Smart Shopping List

Abhinandan Deshpande[1], Deepak Ravindra Patil[2], Kanchan Bisht[3] and Prashant Nagdeve[4]

***Abstract*— How many times have you forgotten to buy something from the supermarket, that you had planned, and realized it after reaching home? It is more common than you think. It is in human nature to forget things. Unlike other items, groceries and daily use items are required to be bought again and again. But, the time gap between the two purchases of each item depends on the item, the consumer and some other parameters. We are trying to create a system which will track the usage of each item for each user and maintain a list of items which are needed to be bought again.**

I. INTRODUCTION

The idea of the Smart Shopping List is highly motivated by the increasing stress levels in general population, let alone the students. In such a stressful scenario, forgetfulness is not a very rare sight. A lot of people don't have the time to go to the supermarket everyday, so the visits are weekly or bi-weekly. And forgetting to buy something which is going to be over before the next visit cannot be afforded. Hence, we have come up with an idea to develop a software that tracks the user's requirements and creates a list of things to be bought after each week.

It can be argued that a list of items on the phone can also be re-used every time, but the most important focus of our application is, to have more accuracy and reduce redundancy at the same time. A list of items in the phone can be re-used, but it does not take into account the difference in cycles of each item. Every item in the list has a different cycle of re-purchase. A bottle of dish washing liquid lasts a month, while one carton of milk is good just for a week. So, the next list should have milk on it and not dish washing liquid. If the user keeps re-using the traditional list of items, the accuracy of it will be less, since a dish washing liquid will be bought four times more than it's requirement. Eventually, the user will know of this redundant item on the list, and will get rid of the dish washing liquid from the list. Here comes another problem. "Out of sight is out of mind", not having dish washing liquid on the list will imply that the user has sufficient quantity of it at home. If the user forgets to put it on the list after four weeks, there is a great chance of him forgetting to buy it, and will have to visit the supermarket again.

Having considered the above scenario, we have come-up with a plan to create a software that uses it's intelligence to determine the cycle for each item, based on the user provided data. One important feature that we would like to introduce is, learning user behavior. There can be a case where the dish washing liquid was used up in twenty days instead of a month. In that case, the system will have to adjust the weights, and take this scenario into account for the calculation of the next cycle.

There is a discussion going on in our group, regarding the parameters that affect an item's usage. For example, in the month of December, most of the users go for a vacation or are in the festive mood. Hence, in such a period, the use of dish washing liquid will be less if the user is off to a vacation. But once the user is back in January, his usage will be as usual. The software needs to take care of such scenarios. This is an example of just one parameter. There are many such parameters which determine the usage of every item, and they have different effect on different users. For example, just like in the above scenario, the use of dish washing liquid would increase every December, if the user hosts a feast every December. Hence, we need the software to learn the parameters and their effect on the user, for every item. In order to facilitate this, we will require the user to supply the software with feedback after each of his visit to the supermarket. This will allow the software to adjust its weights.

There are some smart shopping assistants currently in the market, but most of them predict the things that should be bought by the user based on what he has bought already. Those assistants give us more options from the similar category but do not consider repeating the previous order. Here, we are proposing a system that merges our approach onto the traditional shopping assistant for the daily-use items, to create a shopping list for the user.

II. SURVEY

In order to get a concrete proof, that there is a need for such an application, we conducted a survey. After carefully discussing which questions should be asked in the questionnaire, the team had come up with five questions which help to decide whether there really is a need for an application like this. Our first question was for the volunteers to enter their email ids. The reason for this, being the first question, was to get valid data. It has been found that people give random answers when they know that nobody knows who answered them. After ensuring that the data we get from the survey is valid, we started asking questions that will justify the need of this system in today's world. The surveyed people

*This work is a part of Software Engineering project.

[1]Abhinandan Deshpande is a MCS student in North Carolina State University, Raleigh, North Carolina. Email: aadeshp2@ncsu.edu

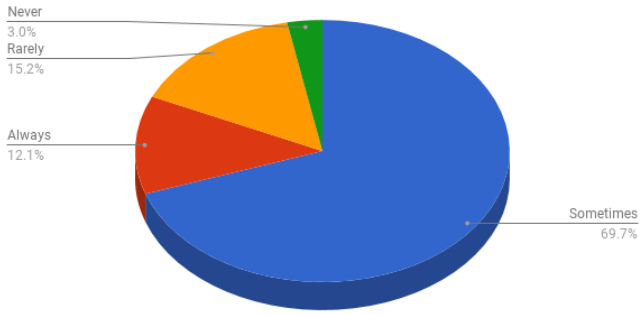
[2]Deepak Ravindra Patil is a MCS student in North Carolina State University, Raleigh, North Carolina.Email: dpatil@ncsu.edu

[3]Kanchan Bisht is a MCS student in North Carolina State University, Raleigh, North Carolina. Email: kbisht@ncsu.edu

[4]Prashant Nagdeve is a MCS student in North Carolina State University, Raleigh, North Carolina. Email: psnagdev@ncsu.edu

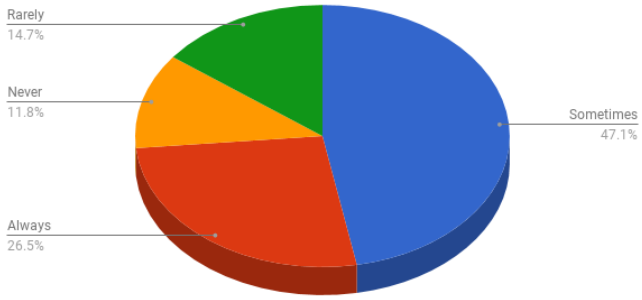
are the students of North Carolina State University. Most of these students are international students who don't have the resources to go for shopping regularly. Hence, they are our target audience.

