

Restaurant Finder

The Developers

Filimon Gebrekidan

James Dulflinger

Ethan Kolkmeier

Carl Trautwein

SOFTWARE DESIGN DOCUMENT

Version 1.0

INDEX

RESTAURANT FINDER	Page No.
1. Introduction	3
2. Overview of architecture	3
3. Use cases	4
A. Use case diagram	
B. Use case specifications	
i. UC1- Login	
ii. UC2- Search Restaurants	
iii. UC3 - Save Favorites	
iv. UC4 - Remove Favorites	
v. UC5 - Select Restaurant	
vi. UC6 – Choose View	
4. Classes and their interactions	8
A. Class diagram	
B. Details of classes	
i. C1 – User Class	
ii. C2 – Selection Class	
iii. C3 – Route Class	
iv. C4 – Restaurant Class	
v. C5 – Favorite Class	
C. Sequence diagrams	
i. S1 – Login	
ii. S2 – Search Restaurant	
iii. S3 – Select Restaurant	
iv. S4 – Add Favorite	
v. S5 – Remove Favorite	
5. State transition diagram	15
6. User interface design	16
A. Screen 1 – Home page	
B. Screen 2 - Login page	
C. Screen 3 - Register page	
D. Screen 4 - Main page	
7. Conclusion	20

1. Introduction

This document describes the Restaurant Finder application. Restaurant Finder provides the user with relevant information about each restaurant selected, such as distance, crowding, and price. The app provides this information in a long list format as well as a map view. In addition RF (Restaurant Finder) helps people decide based on their preferences that each user will have on their own accounts as well as favorites.

Design is one of the most important aspects of software engineering. It provides developers with a clear understanding and implementation of the requirements. It is important for a software development team to think about design so that everyone has an idea about what the end goal will be and what building blocks are needed to create it. Otherwise, team members may be uncoordinated and unfocused.

This document will overview the architecture of the project, which pertains to the technologies used in the project. Classes and their interactions showcase how the different parts of the project are supposed to work together. The software states and transitions further detail the functions of the project and how it will work. Finally, the user interface design shows how users will interact with the finished product.

2. Overview of architecture

The Firebase database will be used to store information about user preferences, credentials, favorites, and most visited locations. Firebase will be queried for this information after sign-in is confirmed and will be updated when a user makes a change to any of this information.

Google Maps will be sent information about the location of the user's device and return information about restaurants nearby. One query will happen on sign-in, but several more could happen depending on when the user moves too far from their starting point or changes their requested filters.

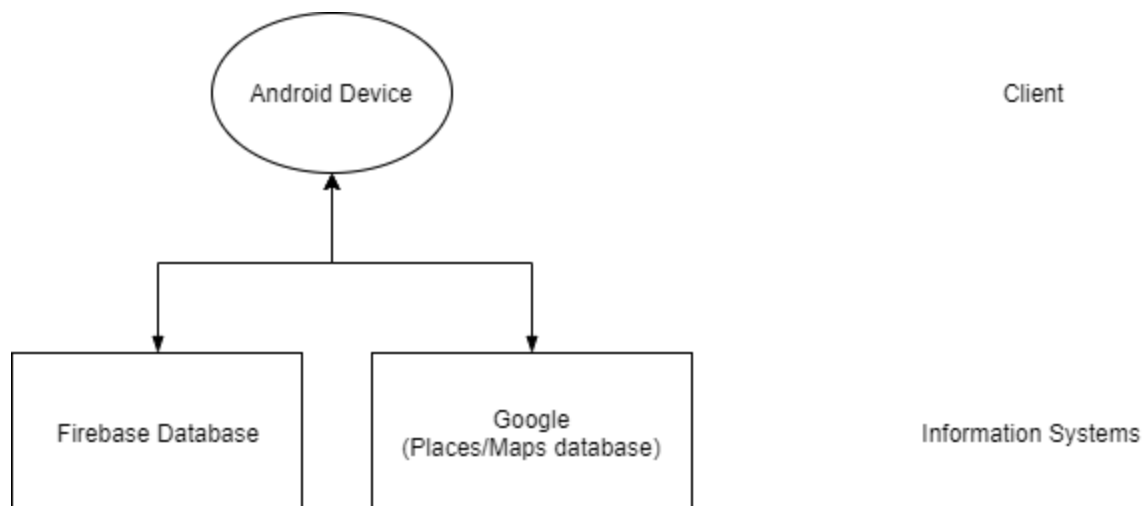


Figure 1. Architecture

Restaurant Finder's architecture most closely resembles a client-server architecture. This architecture is advantageous because it allows most processing to be handled on the user's device. Queries are only made when necessary, reducing the time spent requesting data from over the internet. Firebase also helps keep the space requirement on users' devices small by storing information elsewhere. Firebase is also useful because it allows user credentials to be stored on a server, meaning the user could log-in from any device and account information will transfer.

