**Software Requirements Specification**

for

**My Price History**

**Version 1.2**

**Prepared by:**

**Yousef Baghlaf, Nate Hiblar, Caleb Stanley and Suharto Suharto**

**For:**

**Eastern Washington University, Senior Project**

**2/5/2021**

# Table of Contents

[**Table of Contents**](#_4ny6vlgomlte) **2**

[**Revision History**](#_npv0sk14xbih) **3**

[**Sprint History**](#_ier143g625ld) **4**

[**1. Introduction**](#_kdg5cuj79n4r) **6**

[1.1 Purpose](#_zatucuejaxnf) 6

[1.2 Document Conventions](#_ybwyuhvvnxrm) 6

[1.3 Intended Audience and Reading Suggestions](#_5qe5o318q1al) 6

[1.4 Project Scope](#_2sv9z1c1tm6b) 6

[1.5 References](#_5vak4gd9itih) 6

[**2. Overall Description**](#_piynn3odj5mi) **8**

[2.1 Product Perspective](#_8ov353tdum0h) 8

[2.2 Product Features](#_ryitkceetqd2) 8

[2.3 User Classes and Characteristics](#_grr4riw0bz1u) 8

[2.4 Operating Environment](#_yb26ht2dlg3w) 8

[2.5 Design and Implementation Constraints](#_ke908t9rk8ax) 8

[2.6 User Documentation](#_r3lkim3ybcyc) 8

[2.7 Assumptions and Dependencies](#_qjpqp4n2xuqd) 9

[**3. System Features**](#_xgd15zw7zag3) **10**

[3.1 Install Program](#_cep6e9dhh8hr) 10

[3.2 Account Login](#_s6mladw4eppc) 10

[3.3 Receipt Capture](#_7qnsjd6i5eeu) 12

[3.4 Categorizing Purchases](#_lvuavgciox19) 13

[3.5 Purchase Database](#_e7ss83rf1xkd) 14

[3.6 Manual Data Change](#_v62wxhnj76k6) 15

[3.7 Graphical Representation](#_w0fmffhxo7j7) 17

[4.1 User Interfaces](#_yaytymm7smh7) 19

[4.2 Hardware Interfaces](#_czbkq0y3ewcg) 21

[4.3 Software Interfaces](#_lxspi9v8eq8a) 21

[4.4 Communications Interfaces](#_qjisi0aii37n) 21

[**5. Other Nonfunctional Requirements**](#_obd6fhnpxhwj) **22**

[5.1 Performance Requirements](#_55rg6eab5zy8) 22

[5.2 Safety Requirements](#_vk7atgfqz4sv) 22

[5.3 Security Requirements](#_a04fkifslpd4) 23

[5.4 Software Quality Attributes](#_m4pfzy95zofo) 23

[**6. Miscellaneous Requirements**](#_kvuxo2xhxnwr) **25**

[**7. Flowchart Diagram**](#_q2tvs6x0a2x2) **26**

[**8. Appendices**](#_6gh604zh7sg) **32**

[Appendix A: Analysis Models](#_p7igg3w1vm3l) 32

[Appendix B: Issues List](#_ox35jl8ndm3z) 32

[Appendix C: Abbreviations](#_p83puvi9rs7g) 32

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| First Draft | 1/22/21 | Document Creation | 1.0 |
| Second Draft | 1/29/21 | Added details on user interactions and possible outcomes. | 1.1 |
| Third Draft | 2/5/21 | Flowchart and more user interaction outcomes. Priority hierarchy | 1.2 |

# Sprint History

Sprint velocity is the expected amount of hours until completion. We will use prime numbers to estimate the velocity of sprint hours, with the addition of 0 and 1.

(0, 1, 2, 3, 5, 7, 11, 13, 17... etc)

In progress

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Programmer** | **Sprint Name** | **Velocity** | |
|  |  |  | **Predicted** | **Actual** |
| 2/9/21 | Nate | Barcode and ML Kit research | 5 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Completed

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Programmer** | **Sprint Name** | **Velocity** | |
|  |  |  | **Predicted** | **Actual** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# **1.** **Introduction**

## **1.1** **Purpose**

My Price History software will help users save money by tracking their spending history. Users will be able to see how much they spend on commonly purchased products, how much they spend over varied periods of time, and if products are available at nearby stores for less than the users average price. Version 1.0.

## **1.2** **Document Conventions**

This document is written in Arial font, the size used is 11 points. Important details will be in **bold.**

## **1.3** **Intended Audience and Reading Suggestions**

This SRS (Software Requirement Specification) document is intended for the four person student software development team, and the senior project instructor who will be reading the document as both the client, a grader, and a software development advisor. This SRS may also be used to advertise the functionality of the My Price History application to interested parties. This SRS will have all the information the user needs to get a full understanding of the application and how it functions. **It is important to note that this document could be updated after a new version is released and not before.**

## **1.4** **Project Scope**

The goal of this application is to assist the user in saving money with as little effort as taking a photo of the barcode on their receipt. Tracking small amounts of money is tedious. My Price History software makes it easier to know how much a user spends not only on a specific item before they buy, but also the history of that item. A user can identify patterns in their life where they are spending money based on types of products with filters like “clothing”, “alcohol”, or “fast food”.

## **1.5** **References**

Google’s Machine Learning (ML) Kit for Android will be used to scan barcodes to convert raw data (binary) from receipts into text string..

*https://developers.google.com/ml-kit/reference/android*

Google’s database system, Firebase, will be used to store the data from Google’s ML Kit. User authentication will be achieved using Firebase also.

[*https://firebase.google.com/docs/projects/learn-more*](https://firebase.google.com/docs/projects/learn-more)

Item categorization will be handled by comparing barcode item data against Data Hub, hosted by Global Electronic Party Information Registry.

*https://www.gs1.org/standards/gpc*

Alternative options for UPC (Universal product code) database comparisons :

*https://www.barcode-us.info/official-upc-database-facts/*

# 

# **2.** **Overall Description**

## **2.1** **Product Perspective**

This application is a new, self-contained product. It will be developed by students at Eastern Washington University as their Senior project. The application will be developed for Android phones.

## **2.2** **Product Features**

The application will allow the user to photograph the barcode on a receipt which will read in data such as the products bought, their quantity, and the name of the stores the products were purchased from. Then, the application will store the information in a database to track the users spendings by displaying the product name and price over a single purchase or a given period of time. The user can choose to compare any product in the applications database with products from stores in the area.

## **2.3** **User Classes and Characteristics**

This application is developed for people who want to observe their spending trends, save money, or create accountability for a budget. Also, it is designed in a way that it would be easy to use by everyone as everything is going to be visually understandable by the use of a simple GUI design.

## **2.4** **Operating Environment**

This application is going to run on the Android operating system. Any phone that has Android as its operating system could install this application. **The Android version will be 5.0 (Lollipop).**

## **2.5** **Design and Implementation Constraints**

A constraint of the design and implementation would be using an older version of Android, which could cause troubles using the application such as not storing the required data properly or having inaccurate data showing to the user.

## **2.6** **User Documentation**

There will be a README text file delivered along with the software and it will act as a user manual. The text file is going to teach the user how to efficiently use the application. Also there will be some tips to help the user get the most out of the application.

## **2.7** **Assumptions and Dependencies**

This application will be programmed through Android Studio leveraging Google Firebase for the database.

The minimum mobile SDK will be API 21: Android 5.0 (Lollipop). This will target that application to run on approximately 94% of devices, and won’t suffer too great of loss of functionality.

EAN (European Article Number), UPC-A and UPC-E (Universal Product Codes) barcodes, at a minimum, will be able add data to the application.

UPC lookup for barcode comparison of raw data to items will use Global Electronic Party Information Registry’s database “Data Hub”. This may change as query and response needs to be tested. See alternative options in 1.5, References.

# 

# **3.** **System Features**

## **3.1** **Install Program**

3.1.1 Description and Priority

My Price History will be available to download to an android phone from the Google play store. Google Play is the installing program, but research is needed into what code we need on our side to allow Google Play to install our software. This is the lowest priority feature. It will be the last code we work out, verifying that the program meets Google’s standards. Testing our software will initially occur on virtual machines, then progress to manual installation on the developers personal phones. The final step in this process is publication to the Google Play store.

3.1.1 Stimulus/Response Sequences

* The user interacts with Google Play’s GUI to search for the Application by name or category, then Google Play handles the software installation.

## **3.2** **Account Login**

3.2.1 Description and Priority

In addition to having the database service, Firebase also has the service of authenticating users by many ways, including email verification, phone number verification, and google account verification just to name a few. When the user chooses to create an account, they must verify their email for more security in order for them to successfully create their account. When their account is successfully created, the user will enter their credentials to login to their account. After doing so the user will have access to their data. This function is used to keep users data secure and private. The credentials could be used by multiple people, such as a family, to track a household’s spending. This feature is one of the highest priorities to ensure that user’s data is secure.

* Create account: medium priority
* Enter username and password: medium
* Forgot password: high
* About icon: low

3.2.2 Stimulus/Response Sequences

When the application is opened for the first time it will ask the user for their username and password. If valid credentials are entered, the program will grab the specific data for that person and return it to the application from the cloud. If invalid username and password are entered then the user will be required to retry credentials. If the user does not know their password or is logging in for the first time a password reset or create account option will be available. If the device is not connected to the internet at the time of trying to log in the user will not be able to connect to the cloud making it impossible to use the application until a connection is established again. Once the user has logged in successfully, the App’s main page will be shown. (See Figure 7.1)

* Create account
  + A text field allows the user to type an email address as a user name.
  + A text field allows the user to type a password.
  + Minimum criteria for both email addresses and password are checked.
    - If either is rejected, the user is re-prompted for the rejected field.
  + An email is sent to the given username for email verification.
  + After email verification, the username and password is saved in a database for future reference.
* Enter username
  + Text field entry of an email address.
    - Verify entry is in email format, re-prompt for entry if not.
* Enter password:
  + Text field entry of a password
  + The submitted username is checked against the database, if not found, re-prompt for a new username
  + If the username is found, check the given password against the password saved in the same tuple as the provided user name.
    - If the password fails, re-prompt for a failed user name.
  + A counter feature is needed to record failed attempts and at a given limit new submissions are blocked. Then a prompt for resetting the password through email verification is displayed.
    - This counter feature will need to be time dependent, and check against the current time. The counter will reset after a given time period.
  + A username and password that are successfully matched against data stored in the database will open a new GUI page: the main application page.
* Forgot password:
  + Textfield entry will prompt the user to submit an email address.
  + The email will need a clickable link that sends a confirmation back to the program.
  + Once the confirmation is received, a text field will allow the user to re-enter password that will override the saved password for the given username.
    - Reuse password minimum criteria verification.
* About icon
  + Contains information about the development of the software, including who developed the program and when.
  + High level overview of what the program does.
  + Step by step use of the program. This may be in text in images.

3.2.3 Functional Requirements

REQ-1: This will require a database of usernames and passwords that has been hashed to ensure security.

REQ-2: The login feature will also require internet connection due to the database being in the cloud.

## **3.3** **Receipt Capture**

3.3.1 Description and Priority

The receipt capture feature will allow users to take a photo of their receipt’s barcode to automatically enter the items purchased into their applications database. The user will also have the ability to manually enter data in or modify data captured by the application. This is a high priority feature due to the convenience of having to not manually enter in items purchased. This is a major selling point and feature of our application. It will also be the hardest part of the application to get working properly because it will be outsourced to a third party API.

* Add new receipt (high priority)
* Data verification (medium priority)
* GUI interactions (low priority)

3.3.2 Stimulus/Response Sequences

When the user wants to capture data from their receipt, the application accesses the phone's camera. Once captured, Google’s ML Kit will process the barcode and return data to the main program that will parse and format the date before sending a tuple to the user’s database. If the barcode was not captured properly or the application cannot translate the data, then the app will ask the user to retake the photo. If the barcode persists to fail processing, then the user will be asked if they would like to manually enter the purchased items. There will be a prompt after a successful barcode entry to check if the user wants to verify the data entered into their records. If a feature captures wrong information of one or a few, the user can modify entries without having to manually enter the entire receipt. (Figure 7.4, 7.5)

* Add new receipt
  + Click GUI image button of camera with text underneath: “Add receipt”
  + Checks application permission to access camera
    - If permissions is denied, prompt user to change camera permissions in the phone’s settings
  + Camera program is maximized, My Price History is minimized, and the user photographs the barcode on the receipt.
  + Barcode from the photo is sent to Google’s Machine Learning Kit to translate the barcode into raw data.
    - ML kit verifies if barcode is legible.
  + If barcode is not legible, the program will need a pop-up prompt for the user to retake the barcode photo. Instruct the user the barcode is to be photographed, not the whole receipt.
  + If barcode is decoded into raw data, the ML kit will pass data to a java method to parse the data into usable tuples. Each tuple that is created will be passed to the database.
* Data verification
  + Flags will be created for each tuple passed to the database if there is a missing value in a tuple.
  + For flagged tuples entered into the database (these contain a missing value in one or more data fields), prompt the user with a clickable pop-up if they would like to manually fix data entries.
    - If yes, a new GUI page will need to display each tuple, one at a time, with known and missing values.
    - Missing values will be clickable boxes that allow a text field input of the associated data types.
      * Verify inputs for security, common sense amounts, and data types.
* GUI interactions
  + Fix receipt entries
    - Show previous receipt and allow fixing of more items
  + Return to main page of program

3.3.3 Functional Requirements

REQ-1: Google’s ML kit will translate the barcode of the receipt to text for our use. If the API l cannot translate the image properly, we may ask the user to retake the photo or manually enter in the data if all else fails.

REQ-2: The translated barcode to text must be decoded and sorted by the program to match real items in the database with their attached monetary value. If the item cannot be entered properly then the user will manually add the information necessary.

## **3.4** **Categorizing Purchases**

3.4.1 Description and Priority

The Categorizing Purchases feature will allow the application to sort purchased items into separate categories such as produce, hygiene or alcohol. This will make tracking spending habits more visual and obvious what the user is spending their money on. This is a high priority feature of our application due to the budget planning and visualization that will separate our application from the rest.

* Parse data from barcode (priority medium)
* UPC lookup (priority medium)
* Pass data into database (priority medium)

3.4.2 Stimulus/Response Sequences

When a user enters in receipt data, the program will split and categorize it. It will do this by checking the item name or product code number against a User Product Code (UPC) database. UPC databases to be considered are Global Electronic Party Information Registry, If the product fails to be found, the user will manually enter the category for the item purchased.

* Parse data from the barcode in a java method.
* The UPC value from this method is looked up in the UPC database for item name and categorization.
  + The returned item name and category are then assigned to placeholder variables.
  + Check if name and category variables have values, if null, prompt the user to manually correct data.
  + Prompt will be a GUI popup containing a text field for data entry.
  + Verify inputs for security, common sense amounts, and data types.
* Now that all values are filled in, the data can be passed to the database to add a completed tuple.

3.4.3 Functional Requirements

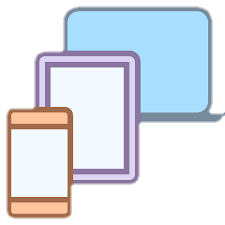
REQ-1: Google’s ML Kit correctly conveys the item's product code for searching the UPC database. The product must exist inside of the UPC database, if not the user must manually enter information.

## **3.5** **Purchase Database**

3.5.1 Description and Priority

The Purchase Database feature will be implemented using Firebase’s cloud-based database service Firestore. It was debated on whether to use SQLite to store the data of the users or Firebase's Firestore database. It turned out that Firestore has many advantages over SQLite, one advantage is that Firestore is a SCHEMA-LESS database service which means that the database could be easily altered as needed without breaking anything. Also, Firebase is made by Google which means that the data will be safely secured in their servers without the user having to worry about them. This will be used to store and get information on purchased items. The database is a top priority as it will hold the data needed for our application to use and display. Without it we could not store data about past purchases with only having information on current purchased items at the time of entry. One more advantage of Firestore databases over SQLite is that since the database is stored in a cloud, it means that the data could be accessed from more than one device. The device could be another phone, tablet, or a computer that runs Chrome OS which has Google Play Store to download the app.

* Firebase database (high priority)



(Database access is cloud based, so multiple devices can access the same users account)

3.5.2 Stimulus/Response Sequences

The program will take in purchase data from the user receipt in order to categorize then upload it to the database. Once in the database, the information will be sorted and accessed by the user to examine. If the information is not stored correctly the data may need to be collected again or the information will be submitted to it again.

* Upload data to cloud

3.5.3 Functional Requirements

REQ-1: Firestore is a cloud-based database that requires an online connection to update or download new data. A constant internet connection will be required for this to work, however if the user loses connection the data already gathered will stay, the app will not be able to update until a connection is established again.

REQ-2: The database must be easily accessed and mutable. If we cannot access the database from the application easily to get data then there will be issues with updating data as well as taking in new data.

## **3.6** **Manual Data Change**

3.6.1 Description and Priority

A major factor in this program succeeding or not is based on the barcode data captured by Google’s machine learning kit. Manually editing data will be necessary due to the complex nature of machine learning integration and ability. Even if the receipt is captured 90% correct, the other 10% can cause major disruption in data representation. This feature will allow the users to manually change or enter in data for the purchased item’s name, price, and category. This will be used in two different areas of our application, the first being right after a Receipt Capture has been taken, and the second being on a separate screen of the application that allows you to modify past data.

* Manual Data Change (medium priority)

3.6.2 Stimulus/Response Sequences

When entering in a receipt, the data will be displayed and the user will have the option to edit that data if errors have occurred. If they choose to edit, then a list of the items will appear and will be able to be selected by the user giving them text box’s or drop down options for the item’s data. Similar options will be available if the user is attempting to change data when not using receipt capture, as in a list of items will be able to be modified based on the time the purchase occurred selected by the user. If the user entered data in incorrectly, then the user can go back through the same steps to modify the data again.

* Manual Data Change
  + Drop down box will allow you to select a category at the top of the screen
    - Second drop down box will allow you to select the time scale (today, last week, last month...)
  + List of all purchases from the filters selected will be shown with their names, date purchased, price, and category
  + If the user finds the item to change, they will click anywhere on the list where the item is located
    - Two buttons will appear, delete and modify
    - If the user clicks anywhere else on the screen it will cancel this
  + If the user choses delete then the item will be deleted from the database
  + If the user choses modify then a new screen will appear that shows 3 text entry boxes for name, price and category. The user will enter into these boxes to change the data
    - There will be an apply button and a cancel button at the bottom. Both return you to the previous screen
    - If the user does not enter anything into the text boxes, a pop up message will appear and tell them “Data entries cannot be blank”

3.6.3

REQ-1: The feature will require internet connection in order to modify the cloud database.

REQ-2: This will require a working interface that allows for large lists of data that can then be modified.

## **3.7** **Graphical Representation**

3.7.1 Description and Priority

A priority of My Price History is to show users when and what they are spending money on. A good way to show this is with graphs that are simple for the user to understand but convey enough detail for analysis of purchases. The goal is to have graphs such as pie charts, bar graphs, and line graphs that are manipulated by user selected time frames. The user will also be able to compare spending on different time spans. This is a feature that incorporates the main theme and reason this app was created, without it data interpretation will be much more difficult. A mockup example: (Figure 4.1)

* Graph view (low priority)

3.7.2 Stimulus/Response Sequences

Once reaching the Graph View screen, the user will be able to select which graph they want to use within a drop down menu at the top. The options to choose from will be bar graph, pie chart, and line graph. Once selected the app will ask for categories you want to view. The user will then select a time frame from which the purchases occurred, and the graph will generate based on the new time span. These time spans include today, last week, last month, last year, lifetime, or specific days the user selects. If the user only selects one category, then the graph will sort by item and could show prices of nearby stores. At the bottom half of the screen there will be a list of categories used in the graph with information on amount spent and percentage of spending for each. The graph will have a key next to it stating the color and category each part represents. The user will also have the ability to check spending from multiple time spans, for example tracking spending for this week compared to spending last week. The data at the bottom will show percentage increase in spending for each category as well as increase or decrease in spending for each category. (Figure 7.3)

* Graph View
  + At the top of the screen there will be a drop down box with multiple graphs to select from
    - Options: Pie chart, Bar graph, Line graph
  + A drop down/text box field will be below. Enter text in for Category or scroll to find Category you want to see
    - Option to see all categories will be available
  + The user will select a time frame from a drop down box below the previous
    - Options: Today, Last week, Last Month, Last Year, Forever, Select days
      * Select days will open a drop down of every day continually generating as you scroll that allows you to select as many items as you want.
  + The graph will generate with a key next to it on the top half of the screen, with a list of the items or categories on the bottom side of the screen
    - User will be able to scroll through the listed items/categories
    - Drop down at the top of the screen showing what graph is currently selected, with the ability to change it to a different graph.
  + Button at top left to back out and make a new graph
  + Plus Icon on top right to add data and compare different time frames
    - On click the user will be given a drop down menu to select a different time frame than the last.
    - Data comparison will be represented with percentage increase or decrease in spending next to the name of item or category of purchase.
    - If comparing months to weeks or weeks to a year, we will break down the purchases to average spending by day for items or categories.

3.7.3

REQ-1: This feature will use the cloud database to access data stored about previous purchases.

REQ-2: This will require a working interface that allows for modifiable and modular graphs that can update with data changes.

**4.** **Interface Requirements**

## **4.1** **User Interfaces**

A graphical user interface that displays the categorization of products as well as having an easily accessible layout to submit new purchases and look at the history of purchases is the ideal GUI for this application. Below is an example of a tab in the application that shows a basic layout that includes a breakdown of recent purchases into a user friendly graph. There will be multiple tabs that will be used to find the history of purchases, the breakdown of the purchases and also importing new purchases. Each of these GUI will be connected with the database of purchases to accurately represent the history. The design should be as sleek and simple as possible. There should be many ways to view this data such as pie graphs, bar graphs and line graphs. The user could also set the time span of the data to look at, for example looking at this last month’s purchases or this entire year's purchases.

4.1.1 Title Screen:

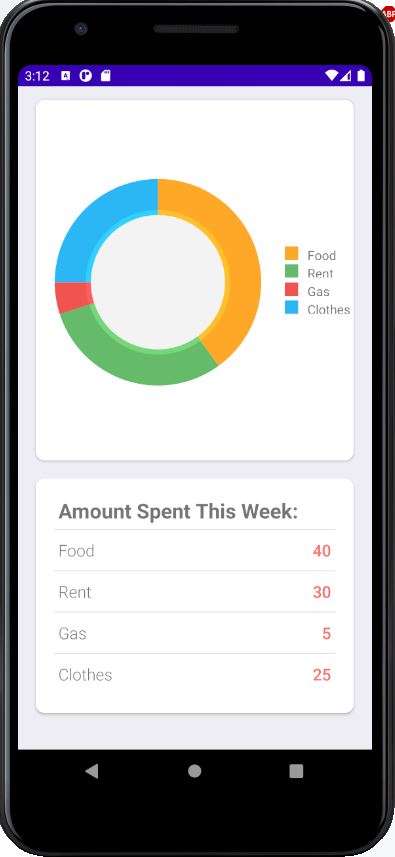
The title screen should be straightforward showing a basic breakdown of purchase history in an easily understandable method like the mockup image below (Figure 4.1a). There should also be an icon in the bottom right that allows the user to enter into Receipt Capture mode, which brings up the phone’s camera. At the top of the screen there will be tabs that allow the user to switch between different screens of the application. Android Studio also includes features that allows the phone to detect swiping features which can be implemented to switch between screens.

4.1.2 Graph View Screen:

The Graph View Screen will allow the user to view purchase data in different graph forms. The user will be able to change the different types of graphs with a dropdown menu at the top. These different types could be pie charts, bar graphs, or line graphs. There will also be a key near the graph describing what the data and graph is representing. There will also be an option to change the time span of the data, such as the last week, month or year.

4.1.3 Edit Entry Screen:

The Edit Entry Screen will be used when users want to change data that is entered into the database, either by mistake of the program or user. This will be separate from the data entry change screen when using receipt capture but will have the same concept. The user will be able to scroll through a list of entries based on specific time constraints and manually change entries. This includes the entries name, price, and category.



(Figure 4.1)

## **4.2** **Hardware Interfaces**

While working with Android Studio we will be able to supply this program to 94.1% of Android device users by using Android 5.0 (Lollipop). This is the most recent version of Android that will still support a majority of phones. Moving up to Android 8 (Oreo) would cut down devices to approximately 60.8%. The phone hardware and application will be communicating together which will require privileges to access the phone’s camera as well as it’s photos.

## **4.3** **Software Interfaces**

This application will require the connection of LeadTools SDK, Firebase Realtime Database, as well as it being created inside of Android studio. This implies that it will be made for the Android operating system. The LeadTools data will be extracted and interpreted in order to be used by our device to display and also to be uploaded to the Firebase Database.

## **4.4** **Communications Interfaces**

This application will require a network connection to login to the web server as well as updating and uploading data to the cloud based database. If the connection is lost while using the application, then the last accessed cache database will be used. The database will be updated and uploaded once connection is established again.

# 

# **5.** **Other Nonfunctional Requirements**

## **5.1** **Performance Requirements**

The framework should be very efficient without a delay, and the defers included are supposed to be minimized. So in each activity reaction of the framework, there are no quick deferrals. If the framework is not being efficient, then this means the framework is being non-intelligent at the moment due to a delay. In the event of opening windows structure, and saving the settings or meetings there is defer much under ten seconds, and of popping blunder messages, if there should be an occurrence of opening data sets, arranging questions and assessment there are no postponements and the activity is acted in under two seconds for arranging/sorting, processing, opening and posting whole files. Likewise when interfacing with the local, the deferral is put together altering with respect to the distance of the two frameworks and the design between them so there is high likelihood that there will be or not a fruitful association in under twenty seconds for sake of good correspondence.

## **5.2** **Safety Requirements**

Data transmission ought to be safely transmitted to the server without adjustments in data. A braking system might be pressure driven sometimes. The braking system could delay the system processing if the product begins not being able to perform multi tasks due to small errors somewhere, and this is how the braking system works. A safety prerequisite might be met by a blend of security capacities, and these might be executed in frameworks of various innovations, for instance, a product based framework alongside the executives techniques, agendas, and approval systems for utilizing it. At the point when a safety capacity is actualized through programming, there likewise should be an equipment stage, in which case a PC framework is essential. At that point, similar requests are made of the whole framework as the product. Moreover, the standard takes into consideration more than one programming security capacity to be actualized on a similar equipment stage, and it forces rules for this.   
  
Safety prerequisites focus on securing the most elevated worth aspects in the kind of mission concerned. For the meeting, catch and takeoff part of a mission it is basic to decide how these high worth components may possibly be undermined during those activities and how the overall safety necessities convert into explicit specialized prerequisites for approach/takeoff directions; and rendezvous control framework design.

Financial record compromise is possible with a database storing purchase history. Access to these records needs to be confidential to the user that inputs the records. We need to verify that the user attempting to access the program is the user that entered the data. There will be a user name, user password, and a recovery feature for forgotten passwords. For simplicity, the user name will be an email that will also serve as the recovery feature for the user password. If the user name (email) is forgotten, there will be no possibility of data access or recovery. This is an accepted trade off to reduce the programming burden on security and focus time elsewhere. This may be re-addressed later into the development of the program.

## **5.3** **Security Requirements**

The fundamental security concern is for clients’ accounts thus appropriate login components ought to be utilized to abstain from hacking. The tablet id enrollment is approaching spam checks for expanding the security. Henceforth, security is given from undesirable utilization of acknowledgment software.

A security prerequisite is an assertion of required security usefulness that guarantees one of various security properties of programming is being fulfilled, Security prerequisites are obtained from industry guidelines, material laws, and a background marked by past weaknesses. Security prerequisites characterize new features or increases to existing highlights to tackle a particular security issue or dispose of a possible weakness.  
  
Security prerequisites give an establishment of confirmed security usefulness for an application. Rather than making a custom way to deal with security for each application, standard security necessities permit designers to reuse the meaning of security controls and best practices. Those equivalent screened security necessities give answers for security issues that have happened before. Necessities exist to forestall the rehash of past security failures.

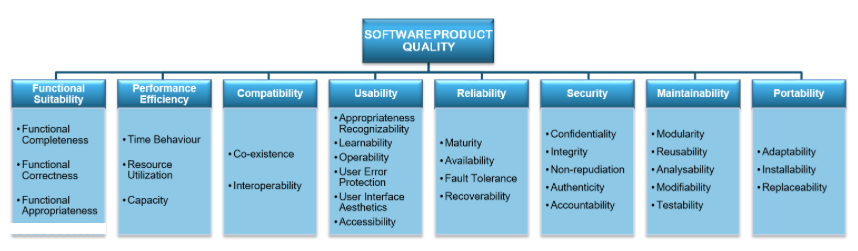
## **5.4** **Software Quality Attributes**

For the availability of safety requirements, when the internet service is getting disrupted while transferring data to the server, the data will be sent twice for verification. For the usability, as the framework is not difficult to deal with and explores in the most anticipated manner without any postponements. All things considered the framework program responds as needs be and crosses over immediately between its states.

A product should have the option to transfer clients’ data to concurrent users every minute of every day everywhere in the world. These non-practical prerequisites urge engineers to consider plan choices that lead to a profoundly adaptable, exceptionally accessible, shortcoming lenient design. Moreover, the necessity to be accessible all around the world implies that product should uphold internationalization, to be restricted for different systems.  
  
A product should be accessible during business hours to a couple hundred clients in a solitary country. These prerequisites propose that the item doesn’t need to be especially adaptable. It presumably doesn’t need to be continually accessible outside “broadened” business hours.  
  
We wanted to declare eight characteristics with the most important quality attributes to launch a software product quality model:

1. Functional suitability: functional fulfillment.
2. Execution proficiency: time conduct.
3. Unwavering quality: accessibility, adaptation to non-critical failure, recoverability.
4. Convenience: operability, availability.
5. Security: classification.
6. Similarity: interoperability.
7. Practicality: modifiability.
8. Versatility: flexibility.

The product quality model diagram(Fig 5.4a) is shown on the next page.



(Figure 5.4a)

# 

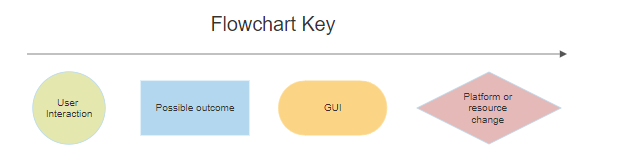
# **6.** **Miscellaneous Requirements**

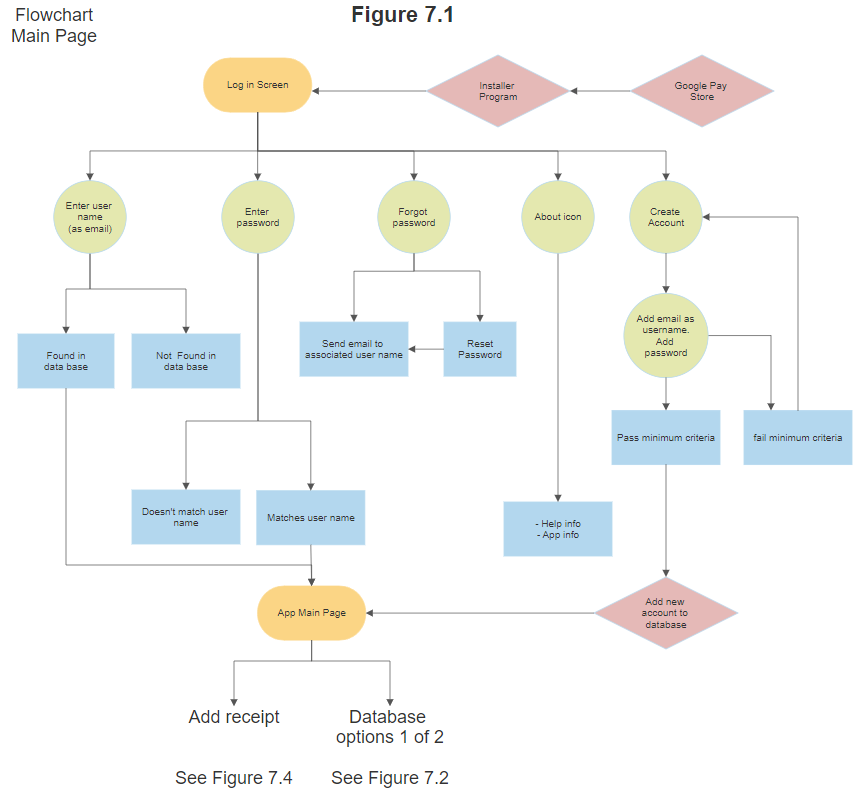
It is required to obtain the permission of the user to use their phone’s camera. Also, it is required to get their permission to use their location service in order to find products in the stores around them.

# 

# **7. Flowchart Diagram**

User interactions with the program are shown in this flowchart. Major program outcomes from user interaction are listed, and major solutions are also shown. This is not an exhaustive diagram of all possible interactions and outcomes, but is intended to provide a general feel of the primary expected features.

****



# 

# 

# 

# 

# 

# 

# **8.** **Appendices**

## Appendix A: Analysis Models

Version two of this SRS will include either a Unified Modeling Language Diagram, Entity Modeling Language Diagram, or another type of software flow diagram.

## Appendix B: Issues List

* When the user is opening the product, then the product could delay the user’s time to wait until the product successfully shows up on the monitor/screen/tablet.
* The developer will get errors in the method’s body function after the developer compiled in the first attempt.
* The lagging often delays our time for waiting.
* If the cache is very large, therefore, it would make the CPU run slower. Hopefully, the cache should be medium in order to be able to make the CPU run smoother.
* Evacuation might go to the wrong segment due to how large the cache’s size is. If it went to the small cache, it would make the hardware’s loading get faster, but this is a rare case. If it went to the large cache, it would make the hardware’s loading get slower.
* This program is unnecessary to be executed by a processor consisting of a set of instructions stored as memory. In fact, it should be executed.

## Appendix C: Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Explanation |
| API | Application Programing Interface |
| EAN | European Article Number |
| JNG | Js Angular |
| MW | Microsoft Word |
| ML | Machine learning |
| PDF | Portable Document Format |
| SDK | Software Development Kit |
| SRS | Software Requirements Specification |
| TXT | Text |
| UPC | Universal Product Code |
| GPC | Global Product Classification |