Akshat Goyal, Ashvin Lohiya, Shreeyan Shayan, Vineel Guntupalli

Incremental and Regression Testing Sprint 2

Classification of Components

Module Description

Login Page

This component allows users to login into the website using their google accounts or bypass the login and use the website in guest mode.

Inputs: Click on guest login.

Outputs: Redirects to Ingredients page.

Ingredients Page

Users can use the two search bars on this page to specify the ingredients they want and don't want. This module delivers these inputs to the machine learning components and pass the received recipe information to the recipe page.

Inputs: Specify/Add ingredients that user wants and does not want in their recipe.

Outputs: Redirects to recipe page.

All Recipe Page

Users can come to this page to look at all the recipes in our database. Interacts with the parsing module to get some number of recipes to display at a time. Once a recipe is selected, gets more information about the recipe and passes it to the recipe page.

Input: Select particular recipe to read detailed description / instruction.

Outputs: Redirect to the recipe page to display the selected recipe in detail

Recipe Template

Displays information received from the ingredients page or the all recipe page and uses the user information received from the login page to facilitate sharing.

Inputs: Takes information of a particular recipe

Outputs: Displays the recipe in a properly formatted manner

Parsing

Parses the dataset to enable the machine learning and clustering components to take full advantage of the data. Also provides information to all recipe page.

Inputs: Json and csv data. Also, receives calls from all recipe page.

Outputs: Produces .txt and .list files for parsed directions, parsed ingredients, etc. Returns recipe information to all recipe page.

Ingredient Clustering

The ingredient clustering model is used by both probabilistic direction generating model and probabilistic ingredient generating model. This module is responsible for clustering the ingredients in the dataset and giving them similarity scores.

Input: The list of ingredients in the dataset. It also receives the ingredient preferences of the user.

Output: Returns clusters to the machine learning modules for further computation.

Probabilistic direction generating model

This module is responsible for finding the closest pre-existing recipe and/or creating a completely new recipe using the ingredients provided by the Ingredients page. Works together with the probabilistic ingredient generating model to find the directions of the computer-generated recipe or the pre-existing recipe itself.

Input: Clusters of the ingredients

Output: Gives a new recipe to the user which is closest to their preference.

Probabilistic ingredient generating model

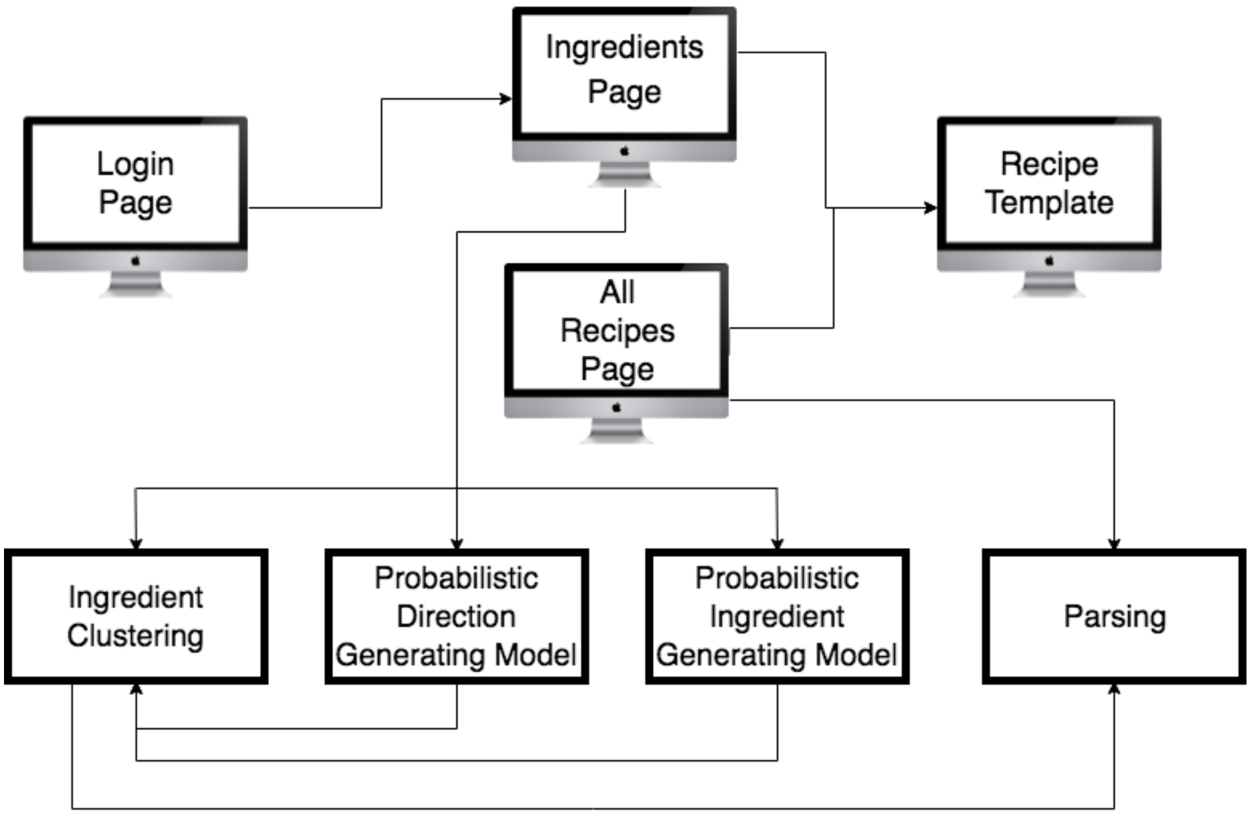
This module finds the ingredient subset with the highest similarity score and have a higher chance of occurring in the same recipe using the ingredient clustering module. Works in parallel to the probabilistic direction model to find the closest pre-existing/computer generated recipe.

Input: Clusters of the ingredients

Output: Generates a new recipe based on the user's preferences and returns it.

Incremental Testing Technique

The testing approach used was top down in which we implemented the modules above and used stubs to simulate the behavior of the lower modules.



As this is the final step of our project, the stubs we used in Sprint 1 were replaced by actual working modules one by one and then tested. The advantage of following a top down approach was that we could quickly pinpoint the location of the error. It also helped us in getting a good prototype of our early app as top down helps follow the blueprint in a logical manner.

Incremental and Regression Testing

Defect Log

In our Sprint 1, we worked on modules Login Page, Parsing, Ingredients Page, All recipe page and Recipe template. Here, we discuss them and the remaining modules - Ingredient Clustering, probabilistic direction generating model, probabilistic ingredient generating model.

Module	Login page		
Defect No.	Description	Severity	Solution
1	The variables were storing User's google information properly but google API was not redirected to main landing page on successful validation instead it was redirected to temporary pop-up window.	3 (The login page was not redirecting to main landing page on successful sign in)	Created a new variable to store URL of landing page (Ingredients page) and used window.location.replace function to redirect to landing page on successful google login.

Module	Parsing		
Defect No.	Description	Severity	Solution

1	The recipe vectors and the ingredients in the directions of a recipe were not consistent. This resulted in many ingredients not showing up even though they were present.	2 (The majority results were not accurate)	The data was pre-processed further to check if each of the value in recipe vector was present in the recipe directions. If present, the recipe vector was updated to contain that value.
2	Multiple verbs occurring together broke the parsing. It causes the verb-verb pair to be linked together rather than the verb-noun pair	2 (The results were inaccurate)	The state transition design was changed to handle multiple verbs occurring together,

Module	Ingredients Page		
Defect No.	Description	Severity	Solution
1	route.py crashes when the toggle button that is used to switch between ML generated and pre-existing recipes is set to “off”. The exact error thrown is “400 bad request” on line togglestr =request.form['toggle']	3 (The user couldn’t login with google, but could always login as a guest and use the app)	Toggle was implemented in a way that if it was set to “off”, it would not be included in the form. Thus, the line threw a bad request. To solve this, I used an exception handling by wrapping the statement with try and except, and in case an exception is thrown I explicitly set the variables value.

Module	All Recipe Page		
Defect No.	Description	Severity	Solution
1	Sorting the recipes on a particular attribute changed the original order of recipes and dataset and thus linked incorrect titles to incorrect recipes	2 (No correct recipe was linked to the correct description)	Along with the recipe values, all the titles and dataset values were shuffled in the same order to achieve the same ordering.
2	Once you hit the next or previous buttons on the page, the sorting order goes back to the original	2 (The results were incorrect)	The functionality was changed at the backend and front end to preprocess and store the sorted values and accordingly display the ones needed
3	There were a lot of missing values in the dataset which caused the sorting to crash	1 (The app crashed)	All the missing values were substituted with +infinity if ascending or -infinity if descending to fix the issue
4	After hitting the next or previous button, the dropdown menu on the front-end were not retaining the values in which the recipes were sorted	3 (The results were correct, this was more to assure the user that recipes are indeed sorted)	The problem was solved by saving the values of drop down menus in variables and explicitly setting the values of the menu each time allrecipes page is loaded. This can be achieved using flask templating.

Module	Recipe Template		
Defect No.	Description	Severity	Solution
1	The panel that is used to display list of ingredients selected and list of ingredients included in the recipe for Pre-existing and ML generated recipes was also included in the recipes from allrecipes page.	3 (There was a workaround for it)	This is because the same recipe template is used for all of them. The problem was solved by if else statements using flask templating.

Module	Ingredient Clustering		
Defect No.	Description	Severity	Solution
1	K means was not clustering well as it put every object into a particular cluster (hard clustering)	2 (The results were not accurate enough)	Used a clustering method based on the principles of collaborative filtering to find similarity measures between all pairs of objects.

Module	<u>Probabilistic direction generating model</u>		
Defect No.	Description	Severity	Solution
1	The score function used to select the subset of ingredients was not taking into account the size of the subset. It hence always ended up picking the largest subset.	3 (The results were not up to the par)	The score function was updated to also take into account the number of ingredients thus bettering the results.
2	The app would crash because the score was being divided by zero at times	1 (The app would crash)	Fixed the error by normalizing the values

Module	<u>Probabilistic ingredient generating model</u>		
Defect No.	Description	Severity	Solution
1	The program always ended up picking the same ingredient for each step.	2 (Inaccurate results)	Performed Laplace smoothing on the dataset files.

2	Results were extremely randomized initially. The issue turned out to be an incorrectly named dataset file	2 (inaccurate results)	The files were correctly renamed, and correct files were used.
3	The final steps had multiple appearance of the same ingredient.	3 (Not up to par results)	Each time an ingredient was considered, it was removed from the list of ingredients to be considered next