Free Food Finder Project 6

Catherine (Yifan) Chen, Michelle Tran, Jett Crowson

Status Summary

In the first portion of the project, we have completed all of the necessary setup and configuration of our project. This includes generating and configuring the database, Spring Boot project, and Angular project. More specifically on each of these components, we have completed the following work:

Angular

- Implemented the google maps javascript API to display a map of CU Boulder
- Implemented UI elements such as the search bar, locations search bar (with autocomplete), and a modal for users to choose filters.
- Implemented the Database service, which utilizes the HttpClient to make calls to the spring service. This also includes a mock and implementation version of the service, where we use dependency injection to dynamically choose which version to use.
- Implemented pins on the map using the Google Maps api.
- Implemented an initial call (on page load) to get all free food events that are currently going on.

Spring

- Created the Spring Controller, and enabled it to handle GET requests for all food events
- Created Spring Service, and enabled it to handle GET requests for all food events
- Created most of the Spring Models (Location, LocationResponse, Event, EventResponse, DietaryRestriction)
- Connected the Spring project to the local database server based on the DDL

Database

- Generated DDL to initialize the database
- Implemented database with MySQL server
- Generated mock data to insert into the database
- Inserted mock data

Our Team Member Work Breakdown

	Description	J	M	C
Generate Angular Project	Use Angular CLI	✅		
Generate Spring Project	Look into spring boot project generator: https://start.spring.io/ Then make branch rule to prevent merge with main once this is done.		✅	
Database Implementation	Finalize database diagram, generate DDL from diagram. Save the script into a folder in the Spring repo			✅

Mock data	Create a script to insert a few rows of mock data into each of the tables. Save script into a folder in the Spring repo			✅
Angular Tour of Heros	https://angular.io/tutorial			✅
Spring REST Controller (api)	GET /free-food (so getting all free food events)		✅	
Spring database connection			✅	
Spring data models	Made the ones necessary for getting all free food. Have not made all the ones for filtering yet.		✅	
Spring service to retrieve all entries			✅	
Implement map	Look into Google Maps Angular component or free equivalent	✅		
Implement placing (mock) pins on map	Creating a pin component, place a few using mock data from Angular - If possible, should be clickable/display info			✅
Implement HTTP call in Angular		✅		
Route real pin data to display real pins		✅		

