COOKBETTER ver 2 : A Bot for Personalized Recipe Recommendation

[Report 2a - Team ’O’ enhancing Team ’E’s work]

Shrikanth N C
NC State University
CS department, NC 27606
snaraya7@ncsu.edu

Karthik M S
NC State University
ECE department, NC 27606
kmedidi@ncsu.edu

Kashyap S
NC State University
ECE department, NC 27606
ksivasu@ncsu.edu

Charan Ram V C
NC State University
ECE department, NC 27606
cvellai@ncsu.edu

Ragavi Kalaignani
NC State University
CS department, NC 27606
rkalaig@ncsu.edu

ABSTRACT

The goal of this project is to improve *Cookbetter bot* by introducing new features and bolstering existing features to provide better usability and applicability of the system. Firstly, we discern existing problems of the current version by analyzing the evaluation report to identify the points where the application requires betterment. Secondly, we understand the scope of each solution proposed and determine use cases that should be built to achieve the aforementioned goal. Thirdly, the report discusses all the use cases in detail which we believe would enhance the functional flexibility of the system, thereby increasing the overall efficiency and ultimately offering high-grade user experience. Lastly, the report outlines the development process to be adopted with the estimated timeline for completion and presents an evaluation plan to assess the project.

Keywords

Recommendation System, Slackbot, Image, Text processing, Visualization, Feedback

1. WHAT WAS DONE BEFORE?

1.1 Overview

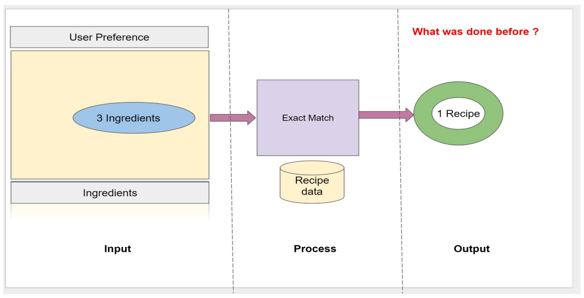


Figure 1: What was done before?

The existing system, Cook Better, is a Slack chat bot for personalized recipe recommendation. It allows the user to configure their dietary and health restrictions in the system. Users can search for recipes based on ingredients and few other criteria. Recipe data from the website Epicurious is used. Based on the ingredients and criteria chosen at the time of searching and also based on each user’s preconfigured dietary and health restrictions, recipes are selected from the database. A list of recipe links is displayed to the user, and upon clicking these links the user is redirected to the Epicurious website to get the exact instructions for cooking. The chat bot currently supports four slash commands - /personalize, /searchrecipes, /surpriseme and /cookbetterhelp.

1.2 Evaluation

Evaluation of the existing system has been carried out by allowing users to interact with the chat bot in a Slack workspace. After using the chat bot, users were asked to fill an online questionnaire. The questions were chosen to validate the following four metrics:

- Reliability
- Scalability
- Cooperativity
- User Satisfaction

1.3 Results

The results of the evaluation that was conducted clearly indicated that the evaluators found the chat bot to be both helpful and reliable. The users also indicated that they would prefer using this chat bot over traditional websites for recipe recommendation.

1.4 Problems

The evaluation results revealed three major problems with the system. Most of the suggestions that were given to change the existing system were broadly related to these problems:

1. **Input**: There were multiple suggestions to use text boxes, checkboxes or radio buttons for selecting ingredients, rather than using drop-down select boxes. Many suggestions mentioned that a more intuitive and interactive user interface would be helpful.

2. **Output**: Some evaluators had these problems with the output - redirecting to a website instead of giving more information within the bot, ending the search abruptly after displaying the results without having an option to continue the search or get more details.

3. **Recommendation**: Some evaluators suggested the incorporation of machine learning algorithms to improve the quality of results that are displayed to the user.

1.5 Learnings

Upon studying the existing system and examining the evaluation results, the following are our major takeaways:

- We need to make the method of inputting ingredients more intuitive and easy-to-use.
- We need to enhance the output that is displayed to the user and provide more information.
- We need to develop an intelligent recommendation module to search for recipes.

2. WHAT ARE WE DOING?

In a nutshell, we are designing the system to accept inputs from various channels. And, building a recommender system to stretch the fruitfulness of the system.

