Kitchen Assistant - A Grocery Management and Recipe Recommendation Application

Akshay Jain
NC State University
Master of Computer Science
Raleigh, North Carolina
Email: anjain2@ncsu.edu

Jithin John
NC State University
Master of Computer Science
Raleigh, North Carolina
Email: jjohn3@ncsu.edu

Surabhi Sudame
NC State University
Master of Computer Science
Raleigh, North Carolina
Email: ssudame@ncsu.edu

Zaheen Khan
NC State University
Master of Computer Science
Raleigh, North Carolina
Email: mkhan7@ncsu.edu

***Abstract*—Home cooked food is a luxury that most of us crave for yet the effort required both in time and money to manage your kitchen and grocery supplies puts us off from cooking most days. A significant amount of time goes into planning the dishes you choose to cook, based on factors like taste and cuisine preferences, time, money, nutritional value, and diet plans. People buy groceries either in brick and mortar stores or through an internet based application, and cook a variety of dishes using these ingredients. Most people throw away the receipts of food items bought and never follow a systematic process to maintain the inventory or make a schedule of the recipes to be cooked for the rest of the week, even though this might save time, money, and help with healthy diet and eating habits. Most people only know about a handful of dishes that they can cook from these groceries and seldom search the internet for other ones, mainly because this takes an unacceptable amount of time. We present an application for people to built and maintain an inventory of groceries with minimal manual intervention, on their phone, be able to view and edit it as they wish, and get recipe suggestions based on these groceries and user preferences to aid in cooking.**

I. Introduction

As students, we buy a variety of groceries and food items for our daily usage. In many instances, the failure to utilize these items in the best possible way results in those turning stale and eventually being thrown away. This causes a lot of wastage of food and money. At other times, we do not understand the best way to utilize the purchased items. We know none or only a handful of food dishes that can be made from these purchased groceries. The search for a recipe that best utilizes the available items usually requires a good amount of time and is inefficient with todays available resources. Ultimately this also leads to wastage.

Large share of the grocery and kitchen supply purchases are still done at the brick and mortar stores in spite of major business ventures attempting to popularize online grocery shopping. Nevertheless, ventures like Amazon Prime Air could soon see a major shift in the way we shop for grocery[1]. Irrespective of the mode of shopping the common factor is that we rarely keep track of receipts. This makes it very hard to keep track of the ingredients available when we choose to cook at home, leaving us with the only choice of scouring through the fridge and kitchen cabinets only to realize that we have run out of basic ingredients.

Although there are web resources available which allows searching recipes that could be prepared based on the available ingredients this is contingent on the tedious task that befalls on the user to feed into the system all the ingredients that are available in our kitchen. Also, purchasing a couple of additional items may result in a possibility of a lot of dishes. But we rarely have the luxury of time to scour through the recipes to see those common ingredients we should purchase or to keep track of how often we would need to purchase these ingredients.

There is no single resource available today that covers all these kitchen management tasks. At one place, we are able to buy groceries, but we need to go to the next application to find the dishes that can be cooked. Checking out the actual recipe may require using another application while maintaining a list of food items might require another. This is a tedious process, and discourages us from maintaining our inventory, keeping track, and trying out different dishes.

We aim to create a mobile application to solve the above-mentioned problems. Our application is targeted for users of all age, sex, nationality, etc.

II. Literature Review

The research on eating habits in America has revealed that people lack cooking skills and tend to eat outside[2], which negatively impacts the overall health. Unhealthy eating habits outside the home may be attributed to lack of time to plan and prepare meals at home[3]. In fact, lack of time has been cited as the major cause for people not cooking their meals at home[4]. The amount spent on food outside the home now is higher as compared to what it was 30 years ago[5]. Research has also shown that social eating alters our diet and we tend to eat more when we eat outside with people around[6]. This has become a large cause of obesity, an upcoming issue in the health domain. Furthermore, one recent paper showed that for children, half of all energy from fast food is consumed at home, demonstrating that even foods consumed within the home are not necessarily home-cooked[7]. Fast-food consumption has also been linked with the presence of potentially harmful chemicals, such as phthalates, that can contribute to severe adverse health effects[8].

The use of technology to encourage people to prepare and consume home cooked healthy meals will be extremely beneficial. A few applications on Google Play Store have tried to achieve the same. BigOven is one such application, that has a smart grocery list function and the ability to search for, upload, and bookmark recipes. However, the grocery items have to be entered manually. This tedious manual effort can discourage people from using the application. Also, there are no recommendations based on personal choices and attributes such as taste, cuisine choices, location, etc. There are also tons of websites that show the recipe for a particular dish. But searching and then filtering based on users choices can be tedious. Users then tend to go for the easy option of buying food outside.

One popular mobile application is Kitchen Stories, which has a large number of filters to search for recipes based on cooking time, meat choices, etc. Once recipes are selected, it also auto-generates a shopping list. However, we see that these applications automate the steps once you select your recipe. We would like to take an inside-out approach to cooking management. Our aim is to add to this an important feature to auto-generate recipes for users once they purchase the daily groceries, and then recommend recipes based on these groceries.

III. User Study

User study is an important part of planning and developing an application, we created a short survey of ten relevant questions and gathered response from family, friends, and colleagues. This user study was conducted to judge the feasibility and adaptability of a kitchen assistant. This survey included 200+ respondents which helped gain significant insights from the results presented here.

A. Employment status

This question was asked to analyze the background of target audience to better understand the lifestyle choices based on which should help adapt the design of the application. The following chart shows the distribution of people who have responded to the survey.

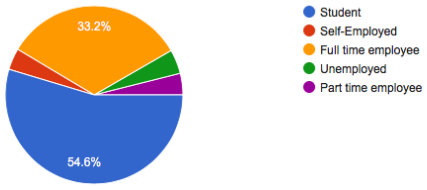

Fig. 1: Employment Distribution Chart.

The results have reaffirmed that the majority of the respondents fall into the two pools Students and Full time employees, which makes up the main target audience, because these two groups tend to have less amount of time available for cooking at home.

B. How often do people cook?

The response to this question gives us insight on how often do people actually cook at home. This possibly shows the frequency at which people will be using our application.

