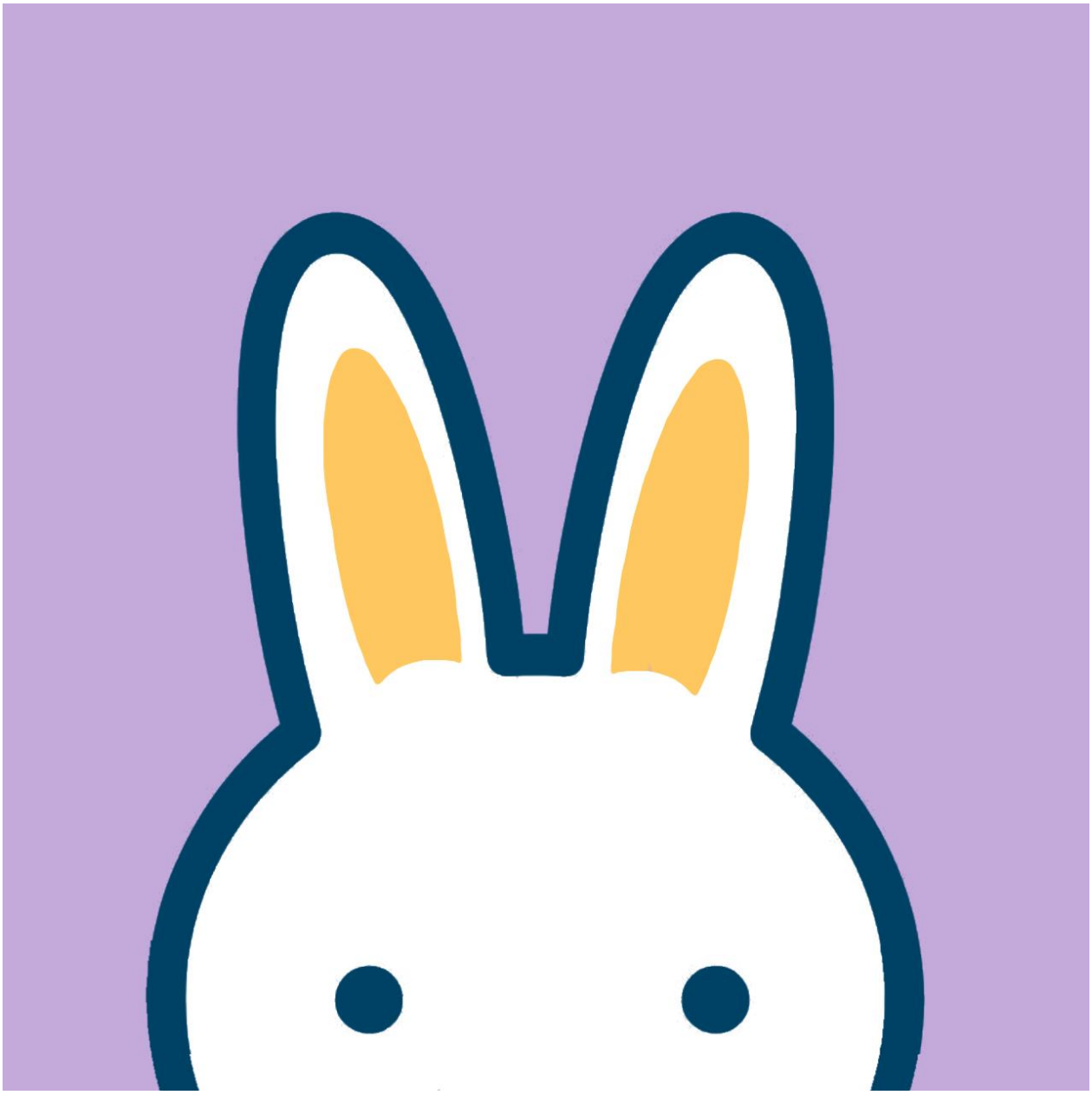
Retail Bunny Design Document



Designed by:
Lucian Irsigler
Banzile Nhlebela
Laaiqah Bayat
Prashan Rajaratnam
Version:1.0
Date Created: 20 May 2023

Contents

1. Introduction 3
1.1 App Purpose 3
2. User Interface Design and Data Validation 4
2.1 Start Up Screens 4
2.1.1 Sign-Up Screens 4
2.1.2 Log-In Screen 6
2.2 Buyer Screens 7
2.2.1 Review Product 7
2.3 Seller Screens 9
2.3.1 Adding Product 9
3. Business Rules and Tables 11
4. Entity Relational Diagrams (ERD) 13
4.1 ERD 13
4.2 Implementation of ERD and Addressing Issues 14
4.3 Normalization 15

1. Introduction

1.1 App Purpose

Retail Bunny is a South African user-friendly mobile marketplace designed for buyers and sellers to interact in a seamless manner. Our app will act as an intermediary between users by use of a multitude of different features. Each user will have to register and then use their respective username and password on each subsequent entry of the app. The following options will be available to the customer.