The Google databases also keep information about geography and businesses on a server that devices can make requests to. This is Google's default way of handling the process and matches with the client-server architecture.

3. Use Cases

A. Use case diagram

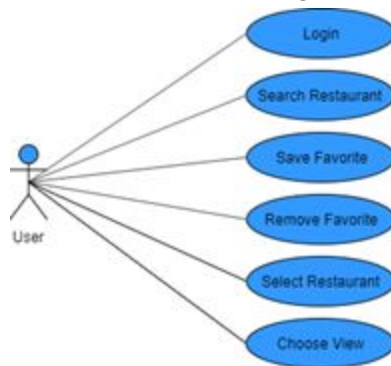


Figure 2. Use Case Diagram

B. Use case specifications

i. UC1 – Login

Use Case Number:	UC-0001
Use Case Name:	Login
Overview:	Existing user logs in.
Type:	Primary
Actors:	User

Pre-Condition:	User starts the app.
Main Flow:	<ol style="list-style-type: none"> 1. System waits for user to input credentials 2. User inputs credentials 3. If credentials are correct, send user to main page
Alternate Flow:	<ol style="list-style-type: none"> 3. If user credentials are wrong restart
Post Condition:	User is logged in.
Cross Reference:	None

ii. UC2 – Search Restaurants

Use Case Number:	UC-0002
Use Case Name:	Search Restaurants
Overview:	User searches for restaurants.
Type:	Primary
Actors:	User
Pre-Condition:	User is logged into the app.
Main Flow:	<ol style="list-style-type: none"> 1. User creates a query 2. System displays restaurants relating to that query
Alternate Flow:	<ol style="list-style-type: none"> 1. No query is sent and a default list of restaurants within 10 miles is shown
Post Condition:	A list of restaurants is shown
Cross Reference:	

iii. UC3 – Save Favorite

Use Case Number:	UC-0003
Use Case Name:	Save Favorite

Overview:	User saves a restaurant as a favorite.
Type:	Primary
Actors:	User
Pre-Condition:	User has selected a restaurant to add to favorites.
Main Flow:	<ol style="list-style-type: none"> 1. User selects option to add the restaurant to favorites 2. System stores the favorite
Alternate Flow:	<ol style="list-style-type: none"> 2. The restaurant is already a favorite, so it can't be added.
Post Condition:	The favorites list has a new entry: the new restaurant.
Cross Reference:	Favorites list

iv. UC4 – Remove Favorite

Use Case Number:	UC-0004
Use Case Name:	Remove Favorite
Overview:	User removes a favorite from favorites list.
Type:	Primary
Actors:	User
Pre-Condition:	User has selected a restaurant that is in favorites already.
Main Flow:	<ol style="list-style-type: none"> 1. Select option to remove the restaurant from favorites 2. Favorite is removed. 3. Favorite is deleted
Alternate Flow:	<ol style="list-style-type: none"> 2. If the restaurant isn't in favorites, then it hasn't been added so do nothing
Post Condition:	The selected restaurant should no longer be in the favorites list.
Cross Reference:	Favorites list

v. UC5 – Select Restaurant

Use Case Number:	UC-0005
Use Case Name:	Select Restaurant
Overview:	User selects a restaurant from a list or map.
Type:	Primary
Actors:	User
Pre-Condition:	User has searched for restaurants and they are on display for selecting.
Main Flow:	<ol style="list-style-type: none"> 1. User chooses a restaurant 2. User is shown select screen 3. User decides whether to confirm or cancel the selection
Alternate Flow:	2. Cancel will take the user back to the previous screen
Alternate Flow:	2. Confirm will take the user to the map view to see a route
Post Condition:	Exit the select screen.
Cross Reference:	Selected Restaurant

vi. UC6 – Choose View

Use Case Number:	UC-0006
Use Case Name:	Choose View
Overview:	User chooses a view to see restaurants searched for.
Type:	Primary
Actors:	User
Pre-Condition:	User has searched for restaurants in his area to view.
Main Flow:	1. Press the “View” Button to change view
Alternate Flow:	1. If the current view is list view, change to map view
Alternate Flow:	1. If the current view is map view, change to list view
Post Condition:	The view has changed.
Cross Reference:	Previous View