A. Have you ever gone to buy items from a store but forgotten to purchase one (or more) of those?



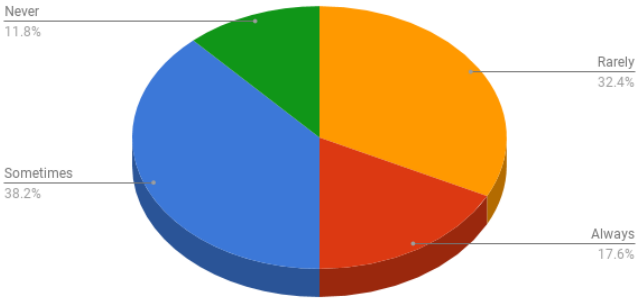
First, we wanted to confirm that we are not the only people who forget to purchase things which we plan to buy. Hence, we asked the people whether they have ever forgotten to buy something from the supermarket, which they had planned to buy earlier. The response to this question tells us that most of the people forget things if not reminded about it. About 97 percent of the respondents tend to forget things. This makes the case for a need of a reminder to people.

B. How often do you make a list of items to be purchased before visiting a store?



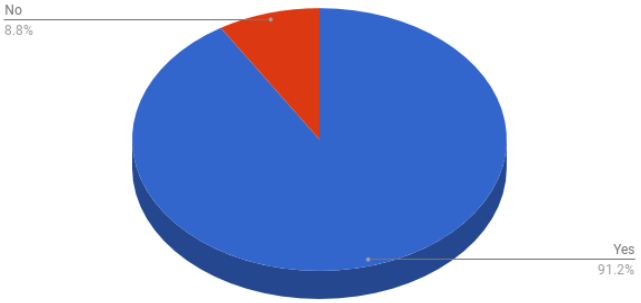
After confirming that there is a need for a reminder system for shoppers to know what things to buy, we asked the respondents whether they use some traditional method to remember the items to be purchased. This tells us whether people are making an effort to resolve this problem. Because, giving a solution to a problem for which no one cares to find a solution is of no use, the suggested software will not be used. Hence, we asked them whether they make a list of items before going to the store. As seen in the Fig.2, around 75 percent people have tried to tackle this problem themselves by making lists before going to the supermarket. From this response we can readily assume that, if we create a list for people to use, it will be received well, since more than 50 percent of the people try it themselves. Making this task of creating a list easier is, what we aim for. In order to do that, we first needed to know the loophole in the current way of making lists.

C. Do you ever forget to put items on your list(if you make one)?



Having known one of the major loopholes in the idea of making lists, we asked people whether they have ever forgotten to put things on their list. The response to this answer gave us the obvious way to proceed to the problem. From Fig. 3, we can see that more than half of the population forgets to put some items on the list. This makes it clear that the root problem is to remember what is to be bought. People usually tend to forget or are too lazy to check for everything. Hence, we figured that this was the main thing that we needed to fix. People need a software to remind them what things to buy, based on their previous behavior and shopping history.

D. Would you like someone to remind you what you need to buy and when?



Finally, we asked everyone whether they would like to have such an application which will remind them of the things to buy. And the result of the response has been unequivocal. More than 90 percent of the people have voted yes for implementing such an application of computer science. This kind of response led the way for us to lock this as our project for the subject. Now, moving forward from this survey, we needed some more data on the existing work done on this subject.

III. Literature Review

The magnanimous increase in the number of people having access to the internet and the increase in the handheld devices [6] has paved the way for a number of applications targeted for these users. Many of such applications exploit the basic everyday needs of a user. The more easy-to-use an application is along with being useful, the more it is expected to be used.

We have gone through the documentation of various patents that assist a consumer with grocery shopping in similar, yet, different ways. The following is the synopsis of these methods/ways:

The documentation of the patent 'Predictive Shopping notifications'[1] briefly enlightens us about the flow of predictive analysis of a list of groceries and items that need to be bought for a consumer. The application analyzes the shopping history of consumers, based on which a purchase pattern for the corresponding consumers is generated. List generated by the application is used to remind the consumers about the products they need to purchase in near future. The list generated by the algorithm is also provided to the potential grocery sellers. If a discount is available on any of the products on the list, the customer is notified of this offer via mobile notification. Due to this smart technique, the customer ends up spending a minimal amount on a particular item. Also, the techniques may remind the consumer to purchase items on the proposed list on sensing that an item may be about to run out. This will, in turn, help the consumer from avoiding the situation of empty stock of important everyday items.

Study of the documentation of another patent, 'Shopping pattern recognition' [2] discusses the technique of data mining to generate a list of items from customers history stored in retailers database. The techniques focus on the fact that the purchase patterns are consumer-specific and along with it, are product-specific, as well as, store-specific. For example, a consumer might buy fruits from Walmart but might buy cornflakes from the local convenience store. The shopping pattern of the consumer is determined by mining the transaction history of the consumer. The list of items that need to be bought in the future is generated based on the past purchases and the respective frequency of each product in the available list data obtained by data mining. The list is also sent to the retailers as a recommendation list. This allows retailers to help customers in shopping different items from their store, maximizing the profit of retailers. Customers save time by not making a list of necessary items and by avoiding extra trips to the shops/stores.

The technique discussed in 'Electronic shopping system and service' [3] makes use of computer servers that are equipped with predictive data collectors that are designed to collect the purchase predictive data of the user, interval of the purchase of the item(s) and frequency of the purchase. There exists a product recommender whose job is to predict the list of products that the user can buy along with the predicted items on his list. This product recommender is implemented with the help of predictive data collector that recommends products similar to what the user has bought in the past. It may even recommend products based on what other users have bought in the past or are inclined to buy in the near future. Product recommender analyzes the consumer's data whether the consumer accepts the recommendation offered regarding any product or not. This information is used to increase its recommendation quality. This is useful for wholesalers, retailers, and consumer packaged goods (CPG) companies.

Another technique [4] utilizes a barcode scanner along with an intelligent base-station (any smart electronic device like a computer, smartphone etc.). The product barcodes are entered into the base-stations with the help of the barcode scanners. The base-stations have a linked database and are connected via the internet. The linked database belongs to a shopping service or a merchant. The database learns the shopping habits of the consumer over a period of time. The shopping list is created with the help of the data available in the database and is stored on the retailer's server. The barcodes are downloaded and required information is gathered. This information is used to fetch the items from a store for a particular customer. Retailers server records the patterns of purchases made by users, and analyses these patterns to learn about the shopping habits of customers. Another advantage of this technique is that customer do not need to search for required items. Retailer suggests the items for the customer based on his past purchase history. This method can be used by the customer to shop via mail-order, telephone-order, in a store or via the internet using the created shopping list.

