

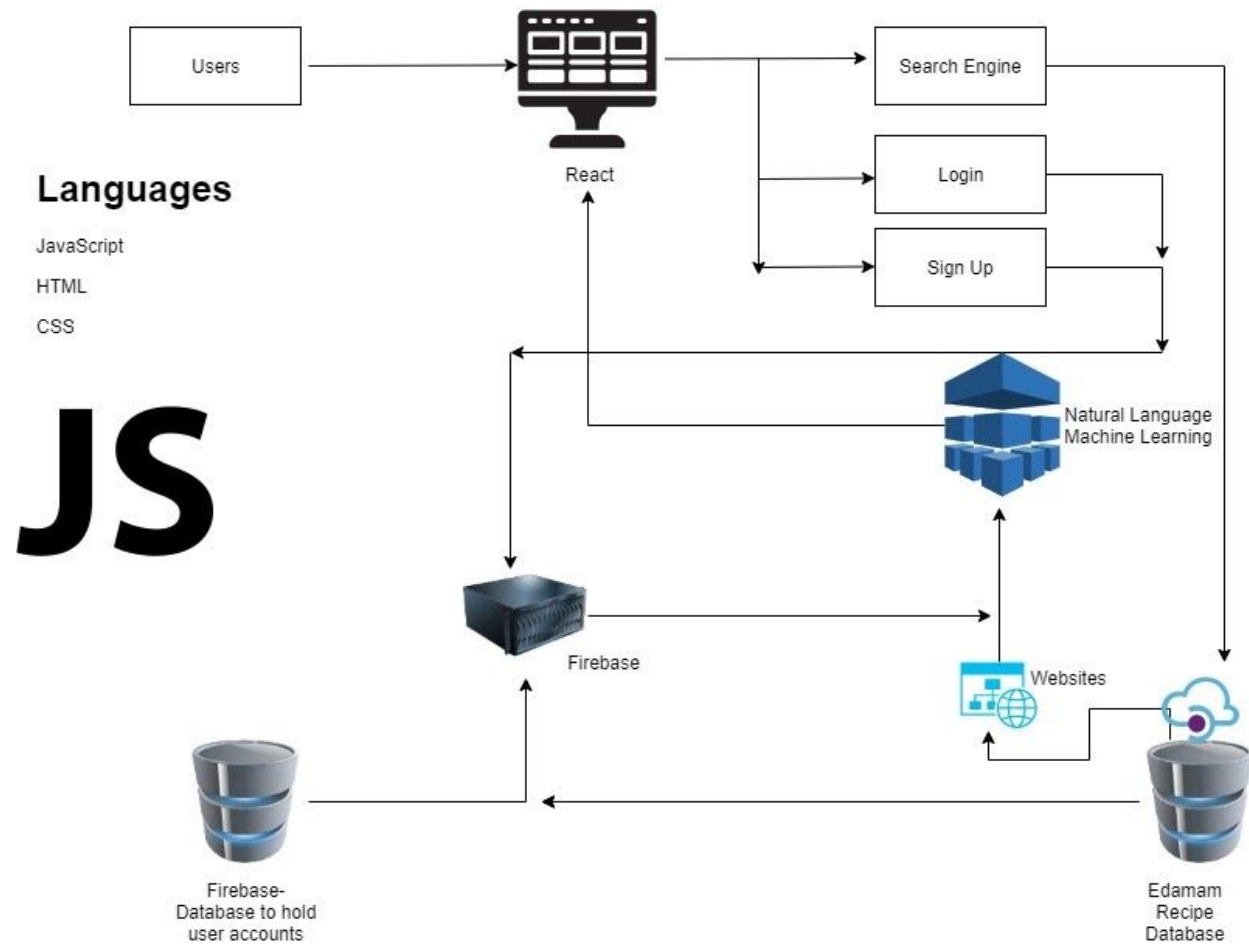
Architecture and Design Document

Table of Contents

Diagrams	2
System Diagrams	2
Component Diagrams	3
Use Case Diagrams	4
Activity Diagrams	5
Sequence Diagrams	6
Class Diagram	7
Front End Design	8
Storage Architecture	9
Recipe Explore Tab UI Example	9
Trade-Off Analysis	10
Machine Learning Analysis	14

Diagrams

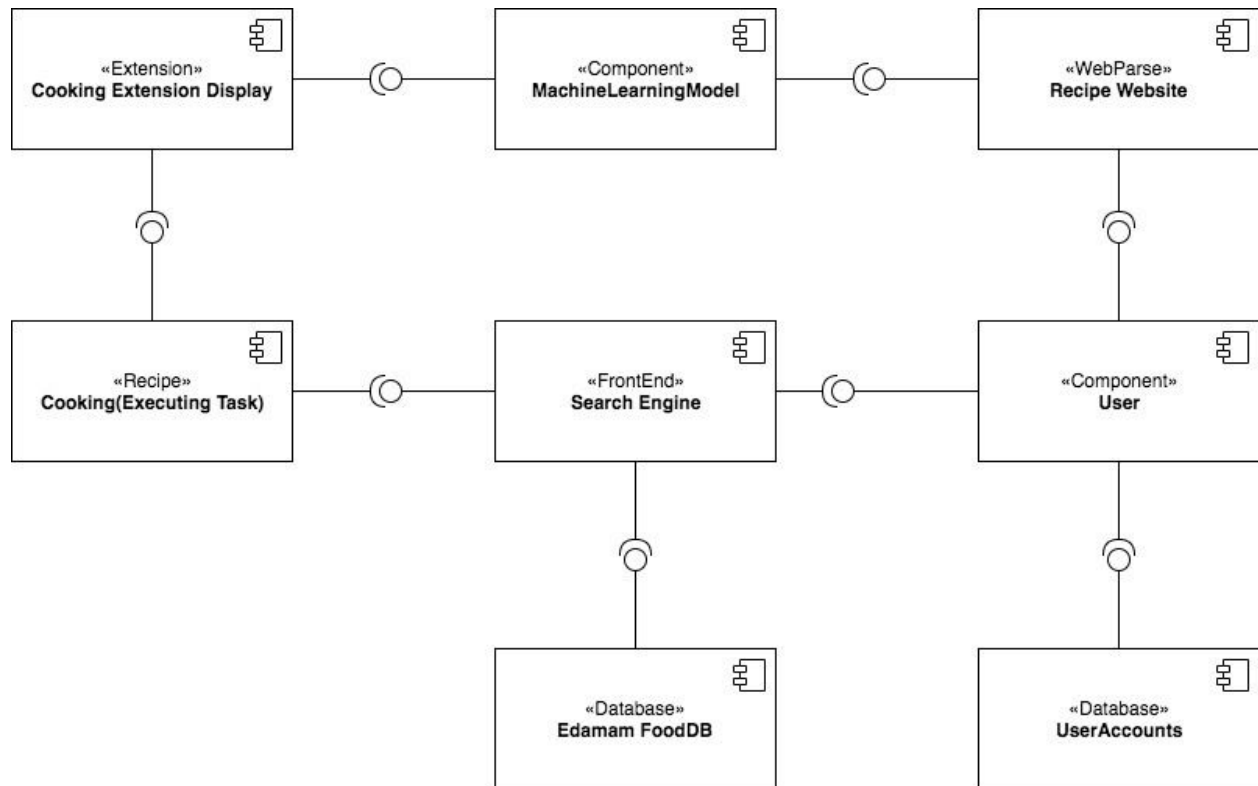
System Diagrams



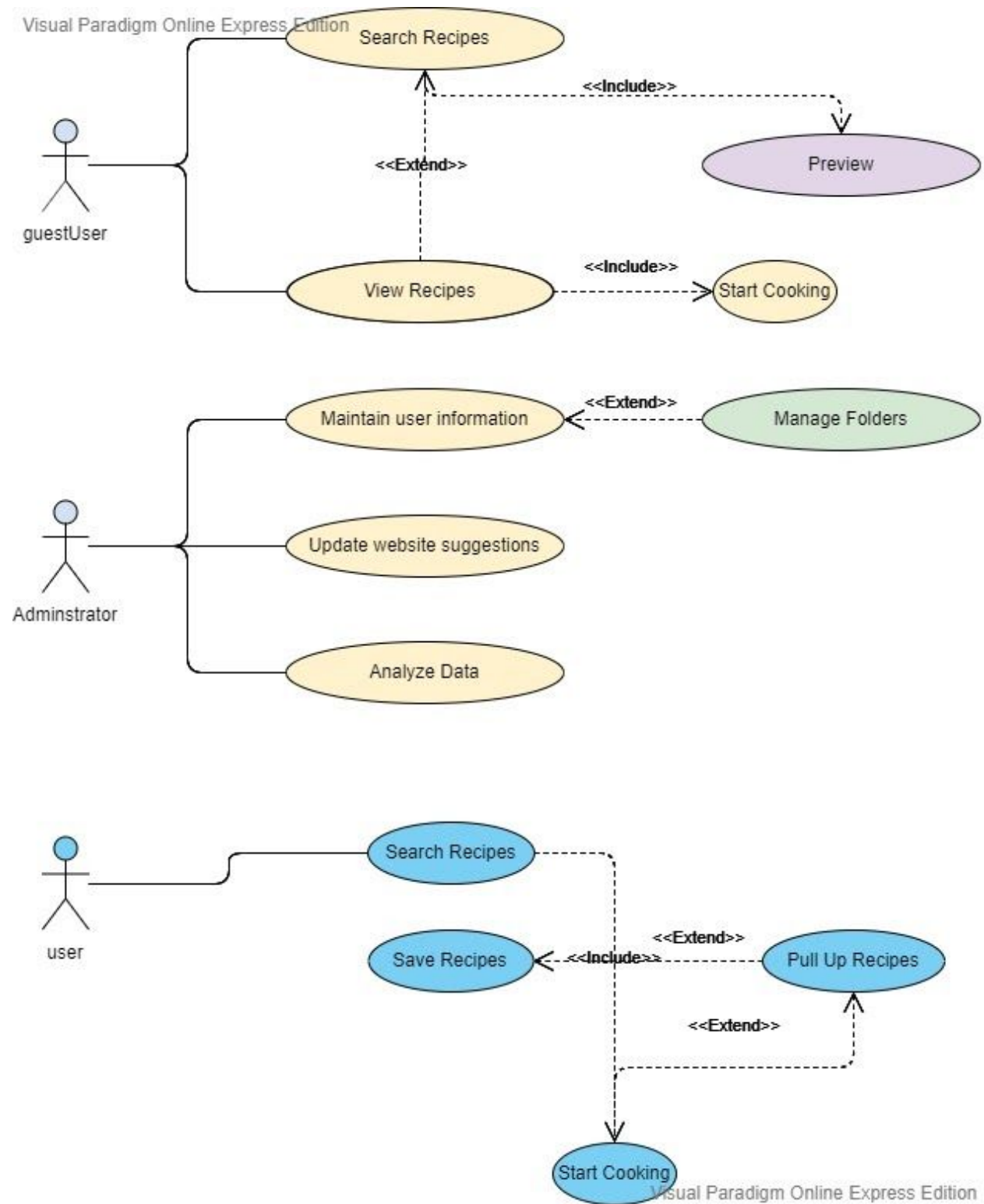
ODM



Component Diagrams

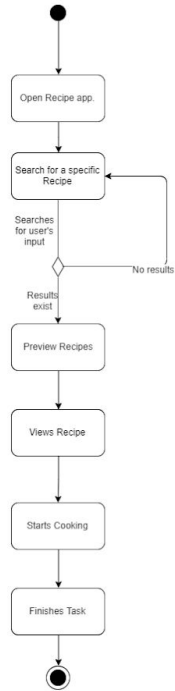


Use Case Diagrams

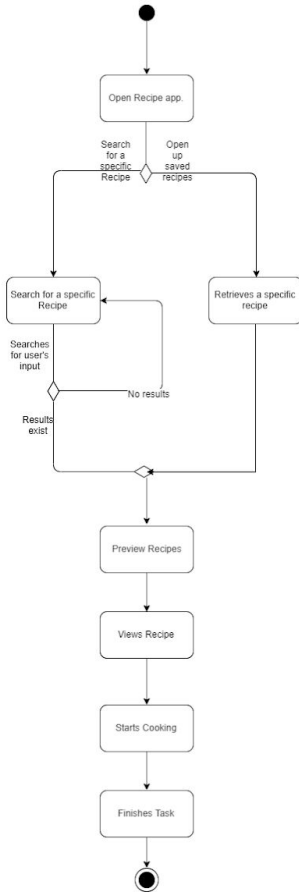


Activity Diagrams

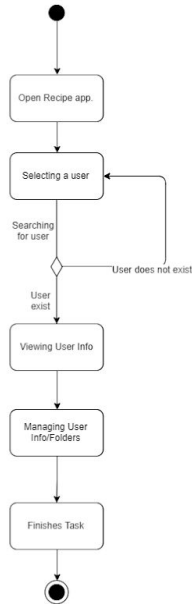
Guest User - Searches Recipe



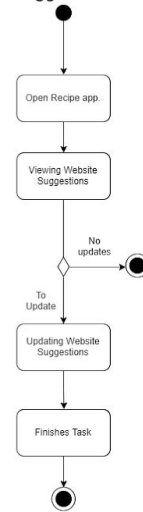
User - Searches Recipe



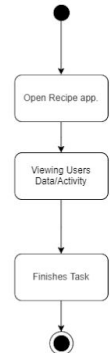
Admin - Maintain User Information



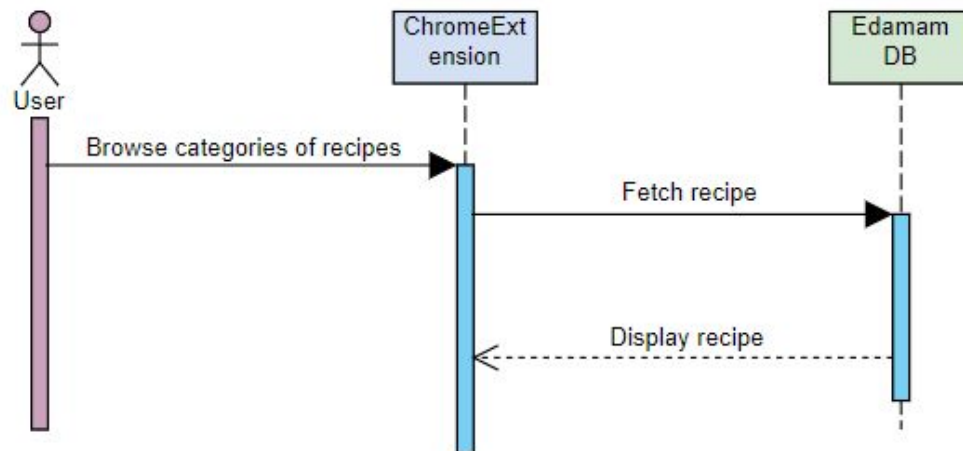
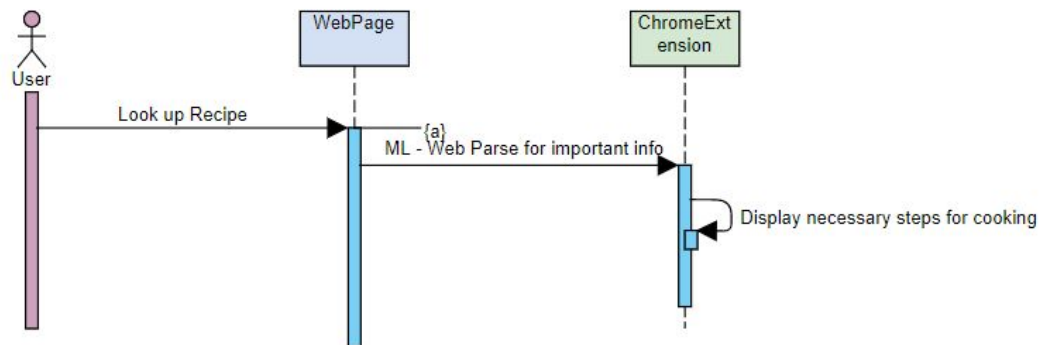
Admin - Update Website Suggestions



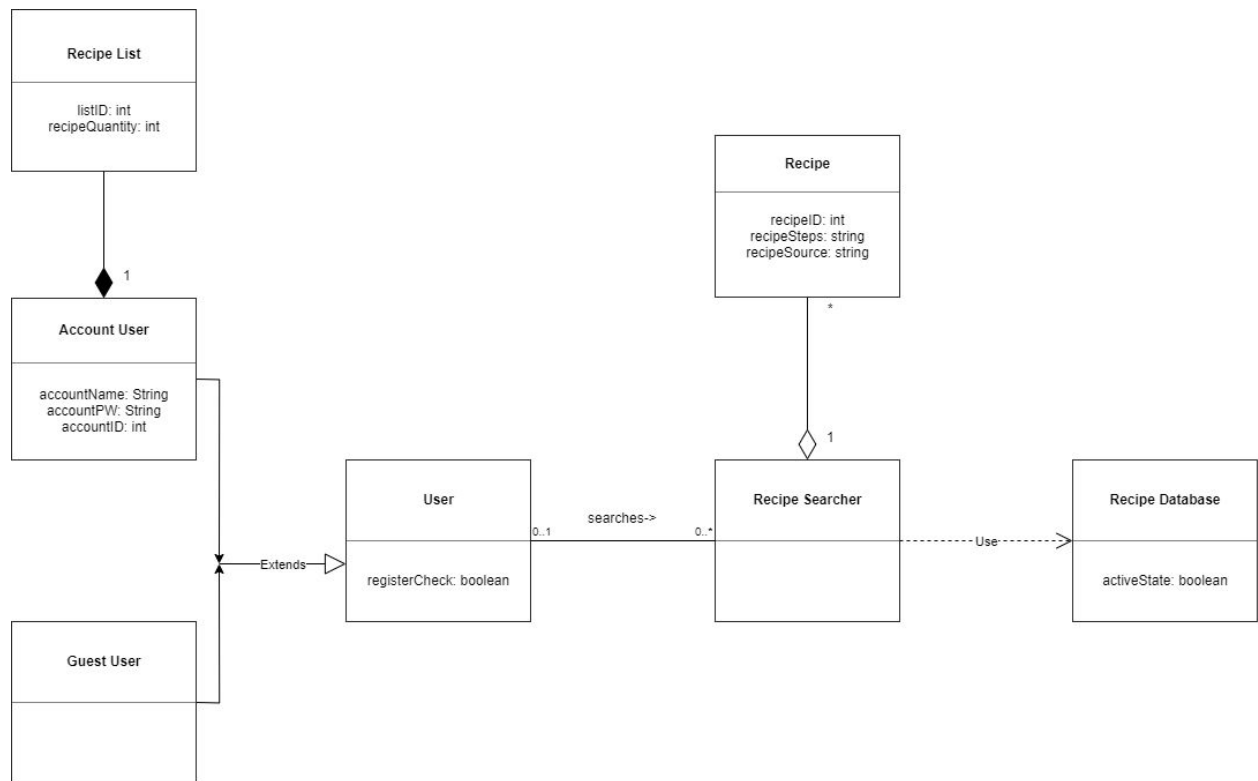
Admin - Analyze Data



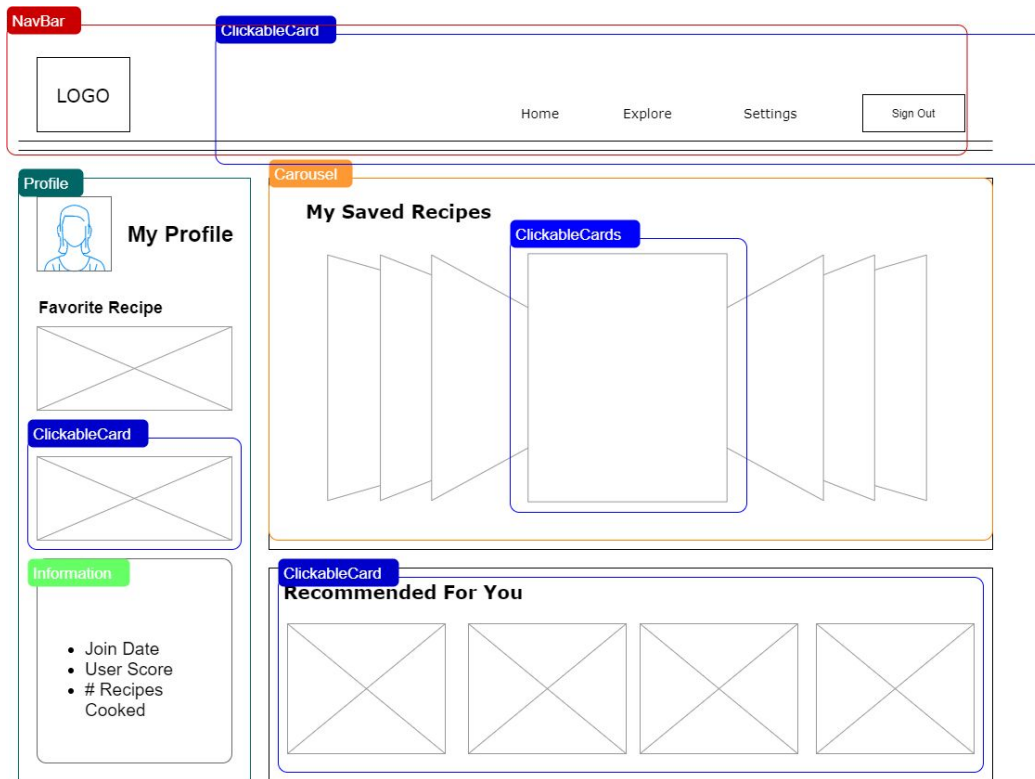
Sequence Diagrams



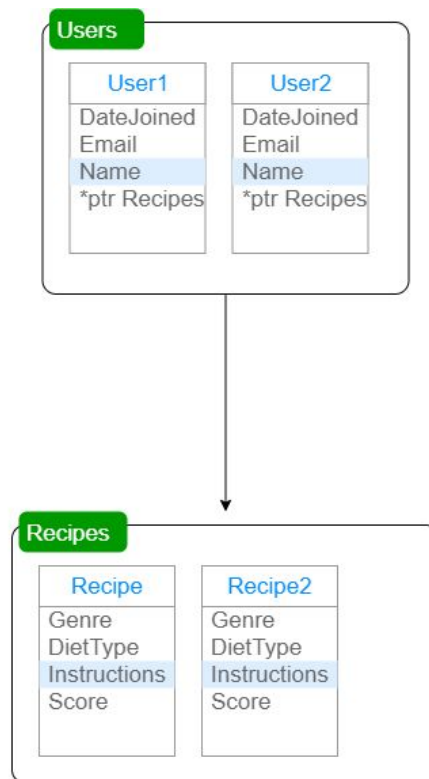
Class Diagram



Front End Design:





Storage Architecture



Recipe Explore Tab UI Example

The image shows a UI example for a "Recipe Finder" application. The interface has a teal background and a white search bar at the top with the placeholder text "Enter type of dish..." and a green "Search" button.

Below the search bar, there are two recipe cards displayed side-by-side:

- Chicken Vesuvio**
 - Ingredients:**
 1. 1/2 cup olive oil
 2. 5 cloves garlic, peeled
 3. 2 large russet potatoes, peeled and cut into chunks
 4. 1 3-4 pound chicken, cut into 8 pieces (or 3 pound chicken legs)
 5. 3/4 cup white wine
 6. 3/4 cup chicken stock
 7. 3 tablespoons chopped parsley
 8. 1 tablespoon dried oregano
 9. Salt and pepper
 10. Salt and pepper
 11. 1 cup frozen peas, thawed
 - Calories: 4055.76
 - [Click Here](#)
 - 
- Chicken Paprikash**
 - Ingredients:**
 1. 640 grams chicken - drumsticks and thighs (3 whole chicken legs cut apart)
 2. 640 grams chicken - drumsticks and thighs (3 whole chicken legs cut apart)
 3. 1/2 teaspoon salt
 4. 1/4 teaspoon black pepper
 5. 1 tablespoon butter – cultured unsalted (or olive oil)
 6. 240 grams onion sliced thin (1 large onion)
 7. 70 grams Anaheim pepper chopped (1 large pepper)
 8. 25 grams paprika (about 1/4 cup)
 9. 1 cup chicken stock
 10. 1/2 teaspoon salt
 11. 1/2 cup sour cream
 12. 1 tablespoon flour – all-purpose
 - Calories: 3033.20
 - [Click Here](#)
 - 

Trade-Off Analysis

The Client-Server Architectural Design was chosen as the architecture for the project. MVVM and MVC were other patterns considered for the project.

Client-Server Architecture	
<u>Pros</u>	<u>Cons</u>
<ul style="list-style-type: none">- Better data sharing since data is all on a single server- Easier maintenance and better security control access- Resources are shared across different platforms- Ability to log into the system despite location or technology of the processor- Users can access the server through an interface rather than having to log into a terminal mode	<ul style="list-style-type: none">- Multiple simultaneous clients can overload the server and cause slowdown- If server fails, no user can use the application until servers are fixed

Model-View-ViewModel (MVVM)	
<u>Pros</u>	<u>Cons</u>
<ul style="list-style-type: none">- Separates UI and application logic to clearly define where certain code goes- Better unit testing, because it allows for testing of individual components without affecting the others- Developers can focus on either the UI or the application logic without worrying about the other, leading	<ul style="list-style-type: none">- Adds complexity to Presentation Layer of the application, which adds a learning curve for some developers- Errors aren't generated at compile time, but instead at run time, usually producing silent errors and making debugging harder

to safer coding	
-----------------	--

Model-View-Controller (MVC)	
<u>Pros</u>	<u>Cons</u>
<ul style="list-style-type: none"> - High cohesion, low coupling - Easier to modify due to separation of concerns - Multiple developers can work components at the same time 	<ul style="list-style-type: none"> - Adds complexity which adds a new learning curve for some developers - Developers need to maintain the consistency of multiple representations at once

Trade-Off Analysis : Front-end

Pros

Cons



- Component Reusability
- Google support
- Third-party integrations for better functionalities

- Steep Learning curve, difficult to learn




- Component Reusability
- Detailed documentation for ease of learning
- Very lightweight and easily integratable

- Small community due to being new, lack of support
- High pace of development

Trade-Off Analysis : Back-end

Pros

Cons

<div data-bbox="207 499 496 674"></div> <ul style="list-style-type: none">• No data capacity - you pay for the amount you use on the cloud• Very detailed documentation for ease of learning• Array of tools ready to be deployed (ex. Database - Amazon Aurora, analytics)	<ul style="list-style-type: none">• Price Packages (Developer, Business, Enterprise)• No control over environment, dependent on Amazon
<div data-bbox="207 1031 496 1325"></div> <ul style="list-style-type: none">• Document-oriented NoSQL database for ease of access of indexing• Direct use of JSON and JavaScript frameworks• Free of cost	<ul style="list-style-type: none">• Less flexibility in queries (ex. No joins)• No support for transactions (updating documents/collections) → risk of duplication of data

1. **Major Architecture:** Client Server
2. **Component Choice:** Undecided.
3. **Language Choices:** JavaScript, HTML, CSS
4. **Framework Choices:** N/A
5. **Database Choices:** Firebase Firestore Database (NOSQL)

- 6. Server vs Serverless Choices:** Firebase
- 7. Front end Framework:** React
- 8. API Choices:** Edaman Recipe Database
- 9. Cloud Decisions:** Cloud Firestore
- 10. Security Decisions:** Firebase Authentication
- 11. Logs/Monitoring Choices:** Firebase Firestore Database (NOSQL)
- 12. Process Decisions:** our program fit very well for the client server architecture due to the fact that users interact with our app, sends a request to the server, then the server returns the data back to the user. In this case, it would be a recipe request to the server.
- 13. Future Additions:** N/A

Machine Learning Analysis

In order to accomplish this, our initial idea of the type of Machine Learning that would be incorporated into our application would be a Natural Language Processing Model. This would allow the user to receive recommendations to other recipes based on the current search of their recipe as well as previous viewings of recipes. Our very first step would be to create an algorithm that differentiates strings that relate to cooking/recipe compared to unrelated strings. In order to do this, we would need to gather up a dictionary of common words used from recipe web pages and there on train our algorithm to recognize and differentiate words apart from each other. We're focusing on a content-based recommendation model to our project to achieve this functionality.

Content based recommendation

Advantages	Disadvantages
Does not depend on data of other users.	When we have a new user, without much information about his transactions, we cannot make accurate recommendations.
There is no cold start problem for new items. This is because, using the item features we can easily find items it is similar to.	Clear-cut groups of similar products may result in not recommending different products. We may end up recommending a small subset over and over again.
Recommendation results are interpretable.	If there is limited information about the content, it is difficult to clearly discriminate between items and group them, resulting in inaccurate recommendations