4. Classes and their interaction

A. Class diagram

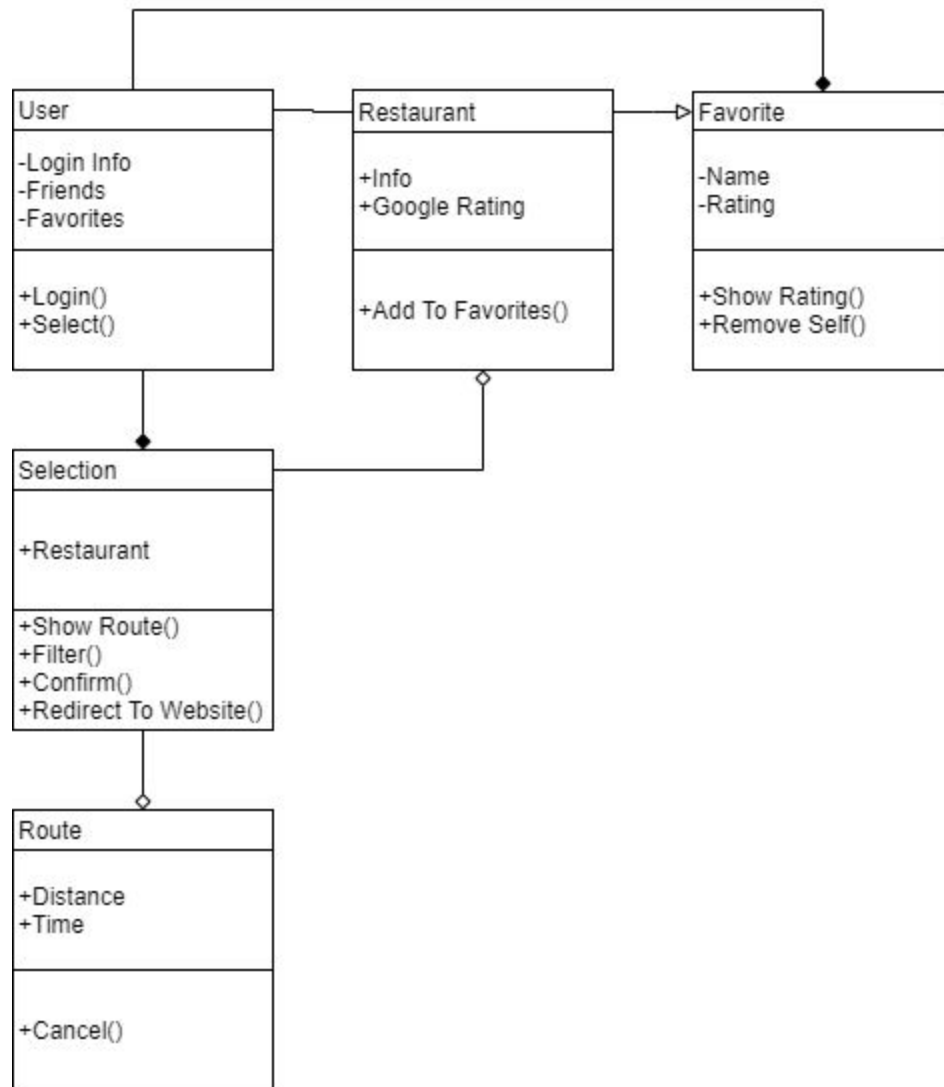


Figure 3. Class Diagram

B. Details of classes

i. C1 – User Class

Attributes:

Attribute Name	Type	Description
Login Info	string	user name, password

Friends	list of users	For connecting with other users on this app.
Favorites	list of favorites	List of restaurants for quick reference.

Operations/Methods:

Method Name	Argument Passed	Expected Return Value	Description
Login	string	None	Function for accessing the app.
Select	void	None	Function for showing information on a restaurant selected.

ii. C2 – Selection Class

Attributes:

Attribute Name	Type	Description
Restaurant	restaurant class	This is the restaurant selected.

Operations/Methods:

Method Name	Argument Passed	Expected Return Value	Description
Show Route	void	None	Create an instance of a route.
Filter	enum	List of Restaurants	Based on some attribute, only specific restaurants will show up.
Confirm	void	None	A restaurant

			becomes the selected restaurant.
Redirect To Website	string	None	Open Website of the restaurant if available in a separate browser.

iii. C3 – Route Class

Attributes:

Attribute Name	Type	Description
Distance	double	Distance to the destination.
Time	double	Estimated time until the user reaches the destination.

Operations/Methods:

Method Name	Argument Passed	Expected Return Value	Description
Cancel	void	None	Delete the route.

iv. C4 – Restaurant Class

Attributes:

Attribute Name	Type	Description
Info	string	Description of the restaurant from Google.
Google Rating	double	Rating of the restaurant from Google.

