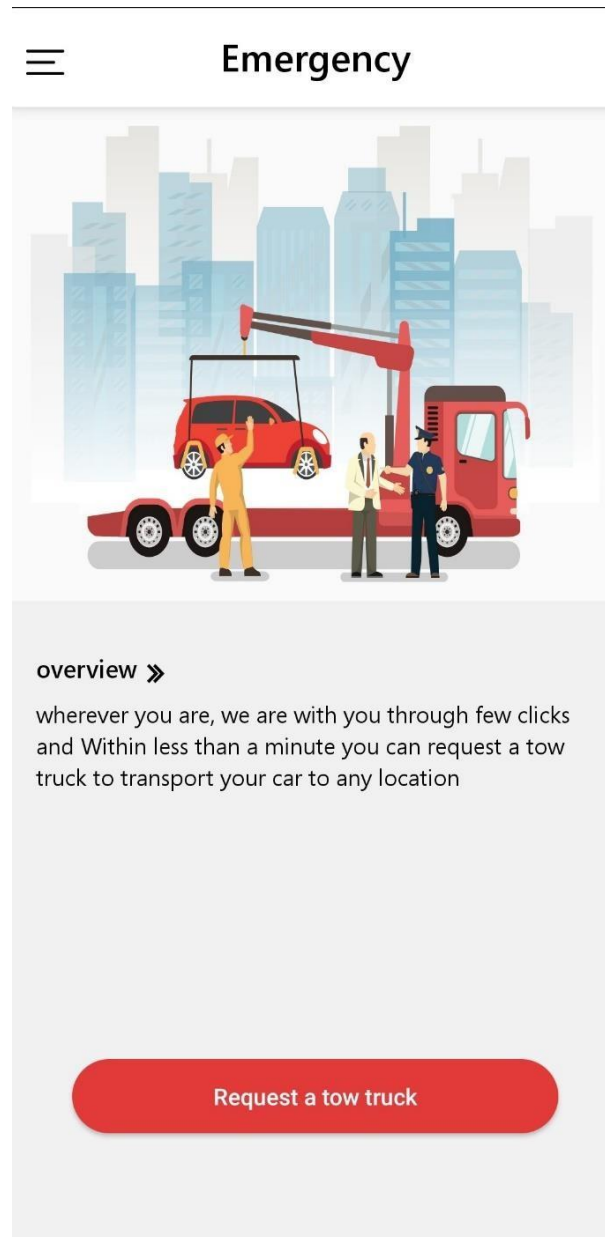


Figure 3.5

- If you face an emergency situation on road and your car unfortunately stopped, you can use emergency service and request a tow truck to solve the problem.

3.6 About us

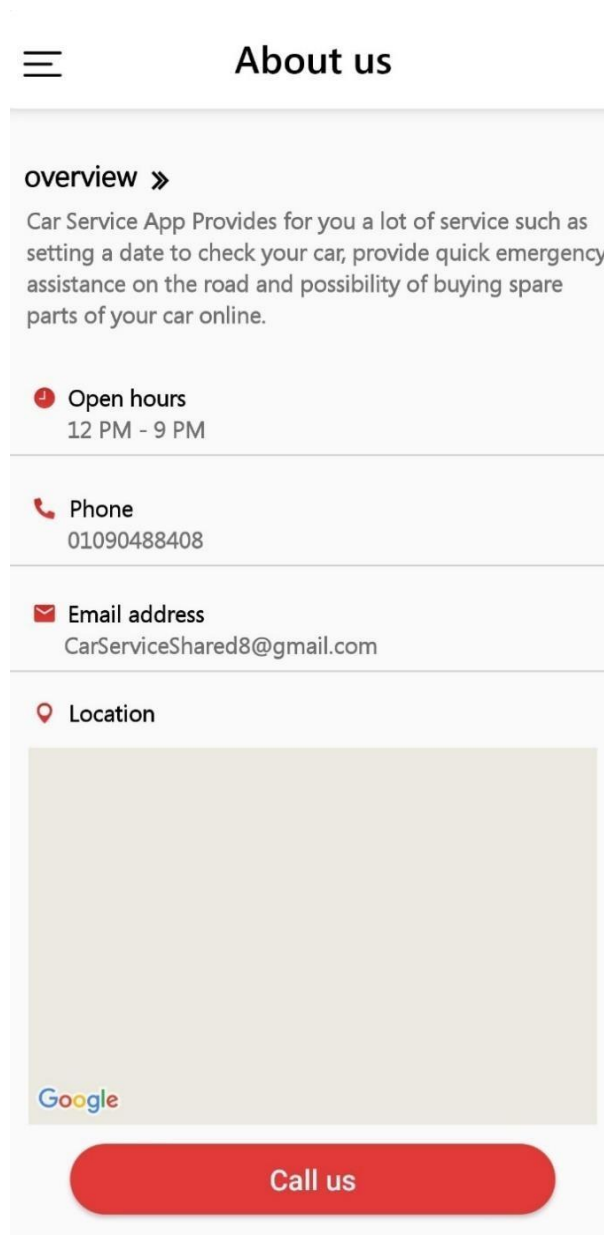


Figure 3.6

- About us page shows you who are we, work hours, customer service number, email address and our location.
- You can call us by clicking on Call us button

Conclusion

At the end of this work we hope that we applied the idea of this application successfully, It took all the time and effort to provide the users the most of ease as we faced many challenges and difficulties during working, some of these challenges is gathering and handling the appropriate parts of each car, also logging into the application via Facebook was one of the difficulties we faced, but in general the application was made in an acceptable form to do its job successfully which means that the application has pros and cons. Some of the pros are : showing all the services of the maintenance center, saving time for the users by booking a service, providing a delivery for spare parts and some regular maintenance at the customer house without him being at the house. One of the cons is that we couldn't provide logging into the application by mobile number and we looking forward to develop the application so it could be used in iOS systems, so that both Android and iOS users would use the application which helps to increase the number of users.

References

We don't have enough references for this application because it is a new idea. We can say that the idea is came by searching for a newer ways to make fixing cars and buying all spare parts easier, but actually there are websites that do this roll and we used its idea to make this application, example of the websites :

1- Google visited

- ❖ <https://www.onlinecarparts.co.uk/car-brands/spare-parts-jeep/grand-cherokee-iii-wh-wk/12569.html>
- ❖ <https://www.carparts.com/>

Appendices (or Appendix)

Login Activity:

```
public class LoginActivity extends AppCompatActivity implements View.OnClickListener {  
    // providers IDs Do Not Change them  
    private final String GOOGLE_PROVIDER = "google.com";  
    private final String FACEBOOK_PROVIDER = "facebook.com";  
    private final String TWITTER_PROVIDER = "twitter.com";  
    // Views  
    private EditText emailET, passwordET;  
    private Button signInBtn;  
    private ImageView showPasswordIcon, facebookIcon, googleIcon, twitterIcon;  
    private TextView forgotPassword, signUp;  
    // Firebase  
    private FirebaseAuth mAuth;  
    private FirebaseAuth.AuthStateListener firebaseAuthListener;  
    private FirebaseUser currentUser;  
    private DatabaseReference dbRef;  
    // Google Client  
    private int RC_SIGN_IN = 1;  
    private GoogleSignInClient mGoogleSignInClient;  
    // Callback manager  
    private CallbackManager callbackManager;  
    private ProgressBar progressBar;  
    // Twitter provider  
    OAuthProvider.Builder provider = OAuthProvider.newBuilder(TWITTER_PROVIDER);  
    // Local Variables  
    private boolean isPasswordVisible = false;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_login);  
        progressBar = findViewById(R.id.login_progressbar);  
        initializeUi();  
        //callbackManager to handle login responses  
        callbackManager = CallbackManager.Factory.create();  
        // Firebase Auth Listener to send user to homepage  
        // After logging in with any of the 4 methods [emailPSW,google,fb,twitter]  
        mAuth = FirebaseAuth.getInstance();  
        firebaseAuthListener = new FirebaseAuth.AuthStateListener() {  
            @Override  
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
```

```

        currentUser = firebaseAuth.getCurrentUser();
        if (currentUser != null) {
            Intent intent = new Intent(LoginActivity.this, HomepageActivity.class);
            startActivity(intent);
            finish();
        }
    }
};

// Click Listeners
signInBtn.setOnClickListener(this);
facebookIcon.setOnClickListener(this);
googleIcon.setOnClickListener(this);
twitterIcon.setOnClickListener(this);
showPasswordIcon.setOnClickListener(this);
forgotPassword.setOnClickListener(this);
signUp.setOnClickListener(this);
} //END OF ON CREATE

// Initialize Views
private void initializeUi() {
    // setFullScreenMode();
    emailET = findViewById(R.id.login_et_email);
    passwordET = findViewById(R.id.login_et_password);
    signInBtn = findViewById(R.id.login_btn_signIn);
    signUp = findViewById(R.id.login_tv_signUp);
    showPasswordIcon = findViewById(R.id.login_icon_showPassword);
    facebookIcon = findViewById(R.id.login_img_facebook);
    googleIcon = findViewById(R.id.login_img_google);
    twitterIcon = findViewById(R.id.login_img_twitter);
    forgotPassword = findViewById(R.id.login_tv_forgotPassword);
}

// Email and Password Methods
private void signInWithEmailPassword() {
    progressBar.setVisibility(View.VISIBLE);
    final String emailAddress = emailET.getText().toString().trim();
    final String password = passwordET.getText().toString().trim();
    mAuth.signInWithEmailAndPassword(emailAddress, password)
        .addOnCompleteListener(LoginActivity.this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (!task.isSuccessful()) {
                    // If sign in fails, display a message to the user.
                    Toast.makeText(LoginActivity.this, "The Email or Password is incorrect",
                        Toast.LENGTH_SHORT).show();
                }
            }
        });
}