As a buyer, you have access to the following features:

1. Search for items
2. View products
3. Review an item that you have purchased

As a seller, you have access to the following features:

1. Post items and all their essential details
2. View how your customer's have reviewed your products.

Retail Bunny offers a reliable and efficient mobile marketplace experience, promoting ease of access, security and seamless transactions between buyers and sellers.

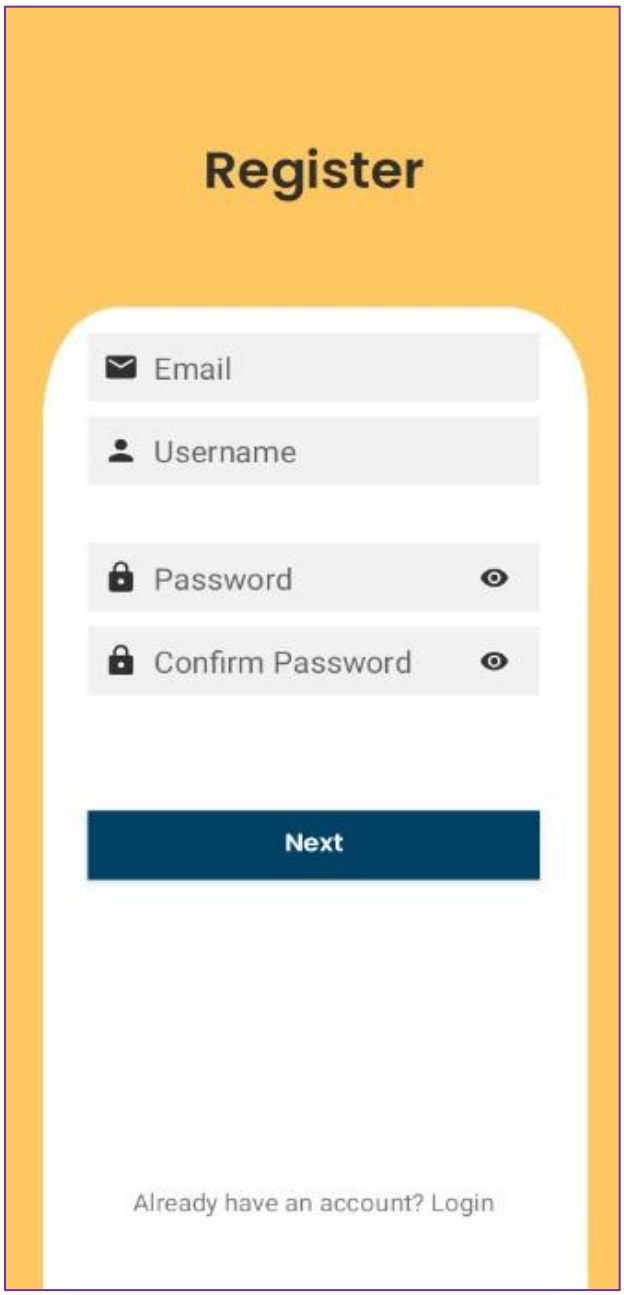
2. User Interface Design and Data Validation

This section will detail all the screens in which data validation and/or data verification is used throughout the app on different screens for all the necessary information to be concise and as accurate as possible.