Operations/Methods:

Method Name	Argument Passed	Expected Return Value	Description
-------------	-----------------	-----------------------	-------------

Add To Favorites	restaurant (this)	None	Add a Favorite
------------------	-------------------	------	----------------

v. C5 – Favorite Class

Attributes:

Attribute Name	Type	Description
Name	string	Name of Favorite
Rating	double	Personal rating of a restaurant

Operations/Methods:

Method Name	Argument Passed	Expected Return Value	Description
Show Info	void	None	Display Favorite Restaurant's name and rating.
Remove Self	favorite (this)	None	Remove from list of Favorites, Delete self.

C. Sequence diagrams

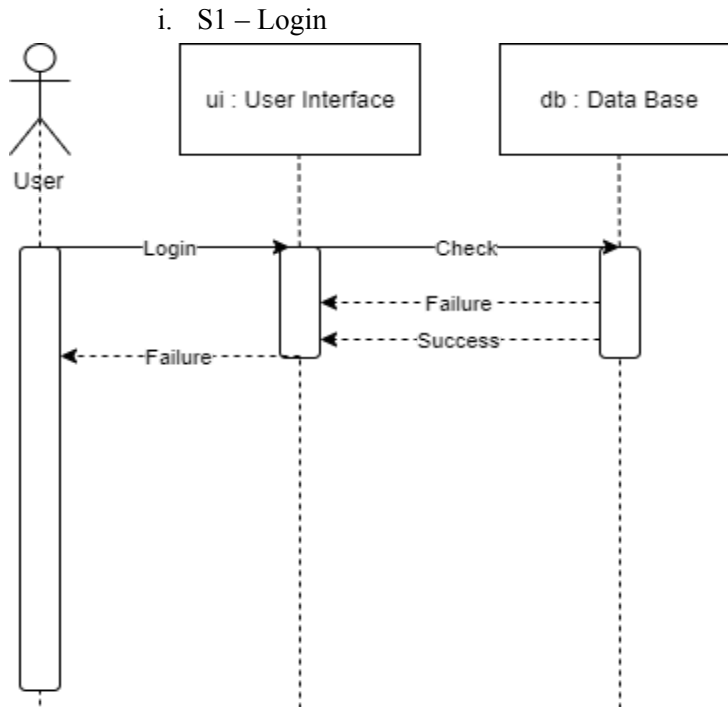


Figure 4. Login Sequence Diagram

The User will attempt to login and will put their login information into the user interface. The information will be checked in the database for existing logins. If it is successful the result will be returned to the user interface. If it is a failure, the user will be notified.

ii. S2 – Search Restaurant

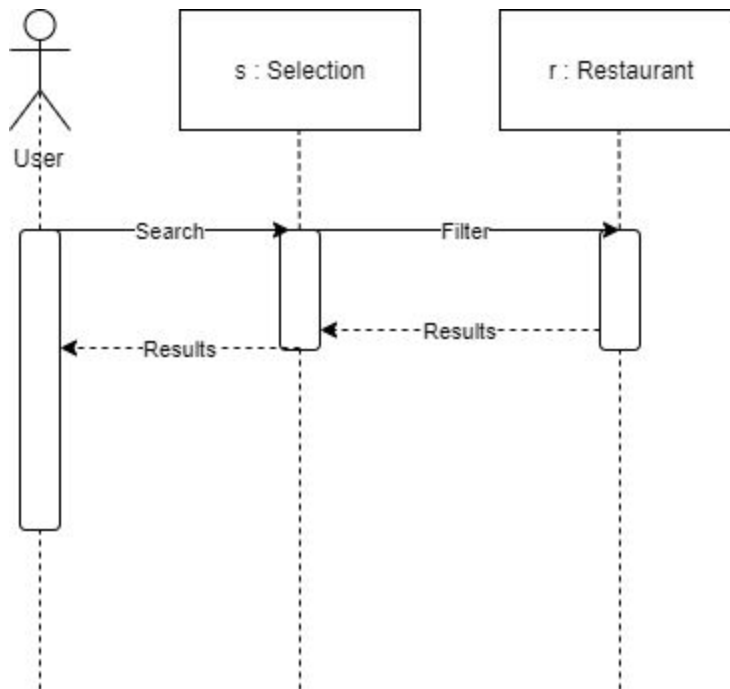


Figure 5. Search Sequence Diagram

Whenever the user searches for a restaurant there will be a default filter so only restaurants within 10 miles are shown. Additionally, details about the restaurant can be added to the filter and only restaurants that match that filter will be shown.

