Quik-Ventory

Shift Team Six
Team Leader:
Cody Ezell
Lead Developer:
Corey Welborn
Graphic Design:
Steven Davis
Database:
Zach Light
Software Engineer:
Matthew Crocker

CSC 424 Software Engineering II
University of Southern Mississippi

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. Users will have account specific inventories and be able to manage their inventories through the application.

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		2a
1.2	Computer Device		-
1.3	Access to Database		1
2.0	**Manual Inventory**		
2.1	Able to view a list of inventory items		1a
2.2	Able to add items manually		1b
2.3	Able to remove items manually		1c
2.4	Able to edit items manually		1d
3.0	**Barcode Inventory**		
3.1	Scanner can obtain information from barcode		2b
3.2	Able to add items using barcode scanner		2bi
3.3	Able to access existing items by scanning the barcode		2bii
4.0	**Accounts**		
4.1	Able to create an account		3a
4.2	Able to keep accounts separate		1a
4.3	Able to log in		3b
4.4	Able to log out		3c
4.5	Able to stay logged in unless logged out		3d

4.0 Design

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 inventories 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 inventory database is working correctly.
    a. View pre-made inventories from multiple test users to test database separation
    b. Attempt to add items to the database
    c. Attempt to delete items form the database.
    d. Attempt to edit items in database
2. To test Device compatibility.
    a. Test app on actual android device
    b. Test that the scanner can obtain information from barcodes
    c. Test the scanner to scan new items into inventory
    d. Test the scanner to scan existing items in inventory
3. To test Accounts
    a. Create multiple user accounts
    b. Sign in to each account
    c. Sign out of each account
    d. Close app while signed in. Open app to see if user remained logged in.

Appendix A. Test Results

1. PASS – Database is working correctly and connecting where connections have been made
    a. PASS – Pre-made test inventories were pulled up correctly and data remains unique to each account.
    b. FAIL – function not implemented due to time constraints
    c. FAIL – function not implemented due to time constraints
    d. FAIL – function not implemented due to time constraints
2. To test Device compatibility.
    a. PASS – app is compatible and works as it should on the android device
    b. PASS – Scanner can retrieve barcode information
        i. FAIL – function not implemented due to time constraints
        ii. FAIL – function not implemented due to time constraints
3. To test Accounts
    a. FAIL – function not implemented due to time constraints
    b. PASS – users can sign in with appropriate account details
    c. PASS – users can manually sign out
    d. PASS – users stay logged in unless manually signed out