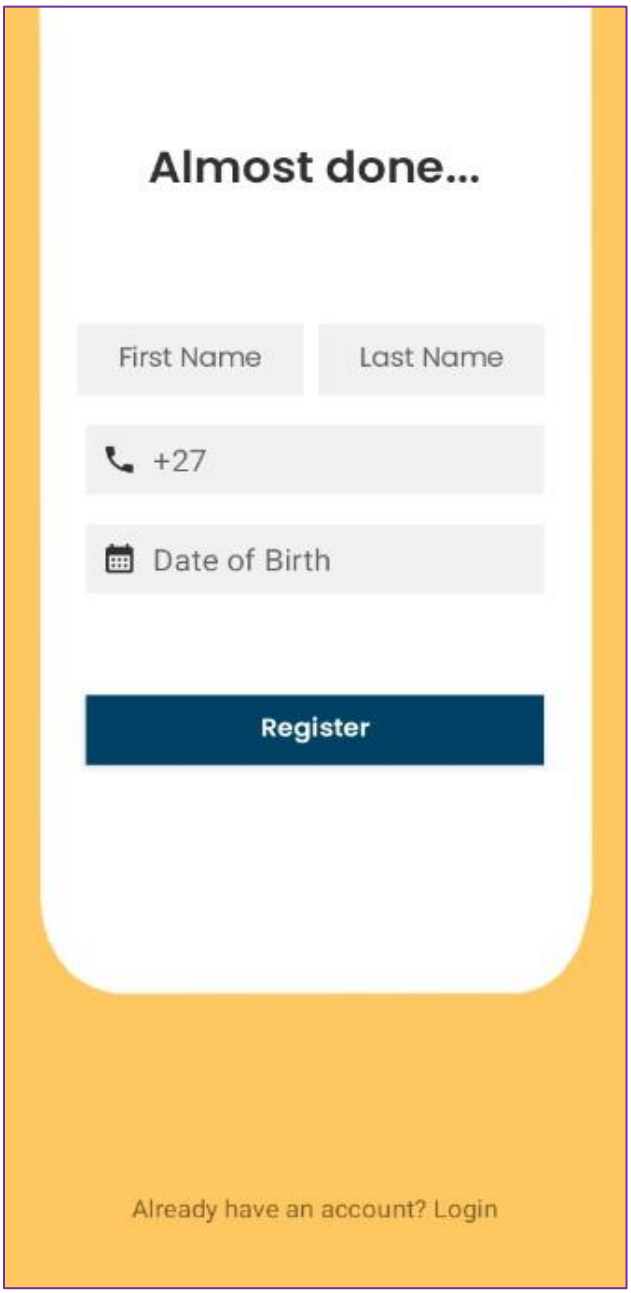
2.1 Start Up Screens

2.1.1 Sign-Up Screens

Register Screen 1



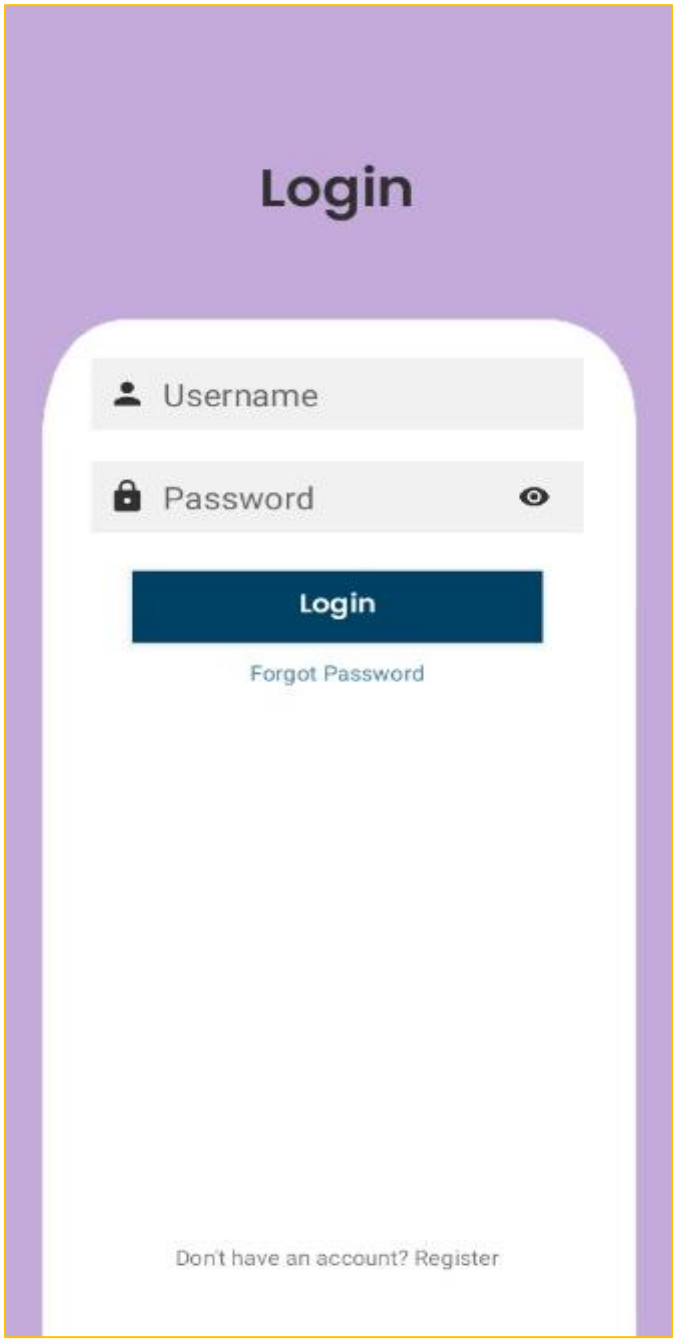
Register Screen 2



	Data Validation/ Verification/ Interactable	Function and Description
First name	Yes	Uses NAME_REGEX to ensure that the user's input starts with a capital letter followed by more letters, spaces, hyphens, or apostrophes. No numbers can be input here.
Last name	Yes	Uses NAME_REGEX to ensure that the user's input starts with a capital letter followed by more letters, spaces, hyphens, or apostrophes. No numbers can be input here.
Phone Number	Yes	Uses TextInputLayout to ensure that the user's input is only numbers. We have made it such that the user must be South African and therefore their number starts with +27 followed by nine other numbers.
Date of Birth	Yes	Uses TextInputLayout to ensure that the user's input is only numbers and dashes. We have also provided the format that the date should be entered such that the user is not confused.
Address	Yes	Uses TextInputLayout to ensure that the users can input anything required for an address, that being letters, numbers, and all types of punctuation.
Username	Yes	Uses TextInputLayout to ensure that the users can input letters, numbers, and all types of punctuation.
Password	Yes	Uses PASSWORD_REGEX to ensure that the user's input has at least one uppercase, lower case, digit, and special character. It must also be minimum 8 characters, this is used to ensure a security within the app.
Email	Yes	Uses EMAIL_REGEX to ensure that the user's input starts with one or more letters followed by an optional hyphen or period. This is followed by followed by the user's email account domain.

The Username and Password will be saved in our database and used in the log in screen when a user wants to enter the app.

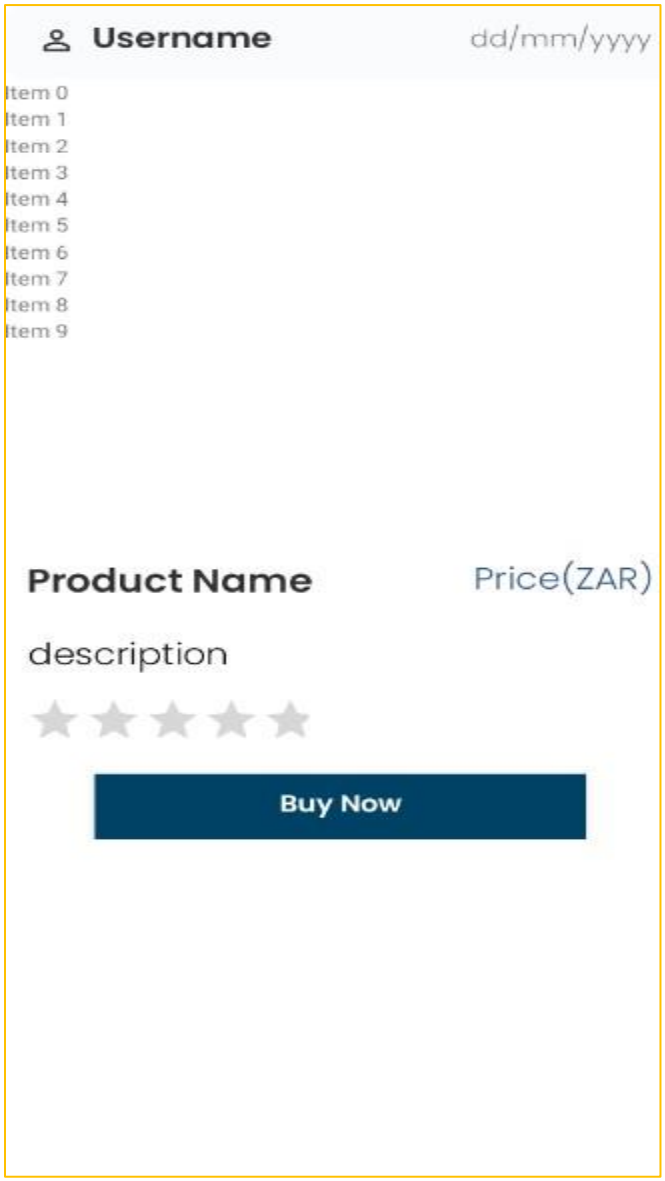
2.1.2 Log-In Screen



2.2 Buyer Screens

After logging in to the app, a buyer will have the chance to input data when searching for a specific product. SearchProduct function: Enables users to search for items by name. This function retrieves items from the Products table that match the search criteria and orders them by date posted with the newer products appearing at the top of the list through use of a recycler view. Our app also uses a rating system that will be presented once a product has been pressed on. The purchase and review screen can be seen below.

2.2.1 Review Product

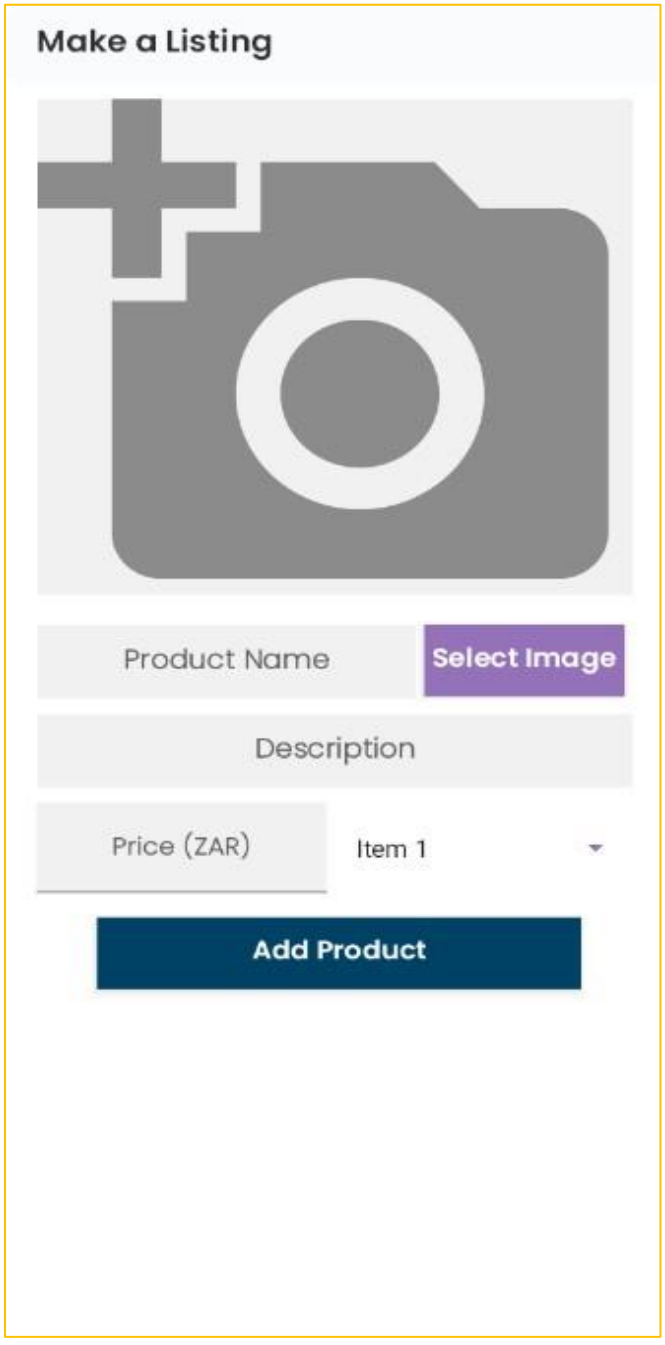


	Data Validation/ Verification/ Interactable	Function and Description
Star Rating	Interactable	User can enter their star rating here.
Buy Button	Interactable	User can buy product here.

2.3 Seller Screens

After logging in to the app, a seller will have the chance to input data when adding their product to the store. AddProduct function: Allows users to add products by providing the necessary details (name, description, price, and images), which will be inserted into the Products table. A picture of this screen and how data validation is used can be seen below.

