|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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** | 4000 | | | **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 TO DO

# **Functional Requirements**

## Anonymous User Functional Requirements

* Can view products by category in a grid arrangement
* Can view a product
* … TO DO

## Registered User Functional Requirements

* Can login into a different account;
* Can register a new account;
* Can sign out
* Access His Account information:
* 1.View Profile

1.1

* View His Products
* View His Offers
* View
* … TO DO
* Barter trading application is required to perform the following procedures:
* -Anonymous users can use a restricted access to the application like:
* - A logged in user can:
* - A registration page to allow users to register including email address must be unique first name, surname and telephone number;
* - Email address will be used as the login;
* - Users are able to login to their accounts using their unique email address and password;
* - In the user profile, a user can view account information (first name, surname, phone number, alias, email), their private inbox of messages-offers, the rating, my products (add item and delete an item), history of the barters;
* - Deny access for users who are not registered to posting reviews, message users, buy any item, rating users, add item;
* - Logged users must view the categories;
* - Logged users must view only their own profile;
* - Logged users must have option to post a new item to barter;
* - Categories listed for logged and anonymous users are: Gadgets, Clothes, Tools, Bicycles;
* - Logged and anonymous users are selecting one of the four categories than all gadgets are displayed in a scrollable list allowing for navigability of the list;
* - Each category lists the same gadgets;
* - The object of class Categories are dynamically read from a firebase database and then set on the activity;
* - If logged user clicks on one of the items/gadgets than details of the item/gadgets are displayed;
* - Logged user must have option to view reviews of the user who posted the item for barter and to message the user;
* - Details of each item/gadget must 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;

# **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

## Class Diagram

### Class Diagram Design

//TO DO insert Class Diagram

### Description of Classes

Because the system is designed as a Model View View-Model architectural pattern //TO DO

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| MainUi | It is an abstract base class for the rest of the user interface classes meaning they have a generalization relantioship between them. |
| 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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **Appendix:**

# **References:**