Predictive Shopping App Project Requirements  
(Rev. 1.07)

Group 3:

Tyler Roland, Ashraya Regmi, Matthew Stevenson, Jesse Cruse

University of Maryland University College

CMSC 495

28 May 2017

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Description of Changes** |
| 1.00 | 5/28/2017 | Jesse Cruse | Original Document |
| 1.01 | 5/31/2017 | Ashraya Regmi | Misc. edits – no revision data |
| 1.02 | 6/1/2017 | Tyler Roland | Misc. edits – no revision data |
| 1.03 |  | Matthew Stevenson | Misc. edits – no revision data |
| 1.04 | 7/5/2017 | Jesse Cruse | * Fixes to original submitted document:   + Added revision table   + Elaborated requirements details   + Removed email functionality   + Deleted GitHub requirement   + Deleted IDE requirement   + Modified topic verbiage |
| 1.05 | 7/8/2017 | Jesse Cruse | * Clarified topic wording |
| 1.06 | 7/13/2017 | Matthew Stevenson | * In keeping in line with a single subject requirement I have separated requirements and sub requirements into their own requirements. |
| 1.07 | 7/13/2017 | Jesse Cruse | * Removed reference to price parameter |

**Topic**

This project entails creating a shopping app that continually stores shopping details, such as individual item ID and description, and purchase dates every time each item is purchased. This database of shopping items is algorithmically analyzed to predict when items will need to be purchased again to replenish stock by calculating the standard deviation of purchase dates for each item, when the user requests a new list, and adding each item if it is expected to require replenishing. This program will be able to generate a shopping list whenever the user requests a new list to be generated, via a simple button on the respective users account. For the time being, this project will be limited in scope to a website that can be viewed via any browser be it mobile or not. All required data, such as user login credentials and the users’ repository of stored shopping item details will be stored within a MySQL database that will be accessed via frontend and backend coding.

**Requirements**

|  |  |
| --- | --- |
| **Requirement #** | **Description** |
| 1 | Web hosting will be required to fulfill project plans. |
| 2 | This program will have a user interface (UI) to interact with stored data. |
| 3 | The UI will have a recommended purchases page to identify items that are determined by the app to be within purchase range. No user input required here. |
| 4 | The UI will have a recent trips page will show a list of recent trips to the store with an itemized list of items purchased during those trips. No user input required here. |
| 5 | The UI will have a popular items page to show the most purchased items. No user input required here. |
| 6 | The UI will have an add/edit list page to add new items to the database and shopping list. User input of item details required here. |
| 7 | The UI will have a login page used for access control. User credentials required here. |
| 8 | The UI will have a print page button to print an offline list. |
| 9 | This program will require a backend database to store purchase and user details. |
| 10 | User details will be stored in the database using a user ID, password, and email address for credentialing purposes. |
| 11 | Purchase item details will be stored in the database using a product ID, product description, purchase date for each time an item is purchased. |
| 12 | This program will allow the user to generate a shopping list. |
| 13 | The shopping list will be dynamically populated at the time of generation with items that are algorithmically determined to be needed to be restocked within the next five days. |
| 14 | The algorithm will make this determination by calculating a standard deviation of all purchase dates stored within the database utilizing the purchase dates as mentioned above in Requirement 11. |
| 15 | If the algorithm determines that there are not enough purchase dates stored within the database (less than five), then the algorithm will continue automatically add the item to the list until there are enough purchase dates to implement the standard deviation function. |