```

```

        }
        mAuth.addAuthStateListener(firebaseAuthListener);
        progressBar.setVisibility(View.INVISIBLE);
    }
});

}

// if user used google or FB or twitter , then check to see if it's the first time
// if it is the first time.. then create an account for him with all the available data we can get from providers
// if not .. then direct him to the auth listener
private void checkFirstTimeSignIn(final String provider, final String userID, final FirebaseUser user) {
    progressBar.setVisibility(View.VISIBLE);
    dbRef = FirebaseDatabase
        .getInstance()
        .getReference()
        .child(Constants.USERS);
    dbRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (!dataSnapshot.hasChild(userID)) {
                createUserProfile(provider, userID, user);
            } else {
                hideProgress();
                mAuth.addAuthStateListener(firebaseAuthListener);
            }
        }
    });
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
        Toast.makeText(LoginActivity.this, databaseError.toString(), Toast.LENGTH_SHORT).show();
    }
});
}

// Create account for user from the data we have taken from providers
private void createUserProfile(String provider, String userID, FirebaseUser user) {
    currentUser = user;
    // prepare user profile object
    UserProfileModel userProfileObject = new UserProfileModel();
    if (provider.equals(GOOGLE_PROVIDER)) {
        GoogleSignInAccount acct = GoogleSignIn.getLastSignedInAccount(this);
        if (acct != null) {
            String userName = acct.getDisplayName();
            String email = acct.getEmail();
            Uri personPhoto = acct.getPhotoUrl();
            userProfileObject.setEmailAddress(email);

```



```

        userProfileObject.setUserName(userName);
        userProfileObject.setUserId(userID);
//        userProfileObject.setPhoneNumber("");
        if (personPhoto != null) {
            userProfileObject.setProfileImageUri(personPhoto.toString());
        }
//        else {
//            userProfileObject.setProfileImageUri("");
//        }
//        userProfileObject.setAddress("");
    }
}

else if (provider.equals(FACEBOOK_PROVIDER) || provider.equals(TWITTER_PROVIDER)) {
    String userName = currentUser.getDisplayName();
    String email = currentUser.getEmail();
    String personPhoto = "";
    String phoneNumber = "";
    userProfileObject.setEmailAddress(email);
    userProfileObject.setUserName(userName);
    userProfileObject.setUserId(userID);
//    if (currentUser.getPhoneNumber() != null) {
//        phoneNumber = currentUser.getPhoneNumber();
//    }
//
//    if (currentUser.getPhotoUrl() != null) {
//        personPhoto = currentUser.getPhotoUrl().toString();
//    }
    if (currentUser.getPhotoUrl() != null) {
        userProfileObject.setProfileImageUri(currentUser.getPhotoUrl().toString());
    }
    if (currentUser.getPhoneNumber() != null) {
        userProfileObject.setPhoneNumber(currentUser.getPhoneNumber().toString());
    }
//    userProfileObject.setPhoneNumber(phoneNumber);
//    userProfileObject.setProfileImageUri(personPhoto);
//    userProfileObject.setAddress("");
}

// prepare firebase root
DatabaseReference userRef = FirebaseDatabase
    .getInstance()
    .getReference()
    .child(Constants.USERS)

```

```

        .child(userID);

userRef.child(Constants.USER_CARS).setValue("0");
userRef.child(Constants.APPOINTMENTS_ORDERS).child(Constants.APPOINTMENTS).setValue("0");
userRef.child(Constants.APPOINTMENTS_ORDERS).child(Constants.ORDERS).setValue("0");
userRef.child(Constants.SHOPPING_CART).setValue("0");

// add user profile to database
userRef.child(Constants.USER_PROFILE)
    .setValue(userProfileObject)
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            if (mAuth.getCurrentUser() != null) {
                hideProgress();
                mAuth.addAuthStateListener(firebaseAuthListener);
            }
        }
    });
}

// Facebook Methods
private void signInWithFacebook() {
    // @alfred
    showProgress();

    LoginManager.getInstance().loginWithReadPermissions(LoginActivity.this, Arrays.asList("email",
"public_profile"));

    LoginManager.getInstance().registerCallback(callbackManager,
        new FacebookCallback<LoginResult>() {
            @Override
            public void onSuccess(LoginResult loginResult) {
                // App code
                handleFacebookAccessToken(loginResult.getAccessToken());
            }
            @Override
            public void onCancel() {
                // App code
                hideProgress();
            }
            @Override
            public void onError(FacebookException exception) {
                // App code
                hideProgress();
                Toast.makeText(LoginActivity.this, exception.toString(), Toast.LENGTH_SHORT).show();
            }
        }
    );
}

```

```

    });
}
private void handleFacebookAccessToken(AccessToken token) {
    // @alfred
    // to get an access token for the signed-in user, put it in FireBase then auth
    final AuthCredential credential = FacebookAuthProvider.getCredential(token.getToken());
    mAuth.signInWithCredential(credential)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success
                    FirebaseUser user = mAuth.getCurrentUser();
                    if (user != null) {
                        String userID = user.getId();
                        checkFirstTimeSignIn(FACEBOOK_PROVIDER, userID, user);
                    }
                } else {
                    // If sign in fails, display a message to the user.
                    hideProgress();
                    Toast.makeText(LoginActivity.this, "Authentication failed.",
                        Toast.LENGTH_SHORT).show();
                }
            }
        });
}

// Google Methods
private void signInWithGoogle() {
    showProgress();
    buildGoogleClient();
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}

private void buildGoogleClient() {
    // Configure Google Sign In
    GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(getString(R.string.default_web_client_id))
        .requestEmail()
        .requestProfile()
        .build();
    // Build a GoogleSignInClient with the options specified by gso.
    mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
}

```

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // onActivityResult method to pass the login results to the LoginManager via callbackManager
    callbackManager.onActivityResult(requestCode, resultCode, data);
    super.onActivityResult(requestCode, resultCode, data);
    // Result returned from launching the Intent from GoogleSignInClient.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        showProgress();
        // The Task returned from this call is always completed, no need to attach a listener.
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        handleSignInResultOfGoogle(task);
    }
}

private void handleSignInResultOfGoogle(Task<GoogleSignInAccount> completedTask) {
    try {
        GoogleSignInAccount account = completedTask.getResult(ApiException.class);
        passCredentialsToFirebaseAuth(account);
    } catch (ApiException e) {
        hideProgress();
        // Toast.makeText(this, String.valueOf(e), Toast.LENGTH_SHORT).show();
    }
}

private void passCredentialsToFirebaseAuth(GoogleSignInAccount acct) {
    AuthCredential credential = GoogleAuthProvider.getCredential(acct.getIdToken(), null);
    mAuth.signInWithCredential(credential).addOnCompleteListener(this, new OnCompleteListener<AuthResult>()
    {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                FirebaseUser user = mAuth.getCurrentUser();
                if (user != null) {
                    String userID = user.getId();
                    checkFirstTimeSignIn(GOOGLE_PROVIDER, userID, user);
                }
            } else {
                Toast.makeText(LoginActivity.this, "Authentication Failed",
                    Toast.LENGTH_SHORT).show();
            }
        }
    });
}

```

```

// Twitter Methods
private void signInWithTwitter() {
    mAuth.startActivityForResultSignInWithProvider(this, provider.build()).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                FirebaseUser user = mAuth.getCurrentUser();
                if (user != null) {
                    String userID = user.getId();
                    checkFirstTimeSignIn(TWITTER_PROVIDER, userID, user);
                }
            } else {
                hideProgress();
                Toast.makeText(LoginActivity.this, "Authentication failed.",
                    Toast.LENGTH_SHORT).show();
            }
        }
    });
}

// Edit Text Validation
private boolean isValid() {
    if (!MyValidation.isValidEmail(emailET)) {
        emailET.setError("Enter a valid email");
        emailET.requestFocus();
        return false;
    } else if (!MyValidation.isValidPassword(passwordET)) {
        passwordET.setError("Password is not correct");
        passwordET.requestFocus();
        return false;
    } else {
        return true;
    }
}

// Progress Bar
private void showProgress() {
    progressBar.setVisibility(View.VISIBLE);
}

private void hideProgress() {
    progressBar.setVisibility(View.INVISIBLE);
}

// Click Events
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.login_btn_signIn:
            if (isValid()) {

```

```

        signInWithEmailPassword();
    }
    break;
case R.id.login_img_facebook:
    signInWithFacebook();
    break;
case R.id.login_img_google:
    signInWithGoogle();
    break;
case R.id.login_img_twitter:
    signInWithTwitter();
    break;
case R.id.login_icon_showPassword:
    if (!MyValidation.isEmpty(passwordET)) {
        if (!isPasswordVisible) {
            // show password and change icon to black eye
            MyCustomSystemUi.showPassword(passwordET, showPasswordIcon);
            isPasswordVisible = !isPasswordVisible;
        } else {
            // hide password and change icon to grey eye
            MyCustomSystemUi.hidePassword(passwordET, showPasswordIcon);
            isPasswordVisible = !isPasswordVisible;
        } }
        break;
case R.id.login_tv_forgotPassword:
    startActivity(new Intent(LoginActivity.this, ForgotPasswordActivity.class));
    break;
case R.id.login_tv_signUp:
    startActivity(new Intent(LoginActivity.this, RegistrationActivity.class));
    break;
    } }
//Activity life cycle
@Override
protected void onStop() {
    super.onStop();
    if (null != firebaseAuthListener) {
        mAuth.removeAuthStateListener(firebaseAuthListener);
    } }
@Override
protected void onPause() {
    super.onPause();
    if (null != firebaseAuthListener) {
        mAuth.removeAuthStateListener(firebaseAuthListener); } }

```