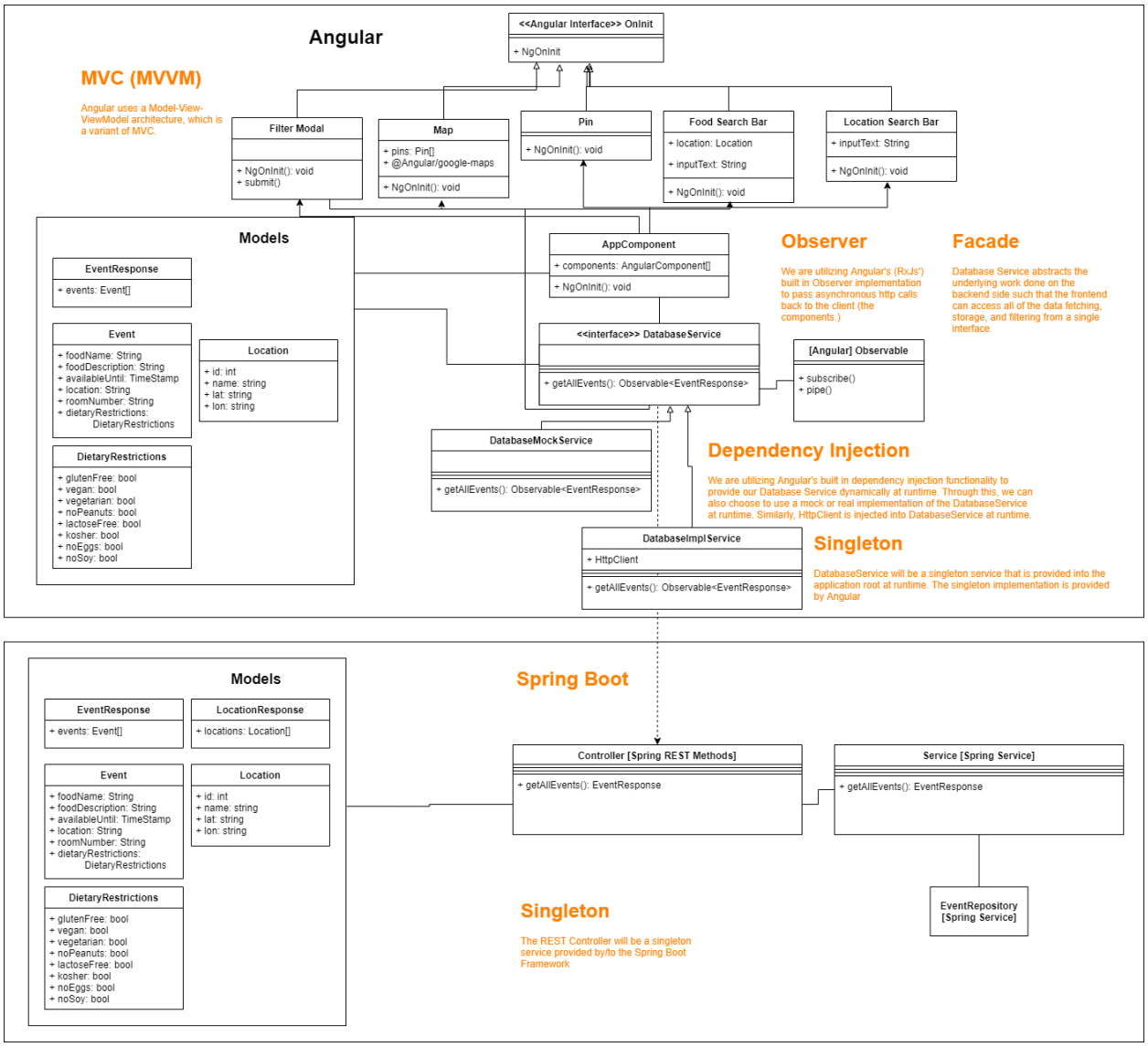
As far as changes go, we have been following our plan pretty closely, with the exceptions primarily being due to quirks with the frameworks that we are using. In Angular, one change was that we wrote a custom provider for the Angular Dependency Injection engine, and in Spring we needed to create an interface to implement the CRUD repository component of Spring to connect with the database. We also removed the DietaryRestrictionsResponse model from our plan, as we realized there's never really a case in which we would need to use it. Additionally, we have made small tweaks to our database, such as naming or allowing for 1-to-1 relationships between food events and rows in the dietary restrictions table.

As for patterns, we have used all of the patterns that we planned to use, including:

- ❖ Dependency injection
  - ➢ Used to provide either the mock or real version of the database service, and provide the HttpClient
  - ➢ Also used to inject Beans in our Java Spring Service
- ❖ Singleton
  - ➢ Used to create a single DatabaseService

- ➢ Used to create a single REST controller in Spring
- ❖ Observer
  - ➢ Used to send responses from the database to the clients (our components)
- ❖ Facade
  - ➢ Used to abstract the backend service to a simple API
- ❖ MVC (MVVM).
  - ➢ Used in Angular for project structure and internals

Class Diagram



Plan for Next Iteration

Now that we have the basic infrastructure for our interactive map, our main focus for the next sprint will be to add filtering capabilities to our app. This will entail the following developments:

Angular

- DatabaseService request to get all locations
- DatabaseService request to get food events, with filters
- Add Food button + modal
- DatabaseService request to POST new food event

Spring

- Adding an endpoint to the Controller & Service to handle GET food event requests with filtering parameters
- Adding an endpoint to the Controller & Service to handle POST requests to add food events
- Adding an endpoint to the Controller & Service to handle GET location requests
- Ensure that events returned are not expired (filter database items and delete expired rows)

Database:

- Create data for location table and insert into database
    - Need to include address attribute (optional)
- Make dietary restrictions → event association 1 to 1 instead of 1 to Many