2.3.1 Adding Product



	Data Validation/ Verification/ Interactable	Function and Description
Product Name	Yes	Uses TextInputLayout to ensure that the user's input can be letters, numbers, and any form of punctuation. We will also have a character limit to ensure that the layout of this page is not made untidy.
Price	Yes	Uses TextInputLayout to ensure that the user's input is only numbers. We have made it such that the user enters both Rands and Cents as our app is South African based and accuracy is important.
Description	Interactable	No form of data validation or verification is needed here.
Category	Interactable	Seller can input the type of category or choose from a pre-made drop-down list

3. Business Rules and Tables

Our app will run based on the following business rules

User Account:

- A seller can sell more than one product.
- A customer can write more than one review but only one per product.
- Each seller that has had a review written on their product, will have an average review score.
- Each user must have a unique account tied to a valid email address and username.
- Users must provide accurate and complete information when registering.
- On the administration end, we will assign the user an ID.
- Each user must have the following required attributes: user ID username, password, Salt, first name, last name, email, phone, and address.

Products List:

- Sellers can list products that they have the rights to sell.
- A product may have many reviews and one average rating.
- Every product belongs to a category and has an image
- Each product must have the following required attributes: a product ID (primary key), a user ID, name, description, price, category (foreign key to Categories table) and a dated added.

Reviews:

- A review belongs to one product.
- A product may have a review with the option of multiple reviews from different buyers as well the option for one average rating
- Each review must have the following required attributes: a review ID (primary key), user ID (primary key as well as foreign key to Users table), product ID, the rating, date and an average rating.

Categories:

- Each product belongs to a category.
- Each category may have many products.
- The categories table has 1 required attribute which will act as the primary key and that is the category name.

Avg Rating:

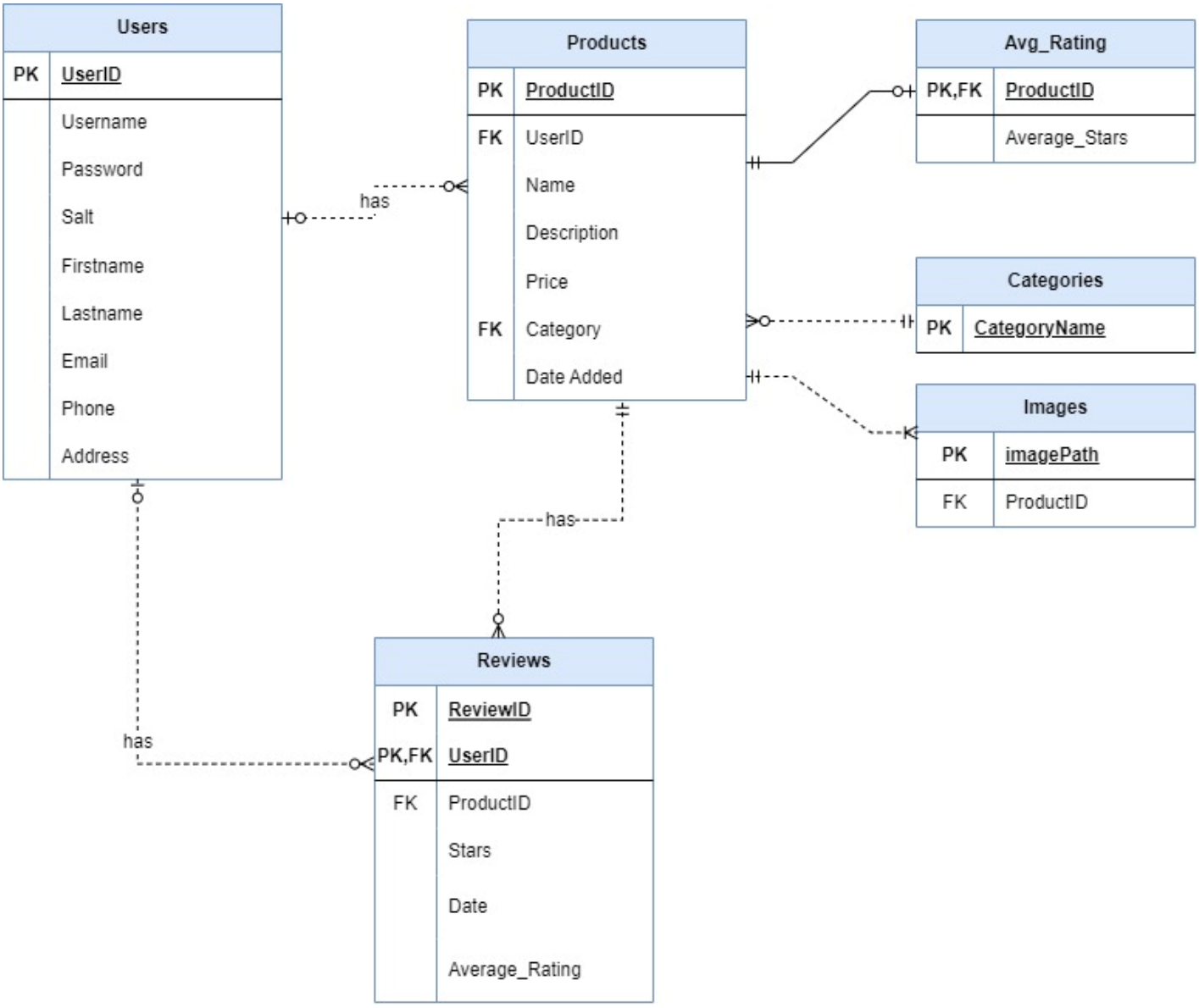
- A product will can have one average rating.
- An average rating belongs to one product.
- Each average rating must have the following attributes: product ID (primary key as well as foreign key to Products table),