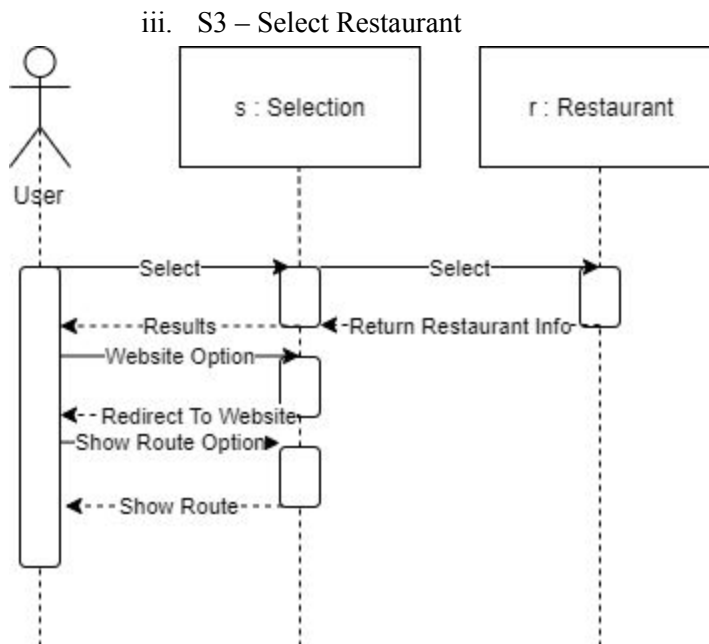


Figure 6. Select Restaurant Sequence Diagram

When the User Selects a restaurant, the restaurant's information is displayed for the user and the user is given other options for what to do. Assuming the user doesn't cancel the selection by searching again, the user can look at the restaurant's website, assuming the restaurant has a website. The user will also be able to show the route of the selection so that he/she can travel to that destination.

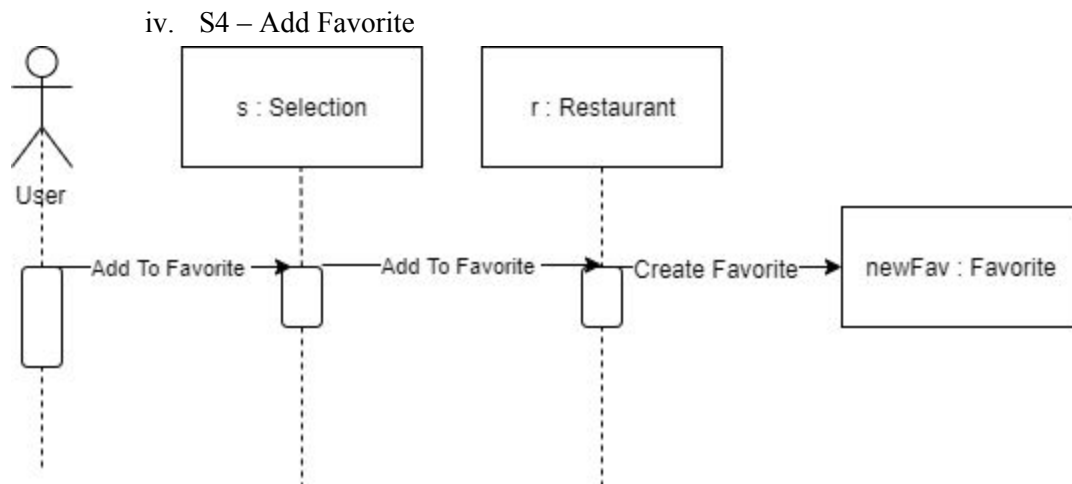


Figure 7. Add Favorite Sequence Diagram

The user can add a selected restaurant to their “favorites”. The request is sent to the restaurant class to create a new favorite object to be stored in the favorites list.

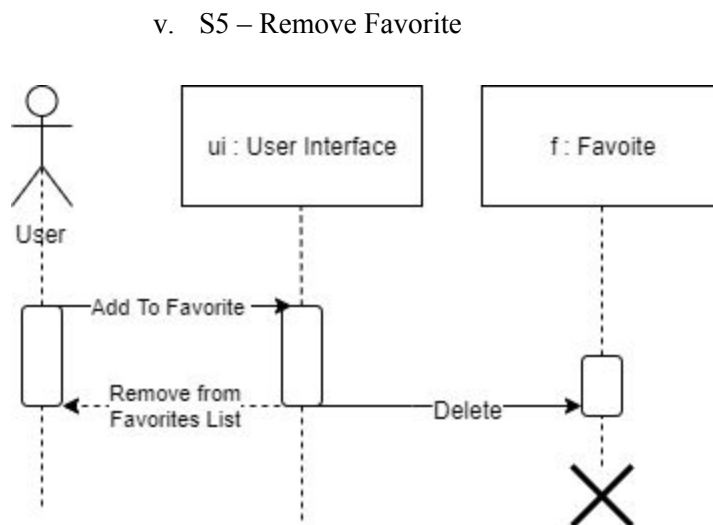


Figure 8. Remove Favorite Sequence Diagram

The user can remove a restaurant from their “favorites” too. The request to remove it is sent to the user's class and the favorite is removed from the list.

5. State transition diagrams

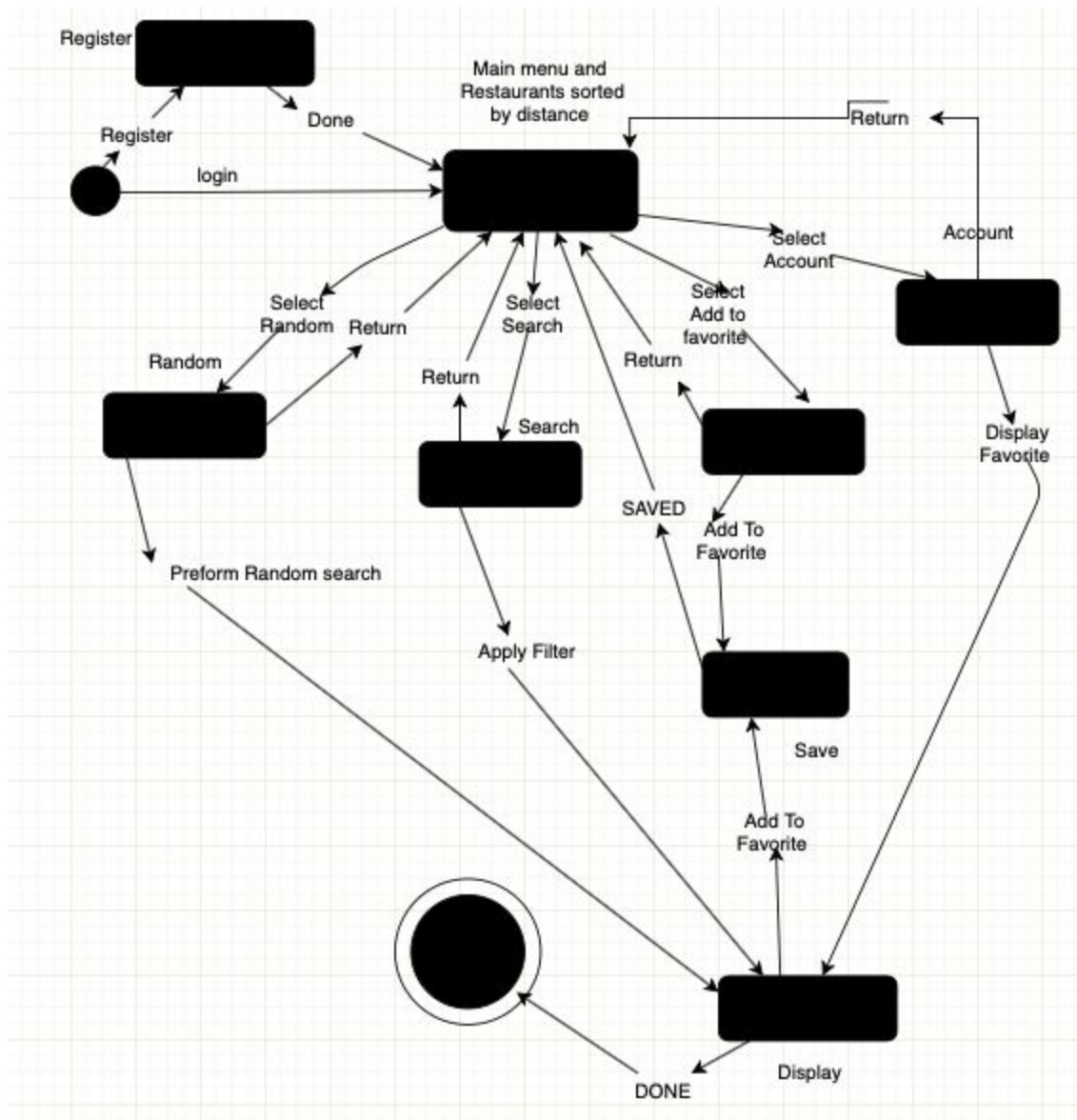


Figure 9. State Transition Diagram

When users open the app, they will enter an idle state. The user can transition to the main page by either logging in or by completing the registration. When the users enter the main page, they have four states to transition to. The four states are Search, Random, Add to Favorite and Account.

