Store Selector

Team: Blair Billings
Timothy Kalpin
Kurt Kohl
Chris Morgan
Kerrick Staley

Client: Google
Advisor: Manimaran Govindarasu

Table of Contents

Section 1 - Introduction

1.1 Acknowledgements

Our senior design team would like to thank the hard work and efforts of Muthu Muthusrinivasan of Google. We appreciate the opportunity we were given as well as all his help on the project. Our team would also like to thank our advising professor, Dr. Manimaran Govindarasu, for his wise counseling, thoughts, and direction during the entire planning and implementation process.

1.2 Problem Statement

There are many ways that people gather information about shopping deals and pricing information, and there are many ways that people purchase these items that they are interested in. However, there is not one source to view, store, and remember this information that is convenient and easy to use. Businesses desire their information to be seen by all, but there is no guarantee that people who don’t read the newspaper will see their print ads. There is no way to guarantee infrequent internet users will see web advertisements that don’t make it into a version of printed media. There is also no existing system that allows consumers to view pricing information for stores in their local community.

In general, there are too many gaps between buyers of products and businesses that want to get their deals and prices out there for all to see.

1.3 Terms and Abbreviations

[No Major Terms or Abbreviations]

1.4 Target Users

This application will have a large range of users. Ideally, anyone who wants to receive information about deals based on their transaction history or general pricing information from businesses from which they regularly (or infrequently) purchase items could use this application. Realistically, users will have to access the internet quite frequently to use this application. This factor probably narrows the range of users to those people that frequently use their computers, smartphones, or other internet devices.

This application will also be used by store managers, who will be able to upload their store’s pricing and sale information. This will allow managers of physical and local stores (especially smaller stores, which may not have a strong web presence) to reach out to customers over the Internet and draw in new business.

There will also be administrative users who can go into the application and monitor inappropriate behavior, or abuse of the application. This will provide a better experience in the application overall.

1.5 Assumptions and Limitations

1.5.1 Assumptions

- Data on deals and products will be provided to us
- Users desire a centralized location for prices, deals, and product advertisements
- There will be deals that interest the Users of this application
- There will be a way to enter deals on the application

1.5.2 Limitations

- Competing deal advertising products
- Limited means to share the application with people
- Limited data available for database while prototyping
- Lack of time to complete all desired features

1.6 Deliverables

- Web Interface
  - Consumer Interface
  - Store Manager Interface
- Mobile (Android) Interface
- Web Scraper (for getting initial pricing data)

This web application will follow a pretty standard Model, View, Controller pattern of design. It will include a database model, a series of web views, a mobile (Android) interface, and several controllers to handle web services. To make our system more modular, interactions between the layers of the application will be standardized. There will be a rich and easy-to-use web interface presented to the user as well.

Section 2 - Approach and Product Design Results

2.1 Approach Used

2.1.1 Design Objectives

For this project, the system we produce has to be highly modular and scalable so it can be easily adapted to new uses and more users. Our design centers on this objective.

2.1.2 Operating Environment

The system requires a host server to house the application. This could be either a physical machine, or a virtual server hosted on some other platform. It should be consistently reachable through an HTTP connection. Further, the host server must run Apache Tomcat to be able to run our code and modules without issues. The server must also be able to make the resources we plan to implement available to mobile users. Because of this, it should not require the client platform to perform much computation, if any. Also, because it will store client information, the host must ensure some degree of security.

Because this service is intended for use by a large number of users it must be scalable and able to handle not only a large number of connections, but also concurrent updates from several users.

We assume that all users will have either a mobile device (like an Android phone) or a desktop computer with which to use our product. As this is a Google project, we will be targeting mobile platforms, making the Store Selector lend itself to its mobile applications, but especially those with an Android operating system.

2.1.3 User Interface Description

The system shall serve two separate types of end-user:

1. Consumers: The interface that is presented to consumers shall allow them to easily select the products they're interested in from among the entries in the database. After the consumer has selected a list, it shall calculate the amount of money that can be saved by shopping at each nearby store, and display this information on a map for the user to view. The software shall also make suggestions for similar but less expensive products as well as track which products the user likes so that they may more easily construct shopping lists in the future.
2. Sellers: The seller interface shall allow store managers to easily enter and update pricing information for many products at once and ensure that the price entries are correctly matched to the product entries in the database. It shall allow this data to be uploaded in common formats such as .csv, .odt, and .xls; it will accommodate the seller's existing computer systems, rather than requiring those systems to accommodate it.

