Project Analysis Document

Group 3:

Ashraya Regmi, Jesse Cruse, Tyler Roland, Matthew Stevenson

University of Maryland University College

Revision 1.05

June 8 2017

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Description of Changes** |
| 1.00 | 06/08/2017 | Ashraya Regmi | Original Document |
| 1.01 | 06/09/2017 | Jesse Cruse | * Added risks 2 and 3 * Adjusted revision table * Added potential enhancement 6 * Grammatical/wording adjustments * Added to I/O data, processing steps |
| 1.02 | 06/09/2017 | Tyler Roland | Modified text for Figure 2 and added Figure 3: Database Diagram |
| 1.03 | 06/09/2017 | Matthew Stevenson | * Edit Describe Subsystems * Edit Requirements section * Add List Subsystems that implement requirements |
| 1.04 | 06/10/2017 | Jesse Cruse | * Added detail to context diagram * Misc. wording corrections/changes * Increased verbosity of description * Adjusted subsystem list for conformity to functional description list * Adjusted diagram headings to conform to assignment document conventions |
| 1.05 | 06/11/2017 | Jesse Cruse | * Added database subsystem * Added potential enhancement 7 * Re-worded risk 3 |

Project Analysis:

1) Analysis

Our project is to create a web application that keeps track of what the customer has purchased in the past and depending on their shopping history, advise them on what they might purchase during their shopping trip to the store.

System Functional Requirements:

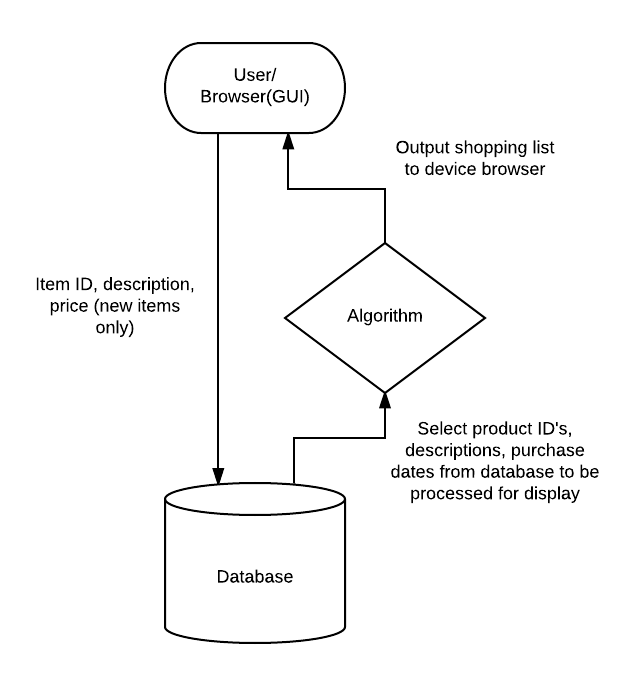
|  |  |
| --- | --- |
| **Requirement #** | **Description** |
| 1 | This program will have a user interface to interact with the web application. The user interface will include a submission form for additional item entries and will also output a shopping list. |
| 2 | The program tracks and stores what the user has bought over time. If an item has already been stored in the database (product ID, description, price), then only new purchase dates will be entered for that item. No other details are required to be stored. |
| 3 | This program will allow the user to generate a shopping list for regular interval trips as well as special trips. The user interface screen will have a button that will generate a list of items that the algorithm deems ought to be bought at that point in time. The shopping list will have check boxes next to each item so that purchase dates are recorded only for items purchased. If an item is displayed on the list but is not needed, the user simply skips checking it and no purchase date will be recorded. |
| 4 | This program will require a backend database to store purchase and user details. The database will store a product ID and product description for each distinct product. |
| 5 | This program will enable the user to email lists to themselves from the website or print the list. |

To go more into detail in this analysis, we will break it down regarding the output data, input data and its processing function:

1. Outside system: the end user
2. Input data: Product ID’s, product names, product prices (for newly entered items)
3. Output data: Shopping list, list of products
4. Data Processing:
   1. Database stores the information of product purchases, including date of purchase.
   2. Algorithmically generate a shopping list based on stored details in the database.
   3. Program displays the generated list on users’ device.

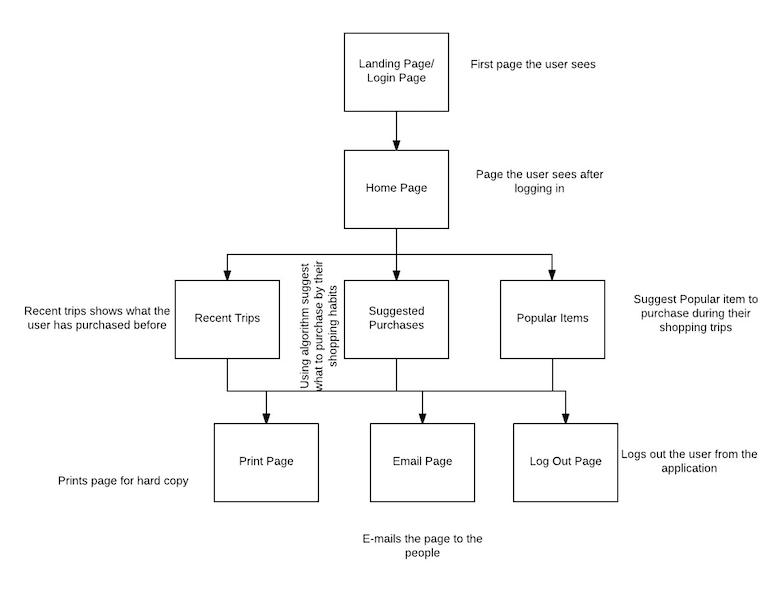
The context diagram shows the above analysis in a diagram below in Figure 1:

**Figure 1  
Context diagram**



Based on the above diagram, we can break down our system in to the following: User Input, Graphical User Interface, and the database that stores and processes the information. These systems are necessary to perform the data to convert the input data to the output data and vice versa.

**Figure 2  
Subsystem diagram**



Subsystem Description:

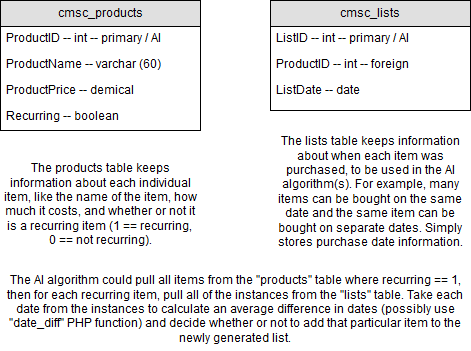
1. Landing Page/Login Page: the user will be provided with a user login page that the user enters the credentials to be authenticated by the database so they can be validated as the user and can proceed with using the website.
2. Home Page: after logging in, the user will have the option to see their recent trips, suggested purchases, popular items, and the ability to add/edit new and current shopping trips.
3. Database: This subsystem stores user purchase item details for later processing. This subsystem integrates with and enables the functionality in most other subsystems.
4. Add/Edit Shopping Trip: this subsystem will allow the user to add additional items to a current or new shopping trip.
5. Recent Trips: the user will be able to see a list of their most recent shopping trips. From this page they will be able to print or email the page as well as log out of their account.
6. Suggested Purchases: this subsystem will prompt the user of any upcoming purchases that they will need to have added to their shopping list.
7. Popular Items: this subsystem will show a list of popular items that the user has purchased more frequently in the past.
8. Print Page: this subsystem will allow the user to print the current page in a hard copy form.
9. Email Page: this subsystem will allow the user to email the current page to themselves or someone else in digital form.
10. Log Out Page: this subsystem will allow the user to log out of the application.

The following table lists the subsystems that implement the requirements.

|  |  |
| --- | --- |
| **Requirement #** | **Description** |
| 1 | Login Page, Home Page |
| 2 | Home Page, Recent Trips, Suggested Purchases, Popular Items |
| 3 | Home Page, Suggested Purchases |
| 4 | Database |
| 5 | Home Page, Recent Trips, Suggested Purchases, Popular Items, Email Page |

**Figure 3**

**Database Diagram**



The database will be the location where user data and lists are stored. Users will add items to their list for use, and when the user clicks save when the shopping trip is complete, the items will be inserted into the cmsc\_items table (if not already present). The purchase date for the saved items will be saved in the cmsc\_lists table for further analysis. When the user chooses to generate a shopping list, all items that the user has chosen to be “recurring” will be pulled from the cmsc\_items table. For each of these items, the average duration between purchases will be determined by selecting all of the purchase dates of that item from the cmsc\_lists table, and then making a decision with if-statements to present this item to the user or not based on the current duration from the next predicted purchase date. The recurrences of these items can be updated by clicking a checkbox in the user’s current list.

**Possible enhancements**:

Some possible enhancements to the application include:

1. Ability to have multiple user accounts
2. Ability to notify/email user what they might need on their next shopping trip
3. Ability to create a shopping list based on the diet of the user
4. Ability for the system to track nutritional information based on the user purchase history for food items
5. Ability to calculate expense per weekly or monthly bases for their shopping trip.
6. Capability to algorithmically reduce/eliminate duplicate item entries in the product table.
7. Capability to integrate app with mobile device geolocation services to implement proximity reminders.

**Possible risks and risk mitigation:**

Risk 1: Password cracking

Mitigation: Password encryption, when user types the password it is not seen visible.

Risk 2: Duplicate product entries within the database, skewing algorithm results.

Mitigation: For the purposes of this project, the database will be preloaded with items that have been filtered of duplicate entries. Although potentially capable, the algorithm design would have to advance beyond the scope of this project to fully implement duplicate entry mitigation techniques.

Risk 3: Newly entered items with too few date entries can provide inaccurate purchase forecasting. This has the potential to render this app pointless if allowed to continue in such a state unaddressed, algorithmically.

Mitigation: For newly entered products, the algorithm makes some assumptions until there is enough purchase history to take over the prediction scheme. The assumption made, for the purposes of this project, is a product purchase interval of five days. Therefore, items that fall under this assumption scheme will continue to populate on shopping lists - although the user will, of course, skip purchasing the item if it is not needed at the time – until there are enough (approximately five) dates in the database to begin predicting a purchase pattern.