- Search - User can search for restaurants
- Random - App will generate a random restaurant
- Add to favorite - Name a restaurant to be added to favorites
- Account - display account information, including favorites

The users can make one of the selections. After the user makes a selection, they will transition to one of the states but have an option to transition back to the main page. If the user selects Search, they transition to the display state. If the user select Random, the user will transition to a display state. If the user selects

Add to Favorite, they can transition to saving state by adding the restaurant to favorites. If the user selects Account, they can transition to display state by choosing show favorites.

6. User interface design

A. Screen 1 – Home page

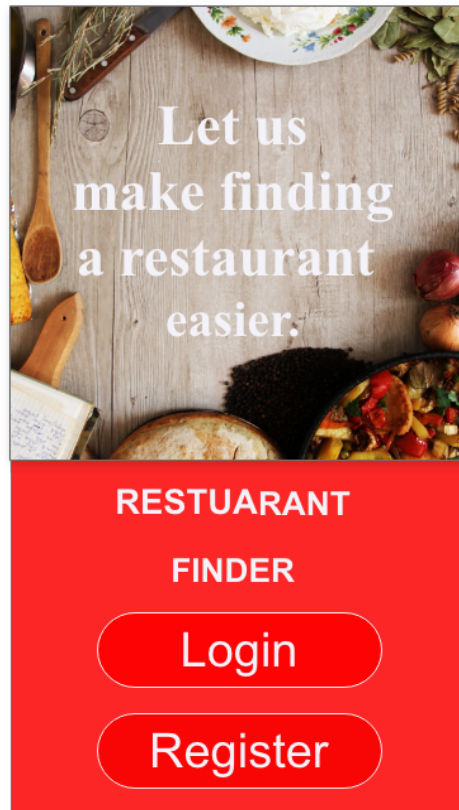
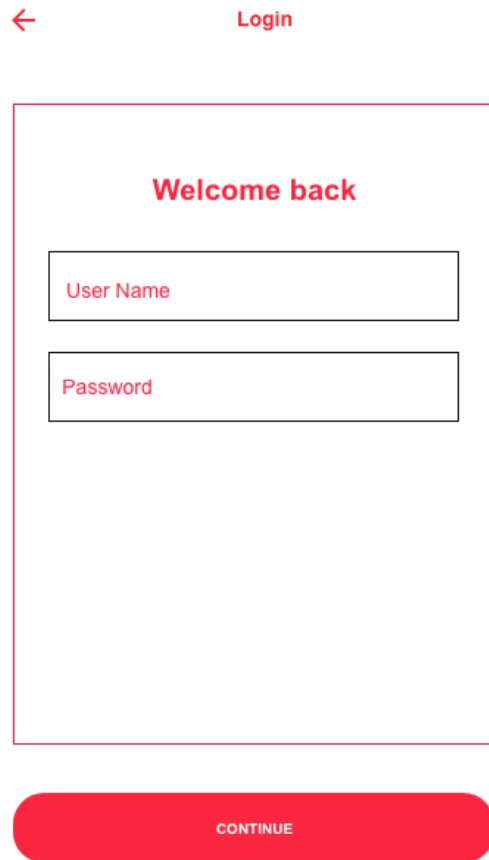


Figure 10. Home Page

This is the home menu that will be displayed as soon as the user opens the app. This page will have a login component for returning users and a register component for new users who wish to register. The login component is located at the bottom of the screen below the name of the app and above the register component. The register component located at the bottom of the home page, below the login component. When the user clicks on the “Login”, the user will be taken to the login page. When the user clicks on “Register”, the user will be taken to the user to the registering page.

B. Screen 2 - Login page

The login page will ask the user for a user name and password. The user can enter the information and click continue, which is located at the bottom of the page or return to the home page by clicking on the arrow located at the top left corner. The image below shows what the user would see if they chose the login component.



The image shows a mobile app login screen. At the top left is a red back arrow, and at the top center is the word "Login" in red. Below this is a large white rectangular box with a red border. Inside this box, the text "Welcome back" is centered in red. Below the text are two stacked input fields, each with a red border and red placeholder text: "User Name" and "Password". Below the white box is a red rounded rectangular button with the word "CONTINUE" in white capital letters.

Figure 11. Login Page

C. Screen 3 - Register page

This component would allow new user to create an account to use the app.

The user will need to enter an email address and create a username and password. The user can enter the information and click continue, which is located at the bottom of the page or return to the home page by clicking on the arrow located at the top left corner. The image below shows what the user would see if they chose the register component.