```

@Override
protected void onResume() {
    super.onResume();
    if (mAuth.getCurrentUser() != null) {
        mAuth.addAuthStateListener(firebaseAuthListener);
    }
    MyCustomSystemUi.clearInput(emailET);
    MyCustomSystemUi.clearInput(passwordET);
}

// Request Focus on Keyboard when clicking near Edit Texts
public void showLoginEmailKeyboard(View view) {
    MyCustomSystemUi.showKeyboard(this, emailET);
}

public void showLoginPasswordKeyboard(View view) {
    MyCustomSystemUi.showKeyboard(this, passwordET);
}
}

```

Registration Activity:

```

public class RegistrationActivity extends AppCompatActivity {
    private static final String TAG = "Registration Activity";
    /**
     * uploaded by ahmed tarek....16/11/2019.....
     * Modified by Mahmoud Badawy 17/11/2019
     */

    private EditText usernameET, phoneET, emailET, passwordET, confirmPasswordET;
    private Button createAccountBTN;
    private CheckBox termsConditionsCHBX;
    private ImageView showPasswordIcon, showConfirmPasswordIcon;
    private FirebaseAuth mAuth;
    private String userUID;
    private boolean isPasswordVisible = false;
    private boolean isConfirmPasswordVisible = false;
    /**
     * link of terms and conditions MODIFIED BY AHMED TAREK 26/11/2019.....
     */

    private TextView conditions;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);
        initializeUi();
    }
}

```

```

// Initialize FireBase Auth
 mAuth = FirebaseAuth.getInstance();

// Create Account
createAccountBTN.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (isDataValid()) {
            createAccount(); } } });

//Show Password
showPasswordIcon.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (!MyValidation.isEmpty(passwordET)) {
            if (!isPasswordVisible) {
                // show password and change icon to black eye
                MyCustomSystemUi.showPassword(passwordET, showPasswordIcon);
                isPasswordVisible = !isPasswordVisible;
            } else {
                // hide password and change icon to grey eye
                MyCustomSystemUi.hidePassword(passwordET, showPasswordIcon);
                isPasswordVisible = !isPasswordVisible; } } } });

//Show Confirm Password
showConfirmPasswordIcon.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (!MyValidation.isEmpty(confirmPasswordET)) {
            if (!isConfirmPasswordVisible) {
                // show password and change icon to black eye
                MyCustomSystemUi.showPassword(confirmPasswordET, showConfirmPasswordIcon);
                isConfirmPasswordVisible = !isConfirmPasswordVisible;
            } else {
                // hide password and change icon to grey eye
                MyCustomSystemUi.hidePassword(confirmPasswordET, showConfirmPasswordIcon);
                isConfirmPasswordVisible = !isConfirmPasswordVisible; } } } });

conditions.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast.makeText(RegistrationActivity.this, "terms and conditions", Toast.LENGTH_SHORT).show(); }

// .....END THE LINK OF CONDITIONS AND TERMS
});
}

```



```

// Logic Methods
private void createAccount() {
    final String username = usernameET.getText().toString().trim();
    final String emailAddress = emailET.getText().toString().trim();
    final String password = passwordET.getText().toString().trim();
    final String phoneNumber = phoneET.getText().toString().trim();
    //@AhmedMahmoud RealTime Database
    mAuth.createUserWithEmailAndPassword(emailAddress, password)
        .addOnCompleteListener(RegistrationActivity.this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // After the user is created Successfully
                    // Get user UID
                    userUID = mAuth.getUid();
                    // Sign out the user immediately from Auth listener to prevent the listener from directing the user to
homepage
                    mAuth.signOut();
                    // prepare user profile object
                    UserProfileModel userProfileObject = new UserProfileModel();
                    userProfileObject.setUserName(username);
                    userProfileObject.setEmailAddress(emailAddress);
                    userProfileObject.setUserId(userUID);
                    userProfileObject.setPhoneNumber(phoneNumber);
                    userProfileObject.setProfileImageUri("");
                    userProfileObject.setAddress("");
                    // prepare firebase root
                    DatabaseReference userId = FirebaseDatabase
                        .getInstance()
                        .getReference()
                        .child(Constants.USERS)
                        .child(userUID);
                    userId.child(Constants.USER_CARS).setValue("0");
                    userId.child(Constants.APPOINTMENTS_ORDERS).child(Constants.APPOINTMENTS).setValue("0");
                    userId.child(Constants.APPOINTMENTS_ORDERS).child(Constants.ORDERS).setValue("0");
                    userId.child(Constants.SHOPPING_CART).setValue("0");
                    // add user profile to database
                    userId.child(Constants.USER_PROFILE)
                        .setValue(userProfileObject)
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Toast.makeText(RegistrationActivity.this, "Account Created Successfully, Please log in ",

```

```

Toast.LENGTH_SHORT).show();

        Intent goToLoginActivity = new Intent(RegistrationActivity.this, LoginActivity.class);
        startActivity(goToLoginActivity);
        finish();
    }));
} else if (!task.isSuccessful()) {
    try {
        throw task.getException();
    } catch (FirebaseAuthUserCollisionException existEmail) {
        Toast.makeText(RegistrationActivity.this, "Email Already Exist .. Please log in ",
Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        Log.d(TAG, "onComplete: " + e.getMessage());} } } });}

private boolean isValid() {
    if (MyValidation.isEmpty(usernameET)) {
        usernameET.setError(" A username is required");
        usernameET.requestFocus();
        return false;
    } else if (!MyValidation.isEmail(emailET)) {
        emailET.setError("Enter a valid email");
        emailET.requestFocus();
        return false;
    } else if (!MyValidation.isPassword(passwordET)) {
        passwordET.setError("Password must be > 8 characters and have at least 1 number");
        passwordET.requestFocus();
        return false;
    } else if (!MyValidation.isConfirmed(passwordET, confirmPasswordET)) {
        confirmPasswordET.setError("Please confirm the password");
        confirmPasswordET.requestFocus();
        return false;
    } else if (!MyValidation.isPhone(phoneET)) {
        phoneET.setError("Enter a valid phone number");
        phoneET.requestFocus();
        return false;
    }
    //NEW MODIFICATIONS BY AHMED TAREK FOR PHNONE NUMBER 1/4/2020
    else if (!MyValidation.ismyphone(phoneET)) {
        phoneET.setError("The Phone Number Must Contain 11 Digits ");
        phoneET.requestFocus();
        return false;
    } else if (!MyValidation.ismyphone1(phoneET)) {
        phoneET.setError("The Phone Number Must Start With 01");
        phoneET.requestFocus();
        return false; }
}

```

//END OF NEW MODIFICATIONS

```
    else if (!MyValidation.isChecked(termsConditionsCHBX)) {
        Toast.makeText(this, "please read and accept our terms and conditions", Toast.LENGTH_SHORT).show();
        return false;
    } else {
        return true;}}
// Views Methods
private void initializeUi() {
    // MyCustomSystemUi.setFullScreenMode(this);
    //first step
    usernameET = findViewById(R.id.registration_et_username); // modified first name to username @badawy
    emailET = findViewById(R.id.registration_et_emailAddress);
    passwordET = findViewById(R.id.registration_et_password);
    confirmPasswordET = findViewById(R.id.registration_et_confirmPassword);
    createAccountBTN = findViewById(R.id.registration_btn_createAccount);
    phoneET = findViewById(R.id.registration_et_phone); //modified last name to phone @badawy
    termsConditionsCHBX = findViewById(R.id.registration_cb_termsCheck);
    showPasswordIcon = findViewById(R.id.registration_icon_showPassword);
    showConfirmPasswordIcon = findViewById(R.id.registration_icon_showConfirmPassword);
    conditions = findViewById(R.id.registration_tv_terms);
    //conditions and terms link ..... \
    conditions.setPaintFlags(conditions.getPaintFlags() | Paint.UNDERLINE_TEXT_FLAG);
}
public void showLoginActivity(View view) {
    finish();
}
public void showUsernameKeyboard(View view) {
    MyCustomSystemUi.showKeyboard(this, usernameET);
}
public void showEmailKeyboard(View view) {
    MyCustomSystemUi.showKeyboard(this, emailET);
}
public void showPasswordKeyboard(View view) {
    MyCustomSystemUi.showKeyboard(this, passwordET);
}
public void showConfirmPasswordKeyboard(View view) {
    MyCustomSystemUi.showKeyboard(this, confirmPasswordET);
}
public void showPhoneKeyboard(View view) {
    MyCustomSystemUi.showKeyboard(this, phoneET);
}
}
```

Homepage activity:

```
public class HomepageActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {
    // time in milliseconds to press back again to exit
    private static final int EXIT_APP_TIME_INTERVAL = 2000;
    private long backButtonPressed;
    private static Toast exitToast;
    private GoogleSignInClient mGoogleSignInClient;
    private static DrawerLayout drawerLayout;
    private NavigationView navigationView;
    private FirebaseAuth firebaseAuth;
    private DatabaseReference dbRef;
    private TextView navUserName, navUserEmail;
    private CircleImageView navProfileImg;
    private AlertDialog.Builder dialogBuilder;
    private AlertDialog alertDialog;
    private ProgressBar progressBar;
    private String userId;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_homepage);
        initializeUi();
        buildGoogleClient();
        // Initialize Firebase Auth Object
        firebaseAuth = FirebaseAuth.getInstance();
        userId = Objects.requireNonNull(firebaseAuth.getCurrentUser()).getUid();
        //Add Click listener to Navigation list
        navigationView.setNavigationItemSelectedListener(this);
        //To Start the Home Fragment once the activity starts
        onNavigationItemSelectedListener(navigationView.getMenu().findItem(R.id.nav_home).setChecked(true));
        // Initialize Shared Preferences
        MySharedPreferences.init(getApplicationContext());
        // Fetch user`s Data and Cars From Firebase
        fetchUserProfileFromFirebase();
        fetchUserCarsFromFirebase();
        // Fetch About us Data
        fetchAboutUsDataFromFirebase();
    }
    // [[ FIREBASE DATA RETRIEVAL ]]
    private void fetchUserProfileFromFirebase() {
        dbRef = FirebaseDatabase
```