Another patent [5] describes the use of a budget to formulate a purchase list for a user based on his past receipts from different retail stores, pharmacies etc. It may also gather the user's online browsing information. It determines whether the items on the list can be purchased within the given budget. If true, the list is available to the user as is, otherwise, the items on the list are given a priority rating. Items with higher priority need to be accommodated on the list first, before making room for the ones with lower priority. After this list is available, the user has the option of including other items as well as removing any of the included items. If the user makes any changes to the list, the amendments are monitored by a machine learning engine that adjusts the priority rating of the concerned items, or the date of next shopping period, accordingly. This enables the system to be more accurate and more customer-specific with time.

In light of the summarized information from the various pieces of literature stated above, we conclude that the grocery products need to be purchased at regular intervals of time. The usage and purchase patterns of different users for the same product may vary depending on a number of factors. Some of these factors may be:

- Appetite of the user
- Affinity for the product
- Budget of the user
- Weather conditions (seasons)

- Routine at the time of concern (how busy the user is)
- Stress level (exams, feasts etc.)

IV. Application Review

- BagIQ: It is an application developed to help the user if he/she is trying to stick to a diet or to a budget. The application will keep tabs on how much you spend on each store and would suggest healthier alternatives that will help user to create a better grocery list. Key features:
  - Lists: Build grocery list quickly from an extensive database of items.
  - Aisles: Create custom aisles and reorder aisles from any list.
  - Favorites: Easily add frequently purchased items to new lists, search and filter favorites to build list quickly.
- Specialty Produce: This application helps its users to know about the freshest produce at the grocery store. It has information on history/nutrition/cultivation/seasonal availability and recipes of different products. Key features:
  - History: Keep track of what you have checked out and filter history by list, easily create list from previous buys.
  - Voice Search: Use advanced voice search to add items to your list.
  - All list items: This feature creates an all items list containing only those products that are bought almost during every purchase.
- Ibotta: This application helps the user to reduce the average budget spent by him/her on groceries. It has the collaboration with leading brands and retailers to help its users save money without the hassle of coupon discounts or promo codes. It has discount information about a variety of brands stores. The application boosts that it has a cashback amount of $220,000,000. Key features:
  - Coupons: Search free coupons and view recommended coupons. Email and add coupons to your store loyalty card.
  - Item details: Add price description, weight, quantity and more.
- Grocery iQ: This application helps its users to make a list and save money on all their grocery shopping. It has some key features like a large database of extensive products that can traversed using text, barcode or voice search. The user can sink and share list with other devices and find the products related to the ones on the list. Key features:
  - Multi-barcode scan: Add multiple items by simply pointing your camera at any product barcode.
  - Store Locator: Find nearby grocery stores and add them to your loyalty card.
  - Sync: Sync your list with the GroceryiQ.com and add your friends and family to it.
- Wegmans: This application allows its users to shop smarter with tools that save their time and money. Key features:
  - Shopping lists: Create and save lists that are organized by the aisles based on the items you buy the most.
  - Shopper's Club: Sign in with your account and you can access your grocery list from any device.
  - Recipes: Find, view, review and save your favorite recipes from Wegmans menu magazine right to your own recipe box. And you can even add the ingredients you need to your shopping list.

V. System Overview

A. Implementation Plan

Our project mainly consists of two parts in which one is the client and another is the server. We have built an android application which will act as a client. The user will be allowed to enter details of purchases made by him. In order to achieve high accuracy, a large number of data consisting purchasing details is needed. Hence, it will be beneficial for the user if he enters data after every purchase. This will help our algorithm implemented at server side to generate a most accurate list of predicted items. Server for this application is implemented using Nodejs and, call to the server is made using a REST API. Backend server will have access to datasets that are stored in the server-side database. MongoDB is used for this purpose. By processing and analyzing the data, our recommendation engine will generate results for the next purchase cycle. When the user logs in for the first time, he will be asked to insert list of items. The user will get notification from the application at regular time intervals as per his purchase cycle. In addition to this, the user can access a list of items from the server anytime he wants.

