Storm Eagles

**Trackoholic: Your Daily Inventory Manager**

Software Design Document

Name (s):

Kyle Chan

Yasmeen Diego

David Eisenbaum

Neal Layumas

Joshua Rogers

Date: January 2020

**TABLE OF CONTENTS**

**1.**  **INTRODUCTION**  2

1.1 Purpose 2

1.2 Scope 2

1.3 Overview 2

**2. SYSTEM OVERVIEW** 2

**3. SYSTEM ARCHITECTURE** 2

3.1 Architectural Design 2

**4. DATA DESIGN** 3

4.1 Data Description 3

4.2 Data Dictionary 3

**5. COMPONENT DESIGN** 3

**6. HUMAN INTERFACE DESIGN** 4

6.1 Overview of User Interface 4

6.2 Screen Images 4

6.3 Screen Objects and Actions 4

**7. REQUIREMENTS MATRIX** 4

1. INTRODUCTION

1.1 Purpose

The SDD lays out the information required to define the architecture and system design of Trackoholics: Your Daily Inventory Manager in order to guide the development team on how the system architecture should be developed.

1.2 Scope

This document is created for the use of Trackoholics version 1.0, released on 5/13/2020. The software was developed in order to create an easy-to-use inventory management system for business, as well as personal, purposes.

1.3 Overview

This document describes the system’s architecture. It is organized into seven major sections. Most sections provide more detailed subsections that pertain to the major section’s topic. Graphics may be inserted for further clarification.

2. SYSTEM OVERVIEW

Trackoholics: Your Daily Inventory Manager is an inventory tracker designed for businesses and personal use. It enables users to quickly and easily document item information such as price, serial number, quantity, and date of purchase.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

Our application initially utilized a layered architecture and was designed to fit that approach. We wanted to have a presentation layer, which included our activities classes, an application layer, which included our recyclerview classes, a domain layer, which includes our model classes that require no dependencies from Android Studio, and an infrastructure layer that included our database handler classes.

However, we ultimately decided to change our approach and we had a realized repository architecture that was built around our sqLite database. It made more sense to build our system this way as we had data that needed to be shared in between classes. Regardless, our information sharing was kept at a minimum in order to minimize any mutability issues.

4. DATA DESIGN

4.1 Data Description

When the user creates any object, such as profile, location, or item, that data gets added to an arraylist of their respective objects. Each arraylist gets serialized as a JSON object, and it is then inputted into the sqLite database where our data is stored and is accessible..

4.2 Data Dictionary

SQLite - The database we are using to store our data as arraylists.