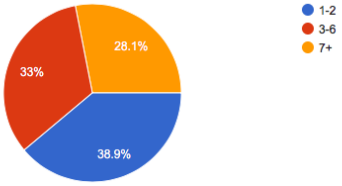

Fig. 2: Frequency of Cooking.

About 28% of the people cook more than 7 meals in a week. These will be frequent users of an application for cooking management. Around 33% people cook 3-6 meals in a week and another 35% of the people cook even less often. With the help of this application, we intend to push a part of this group into the 7+ bracket. The ease of managing inventory and reducing time spent in finding the right recipe could encourage more people to cook at home, especially the groups that have time concerns and budget restrictions.

C. How often do people buy their groceries in a week?

This was asked to gain a measure of how much time people currently spend on buying groceries. In most cases, additional trips to the store can be reduced with efficient planning of shopping lists and better usage of the available ingredients

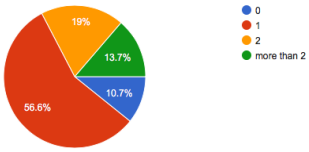

Fig. 3: Frequency of Grocery Shopping.

More than around 33% of people have to shop for groceries on an average 2+ times a week, and our goal would be to reduce this to once a week by helping them maintain shopping lists and predicting the groceries that they should be buying according to their consumption.

From Fig. 5 we can see that people cooking less than 7 meals a week (72% of the users) spend more time for grocery shopping than people who cook 7+ meals per week (28% of the users). This might be further discouraging them from cooking. Thus such an application can actually help a major chunk of users to manage their kitchen and cook more often.

D. How much do people rely on the Internet for cooking?

This chart shows that around 60% of the people somewhat rely on the Internet for cooking. The application is targeted on this bracket of users as it brings ingredients, recipes,

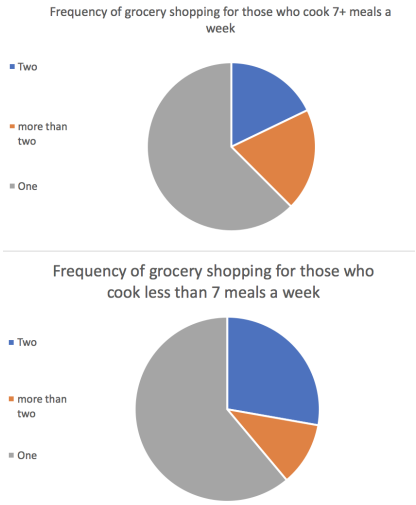

Fig. 4: Frequency of grocery shopping according to their frequency of cooking.

shopping lists, inventory management all in one place rather than them having to scour the Internet to gather information. The user would receive recipe suggestions based on on the user's preferences, eating habits and the ingredients they have available at home. The user can also cook while taking a walk through the recipe as easily and efficiently as possible.

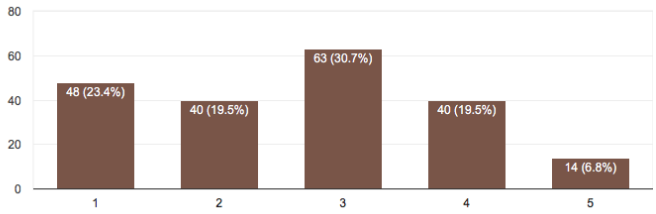

Fig. 5: Dependency on the Internet for cooking purposes.
1=very rare 5=very often.

E. How often do people have a problem in deciding what to cook and how many times have people realized that they are falling short of ingredients in a dish that they have wanted to cook?

These charts show us that more than 70% of the people face a difficulty in deciding what to cook for a meal. This application mainly targets this problem faced by a majority of users by sending them the recipes they can make according to their eating habits, preferences, and available ingredients. The solution involves intimating the user to buy ingredients beforehand so that they never miss out a chance of cooking their preferred meals.

F. Would people like to use a kitchen assistant which helps them manage ingredients and walks them through recipes?

These questions help learn the feasibility of an application such as the kitchen assistant.

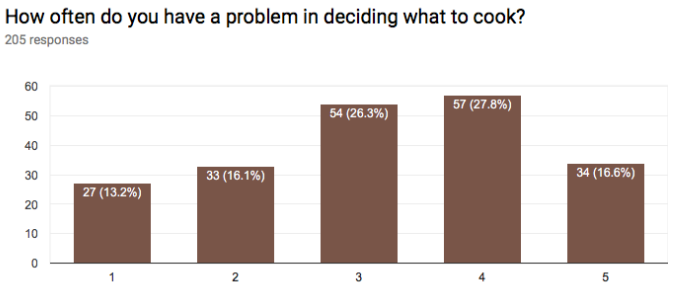

Fig. 6: Problems faced while cooking.
1=very rare 5=very often.

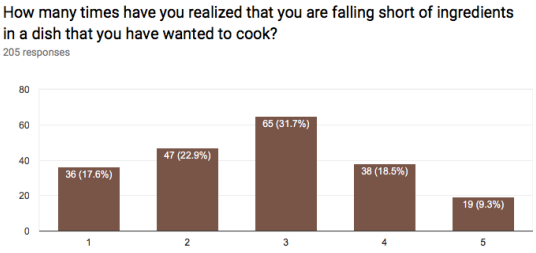

Fig. 7: Falling short on ingredients while cooking
1=very rare 5=very often.

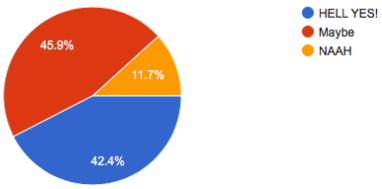

Fig. 8: Will users like to use a kitchen assistant.

G. What kind of an application do people think a kitchen assistant should be?

This question was asked to gather a public opinion on the platform they would like to use such an application.

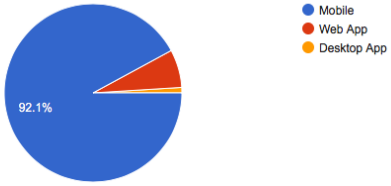

Fig. 9: What platform should the application be based on.

The results conform to the decision to make it a mobile application for ease of use.