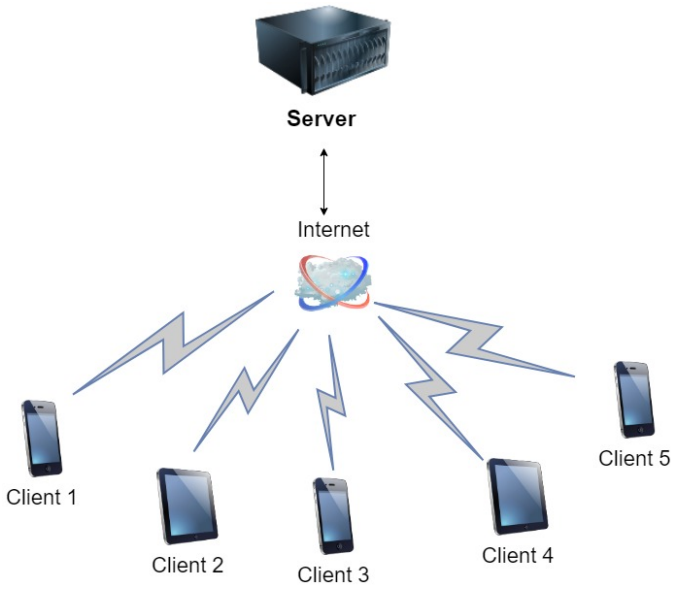


Fig. 1. Client server architecture

B. System Flow

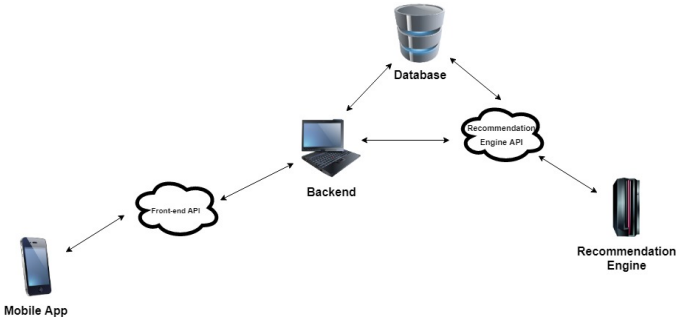


Fig. 2. System architecture

First, the user will sign up for our application. After he signs up, initially he has to add his/her purchase data so that our algorithm can predict the items that need to be bought in the next cycle. After adding multiple purchase history, the application will be able to predict list for next purchase. The user will be able to sign in any time and he can get a list of items to be bought. As the user adds items frequently, he will get more efficient predicted list. When the user enters data, it will be sent to the server and will be stored in NoSQL database. Recommendation engine which contains prediction algorithm will use this data to generate results which will be sent to the user. If the user wishes to erase the records of shopping, the application will ask for confirmation. If he confirms, his records (stored on the server) will be reset. The application will act as a medium to get the input of data in the form of a list of items and to display the predicted list of items. All the data will be analyzed on the server side and analyzed data will be used to display an output on the android application.

C. Development Life-Cycle

1) Requirement Gathering: The first thing that we planned to do was to get the overall idea of user requirements. Hence, we conducted a survey which involved multiple questions. Each of the questions was designed to get a general idea of the application. According to the conducted survey, we found that most of the users are interested in the idea of such a reminder application. If the items missed by the user are essential, the user ends up making another trip to the grocery store. This trip wastes his money and time.Hence most of the users agreed to the idea of the application. We discussed this idea with few people and tried to assess the usability of software.We got suggestions from users about the requirement of software with specific functionalities.Based on the feedback from survey and suggestions from different people,we listed all the functional requirements and focused on incorporating them into our application.

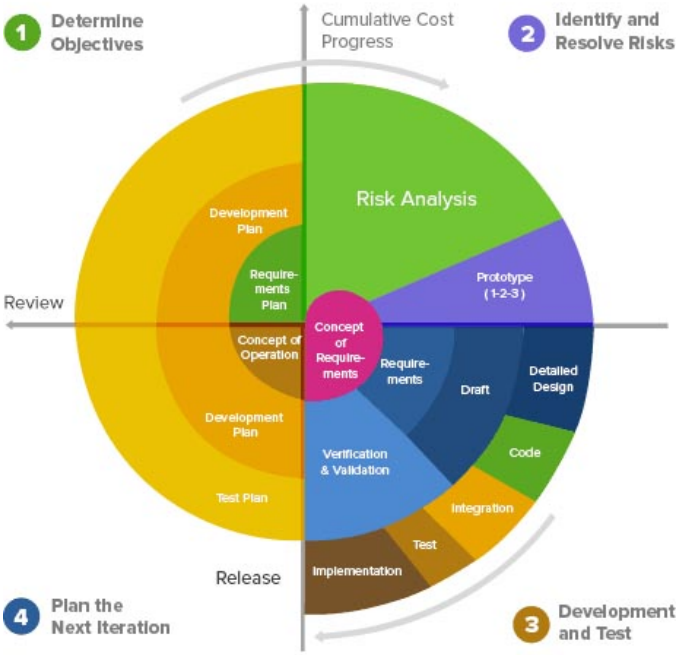


Fig. 3. Development process. Image source:https://onlinedigitaltechnology.com/software-development-risk-aspects-and-success-frequency-on-spiral-model/spiral-model12/

2) Android Application Development: In this stage, we have focused on making an efficient and easy-to-use android application which will allow users to send the data to the server. The data processed on the server will be sent back to users via this android application. We have used the Android Studio for developing the android application. For rest APIs, we have used Volley library and Gson library for parsing json in string and vice-versa.

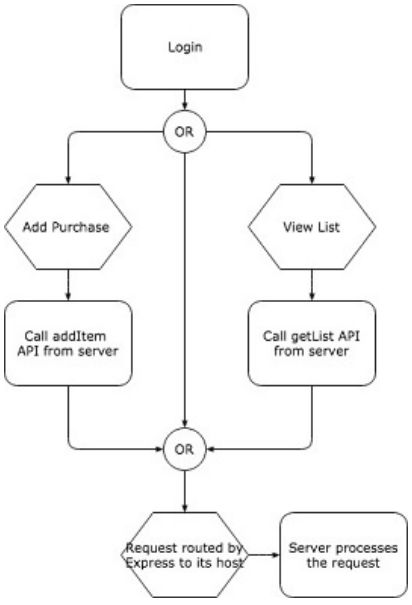


Fig. 4. Flow of Android Application

- Login/Sign Up: After successful installation of the application, the user will be asked for his login initials if he is already registered with our application. If he is not registered, then he'll be asked to sign up with us. If the user has an existing account, all his data and the predicted list will be synced with his new mobile device.

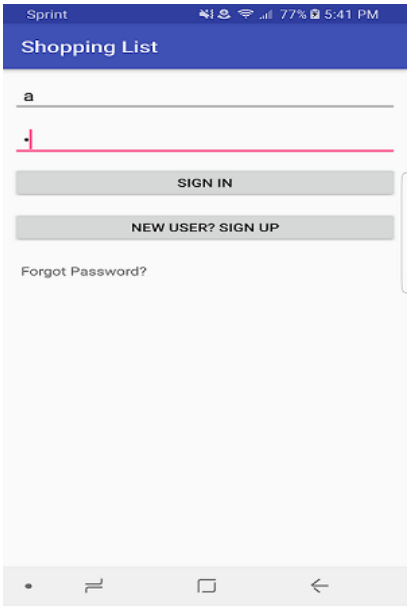


Fig. 5. Sign in

- Data collection: In case of a new user, he/she will be asked to add his/her purchase history so that our application's algorithm can predict the items he/she needs to buy in the next cycle. The algorithm takes 10-15 entries to give a 70% accurate result, that is, 7 out of 10 items in the predicted list (created by our algorithm) were actually needed to be on list for the user. As the user continues to add details of more and more bills, he/she will get a more accurately predicted list. For data collection, user will be asked to fill a form having the following details:
  - Item: The name of the product being purchased.
  - Quantity: The number/weight/volume/amount of the purchased item.
  - Price: This is the per unit price of the item. We have included this attribute to enhance the product in feature, that is, inclusion of budget.
  - Date: This is the most prominent field in the form. The algorithm will track the usage pattern for every product, that is, the average time taken by the individual/family to finish the particular product.

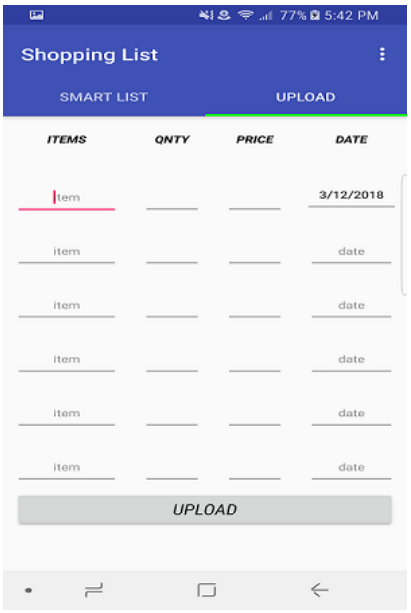


Fig. 6. Upload items

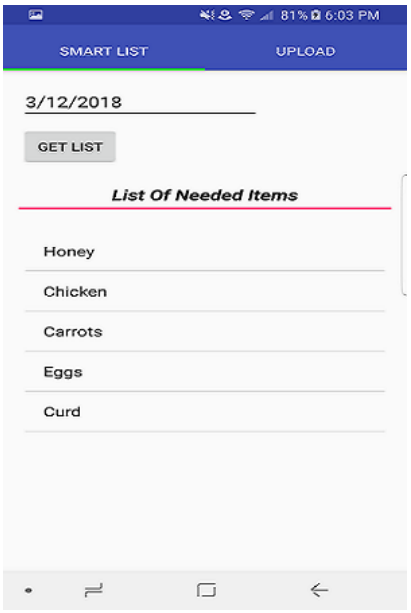


Fig. 7. List of items retrieved

- Storage: When the user enters data, it will be sent to the server and will be stored in the NoSQL database. We have used MongoDB as the database because of its ease of use, quick access and liberty to store data in any format (unlike RDBMS). This is important because the data will vary from product to product. For example, the quantity of milk is calculated in gallons/liters whereas the quantity of eggs is calculated in dozens.
- List generation: A smart list will be created by the algorithm at the server to aid the user in grocery shopping. This list will be user specific.
- Continual feedback: A smart list will be created for the user. Once he makes the next purchase and enter

its details, a comparison will be made between the previous list and the data entered by the user. It will help our algorithm to gain more accuracy and precision. Continual feedback is a real time user interaction that will help our algorithm to evolve overtime to provide better results

- Customization: Our application can be used by individuals/families. Since, the need of groceries or other daily routine products depends upon a number of factors and differs from user to user, number of people in a family, and the current time of the year(exams, holidays, feast, summer, winter). The application can be customized as per the user's need.

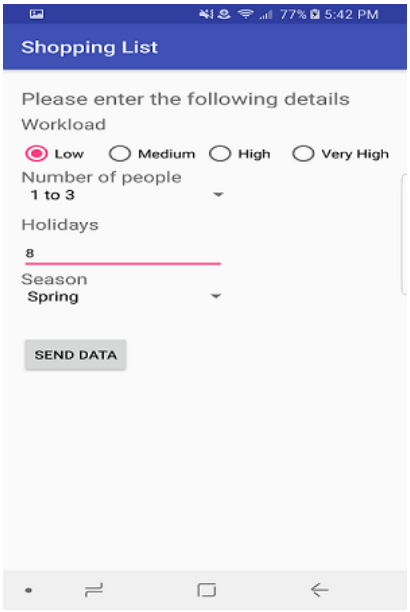


Fig. 8. Survey before uploading items

3) Server-side Implementation: We used NodeJs for the server-side programming, and MongoDB for the database. Use of NodeJs enabled us to use a language that was familiar to all the team members and would help to increase the parallel request processing in case of multiple users trying to create access the service. MongoDB is a NoSql database. We needed to store unstructured data from the users, hence we used a NoSql database. Along with this, we used Express module for routing the API requests from the android app to its exact location in the code.

Fig. 9. Server side workflow 1

We three major API endpoints, which provide some major functionalities:

- Login/ Signup: The first API endpoint is an important one with respect to security. This helps to authenticate the user. This has also enabled us to maintain data privacy of the user.
- Add Item: When the user adds an item to his shopping list in the android app, and presses the upload button, this API is called. Along with the url, the app send some parameters which might influence the usage of the user. For example, the variables like, workload, number of members at home, etc. will be sent in the parameters. These are then taken into consideration while creating the neural network. Basically, this API service updates all the collections.
- Get List: Get list API will take email id and date as its parameters. It goes through each item in the end-date collection and checks if that item gets depleted before the given date. All the items that will get depleted before the given date will be returned as a response to the app.

Another major part of our system is the database. The code is divided into multiple services which updates each collection separately. We have divided our database in six different collections:

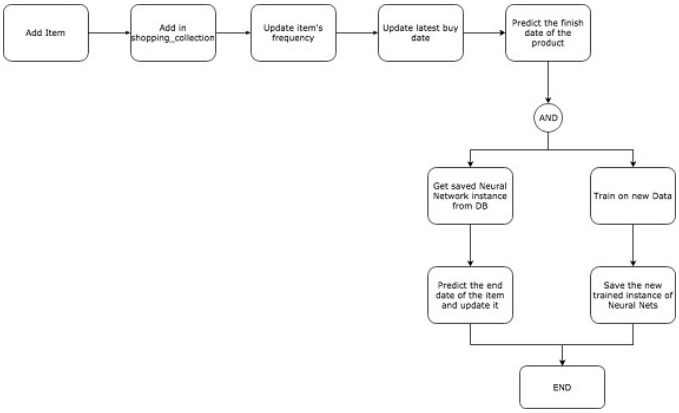


Fig. 10. Server side workflow 2

- User collection: This collection stores the username and password of all the users registered in the system.
- Frequency collection: It stores the purchase history of each item for every user. This collection stores it item-wise.
- Shopping collection: Shopping collection stores history of each purchase by the user. The difference between the shopping and frequency collection is that, shopping collection stores the shopping history date-wise.
- Net collection: This collection is used to store the neural network instance for each item of every user. This instance is used to predict the finishing date of the new items bought.
- Latest buy date collection: Latest buy date collection stores the last date of purchase of each item by the user.
- End date collection: End date collection stores expected finish date for every item of each user.

Apart from the database collections, Neural Networks is the most integral part of our system. We have used 'BrainJs' package for the implementation. In order to speedup the

process of training the network, we have used the developer branch of the BrainJs, which has the functionality of using the already trained network again and again. In this process, we first check if the Neural network is already created of that item for that user. If the network is already created, we run our new shopping entry through the network and get the expected consumption days for that item. This expected consumption days are added to the latest buy date collection, which gives us the expected last date of that item. This end date of the item is then added to the End date collection. And at the same time, we have a new entry in that user's database, which can be used to train the network. Hence, we use the asynchronous call to train the network, on this new data as well. This training is done in the background and then the instance of this trained network is stored in the database (net collection), which is then available to be used for the next purchase.

D. Proof of concept

For our notifier application to work correctly, we decided to look for supervised learning approach as unsupervised variable does not identify target(dependent) variable wiz in this case is number of days for product to last in stock. Unsupervised learning fails for our case as its goal is not to predict value of targeted variable but to look for patterns in similar objects or find groupings that may lead to some understanding about data. In supervised learning approach we shortlisted 3 approaches 1. Regression 2. Decision Tree 3. Neural Networks. For Regression to be used, data should be modeled by linear function, also it is very sensitive to outliers. Hence, we discarded this option and focused was given on decision tree and Neural networks.

One of the major factor for not considering decision tree is fragmentation of data. Whenever there is a split in a tree, it reduces dataset under consideration by considerable amount. This leads to some bias output. Another thing with the decision tree is that, if in data set there is no record for certain variables, there is no interaction between the variables then decision tree provides biased results. Apart from this, dataset under consideration did not have categorical target variable, instead variable had continuous set of values. Hence, decision tree approach was not good for our problem.

Main reason for using neural network approach is very straight forward. Our application/system, takes users data on continues basis. Over a period of time with large amount of data, our system will produce more and more accurate results. Also, accuracy of model built using neural net is pretty good with considerable amount of data. To prove this fact, we developed proof of concept using R.

We gathered data from 9 different users. This data includes regular grocery items such as bread, eggs, cereals, milk. For POC (proof of concept) we focused only on these few attributes because, items listed above are more frequently bought and consumed among the students. Also, it served our purpose of having considerable amount of data for training and testing. Other grocery items such as cheese, juices, butter, peanut butter were also among the times which were considered for POC, but majority data collected from users had very less or no data about these items.

For our dataset, we considered various conditions on which purchase of items depends. These factors include workload on individual, as it is one of the major factors which decides what students buy, make and eat. Many times, people prefer to eat out or in college cafeteria rather than making or buying food from grocery shops. This factor influences majority of purchasing pattern. Season, it also affects purchasing habits of students. For example, cold drinks (aerated drinks), ice creams, juices are the things which are less likely to be bought by individuals in winter season. Also, there are various other factors which influence buying patterns.

No_of_people, this variable refers to number of people which lives in the same household of the registered user. It is important as consumption of goods mainly depends on number of consumer of goods. Week_of_month, it is also one of important variable but often neglected. As mentioned earlier, workload defines what an individual buy, make and eats, most of time workload can be judged using season/month and week of month. For example, first week of October (fall season) has heavy workload due to exams, it is similar for first and second week of December (winter season). Holidays, it also important factor in buying habits of customers. During holidays, some people may travel to explore new, most trending exotic travel destinations or enjoy holidays at home while cooking some exquisite and exotic cuisines. This effect purchasing patterns as, those who went to travel destinations will not buy grocery, conversely those who are cooking some delicious meal may buy additional things or additional quantity of items.

After data was gathered, pre-processing was crucial stage of POC. Pre-processing involved converting categorial data into numeric data. This was done using functionality provided by R packages. After getting all numeric values, the most important part of pre-processing was implemented. It is normalization. If data is not normalized, then the neural network may find it difficult to converge before the maximum number of iterations allowed. Therefore, it is extremely important to normalize the data before training neural network on it.

After normalization, training data is provided to model. But before that, we created formula to be inserted into learning model. For implementing neural network, we used neuralnet library of R. After formula generation, we passed that formula along with training data to neuralnet() function. This function also had one important argument hidden. It is a vector of integers that specifies the number of hidden neurons in each layer. For our POC, we used 3 layers with 10 neurons in each layer.

Predicted results were generated using compute() function provided against the test data that we had. Multiple runs were needed to achieve accuracy which is shown and described using table given.

USER	Bread	Cereals	Eggs	Milk
1	0.67	0.67	0.33	0.75
2	0	1.33	0.33	0.67
3	0.67	1	0.33	0.67
4	0	0.67	0	0.33
5	0.33	0.33	0.33	0
6	0.33	0	0	0
7	0.33	0.25	0.67	0
8	0.33	0.67	0.33	0.33
9	0	0	0	0

Fig. 11. Average Error rate for tested data

We trained and tested for each user data and for each product mentioned in above table. We compared predicted results with test data. For every user and for every product we had test data values and predicted values. We took absolute difference of values of test output and predicted output and then divided them by number of test data rows for that product. This table shows that we got less than 1 average day error for each of user for each product for most of the cases. As our data will grow we will get more and more accurate results and also for more number of products. Thus, result of our POC shows Neural Network works well with limited amount of data giving high accuracy.

