1. Introduction

Whenever anyone wants to buy something they ask a very important question: can I find this somewhere else for cheaper? With our app, we hope to find an answer to that question with no hassle. With realistic data, we hope to be able to find and compare a product between different stores and output which has the cheaper product. We want to take into account different deals and calculate the price so the user does not have to do any math, and is presented the better deal with ease. Our app will also allow the user to place orders and make purchases for the products they want to buy from the retailers of their choice, allowing the user to maximize their savings.

1. Problem

The problem we want to address is one of the everyday consumer. No matter what product a consumer wants, they ultimately want the best deal they can get. We want to be able to find out which stores have deals on a certain product and have the consumer be able to purchase the item directly from the app. We believe that everyone deserves the best deal without having to deal with the hassle of calculating discounts or searching through multiple store websites. Finding the cheapest price between different stores and ordering the products in one app can help save time and energy for consumers.

1. Scope/Functionality

There are 3 major parts for this app.

1. Being able to search a store and find a product
2. Being able to compare between 2 products
3. Being able to order directly from the app itself

The ability of searching for a product will be done by either checking the item number or a substring of the item. We plan to have an edit text field where we will check for characters other than letters or numbers to cut down on errors. We will then take the input and run a SQL query such as **SELECT** *product\_name* **FROM** *Products* ***LIKE*** *‘%milk%’;* Doing this allows us to search though the product names from the list of products. When we find the list we can populate the screen with all of the options that fit the search of the consumer. Consumers who input a substring of the item number will not get returned the product, only entering the correct, full item number will return the correct item.

Comparing two stores’ products is a bigger task. Firstly, we are going to have a foreign key for all the stores to carry the same product ID so we can compare the same exact product. Next we have to be able to get the product price between both stores and display both prices on the screen. By pulling the same product ID we guarantee that the products are exactly the same and that there is no discrepancy. We will take into account whether the product has a sale going on right now and calculate the actual price of the product by doing a simple multiplication of the discount rate and the price. We then plan on outputting the Price and Store at which the price is cheapest. If both are the same we plan on just showing both stores and one price on the screen. Properly setting up a store table and product table are key in getting this part of the project to work flawlessly. If those tables are not set up properly one of the major problems we can have is one of not being able to find a product that is in both stores.

The ability to order in-app means that we have to include security measures to keep user data safe and secure. When the user chooses to store the card data we will encrypt the credit card info and save it in the database. We can also prevent SQL Injection by validating the info put into the text fields before running the queries. Ordering should be simple and intuitive. We would like to have a one-click button, which when clicked uses the primary card and purchases all the items in the cart for every store.

The app will be developed for Android devices. Data will be stored in a SQL database on a remote server and data from the app will be synced with this database when online. Data for products will initially be collected from retailers via calls to their catalog API’s and stored on our remote database. When the app wants to request information from the database, an intermediary web service will be called which will pull the data from the database and send it to the app. An intermediary is necessary due to the difficulty in directly communicating with a SQL server in Android.

4. Information needed

The information that we are going to need for this is the user’s log-in info, which consists of an e-mail username and a password. We are also going to need basic user information such as first and last name. To be able to have in-app purchasing we are going to need at least 1 form of payment, either linking a bank account or adding credit card details. So to have that feature we need the payment ID aka the card number or bank account, a payment account name, and a field to differentiate between card or bank account. For products we are going to need the product name, product item number, the price of the product, the product’s UPC, the brand of the product, and which store the product is in. This is necessary to find the store with the lowest price. Once we have a form of payment and a product, we can carry on with the order. For the order we need an order ID, the date of the order and if the order was successfully completed. For the store we need the store ID, the name of the store as well as the URL of the store. Lastly we have a Discounts table. The data we require for the discounts table is an ID for the discount, the percentage of the discount, and if is a recurring weekly deal or not.

Customer data, such as username, password, first name and last name, will be gathered from the user when they create an account. Users will also be responsible for entering information about their payment accounts. Stores will be chosen manually during development of the app and only large retail companies with online shopping services will be picked. Other factors in the selection process include the store’s popularity, the variety of the products it sells, and whether or not a membership is required to shop there. Potential candidates include Wal-Mart, Target, and Best Buy. In order to gather data on products, we will have to utilize the API’s of the chosen retail companies to request information such as name, price, and discounts from each retailer. Data on orders will be created and stored on the database when a customer places an order from the app.