Project Title: ReciPull  
Team Name: AJARA  
Authors: Alexis Cole, Justin Dong, Rhianna So, Austin Wong, Allen Yao

Member Emails: alexisrosecole@gmail.com, jsdong@ucsb.edu, rhiannaso@ucsb.edu, austinwong@ucsb.edu, allenxueyiyao@gmail.com

Video: Web-App || Alexa

Vision Statement

**Project Summary:** ReciPull is a web application designed by college students for college students. A common problem of college students around the nation is the inability to produce quality food on a tight budget. Many students find themselves at a loss for inspiration in different recipes to cook. ReciPull is the perfect application to solve this crisis. A user is able to input their available ingredients to see a list of various recipes to cook from a database. With this application, college students can easily access recipes that they can make with the ingredients they already have, which makes it easy for them to cook when they are on a tight budget or do not have time to go grocery shopping.

**Outcome:** ReciPull will be a working web-app that will take user input on what ingredients they are working with and will output a list of recipes that they could cook using their available ingredients. Each recipe will be accompanied with essential information: nutritional facts and the number of owned ingredients included. Recipes can also be upvoted or liked, allowing users to see the most favorited recipes, along with other options to filter recipes. Finally, Recipull will integrate Echo voice recognition for easier search capabilities and employ a social media platform that will allow users to connect with their friends for recipe and ingredient sharing.
**Project Milestones:** We intend on having all initial project specifications solidified by April 12. Specifications can be modified as the project progresses, however. We plan to have a rough design of the overall layout of the web-app by April 24. We want to have a working prototype that resembles the final product by May 10. The working prototype should have the front-end, back-end, and database components all connected and functioning properly together. One or two key features should be implemented and working according to specifications. Testing should be done throughout the process of implementing the web-app, especially after the implementation of each new feature. Final testing should be done by June 5 to ensure complete functionality, leaving time to resolve minor issues or improve upon certain features.

**Implementation Platform and Technologies:** We plan on developing a web application for ReciPull and would like to develop ReciPull into an Alexa Skill as well. The technologies we will utilize for the client side code of the web application are React, HTML, and CSS. The technology we will be utilizing for the server side code is Java. For the database, we will be using Amazon Web Services and MySQL. For the Alexa skill, we will be using Python (may switch once we implement the web framework) and JSON.

**Initial Milestones (Sprints/Dates) and Planned Implementation:** By the end of Sprint 1, on 4/26, we want our web app to have the following functionalities: the ability to take a list of ingredients as user input and run a search to find recipes with said ingredients. This is the main purpose of ReciPull, thus we want to focus on this first. We plan to use an API to implement this. If possible, this would also be where we would integrate the echo, allowing the user to speak ingredients that trigger the search. By the end of Sprint 2, on 5/10, we want to implement a social networking aspect of ReciPull. We plan to use the Pinterest API to allow users to share which
recipes they have tried. We would also like to create a platform for users to be able to borrow and lend ingredients. By the end of Sprint 3, on 5/22, we want to implement health and tracking functionalities of the web app. These would include nutritional facts for recipes, information on what is in season, and meal planning and tracking capabilities.

**Completed Tasks During Sprints:** By the end of Sprint 1, on 4/26, we were able to set up the recipe and ingredient databases, as well as retrieve applicable recipes given an ingredient and retrieve ingredients given a recipe. We achieved this by connecting the back-end (Java) to the database and created dummy Java programs to demonstrate this, since we had not yet connected the front-end to the back-end. In terms of front-end, we finished each individual component of the user interface (ingredient list, recipe list, header) but did not get the chance to combine all components into a cohesive page. In terms of the back-end, we created templates for recipe objects and set up the framework to process through recipes retrieved from the database. We also began to experiment with the Amazon Alexa. By the end of Sprint 2, on 5/10, we were able to update the database to handle recipe upvoting and downvoting, as well as filters. We created methods to sort through the recipes according to filters, returning recipes that contained a specific filter. We also were able to directly create recipe objects in Java directly from the data we retrieved from the database. We figured out a way to update the actual database information when receiving upvotes or downvotes. Regarding Echo integration, we were able to create a basic Alexa skill and created an Alexa lambda function. For front-end, we combined the previously completed components of the webpage into a seamless user interface. We are in the process of working on combining our front-end and back-end together, as it has been a major blocker during the sprint. By the end of Sprint 3, on 5/22, we were able to connect our front-end
and back-end together, which allowed us to have the basic functionality of ReciPull working. We also completed test suites for our back-end, front-end, and Alexa skill. After getting the basic functionality working, we were able to implement the upvoting and downvoting functionality through the upvote and downvote buttons on the webpage, as well as the filtering functionality through the filter buttons on the page. We also were able to implement the Alexa skill to take an ingredient and return recipes that include that ingredient. The Alexa skill also emails the user a list of applicable recipes and links to those recipes.