VI. Evaluation of Software

Evaluation of any software is necessary to determine if that software works correctly. We have done the evaluation of our software by comparing the list of items predicted by our android application with the simulation of actual data using R language. First, we inserted the data collected from bills to our application and got a predicted list of items generated by algorithm on the server side. After that, we compared it with simulation of same data to check the accuracy. We tried decision tree algorithm to generate the list of predicted items and them Neural network algorithm. After comparing the accuracy of output from both the algorithms, we have decided to use Neural network algorithm for prediction. To check the correctness of algorithm, we have used real world data of bills which was collected from few people. We have tested our application based on the data of around 9 people and found out that, 8 people got a correct prediction using smart shopping application. In some cases, predicted items differed by a single day but not more that.

Based on the result of all these datasets, we have determined the correctness and accuracy of our smart shopping list. Evaluation of an application is done on the following parameters:

- Accuracy & Precision: We have given our application to 9 users to check the accuracy of prediction of application. Based on the feedback provided by users, we conclude that Smart shopping application predicts the data correctly and user always gets required items on the list.
- Scalability: Our application handles increasing number of users and provide a smooth service to each and every user using it. We have used NodeJS for this purpose. As every request is processed in asynchronous mode, NodeJs concurrently handles many requests on the server-side. While testing the application, 9 people simultaneously entered a data and user experience was very smooth. All these people got a predicted list of items correctly.
- Reliability: Once the data is stored on the server, the user can fetch the data anytime he wants. If the user deletes the application accidentally, he does not lose his data. Once he reinstalls the application, he can request for his data and his smart shopping list from the server. The data about the user is maintained along with his username. Hence, whenever the user logs in with his credentials, he is able to see the smart shopping list.
- Security and Privacy: We have used user authentication in order to allow access to users data. If he enters wrong credentials, he cannot access his account. Every user has separate data and one user cannot see the data items of other users.
- Ease of access: We have implemented our idea using an android application. Since, almost everyone has a mobile device of his/her own, it is easy to access this application. We have tweaked applications user Interface as suggested by users through feedback and it is easy to use. Users can easily enter shopping data using application and get the predicted list and refresh the predicted list.

VII. Evaluation Results

A. *Were you able to successfully create an account?*

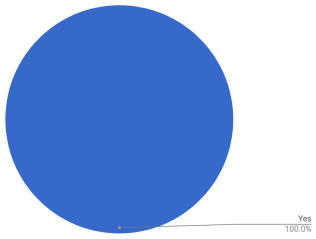


Fig. 12. Chart 1:All users were able to create an account

B. *Were you able to login securely to your account?*

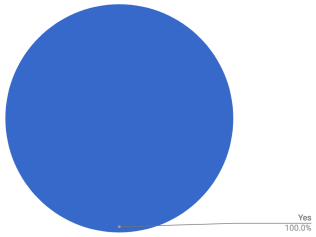


Fig. 13. Chart 2:100 percent secure login

C. Is the application easy to use and intuitive?

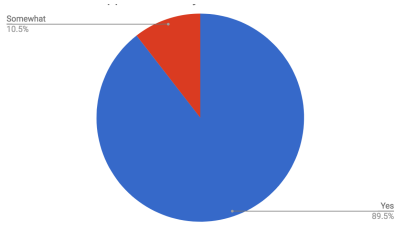


Fig. 14. Chart 3:Ease of Use

D. Were you able to upload your previous grocery bills in the upload tab?

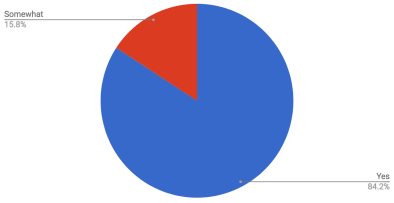


Fig. 15. Chart 4:Successful Uploads

E. How accurate was the smart shopping list predicted by our app?

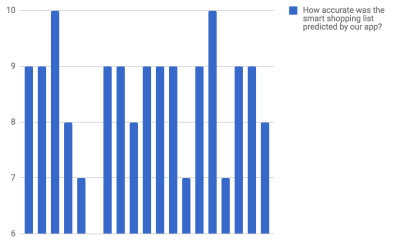


Fig. 16. Chart 5:Accuracy of the Application

F. Were you able to retrieve your smart shopping list at a later point as well?

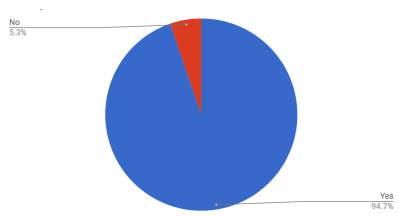


Fig. 17. Chart 6:Successful Retrieval of items from the Server

G. How useful do you think this app is?

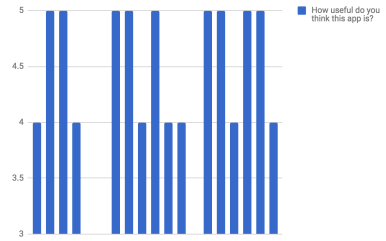


Fig. 18. Chart 7:Usefulness

From the above evaluation results, we conclude that our application is secure, reliable, is intuitive and easy to use, is scalable and with an average accuracy of 86%. Also, 95% respondents consider our app useful.

VIII. CONCLUSIONS

The past experiences of our friends, relatives, acquaintances, as well as, that of our own selves with regard to grocery shopping, led us to the idea of developing smart shopping assistant, in the form of a smart shopping list. After thoughtful consideration, we conducted a survey to gain insights into the usability of smart shopping assistant application. Results have confirmed the need for such an assistant that helps in remembering the shopping list. Our product would also help the users economically by helping them avoid extra trips to the market/stores.