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

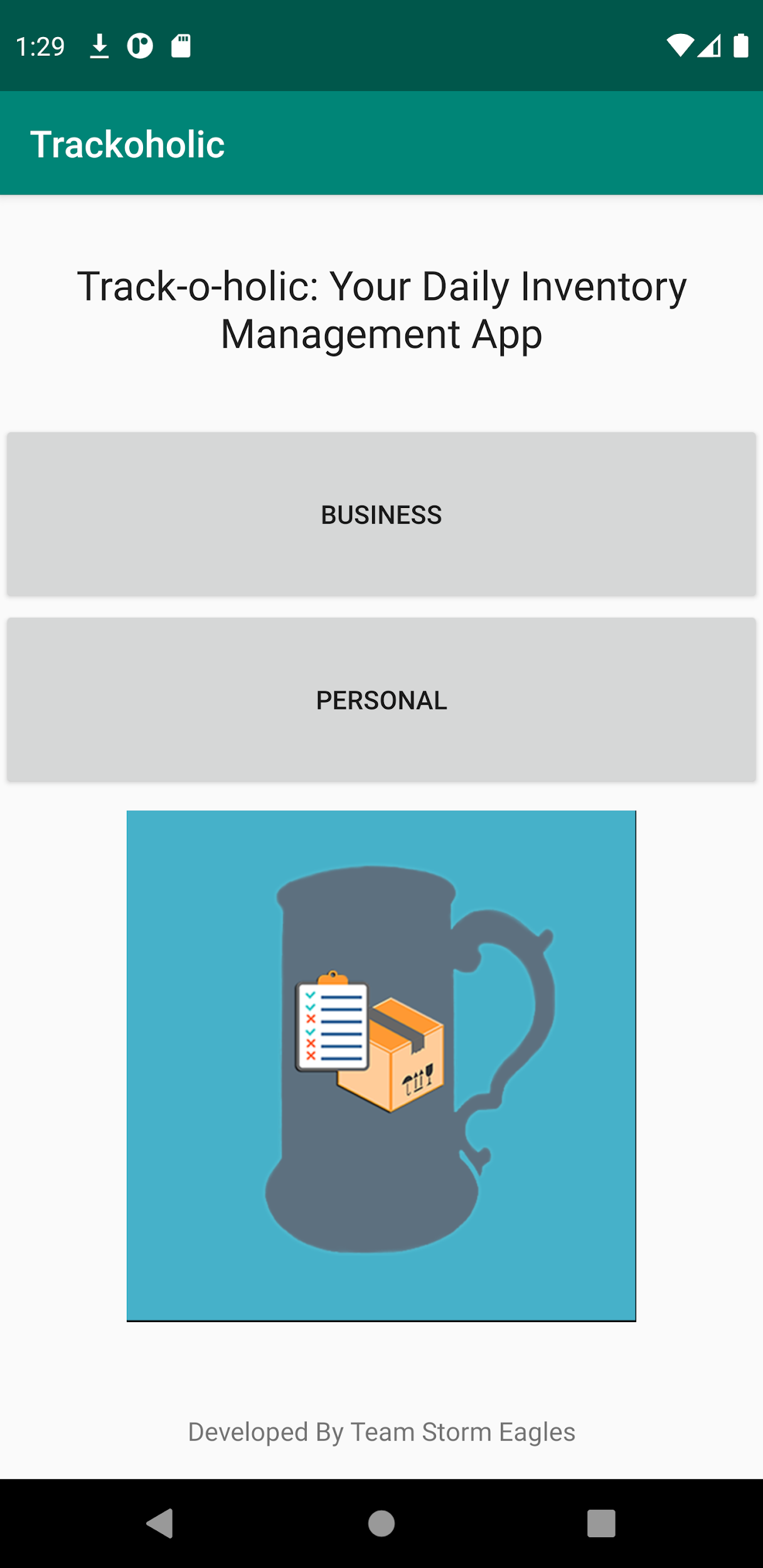