Images:

- A product must have at least one image associated to it.
- Each image relates to one product.
- Each image must have the following attributes: an image path (primary key) and product ID (foreign key to the Products table)

4. Entity Relational Diagrams (ERD)

4.1 ERD



4.2 Implementation of ERD and Addressing Issues

PHP: For the PHP file structure, each table has a PHP file that contains its corresponding SQL query, i.e., insert, search and update. An example of this in practice is the Products table has a products.php that has the selectFromProducts, insertIntoproducts and so on.

Multivalued Attributes: Other than the user's address in the Users table, there are no multivalued attributes. For the address attribute, we kept the address as one field instead of splitting it into street, suburb, and city because our mobile market app does not use any form of location-based services.

Entity Relations: We have developed our ERD in such a way that we have avoided the use of M: N to relationships such that fewer data anomalies occur. Although at first sight of our ERD, it seems inefficient to use three tables (namely, Category, Avg_Rating and Images) with only one relation each to the Products table as well as two attributes each, the reason it was done this way was to help validate the data from these tables.

Null variables: All of this has already been discussed in the data input screens. To recap, all necessary fields will always have some value in it because we used TextInputLayout to ensure that the data is correct and not Null.

HTTP Handler Class: Offers a structured and reusable approach to handle HTTP requests in our application. The class has the logic required to perform both HTTP GET and POST requests. It provides separate methods for each type of request, making the code modular and easier to understand. The class includes error handling mechanisms to deal with various scenarios, such as network failures or unexpected responses. It provides onFailure callbacks in the request callbacks to handle exceptions and failures appropriately. The class supports multiple response types such as String, JSONArray, and JSONObject based on the provided responseType parameter.

4.3 Normalization

Legend for Normalization:
Underlined means primary key.
Red means foreign key.

2NF:

Users (UserID, Username, Password, Salt, Firstname, Lastname, Email, Phone, Address)
Products (ProductID, UserID, Name, Description, Price, Category, Date Added)
    Foreign key to link the products a customer buys or sells.
Category (CategoryName)
Reviews (ReviewID, UserID, ProductID, Date, Average_Rating)
    Foreign key to link Reviews to the product being reviewed.
Avg_Rating (ProductID, Average_Stars)
    Foreign key to link average rating to product table.
Images (imagePath, ProductID)
    Foreign key to link Images to the product in question.

3NF:

Users (UserID, Username, Password, Salt, Firstname, Lastname, Email, Phone, Address)
Products (ProductID, UserID, Name, Description, Price, Category, Date Added)
    Foreign key to link the products a customer buys or sells.
Category (CategoryName)
Reviews (ReviewID, UserID, ProductID, Date)
    Foreign key to link Reviews to the product being reviewed.
    Average_Rating (ReviewID, Average_Rating)
        Foreign key to link the transitive relation to Review table.
Avg_Stars (ProductID, Average_Stars)
    Foreign key to link average rating to product table.
Images (ImageID, ProductID)
    Foreign key to link Images to the product in question.