IX. FUTURE SCOPE

Smart Shopping List has a wide scope for future enhancement. There a many features that we could think of that can be there in a smart shopping assistant, many more might emerge with passing time. This product when developed with all its potential features would prove very useful for the users and would change the shopping trend in general. Since we had a limited time period to develop this product, we were not able to develop it to its full extent. There were many features that we wanted to implement, but the lack of time restricted us from doing so. Some of the features that can be included in our product in near future are as follows:

- Bill parser: In order to improve the efficiency of our algorithm, the user needs to enter the details of his/her purchasing. The information includes the date of purchase, items purchased along with their cost and quantity. It is often tiresome for the user to input these details every time he goes for shopping, especially when there are a number of details to be entered (consider the case in which the user buys 50 different items). Since our product is an android application that will be run on a handheld device, scanning a bill will be easier for the user. Imagine with a single scan, the entire bill comprising of a large number of items will be so easily casted as input. After scanning the bill, there can be a bill parser for parsing the bill. It will primarily be a text recognition algorithm that will extract the required

information (name of the item purchased, its quantity, its cost per unit and date of purchase) from the bill and store it into the database linked to the server. This will save the user's time and effort and would make our application easier to use. This would also reduce the chances of errors that can occur on manually entering the data.

- Push notifications: We will use the Twilio API to include a push feature in our android application. The notification can be about a discount on a certain product that is preferred by the user. This can be achieved by sending the details of the user's predicted list to the preferred retailers or supermarkets where the user is inclined to purchase those items from. On receiving the list, retailers can notify the users if there is any discount available in any of the items in a user's list. They can additionally inform the user about other items on discount too. The user will also be able to receive a push notification about a product that is about to finish before the next visit to the shopping mart by making use of this API. This will be extremely useful if an essential item is about to get exhausted before the next visit to the store.
- Integrated Predictive List: This feature takes into account the scenario in which more than one person live together, share groceries and use this application. For example, consider a family, the groceries are usually bought by the parents. So, in order to smoothen the purchasing process and to avoid confusion, we can have one integrated predictive list for the entire family. Then, it wouldn't matter who visits the supermarket, as each member of the family would have access to the integrated shopping list. Also, any member can upload the grocery bills. This feature can also be used by students living together, sharing their groceries. This feature works in real time, that is, the list will keep getting updated if any of the sub-predicted (predicted list of any individual member of the family) list is created.
- Location tracker: This feature will notify a user (if there is a predictive list ready for him, it may not be ready in case he just shopped) if his current location is near to his or her preferred shopping place. In case if there is an integrated predictive list, all the members of the group will be notified about the location of the user if he/she is near to a shopping place that the family prefers. This will allow other members of the family to inform the individual near the shopping store to buy specific items of their need (especially if they are not in the list). Location tracker works in real time since it is synced with the GPS of the hand-held device.
- Budget tracker: Besides serving as a grocery assistant, this application can also be enhanced to monitor the money spent on groceries by an individual or a family. The predictive list will be created based on a priority based selection procedure, that is, the products that have higher priority will always be accommodated in the predictive list first, while the products with lower priorities will either be removed from the list or will not be selected to be a part of the list if the cost of the predictive list exceeds the budget of the user. The priority given to a product will be modified if the user modifies the predictive list, that is, if the user removes a product from the suggested list, the priority of that product would be reduced and vice-versa. This feature also raises an alert if the total cost of the predictive list exceeds the budget. It will raise an alert in such a case, listing all the products that are affecting the budget and may also provide alternative products of different brands as replacement that help reduce the cost.
- Product Recommender: The application will also display products that are not used by the user currently but are popular among other users. This feature can also be used as an indicator for the companies that are developing products, giving them a first-hand review of the product. The information provided by the product recommender will be of great help to the retailers as they can stock up their products in accordance with the popularity among the users. That is, the products with higher popularity rating should be bought in higher stocks or should be stocked up more whereas the ones that are least popular should be bought less by the retailers. This way the loss of the retailers, due to a large number of unsold products crossing their expiry date, would be reduced.

ACKNOWLEDGMENT

We express our sincere gratitude towards our Software Engineering professor, Dr. Timothy Menzies, for giving us the opportunity to freely choose project idea, implement and test it using the tools and technologies of our choice. We also thank our Teaching assistant Amritanshu Agrawal for his valuable guidance and our fellow peers for helping us in requirement gathering by participating in the survey made by us.

REFERENCES

[1] Predictive shopping notifications US 8538807 B2 https://www.google.com/patents/US8538807
[2] Shopping pattern recognition US 20160063511 A1 https://patents.google.com/patent/US20160063511A1/en
[3] Electronic shopping system and service US 20140279208 A1 https://www.google.com/patents/US20140279208
[4] System and method for a complete and convenient shopping experience US 7599855 B2 https://www.google.com/patents/US7599855
[5] Predictive shopping WO 2017156067 A1 https://www.google.com/patents/WO2017156067A1
[6] Rachel Bellamy, Calvin Swart, Wendy A. Kellogg, John Richards,and Jonathan Brezin IBM T. J. Watson Research Center, 'Designing an E-Grocery Application for a Palm Computer: Usability and Interface Issues', IEEE Personal Communications (Volume: 8, Issue: 4, Aug 2001, pg 60-64)
[7] https://play.google.com/store/apps/details?id=com.bagiq.BagIQ
[8] https://play.google.com/store/apps/details?id=com.specialtyproduce.android.sp
[9] https://play.google.com/store/apps/details?id=com.ibotta.android
[10] https://play.google.com/store/apps/details?id=com.coupons.GroceryIQ
[11] https://play.google.com/store/apps/details?id=com.wegmans.wegmansapp
[12] https://rpubs.com/julianhatwell/annr
[13] https://www.r-bloggers.com/r-code-example-for-neural-networks/
[14] https://gist.github.com/AgEconomist/d2d4f28aeaa9b74496076c4dba18ecf6

[15] https://www.kdnuggets.com/2016/08/begineers-guide-neural-networks-r.html
[16] https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf
[17] https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/
[18] https://beckmw.wordpress.com/tag/neural-network/
[19] https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf