|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Submission Coversheet (All Programmes) | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Student ID Number**  *(Do not include student name as anonymous marking is implemented)* | RAD18457818  DRO18457544  ANI18457381 | | | | | | |
| **Programme Title** | COMPUTING TECHNOLOGIES | | | | | | |
| **Module Title** | Mobile App Design and Development | | | | | | |
| **Module Code** *(listed on Moodle and in LTAFP)* |  | | | | | | |
| **Module Convenor** | Tendai Mhlanga | | | | | | |
| **Coursework Title** | Barter Trader-Assignment 1 | | | | | | |
| **Academic Declaration:**  *Students are reminded that the electronic copy of their essay may be checked, at any point during their degree, with Turnitin or other plagiarism detection software for plagiarised material.* | | | | | | | |
| **Word Count** |  | | | **Date Submitted** | | **26/11/2019** | |
|  |  | | |  | |  | |
| Please save your file as:  student id\_module code\_module\_title\_assessment name  for example: COR15416549\_QAB020N592S\_Managing Organisations)\_Essay 1 Teamwork | | | | | | | |

Table of Contents

[**Project Details** 3](#_Toc24263052)

[Project Name 3](#_Toc24263053)

[Description 4](#_Toc24263054)

[**Functional Requirements** 4](#_Toc24263055)

[Anonymous User Functional Requirements 4](#_Toc24263056)

[Registered User Functional Requirements 4](#_Toc24263057)

[**Wireframes** 5](#_Toc24263058)

[**Use Case Diagram** 5](#_Toc24263059)

[Diagram Design 5](#_Toc24263060)

[Brief Descriptions of Use Cases 5](#_Toc24263061)

[**Application Design Model** 5](#_Toc24263062)

[Architectural Pattern 5](#_Toc24263063)

[Class Diagram 5](#_Toc24263064)

[Class Diagram Design 5](#_Toc24263065)

[Description of Classes 5](#_Toc24263066)

[**Firebase Design** 7](#_Toc24263067)

[Authentication 7](#_Toc24263068)

[Cloud Firestore 7](#_Toc24263069)

[Cloud Storage 7](#_Toc24263070)

[**Project Implementation** 8](#_Toc24263071)

[Project Structure 8](#_Toc24263072)

[Source Code 8](#_Toc24263073)

[**User Interface** 8](#_Toc24263074)

[Main View 8](#_Toc24263075)

[Select Category Gadgets 8](#_Toc24263076)

[Select Category Clothes 8](#_Toc24263077)

[Select Category Tools 8](#_Toc24263078)

[Select Category Bikes 8](#_Toc24263079)

[Select Category All 8](#_Toc24263080)

[View Selected Product 8](#_Toc24263081)

[Barter for Selected Product 8](#_Toc24263082)

[View Reviews on Selected Product 8](#_Toc24263083)

[View User’s products on Selected item 8](#_Toc24263084)

[View Products’ Photo 8](#_Toc24263085)

[View Products’ Video 8](#_Toc24263086)

[Main Menu Selection 8](#_Toc24263087)

[Sign in Page 8](#_Toc24263088)

[Sign in –Reset Password Page 8](#_Toc24263089)

[Sign up Page 8](#_Toc24263090)

[About Page 8](#_Toc24263091)

[Contact Page 8](#_Toc24263092)

[My Account 8](#_Toc24263093)

[Profile Page – View Profile 8](#_Toc24263094)

[Profile Page – User’s Reviews 8](#_Toc24263095)

[Products – View User’s Products 8](#_Toc24263096)

[Products – View Product 8](#_Toc24263097)

[Offers – View Received Offers 8](#_Toc24263098)

[Offers – View Offer 8](#_Toc24263099)

[Reviews – View User’s Accepted/Rejected Offers 8](#_Toc24263100)

[Reviews – View and Set Review for Accepted/Rejected Offers 8](#_Toc24263101)

[Add Product 8](#_Toc24263102)

[Add Product Page 9](#_Toc24263103)

[Add Product – Select Category 9](#_Toc24263104)

[Add Product – Select Photo 9](#_Toc24263105)

[Add Product – Select Video 9](#_Toc24263106)

[Add Product 9](#_Toc24263107)

[**Test Documentation** 9](#_Toc24263108)

[**Appendix:** 9](#_Toc24263109)

[**References:** 9](#_Toc24263110)

# **Project Details**

## Project Name

Barter Application

## Description

The Project Barter App

Mecachrome Software Ltd has employed us for creating a new application called ”Barter Trader”. The purpose of the app is to provide a platform to facilitate the barter (exchange) of goods among users who wish to barter various items like clothes, gadgets, tools, toys, bicycles etc.