IV. Software Architecture

A. Implementation choices

1) Mobile application v/s web application: There were two possible ways of implementing this application- web, and mobile. We have moved ahead with designing a mobile application as we feel that using a mobile in a kitchen is much more convenient and efficient than using a laptop or a desktop computer. It is possible to use a web application on a mobile phone but that certainly does not provide the best User Experience. The user survey has confirmed that users will be more comfortable in using a mobile application and thus we have decided to stick that option.
2) Android Application: We have used the Android architecture to develop our application. Android phones hold more than half of the market share. Also, one of the biggest reasons for choosing Android over an iOS application was that Java is compatible with android development whereas iOS requires specific languages like objective C or swift. It is also much easier, quicker and cheaper to publish an application onto the Android Play Store than the Apple App store.
   Another alternative was to build a cross-platform application using Xamarin. However, Xamarin is recommended for experienced mobile application developers. Android Studio is simple and easy to understand and has no license fee associated.
   Also, the online community states that the developer base for Xamarin is far smaller as compared to Android Studio. Thus, any errors lead to high wastage of time in Xamarin, whereas most technical difficulties while using Android Studio are resolved faster due to the availability of better support infrastructure.
3) MVC: For any application, the architecture is as important as the quality of code. Model view controller is one of the most popular architecture for applications. We have designed the server which runs a Node.js application that uses an MVC architecture using the Express.js framework and utilizing a Sequelize plugin for integration with the Database.
   The choice of Node.js was inspired by its capability to handle a large number of simultaneous connections with high throughput, which equates to high scalability. Node.js operates on a single-thread, using non-blocking I/O calls, allowing it to support tens of thousands of concurrent connections (held in the event loop).

4) Google Cloud Vision API: Google Cloud Vision API enables developers to understand the content of an image by encapsulating powerful machine learning models in an easy to use REST API. Optical Character Recognition (OCR) enables detects text within images, along with automatic language identification. Vision API supports a broad set of languages[14]. We tested OCR API on 20 different receipts from 2 different stores.The receipts were tested in varying levels of deterioration, as is expected from grocery receipts. Nevertheless, these APIs gave outputs at acceptable levels of accuracy for all such receipts. Some of the lighter texts (not bold enough) were misread by the API (the letter @ was most misread as 1), but that these did not extensively affect the final outcome of the process to extract the items along with the quantity from the grocery receipts as a loose string matching with the list of possible ingredients increases the accuracy of the OCR extraction of the grocery items.

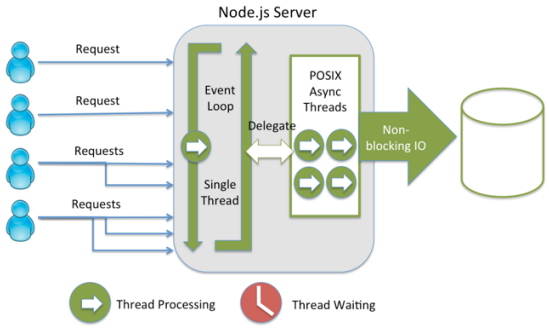

Fig. 10: Server Architecture.

B. Features

We have developed an application that works as a kitchen assistant and helps the users manage their ingredients, kitchen supplies and feeds them recipes based on their inventory. The following are the 2 main features that we have incorporated into the application

1) Inventory Management: There are three ways to keep the inventory updated:
   A. Manually add/delete/edit items and quantity
   B. Using the phones camera to take a scan the paper receipt and convert them to electronic inventory.
   C. Deleting ingredients automatically based on item cooked.

   Options B and C enable quick and convenient inventory management.
   We have used image analysis and character recognition to convert the receipt users upload of their grocery purchases through the application. We have utilized Googles Vision API (Image Content analysis) to extract the food items and their quantity from the receipt. The receipt obtained after purchasing groceries manually is a piece of paper. Even when groceries are bought online, the receipt is obtained as the electronic version of the text. We have a system to generate content for the inventory from all such formats. Googles Vision API (Image Content Analysis) was our choice for conversion of hard copies. When the user chooses option (b), we send the electronic version of the receipt as an image/pdf to Googles Vision API to convert the receipt to text

form which is then used to build up the inventory. Since receipts are computer generated (and not handwritten), these APIs give accurate results. We have stored keywords (food items and quantity) on our end and perform a loose string match on the results returned by the API to compare and update the quantities of ingredients.
The user is then navigated to an editable list of groceries extracted from the receipt. Once the user is satisfied, the user clicks on submit and the inventory is saved successfully on the server. The user can now log in using the same credentials on any android device to view and edit the saved inventory.
The inventory is also updated when the user cooks a particular suggested item. The user only has to press a button for the same, and the ingredients required to cook the item are deducted from the inventory appropriately.

2) Recipe Recommendation: Now that we have the users inventory and a large database of recipes that could be sent to the user, we will filter all the recipes that the users can cook.
The application queries the database and searches for recipes using the users current inventory. If all the required ingredients are present in the users inventory, the item is shown to the user along with the recipe including instructions to cook the recipe. The application does not show the items that the user cannot cook, and thus avoids any confusion.

C. Database Schema

Following is the database schema of our application. It displays all our various models and how they are connected with each other.

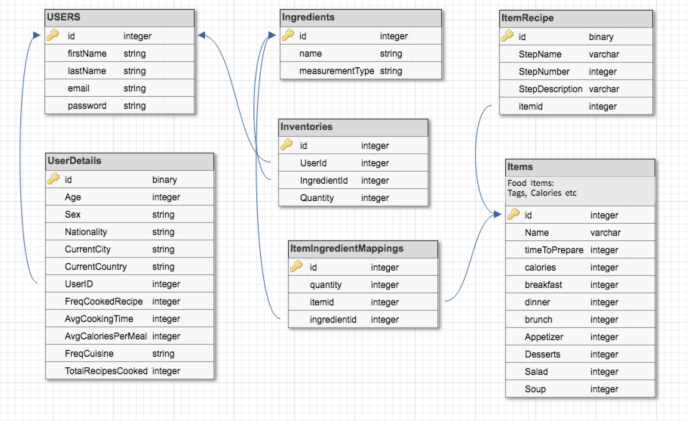

Fig. 11: Database Schema.

D. Object Relational Mapping

As our objects have many 1:1, 1:m, m:n mappings, We decided to go for an Object Relational Mapper, Sequelize, to reduce Object relational impedance mismatch. It allowed us to directly access object attributes and expose apis to perform CRUD operation. Sequelize is a promise-based ORM for Node.js v4 and up. It supports the dialects PostgreSQL, MySQL, SQLite and MSSQL and features solid transaction support, relations, read replication, migrations, hooks, scopes and more.[12]

E. Server Models and Functionality

1) **User**
   - Sign-Up
   - Login
   - Authentication
   - Session Management
   - Delete User
   - Update User Details
2) **Ingredient**
   - Add/Delete/Edit Ingredient
   - Get Measurement Type for Ingredient
   - Get Ingredient Id from Name
3) **Inventory**
   - Add Ingredient and quantity for a user in inventory table when a user add items to his inventory.
   - Update inventory, when a meal is prepared based on recipe Quantity
   - Fetch Inventory of a user from database.
4) **UserDetails**
   - Add/Edit/Delete UserDetails for a User
   - Update AvgCookingTime, TotalRecipesCooked when a user cooks a meal.
   - Update AvgCaloriesPerMeal, FreqCuisine based on User cooking history.
   - Fetch UserDetails for a given user for recommendation system
5) **Item**
   - Add/Edit/Delete Items
   - Retrieve Items a user can prepare based on available Ingredients in User inventory.
   - Update AvgCaloriesPerMeal, FreqCuisine based on User cooking history.
   - Update User Inventory when this Item is cooked.
6) **ItemRecipe**
   - add/update/delete Recipe details
   - Fetch recipe details ie. all steps in order with description for a given Itemid.
7) **ItemIngredientMapping**
   - Mapping of Ingredients with Quantity required by a recipe(item)

F. Client Server Interaction

To enforce statelessness on the server, we are using RESTful Apis for client server communication. This helps in maintaining a good Separation of Concerns between the client and server, making the application easy to scale. REST stands for Representational State Transfer. It relies on a stateless, client-server, cacheable communications protocol - like the HTTP protocol. We are exposing following REST Apis to the client:

Method	API	Callback
POST	/api/users	Create new User
POST	/api/inventory/userid	Create one inventory record for User
GET	/api/inventory/userid	Retrieve inventory of a particular user
PUT	/api/inventory/userid	Latest Inventory is sent as JSON object in the body of the API, Callback Updates inventory table for the user
DELETE	/api/inventory/userid	Deletes items from inventory for the user
GET	/api/ingredients	Retrieves list of ingredients in the database
GET	/api/items	Retrieves list of Recipes in the database
GET	/api/items/recipesuggestion/userid	Searches list of recipes a user can prepare based on available ingredient in the inventory
GET	/api/items/recipetags/ userid	Recommends Recipes based on selected Tags, Calorie Requirement and Time to Prepare. All parameters are sent as query parameter
GET	/api/items/completecooking/userid	Updates Quantity of the ingredients used for preparing a recipe; recipe id is sent as query parameter

Fig. 12: Restful API's.

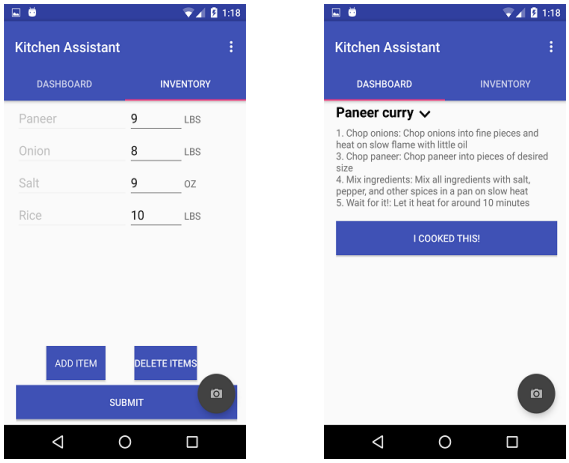

Fig. 13: Inventory Dashboard and Recipe Suggestion

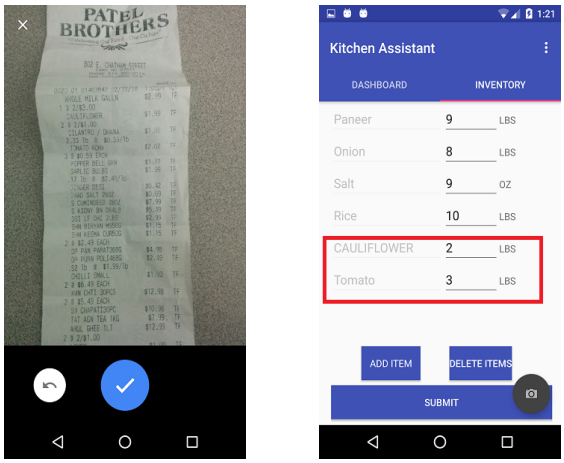

Fig. 14: OCR

G. User interface and application flow

The following describes the basic user interface and the application flow:

1) User login: The user has the option to log in manually with email id or via Facebook. Login via Facebook will eliminate the need to verify the email id. If the user chooses to proceed with facebook login then the user is asked to enter the credentials only the first time. The next time, the user will only be asked to continue with already entered credentials. This provides a better user experience
2) Inventory: The user can view the inventory, as shown in Fig 15, that is stored in the database at the server-end. It shows the ingredients and the associated quantity. The user can edit the quantity, delete, and add ingredients manually to the inventory.
3) The user receives item suggestions based on the inventory, as shown in Fig 15. Only items that can be cooked using the users inventory shows up on this page. This avoids the need for the user to filter out the items that cannot be cooked presently.
4) Since manually editing the inventory is cumbersome, there is an option to upload the receipt of the purchased grocery. We then use Googles Image Content Analysis API to extract the text from the receipt and run our algorithms to extract the ingredients with their quantity from the paper receipt. Currently, we have a limited set of ingredients on our side, so all ingredients are not extracted. Figure 16 shows the screenshot of the receipt that is being scanned to be sent to Google Vision API and the second screenshot highlighting the items that were added to the user's inventory based on the text extracted from the receipt.
5) Figure 16 highlights the two new ingredients along with their accurate quantity that was extracted successfully from the receipt. The user is then asked to review the modified inventory before submitting. Once the user clicks on submit button at the bottom of the screen, a notification pops up notifying whether the inventory was updated successfully. Figure 17 highlights the message that is displayed when the database on the server-side has been successfully updated with new ingredients extracted from the bill.
6) After addition of the two new ingredients extracted by OCR to the inventory, the items that the user can now cook have been updated, and these items are displayed on the dashboard as highlighted in Figure 17.
7) Deductions from inventory: Once the user cooks a suggested item, there is no need for the user to update (delete) ingredients manually. The user can simply click a button to notify the application that a particular item has been cooked, and the application takes care of updating the quantities of ingredients automatically. Figure 18 highlights deductions from inventory.

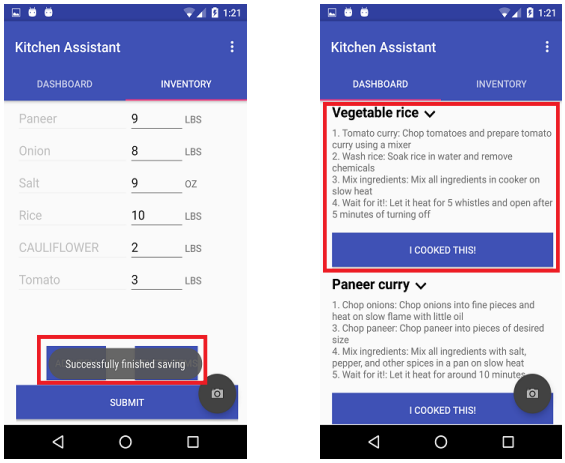

Fig. 15: Recipe Recommendation after update

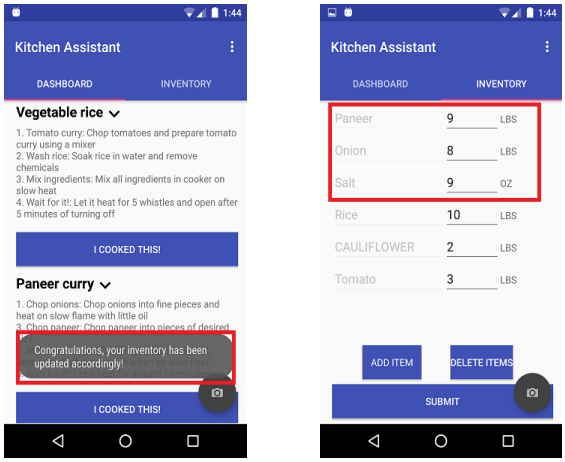

Fig. 16: New Recommendations after update

V. Development Plan

A. RoadMap and Estimation

Figure 17 describes our estimated timelines and RoadMap

Milestone	Description	Date planned
Project Scope Discussion	Understanding the scope and value of project	18th Jan 2018
Initial Survey	User Opinion for our Idea	24th Jan 2018
Complete Requirement gathering	Define major deliverables	28st Jab 2018
Complete Application Design	Define and prioritize preliminary list of functions.	5th Feb 2018
Complete Application Coding	DB schema, Android Application, Async Request Handling Server	19th Feb 2018
Complete Application Testing And Debugging	Unit and Integration testing	26th Feb 2018
Evaluation	Complete Application Evaluation	1st Mar 2018
Report and Supporting Documentation	Final Report and README	1st Mar 2018

Fig. 17: RoadMap Planned

B. Data Collection for Recommendation

There were various options for us to collect data to be used in building the database for recipe recommendation. The first approach was to use the dataset available on Kaggle[13]. However, this dataset directly gave the items, rather than the group of the item to which it belonged. Moreover, we did not have the accurate recipe for these items. Also, it has comfort food such as chocolates, ice-cream, etc. which are basically just dessert items, and do not solve the problem of unhealthy food.

We thus decided to gather our own data for recipe suggestions that suit our needs. We used the following survey for the same: Data Collection Survey

The result of this dataset will help us build intelligent classification models and suggest healthy food items that can be cooked at home for a particular cuisine of predicted preference.

C. Design decisions

1) Azure vs AWS (and why we moved to AWS from Azure): AWS Relational Database free tier provides a storage quota of 20 gigabytes, whereas the Azure Database service at the time of evaluation was only providing few MB of free storage, though this has been increased to 250GB at the time of writing this report. However, for an Azure MySQL database service, the free quota is still somewhere in the range of 5GB. And the MySQL azure database is still in the preview version. On the other hand, AWS MySQL database is a very mature service in comparison to Azure SQL server service. Azure Database instance is still paid a for instance, though within the free credits in sign up, this will run out after few initial weeks.
2) MongoDB vs MYsql: MongoDB is a non-structured database whereas MySQL is a structured relational database. For an application like ours, we decided to go with a structured database as all our tables would be easily parsed when structured. Each recipe uses a lot of common ingredients. Every user inventory would also contain lots of ingredients which be similar and overlapping between many users. This would create a lot of wasteful and redundant data in our database if the choice was an unstructured database. Thus we decided to go ahead with a structured database keeping future storage space requirements in mind beyond the project presentation.

D. Testing Considerations

1) Unit Test:
   a) User Sign up : User should be sent to the sign up page, and should be able to fill in personal details and sign up.
   b) User sign in : User should be able to enter a valid and registered email-id and password, hit the login button and should be transferred to the dashboard.

c) Add Item in inventory : User should be able to manually add an item into his inventory.
d) Delete Item in inventory : User should be able to manually delete an item from his inventory.
e) Update item in inventory : User should be able to manually update the quantity of an item in his inventory.
f) Add item using camera : User should be able to scan a bill and automatically update his inventory.
g) Selecting recipe to cook : User should be able to view all recipes possible for him to cook according to his inventory.
h) Finish cooking recipe : After the user hits 'I cooked this' button, the quantity of the ingredients used should get updated automatically.

2) Integration Test:
a) Inventory retrieval API call from android application to node.js server must be completed successfully.
b) Inventory update API call from android application to node.js server must be completed and data must be reflected in the MySQL database.
c) Inventory delete API call from android application to node.js server must be completed successfully and data must be deleted from the corresponding tables on the MySQL database on the server.
d) Authorization of the user session token must be completed with calls to facebook API to validate the users facebook login.
e) Android application must use the device camera to click photo of the paper receipts that has to be sent to the Google Vision API that must return the text form of the paper receipt.

E. Deployment

Fig. 18: Application Deployment

1) Server Deployment Architecture: The server is deployed on Amazon Web Services Platform running Linux, with platform support for Node.js. The server involves one or more instance of Amazon EC2 instances, as determined by the Load balancer as part of Elastic Beanstalk framework. The server hosts the node.js codebase which provides integration with the Amazon RDS instance running MySQL engine. The free tier of AWS provides 20 Gigabytes of free storage with 750 hours of RDS instance running time. The integration with the database as mentioned previously utilized Sequelize plugin. Amazon Deployment was handled using elastic beanstalk CLI. Installed in the development environment the EB CLI allows fast deployment of the code base to the AWS EC2 instance. The EB environment created to support the Node.js application needs to be configured with npm command npm run start:dev. If no npm command is specified the elastic beanstalk environment uses app.js, then server.js, then "npm start" in that order.
2) Client Deployment: The client can be deployed on any android device using distributable APK, on any devices running Android API Version 21 and above. The client requires an active Internet connection via wifi or mobile network and minimum of 50 MB storage space.

Third Party Platforms: The following third-party platforms host APIs that are being used by either the client package or the server.
A. Google cloud platform: This google vision API is supported by the Google cloud platform which is used by the client to actively scan the grocery bills to update the inventory.
B. Facebook APIs: The facebook login service along with facebook graph API allows authorization and authentication using facebook credentials and also provides information regarding the users background, which can be utilized to retrieve details that can be valuable in recommending recipes from all available recipes.

F. Software Development Strategy

1) Agile methodology: We started with an Iterative model in the first 2 weeks, and switched to Agile once we had a clear picture of implementation details.
2) Test-driven development: We started with TDD, but in later parts, found it hard to keep up to it. We then developed first before writing tests.
3) Continuous Integration Continuous Deployment: Although manually initiated, we tried to follow CI-CD so that we do not face deployment issues later. Elastic Beanstalk CLI was installed and integrated into the development environment to allow smooth and effortless continuous deployment.
4) Fail fast: We discussed the idea of the application with a few users early in the development phase to gain feedback and decide features to develop accordingly.

G. Open Bugs

• When the quantity of ingredients becomes 0, the ingredient should be removed from the users inventory.
• OCR should recognize the ingredient already present in the inventory and update the quantity on client side rather than adding a new entry.
• One user was unable to log in to the application. This happens for a subset of users that have the Facebook application already installed because of issue with access token.

VI. Evaluation Analysis

Once the first version of our mobile application was ready, we rolled it out to a few users to gain feedback on the application and understand if they were satisfied with the interface and features and also know what features they thought were missing.

We evaluated our Software based on following Metrics:
1. Simplicity
2. Utility
3. Accuracy

The questions asked along with the results were as follows:

A. Were you able to login into the application ?

Users are unable to do anything unless they log in to the application using either their email address or via Facebook. Hence, this was an important question to check if users were able to use our application at all.

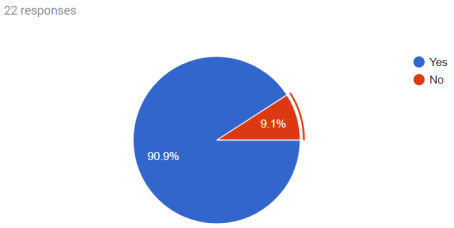

Fig. 19: Login Success

As we see in Figure 20, majority of the users were able to log in to our application. We uncovered that 1 individual was unable to log in and proceed further. On analysis, we found out that this happens only when the individual already has the Facebook application installed and a new access token is not requested for logging in to our application. This is a serious bug and has been mentioned in the Open Bugs section.

B. How would you like to rate the GUI of the mobile application?

The user interface is the first impression of the application to the outside world, and it is extremely important to have a simple, easy-to-understand, and clean interface design.

As seen in figure 21, the majority of the users were satisfied with the user interface of our application. However, some users have rated the GUI as 3 out of 5, which can be inferred as not up to the mark. Further improvements to the interface

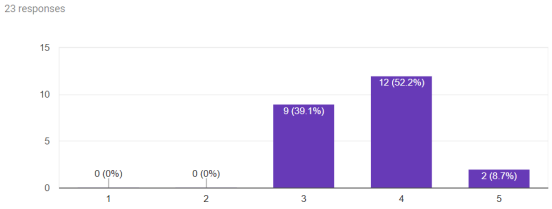

Fig. 20: User Friendly Interface

are certainly possible. Users wanted more responsive buttons and hints as to what the application and certain components of the application are capable of doing. There are instances where users expected more information on the progress of a background action to be communicated on the front end. This can be achieved with additional progress bars and message boxes.

C. How easily were you able to navigate through the application?

Users tend to use the application lesser if they are unable to navigate easily i.e. without a lot of explanations and suggestions. We wanted to understand if users had any difficulty in navigating through our application and understanding where to click for what functionality.

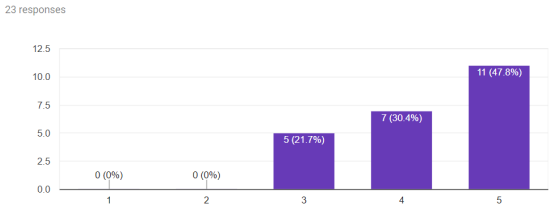

Fig. 21: Convenient Navigation

As seen in figure 22, users had minimum to no issues in navigating through the application. Our application uses the Fragment tabs used in a lot of popular applications for navigation and the users are accustomed to this navigation style.

D. Did you like the idea of using OCR to track your groceries / ingredients?

We believe one of the cool features of our application is using Optical Character Recognition for extracting grocery and managing inventory by uploading a receipt rather than manually entering each grocery item bought. This seemed cool to us while developing the application but the time investment to support various receipts was a huge requirement, so we wanted to ensure that it was worth it.

From the response shown in figure 23, we affirmed that this was indeed a feature much liked by users as it reduces manual efforts and is definitely worth something investing time on.

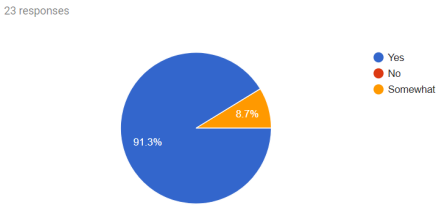

Fig. 22: Inventory Extraction

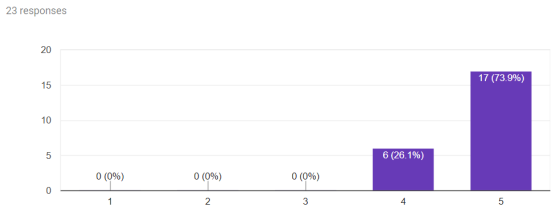

Fig. 24: Feature Overview

E. How accurate is the OCR ingredient tracking functionality according to you?

It was difficult to extract grocery items along with the quantities of various types of receipts. Note that for this functionality, users were made aware that only 2 types of receipts of stores commonly visited by students were accepted. Users were also made aware that a subset of the ingredients was not a part of our data and would not be recognized.Any upgrades to this situation require significant efforts in enhancing entity identification and classification requiring more practical data or business partnership with grocery stores or software company specializing in billing software. The responses to this question were obtained with the above two facts already in users mind.

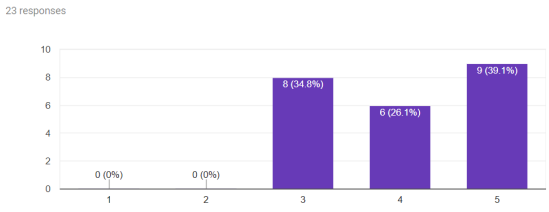

Fig. 23: Inventory Extraction Accuracy

The responses to this (as shown in Figure 24) were varied as expected. There is certainly scope to improve the algorithm for accurate extraction, but balancing the trade-off was essential. We decided to go ahead with some manual intervention by users to check the updated inventory after analyzing image contents and before saving it, rather than investing a large amount of time trying to support all ingredients and all receipt formats.

F. How do you like the idea of keeping track of groceries and suggesting recipes accordingly?

This question gave a gist of the two main functionalities of our application and asked if the user liked our application's implementation of these functionalities.

The results, as shown in figure 25, indicate that users were satisfied and interested in the application that solved the problem of maintaining inventory and getting recipe suggestions based on the inventory.

G. What other features do you think can/should be added to this application?

We wanted to have an open-ended question to understand if users had any particularly useful features in mind that were missing from our application to determine the future scope of the application.

The responses by users were extremely critical in analyzing and evaluating the work needed in future on the application. Some useful features suggested by users were:

a. Hints for using the application and its components.
b. Group-sharing feature.
c. Recipe recommendations based on nationality, meat preferences, etc.
d. Recipe creation feature.
e. Automatic creation of shopping lists based on missing ingredients.

Most of the suggested features have been included in the future scope section of the report.

H. What are some areas where we can work to improve the application?

We wanted to understand more non-functional requirements that users were expecting. Some of the important suggestions were as follows:

a. More extensive data with better recipes.
b. Better user-interface with some explanations and hints.
c. Better responsiveness.
d. Reduced response time.
e. Support for various receipt formats.

Note: A couple of points for G and H were shuffled in the report as compared to the user survey to maintain the distinction between functional and non-functional requirements.

I. How useful do you think the application is in solving its purpose?

No matter how easy to use, efficient, or beautifully designed an application is, the utility of the application is the prime reason for increased usage.

The responses, as shown in figure 26, indicate that our application has high utility and suggests that with good enough features and design, our application will be used by a large number of users. Note: We would like to state that the evaluation was majorly done by users staying outside their home country with friends and peers who have the need to cook their own food. We expect the utility to go down if users from various groups are surveyed.

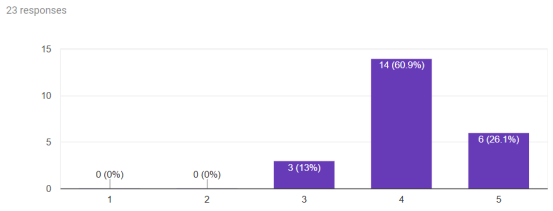

Fig. 25: Application Utility

VII. Comparison with similar softwares

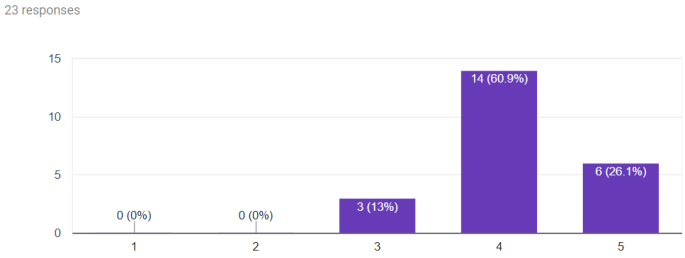

Fig. 26: Application Utility

VIII. Future Scope

A. Ability to delete ingredients after cooking an item for multiple people

Currently, our application supports managing the inventory for a single user by deleting the quantities of ingredients required for 1 individual or sometimes varying servings according to the recipes in our database. This should be customizable for any number of users. That way, one person cooking food for X people can manage his inventory by providing input X. Right now, the user will have to click button X times if he or she wants to auto-delete ingredients from his or her inventory.

B. Recommend dishes based on user preferences and features

The users receive suggestions of all the dishes they can cook from their inventory in no particular order. The order of these suggestions should be modified according to their features, preferences, and taste. Factors such as health consciousness, nationality, age, etc. should also be taken into consideration. It is a well-known fact that people from different regions have different staple foods. A person trying to maintain a healthy diet should not be suggested junk food. While some of these suggestions will be taken care of by the ingredients the user buys, since we do not suggest an item if the user does not buy the ingredient required to cook that item, it can be further enhanced by ordering the suggestions based on classification and clustering techniques by analyzing the data. It could be augmented with options to further filter the recipes displayed based on users choices.

C. Support bills from multiple sources

One of the cool features of our application is auto-extraction of ingredients (grocery) from a receipt. However, this was tested on a limited set of receipts, and the algorithms were adjusted for successful extraction of grocery items from receipts of particular stores. One major drawback is that each store has its own receipt format. Three options to overcome this are:

A. Customized algorithm:

1) Analyze the receipt heading and identify the store.
2) Extract the grocery items based on a customized algorithm for that particular store.

However, this solution is limited in nature by the fact that we cannot expect developers to keep customizing algorithms for every different receipt from all stores. At max, we can have adjustments for a few popular, daily stores.

B. Integrate with grocery stores: Since solution A is limited in nature, an integration with majority stores should be the end goal for the application. With this integration, we should have the database of ingredients and the measurement type from all these integrated stores. With this, we can create a more generalized algorithm to extract the inventory from all these stores. We would still need to customize the algorithms, but this customization would be easier by integrating our servers with the store data. This would be more flexible and extensible and can be used to grow the number of store receipts that our application supports

C. Sophisticated text analytics along with entity detection and classification on the text extracted from paper receipts using Google vision API and utilizing reinforced learning to adapt to various receipts that a user might frequently encounter. This can be CPU intensive and can affect performance by affecting response times.

D. Shared inventory management

Since multiple people stay in the same household, the grocery items are shared by among inmates. We should support a group feature that allows multiple people to be a part of a specific group and share the same inventory. The inventory should be refreshed for the entire group once a particular user in the group updates it. Also, there should be an option to add individual ingredients along with the group inventory. The group management feature from Splitwise mobile application can be analyzed and used in our application in similar ways.

E. Item suggestions with limited grocery requirement

Currently, we only show those items that the users are able to cook based on their inventory. However, it would be helpful for users if they know what items they can cook if they buy particular ingredients. Hence, our application should show the items that the user can cook by buying a few ingredients. Here, few is a relative term, and the decision to filter out the items should be taken with care. There can be a cap on the number of ingredients users do not have but the items should still be suggested. Additionally, the users should be able to distinguish between these two, i.e. items that they can cook

from their inventory and the ones that they cannot. A simple green tick and red cross would be visually appealing along with the missing ingredients mentioned with the red ones.

F. Auto-order

One important feature missing from our application is the ability to auto-order groceries based on the inventory for people who prefer to purchase their groceries online. In future, we would like to integrate our application with Instacart / ZINC to use their infrastructure to order groceries automatically for our users. Amazon business API's allows the creation of shopping carts on behalf of users which can later be used by the Users to review and check out suggested items. This also allows for monetization of this product.

G. Integration with smart digital assistants

Google Assistant and Amazon Alexa are some interfaces we would like to use in the future to make the application more convenient to use while cooking.

H. Recipe share

Currently, we have our own database of recipes, items, and ingredients. However. we understand that new recipes are discovered by people all over the world. We would like to give them a platform to share these recipes with all the users of our application.

IX. Conclusion

The research statistics and our survey results detailed here indicate that although various solutions have been proposed for easing the preparation of healthy home cooked meals, people of all ages are still looking for a better one with merged functionalities from different platforms. We present this by reversing the strategy proposed in other systems. Once the user purchases the daily groceries, we automate the entire process from recommending recipes that can be cooked from them. Our application will eliminate the need to remember the available groceries by creating and managing the inventory using his receipts. We shall also keep track of the users inventory and update it as the user buys or cooks with automated methods requiring minimal effort. This application can be really useful in helping people manage their groceries, prepare more home cooked meals, avoid eating outside and achieve better health.

References

[1] Amazon Unveils Futuristic Plan: Delivery by Drone". CBS News. 1 December 2013. Retrieved 6 May 2014.

[2] Soliah LL, Walter JM, Jones SA. *Benefits and barriers to healthful eating: what are the consequences of decreased food preparation ability?*, Am J Lifestyle Med. 2012;6:152158.

[3] Jabs J, Devine CM. *Time scarcity and food choices: an overview. Appetite.*, 2006;47:196204.

[4] R. Lappalainen, A. Saba, L. Holm, H. Mykkanen, M.J.Gibney, A. Moles. *Difficulties in trying to eat healthier: descriptive analysis of perceived barriers for healthy eating*, Eur J Clin Nutr, 51 (S2) (1997), pp. S36-S40

[5] Clausen A. Food CPI and Expenditures Briefing Room, Table 10. [Accessed 4-4-12];US Department of Agriculture, Economic Research Service, 2011

[6] Suzanne Higgs, Jason Thomas. *Social influences on eating*,Current Opinion in Behavioral Sciences,Volume 9, 2016, Pages 1-6,ISSN 2352-1546, https://doi.org/10.1016/j.cobeha.2015.10.005. (http://www.sciencedirect.com/science/article/pii/S235215461500131X)

[7] Poti JM, Popkin BM. Trends in Energy Intake among US Children by Eating Location and Food Source, 19772006. J Am Diet Assoc. 2011;111:11561164. doi: 10.1016/j.jada.2011.05.007.

[8] Roberto A. Ferdman. *Researchers have found a striking new side effect from eating fast food*, https://www.washingtonpost.com/news/wonk/wp/2016/04/15/researchers-have-found-an-alarming-new-side-effect-from-eating-fast-food/

[9] M. Pazzani, *A Framework for Collaborative Content-Based and Demographic Filtering*, Artificial Intelligence Rev., pp. 393-408, Dec. 1999.

[10] Xiaoyuan Su and Taghi M. Khoshgoftaar, *A Survey of Collaborative Filtering Techniques*, Advances in Artificial Intelligence, vol. 2009, Article ID 421425, 19 pages, 2009. doi:10.1155/2009/421425

[11] Jagithyala, Anirudh, *Recommending recipes based on ingredients and user reviews*, http://hdl.handle.net/2097/18154

[12] Sequelize Homepage

[13] *Food choices - Kaggle*, kaggle data set

[14] *Vision API - Image Content Analysis - Google Cloud Platform*, https://cloud.google.com/vision/

X. Review Chits

- PGR
- IWL
- BJJ
- XQU
- ZPX
- VFT
- QKB
- UHB
- FEP
- MHH
- HLL
- BLU
- JWL
- OFB
- UCO
- HHZ
- BYL
- YUZ
- PZW
- XBU
- NBZ
- MQA
- COQ
- CAT
- RHQ