**Challenges Faced/Overcome**

Our biggest challenge throughout the quarter was connecting our front-end with our back-end. When we were preparing for Sprint 1, we initially thought we would be able to get everything connected by the end of Sprint 1. In the end, we were not able to solve this solution until midway through Sprint 3. Our biggest setback was the fact that none of us were very familiar with how to relay information across different languages, as well as the limited documentation available for specifically connecting a React front-end with a Java back-end. This made it significantly more difficult for us to figure out how to connect our project components together. We were able to solve this problem with the help of both Nazmus and Chandra. To connect our front-end with our back-end, we used a jetty server and get requests to send information to and from our front-end and back-end. We used elements in our main HTML file to store and retrieve information in their innerHTML, which occurred as a result of our get requests. This allowed us to pass and retrieve information from both ends of our webpage.

Another challenge we faced was timing issues for our webpage button functionality. We had an onclick function in both our HTML and React, so there was a race to see which function
would execute first. We wanted our React function to run first and update the innerHTML of an element in our main HTML file, and then have the HTML function run and retrieve the information from the innerHTML. However, we found that the HTML function seemed to always execute first. This issue had us blocked for quite some time, but we were able to solve it by wrapping our HTML function in an if statement that checked if the innerHTML of our specified element was empty or not. This helped us ensure that our HTML function would run after our React function, because it would only run if the innerHTML was not empty.

**Missing Features/Functionality**

During the three sprints, we were not able to implement all the features that we originally wanted to. We intended on having some form of social media integration, specifically Pinterest, which would allow users to login to their Pinterest account and share or save the recipes that they found on our site. We were able to do some research on the Pinterest API, but did not have enough time to implement this feature during the sprints. We also discussed having some form of nutritional value information available on our webpage. Originally, we hoped to provide the nutritional value of each recipe with every recipe that was outputted. However, it was difficult to figure out how to cleanly format this information, as well as retrieve this information for every recipe that we found. As a result, we did not get a chance to implement this in our project. An important functionality that we briefly discussed and hope to implement in the future is the usage of a recipe API. Currently, we are using our own database of recipes and ingredients that we have manually inputted. While this works, we would like to be able to implement a recipe API so that we could have a more expansive collection of recipes to pull from, without having to manually input all the information. This is something we would definitely want to implement.
moving forward. We would also want to use a web host in the future so that people can access our web-app from everywhere, seeing as we can only run the web-app from a local host as of now. As for our Alexa skill, we hope to be able to find a way to take in a list of ingredients, seeing as it only takes in a single ingredient as of right now.

**System Architecture**

- High-level overview

---

**System Architecture Diagram**

- User Interface:
  - Take User Input
  - Display Recipes with inputted ingredients

- Back-End Processor:
  - Query to Database
  - Find Recipe IDs that Match to Ingredient IDs
  - Sort List of Recipes According to Any Filters
  - Return List of Recipes

- Database:
  - Ingredient Table
    - Match Ingredient to Ingredient ID
  - Pivot Table
    - Match Recipe ID to Recipe
  - Recipe Table
- Detailed Design
  - UML Diagrams

- Sequence Diagrams
  - User Interaction
Synchronous Interactions

UI Mockups
Requirements

- Use Cases found on team Trello
  - Use cases under each Sprint task (i.e. UI Card in Sprint 1 List)
- GitHub
- Travis-CI
- Video Demos
  - Web-App
  - Alexa

Retrospective Info

- Things we chose to fix:
  - After our Sprint 1 retrospective, we decided to work on breaking up our use cases into smaller tasks. We thought that breaking up use cases into smaller, more specific tasks would help us have a better idea of what we needed to work on, as well as make the tasks seem more achievable.
  - After our Sprint 2 retrospective, we decided to work on working and communicating with one another better. We thought that this would help us progress more efficiently. Our goal was to keep each other updated on blockers more regularly and collectively decide what our priorities needed to be.
  - After our Sprint 3 retrospective, we decided to focus on prioritizing necessary functionalities for our demo. We thought that this would allow us to see dependencies more clearly and formulate a more useful strategy for finishing our project.
- What we did to fix these things:
  - In order to break our use cases into smaller tasks, we took the time to thoroughly discuss what we wanted to get done for the next sprints. We then determined what steps we would need to take to get each of these done, ensuring that the steps were as specific and atomic as possible. We turned these steps into tasks and assigned the tasks to each of us accordingly.
  - In order to communicate more clearly, we set up specific times to meet outside of class. We discussed whether the entire group needed to meet or just subsets. Physically putting these times into our calendars before issues came up, meant that we weren’t scrambling for help once blockers inevitably surfaced. It also held us accountable for the work that we were assigned to.
  - In order to prioritize our tasks, we utilized the meetings that we had started after the Sprint 2 retrospective. We all discussed what we pictured our end result to be,
and collectively decided what the highest priorities should be. We were clear and honest about the feasibility of certain functionalities. We focused on the bare minimum of what we needed for the demo, then continued to add functionalities after those were completed.

- Were we able to fix these things?
  - We were definitely able to break up our use cases into smaller tasks, which helped us accomplish more in future sprints and gain better focus on what needed to be done.
  - We were able to fix our communication issues by setting specific times to meet in order to accommodate for everyone’s busy schedules. Our progress definitely reflected this change.
  - We were eventually able to improve our priorities, using the strategies listed above. However, this fix happened later than the others, which resulted in us finishing our base functionality later than we had originally hoped.

**Commits**
### Commits on May 29, 2019

**fixed voting system to allow for only one vote**
- Justin Dong authored and Justin Dong committed 4 days ago

**Merge pull request #56 from ReciPull/UI**
- JustinDong1 committed 4 days ago

**Updated UI**
- rhiannase committed 4 days ago

### Commits on May 28, 2019

**Merge pull request #54 from ReciPull/Connector**
- alexisole99 committed 5 days ago

**Trying different host**
- rhiannase committed 5 days ago

**Merge pull request #53 from ReciPull/alexa**
- justindong1 committed 5 days ago

**Merge pull request #52 from ReciPull/filters**
- rhiannase committed 5 days ago

**added filter functionality**
- Justin Dong authored and Justin Dong committed 5 days ago

**alexa skill**
- alexisole99 committed 5 days ago

### Commits on May 23, 2019

**Merge pull request #51 from ReciPull/lambda**
- austinwong3 committed 10 days ago

**updated**
- alexisole99 committed 10 days ago

**added link functionality for cards**
- Justin Dong authored and Justin Dong committed 10 days ago

**added link functionality for cards**
- Justin Dong authored and Justin Dong committed 10 days ago

**package resolution**
- austinwong3 committed 10 days ago

**React tests for searchbar and ingredient section**
- austinwong3 committed 10 days ago

**Merge pull request #50 from ReciPull/Connector**
- austinwong3 committed 10 days ago

**Updated interface**
- rhiannase committed 10 days ago

**Implemented vote buttons and set up for filters**
- rhiannase committed 11 days ago
Commits on May 22, 2019

Merge branch 'master' of github.com:ReciPull/recipull.github.io into ...
  rhiannaso committed 11 days ago

Updated driver to pass votes
  rhiannaso committed 11 days ago

Merge pull request #49 from ReciPull/output
  rhiannaso committed 11 days ago

added voting functionality to cards
  Justin Dong authored and Justin Dong committed 11 days ago

added basic voting functionality to cards
  Justin Dong authored and Justin Dong committed 11 days ago

Merge branch 'master' of github.com:ReciPull/recipull.github.io into ...
  rhiannaso committed 11 days ago

Wrote potential functions for filters
  rhiannaso committed 11 days ago

Merge pull request #47 from ReciPull/Connector
  xueyiyao committed 11 days ago

Merge pull request #48 from ReciPull/output
  justindong1 committed 11 days ago

completed functionality for recipe output
  Justin Dong committed 11 days ago

Merge branch 'master' of github.com:ReciPull/recipull.github.io into ...
  rhiannaso committed 11 days ago

Updated driver
  rhiannaso committed 11 days ago

Merge pull request #46 from ReciPull/lambda
  rhiannaso committed 11 days ago

GetNameAlexa
  alexiscole99 committed 11 days ago

Merge pull request #45 from ReciPull/output
  justindong1 committed 11 days ago

fixed color and text for IngredientNum
  Justin Dong committed 11 days ago

Merge pull request #44 from ReciPull/output
  justindong1 committed 11 days ago
Commits on May 22, 2019

- fixed numingredient color
  Justin Dong committed 11 days ago

- fixed numingredients output
  Justin Dong committed 11 days ago

- Merge pull request #43 from ReciPull/Connector
  alexiscode99 committed 11 days ago

- Merge branch 'master' of github.com:ReciPull/recipull.github.io into ...
  rhiannaso committed 11 days ago

- Added tests
  rhiannaso committed 11 days ago

- Merge pull request #42 from ReciPull/output
  justindong1 committed 11 days ago

- added numingredients to card
  Justin Dong committed 11 days ago

- fixed blank outputs for searchbar
  Justin Dong committed 11 days ago

- delete fixed
  suustinwong3 committed 11 days ago

Commits on May 21, 2019

- completed card output
  Justin Dong committed 12 days ago

- added array card output
  Justin Dong committed 12 days ago

- Merge branch 'master' of https://github.com/ReciPull/recipull.github.io...
  Justin Dong committed 12 days ago

- Merge pull request #41 from ReciPull/Connector
  justindong1 committed 12 days ago

- Modified Driver Program
  rhiannaso committed 12 days ago

- initial commit
  Justin Dong authored and Justin Dong committed 13 days ago

Commits on May 17, 2019

- Pushing src
  rhiannaso committed 16 days ago

Commits on May 16, 2019

- Connect back-end to front-end
  rhiannaso committed 17 days ago

- Finally connected
  rhiannaso committed 17 days ago
## Commits on May 18, 2019

- **Fix merge conflict**
  - rhiannase committed 18 days ago

- **Connector files**
  - rhiannase committed 18 days ago

## Commits on May 14, 2019

- **Merge pull request #40 from ReciPull/ReactUI**
  - austiniweng3 committed 19 days ago

- **Ds Store**
  - austiniweng3 committed 19 days ago

- **search recipe button and scroll fix**
  - austiniweng3 committed 19 days ago

## Commits on May 13, 2019

- **Fixing merge conflict**
  - rhiannase committed 20 days ago

- **Fixing merge conflict**
  - rhiannase committed 20 days ago

- **Merge branch 'sqlTest'**
  - rhiannase committed 20 days ago

- **Merge branch 'master' into sqlTest**
  - rhiannase committed 20 days ago

## Commits on May 10, 2019

- **"Resolving merge conflict"**
  - rhiannase committed 23 days ago

- **Merge pull request #39 from ReciPull/Back_End_Home**
  - xueyiya committed 24 days ago

- **finished Recipe class methods and updated Back_End_Processor**
  - Allen Yao committed 24 days ago

- **finished Recipe class methods**
  - Allen Yao committed 24 days ago

- **Merge pull request #38 from ReciPull/Alexis**
  - rhiannase committed 24 days ago

## Commits on May 9, 2019

- **dummy Python simple http server**
  - alexiscle90 committed 24 days ago

- **dummy Python simple http server**
  - alexiscle90 committed 24 days ago

- **dummy Python simple http server**
  - alexiscle90 committed 24 days ago

- **dummy Python simple http server**
  - alexiscle90 committed 24 days ago
<table>
<thead>
<tr>
<th>Commit Message</th>
<th>Author</th>
<th>Date</th>
<th>Commit ID</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>gettingFromRecipe working in Python</td>
<td>alexiscole99</td>
<td>24 days ago</td>
<td>7d919e8</td>
<td></td>
</tr>
<tr>
<td>gettingFromRecipe working in Python</td>
<td>alexiscole99</td>
<td>24 days ago</td>
<td>c431d10</td>
<td></td>
</tr>
<tr>
<td>connects</td>
<td>alexiscole99</td>
<td>24 days ago</td>
<td>87774c7</td>
<td></td>
</tr>
<tr>
<td>connects</td>
<td>alexiscole99</td>
<td>24 days ago</td>
<td>396c3f2</td>
<td></td>
</tr>
<tr>
<td>Merge pull request #37 from ReciPull/Alexis</td>
<td>Verified</td>
<td>0bc56047</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge pull request #37 from ReciPull/Alexis</td>
<td>Verified</td>
<td>b875817</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge pull request #35 from ReciPull/Back_End</td>
<td>Verified</td>
<td>fcd895d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adding db.json</td>
<td>Verified</td>
<td>de36537</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge pull request #34 from ReciPull/ReactUI</td>
<td>Verified</td>
<td>01cc6ea</td>
<td></td>
<td></td>
</tr>
<tr>
<td>index</td>
<td>Verified</td>
<td>5c8c248</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Created driver program and handles votes/filters</td>
<td>Verified</td>
<td>98e94da</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge pull request #32 from ReciPull/ReactUI</td>
<td>Verified</td>
<td>5ff5747</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ignore test</td>
<td>Verified</td>
<td>db24792</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ignore hack</td>
<td>Verified</td>
<td>50f85f5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge pull request #31 from ReciPull/ReactUI</td>
<td>Verified</td>
<td>c464f52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>node fix again</td>
<td>Verified</td>
<td>cbf8484</td>
<td></td>
<td></td>
</tr>
<tr>
<td>folder test</td>
<td>Verified</td>
<td>fd5d7e9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge pull request #30 from ReciPull/ReactUI</td>
<td>Verified</td>
<td>03e98f8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>node fix 2</td>
<td>Verified</td>
<td>13e8e68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Commits on Apr 26, 2019

**Merge pull request #18 from ReciPull/Back_End_Home**
- **Back_End_Processor working with dummy inputs and data**
- **added card component for frontend ui**
- **Ingredient List Ready**

**Commits on Apr 25, 2019**

**Merge pull request #16 from ReciPull/sqlTest**
- **Test files for retrieving ingredients and recipes**
- **Added jar file**
- **Merge pull request #14 from ReciPull/Back_End_Home**
- **made changes to Back_End_Processor**

**Commits on Apr 24, 2019**

**Merge pull request #13 from ReciPull/sqlTest**
- **Tentative queries for interaction with database**
- **Merge pull request #12 from ReciPull/Back_End**
- **Merge branch 'master' into Back_End**
- **made changes to Recipe class**
- **Merge pull request #11 from ReciPull/Back_End_Home**
- **Merge pull request #10 from ReciPull/sqlTest**
- **Dummy program for database/java interaction**
- **made changes to Back_End_Processor and Database classes**
<table>
<thead>
<tr>
<th>Date</th>
<th>Commit Message</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr 24, 2019</td>
<td><strong>Merge pull request #9 from ReciPull/Back_End</strong></td>
<td>xueyiya committed on Apr 24</td>
</tr>
<tr>
<td></td>
<td>Added Back_End_Processor and Database classes</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Merge pull request #8 from ReciPull/Back_End</strong></td>
<td>austinwong3 committed on Apr 22</td>
</tr>
<tr>
<td></td>
<td><strong>Merge pull request #7 from ReciPull/reactTutorials</strong></td>
<td>austinwong3 committed on Apr 22</td>
</tr>
<tr>
<td></td>
<td>added backend java folder</td>
<td>xueyiya committed on Apr 22</td>
</tr>
<tr>
<td></td>
<td>created react template recipull</td>
<td>Justin Dong committed on Apr 22</td>
</tr>
<tr>
<td></td>
<td><strong>Merge pull request #6 from ReciPull/awsTutorial</strong></td>
<td>rhiannaso committed on Apr 20</td>
</tr>
<tr>
<td></td>
<td>Completed basic RDS tutorial</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Merge pull request #5 from ReciPull/awsTutorial</strong></td>
<td>rhiannaso committed on Apr 20</td>
</tr>
<tr>
<td></td>
<td>AWS info</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Merge pull request #4 from ReciPull/echoTutorial</strong></td>
<td>rhiannaso committed on Apr 17</td>
</tr>
<tr>
<td></td>
<td>tutorial link</td>
<td>Alexis Cole committed on Apr 17</td>
</tr>
<tr>
<td></td>
<td><strong>Merge pull request #3 from ReciPull/travisTest</strong></td>
<td>rhiannaso committed on Apr 17</td>
</tr>
<tr>
<td></td>
<td>Travis Test</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Travis Test</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Travis Test</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Travis Test</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Travis Test</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Travis Test</strong></td>
<td></td>
</tr>
</tbody>
</table>
Note: Alexis and Rhianna worked together on the ‘Test files for retrieving ingredients and recipes’ and ‘Tentative queries for interaction with database.’

**Test Files**

- **Back-end: Java JUnit Testing**
  The files used for JUnit testing of the back-end are: TestGetIngId.java, TestIng.java, TestRecipeIds.java, TestRecipes.java, TestRecipeOrder.java, TestFilter.java.

  TestGetIngId.java covers the testing of the function that retrieves ingredient IDs from the database given an ingredient(s) as a string. TestIng.java covers the testing of converting the retrieved ingredient IDs to a string that is formatted in proper mySQL syntax. TestRecipeIds.java covers the testing of retrieving all the recipe IDs of recipes (that contain at least one of the ingredients) from the database, specifically from the pivot table, and formatting them into a string of mySQL syntax. TestRecipes.java covers the testing of creating recipe objects (from the retrieved recipe IDs) with all the fields containing correct information. TestRecipeOrder.java covers the testing of correctly ordering the recipes according to their relevance, with recipes containing the most inputted ingredients being the most relevant. TestFilter.java covers the testing of the filter functionality, such as alphabetical order or a specific tag.
In order to test each of these functionalities, we used the automated JUnit test framework. We used several “sample” recipes or ingredients to observe how they should behave after running certain functions. We then used part of the JUnit API, assert statements, to check whether or not the “sample” recipes or ingredients were behaving as they were supposed to after running each function. The automated nature of the JUnit tests reported whether or not each test failed or not, which indicated whether or not our functions were working properly. I ran these tests by first exporting the classpath: export CLASSPATH=.:./gson.jar:./jetty-all.jar:./json-simple.jar:./mysql-connector-java-8.0.16.jar:./junit-4.8.2.jar. Then, to run each test, I entered these commands at the command line one at a time: java org.junit.runner.JUnitCore TestIng, java org.junit.runner.JUnitCore TestGetIngId, java org.junit.runner.JUnitCore TestRecipes, java org.junit.runner.JUnitCore TestRecipeIds, java org.junit.runner.JUnitCore TestRecipeOrder, java org.junit.runner.JUnitCore TestFilter.

- Front-end: React Enzyme/Jest Testing
  The files used for testing the Front-end are SearchBar.test.js and IngredientSection.test.js. Both located in recipull.github.io -> src -> components. IngredientSection.test.js tests the adding and removal of ingredients to the ingredient list before it is passed to the back-end using inner.HTML. It does this by creating an instance of IngredientSection and then inserting hard coded values into the list with the addIngredient() function and then removing it with either delIngredient() or clearAll(). After each function call, it checks if the state of the object matches the expected result of calling each respective function. SearchBar.test.js tests that the Search Bar functions appropriately returns the user input and clears the bar after it returns the value. It does this by creating an instance of Search Bar and setting the state to a desired input (“Salt” in this case) because the state is suppose to update for every change in the bar, then it triggers a submit to simulate a user pressing “enter”.

  To run these tests, make sure you have the “yarn” command installed (Instructions here: https://yarnpkg.com/en/docs/install#mac-stable). After you have “yarn” installed, call in the command line “yarn test” and then press “a” after it loads to run all tests in the suite. It should take about six seconds to process and then output either passing or failing the tests. Press “q” to exit the testing suite.

- Alexa:
  For testing the Alexa skill we used the Alexa Development Console and AWS CloudWatch. The Alexa Development Console allowed us to interact with an Alexa simulation to check if it was performing the way that it was supposed to. However, it didn’t provide much feedback if there was an error. For more comprehensive insight into the errors we were receiving, we integrated AWS CloudWatch into our project. By allowing the Lambda Function to have access to the CloudWatch logs, we were able to use print statements and other typical debugging strategies in order to find out where our code was breaking. Screenshots of a successful interaction can be seen below.
Here are the recipes that you can make with egg:
- French Toast
- Scrambled Eggs
- Brownies
- Chocolate Chip Cookies
- Upgraded Ramen
- Eggs Over Sweet Potatoes
- Spaghetti Carbonara
- Eggs Baked in Avocado
- Breakfast Skillet
- Lemon Bars
- Chocolate Chip Cookie Cheesecake Bars
- Flourless Almond Butter Cookies

**JSON Input**
```
{  
  "version": "1.0",  
  "session": {  
    "attributes": {},  
    "application": {  
      "applicationId": "amzn1.ask.skill.1843f4b0-df4b-df4b-88f1-84b0-df4b-df4b",  
      "context": {  
        "System": {  
          "applicationId": "amzn1.ask.skill.1843f4b0-df4b-df4b-88f1-84b0-df4b-df4b",  
          "user": {  
            "userId": "amzn1.ask.account.A4F34522103F3A2",  
            "device": {  
              "deviceId": "amzn1.ask.device.064a8eb2-4a68-43d4-8864-4a68-43d4-8864",  
              "supportedInterfaces": {}  
            }  
          }  
        }  
      },  
      "requestId": "1234567890abcdef",  
      "locale": "en-US"  
    }  
  },  
  "outputSpeech": {  
    "type": "SSML",  
    "ssstat": "What are the recipes that you can make with egg?.",  
    "type": "REACTIVATE",  
    "message": "Goodbye!
"  
}  
```

**JSON Output**
```
{"sessionAttributes": {  
  "recipes": [  
    {  
      "name": "French Toast",
      "description": "A popular breakfast dish with eggs.
"  
    },  
    {  
      "name": "Scrambled Eggs",
      "description": "A quick and easy breakfast option.
"  
    },  
    {  
      "name": "Brownies",
      "description": "A favorite dessert with eggs.
"  
    },  
    {  
      "name": "Chocolate Chip Cookies",
      "description": "A classic dessert with eggs.
"  
    },  
    {  
      "name": "Upgraded Ramen",
      "description": "A healthier and tastier ramen dish with eggs.
"  
    },  
    {  
      "name": "Eggs Over Sweet Potatoes",
      "description": "A comforting and delicious breakfast.
"  
    },  
    {  
      "name": "Spaghetti Carbonara",
      "description": "An Italian classic with eggs.
"  
    },  
    {  
      "name": "Eggs Baked in Avocado",
      "description": "A creative and healthy breakfast option.
"  
    },  
    {  
      "name": "Breakfast Skillet",
      "description": "A hearty and flavorful breakfast.
"  
    },  
    {  
      "name": "Lemon Bars",
      "description": "A delicious dessert with eggs.
"  
    },  
    {  
      "name": "Chocolate Chip Cookie Cheesecake Bars",
      "description": "A rich and indulgent dessert.
"  
    },  
    {  
      "name": "Flourless Almond Butter Cookies",
      "description": "A healthy and satisfying dessert.
"  
    }  
  ]
}
```

**Example Output**
```
"recipes": [  
  {  
    "name": "French Toast",
    "description": "A popular breakfast dish with eggs.
"  
  },  
  {  
    "name": "Scrambled Eggs",
    "description": "A quick and easy breakfast option.
"  
  },  
  {  
    "name": "Brownies",
    "description": "A favorite dessert with eggs.
"  
  },  
  {  
    "name": "Chocolate Chip Cookies",
    "description": "A classic dessert with eggs.
"  
  },  
  {  
    "name": "Upgraded Ramen",
    "description": "A healthier and tastier ramen dish with eggs.
"  
  }
]
```
Burndown Charts

- Charts found [here](#)

Appendix

- Technologies employed
  - Java
  - Python
  - JSON
  - React
  - HTML
  - CSS
  - Amazon Web Services
    - Amazon RDS (MySQL)
    - Alexa API
  - JUnit
  - Jest
  - Enzyme