# **Functional Requirements**

## Anonymous User - Functional Requirements

* Can view products by category in a grid arrangement
* Can view a product
* Can sign in, after registration
* Can read About, contact, terms and conditions of the application when the person decide to register.
* An anonymous can not register without accepting terms and conditions of the app

## Registered User - Functional Requirements

* Can login into a different account;
* Can register a new account;
* Can sign out
* Can recover the password
* Access His Account information:
* View Profile- including his first name, surname, telephone number, alias, email address, rating, if user is a flagged one-negative feedback and average number of the rating from other users. When rating falls below zero and the user is flagged more than 5 times, the user can no longer add new items and cannot message other users.
* View his Products-including image and title; if the user click on his product than he can see image of his product, recording, title and description
* View His Offers
* View reviews- on this page user has the option to add reviews of the user who posted the item for barter and to message the user;
* Add a product -including image, video, title, description of the item and option available to choose the right category
* Can read reviews from other users
* Can login to their accounts using their unique email address and password
* Can read About, contact, terms and conditions of the application when the person decide to register.
* Add a product including image, recording, title and description and option to choose the categories
* View the categories- From the Dashboard, when a category of item is selected (**Gadgets, Clothes, Tools, Bicycles, Others)**, the user is taken to a list of items for barter in that category
* - Categories listed for logged and anonymous users are: Gadgets, Clothes, Tools, Bicycles, Others;
* If users are selecting one of the four categories than all gadgets are displayed in a scrollable list allowing for navigability of the list;
* - The object of class Categories are dynamically read from a firebase database and then set on the activity;
* Details of each item/gadget will include: description of the item, rating of the user who posted the item, option to list all the items that this user whose item the logged user is currently viewing;
* If user is interested in the item and want to offer something in the exchange of it than user is able to message the other user with his/her offer;
* If the users agree with the barter than status of the items is changed to agreed;
* Once the item status is changed to agreed it is no longer listed in the application;

**Non-functional requirements:**

* Each category lists the same gadgets;
* When user get a call in the same time when he is using the app, then when he opens the app he should log in into his account;
* When user is adding a new product, he is acknowledged with message: “Product added successfully”
* When user open the app, it is loading within 10 seconds in such way that user doesn’t have to wait too long for app to respond

# **Wireframes**

# **Use Case Diagram**

## Diagram Design

//TO DO insert Use Case Diagram

## Brief Descriptions of Use Cases

//TO DO briefly describe each use case

# 

# **Application Design Model**

## Architectural Pattern

We’ve chosen the Model-View-View Model (MVVM) architectural pattern because Google embedded this pattern into the Android’s framework so that developers can easily implement their logic into mobile applications. Beside the ease of use, it is one of the architectural patterns which enforces separation of concerns, which divides the user interface from the business logic. The advantages of this is that the code is easy to maintain by dividing the work to frontend and backend developers, it is simple to isolate, and unit test the components and least but not last the reusability and extensibility of the classes increases the implementation speed of the project.

As seen in the figure below the implementation is divided into three layers:

* Model:

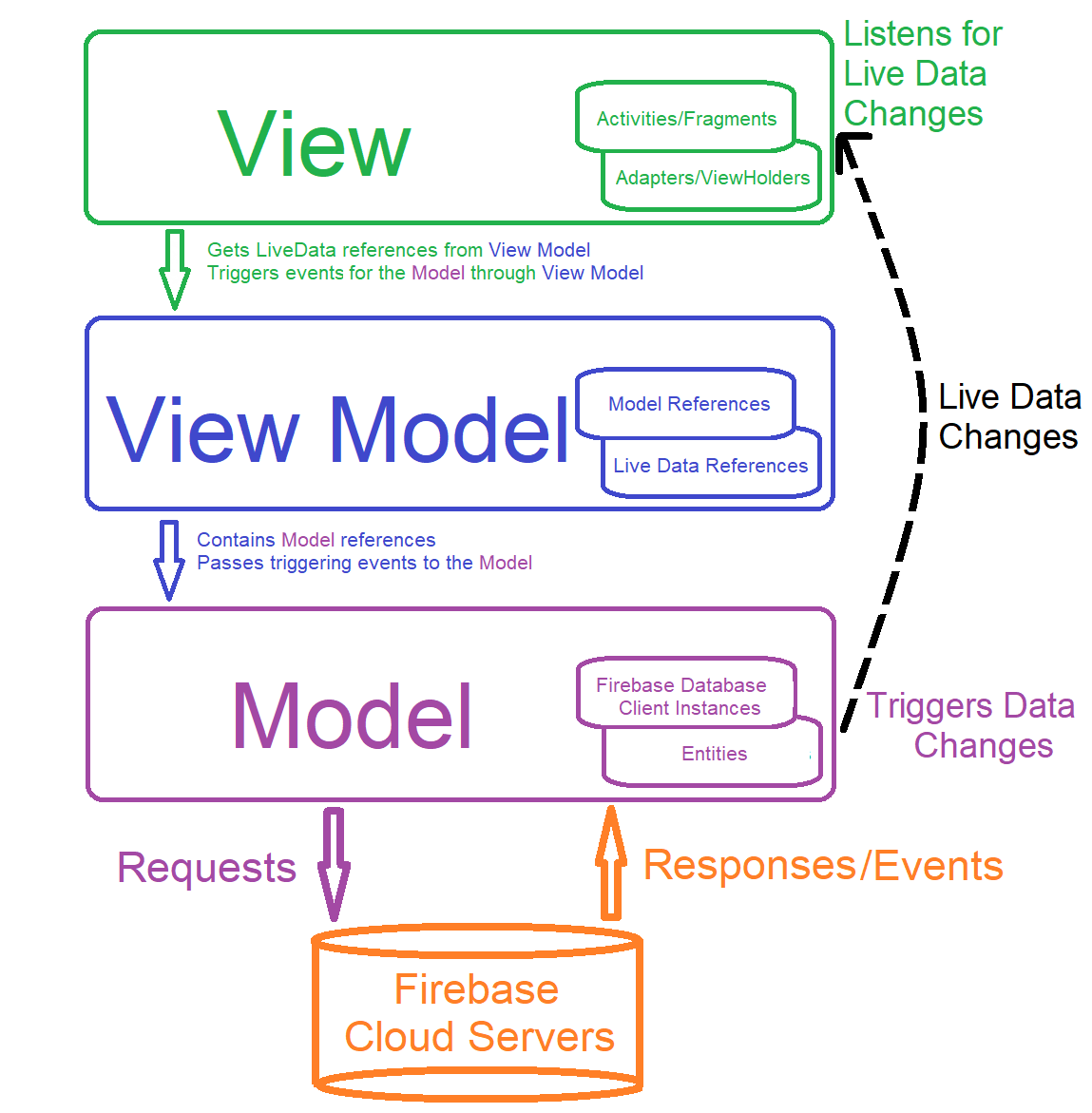
This layer represents the business logic of the application and exposes its data through the framework’s own observable data holder class, called LiveData. Beside triggering data changes, the model also contains entity classes used for structuring the data in the application and in the database. The Model also contains Firebase instances of its different clients for requesting and receiving data from the Google’s Servers.

* View Model:

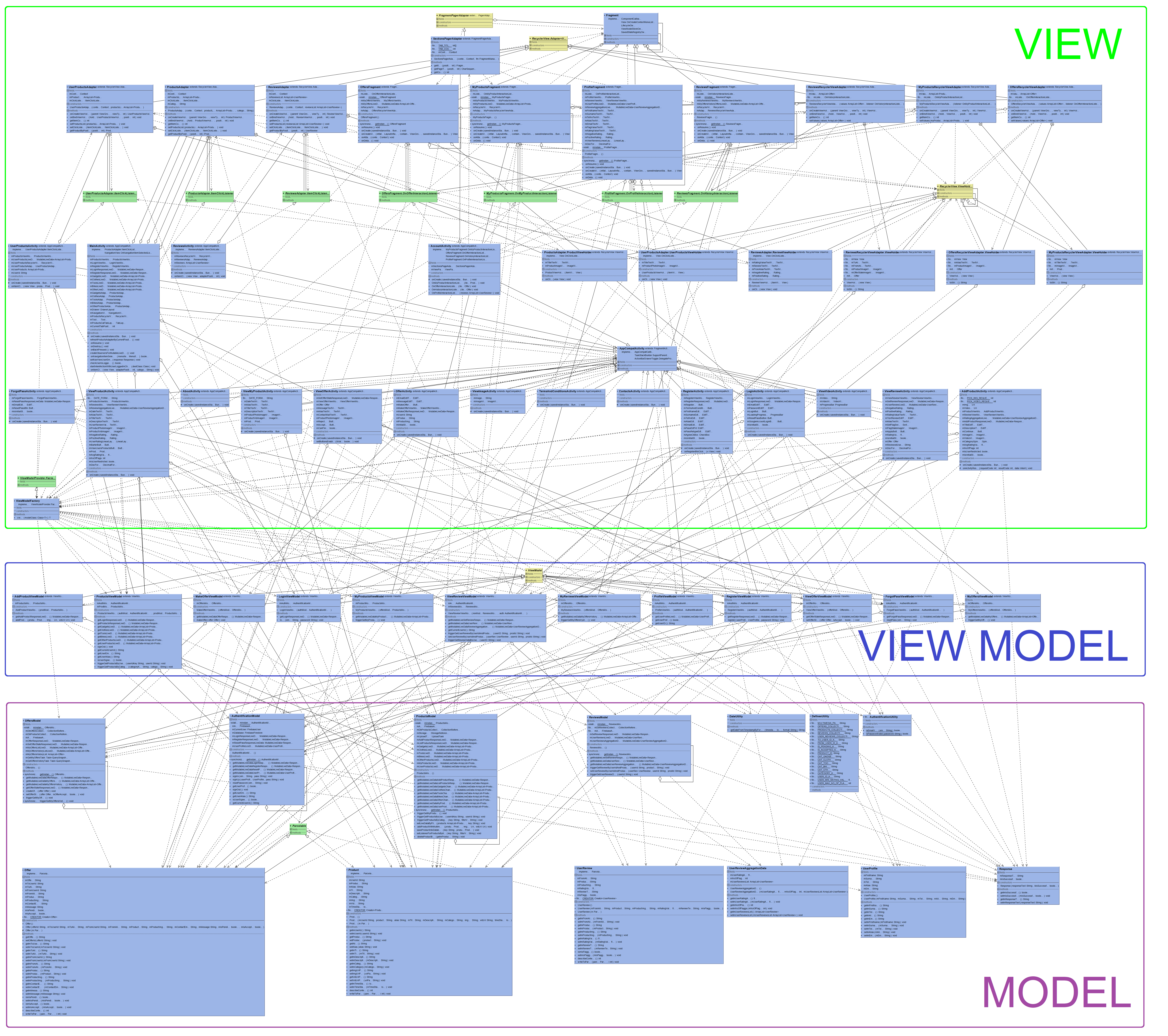
The View Model manages the Model components and provides references of the observers to the View components. Aside this, it also provides methods for passing events to the business logic. The whole idea behind the View Model is to decouple the View components from the implementation logic of the application. Android’s framework facilitates this through the ViewModel class, which is lifecycle aware and because of this allows data to be persistent even if views change their state.

* View:

The View layer only manages updates on Activity and Fragment classes with the help of their Adapters and View Holders. On this layer the View components observe for data changes in the Model and update the user interface accordingly. It also triggers events for the Model through the View Model in order to get data changes.



**Class Diagram**

Class Diagram Design

### Description of Classes

Because the system is designed as a MVVM architectural pattern and the number of classes surpasses 50, the description of the classes will be divided into Model/View/View Model categories and the common descriptions of the classes will be gathered under each group.

* **View Classes**

All Activity classes extend from the AppCompatActivity and override some of the lifecycle call-back methods like onCreate(), onResume(), onDestroy. Because of the complexity of the project we’ve used fragments which extend from the Fragment class and at least override onCreate(), onResume(), onCreateView, onAttach(), onDetach(). In the onCreate() method we only initialize, populate and set listeners for the UI elements, create the required view models, get the references of the exposed LiveData objects and create observers for them. In the onResume() method we only trigger events for refreshing the data in the UI elements. The onCreateView() method is used in the fragments classes to have access and inflate its layout and also populate and set listeners to the UI elements. The onAttach() and onDetach() methods are also used in fragments to set/unset the base activity’s listener.

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| MainActivity | It is the launcher activity, and it exposes the grid view of the products categories, |
| StudentUi | It derives from the MainUi class and contains the SubmitWorkButton user interface object to execute the method SubmitCoursework. |
| TutorUi | It derives from the MainUi class and contains buttons, inputs and widget lists to call the methods which handle students, viewing students’ coursework, managing resource material, adding remarks and handling attendance. |
| AdminUi | It derives from the MainUi class and contains buttons, inputs and widget lists to call the methods which search for items, manage accounts, handle courses, enrol students to courses, check student’s attendance and review student’s grade. |
| LoginController | It is used for logging in and out of the system. |
| ResourceController | It manages data like course material documents, coursework files and it stored in binary format by using the string instances for conversion to raw data. |
| AttendanceController | It handles all the aspects of the students’ attendance. |
| NotificationsEventHandler | This class sends notifications to the Ui and continuously checks for events like no coursework submitted, marks ready for viewing and excludable students. |
| AccountsController | The class is specialized in managing all the data from the account’s objects including adding remarks and reviewing student’s grades. |
| CourseController | It handles the programmes courses and modules including the enrolling of a student to a course. |
| IAccount | It is an interface class used as a polymorphic property to generalize the account instances in data collections like lists and maps. It also works as a contract to constrain the derived classes by forcing the implementation of the defined static methods in the interface. |
| Account | It is an abstract class which contains the common variables and methods of the users in the system like name, email, address, telephone number and their handling methods. It is the base class for the Admin, Student and Tutor classes. |
| Admin | This class derives from the Account class and differentiates from the other user classes by having a flag which says if it is the main admin or not. |
| Tutor | This class derives from the Account class and differentiates from the other user classes by having a flag which says if it is a convenor or not and contains the tutor’s enlisted modules. |
| Student | This class derives from the Account class and differentiates from the other user classes by having a grades list, the remarks, course id and year of study. |
| Course | It contains the assigned modules and manages the course related information. |
| Module | It is created and contained by the Course class and is stored in a list which means it has a one to many composition relationships with it. It handles all the module related data. |
| Resource | It handles and contains a material resource like submitted files, module documents and so on. |
| Attendance | It manages and contains the attendance of a student. |
| Model | It stores and manages all the data in the system and contains the connection handler with the main web service of the system. |
| Gateway | It represents the connection with the server and sends/receives data over the network. It also contains the token for two-factor authentication security, received on login. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# **Firebase Design**

## Authentication

## Cloud Firestore

## Cloud Storage

# **Project Implementation**

## Project Structure

## Source Code

# **User Interface**

## Main View

### Select Category Gadgets

### Select Category Clothes

### Select Category Tools

### Select Category Bikes

### Select Category All

## View Selected Product

### Barter for Selected Product

### View Reviews on Selected Product

### View User’s products on Selected item

### View Products’ Photo

### View Products’ Video

## Main Menu Selection

### Sign in Page

### Sign in –Reset Password Page

### Sign up Page

### About Page

### Contact Page

## My Account

### Profile Page – View Profile

### Profile Page – User’s Reviews

### Products – View User’s Products

### Products – View Product

### Offers – View Received Offers

### Offers – View Offer

### Reviews – View User’s Accepted/Rejected Offers

### Reviews – View and Set Review for Accepted/Rejected Offers

## Add Product

//TO DO show both options of adding a product(from menu and from button)

### Add Product Page

### Add Product – Select Category

### Add Product – Select Photo

### Add Product – Select Video

## Add Product

# **Test Documentation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Input** | **Output** | **Result** |
| Start Application | Tap the application’s icon. | The application starts in the main view showing the products under the gadgets category. | Successful |
| Register |  |  |  |
| Sign in |  |  |  |
| Sign Up |  |  |  |
| Sign out |  |  |  |
| About |  |  |  |
| Contact |  |  |  |
| Add Button |  |  |  |
| Categories |  |  |  |
| Add Product |  |  |  |
| Search item |  |  |  |
| Flagg user |  |  |  |
| Rate user |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **Appendix:**

# **References:**