2.1.4 Functional Requirements

1. The system shall allow users (e.g. shoppers) to find the total price of items they are seeking, using their Android device.
    a. The system shall accept a product name via voice and/or keyboard input.
    b. The system shall match this input to its database of known products.
    c. If there are multiple matching products, the system shall present a list for the user to select from.
        i. For keyboard input, this list will be generated as the user types (i.e. it will autocomplete the input).
    d. The system shall present a means of adjusting item quantity.
    e. The system shall build an on-screen list of items during this process.
    f. The list shall allow deletion of entered items.
    g. Once the user has completed the list, the system shall calculate the total price of the items at various nearby stores, and display this information on a map.
    h. The system shall allow saving, opening, and modification of product lists.
2. The system shall allow users (e.g. store managers) to submit price information for their products via a web interface.
    a. The system shall allow users to upload a price table in .odt, .xls, or .csv format or to link to a Google Doc containing price information.
    b. The system shall allow users to indicate which columns in the table indicate the name and price of each product.
    c. The system shall allow users to view, search, and edit uploaded data.

2.1.5 Non-Functional Requirements

1. The user interface shall be easy to use and aesthetically pleasing.
2. The backend shall process 120 queries per minute, and be able to scale to process 10,000 queries per minute with sufficient hardware.
3. The user shall have a way to ensure pricing data is accurate.
4. The code in the final product shall be modular, readable, and well-documented.

2.1.6 Testing Approach

1. Requirements:
   The obvious goal of any software testing suite is to discover and report as many bugs as possible in the software.
2. Front End Testing:
   Our application will undergo strenuous usability testing on the frontend. As a whole our application needs to be functional, but still very easy to use. Having usability tests will help us gauge our progress on that front. Given the opportunity to log in, we will have test users try to perform specific, necessary functions. They will then report how they feel the process went, and allow us to receive user feedback based on how they want the system to act, and how easy they feel it is to interact with the system. This will also present an opportunity for more bug reporting throughout the system.
3. Back End Testing:
   The largest concern with the back end of our application is stress testing. We need to see if our application can handle many users all writing to, or requesting information from our database simultaneously. We will also use fuzzing to give our application random and incorrect data to see how it handles that data. This will also show us if any random data could crash our application. Before a user base is generated, this kind of testing is impossible without automation. This kind of testing is prevalent, so many options are available to generate and deploy tests of this nature.

2.2 Detailed Design

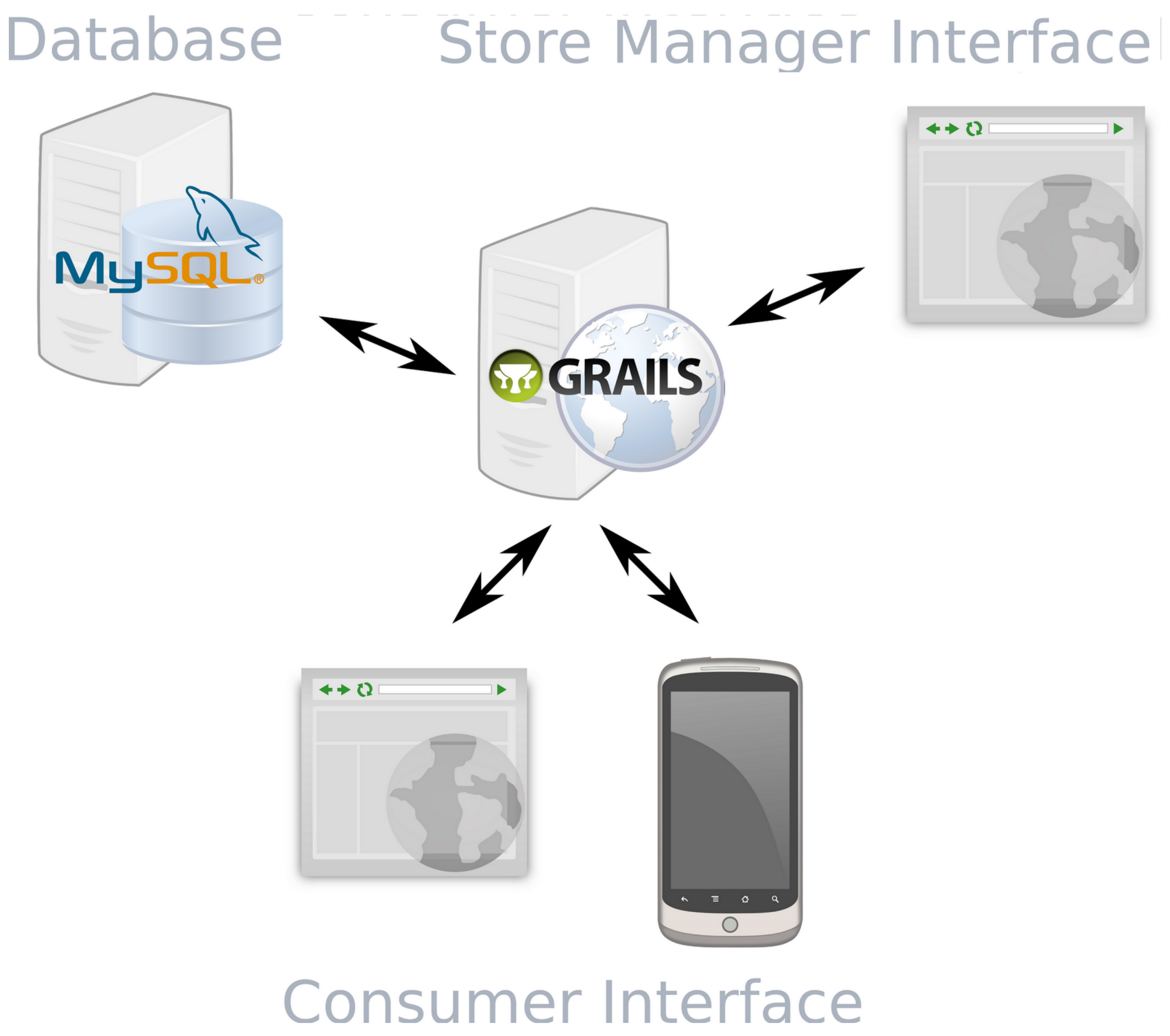

Figure 2.2.a - Web Service diagram

2.2.1 Data Web Service & Database

The data web service manages user interactions with the database. It will take care of requests for information, formatting and relaying those requests on to the database, then formatting the database results. The web service will also keep track of login information and saving individual user’s preferences.

