Quik-Ventory

Shift Team Six

Class: CSC 424 Software Engineering II

Due Date: April 14, 2021

1.0 Scope

1.1 Identification

The application is called “Quik-Ventory” and is currently version 1.0

1.2 System Overview

The purpose of this application is to provide an easy-to-use inventory management system for use in both homes and in businesses by allowing for both barcode inputs, as well as manual input of inventory items.

1.3 Document Overview

This document is going to contain the requirements, design process, and testing results of the inventory application.

2.0 Referenced Documents

- Making the recycler view clickable - https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwj0-NS0_o3wAhXpnuAKHQxlCv0QwqsBMAF6BAgHEAk&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DwKFJsrdiGS8&usg=AOvVaw0QYiUHxERbfnzBhSpldVyG
- Kotlin documentation - https://kotlinlang.org/docs/home.html
- Android Studio documentation - https://developer.android.com/docs

3.0 Requirements

Requirement ID	Requirement Statement	Design Section Number	Test Section Number
1.0	**Tools/Tech**		
1.1	Android Device		1a
1.2	Computer Device		1b
1.3	Access to Database		1c
2.0	**Manual Inventory**		
2.1	Able to add items manually		2a
2.2	Able to remove items manually		2b
2.3	Able to edit items manually		2c
3.0	**Barcode Inventory**		
3.1	Able to add items using barcode scanner		3a
4.0	**Accounts**		
4.1	Able to create an account		4a
4.2	Able to keep accounts seperate		4b

4.0 Design

How are you going to meet the requirements? This is technical requirements and what the user thinks it should do or behave. Examples of technical requirements are what OS will you use, tools need for development, device drivers, libraries, etc… A user requirement example would be a configuration file that is ASCII rather than binary or always put the OK button in the lower right-hand corner.

In order to ensure that the application would properly run on an Android device, we used Android Studio as our IDE, and made sure to use the more up-to-date version of the OS to create the application. For the database, we pitched some money into hosting an online server so that the application could connect from any device. The database itself used MySQL as it’s basis with MySQL Workbench as the IDE for the database. The connection from database to app was handled using PHP.

Given the nature of a remote server storing user data and a mobile application utilizing cellular data, the Retrofit library developed by Google was used to make API calls to our hosted server/database. The different API endpoints were stored on the server in folders and could be called anywhere in the application. LiveData was also used to make sure the state of the application was always up to date. Shared Preferences allowed us to store in a user’s device their unique ID number upon logging in (The response from the API call to login), all calls to the data base passed along this ID number to embed in SQL queries written in PHP. Another technology implemented that was developed by Google was the RecyclerView library. RecyclerView is great for displaying a list of data, anything that is not immediately visible on the screen is not computed. Given the nature of an inventory application, a list of inventory could possibly be very long, the adoption of RecyclerView allows the application to remain responsive to user input (Scrolling, Clicks, etc).

When developing the application, we wanted to ensure that it was easy to use, so we made sure that the main screen that showed the options was as simple as possible, minimizing how many buttons the user was presented with at once. Each button led to a specific section of the app and handled what it needed to handle.

5.0 Test Plan

1. To test that the database is working correctly.
    a. Make simple, test calls to confirm reaction from the database.
    b. Create multiple user accounts and sign in and out of each.

    c. Create a list of unique items under at least two accounts to test database separation.
    d. Attempt to add and delete items form the database.
2. To test Device compatibility.
    a. Test app on actual android device
    b. Test the scanner to scan items.

Appendix A. Test Results

1a) Has been loaded successfully onto an android device and run with the working components.

1b) Has been able to run on emulator software on PC.

1c) Able to access the database and successfully pull inventory to the device.

2a, 2b, 2c) Unable to test due to not implemented at the time.

3a) Barcode can be scanned and bring up info page.

4a) Works if manually inputted into the database. Unable to test from the app as not implemented.

4b) Accounts are kept separate by different logins.