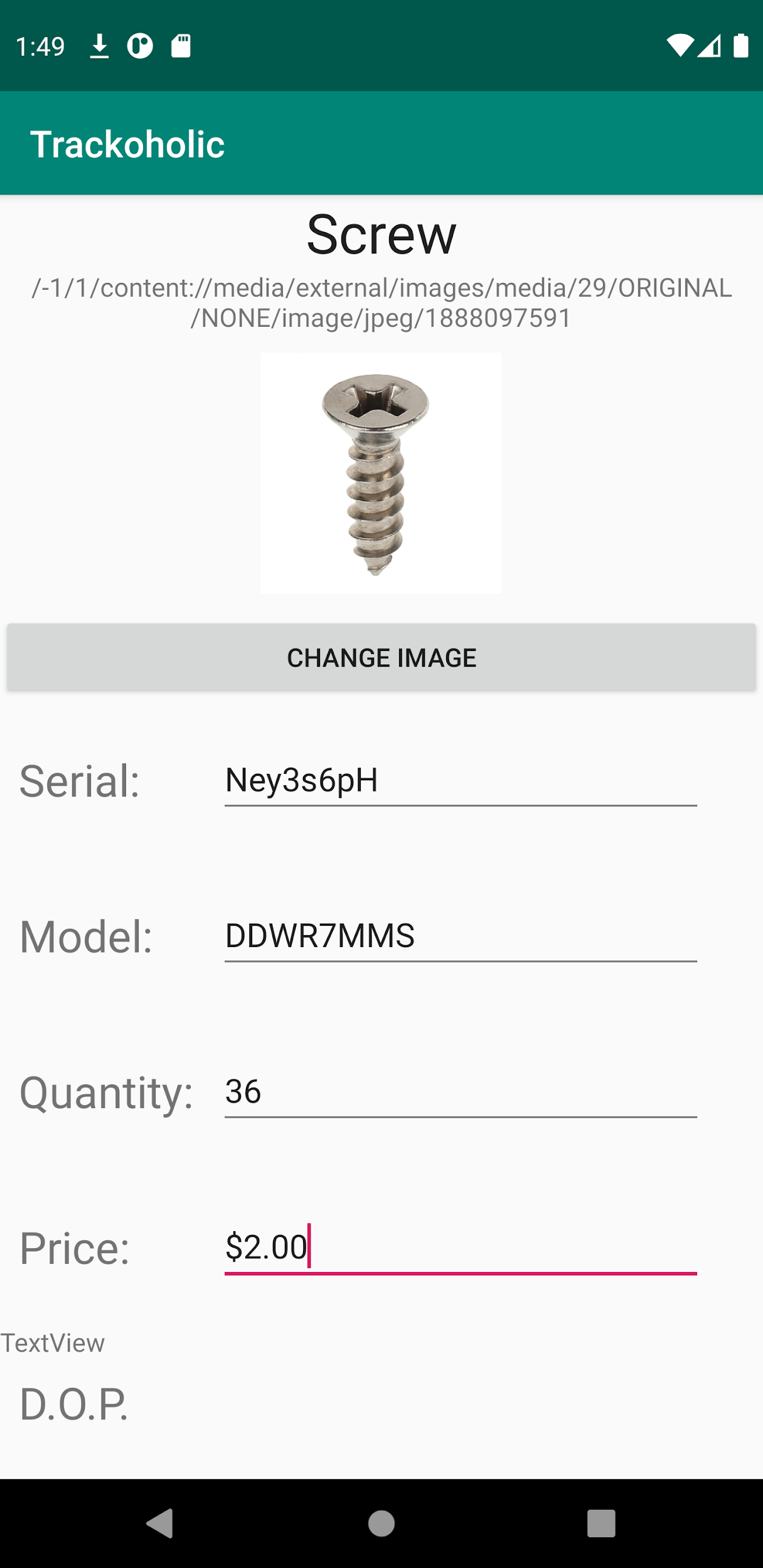
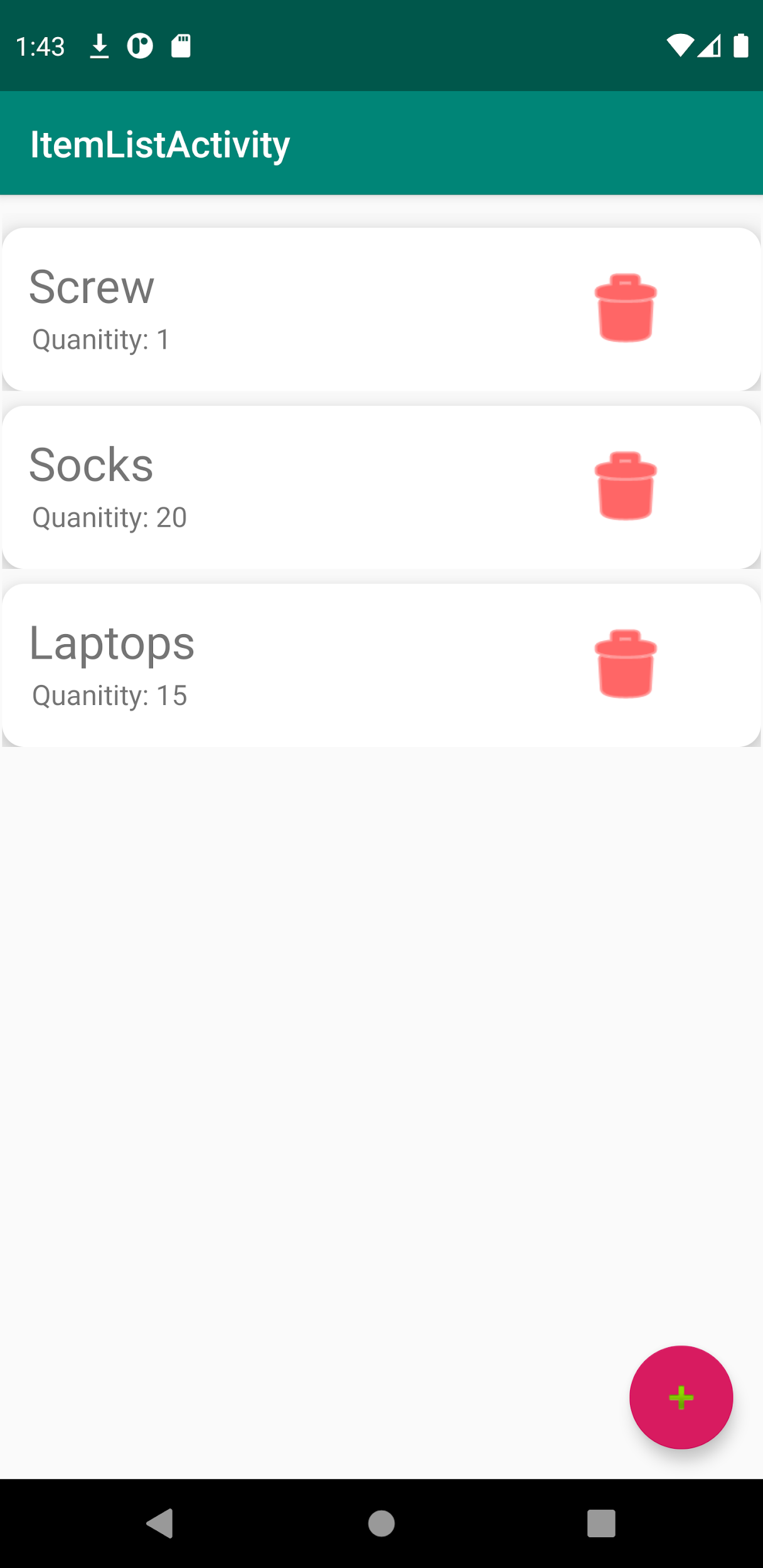
5. COMPONENT DESIGN