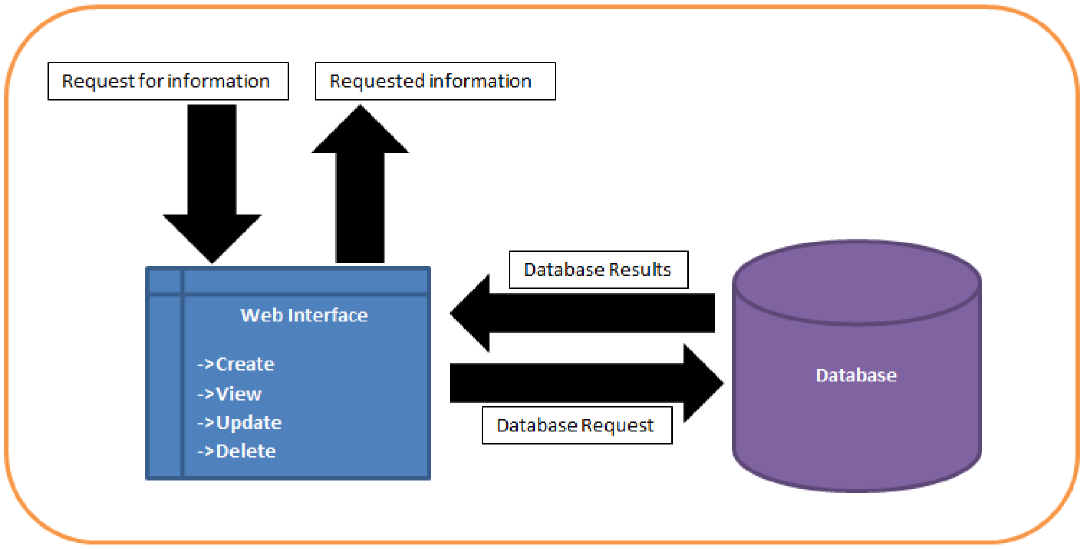

Figure 2.2.1.a - Web Service diagram

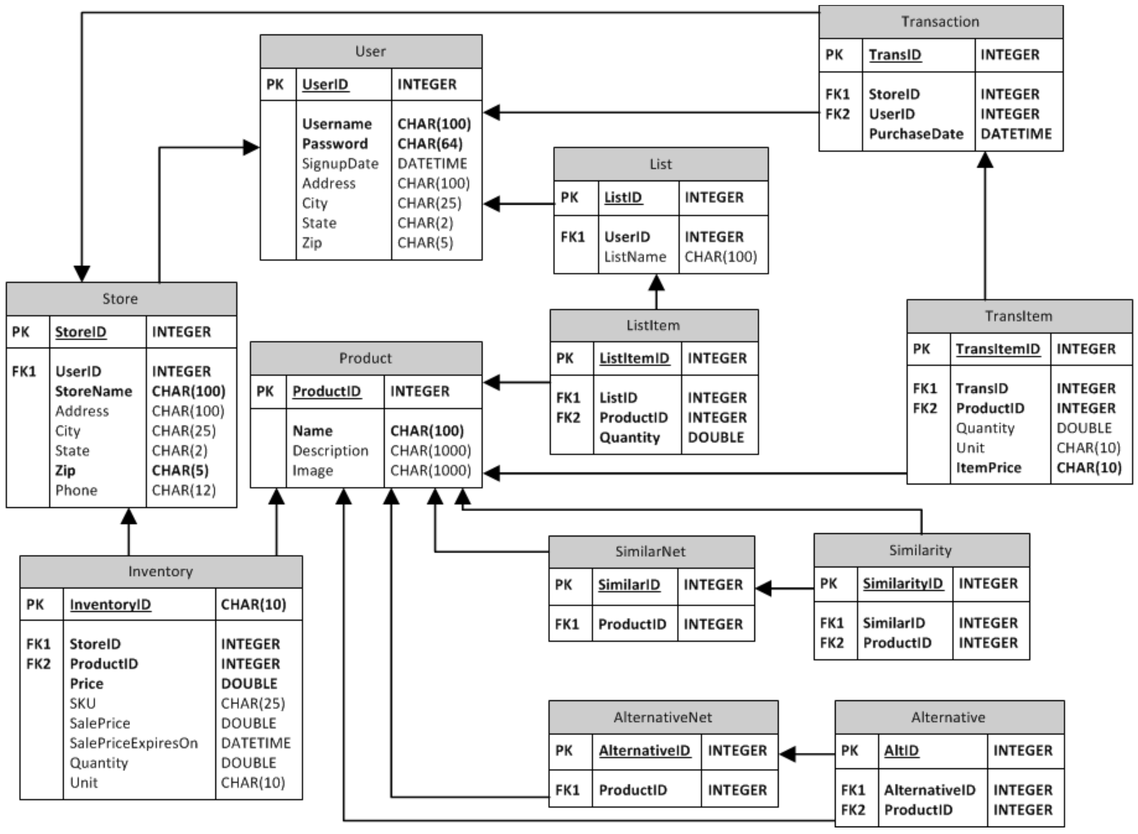

Figure 2.2.1.b: Database Schema

2.2.2 User Interface

Our product will be used by customers via a web page interface targeted both for desktop and mobile browsers, as well as a mobile interface for the Android platform. Through these interfaces, all of our intended functionality will be available to our users: signing in, browsing deals, saving deals, viewing saved deals, and changing preferences.

Section 3 - Standards

3.1 Applicable Standards

- Groovy language API (superset of Java API)
    - Used for web application
- HTML/CSS web standards
    - Used for web application
- Android API
    - Used for native Android application
- Comma-separated values file format
    - Used for uploading inventory information

Section 4 - Future Work

4.1 Community-driven Suggestions

The eventual goal of the Similar and (to an extent) the Alternative relations is that they would be driven by community purchase histories. For example, if many people purchase a toothbrush and toothpaste at the same time, and you purchase a toothbrush or add a toothbrush to your shopping list, the system will suggest that you purchase toothpaste also. Also, if you have a toothbrush and/or toothpaste in your purchase history, the system will be able to notify you if deals on toothbrushes or toothpaste are available.

For our project, we will be adding similar relations to our database manually, as a community-driven system is out of our scope.

4.2 Bundled Offers

Bundled offers (e.g., buy one get one free promotions, etc.) are something we would be easily able to add to the system in future due to the methodologies we have chosen. It would require a trivial addition to the database schema, but the additional data handling code required puts it out of scope for our project.