← Register

Create an Account

User Name

Email

Password

Confirm Password

By creating an account you agree to our
Terms of Service and Privacy Policy

CONTINUE

Figure 12. Register Page

D. Screen 4 - Main page

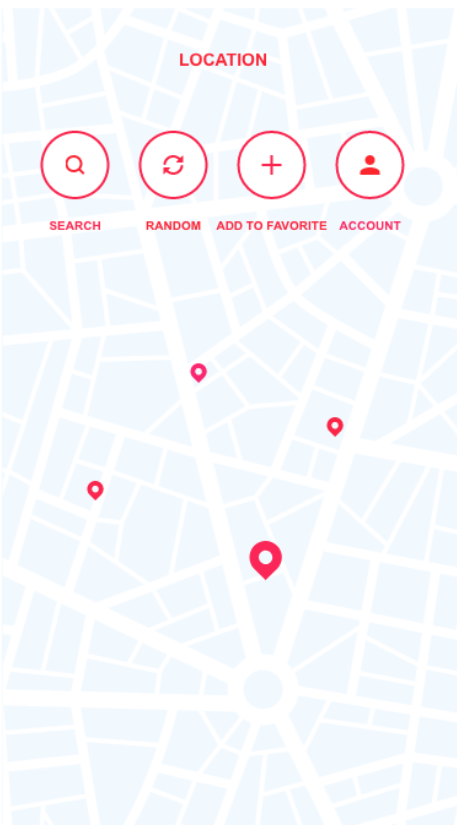
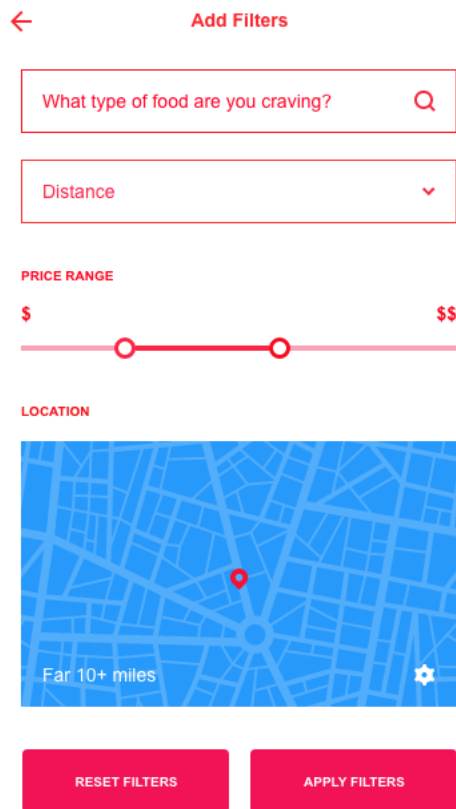


Figure 13. Main Page

This page will appear as soon as the user logs in or creates an account. The page displays the restaurants close by and four options that a user can perform : Search, Random, Add to favorite, Account. The “SEARCH” option, the first of the four options located at the top of the page, allow the user to search restaurants by creating filters on the search. The picture below shows what will appear after clicking on the Search option.



← Add Filters

What type of food are you craving? 🔍

Distance ▼

PRICE RANGE

\$ ————— \$

LOCATION

Far 10+ miles

RESET FILTERS APPLY FILTERS

Figure 14. Search Page

The user can start by inputting the desired type of food, choose other sorting options such as popularity, and price. After creating the filter the user can perform a search by clicking on the “APPLY FILTERS” option located at the bottom right corner. The user can reset to the default search filter, which is sort by distance, by clicking on the “RESET FILTERS” located on the top bottom left corner. The user also has the option to go back to the main page by clicking on the arrows located at the top left corner.

The “RANDOM” option, the second of the four options located at the top of the page, generates a restaurant using an algorithm. The user also has the option to go back to the main page by clicking on the arrows located at the top left corner.

The “ADD TO FAVORITE” option, the third of the four options located at the top of the page, allows the user to add a restaurant by entering the name and address. The user has an option to add restaurants after performing a search or when a random restaurant is generated. The user also has the option to go back to the main page by clicking on the arrows located at the top left corner.

The “ACCOUNT” option, the fourth of the four options located at the top of the page will display the users account. The page will have a username, email address and the favorites restaurants the user saved. The user also has the option to go back to the main page by clicking on the arrows located at the top left corner.

7. Conclusion

The design of this project has been helpful in determining the best technologies to use and how to implement the system's requirements. It helped us address the overview of the architecture and how our classes interacted with each other. It has also helped clarify some of the vagueness around particular features of the project that previously were still just ideas for the most part. The main difficulty of this project will be learning and integrating the different API used and implementing them correctly. There was also difficulty in implementing version control in the project; in the case of restaurant finder it was using GitHub. We have been attempting to mitigate this issue by learning all we can about the different architectures before programming and using online resources.

References:

For references please follow the IEEE guidelines shown below link:

<https://ieeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf>