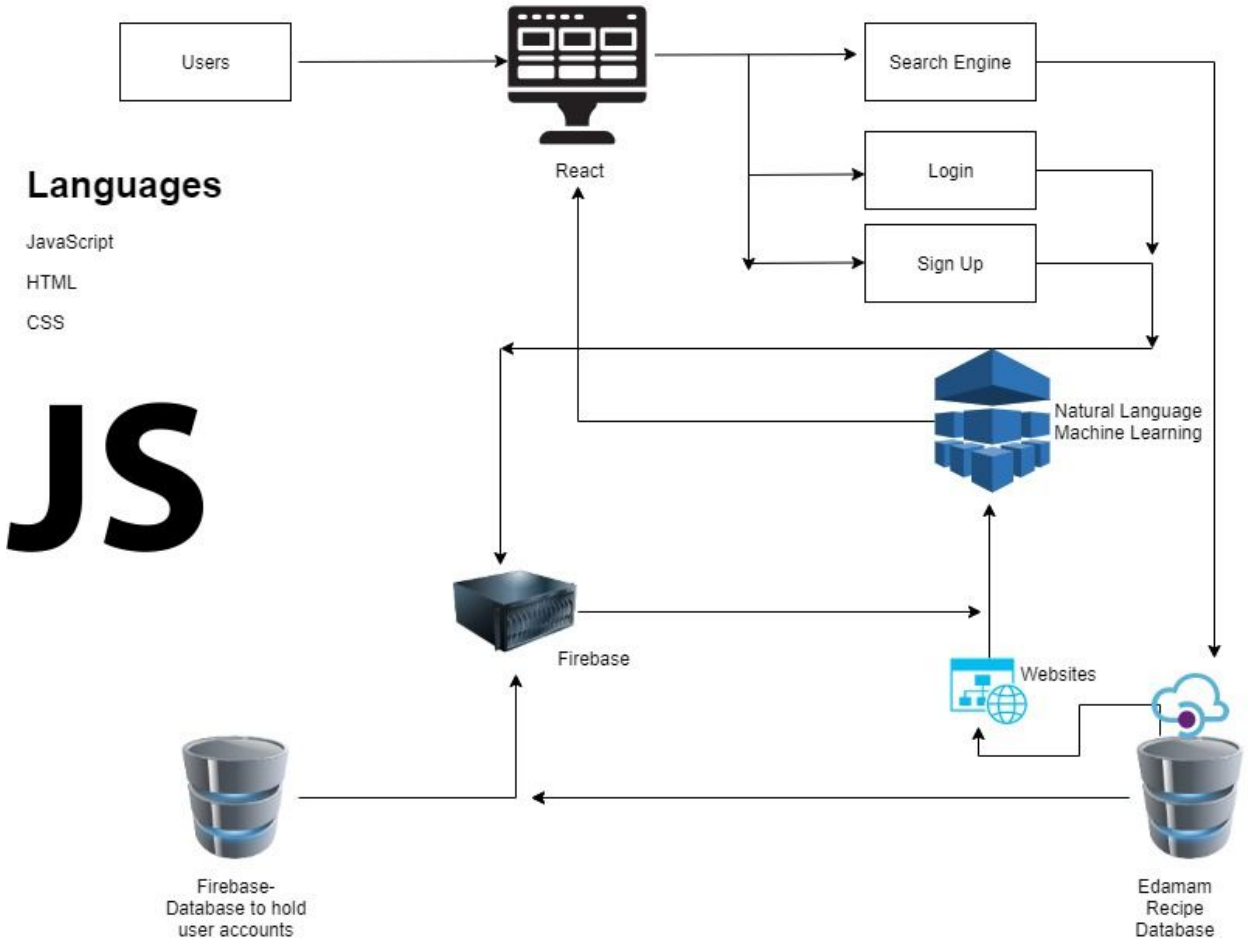
Architecture and Design Document

Table of Contents

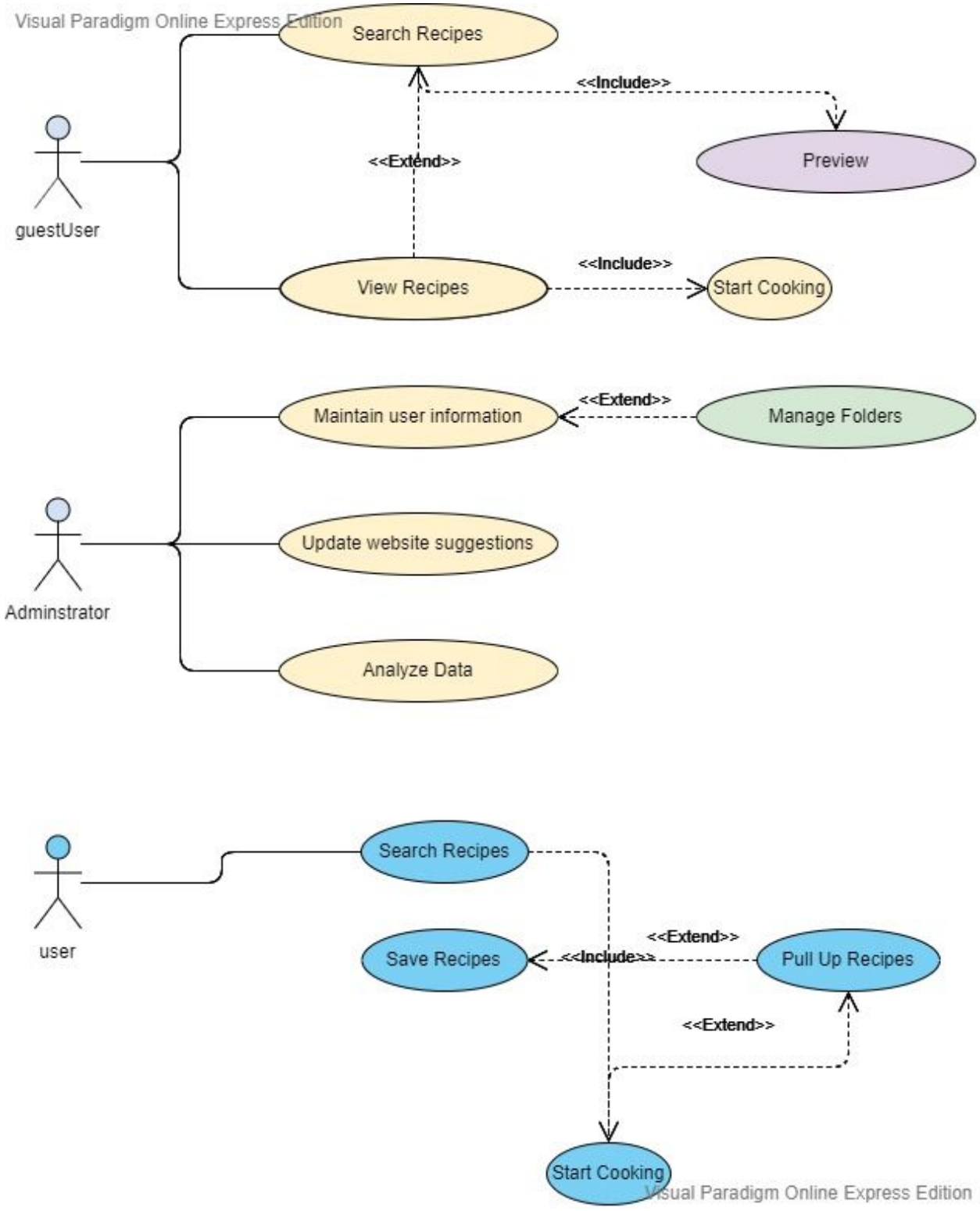
Diagrams

System Diagrams

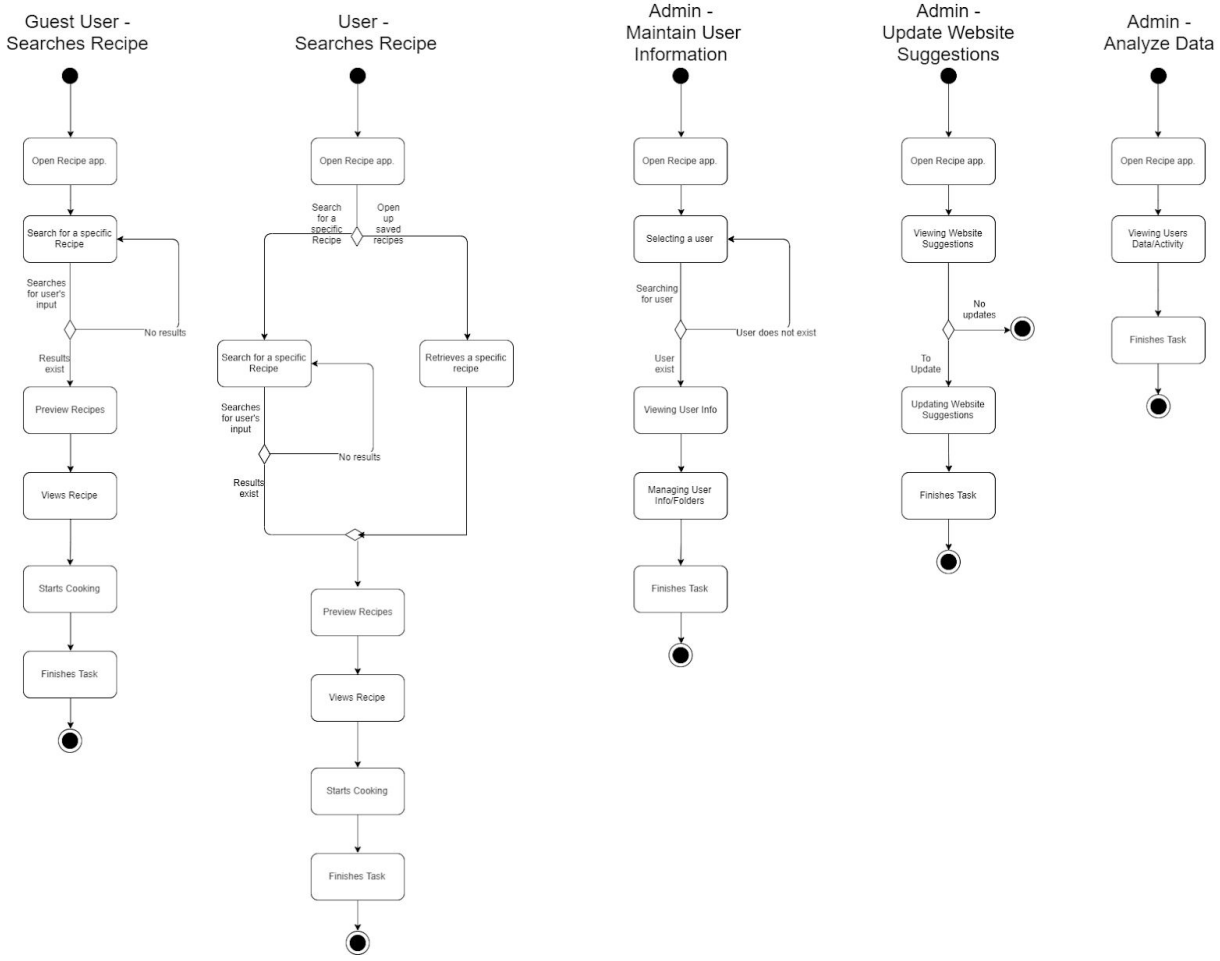


Component Diagrams

Use Case Diagrams



Activity Diagrams



Sequence Diagrams

Class Diagram

Entity Relationship Diagram

Analysis

Trade-Off Analysis

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

Client-Server Architecture	
Pros	Cons
- 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	- 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)	
Pros	Cons
- 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 to safer coding	- 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

Model-View-Controller (MVC)	
Pros	Cons
- High cohesion, low coupling - Easier to modify due to separation of concerns - Multiple developers can work components at the same time	- Adds complexity which adds a new learning curve for some developers - Developers need to maintain the consistency of multiple representations at once

1. **Major Architecture:** Client Server
2. **Component Choice:** N/A
3. **Language Choices:** JavaScript, HTML, CSS
4. **Framework Choices:** N/A
5. **Database Choices:** Firebase Database (NOSQL)
6. **Server vs Serverless Choices:** Firebase
7. **Front end Framework:** React
8. **API Choices:** Edaman Recipe Database
9. **Cloud Decisions:** Firebase
10. **Security Decisions:** Firebase
11. **Logs/Monitoring Choices:** Firebase
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