```

        .getInstance()
        .getReference()
        .child(Constants.USERS)
        .child(userId)
        .child(Constants.USER_PROFILE);
dbRef.keepSynced(true);
dbRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        UserProfileModel userProfileObject = dataSnapshot.getValue(UserProfileModel.class);
        saveUserDataIntoSharedPreference(userProfileObject);
    }
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) { } });
private void fetchUserCarsFromFirebase() {
    dbRef = FirebaseDatabase
        .getInstance()
        .getReference()
        .child(Constants.USERS)
        .child(userId)
        .child(Constants.USER_CARS);
dbRef.keepSynced(true);
dbRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (!dataSnapshot.hasChildren()) {
            Intent intent = new Intent(HomepageActivity.this, AddCarActivity.class);
            HomepageActivity.this.startActivity(intent);
        } else {
            ArrayList<CarModel> carList = new ArrayList<>();
            for (DataSnapshot ds : dataSnapshot.getChildren()
            ) {
                carList.add(ds.getValue(CarModel.class));
            }
            saveUserCarsIntoSharedPreference(carList); } }
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) { } });
private void fetchAboutUsDataFromFirebase() {
    // initialize Firebase reference
    dbRef = FirebaseDatabase.getInstance().getReference().child(Constants.APP_DATA);
    // Retrieve data from firebase
    dbRef.child(Constants.ABOUT_US).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override

```

```

        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            AboutUsDataModel obj = dataSnapshot.getValue(AboutUsDataModel.class);
            if (obj != null) {
                saveAboutUsDataIntoSharedPreference(obj); } }
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) { } }); }
// [[ HANDLE USER`S DATA ]]
private void saveUserDataIntoSharedPreference(UserProfileModel userData) {
    Gson gson = new Gson();
    String userSerializedData = gson.toJson(userData);
    MySharedPreferences.write(MySharedPreferences.USER_DATA, userSerializedData);
    bindUserDataIntoNavHeader(); }
private void bindUserDataIntoNavHeader() {
    Gson gson = new Gson();
    String userSerializedData = MySharedPreferences.read(MySharedPreferences.USER_DATA, "");
    if (!userSerializedData.equals("")) {
        UserProfileModel userData = gson.fromJson(userSerializedData, UserProfileModel.class);
        // Profile image
        if (userData.getProfileImageUri().equals("")) {
            navProfileImg.setImageResource(R.drawable.ic_default_profile);
        } else {
            Glide.with(this).load(userData.getProfileImageUri()).into(navProfileImg);
        }
        navUserName.setText(userData.getUserName());
        navUserEmail.setText(userData.getEmailAddress());
    } else
        Toast.makeText(this, "Race Condition", Toast.LENGTH_SHORT).show();
}
// [[ HANDLE USER`S CARS ]]
private void saveUserCarsIntoSharedPreference(ArrayList<CarModel> carList) {
    Gson gson = new Gson();
    String userSerializedCars = gson.toJson(carList);
    MySharedPreferences.write(MySharedPreferences.USER_CARS, userSerializedCars);
}
// [[ HANDLE ABOUT US DATA ]]
private void saveAboutUsDataIntoSharedPreference(AboutUsDataModel obj) {
    Gson gson = new Gson();
    String serializedAboutUsData = gson.toJson(obj);
    MySharedPreferences.write(MySharedPreferences.ABOUT_US, serializedAboutUsData); }
// init Ui
private void initializeUi() {
    drawerLayout = findViewById(R.id.homepage_drawerLayout);
    navigationView = findViewById(R.id.navigation_view);

```

```

View navHeaderView = navigationView.getHeaderView(0);
navUserName = navHeaderView.findViewById(R.id.nav_username);
navUserEmail = navHeaderView.findViewById(R.id.nav_email);
navProfileImg = navHeaderView.findViewById(R.id.nav_profileImage);
progressBar = findViewById(R.id.homepage_ProgressBar); }

// [[ HANDLE NAVIGATION ]]

@Override
public void onBackPressed() {
    // Know which fragment is currently Displayed
    Fragment currentFragment =
getSupportFragmentManager().findFragmentById(R.id.homepage_fragment_container);
    // if nav drawer is open >> close it
    if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START);
    }
    // if nav cars is open >> back to homepage
    else if (currentFragment instanceof NavCarsFragment) {
        openHomepage();
    }
    // if nav Appointments and Orders is open >> back to homepage
    else if (currentFragment instanceof NavAppointmentsFragment) {
        openHomepage();
    }
    // if nav Shopping Cart is open >> back to homepage
    else if (currentFragment instanceof NavShoppingCartFragment) {
        openHomepage();
    }
    // if nav Settings and Account is open >> back to homepage
    else if (currentFragment instanceof NavSettingsFragment) {
        openHomepage();
    }
    // if Homepage is open and sub fragments are open >> back from fragments
    else if (getSupportFragmentManager().getBackStackEntryCount() != 0) {
        super.onBackPressed();
    }

    // if Homepage is open and no sub fragments are open >> exit the application after pressing back button twice in 2
seconds
    else if (getSupportFragmentManager().getBackStackEntryCount() == 0) {
        if (backButtonPressed + EXIT_APP_TIME_INTERVAL > System.currentTimeMillis()) {
            exitToast.cancel(); // stop toast after exiting the application
            super.onBackPressed();
        } else {

```

```

        showExitMessage();
    }
    backButtonPressed = System.currentTimeMillis(); } }
//To take actions when an item is clicked in the navigation list
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
    switch (menuItem.getItemId()) {
        case R.id.nav_home:
            clearBackStack();
            replaceFragment(new NavHomeFragment());
            break;
        case R.id.nav_cars:
            clearBackStack();
            replaceFragment(new NavCarsFragment());
            break;
        case R.id.nav_settings:
            clearBackStack();
            replaceFragment(new NavSettingsFragment());
            break;
        case R.id.nav_appointments:
            clearBackStack();
            replaceFragment(new NavAppointmentsFragment());
            break;
        case R.id.nav_shopping_cart:
            clearBackStack();
            replaceFragment(new NavShoppingCartFragment());
            break;
        case R.id.nav_signOut:
            firebaseAuth.signOut();
            mGoogleSignInClient.signOut();
            MySharedPreferences.remove(MySharedPreferences.USER_DATA);
            MySharedPreferences.remove(MySharedPreferences.USER_CARS);
            Intent intent = new Intent(HomepageActivity.this, LoginActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(intent);
            finish();
            break;
        default:
            break;
    }
    drawerLayout.closeDrawer(GravityCompat.START);
    return true;
}

```



```

//To open the navigation list from inside the Fragments
//Called inside multiple fragment classes
public static void openDrawer() {
    if (!drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.openDrawer(GravityCompat.START);
    } else {
        drawerLayout.closeDrawer(GravityCompat.END); } }
private void replaceFragment(Fragment fragment) {
    getSupportFragmentManager().beginTransaction()
        .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
        .replace(R.id.homepage_fragment_container, fragment)
        .commit();}
private void clearBackStack() {
    FragmentManager fm = getSupportFragmentManager();
    for (int i = 0; i < fm.getBackStackEntryCount(); ++i) {
        fm.popBackStack(); } }
public void openHomepage() {
onNavigationItemSelectedListener(navigationView.getMenu().findItem(R.id.nav_home).setChecked(true));
}
// Used in Spare Parts Shop to Open the Shopping Cart
public void openShoppingCart() {
onNavigationItemSelectedListener(navigationView.getMenu().findItem(R.id.nav_shopping_cart).setChecked(true));
}
// Used in Check Out to Open the Settings Tab
public void openSettings() {
onNavigationItemSelectedListener(navigationView.getMenu().findItem(R.id.nav_settings).setChecked(true));
}
public void prepareDialog(int layout) {
    // Dismiss any old dialog.
    if (alertDialog != null) {
        alertDialog.dismiss();
    }
    clearBackStack();
    openHomepage();
    if (layout == R.layout.dialog_order_successful) {
        dialogBuilder = new AlertDialog.Builder(this);
        View layoutView = getLayoutInflater().inflate(R.layout.dialog_order_successful, null);
        TextView okTv = layoutView.findViewById(R.id.dialogOrderSuccessful_okTv);
        dialogBuilder.setView(layoutView);
        alertDialog = dialogBuilder.create();
        Objects.requireNonNull(alertDialog.getWindow()).setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        alertDialog.show();

```

```

        okTv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.dismiss(); } });
    } else if (layout == R.layout.dialog_appointment_successful) {
        dialogBuilder = new AlertDialog.Builder(this);
        View layoutView = getLayoutInflater().inflate(R.layout.dialog_appointment_successful, null);
        TextView okTv = layoutView.findViewById(R.id.dialogAppointmentSuccessful_okTv);
        dialogBuilder.setView(layoutView);
        alertDialog = dialogBuilder.create();
        Objects.requireNonNull(alertDialog.getWindow()).setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        alertDialog.show();
        okTv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.dismiss(); } });
    }
}

public void showProgressBar(boolean visible) {
    if (visible) {
        progressBar.setVisibility(View.VISIBLE);
    } else {
        progressBar.setVisibility(View.GONE);
    }
}

private void showExitMessage() {
    exitToast = Toast.makeText(getApplicationContext(), "Tab back button again to exit",
Toast.LENGTH_SHORT);
    exitToast.show();
}

private void buildGoogleClient() {
    // Configure Google Sign In
    GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(getString(R.string.default_web_client_id))
        .requestEmail()
        .requestProfile()
        .build();

    // Build a GoogleSignInClient with the options specified by gso.
    mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
}
}

```

Emergency Fragment:

```
/**
 * A simple {@link Fragment} subclass.
 */
public class EmergencyFragment extends Fragment {
    private boolean mLocationPermissionGranted;
    private static final int PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION = 1 ;
    public EmergencyFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_emergency, container, false);
        ImageView navMenuBtn = view.findViewById(R.id.emergency_navMenuBtn);
        Button requestTruck = view.findViewById(R.id.emergency_requestTruckBtn);
        requestTruck.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (ContextCompat.checkSelfPermission( getContext(),
                    Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
                    replaceFragment(new EmergencyRequestTruckFragment());
                } else {
                    requestLocationPermission();
                }
            }
        });
        navMenuBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                HomepageActivity.openDrawer();
            }
        });
        return view;
    }
    private void requestLocationPermission() {
        if (ActivityCompat.shouldShowRequestPermissionRationale(getActivity(),
            Manifest.permission.ACCESS_FINE_LOCATION)) {
            new AlertDialog.Builder(getContext())
                .setTitle("Permission needed")
                .setMessage("This permission is needed to access your location")
        }
    }
}
```

```

        .setPositiveButton("ok", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                ActivityCompat.requestPermissions(getActivity(),
                    new String[] {Manifest.permission.ACCESS_FINE_LOCATION},
                    PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);
            }
        })
        .setNegativeButton("cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        })
        .create().show();
    } else {
        ActivityCompat.requestPermissions(getActivity(),
            new String[] {Manifest.permission.ACCESS_FINE_LOCATION},
            PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
    if (requestCode == PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(getActivity(), "Permission GRANTED", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(getActivity(), "Permission DENIED", Toast.LENGTH_SHORT).show();
        }
    }
}

private void replaceFragment(Fragment fragment) {
    getActivity().getSupportFragmentManager().beginTransaction()
        .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
        .replace(R.id.homepage_fragment_container, fragment)
        .addToBackStack("EmergencyFragment")
        .commit();
}
}

```

Delivery Car Fragment:

```
/**
 * A simple {@link Fragment} subclass.
 */
public class DeliveryCarFragment extends Fragment implements View.OnClickListener,
SpeedFixCategoryAdapter.OnItemClickListener {
    // Global Variables
    private boolean isListVisible = true;
    private SelectCarRecyclerAdapter selectCarRecyclerAdapter;
    // private LinearLayout categoryLayout;
    // private ConstraintLayout categoryParent;
    private ImageView arrow, navMenuBtn;
    private CardView sparePartsCV, speedFixCard;
    private RecyclerView carRecyclerView, categoryRecyclerView;
    private ArrayList<CarModel> carList;
    private DatabaseReference dbRef;
    private ArrayList<SpeedFixCategoryModel> speedFixCategoryList;
    public DeliveryCarFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_delivery_car, container, false);
        // Initialize Ui
        initializeUi(view);
        // Get User Cars
        getUserCarsFromSharedPreferences();
        // Fetch Speed Fix Data From Firebase
        // fetchSpeedFixDataFromFirebase();
        // Set ClickListeners for views .. Function implemented in @onClick
        navMenuBtn.setOnClickListener(this);
        speedFixCard.setOnClickListener(this);
        sparePartsCV.setOnClickListener(this);
        return view;
    } // END OF ON CREATE
    private void fetchSpeedFixDataFromFirebase() {
        speedFixCategoryList = new ArrayList<>();
        dbRef =
        FirebaseDatabase.getInstance().getReference().child(Constants.APP_DATA).child(Constants.SPEED_FIX);
        dbRef.keepSynced(true);
    }
}
```

```

dbRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        for (DataSnapshot child: dataSnapshot.getChildren()
            ) {
                speedFixCategoryList.add(new
SpeedFixCategoryModel(R.drawable.ic_speedfix_air_conditioning,child.getKey(), (int) ((long) child.getValue())));
            }
            bindSpeedFixData(speedFixCategoryList);
        }
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) { } });
    private void bindSpeedFixData(ArrayList<SpeedFixCategoryModel> speedFixCategoryList) {
//      SpeedFixCategoryAdapter categoryAdapter = new SpeedFixCategoryAdapter(getActivity(),
speedFixCategoryList);
//      RecyclerView.LayoutManager categoryManager = new LinearLayoutManager(getActivity(),
RecyclerView.VERTICAL, false);
//      categoryRecyclerView.setLayoutManager(categoryManager);
//      categoryRecyclerView.setAdapter(categoryAdapter);
    }

    private void initializeUi(View view) {
        sparePartsCV = view.findViewById(R.id.delivery_spareParts_CV);
        navMenuBtn = view.findViewById(R.id.delivery_navMenuBtn);
        speedFixCard = view.findViewById(R.id.delivery_speedFix_CV);
//      categoryLayout = view.findViewById(R.id.delivery_speedFix_categoryLinearLayout);
//      categoryParent = view.findViewById(R.id.delivery_speedFix_category_parent);
//      arrow = view.findViewById(R.id.delivery_speedFix_listArrow);
        carRecyclerView = view.findViewById(R.id.delivery_chooseCarRecyclerView);
//      categoryRecyclerView = view.findViewById(R.id.delivery_speedFix_category_recyclerView);
    }
// get User Cars
    private void getUserCarsFromSharedPreferences() {
        Gson gson = new Gson();
        Type type = new TypeToken<ArrayList<CarModel>>() {
        }.getType();
        String serializedUserCarList = MySharedPreferences.read(MySharedPreferences.USER_CARS, "");
        if (!serializedUserCarList.equals("")) {
            carList = gson.fromJson(serializedUserCarList, type);
        }
        bindUserCars(carList);
    }
    private void bindUserCars(ArrayList<CarModel> carList) {

```

```

        selectCarRecyclerAdapter = new SelectCarRecyclerAdapter(getActivity(), carList);
        carRecyclerView.setAdapter(selectCarRecyclerAdapter);
    }

    private void fakeDataTest() {
        // Old name of speed fix was maintenance
        // speedFix Category Data
        // int[] categoryImages = {R.drawable.ic_speedfix_fix, R.drawable.ic_speedfix_air_conditioning,
        R.drawable.ic_speedfix_oil, R.drawable.ic_speedfix_battery};
        // String[] categoryName = {"Fix & Repair", "Air Conditioning", "Oil Change", "Check Battery"};
        // Simple Testing Data for Car Selection
        // byte[] carId = {0, 1, 2, 3, 4};
        // int[] carImage = {R.drawable.ic_nav_cars_car_test, R.drawable.ic_nav_cars_car_test,
        R.drawable.ic_nav_cars_car_test, R.drawable.ic_nav_cars_car_test, R.drawable.ic_nav_cars_car_test};
        // String[] carName = {"2018 Audi TT", "2014 Mazda Mk4", "2018 Audi TT", "2014 Mazda Mk4", "2018 Audi
        TT"};
        // // initialize Lists that will Hold the data
        // ArrayList<SelectCarModel> carList = new ArrayList<>();
        // ArrayList<SpeedFixCategoryModel> speedFixCategoryList = new ArrayList<>();
        //
        // // Filling category list
        // for (int i = 0; i < categoryImages.length; i++) {
        //     speedFixCategoryList.add(new SpeedFixCategoryModel(categoryImages[i], categoryName[i]));
        // }
        //
        // // Filling car list
        // for (int i = 0; i < carId.length; i++) {
        //     carList.add(new SelectCarModel(carId[i], carImage[i], carName[i]));
        // }
        //
        // // Bind Car Data to CarRecyclerView
        // final SelectCarRecyclerAdapter carAdapter = new SelectCarRecyclerAdapter(getActivity(), carList);
        // RecyclerView.LayoutManager manager = new LinearLayoutManager(getActivity(),
        RecyclerView.HORIZONTAL, false);
        // carRecyclerView.setLayoutManager(manager);
        // carRecyclerView.setAdapter(carAdapter);
        // // Bind Category Data to CategoryRecyclerView
        // speedFixCategoryAdapter categoryAdapter = new speedFixCategoryAdapter(getActivity(),
        speedFixCategoryList);
        // RecyclerView.LayoutManager categoryManager = new LinearLayoutManager(getActivity(),
        RecyclerView.VERTICAL, false);
        // categoryRecyclerView.setLayoutManager(categoryManager);
        // categoryRecyclerView.setAdapter(categoryAdapter);
    }

```

```

private void showCategory() {
    // Category Animation
    // int CATEGORY_ANIMATION_DURATION = 350;
    // Transition transition = new Slide(Gravity.TOP);
    // transition.setDuration(CATEGORY_ANIMATION_DURATION);
    // transition.addTarget(R.id.delivery_speedFix_categoryLinearLayout);
    // TransitionManager.beginDelayedTransition(categoryParent, transition);
    // categoryLayout.setVisibility(isListVisible ? View.VISIBLE : View.GONE);
    // // Arrow Animation
    // if (isListVisible) {
    //     arrow.animate().rotation(270);
    // } else {
    //     arrow.animate().rotation(90);
    // }
    // // Switch the Click from show to hide and vice versa
    // isListVisible = !isListVisible; }

private void replaceFragment(Fragment fragment) {
    Gson gson = new Gson();
    String serializedSelectedCarObject = gson.toJson(selectCarRecyclerAdapter.getSelectedCarObject());
    Bundle bundle = new Bundle();
    bundle.putString(Constants.SELECTED_CAR, serializedSelectedCarObject);
    fragment.setArguments(bundle);
    getActivity().getSupportFragmentManager().beginTransaction()
        .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
        .replace(R.id.homepage_fragment_container, fragment)
        .addToBackStack("DeliveryCarFragment")
        .commit(); }

@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.delivery_navMenuBtn:
            // Open side menu
            HomepageActivity.openDrawer();
            break;
        case R.id.delivery_speedFix_CV:
            // Show or hide speedFix category
            //showCategory();
            replaceFragment(new DeliveryCarSpeedFixFragment());
            break;
        case R.id.delivery_spareParts_CV:
            replaceFragment(new DeliveryCarSparePartsShopFragment());
            break;
        default:
    }
}

```



```

        break; } }

@Override
public void onItemClick(int position) { }
} // END OF CLASS

```

Car Center fragment:

```

/**
 * A simple {@link Fragment} subclass.
 */
public class CarCenterFragment extends Fragment implements View.OnClickListener {
    // Constants
    private final int SERVICE_TYPE_REQUEST_CODE = 1003;
    // Firebase
    private DatabaseReference dbRef;
    private Gson gson = new Gson();
    private DatabaseReference timeRef;
    private SimpleDateFormat dateFormatter = new SimpleDateFormat("EEEE, d MMMM", Locale.ENGLISH);
    // Lists
    private ArrayList<ServiceTypeModel> cycleList;
    private ArrayList<ServiceTypeModel> specificFixList;
    private ArrayList<ServiceTypeModel> carCareList;
    private ArrayList<TimeAppointmentModel> timeList;
    private ArrayList<CarModel> carList;
    // Views
    private ConstraintLayout serviceLayout;
    private RecyclerView timeRecyclerView, carRecyclerView;
    private CustomCalendar customCalendar;
    private TextView serviceTypeDescription, serviceTypePrice, serviceNameTv;
    private Button bookNowBtn;
    private ImageView navMenuBtn;
    private Intent intent;
    private TimeAppointmentRecyclerViewAdapter timeAdapter;
    private String selectedDay;
    private SelectCarRecyclerViewAdapter selectCarRecyclerViewAdapter;
    private String serviceName;
    private Activity activity;
    private int count;
    private boolean isServiceSelected = false;

```

```

public CarCenterFragment() {
    // Required empty public constructor
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_car_center, container, false);
    activity = getActivity();
    initializeUi(view);
    // Get Service Name to Decide if its Car Care or Vehicle Inspection
    assert getArguments() != null;
    serviceName = getArguments().getString(Constants.SERVICE_NAME_BUNDLE_KEY);
    // Intent to send lists to Pop up activity
    intent = new Intent(getActivity(), CarCenterSelectServicePopUpActivity.class);
    intent.putExtra(Constants.SERVICE_NAME_BUNDLE_KEY, serviceName);
    timeRecyclerView.setVisibility(View.GONE);
    // Init Appointment Time Root
    timeList = new ArrayList<>();
    timeAdapter = new TimeAppointmentRecyclerViewAdapter(getActivity());
    timeRef = FirebaseDatabase
        .getInstance()
        .getReference()
        .child(Constants.AVAILABLE_APPOINTMENTS)
        .child(Constants.CAR_CENTER);
    timeRef.keepSynced(true);
    // Decide if this is car care or Vehicle inspection
    // [[ Car Care]]
    assert serviceName != null;
    if (serviceName.equals(Constants.CAR_CARE)) {
        count = 0;
        serviceNameTv.setText(R.string.car_care);
        // init carCare list
        carCareList = new ArrayList<>();
        // Fetch carCare Data from firebase and add to intent
        fetchCarCareDataFromFirebase();
        isCurrentTimeIsAfterNinePMCarCare();
        // Get Current Day to Retrieve Available Appointments
        selectedDay = customCalendar.getDateInYearFormat();
        // Get Available Appointments from firebase
        checkTimeOfCarCare();
        customCalendar.setResetDayClick(new View.OnClickListener() {
            @Override

```

```

        public void onClick(View v) {
            customCalendar.resetDate();
            isCurrentTimeIsAfterNinePMCarCare();
            selectedDay = customCalendar.getDateInYearFormat();
            checkTimeOfCarCare();
        }
    });
    customCalendar.setArrowClick(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            customCalendar.getNextDay();
            selectedDay = customCalendar.getDateInYearFormat();
            checkTimeOfCarCare();
        }
    });
}

// [[ Vehicle Inspection ]]
else if (serviceName.equals(Constants.VEHICLE_INSPECTION)) {
    serviceNameTv.setText(R.string.vehicle_inspection);
    // init Lists
    cycleList = new ArrayList<>();
    specificFixList = new ArrayList<>();
    // Fetch Vehicle Inspection Data from firebase and add to intent
    fetchVehicleInspectionDataFromFirebase();
    isCurrentTimeIsAfterNinePMVehicleInspection();
    // Get Current Day to Retrieve Available Appointments
    selectedDay = customCalendar.getDateInYearFormat();
    // Get Available Appointments from firebase
    checkTimeOfVehicleInspection();
    customCalendar.setResetDayClick(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            customCalendar.resetDate();
            isCurrentTimeIsAfterNinePMVehicleInspection();
            selectedDay = customCalendar.getDateInYearFormat();
            checkTimeOfVehicleInspection();
        }
    });
    customCalendar.setArrowClick(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            customCalendar.getNextDay();
            selectedDay = customCalendar.getDateInYearFormat();

```

```

        checkTimeOfVehicleInspection(); } });}

// Get User Cars
getUserCarsFromSharedPreferences();
serviceLayout.setOnClickListener(this);

// Add Click Listeners
navMenuBtn.setOnClickListener(this);
bookNowBtn.setOnClickListener(this);

return view;
}

// fetch Service Lists
private void fetchVehicleInspectionDataFromFirebase() {
    dbRef = FirebaseDatabase
        .getInstance()
        .getReference()
        .child(Constants.APP_DATA)
        .child(Constants.VEHICLE_INSPECTION);
    dbRef.keepSynced(true);
    dbRef.child(Constants.VEHICLE_INSPECTION_CYCLE)
        .addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                for (DataSnapshot child : dataSnapshot.getChildren()) {
                    cycleList.add(new ServiceTypeModel(child.getKey(), (int) ((long) child.getValue())));
                }
                String serializedList = gson.toJson(cycleList);
                intent.putExtra(Constants.VEHICLE_INSPECTION_CYCLE, serializedList);
                hideProgress();
            }
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                // Do nothing
            }
        });
    showProgress();
    dbRef.child(Constants.VEHICLE_INSPECTION_SPECIFIC_FIXES)
        .addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                for (DataSnapshot child : dataSnapshot.getChildren()) {
                    specificFixList.add(new ServiceTypeModel(child.getKey(), (int) ((long) child.getValue())));
                }
                String serializedList = gson.toJson(specificFixList);
                intent.putExtra(Constants.VEHICLE_INSPECTION_SPECIFIC_FIXES, serializedList);
                hideProgress();
            }
        });
}

```

```

    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
        // Do nothing
    }); }

private void fetchCarCareDataFromFirebase() {
    dbRef = FirebaseDatabase.getInstance().getReference().child(Constants.APP_DATA)
        .child(Constants.CAR_CARE);
    dbRef.keepSynced(true);
    dbRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot child : dataSnapshot.getChildren()) {
                carCareList.add(new ServiceTypeModel(child.getKey(), (int) ((long) child.getValue())));
            }
            String serializedList = gson.toJson(carCareList);
            intent.putExtra(Constants.CAR_CARE, serializedList);
            hideProgress();
        }
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
        }
    });
}

// fetch Available Appointments
private void checkTimeOfCarCare() {
    count = 0;
    showProgress();
    // Data for Time
    final String[] time = {"12:00", "1:00", "2:00", "3:00", "4:00", "5:00", "6:00", "7:00", "8:00", "9:00"};
    final String[] timeOfDay = {"pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm"};
    timeRef.child(Constants.CAR_CARE).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (!dataSnapshot.hasChild(selectedDay)) {
                // Filling time list
                for (int i = 0; i < time.length; i++) {
                    TimeAppointmentModel obj = new TimeAppointmentModel();
                    obj.setId(i);
                    obj.setBooked("no");
                    obj.setTime(time[i]);
                    obj.setTimeOfDay(timeOfDay[i]);
                }
            }
        }
    });
}

```

```

        timeRef.child(Constants.CAR_CARE)
            .child(selectedDay)
            .child(String.valueOf(obj.getId()))
            .setValue(obj);
        hideProgress(); }}
    else {
        timeList = new ArrayList<>();
        // if day exist ... return available appointments
        for (DataSnapshot ds : dataSnapshot.child(selectedDay).getChildren()) {
            TimeAppointmentModel obj = new TimeAppointmentModel();
            obj.setBooked(ds.child("booked").getValue(String.class));
            obj.setId((int) ((long) ds.child("id").getValue()));
            obj.setTime(ds.child("time").getValue(String.class));
            obj.setTimeOfDay(ds.child("timeOfDay").getValue(String.class));
            timeList.add(obj);
            updateView(timeList); } } }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) { } }

private void checkTimeOfVehicleInspection() {
    count = 0;
    showProgress();
    // Simple Testing Data for Time
    final String[] time = {"12:00", "1:00", "2:00", "3:00", "4:00", "5:00", "6:00", "7:00", "8:00", "9:00"};
    final String[] timeOfDay = {"pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm"};
    timeRef.child(Constants.VEHICLE_INSPECTION).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (!dataSnapshot.hasChild(selectedDay)) {
                // Filling time list
                for (int i = 0; i < time.length; i++) {
                    TimeAppointmentModel obj = new TimeAppointmentModel();
                    obj.setId(i);
                    obj.setBooked("no");
                    obj.setTime(time[i]);
                    obj.setTimeOfDay(timeOfDay[i]);
                    timeRef.child(Constants.VEHICLE_INSPECTION)
                        .child(selectedDay)
                        .child(String.valueOf(obj.getId()))
                        .setValue(obj);
                    hideProgress();
                }
            }
        }
    })
}

```

```

else {
    timeList = new ArrayList<>();
    // if day exist ... return available appointments
    for (DataSnapshot ds : dataSnapshot.child(selectedDay).getChildren()) {
        TimeAppointmentModel obj = new TimeAppointmentModel();
        obj.setBooked(ds.child("booked").getValue(String.class));
        obj.setId((int) ((long) ds.child("id").getValue()));
        obj.setTime(ds.child("time").getValue(String.class));
        obj.setTimeOfDay(ds.child("timeOfDay").getValue(String.class));
        timeList.add(obj);
        updateView(timeList); } } }

@Override
public void onCancelled(@NonNull DatabaseError databaseError) { } }); }

// Update Time Recycler Adapter
// TEST AFTER NOON TO CHECK IF DATA IS RETURNED SUCCESSFULLY AFTER CURR TIME PASS
APPOINTMENT TIME

private void updateView(ArrayList<TimeAppointmentModel> timeList) {
    count= timeList.size();
    ArrayList<TimeAppointmentModel> availableTimeList = new ArrayList<>();
    for (TimeAppointmentModel obj : timeList
    ) {
        String appointmentTime = obj.getTime();
        if (customCalendar.getDay().equals(dateFormatter.format(Calendar.getInstance().getTime())) {
            if (customCalendar.getTimeOfDay().equals("am")) {
                if (obj.getBooked().equals("no")) {
                    availableTimeList.add(obj);
                }
            } else if (customCalendar.getTimeOfDay().equals("pm")) {

                if (appointmentTime.compareTo(customCalendar.getTime()) > 0) {
                    if (obj.getBooked().equals("no")) {
                        availableTimeList.add(obj); } } }

            } else {
                if (obj.getBooked().equals("no")) {
                    availableTimeList.add(obj); } } }
    if (count==10) {
        if (availableTimeList.size()>=1) {
            timeAdapter.setTimeList(availableTimeList);
            timeRecyclerView.setAdapter(timeAdapter);
            hideProgress();
            timeRecyclerView.setVisibility(View.VISIBLE);
        }
    }
}

```

```

else{
    customCalendar.getNextDay();
    selectedDay = customCalendar.getDateInYearFormat();
    if (serviceName.equals(Constants.CAR_CARE)){
        checkTimeOfCarCare();
    }
    else if(serviceName.equals(Constants.VEHICLE_INSPECTION)){
        checkTimeOfVehicleInspection(); } } }
// get User Cars
private void getUserCarsFromSharedPreferences() {
    Type type = new TypeToken<ArrayList<CarModel>>() {
    }.getType();
    String serializedUserCarList = MySharedPreferences.read(MySharedPreferences.USER_CARS, "");
    if (!serializedUserCarList.equals("")) {
        carList = gson.fromJson(serializedUserCarList, type);
    }
    bindUserCars(carList);
}
private void bindUserCars(ArrayList<CarModel> carList) {
    selectCarRecyclerAdapter = new SelectCarRecyclerAdapter(getActivity(), carList);
    carRecyclerView.setAdapter(selectCarRecyclerAdapter);
}
// init Ui
private void initializeUi(View view) {
    serviceNameTv = view.findViewById(R.id.carCenter_serviceTypeName);
    serviceLayout = view.findViewById(R.id.carCenter_serviceLayout);
    serviceTypeDescription = view.findViewById(R.id.carCenter_serviceTypeDescription);
    serviceTypePrice = view.findViewById(R.id.carCenter_serviceCost);
    timeRecyclerView = view.findViewById(R.id.carCenter_timeRecyclerView);
    customCalendar = view.findViewById(R.id.carCenter_customCalender);
    carRecyclerView = view.findViewById(R.id.carCenter_chooseCarRecyclerView);
    bookNowBtn = view.findViewById(R.id.carCenter_bookNowButton);
    navMenuBtn = view.findViewById(R.id.carCenter_navMenuBtn);
}
@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == SERVICE_TYPE_REQUEST_CODE && resultCode == RESULT_OK) {

        if (data != null) {
            // retrieve the serialized list and convert it to LIST type
            String serializedReturningList = data.getStringExtra(Constants.SERVICE_TYPES_NAME_RESULT);
            Type type = new TypeToken<ArrayList<ServiceTypeModel>>() {

```



```

        }.getType();
        ArrayList<ServiceTypeModel> selectedList = gson.fromJson(serializedReturningList, type);
        StringBuilder builder = new StringBuilder();
        if (selectedList.isEmpty()) {
            serviceTypeDescription.setText(R.string.click_to_select_service);
            isServiceSelected = false;
        } else {
            for (int i = 0; i < selectedList.size(); i++) {
                if (i + 1 != selectedList.size()) {
                    builder.append(selectedList.get(i).getServiceName()).append(" + ");
                } else {
                    builder.append(selectedList.get(i).getServiceName());
                }
            }
            serviceTypeDescription.setText(builder.toString());
        }
        int servicePrice = data.getIntExtra(Constants.SERVICE_TYPES_PRICE_RESULT, 0);
        if (servicePrice == 0) {
            serviceTypePrice.setText(R.string.please_select_a_service_first);
        } else {
            serviceTypePrice.setText(String.valueOf(servicePrice));
        }
        isServiceSelected = true;
    }
}

private void fakeDataTest() {
    // // Simple Testing Data for Time
    // byte[] id = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};
    // String[] time = {"11:00", "12:00", "1:00", "2:00", "3:00", "4:00", "5:00", "6:00", "7:00", "8:00", "9:00"};
    // String[] timeOfDay = {"am", "pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm", "pm"};
    // String[] booked = {"yes", "yes", "yes", "yes", "yes", "no", "no", "no", "no", "no", "no"};
    //
    // // Simple Testing Data for Car Selection
    // byte[] carId = {0, 1, 2, 3, 4};
    // int[] carImage = {R.drawable.ic_nav_cars_car_test, R.drawable.ic_nav_cars_car_test,
    R.drawable.ic_nav_cars_car_test, R.drawable.ic_nav_cars_car_test, R.drawable.ic_nav_cars_car_test};
    // String[] carName = {"2018 Audi TT", "2014 Mazda Mk4", "2018 Audi TT", "2014 Mazda Mk4", "2018 Audi
    TT"};
    // Fake Data
    // ArrayList<TimeAppointmentModel> timeList = new ArrayList<>();
    // ArrayList<SelectCarModel> carList = new ArrayList<>();
    //

```

```

// // Filling time list
// for (int i = 0; i < time.length; i++) {
//
//     if (booked[i].equals("no")) {
//         timeList.add(new TimeAppointmentModel(id[i], time[i], timeOfDay[i], booked[i]));
//     }
//
// }
//
//
// // Filling car list
// for (int i = 0; i < carId.length; i++) {
//     carList.add(new SelectCarModel(carId[i], carImage[i], carName[i]));
// }
//
//
// // Testing time list
// RecyclerView timeRecyclerView = view.findViewById(R.id.carCenter_timeRecyclerView);
// final TimeAppointmentRecyclerViewAdapter timeAdapter = new TimeAppointmentRecyclerViewAdapter(getActivity(),
timeList);
// timeRecyclerView.setAdapter(timeAdapter);
//
// //Testing car list
// RecyclerView carRecyclerView = view.findViewById(R.id.carCenter_chooseCarRecyclerView);
// final SelectCarRecyclerViewAdapter carAdapter = new SelectCarRecyclerViewAdapter(getActivity(), carList);
// carRecyclerView.setAdapter(carAdapter);
//
//
// Button book = view.findViewById(R.id.carCenter_bookNowButton);
// book.setOnClickListener(new View.OnClickListener() {
//     @Override
//     public void onClick(View v) {
//         Toast.makeText(getActivity(), timeAdapter.getTimeId() + "time", Toast.LENGTH_SHORT).show();
//         Toast.makeText(getActivity(), carAdapter.getCarId() + "car", Toast.LENGTH_SHORT).show();
//     }
// });
//
//
// Testing comparing time for future purpose
// CustomCalendar customCalendar = view.findViewById(R.id.carCenter_customCalendar);
// String time = "6:58";
// if (customCalendar.getTime().compareTo(time)>0||customCalendar.getTime().equals(time))

```

```

//      Toast.makeText(getActivity(), "yes "+customCalendar.getTime()+" is after "+time,
Toast.LENGTH_SHORT).show();
//      else Toast.makeText(getActivity(), "no "+customCalendar.getTime()+" less than "+time,
Toast.LENGTH_SHORT).show();
    }

    private void bookNow(String tag) {
        if (!isServiceSelected) {
            Toast.makeText(getActivity(), "Please Select a service First", Toast.LENGTH_SHORT).show();
        } else {
            showProgress();
            final BookingModel bookingObject = new BookingModel();
            Gson gson = new Gson();
            String userSerializedData = MySharedPreferences.read(MySharedPreferences.USER_DATA, "");
            if (!userSerializedData.equals("")) {
                UserProfileModel userDataObject = gson.fromJson(userSerializedData, UserProfileModel.class);
                bookingObject.setServiceName(serviceNameTv.getText().toString().trim());
                bookingObject.setPrice(serviceTypePrice.getText().toString().trim());
                bookingObject.setServiceDescription(serviceTypeDescription.getText().toString().trim());
                bookingObject.setDate(selectedDay);
                bookingObject.setTimeObject(timeAdapter.getTimeObject());
                bookingObject.setUserId(userDataObject.getUserId());
                bookingObject.setCarId(selectCarRecyclerAdapter.getSelectedCarObject().getCarID());
                bookingObject.setAddress(getResources().getString(R.string.university_address));

                dbRef = FirebaseDatabase.getInstance().getReference().child(Constants.AVAILABLE_APPOINTMENTS)
                    .child(Constants.CAR_CENTER).child(tag).child(bookingObject.getDate())
                    .child(String.valueOf(bookingObject.getTimeObject().getId()));

                final DatabaseReference bookingRef =
                    FirebaseDatabase.getInstance().getReference().child(Constants.BOOKING);

                final DatabaseReference userRef = FirebaseDatabase.getInstance().getReference().child(Constants.USERS);
                dbRef.addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        dbRef.child("booked").setValue("yes");
                        String orderID = bookingRef.push().getKey();
                        assert orderID != null;
                        bookingObject.setOrderId(orderID);
                        bookingRef.child(Constants.APPOINTMENTS).child(orderID)
                            .setValue(bookingObject);
                        userRef.child(bookingObject.getUserId()).child(Constants.APPOINTMENTS_ORDERS)
                            .child(Constants.APPOINTMENTS)
                            .child(orderID).setValue(0);
                        hideProgress();
                        if (activity instanceof HomepageActivity) {

```

```

        hideProgress();
        ((HomepageActivity) activity).prepareDialog(R.layout.dialog_appointment_successful);
    }
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {
}

});
}
}
}

private void showProgress() {
    if (activity instanceof HomepageActivity) {
        ((HomepageActivity) activity).showProgressBar(true);
    }
}

private void hideProgress() {
    if (activity instanceof HomepageActivity) {
        ((HomepageActivity) activity).showProgressBar(false);
    }
}

private void isCurrentTimeIsAfterNinePMCarCare() {
    Calendar cal = Calendar.getInstance();
    int currTime = cal.get(Calendar.HOUR_OF_DAY);
    int targetTime = 21;
    if (currTime >= targetTime) {
        customCalendar.getNextDay();
        selectedDay = customCalendar.getDateInYearFormat();
        checkTimeOfCarCare();
    }
}

private void isCurrentTimeIsAfterNinePMVehicleInspection() {
    Calendar cal2 = Calendar.getInstance();
    int currTime = cal2.get(Calendar.HOUR_OF_DAY);
    int targetTime = 21;
    if (currTime >= targetTime) {
        customCalendar.getNextDay();
        selectedDay = customCalendar.getDateInYearFormat();
        checkTimeOfVehicleInspection();
    }
}

@Override
public void onClick(View v) {

```

```

switch (v.getId()) {
    case R.id.carCenter_navMenuBtn:
        HomepageActivity.openDrawer();
        break;
    case R.id.carCenter_serviceLayout:
        startActivityForResult(intent, SERVICE_TYPE_REQUEST_CODE);
        break;
    case R.id.carCenter_bookNowButton:
        if (serviceName.equals(Constants.CAR_CARE)){

            bookNow(Constants.CAR_CARE);
        }
        else
            bookNow(Constants.VEHICLE_INSPECTION);
        break; } }}

```

About Us Fragment:

```

/**
 * A simple {@link Fragment} subclass.
 */
public class AboutUsFragment extends Fragment implements OnMapReadyCallback, View.OnClickListener {
    // Views
    private TextView overView, openHours, phoneNumber, emailAddress;
    private ImageView navMenuBtn;
    private Button callUsBtn;
    // Map Variables
    private static final int REQUEST_CALL = 1;
    private MapView mMapView;
    private static final String MAPVIEW_BUNDLE_KEY = "MapViewBundleKey";
    private double lat, lng;
    private String number;
    public AboutUsFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_about_us, container, false);
        initializeUi(view);
    }
}

```

```

// Load the Data from shared preference
loadDataFromSharedPref();

// Build the map
Bundle mapViewBundle = null;
if (savedInstanceState != null) {
    mapViewBundle = savedInstanceState.getBundle(MAPVIEW_BUNDLE_KEY);
}
mMapView.onCreate(mapViewBundle);
mMapView.getMapAsync(this);

// Click Listeners
navMenuBtn.setOnClickListener(this);
callUsBtn.setOnClickListener(this);
return view;
}

private void initializeUi(View view) {
    overView = view.findViewById(R.id.aboutUs_overviewDescription);
    openHours = view.findViewById(R.id.aboutUs_openHoursDescription);
    phoneNumber = view.findViewById(R.id.aboutUs_phoneNumberDescription);
    emailAddress = view.findViewById(R.id.aboutUs_emailAddressDescription);
    callUsBtn = view.findViewById(R.id.aboutUs_callUsBtn);
    navMenuBtn = view.findViewById(R.id.aboutUs_navMenuBtn);
    mMapView = view.findViewById(R.id.mapView);
}

private void loadDataFromSharedPref() {
    Activity activity = getActivity();
    if (activity != null) {
        // Get Data Object From Shared Preference
        Gson gson = new Gson();
        String serializedAboutUsData = MySharedPreferences.read(MySharedPreferences.ABOUT_US, "");
        AboutUsDataModel obj = gson.fromJson(serializedAboutUsData, AboutUsDataModel.class);
        overView.setText(obj.getOverView());
        openHours.setText(obj.getOpenHours());
        phoneNumber.setText(obj.getPhoneNumber());
        emailAddress.setText(obj.getEmailAddress());
        lat = obj.getLat();
        lng = obj.getLng();
        number = phoneNumber.getText().toString().trim();
    }
}

private void callUs(String num) {
    if (num.trim().length() > 0) {
        if (ContextCompat.checkSelfPermission(getActivity(), Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {

```

```

        ActivityCompat.requestPermissions(getActivity(), new String[]{Manifest.permission.CALL_PHONE},
REQUEST_CALL);
    } else {
        String dial = "tel:" + num;
        startActivity(new Intent(Intent.ACTION_CALL, Uri.parse(dial)));
    }
}
}

// Life Cycle
@Override
public void onStart() {
    super.onStart();
    mMapView.onStart();
}

@Override
public void onStop() {
    super.onStop();
    mMapView.onStop();
}

@Override
public void onResume() {
    super.onResume();
    mMapView.onResume();
}

@Override
public void onPause() {
    mMapView.onPause();
    super.onPause();
}

@Override
public void onDestroy() {
    mMapView.onDestroy();
    super.onDestroy();
}

@Override
public void onLowMemory() {
    super.onLowMemory();
    mMapView.onLowMemory();
}

// Map callback
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]

```

```

grantResults) {
    String number = phoneNumber.getText().toString().trim();
    if (requestCode == REQUEST_CALL) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            callUs(number);
        } else {
            Toast.makeText(getActivity(), "Permission Denied", Toast.LENGTH_SHORT).show();
        }
    }
}

@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    Bundle mapViewBundle = outState.getBundle(MAPVIEW_BUNDLE_KEY);
    if (mapViewBundle == null) {
        mapViewBundle = new Bundle();
        outState.putBundle(MAPVIEW_BUNDLE_KEY, mapViewBundle);
    }
    mMap.onSaveInstanceState(mapViewBundle);
}

@Override
public void onMapReady(final GoogleMap map) {
    map.addMarker(new MarkerOptions()
        .position(new LatLng(lat, lng))
        .title("Car Services"));
    LatLng currentLocation = new LatLng(lat, lng);
    map.moveCamera(CameraUpdateFactory.newLatLngZoom(currentLocation, 15));
}

@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.aboutUs_callUsBtn:
            callUs(number);
            break;
        case R.id.aboutUs_navMenuBtn:
            HomepageActivity.openDrawer();
            break;
        default:
            break;
    }
}

```


Arabic summery of the project

"car service"، تطبيق للموانف الذكية صمم لمساعد العمال على العناية بسياراتهم عن طريق توصيلهم بمركز خدمات السيارات عن طريق الحجز من خلال التطبيق وتحديد ميعد لفحص السيارة بدال من الذهاب الى المركز للحجز، وأيضا يمكنهم طلب فحص شامل للسيارة والقيام بعمل صيانات دورية وطلب قطع غيار عن طريق التطبيق دون الحاجة للذهاب إلى المركز، كما توفر في هذا التطبيق خدمة الطوارئ عند العطل المفاجئ وذلك عن طريق تحديد الموقع الحالي للعميل وإرسال الدعم له بالونش أو المقطورة. يؤم هذا التطبيق بعرض جمیع خدمات السيارات المتاحة فی الشركة وهی : فحص السيارة - عمل صيانة فورية للسيارة - امكانية شراء قطع غيار مناسبة للسيارة - اكسسوارات للسيارة - وخدمات أخرى. كما يتيح للعميل امكانية الدفع خلال الانترنت بدون الحاجة للذهاب الى المركز. كل الغرض من هذا التطبيق هو تسهيل عملية التواصل بين العميل ومركز خدمات سيارات وتوفیر أكبر وأفضل خدمة ممكنة لراحة العميل.