The homeview class allows the user to choose between a business profile as well as a personal profile. Following that, each profile is taken to similar screens, both of which are implemented by our itemlistactivity class. In this class, the user may create or delete classes, and may also enter the next screen, which is held by the itemviewactivity class. Each component is able to communicate with each other through our recyclerviewadapter class and all the data needs to be stored gets placed in arraylists, serialized, and then placed in a database by the database handler classes.

6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

Once the user opens the application, they will be able to create a profile that shows where their data is being stored. Once they open that profile, they will be able to add different locations in which they can place items. In the locations page, the user is able to add items. These items have descriptions in them, including a photo, barcode, serial number, model, quantity, price, date of purchase, as well as a description. Once complete, the user may also edit any of the user-created instances of profiles, locations, and items.

6.2 Screen Images 

6.3 Screen Objects and Actions

Users may click the business or personal profile in the main screen. The plus button on the bottom right side of the following screen may then be pressed in order to add a new item. The user is prompted to add a name and a description for the item that is to be added. Should the user decide to make a deletion, all it takes is a single tap on the garbage can icon as well as confirming the deletion on the pop up screen. When the user presses an item, they are brought to an item edit screen where they can input information about the item using text fields. The user may then save their changes using the save button at the bottom of their screens.

7. REQUIREMENTS MATRIX

The user interfaces mentioned in 3.1 of the SRS document is taken care of by the activities classes built on android studio. Meanwhile, our databasehandler class takes care of the SQLite requirements described in section 3.2.

|  |  |
| --- | --- |
| SRS | SDD |
| 4.1.1 / 4.1.2 / 4.1.3 | 6.1 |
| 3.1 | 6.3 |
| 3.2 | 4.2 |
|  |  |