|  |  |
| --- | --- |
|  |  |

**University of Arkansas – CSCE Department**

**Capstone II – Final Report – Fall 2021**

# **The Apron App**

**Kayla Boyd, Bertrand Kalisa, Gregory Renteria, Luis Pinzon, Soroush Shirali, Rachel Culbertson**

## **Abstract**

The complexity of meal planning ranges from ordering fast food and buying ready-to-eat meals to planning out complex meals. In most cases, people plan out what meals they're going to make along with what groceries they need, usually on a weekly basis. This task can end up requiring a significant amount of time and consideration, especially when factoring in diets and fixed budgets. Poor planning also results in wasted food and money and unhealthy diets. Our approach towards simplifying this process is to create an easy-to-use web application that could handle much of the work involved with planning meals. The app can create a grocery list based on meals you have selected in the click of a button and filter out meals based on dietary restrictions saving users time and effort. It can also allow users to enter in what groceries they already have and recommend recipes accordingly, reducing food waste.

## **1.0 Problem**

Meal planning can be a time-consuming and cumbersome task and is often regarded as the worst part of cooking. Many people choose to avoid spending much time planning meals, instead opting for easier methods for obtaining food such as ordering takeout, getting fast food, and eating quick frozen/microwavable meals. This is especially true for college students and young adults who are new to cooking and/or feel they do not have time in between studying, working, and socializing. Although fine in moderation, these methods can become problematic in terms of receiving adequate nutrition and high costs.

This project also addresses the issue of food waste. It is estimated that in the US, up to 40% of food produced goes to waste. US households alone waste approximately 76 billion pounds of food per year, accounting for 21 percent of all food waste. [1] Although wasting food may seem like a trivial issue to some, that waste can quickly accumulate leading to adverse effects on the environment including greenhouse gas emissions and excess water usage. A lot of food is wasted due to poor planning when buying ingredients for meals. In the average American household, many ingredients end up expired before they are used, and portions of meals are often thrown away because the portion sizes are too large.

Additionally, a significant portion of the population has at least one type of dietary restriction. According to a study on special diets by the NCHS, around 17.1% of adults aged 20 and up were following a special diet at any given day in 2015-2018. [2] This study asked about health or weight loss related diets, excluding diets for ethical or religious reasons. Therefore, an even larger percentage of the population has specific dietary requirements and that number is estimated to grow. People with dietary restrictions need to take extra care when planning what meals to eat and may have additional requirements related to their diet. For example, someone on a vegan diet may also have a fixed budget for their groceries, further restricting their options for food. These details highlight a growing need for resources for planning meals.

## **2.0 Objective**

The objective of this project was to create a web application to assist users in meal planning. This app supports a variety of features including login/sign up authentication, saving favorite meals, profile management, recommending meals based on ingredients the user already has, and a calendar. The project addresses the issues of time management, food waste, and dietary restrictions related to meal planning.

## **3.0 Background**

### **3.1 Key Concepts**

In this section, we discuss some of the technologies that we used for our application’s construction. These technologies were selected considering each team members' knowledge or suggestions about languages, databases, frameworks, and APIs that are easy to work with. Therefore, the key components used in this web application are based on software needs such as tools for frontend/backend development, databases, website frameworks, APIs, source control, and collaboration project boards.

In terms of the frontend development, our decision was to use React because this technology will facilitate for us the creation of a web app which can synchronize with databases for data interaction and at the same time being fast, scalable, and simple.

For our database, we thought that MongoDB would be a great option to use thanks to the flexible document schemas it offers. After reading some basic documentation, we will learn how to model and manipulate data using any kind of data structure. Also, MongoDB is code-native, which will allow us to access it from the language of our preference.

Considering database interactions and calculations, we chose NodeJS to be our main technology for backend development. This is a great option because of the experience we already have working with OOP languages. Its simplicity will be beneficial to web development cross-platform having the ability to scale up quickly.

In order to have more tools and resources at our disposal, we will use the Express framework. Since it only requires JavaScript knowledge, we can easily build different kinds of web components in a short amount of time. Moreover, it provides a simple routing for requests made by clients.

After looking up some technologies available for recipe/calendar APIs, we found interesting options. For example, for food/recipes we have APIs such as Spoonacular, Nutritionix or Edamam some of them being free to use. At the moment we do not have a specific recipe API that we will use. However, we know that the main use of the API will be to establish matching relationships between the ingredients we have available with the recipe options the tool recommends. For calendar APIs, we chose Google Calendar which is free to use and allows us to edit meal planning events, set reminders, or simply complete tasks.

Because of our previous experience in classes such as Software Engineering, we agreed on using Github as our source control platform. This tool simplifies the process of working with other people and makes it easy to collaborate on projects. Our team members can work on files and easily merge their changes in with the master branch of the project.

Finally, for our project collaboration tool, we used Trello. It is a visual work management tool that helped us ideate, plan, manage, and accomplish tasks together in a collaborative, productive, and organized way. We created a board where we had a series of lists, with a bunch of cards attached categorized by completion levels (To do, Doing, Done). Moreover, these cards were labeled with a team member to their assigned task.

Deployment:Amazon Web Service(aws) was the cloud computing network we used to host our dynamic web application built using MERN Full-Stack protocol. This came at a divergence from Google Cloud Platform(gcp) as cost for their Databases and system was extremely expensive compared with aws. We had to create a PR branch on github which contained simple clean up to help practice DevOps. Using the AWS console I realised the entire service was down(AWS Technical Support in us-east-1,which was down) so nobody could open new support tickets!\*(yikes!). We used Docker containers running on EC2 instances instead of AWS Fargate, which we will dive into in more detail in the architecture section. Main reason though was for its own implementation of a fully self managed server compared with auto managed by aws.

### **3.2 Related Work**

During some of our team meetings, we discussed how authentic our idea needed to be, since we did not have a clear goal for our website. Therefore, we came up with the idea of implementing a web app for giving people ideas about what to cook with what they already have in their houses. However, since there were several websites with a somewhat similar goal, it was a clear indication that we had to change our approach slightly.

We found that some developers created websites in a blog style to feature delicious, healthy, family-friendly recipes. For instance, the blog website *Skinnytaste* of the recipe developer Gina Homolka began in 2008 with the idea of presenting recipes anyone is able to cook to improve people’s self-esteem or just to spend valuable time with other members of the family in the kitchen. [3] At Skinnytaste.com, we can find different types of dishes for people hungry and eager to learn how to cook on a daily basis.

Even though this website is similar to the website theme we have, it is just a blog with several sections classified by special diets, travel meals, and cookbooks, where Homolka can publish her own experience after cooking. Definitely, we improved this idea with our website since it contains several other features than just recipes with brief descriptions.

Another example found was the website *Recipeat*, which contains multiple functionalities similar to the ones we have in our website such as having a sign up/sign in page, profile page, and recipe generator based on ingredients available. [4] The way how *Recipeat* works is by first creating a profile to become part of the community. Then, after users check what ingredients are available in the kitchen, they list them on the website for it to generate a recipe. Another feature that this website supports is that it can synchronize with social media such as Facebook, Twitter, or Instagram.

We thought that we could use this website as a reference for our idea since we like how the developers created it. However, with our website we have implemented some other extra features like a calendar to organize and save food/recipes for specific days of the week.

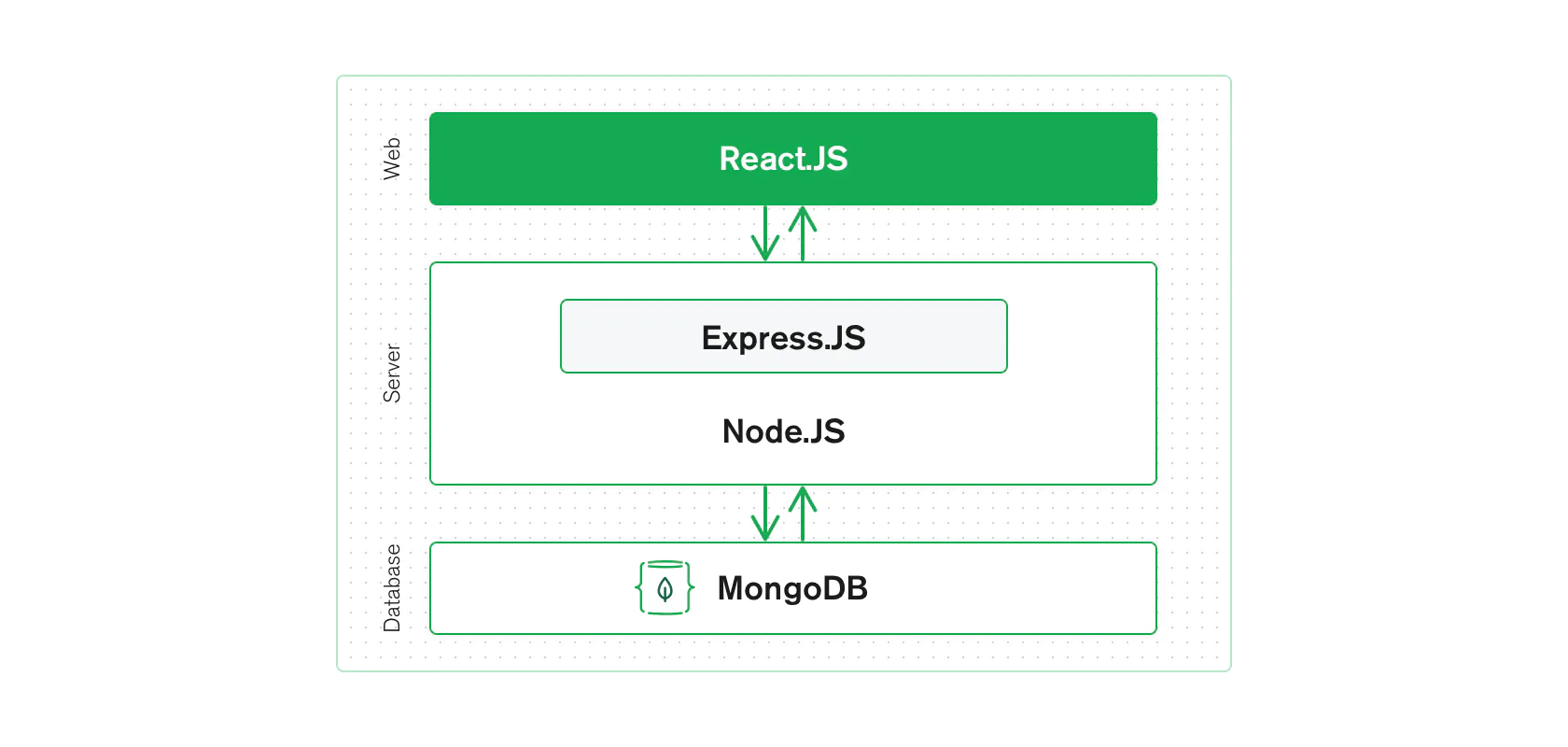
## **4.0 Design**

### **4.1 Requirements and/or Use Cases and/or Design Goals**

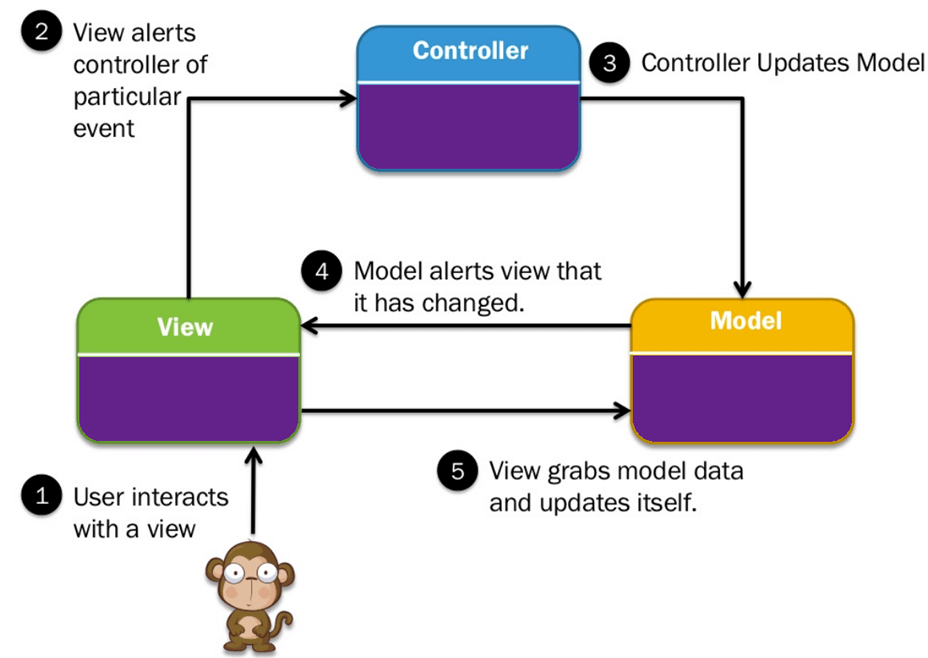
### For our project we provided a list of requirements in our proposal and mostly stuck to those initial requirements as we developed the Apron App. We divided recipes into different categories based on the user's food requirements such as recipes that are vegan, vegetarian, sugar free, gluten free, and dairy free. We were able to provide more food options than we initially listed in our proposal. We also made a requirement to use a navigation bar that will hold the different pages for our website. We needed to use a calendar api for the calendar feature which allows the user to schedule different meals throughout the month for breakfast, lunch, or dinner. Another requirement for this application was to have a pantry feature as part of our design, that will help users save whatever ingredients they have with them at home and add or remove them accordingly. Additionally, we needed a ‘favorites’ feature where users will be able to save their favorite recipes and easily view/ access them from the save feature. This is implemented under the “Saved Recipes” tab. We added more requirements along the way including a profile for each user and grocery lists. The profile allows the user to change their profile picture and add a short bio, and the grocery lists allow for the user to create as many lists as they need.

### **4.2 Detailed Architecture**

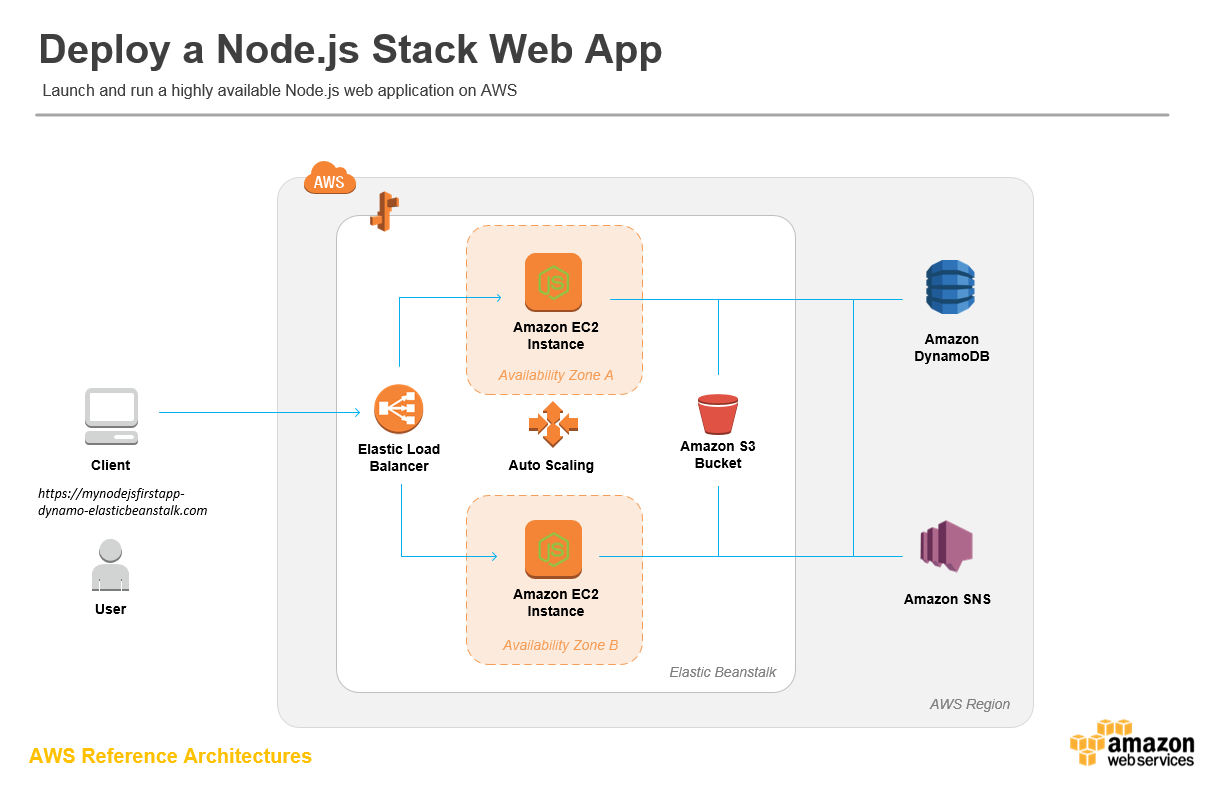
The MERN stack is a full-stack development protocol, which is an industry standard created by the teams at facebook, for dynamic web applications acronym: Mongo, Express, React and Node. These allow client side rendering while all being based in javascript. Mongo is the database in JSON form, express it supports the backend framework built on top of node.js, react is the front end framework for UI/UX, and Node is the javascript run-time environment. [5]



Our project utilized the Model-View-Controller (MVC) architecture pattern. MVC divides an application into three components: Model, View, and Controller that each handles a specific development aspect of an application. [6] The Model handles the app’s data storage/management and its main logic. It works independently of the User Interface. The Controller provides an interface between the Model and View and is used to update the Model based on user interaction. The View defines how the information from the model will be displayed and how the user can interact with the application. MVC results in a separation of logic between different components, allowing for easier development and debugging.



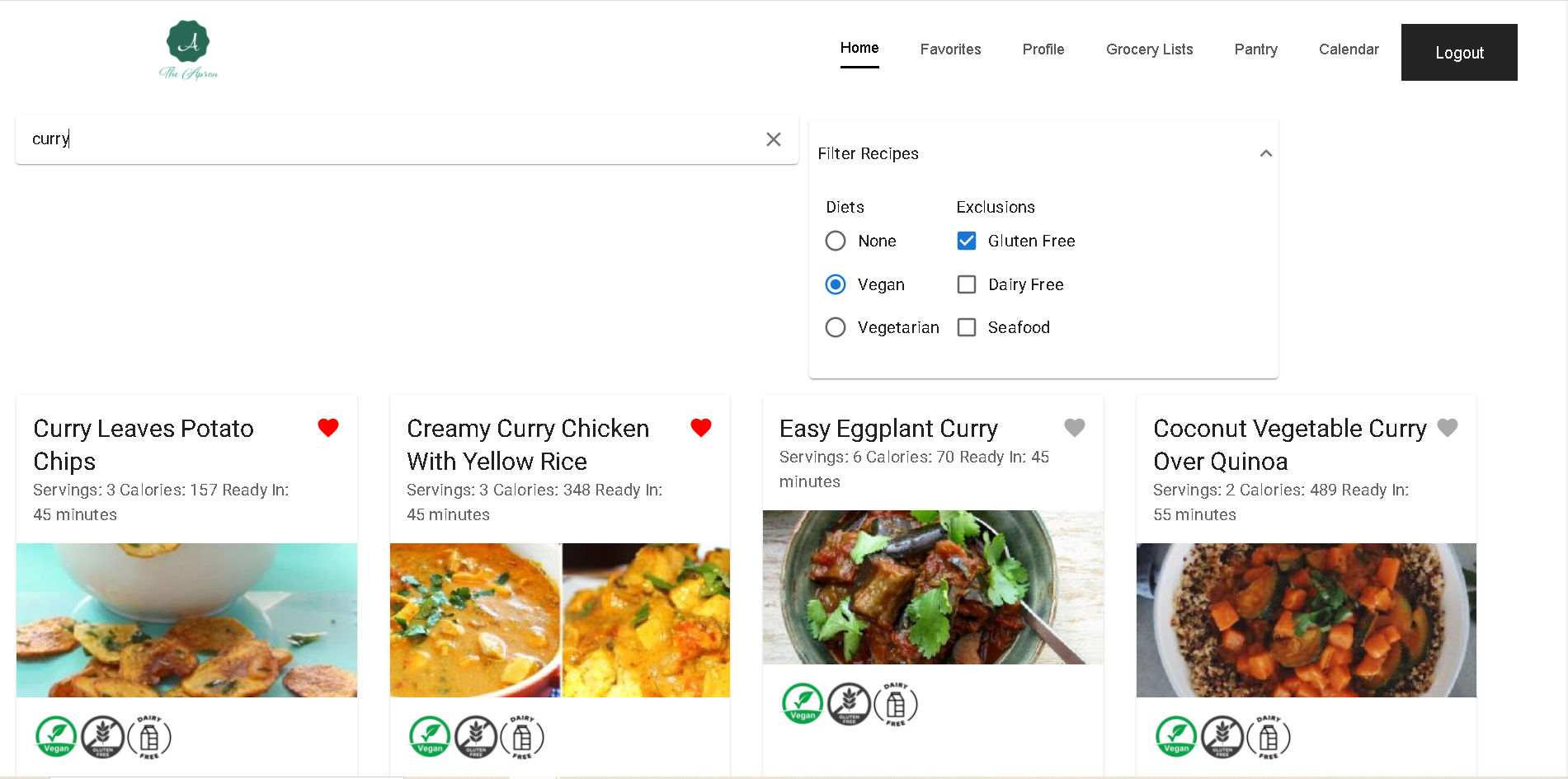
For our project particularly, the View consists of a number of React components that together create the main pages of the application. The View also has css files for styling and a root HTML file that contains the app’s content. We used the Material-UI library extensively in our code for designing the User Interface. [7] The components handle user input and interaction such as clicking on the heart icon to save a recipe or entering a query into the home page’s search bar. Components can also update and retrieve data from the application state through React Redux. For example, when a user is authenticated their unique user id is stored in the Redux store so that it can easily be retrieved later for user-specific backend calls. Redux is also used for saving recipes that have been selected to be aggregated into a single grocery list. There are seven pages that have separate routes and compose our website: Home, Profile, Calendar, Pantry, Grocery Lists, Sign-In, and Sign-Up. These pages contain other components such as the recipe cards that are displayed on the home page. Building separate components is a core feature of React and allows for more modularity and readability when developing applications. In the views, there is no direct accessing/updating the database. Instead, the views call the backend Node.js server to perform operations on the database. This is done through the Axios React library, a promised-based HTTP client for sending asynchronous requests to a Rest API. [8] There are seven pages that have separate routes and comprise our website: Home, Pantry, Grocery Lists, Profile, Calendar, Sign-In, and Sign-Up.

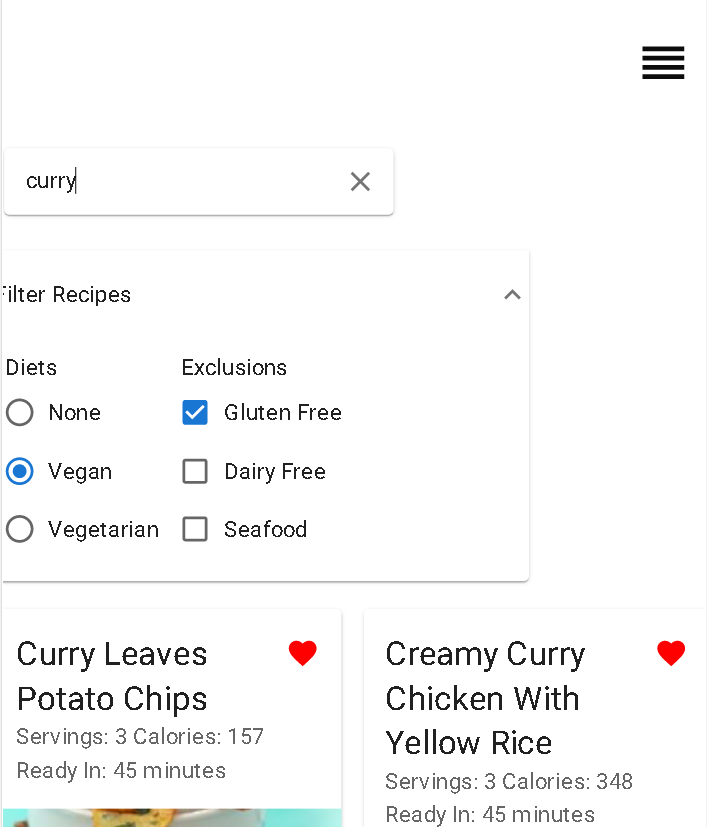


Deployment: Amazon Web Services as the choice host. Heroku/Firebase and some others were not used as they are not industry standard. The main three: Amazon Web Services, Microsoft Azure, and Google Cloud Platform. AWS did have some learning curve with all the configurations/options along with it being a premium service. Azure was the last choice as it is a nightmare to deploy containerized applications(DOCKER/NGINX). Google Cloud is a newer way of deploying your applications, and has documentation with references. Though expensive on the database side with less UI/UX compared with AWS. Now let us talk about the containers which we mentioned above. AWS Fartgate is an auto managed server once you provide the definitions, but you can not control events relating to: server maintenance window, subnet configurations, VPC Replication and much more. We chose to go with AWS EC2 fully self managed servers. In the event something relative to yesterday’s AWS outage we can theoretically configure a multi-region failover, which would spin up new containers in say us-west-2 if things went down in us-east-1. Also, the server images are publicly available (OS running on servers). Most people just grab the latest and run with it, which is fine for most cases. If you are dealing with a major corporation the better option may be to copy the image onto your own private image and start from there; So when hackers find a weakness to the OS and try to exploit all of them on AWS your server will not be picked up since it is not “publicly available”.

1. **Home Page**

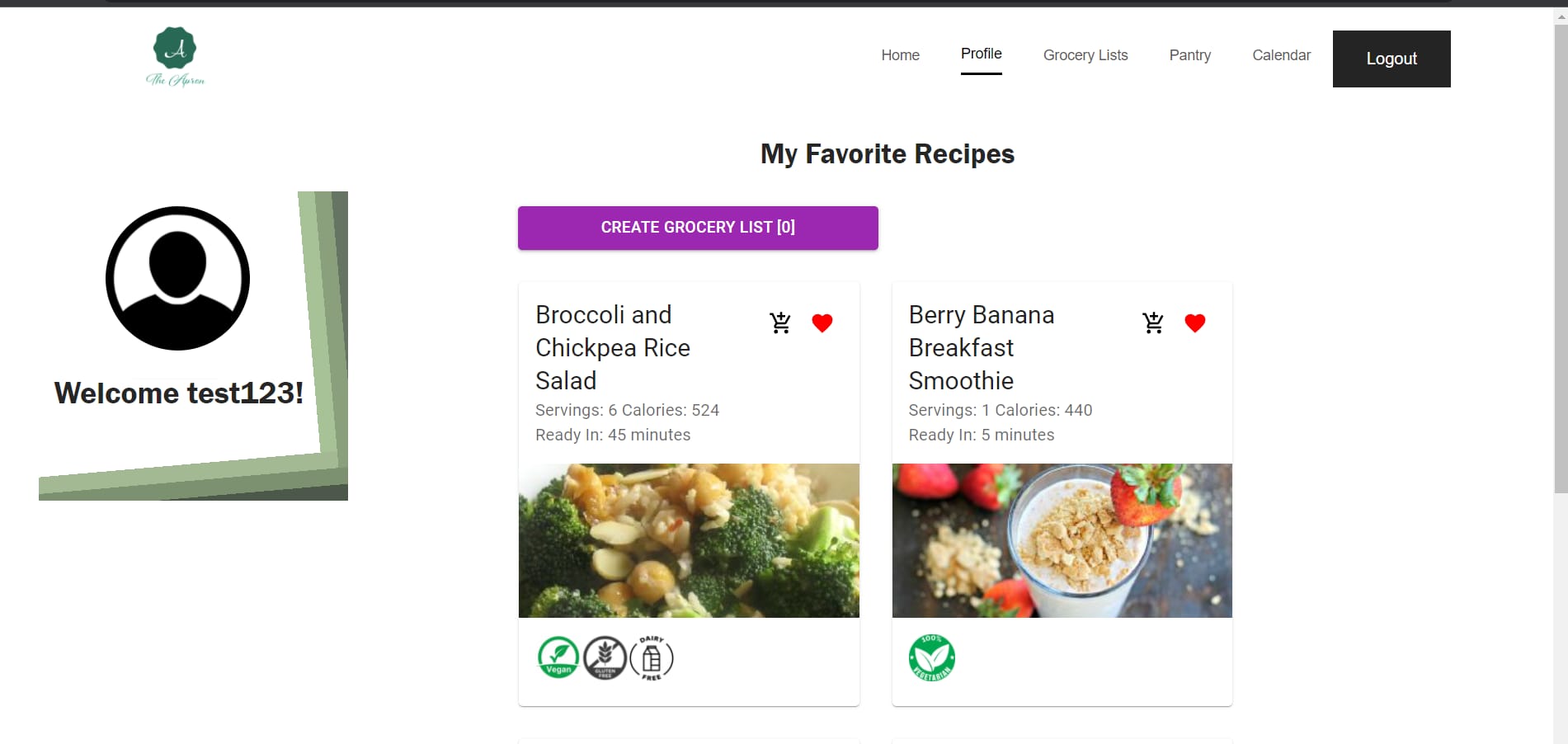
The home page is the page that is initially displayed when the app is started. At first, it shows two-hundred recipes that have been randomly selected from the Spoonacular API. Twenty recipes are shown on each page and pages can be changed through a page handler on the bottom of the page. Each recipe is displayed in a card component that is defined externally. If an individual card is clicked, additional recipe information is displayed including a table for the ingredients and a list of instructions. The user can also click on the heart icon in a recipe card to add the recipe to their favorites. Clicking again will remove the recipe and the heart’s color will change accordingly. If the user is authenticated, a search bar and filter menu will appear so they can search through a wider range of recipes. In the screenshot below, the user has selected vegan and gluten free recipes that match the search query “curry”. A navigation menu is displayed throughout every page of the app so that the user can easily navigate to different pages. If the user is not authenticated, only the home, sign in, and sign up tabs will appear. The navigation menu is also an example of responsive design, since it is replaced by a hamburger menu for smaller screen sizes

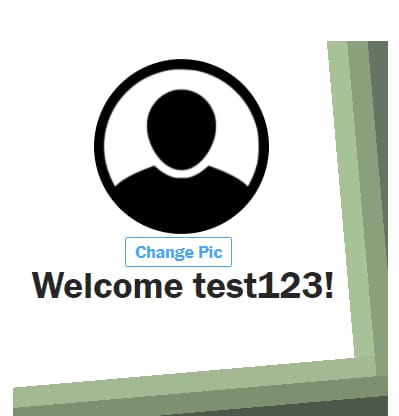


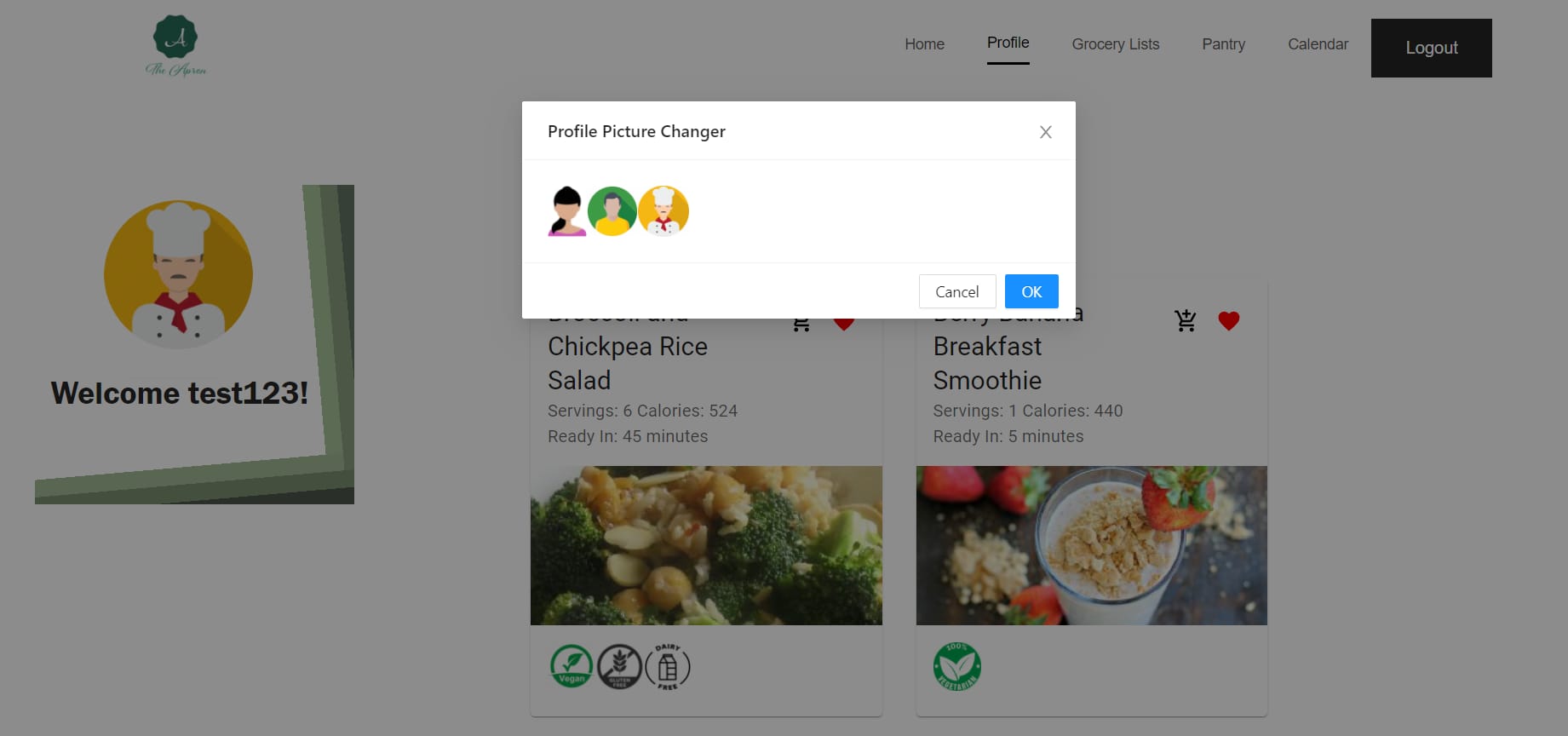


1. **Profile Page**

As in every website design, a user profile page is important to maintain generic user data. In our application design, we decided to include components such as welcoming the user after logging in, a personalizable user image page that can be modified by using an image modal component, and a display of all the favorite recipe cards selected by the user. This page also contains a create action button that creates a grocery list containing all the ingredients needed to cook the respective recipes and such list can be found in the ‘Grocery Lists’ tab.

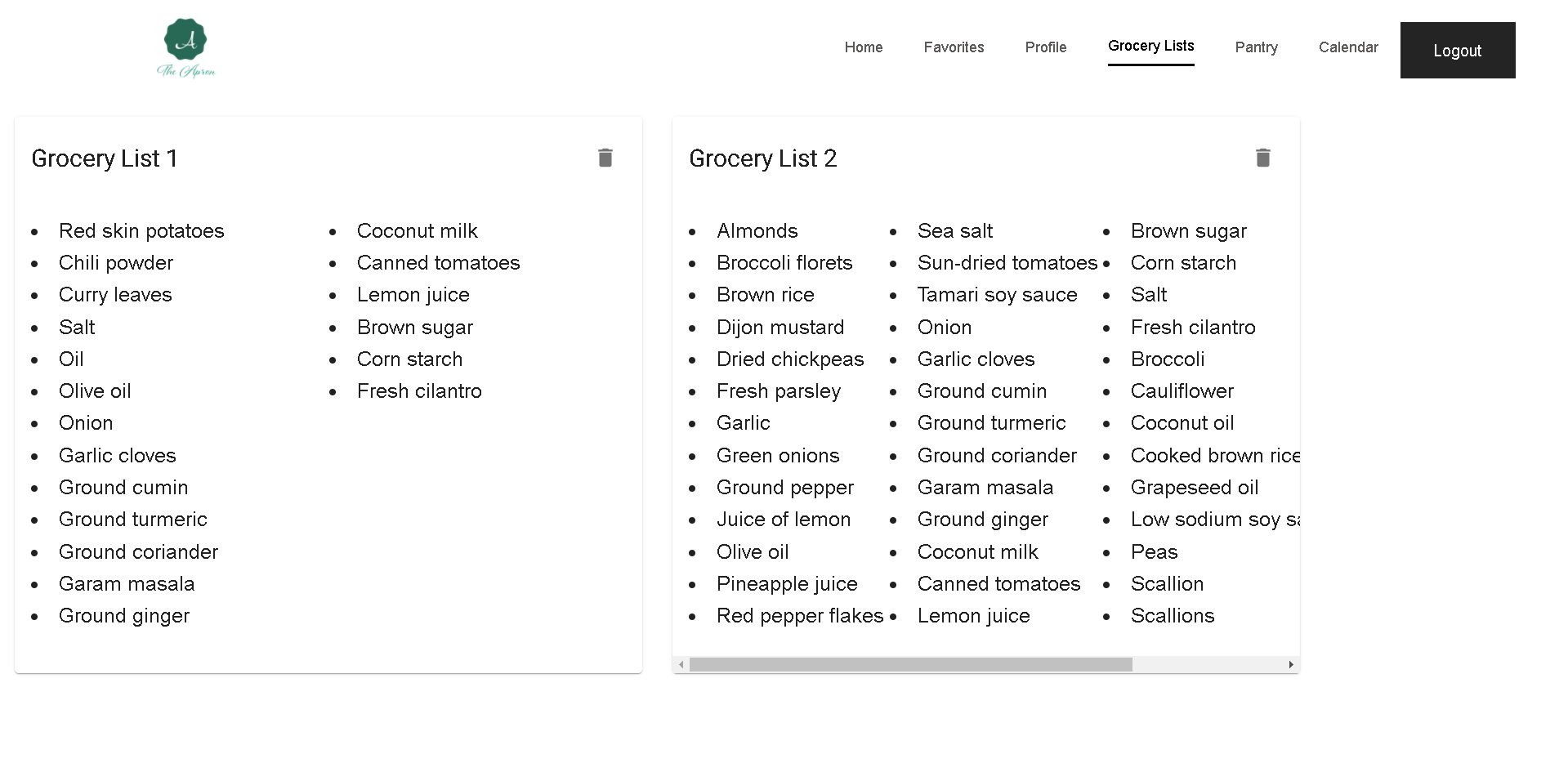






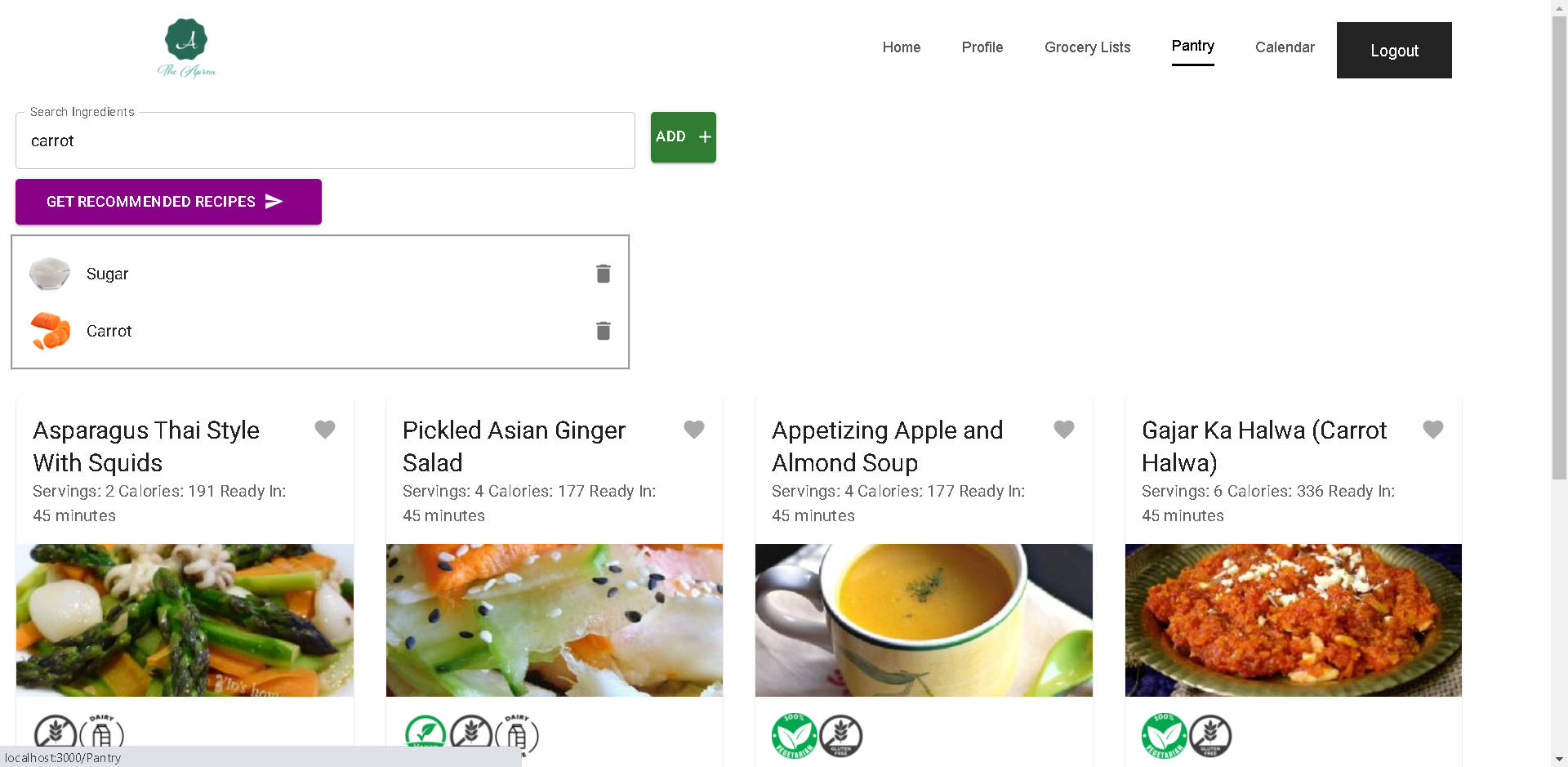
1. **Grocery Lists**

This page contains the grocery lists that the user has created from the saved recipes. Every list is given an automatically created name based on the order in which it was created (e.g. “Grocery List 2” for the second grocery list). The ingredients are shown in a bullet point separated list and a horizontal scrollbar is added for lists that exceed a maximum defined width. A list can be deleted by clicking on the trash icon in the upper right hand corner and the UI is automatically updated.



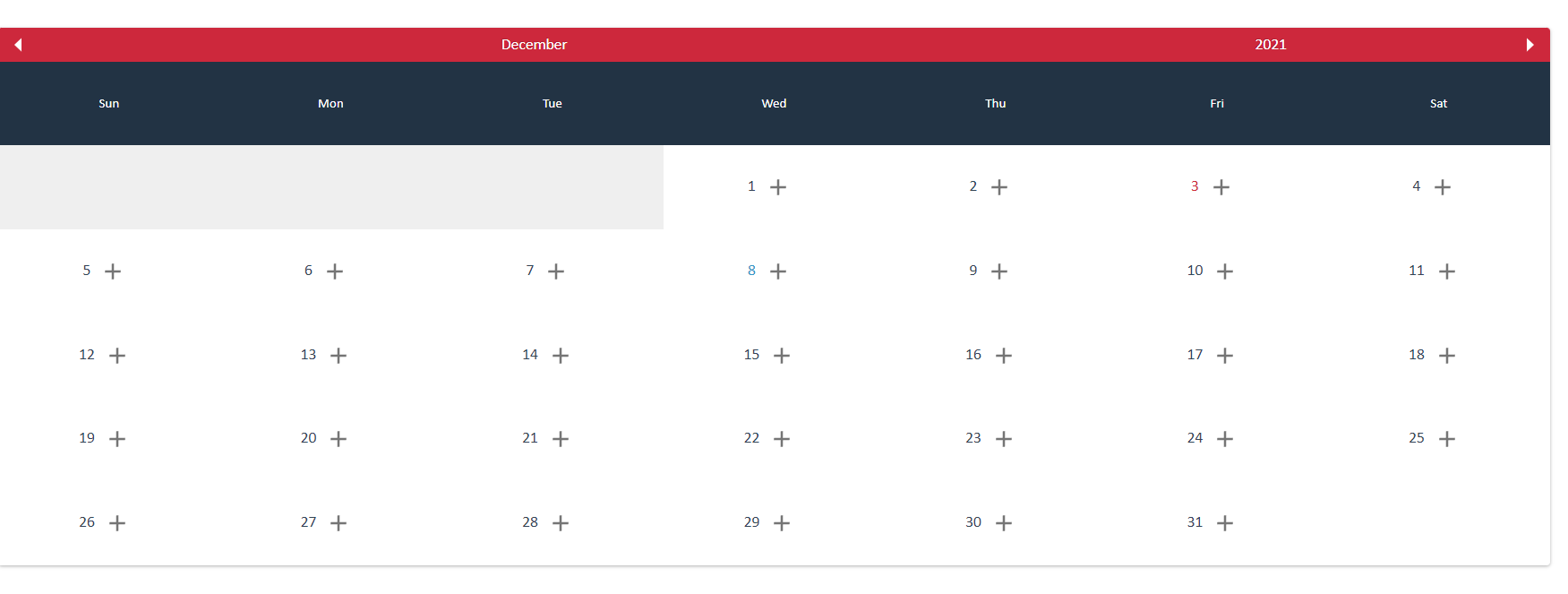
1. **Pantry**

The primary purpose of the pantry page is to allow the user to add the ingredients they currently have and then use those ingredients to get recommended recipes from the API. For input validation, the ingredient search bar displays a list of up to five valid ingredients that match the user’s current search query. An ingredient can only be added if it matches one of the results in the query. Ingredients are displayed in a list with their image, name, and an icon for removing the ingredient from the list. The user can get recommended recipes based on what they’ve added by clicking the “Get Recommended Recipes” button.



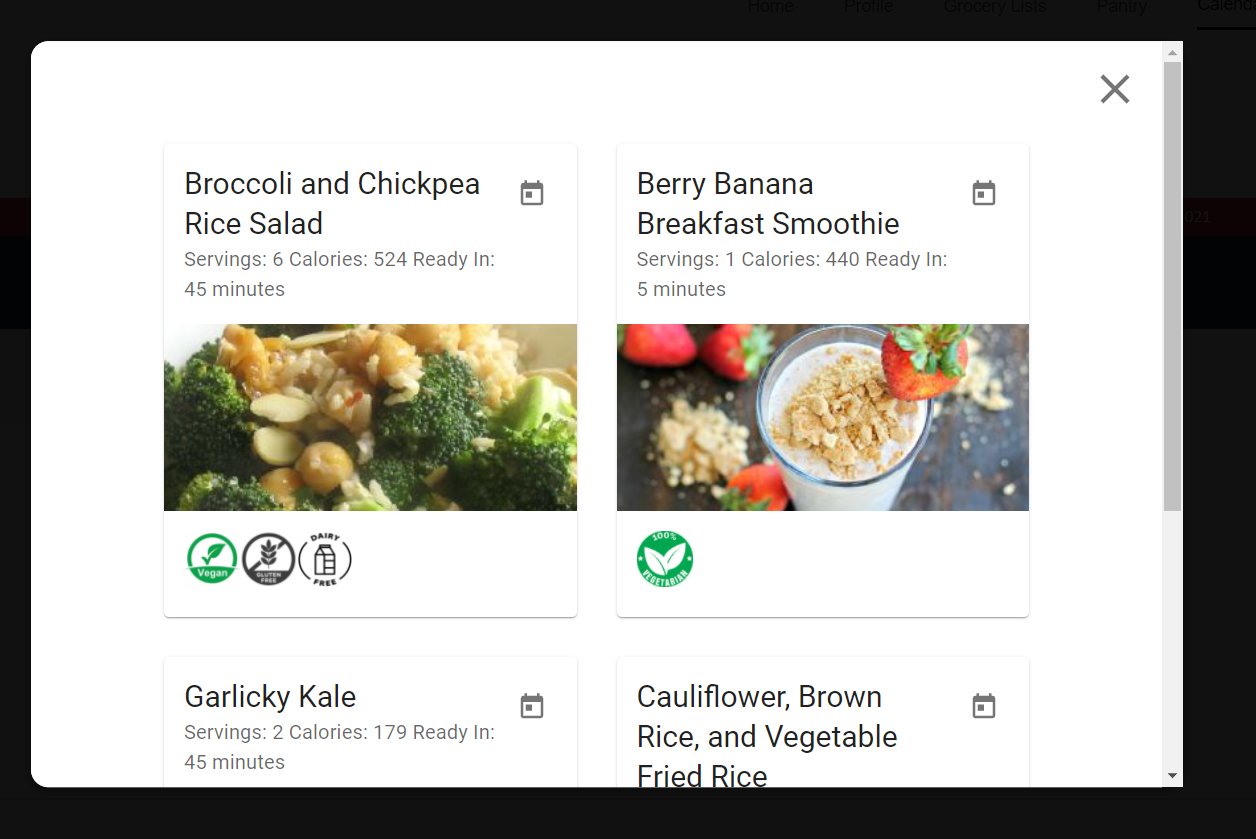
1. **Calendar**

The calendar provides the user the ability to check the recipes that he/she has saved as favorite. The only recipes that can be added to the calendar are the ones that the user has marked as favorite. This is implemented using an array of saved recipes which are added when the boolean value of favorite in the calendar changes from false to true.



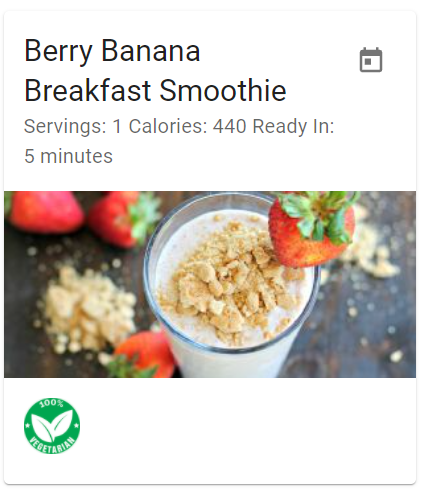
One important thing to mention before explaining how the calendar works is that we need to create an account and be signed in in order to see the calendar tab.

How does the calendar work? In order for the user to use the calendar, the user can click on the plus buttons which are located in the cells of the calendar. This will display a modal that has all the favorite recipes for that specific user. The modal looks as in the following picture

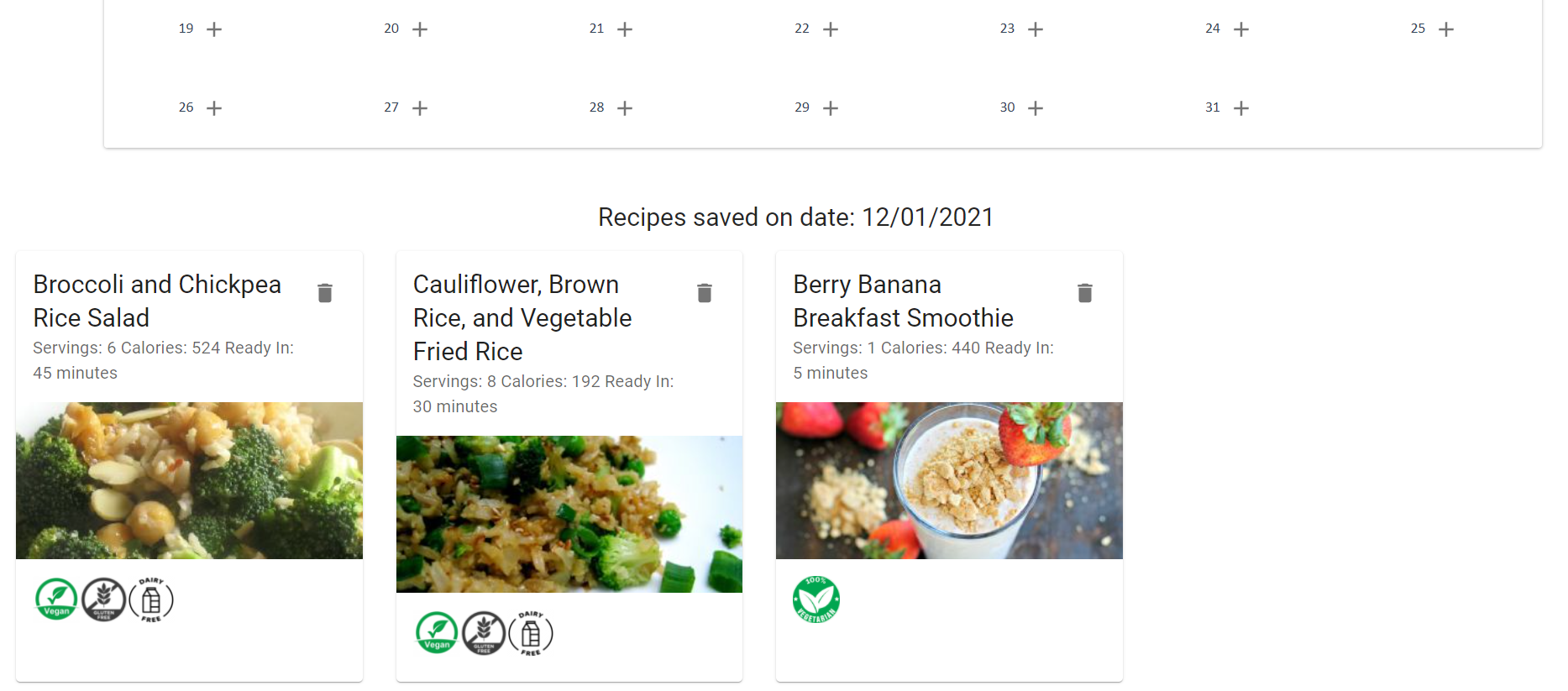


Inside this modal, there are recipe cards which provide the user with a little calendar icon.When the user clicks on this calendar icon, he can add or save the recipe to that specific date.

For example, from the modal above, if we want to add the Berry Banana Breakfast Smoothie, we click on the button in the right-top corner and it will be automatically add that recipe to the calendar.



Also, in order to see the saved recipes on the calendar, we can click on the day to see the recipes that we have saved. This will pop out the list of recipes that have saved on that date for that specific user. For example, for December the first, the testing user [test123@test.com](mailto:test123@test123.com) has 3 recipes that he would like to cook on that date. When he clicks on December the 1st , the following recipes will pop out below the calendar.

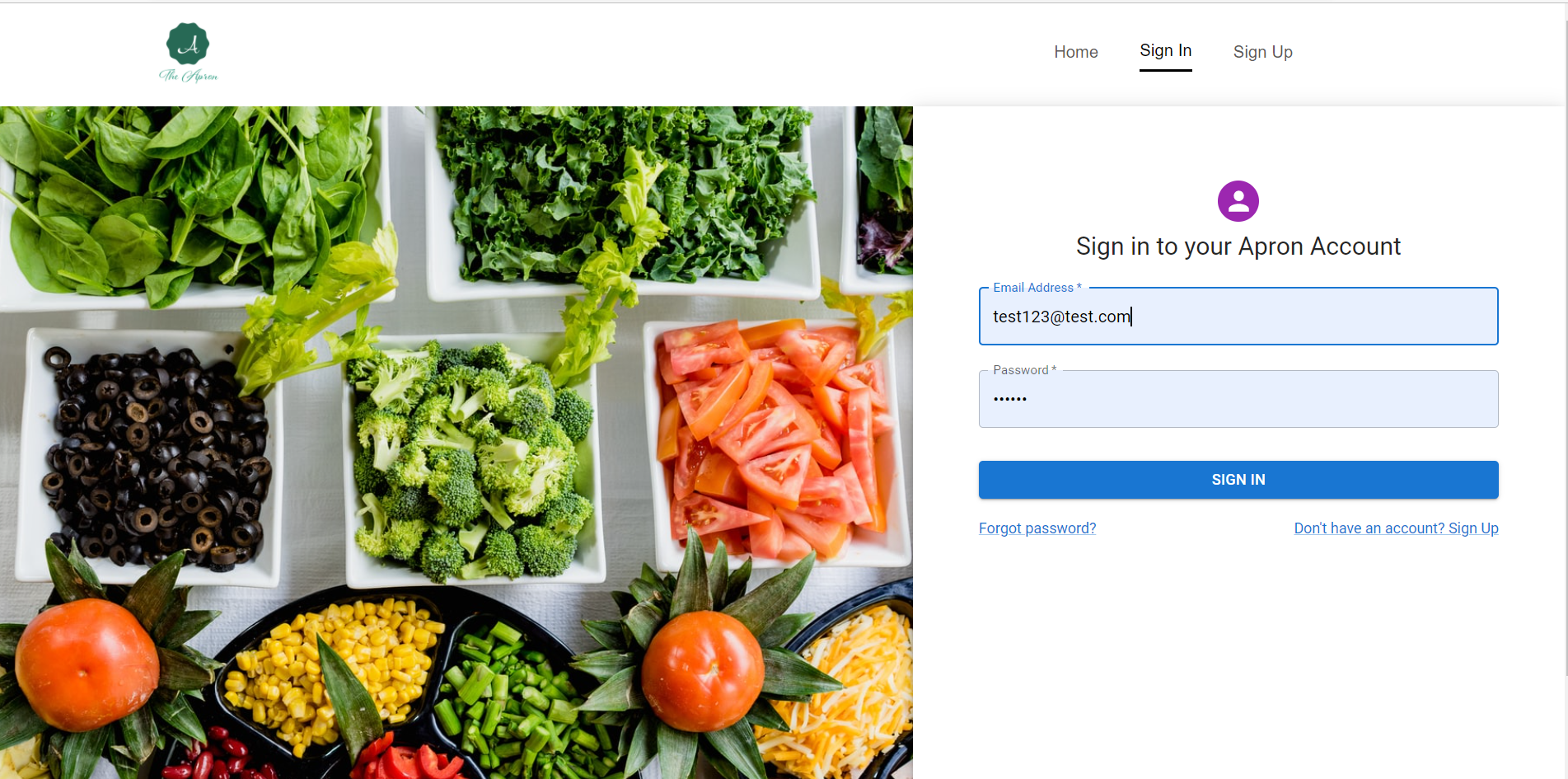


Finally, the saved recipes cards will have a little delete icon button on the top right of the cards. This delete button allows the user to delete the recipes from the calendar.

**Sign-In/Sign-up**

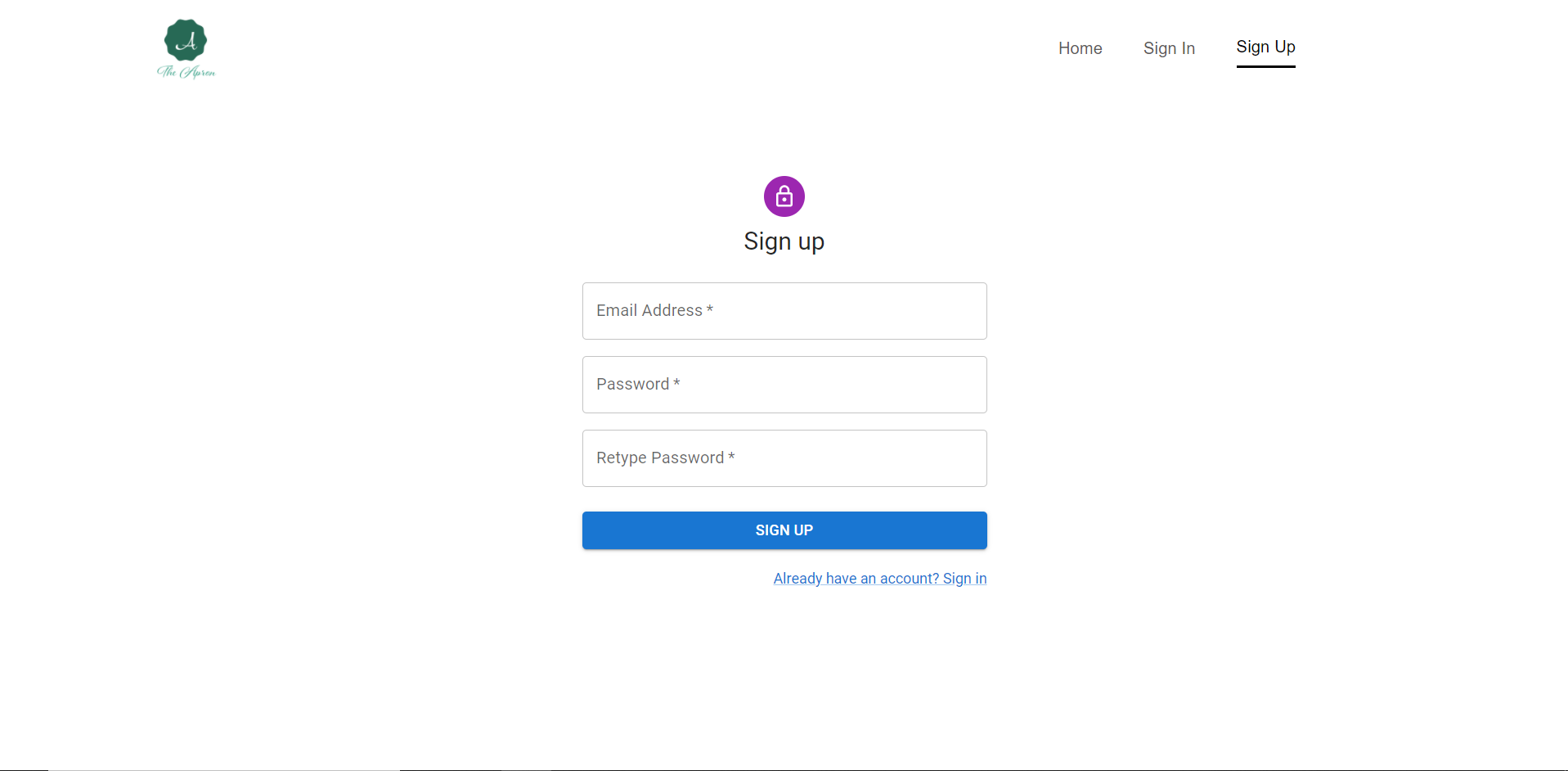
Sign-In

This page is located in the Navigation Bar when the user is signed in. After you click that button, you are brought to this page (see below). The user can enter in their credentials if they have already created an account through the Sign-Up page. For testing purposes we created a user with the email as test123@test.com and the password as 123456. You can use this login or create your own to use the other features. Under the Sign-In inputs there is also a link that will take you to an account if you have not created one yet.



Sign-Up

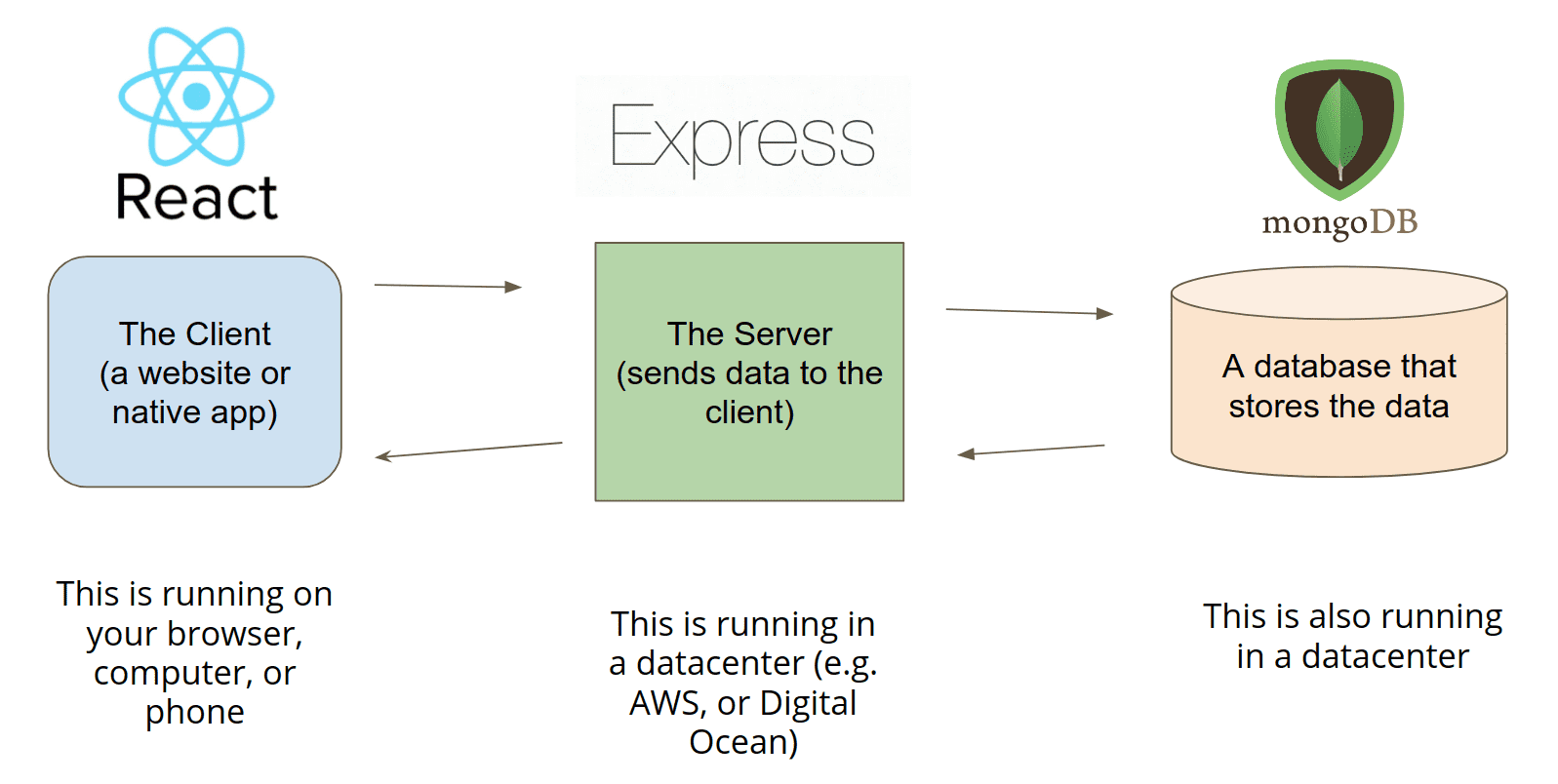
The Sign-Up page prompts the user to enter their email and their password twice to make the page more user friendly. When you are signed in, you have a lot more features available to you such as the pantry, grocery lists, profile page, etc.



The controllers for our project are defined in the server-side code. The controller functions are called based on the route and type of requests sent by the client. For extra security purposes, we also defined a middleware function that will check if the user is authenticated before calling certain functions such as saving a recipe. Controllers take in the body of the request sent by the client and use that information to query the database or send a request to the Spoonacular API. We utilized Mongoose to interact with our MongoDB database. [9]

Our model consists of five schemas that are defined in NodeJs through the Mongoose extension. These schemas are recipe, user, ingredients, grocery list, and calendar. The recipe schema stores information about the recipes saved such as the recipe’s name, summary, instructions, ingredients, and nutritional information like calories and serving size. The ingredients schema stores an array of ingredients that the user has added in the pantry with their name and image. User contains a user’s email, encrypted password, image, and their automatically generated UUID. The calendar schema contains an array of recipes that have a corresponding date that were added to. Finally, grocery list stores an array of grocery lists that have a unique name and an array of ingredients. Each schema also contains a userId for uniquely identifying a specific user’s information.

Seen below is a high level overview of our application architecture.



### **4.3 Risks**

| **Risk** | **Risk Reduction** |
| --- | --- |
| Enabling Proper Encryption for database regarding passwords/user information | We used a facebook standard for creating a dynamic web page (MERN Stack) and handled all user encryption using SHA-256 Hash/SALT to allow for best industry standard encryption. |
| Scalability when dealing with an influx of client requests | By using Mongo we have the built in scalability, along with running on multiple servers to ensure better probability of losing crucial data. |
| Testing a vast majority of different OS/Device intercompatibility to be best optimized for all users. | We used a responsive web which allowed us to adjust website content into the best layout for the device it is being displayed on. Creating a responsive web page was crucial for making the Apron App a user friendly experience. |
| Authentication | Users are authenticated via JSON Web Tokens (JWTs) with a secret key. Tokens expire within 24hrs for additional security. |

### **4.4 Tasks**

1. Work on preliminary proposal and presentation

* Identify the concepts and objective of the projects
* Understand the goals and obstacle of future users
* Write initial proposal
* Delivery proposal presentation

2. Initial app planning

* Work on the wireframe for each of the screens of the app:

-Discuss UI design pattern and color palette

- Create a user-flow diagram and discuss navigation for each screen.

- Discuss improvement in the layout of the app

- Balance simplicity vs usefulness

* Make final decisions on what features the app will have
* Make final decision on technologies which is best suited for the project
* Create mockup of website
* Define architecture
* Gather recipes (Research APIs for finding online recipes)

3. Setup for environment

* Create a Github repository for the team.
* Create and design MongoDB schema, beginning with user authentication.
* Become familiar with MERN architecture and install all needed technologies
* Download any needed dependencies in app directory

4. Development

* Create and design MongoDB schema for storing user data.
* Set up database
* Implement API for user authentication
* Design and create UI and implement login and sign up pages
* Design and implement Homepage
  + Retrieve and display recipes
  + Implement recipe save feature
* Design and Implement Sign In/Sign up
* Design and Implement Pantry
  + Search bar with autocomplete feature for valid ingredients
  + Display ingredient name and picture with a delete button
  + Retrieve and show recommended recipes from the API
* Design and Implement Saved Recipes Page
  + Take liked recipes from Homepage and put them in a centralized location
* Design and Implement Profile
  + Create option for profile picture
  + Allow user to edit bio
* Create Calendar Page
  + Displays a stylized calendar
  + Clicking on a day shows recipes added for that date
  + User can add a recipe for a selected date

5. Deployment of the app

6. Finish fine tuning the website and testing use cases

* Perform functional testing
* Perform performance testing

7. Final report and presentation

* Record demo of our site

8. Project website and poster

### **4.5 Schedule**

| **Tasks** | **Dates** |
| --- | --- |
| 1. Work on preliminary proposal and presentation | 8/23-9/13 |
| ● Identify the concepts and objective of the project | 8/23-8/27 |
| ● Understand the goals and obstacle of future users | 8/30-9/3 |
| ● Write initial proposal | 9/7-910 |
| ● Deliver proposal presentation | 9/10-9/15 |
| 2. Initial app planning | 9/17-9/24 |
| ● Work on the wireframe for each of the screens of the app:  -Discuss UI design pattern and color palette  - Create a user-flow diagram and discuss navigation for each screen.  - Discuss improvement in the layout of the app  - Balance simplicity vs usefulness | 9/17-9/20 |
| ● Make final decisions on what features the app will have | 9/20-9/22 |
| ● Make final decision on technologies which is best suited for the project | 9/20-9/22 |
| ● Gather recipes (Research APIs for finding online recipes) | 9/22-9/24 |
| ● Define architecture | 9/22-9/24 |
| ● Have final mockup | 9/24-9/27 |
| 3. Setup for development | 9/27-10/1 |
| ● Create a Github repository for the team. | 9/27-9/29 |
| ● Become familiar with MERN architecture. | 9/27-10/1 |
| 4. Development | 10/1-11/15 |
| ● Create login/sign-up UI | 10/1-10/6 |
| ● Design navigation tabs for various future app components | 10/6-10/9 |
| ● Create and design MongoDB schema for storing user data. | 10/9-10/10 |
| ● Design and create UI for login and sign up pages with ReactJS | 10/10-10/15 |
| ● Implement API for user authentication | 10/-10/15 |
| ● Retrieve and display recipes on home page | 10/15-10/20 |
| ● Add search bar and filter features for the home page | 10/20-10/25 |
| ● Add saved recipes feature | 10/25-10/30 |
| ● Create user profile page with option to change picture | 10/20-10/28 |
| ● Implement pantry UI to save ingredients and recommend recipes based on current ingredients | 10/28-11/2 |
| ● Setup backend models and controllers for saved recipes and ingredients | 11/2-11/5 |
| ● Design Calendar page | 11/5-11/10 |
| ● Add logic for user input for the Calendar page | 11/10-11/14 |
| ● Setup backend model for saving recipes for a specific calendar date | 11/14-11/18 |
| ● Modify saved recipes page to also save a recipe to be added to a grocery list | 11/18-11/20 |
| ● Grocery List backend model and logic for aggregating ingredients to create a list | 11/20-11/24 |
| ● Connecting profile image to database and adding pre selected images to be chosen from | 11/24-11/28 |
| ● Finish implementing Calendar page | 11/28-11/30 |
| 5. Deploy app | 11/30-12/4 |
| 6.Finish fine tuning the website and testing use cases | 12/4-12/8 |
| ● Perform functional testing  ● Perform performance testing |  |
| 7. Final report and presentation | 12/8 |

### **4.6 Deliverables**

* Recorded Demo: Will show the user how to find recipes based off of the items they have in their pantry along with other features including: a login/signup page, home page/feed, profile, pantry, calendar, saved recipes, and navigation tabs. The demo will also display all features that the Apron App has.
* Poster: Will be a reflection of our report and will be displayed on our group website.
* Apron App code: A link to our GitHub repository will be provided
* Final Report

## **5.0 Key Personnel**

**Rachel Culbertson** – Culbertson is a senior Computer Science major at the University of Arkansas. She has completed courses in Computer Graphics, Database Management Systems, and Artificial Intelligence. In the summer of 2021, she worked as an IT Intern for Tyson. She currently works as a Research Assistant in the Data Science lab.

**Kayla Boyd** - Kayla is a senior Computer Science major at the University of Arkansas. She has completed courses in Big Data Analytics and Management and Algorithms. Since the summer of 2021, she has been an Application Development intern in the Engineering and Technology Department at JBHunt. She also has experience in front end web development.

**Gregory Renteria** - Gregory is a senior double majoring in Computer Science and Computer Engineering at the University of Arkansas. He has completed courses in Computer Graphics, Database Management Systems, and Introduction to Game Design I/II. He is currently interning as a software developer at Junction AI, where he is in charge of the data ingestion pipeline using AWS environments.

**Luis Pinzon**- Luis is a senior double majoring in Computer Science and mathematics at the University of Arkansas. He has completed courses in computer networks, algorithms and Database Management Systems. Since the Summer of 2021, he is part of the Network Enterprise Team at the University of Arkansas as network technician.

**Bertrand Kalisa-** Bertrand is a senior majoring in Computer Science at the University of Arkansas. He has completed courses in Database Management Systems, Computer Networks, Algorithms and Formal Languages. He is currently working at the TechSpot at the University of Arkansas as a lab technician.

**Soroush Shirali -** Soroush is a senior major at University of Arkansas. He has worked for Tech Genius as a project manager. He is currently TA for digital design and on the All University Academic Integrity Board as an undergraduate representative.

**5.0** **Facilities and Equipment**

Facilities used for this project include the Acxiom lab at JBHunt, TechSpot labs located across campus, and Mullins Library meeting rooms. Moreover, our meetings have primarily taken place virtually via Microsoft Teams. In terms of the equipment required, we used our own laptop/desktop computers to develop our project.

## **7.0 References**

[1] Gunders, Dana, and Jonathan Bloom. “Wasted: How America Is Losing Up to 40 Percent of Its Food from Farm to Fork to Landfill.” *Https://Www.nrdc.org/Sites/Default/Files/Wasted-2017-Report.pdf*, Natural Resources Defense Council 2017, Aug. 2017.

[2] Stierman, Bryan, and Suruchi Mishra. *Special Diets Among Adults: United States, 2015–2018*. NCHS Data Brief, Nov. 2020, www.cdc.gov/nchs/data/databriefs/db389-H.pdf.

[3]“Delicious Healthy Recipes Made with Real Food.” *Skinnytaste*, AN ELITE CAFEMEDIA FOOD PUBLISHER, www.skinnytaste.com/.

[4]*Recipeat*, 2020, recipeat.in/.

[5] *What is The MERN Stack? Introduction & Examples*. MongoDB. (n.d.). Retrieved December 8, 2021, from https://www.mongodb.com/mern-stack.

[6] Martin, Matthew. “MVC Framework Tutorial for Beginners: What Is, Architecture & Example.” *Guru99*, 28 Aug. 2021, www.guru99.com/mvc-tutorial.html.

[7] *The React Component Library You always wanted*. MUI. (n.d.). Retrieved December 8, 2021, from https://mui.com/.

[8] DigitalOcean. (2021, December 2). *How To Use Axios with React*. DigitalOcean. Retrieved December 8, 2021, from https://www.digitalocean.com/community/tutorials/react-axios-react.

[9] *Mongoose*. Mongoose ODM v6.1.0. (n.d.). Retrieved December 8, 2021, from https://mongoosejs.com/.