4.3 Additional Recommendations Regarding Project Continuation or Modification

- Wiki-style deal entry for users
- Deal rating system
- Multi-platform (mobile) support
- Provide an API to allow access to users' saved lists

4.4 Intellectual Property Concerns

This is currently an academic project, but may eventually be used by Google. This raises concerns as to which software systems are available. MySQL is available for use as a database, but is currently released under a FOSS license, and may at some point require availability of the source for this project if it were ever deployed by Google. This is not the only part of the software in which this issue arises, but shows the important issue. This will of course be alleviated if Google substitutes its own software for the libraries we choose to use. At which point the FOSS license will become a non-issue.

Section 5 - Closure Material

5.1 Project Team Information

5.1.1 Client

Google
Muthu Muthusrinivasan
muthup@google.com

5.1.2 Advisor

Manimaran Govindarasu
3227 Coover
Ames, IA 50011-3060
515-294-9175 (Office)
gmani@iastate.edu

5.1.3 Team Members

Blair Billings
Computer Engineering
bbill7@iastate.edu

Chris Morgan
Computer Engineering
cwmorgan@iastate.edu

Timothy Kalpin
Computer Engineering
trkalpin@iastate.edu

Kerrick Staley
Computer Engineering
kerrick@iastate.edu

Kurt Kohl
Computer Engineering
kdkohl@iastate.edu

5.2 Closing Summary

This project plan demonstrates our proposal for the Store Selector and the implementation thereof. The product has been designed such that features can be added without breaking the existing implementation.

The design has been broken down into three modules for ease of implementation and use; additional modules may be added at a later date. The implementation will utilize free tools and languages, but we will require dedicated server systems to handle the hosting, especially considering the potential user scope of any Google product.

Appendix A - Operational Manual

A.1 Setup

Below, are instructions of how to set up the application for development and deployment. The setup instructions split into three major categories: the Git repository, the web application, and the mobile application. The Git repository setup is for making a local repository of the code base, which will allow developers to import the web or mobile application into the respective IDE for development. The web application contains two sections, one to set up a local development workspace for developing the application, while the other is instructions on how to deploy the application to the server. The mobile development setup is fairly straightforward, and can easily be distributed from the Eclipse IDE.

A.1.1 Git Repository

1. The source code is hosted on GitHub at http://github.com/kerrickstaley/store-selector. If you need access to the repository, please send an inquiry to kerrick@kerrickstaley.com.
2. Git usage is detailed at http://git-scm.com/documentation

A.1.2 Web Application

1. Local Development Setup
2. Download Groovy Grails Tool Suite
    a. http://grails.org/products/ggts
3. Clone the GrailsProject from the Github repository
4. Right click in package explorer pane >> import existing projects into workspace >> <location of cloned GrailsProject from Github>
5. Hold down “Ctrl-Alt-Shift-G” to bring up the Grails development console
6. Type “run-app” in the the development console to run the application

A.1.3 Server Setup

1. MySQL
    a. Install MySQL in the manner appropriate for your OS (e.g. using the package manager on a Unix system)
    b. Create a MySQL user and database, and grant all privileges on the database to the user. For demo purposes, the username, password, and database are both “sselector”.
    c. Run the commands from ddl.sql (in the Git repository) in the MySQL command prompt to create the necessary tables for the Store Selector application
2. Tomcat
    a. Install Tomcat in the manner appropriate for your OS (e.g. using the package manager on a Unix system)
    b. Enable the Tomcat manager webapp by editing the Tomcat configuration file
3. Deploy Store Selector
    a. Run the “war” command from within the Grails Tool Suite to package the application as a WAR file
    b. Navigate to the manager webapp in Tomcat and upload the WAR file
    c. Click “Deploy” within the manager webapp to deploy the application

A.1.4 Mobile Application

1. Download latest Eclipse IDE
    a. http://www.eclipse.org/downloads/
2. Download the Android Standard Development Toolkit
    a. http://developer.android.com/sdk/index.html
3. Inside the Android SDK, download the Ice Cream Sandwich OS
4. Clone the Store Selector Application from the Github repository
5. Attach your Android device to your computer using a USB cable
6. In Eclipse
    a. Right click in package explorer pane >> import existing projects into workspace >> <location of cloned Store Selector Application from Github>
    b. Right click on Store Selector application >> Run as >> Android Application

A.2 Demo

Below, there are instructions of how to demo the application split into two major categories (the Web Application and Mobile Application), followed by a division into the two major users (Consumer User and Store Manager User). The outline details critical sections of the application to demonstrate to give an audience a sense of how our application functions.

A.2.1 Web Application

A.2.1.1 Consumer User

1. Registration
    a. After loading the first initial page of the application, click the “Register” button, and fill out the necessary fields. After filling out the fields, click “Create”
2. Login
    a. A separate demonstration from Registration, fill out the email and password fields on the initial Login page. Click “Login” and be re-directed to the MyLists page.
3. My Lists
    a. Create a List
        i. Initially, the “My Lists” page will be indicate that you need to create a list. Click the “Create List” button at the bottom of the page. You will be taken to a separate page to enter a name for your new list.
    b. Add List Items
        i. Click the “Quick Add Item” button at the bottom of the page. When a window pops up, fill in the required fields and click “Add” to include a new item.
    c. Modify (edit/delete) List Items in a given List
        i. Available next to each List Item quantity is a pencil icon.
            1. Click the pencil icon to edit the quantity of the given product.
            2. To save the change, click the checkmark.
        ii. A List Item can be deleted by clicking the trash can on the far right-hand side of the row corresponding to the List Item which you wish to delete.

4. Recent Deals
    a. Click the “Recent Deals” link on the left-hand navigation of any page. Here you can browse deals with the ability to add any product to any of your given Lists. Click the corresponding button that appears next to any given product to choose which List to add the given product to.
5. Similar Items
    a. Click the “Similar Items” link on the left-hand navigation of any page. Similar to Recent Deals, you can browse these Similar Items, and add them to your Lists.
6. Search
    a. Click the “Search for Items” link available on the left-hand navigation of an page. Fill out at least one of the fields presented to you (name and description). Then click the “Search” button. You will be presented with Products that match the search criteria you entered. Again, you will be given the ability to add these Products to a selected List.
7. Locate Items
    a. Select the “Locate Items” link on left-hand navigation of any page. Select at least one of your shopping Lists and click the Locate button.
    b. Next, you will be presented with a table of “Located Items”. By clicking a Store name, a map will pop up giving you directions to that store from your registered address.
    c. Prices per item (for a given store) are displayed to the user.
    d. By clicking the arrow chevrons next to each item, the user will able to see the price of a Product at each store which it exists.

8. Transactions
    a. Create a transaction
        i. Click the “Create Transactions” link available on the left-hand navigation of any page. The next page is divided into two sections. On the left-hand side of the main content exist many drop-down menus for specifying the List Item, quantity, and cost of an item you purchased. The right-hand side of the page displays each of your purchased items and includes the “Done” button which, when clicked, completes your transaction.
    b. View Transaction history
        i. Click the “Transaction History” link available on the left-hand navigation of any page. The page will default to showing all the transactions made in the most recent month. An drop down menu allows the user to choose different months that contain transaction history. The transactions that were performed in the selected month will be listed in a data table in the center of the page. You can click on the transaction total(which is highlighted in blue) to view the items that were purchased in that transaction and the quantity of those items.
9. Account Settings
    a. View/Modify account settings
        i. On the top of each page is a link titled “Account Settings”. Click this link and you will be presented with your account-specific information. You can edit this information by clicking the edit button near the bottom of the page. After clicking edit, you will also see the option to delete your account if desired.

A.2.1.2 Store Manager User

1. Store Management
   b. Upload Inventory
      i. Click the “Upload Inventory” link available on the left-hand navigation of any page.
      ii. Click to choose a file to upload that meets the guidelines given on the page.
   c. Manage Inventory
      i. Click the “Manage Inventory” link available on the left-hand navigation of any page.
      ii. Available next to each Inventory Item is a pencil icon.
         1. Click the pencil icon to edit the given product.
         2. To save the change, click Update.
      iii. An Inventory Item can be deleted by clicking the trash can on the far right-hand side of the row corresponding to the Item which you wish to delete.
2. Account Settings
   a. View/Modify account settings
      iv. On the top of each page is a link titled “Account Settings”. Click this link and you will be presented with your account-specific information. You can edit this information by clicking the edit button near the bottom of the page. After clicking edit, you will also see the option to delete your account if desired.

A.2.2 Mobile Application

Note: The consumer mobile application functions very similarly to the consumer web application, so the process is largely similar to that described in the “Web Application” section. Below the main differences are explained.

1. The main screen of the app contain several buttons that, when clicked, display views corresponding to the various web app panes described above.
2. To return to the main screen, use the (physical) Back button on the Android phone.