Amrit Kooner		Andy Wong
Benjamin Lang		Ryan Liu
Leo Belanger		John Sun

CheckedOut

Product Vision Statement

Our product is an Android and web-based service that will allow users to search for items from Canadian stores and view their current prices, location and other related information. In addition, users can create multiple lists of differing items that they want to buy and the service will plot a route in order for users to minimize travel time. Furthermore, the service will recommend related items to users so that they can find items that they had not previously thought or heard of. Our service will be useful as it allows users to quickly find out where the best prices are, without having to read through multiple fliers and online websites.

Currently, price comparison services such as Trivago or PCPartPicker only exist for big ticket items; however, for the more commonly bought, lower cost items, these services do not exist or are not intuitive or user friendly. Additionally, these services do not exist for certain item types such as groceries, so our service would be able to fill this void. To distinguish our product from competitors, we will create a user experience that is both intuitive to all users and appealing to use through a clean UI design and responsive controls. We will also implement features that existing products do not have such as automated routing and a recommendation engine.

Problem Statement

The problem of	Searching through multiple fliers/websites to find the cheapest prices for a list of items
Affects	Anyone shopping on a budget or who wants to save money such as seniors, low income families, or young people
The impact of which is	That a lot of time and energy is spent on searching for good deals for exactly what the user wishes to purchase. In addition, after finding the right items, it is still hard to find the right route to the stores.
A successful solution would be	To create a website and mobile application which will allow users to search for items from local stores and create shopping lists. The solution would allow for users to sort by a variety of options including price and store. Afterwards, the solution will create an optimized route in order to minimize cost and time.

Amrit Kooner	Andy Wong
Benjamin Lang	Ryan Liu
Leo Belanger	John Sun

Product Position Statement

For	Frequent users such as young families and senior citizens
Who	Are searching through fliers and online postings in order to find items at the lowest possible prices
Our System	Is an Android and web based service that retrieves item information from a database which is populated via a web crawler
That	Allows users to find items based on price and create shopping lists which can be used to plot routes in order to minimize travel time when shopping
Unlike	Trivago which only compares hotel prices or shopbot.ca which has a cluttered user interface. These services are targeted at online shopping and do not help users buy items such as groceries
Our Product	Uses a clean and responsive UI to provide an enjoyable user experience, incorporating features such as search/sorting options and store/location path finding based on user created shopping lists.

Users: Users include shoppers who are on a budget and/or are looking to minimize their costs when shopping. Our product will not expect the user to have any specific skills or experience to use the service. It will only require basic knowledge of how to use a smartphone or navigate a website, as we provide both interfaces for users to choose from. Our interface will be very intuitive and provide help so that even older users who are not used to computer technologies will be able use our product.

Feature List:
- *Shopping List:* a list from which the user can add and remove items.
- *Optimized Route:* provide a route from store to store that will provide minimize travel time when buying items.
- *Recommendation Engine:* give users suggestions of what they might want to purchase based on previously searched items and saved shopping lists.
- *User Authentication & Account:* provides an account for users and is verified through email confirmation.
- *Saved Lists:* the service will store past purchases so users can quickly find results for previously bought items.
- *Offline Routing:* If users have offline maps, they will be able to use the Optimized Route feature without internet.
- *Autocomplete Search:* The system will help the user correctly spell items that they want to search for.

Amrit Kooner Andy Wong
Benjamin Lang Ryan Liu
Leo Belanger John Sun

Constraints: One of the main constraints on our system is that our web-crawlers should follow the robots.txt specification of each website so that we do not overburden their website. Another constraint is that some websites such as Walmart.ca do not allow web-crawling and thus, we must extract information about their items from their open API.

Scope and Limitations: The features that will be in our product are listed above. Due to time constraints and other considerations, the features that will not be included are:

- iOS App
- Automated Cost Minimization
- Mandatory surveys about shopping experience.
    - Users can leave feedback if they so wish, but will not be forced to
- Mandatory user account requirements.
    - Users may want to try out the service without committing and providing their email
- Forced notifications/emails
    - Users will be able to choose between different options for notifications/emails, mainly for stores they follow, recommended deals, and items on their wish list
- Social Media Integration
    - Users will not be asked to connect social media services such as Facebook or Twitter to their CheckedOut account
    - It might be an option in the future but as of right now it is not within the scope of our implementation to include this

Assumptions and Dependencies: The product will have some dependencies on open source services/libraries. The database will be provided by MongoDB running on Linux and the web crawler, written in Python, will use Beautiful Soup for parsing web pages and urllib for fetching website data. The webserver will be binded to port 80 using an NGINX server. The website will be delivered by an Express.js web server built on Node.js, and it will use google libraries such as Google Maps, which the Android app will also use.

Use Cases

1. Item Search

- **Identification:** Search for the best price for a specific item.
- **Primary Actor:** Shopper
- **Stakeholders and Interests:** Stakeholder is the shopper, and their interests are finding the best deal on the items they're seeking to buy.
- **Preconditions:**
    - N/A
- **Postconditions:**
    - All known items matching the user's search request are displayed.
    - The price, store, and link to the original store are displayed under each item.

- **Main Success Scenario:**
    1. Shopper selects search options and searches for the item
    2. Results are displayed consisting of items matching their search request and options, along with the price and store associated with the item
- **Extensions and Alternative Flows:**
    1a. User misspells the item
        1a1. Autocomplete will help the shopper correctly spell items
    2a. The requested item does not exist in the database or the shopper did not follow the autocomplete suggestion
        2a1. The system will display a message to the user that the item does not exist
- **Open Issues:**
    - N/A

2. Route Optimization

- **Identification:** Find out the best route to take in order to buy desired items.
- **Primary Actor:** Shopper
- **Stakeholders and Interests:** Stakeholder is the shopper, and they're interested in finding the most efficient route to get to the stores offering the items in their list.
- **Preconditions:**
    - A shopping list has been created by the user.
- **Postconditions:**
    - A route will be generated using Google Places API, which suggests the most efficient route depending on transportation options.
- **Main Success Scenario:**
    1. Shopper selects the option to show route on a list they have created
    2. Shopper selects the mode of transportation
    3. A route is displayed to the shopper
    4. Shopper may choose to change the form of transportation, and the route is automatically adjusted to match the new form of transportation
- **Extensions and Alternative Flows:**
    1a. Shopper tries to bring up the map with an empty shopping list
        1a1. Option for route will not be available.
    2a. Shopper doesn't specify mode of transportation.
        2a1. Route calculation defaults to using car as the mode of transportation
- **Open Issues:**
    - Will the program suggest an alternative route if a certain section of road is closed off or has heavy traffic?

3. Account Registration

- **Identification:** Register an account
- **Primary Actor:** Shopper

- **Stakeholders and Interests:** Stakeholder is the shopper, and they're interested in creating an account that will allow them to save shopping lists and receive recommendations on items.
- **Preconditions:**
  - N/A
- **Postconditions:**
  - A new email-confirmed CheckedOut user account is created.
- **Main Success Scenario:**
  1. Shopper goes to registration page and enters important details
     a. Name, Password, and Email
  2. A verification test confirms that the shopper is not a robot
  3. Shopper receives an email from CheckedOut to confirm their account
  4. Shopper is redirected back to home page of CheckedOut
  5. The account can now be logged into to access features like saved lists and recommended items
- **Extensions and Alternative Flows:**
  5a. User tries to login without confirming their account via email
    5a1. User will be prompted to confirm their email or resend the confirmation email
- **Open Issues:**
  - N/A

4. List Creation

- **Identification:** Creating a shopping list.
- **Primary Actor:** Shopper
- **Stakeholders and Interests:** Stakeholder is the shopper, and they're creating a shopping list to store the items they wish to purchase
- **Preconditions:**
  - N/A
- **Postconditions:**
  - A list is generated which the user can save and optionally use to create a route (see use case #2)
- **Main Success Scenario:**
  1. Shopper searches for an item and finds the one they want to buy (see use case #1)
  2. Shopper adds selected item to the list
  3. Steps 1-2 can be repeated for additional items
  4. Once the shopper feels that they are satisfied with the items, they have the option to store the shopping list on the database if they have an account
  5. Shopper may utilize the routing feature to create the optimal route from their location to the stores (see use case #2)
- **Extensions and Alternative Flows:**
  4a. Shopper tries to store Shopping Cart to database without an account

4a1. Shopper will be notified that they cannot store lists without an account and will be prompted to create an account (see use case #3)

- **Open Issues:**
  - N/A

Non-Functional Requirements

Performance Requirements:

The Android app and website both need to be responsive to user input; otherwise the experience will feel sluggish, despite having clean appearance. In particular, the website and app will need to quickly retrieve information from the database in order to display results to the user. Quantitatively, our system should be able to support 1000 connections in quick succession without degrading an individual shopper's performance. In particular, a shopper should not have to wait more than 5 seconds for a search, provided that they have a decent internet connection. This will require proper database organization and query design. Furthermore, the web crawler that populates the database must be able to complete its search regularly to make sure items in the database are up to date.

Security Requirements:

When the user signs up, they need to provide their full name, email, and a password. With this sensitive information, we will have to find a solution to encrypt/secure the information when sending it between the server and the application so that no sensitive data is leaked when communicating across TCP. We will also be performing email verification when a user signs up; this will help prevent bot accounts.

Software Quality Attributes

Correctness and reliability will be key characteristics of the application; specifically, the search results and route optimizations will need to be accurate and up-to-date so that users can rely on the application for their shopping needs. We will be carefully planning all aspects of the development cycle to integrate correctness and reliability as well as maintaining proper documentation our code to ensure flexibility, adaptability and readability. In addition, we can implement modularity to create individual unit tests that can be run every time a change was made to detect possible errors.