2.1 New Requirements

Inferring from the results elucidated in the previous section and in close discussion with our mentors we identified 5 use cases that would improve cookbetter to a large extent.

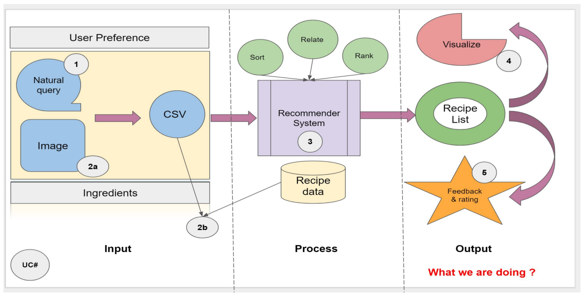


Figure 2: What we are doing

2.2 Use Case 1: Natural Query

Presently, cookbetter takes fixed 3 ingredients in a user interface consisting of drop downs. We wish to discard this and replace this with an enhancement that could take natural queries as inputs to respond with recipes or ingredients as shown Figure 2 & 3.

- Sample Input 1 : I have apple, milk, yogurt and sugar what can I make?
- Sample Output 1 : Apple smoothie, well you also need honey and almonds.
- Sample Input 2 : Can you list ingredients for butter chicken?
- Sample Output 2 : Butter chicken Ingredients list

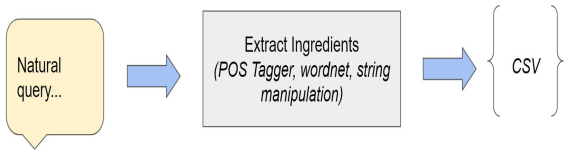

Natural query...
Extract Ingredients (POS Tagger, wordnet, string manipulation)
CSV

Figure 3: Natural Query as Input

2.3 Use Case 2: Image & CSV

In this use case (2a) user captures images of items say from a refrigerator and uploads them to cookbetter. We translate those images and direct them to the recommender system (see Figure 2 & 4) for it to recommend recipes.

We plan on using the Clarifai API [4] which offers image and video recognition as a service. We compared 3 different image analysis APIs to find the one most suitable for our project: Clarifai, Cloud Vision API by Google [5] and Amazon Rekognition [1]. While we only plan on performing image recognition in our current project, we wanted to keep the option of video recognition open for future enhancements. Hence, we decided to choose an API that also supported video analysis, and ruled out Cloud Vision API by Google. We chose Clarifai API because we found that it was flexible enough to increase our rate limits to 30 requests per second.

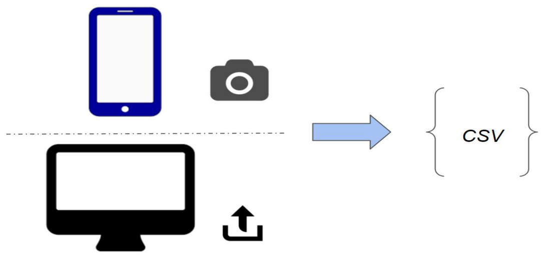


Figure 4: Image & CSV inputs

- Sample Input 1 : Pictures of apple, milk, and yogurt (captured from a smartphone).
- Sample Output 1 : Apple smoothie, well you also need honey and almonds.

Additionally, use case (2b) is for users who are crisp to the point, we intend to accept the widely recognized comma separated value format as an input to recommend recipes. For example,

- Sample Input 1 : apple, milk, yogurt
- Sample Output 1 : Apple smoothie, well you also need honey and almonds.

2.4 Use Case 3: Recommender

The recommender system is the central piece in our enhancement in this phase. We envision the same to take a structured input (csv) as shown in Figure 2 to recommend a list of matching recipes. In scenarios, if exact matches are not found, we intend to replace with the nearest neighbor (recipe). We also propose to show the missing ingredients in such cases. Interestingly, this would demand us to a build a sort, relate and rank the recipe data-set across various dimensions for each csv request.

- Sample Input 1 : ingredient1, ingredient2, ... ingredientN
- Sample Output 1 : recipe1,recipe2,...recipeN (as applicable with other info such as missing ingredients)

2.5 Use Case 4: Visualization

In this use case, we intend to present the recommended recipes intuitively with images and relationships that user could quickly infer about the recipe and its peripheral data. Rather than a flat mundane list of items, now the user can easily make judgments based on this rich visualization. The visualization is intended to exhibit recipe relationships and dept some stats related to user preferences.

For example, see Figure 5 below.

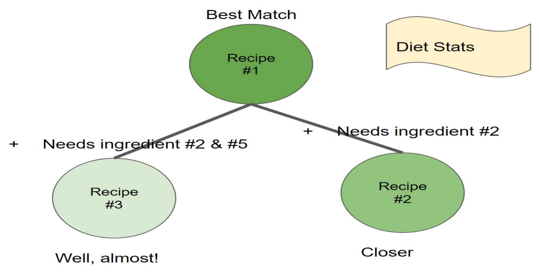


Figure 5: Recipes Presentation

2.6 Use Case 5: Feedback & Rating

We intend to accept a simple rating and allow users to add comments on the recommended recipes refer Figure 6. This means that the users are also presented with recent comments for recipes recommended to them. Now the recommender system takes the user's rating into consideration while sorting recipes to recommend. In other words, the recommender system is not deterministic, rather optimally tuned to a specific user.

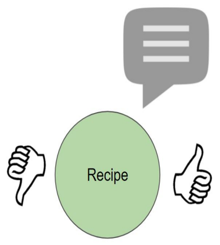


Figure 6: Rating & Feedback

2.7 Architecture

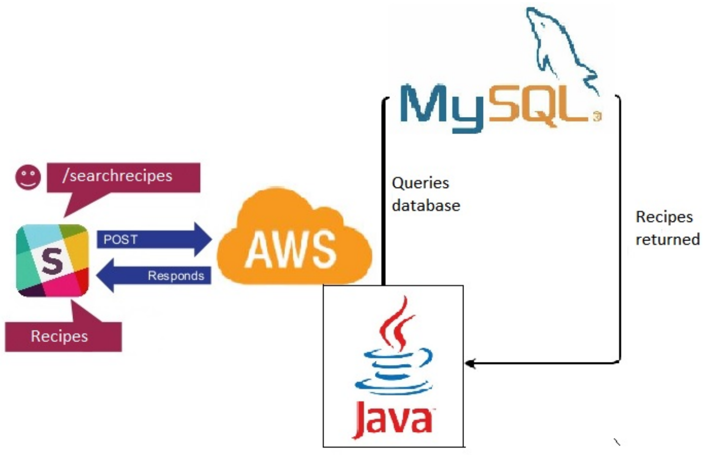


Figure 7: Architecture

We are enhancing the existing system without affecting the current architecture. We will be adding our enhancements to the REST service running on the server. The architectural design is shown in Figure 7.

When the user interacts with the chat bot, Slack sends a POST request to our server running on AWS [2]. The Spring application [8] queries the MySQL database [7] and constructs a JSON response. The response is sent to Slack and displayed to the user.

3. WHAT ARE WE TRYING TO DO BETTER?

We are trying to solve the problems with the existing system presented in Section 1.4 with the use cases proposed in Section 2 in the following way:

1. **Input**: Use cases 1 & 2 (Natural Query, Image & CSV) address the problems related to lack of intuitiveness of input methods. The usability of the system is greatly affected when drop-down selectors are used with over 500 ingredients displayed as options. Using natural language querying and uploading images of ingredients would provide better usability.
2. **Output**: Use cases 4 & 5 address the problems related to the output of the system, by providing rich visualizations of the results and letting the user provide and view feedback for each recipe.
3. **Recommendation**: We plan on incorporating a more intelligent recommendation module, proposed in use case 3, to increase the quality of the results presented to the user.

3.1 New Survey Results

We studied the feasibility of our proposed enhancements with an online questionnaire with 4 questions, which was answered by around 23 participants.

Section 1: Intuitiveness of Input Methods
Question 1: To search for recipes based on ingredients, which of these methods of inputting ingredients seem most intuitive/easy-to-use to you?

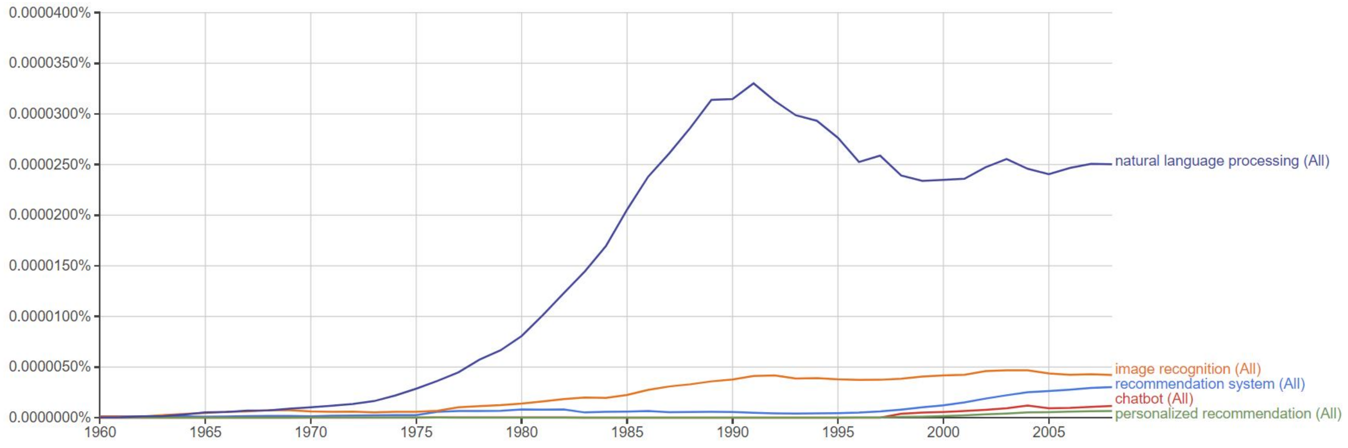


Figure 8: Results from Google Ngram Viewer

This question was aimed to study the proposed use cases 1 & 2. 87% of the participants preferred capturing images of the ingredients. The results shown in Figure 9 show that all 3 of our proposed input methods received higher votes than the current method of selecting ingredients from a drop-down select box.

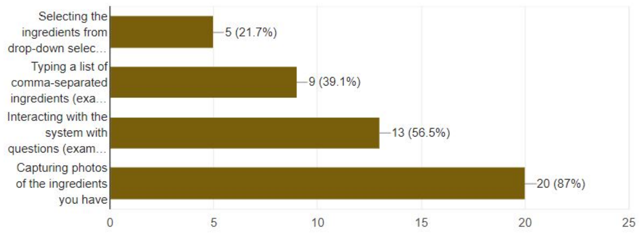


Figure 9: Intuitiveness of Input Methods

Section 2: Usefulness/Necessity of Features

Description: How useful/necessary do you think the following features will be for a recipe recommendation chat bot? (Scale: 1 - Would not use/unnecessary, 5 - Very useful/necessary)

Question 2: Feature: Feedback for Recipes
Question 3: Feature: Visualization
Question 4: Feature: Recommender System for Ranking Recipes

Questions 2-4 were intended to determine the usefulness and necessity of the features proposed in use cases 5, 4 and 3, and the results are depicted in figures 10, 11 and 12 respectively. We can see that on a scale of 1 to 5 (with 5 being very useful/necessary), all 3 features received a rating of 3 and above.

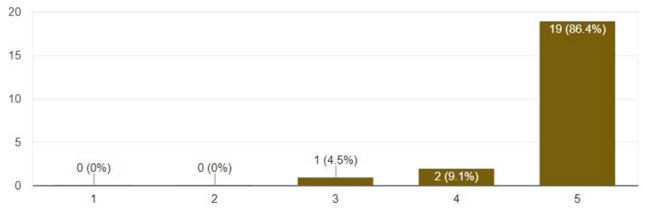


Figure 10: Feature: Feedback for Recipes

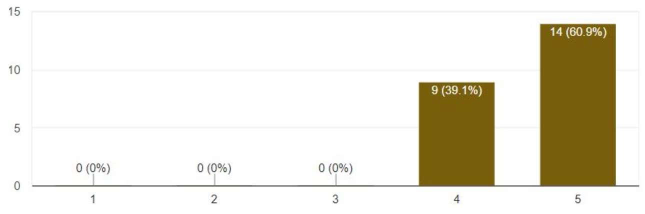


Figure 11: Feature: Visualization

3.2 New Literature Study

In 2011, Ricci et al [9] studied various techniques for recommendation systems such as collaborative filtering, content-based filtering, hybrid recommender systems, knowledge-based, etc. We are adapting the knowledge-based recommendation technique discussed in this paper by recommending recipes based on specific domain knowledge about how each recipe matches user's personalization preferences.

We extended the existing literature study done with Google NGram Viewer to study the feasibility of our enhancements. Google NGram Viewer is used to chart the frequencies of the search keywords mentioned in the vast amount of books available in the Google Books library.

The following keywords are used in the existing study: recommendation system, personalized recommendation, and chatbot. Based on our use cases 1 and 2, we included the following keywords in our study: natural language processing and image recognition. The frequencies of mentions of these keywords in books published between the years 1960 and 2018 are shown in Figure 8 [6]. This time frame was chosen as there was either negligible or no mentions of our keywords in books prior to 1960.

We observed that among all the keywords, natural language processing, and image recognition had been written about the most. In fact, the number of mentions of natural language processing is so high that the mentions of other keywords seem diminished in the graph in comparison. This indicated to us the popularity of input methods using natural language processing and image recognition, which we decided to incorporate in our project.

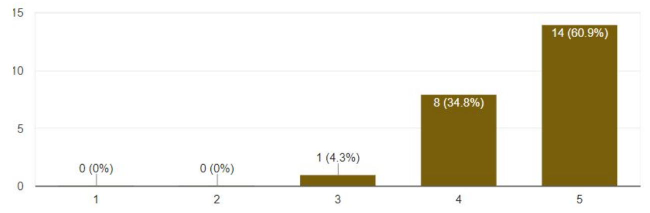


Figure 12: Feature: Recommender System for Ranking Recipes

4. TIME-LINE

We plan to increase parallelism by allocating tasks within ourselves and following a modularized approach to software development. By doing so we hope to find and fix errors faster and complete development such that we have sufficient time for a proper validation phase. Our timeline of activities for this project is given in Figure 13.

5. EFFORT ESTIMATION

We have used Use Case Points [3] to arrive at an estimate for the effort required for the implementation of the feature enhancements discussed in this document. Figure 14 shows the results of our effort estimation. In order to complete our project, we have come up with an estimate of 187 man hours. The Technical Complexity of the project, as well as several environmental factors, have been considered while determining the effort required. The motivation within the team, familiarity with the project, programming experience of the team members and the capability of the lead analyst were some of the environmental factors that played a role in the estimation process. The reusability of the existing codebase and the use of third-party software and libraries (Actors) were also taken into account to provide a more accurate result. We understand that this is not an ideal estimate. However, this activity helped us to dissect and view the factors that could affect the delivery of our application.

6. SOFTWARE DEVELOPMENT LIFE CYCLE

A software development life cycle is aimed at enabling rapid development of software with minimization of errors through substantial planning and management. It is crucial to adopt a software development process which accurately describes the aspects of the project in hand considering the requirements specified and resources available. Bearing these factors in mind, a Prototyping model of development has been chosen. Further, this model of development provides a means to deliver results early in the timeline of the project. We will also adopt TDD as a primary model for development. This will help us to learn quickly by failing fast. TDD will also help us to streamline our approach helping us to adhere to the requirements of the project. The management and support tasks for the project will be accomplished through github primarily and the development is to be done on IntelliJ Idea IDE for development, testing and compilation of the code.

6.1 Plan

This was the initial stage of our development process where we found the scope of the problem and possible solutions for the problem. First, the problem was identified as increasing the user experience of the application and need of sophisticating the application. The scope of the problems was analyzed with the timeline and resources in hand and the five use cases discussed in section 2 were proposed as a result.

6.2 Requirements

In this section, we will see in detail about how we collected the requirements for improving the existing project. The inputs were majorly from previous evaluation report which included suggestion from peers. From the inputs that were gathered, the 5 use cases that were discussed in section 2 were decided to be worked on. All the use cases which are proposed in Section 2 in one way or the other tries to better the user experience of the application and aims at increasing functional flexibility of the application.

6.3 Design & Implementation

The method of implementation that we have thought of is to divide the development of the new use cases among the team members. And each team member will be responsible for writing a test case for another use case that the member is not working on. This would make each team member gain exposure to both development and testing side of the development process and would also give functional knowledge of two use cases of the application. As stated above, we would be following TDD methodology, which we believe would help us find critical errors at an early stage of development and would enable us to rectify them rapidly.

6.4 Test

It is highly essential that any software built must be subject to rigorous testing before making it available for usage. This testing must not only aim at maintaining a high quality of code but also provide a mechanism to ensure that the user requirements have been satisfied accurately. Following a test-driven methodology has thus been adopted for our project. This allows maintaining simplicity, modularity, and flexibility in the code base while also providing a built-in process of evaluation against the design. Our objective would be to build test cases to each of the requirements corresponding to the use cases presented in section 2. This further enables us to obtain the perspective of an end user as TDD makes the developer focus on the requirements before writing the code. Following the construction of these test cases, the actual software coding is done to pass the test cases. Intuitively, this may result in failure, but this provides crucial insight into the short-comings of the current code. Next, the code is improved with an aim of barely passing the existing test cases to satisfy the bare-bone design requirements. The written code is then validated against all the test cases and upon success, the code is deemed to meet the test requirements.

6.5 Deploy & Maintenance

As we know, any web-based application requires very little effort on part of the user for the installation. The architecture of our software application uses a web archive file that is deployed on a remote server hosted on an Amazon Web Services Elastic Computing instance. The user interaction with our application is through slash command requests to the Slack interface which invokes the Slack API and then

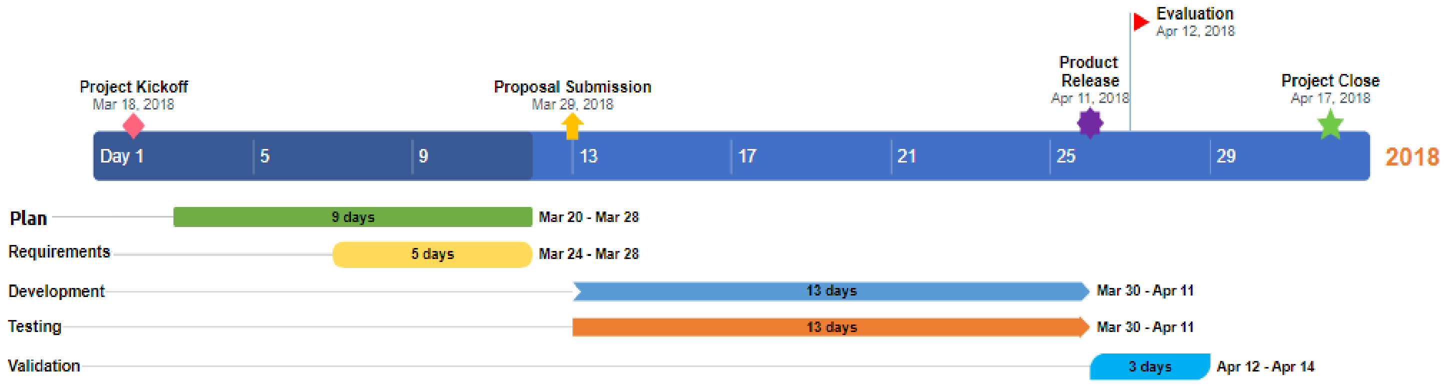
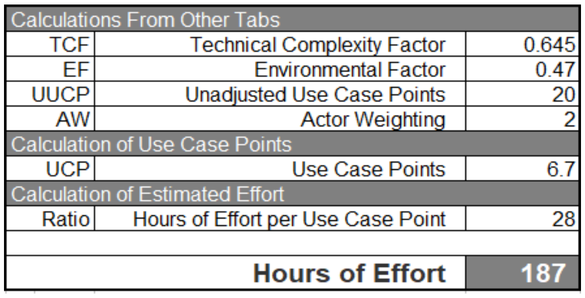


Figure 13: Project Time Line



Calculations From Other Tabs		
TCF	Technical Complexity Factor	0.645
EF	Environmental Factor	0.47
UUCP	Unadjusted Use Case Points	20
AW	Actor Weighting	2
Calculation of Use Case Points		
UCP	Use Case Points	6.7
Calculation of Estimated Effort		
Ratio	Hours of Effort per Use Case Point	28
	Hours of Effort	**187**

Figure 14: Effort Estimation results

this request is redirected to an URL hosted on our server. A Kaggle database is maintained at this server which acts as the knowledge base of recipes for processing. This simple design has allowed us to improve upon processing times and provides a performance boost by removing redundant interactions between different modules of the software.

7. NEW EVALUATION PLAN

The evaluation process for this project focuses on **quantitatively** characterizing the possible benefit due to the feature upgrades that have been made to the existing software. A primary emphasis on this concern has been placed while designing the evaluation for the software. In this project undertaking, improvements in the method of input to the system and allowing for a complete output have been proposed.

Aspect	CookBetter v1.0	CookBetter v2.0
Input	Form filling	CSV, Natural Language and Image
Output	Link to recipe website	Recipe(s), Intuitive presentation with Feedback
Process	Rigid Recommendation	Optimal Recommendation

Table 1: Overall Feature comparison

An online survey has been selected as the evaluation tool for this project and the survey points have been made specific to center on the feature upgrades. Table 1, is one possible rubric to measure actual improvement is shown.

As most of the feature upgrades aim at improving the usability(user experience), the evaluation form must have pointers to record user experience improvement due to these feature upgrades. Additionally, recording feedback on the user's satisfaction in completing a specific task on the software is important. The following sections are some of the significant survey points.

7.1 Task Completion

We plan to provide the user with a list of tasks that he will have to complete using the different versions of the software and then record the benefits (if any) or difficulties faced by the user in the process. The tasks will be designed to simulate real-world user quandaries and will capture the nuances of user requirements in increasing proportion. The primitive survey pointer is aimed at recording the basic success or failure of the task itself while further sections will seek to investigate the points pertaining to ease, user experience, completeness, etc.

7.2 Ease of Performing Task

An ideal system handles border cases just as smoothly as the normal cases. With our multitask-based evaluation, we plan to compare the new system's response to increasingly difficult searches with that of the previous release of the software. This aspect of our evaluation has a direct bearing on capturing the implications of the feature upgrades of 'image-based search' and 'natural language-based search'.

7.3 Completeness & Accuracy of Output

As the new features include a purchase suggestion form, as well as a video for the recipe, suggested, the evaluation seeks to record the participant's feedback on whether these features contribute the completeness of the solution with respect to the initial problem statement that the software seeks to tackle. Also, the evaluation will determine if the goal of enabling user engagement with the software by providing the like, dislike and comment section, has been met.

Further, the intuitive recipe recommendation system that is part of the new release of the software targets the accuracy with which the user's request is accomplished. The evaluation form will record the user experience difference to this feature from the previous version of the software.

8. CONCLUSION

The arduous task of searching for recipes that meet our personal dietary restrictions has been made easy using the initial project idea of the ”Cook Better bot” in a very rudimentary fashion using a recipe-search application integrated into the easy-to-use Slack interface. In the current phase of software development, our primary focus has been on improving every aspect of the system. On one hand, we have planned to implement an image and natural language-based input that allows the user interaction with the system to be more flexible, while on the other, the output has been decided to be made more engaging with visualizations and user involvement (feedback rating). Further, the recommendation process itself has been proposed to be overhauled with a multi-factor-based searching algorithm. Also, with the new user feedback feature, the users can take opinions of others into account while finalizing a recipe.

It has been proposed that these features will be developed and tested in a strict conformation to established software development principles. Finally, to validate the said improvements we have put forth a comprehensive evaluation plan that takes the feature specifications made in this design document as the criteria for assessment of our improved software.

References

[1] Amazon. *Amazon Rekognition.* https://aws.amazon.com/rekognition/. [Online; accessed 3/22/2018]. 2018.

[2] Amazon. *Amazon Web Services.* https://aws.amazon.com/. [Online; accessed 3/22/2018]. 2018.

[3] Tyner Blain. *Use Case Points Calculator.* http://tynerblain.com/downloads/UseCasePoints.xls. [Online; accessed 3/22/2018]. 2018.

[4] Clarifai. *Clarifai.* https://www.clarifai.com. [Online; accessed 3/22/2018]. 2018.

[5] Google. *Cloud Vision API by Google.* https://cloud.google.com/vision/. [Online; accessed 3/22/2018]. 2018.

[6] Google. *Google NGram Viewer.* https://books.google.com/ngrams. [Online; accessed 3/22/2018]. 2018.

[7] *MySQL.* https://www.mysql.com/. [Online; accessed 3/22/2018]. 2018.

[8] Pivotal. *Spring Boot.* https://spring.io/guides/gs/spring-boot/. [Online; accessed 3/22/2018]. 2018.

[9] Francesco Ricci, Lior Rokach, and Bracha Shapira. “Introduction to recommender systems handbook”. In: *Recommender systems handbook.* Springer, 2011, pp. 1–35.