Software Detailed Design

Table of Contents

6.3 Detailed CSC and CSU Descriptions Section

6.3.1 Detailed Class Descriptions Section

The following sections provide the details of all classes used in the Food Recovery Network (LMU chapter) website:

6.3.1.1 GetMap

This class is the Google Maps API wrapper that is used to produce the Map interface. The fields of this class include `startingLocation` and `endingLocation`. The `startingLocation` is at the Lair, which is Loyola Marymount University's main dining hall. The `endingLocation` is the location in which the route ends – currently being at St. Joseph's Center. The methods of this class include `generateMap()`. This method uses the Google Maps API to produce a map that showcases the route for deliveries.

6.3.1.2 User

This class calls the Firebase database to get the user information and stores all of the data about the user. This class contains the fields `email`, `name`, and `role`. The `email` field contains the email address of the user. The `name` field contains the actual, saved first and last name of the user. The `role` field contains the role of the user (e.g., student volunteer, staff, admin). This class also contains the methods `getUser()` and `updateUser()`. The `getUser()` method calls the Firebase to get any information that is stored in the fields. The `updateUser()` calls Firebase to update any user information that the user has changed.

6.3.1.3 FoodTracker

This class contains four fields: `trackerID`, `foodType`, `quantity`, and `deliveryDate`. The `trackerID` field sets a unique identifier for each food that is inputted by the user. `FoodType` contains the name of the food item (e.g., sandwich, salad, cupcake). The `quantity` field stores the amount of food picked up and delivered on a specific date. The `deliveryDate` field is the day in which a volunteer received and dropped off the food items. This class also contains the `addItem()`, `editItem()`, and `removeItem()` methods. These methods allow users to add another food item, edit their current entry, or remove a food item, respectively.

6.3.1.4 VolunteerSchedule

This class has `volunteer`, `pickupDate`, and `availability` fields. The `volunteer` field references the student volunteer assigned to do the FRN run on a specific date. `PickupDate` stores the date and time of the scheduled pickup and delivery. `Availability` has the user's preferred availability for FRN runs. For this class, there are three methods: `getVolunteer()`, `scheduleRun()`, and `editSchedule()`. `GetVolunteer()` retrieves all of the user's information, including their volunteering availability. The `scheduleRun()` method schedules a volunteer to do a run on a certain day and sends them information on what to do. Finally, the `editSchedule()` method allows users to edit the schedule in case they are no longer able to volunteer or want to add another volunteer date.

6.3.2 Detailed Interface Descriptions Section

The following sections provide the details of all interfaces used in the Food Recovery Network (LMU chapter) web application:

- User Interface (UI): Front-end
  - This interface allows users to either log in or sign up, and then stores all of the user input including their availability preferences, their scheduled sign-ups, and the food items they track for the days they volunteer. With all of these inputs on different pages, this interface then outputs a map of the delivery route, a schedule for the current semester, and a list of food items delivered on a certain FRN run.
- Google Maps API Interface
  - The Google Maps interface allows the user to view a graphical depiction of the delivery route created by the algorithms. This creates an accessible visualization for users to know how to get to the drop off location at St. Joseph's Center. If the placement is closed or a different placement is inputted, this interface will make this clear. The inputs for this interface are the starting and ending locations.
- Node.js Interface: Back-end
  - The Node.js interface manages the communication between the front-end and the back-end. The Node.js interface allows the back-end to access any information provided by the user, such as email / password, volunteer preferences, etc. Furthermore, Node.js allows me to send information from the APIs that I am utilizing to the front-end for the user to view.

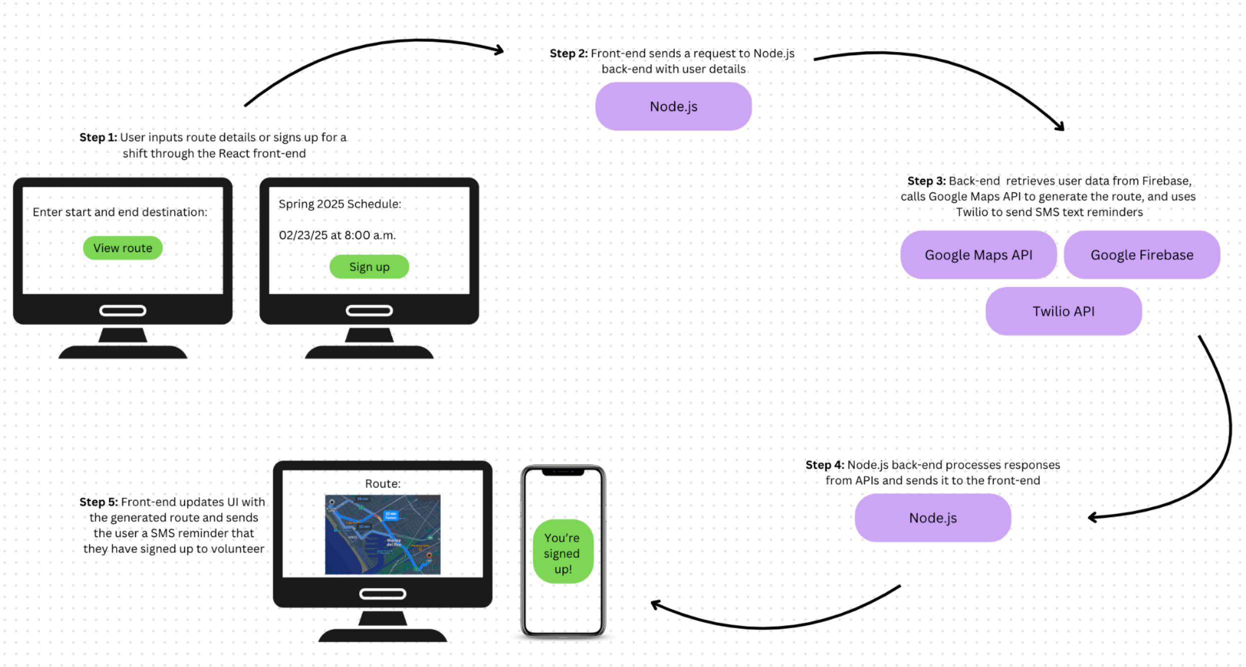
6.3.3 Detailed Data Structure Descriptions Section

The primary data structures that I use in this project are to store user information and the amount of food donated for this program. When a user creates an account, their information is sent to Google Firebase. User information includes their first and last name, email, password, and role. This data can then be queried by the software when needed, such as a returning user.

The specific data structures are Strings, and no other data format. In the FoodTracker protocol, I convert a list of the types of food items, quantity of each food, and date of the food drop-offs into a JSON format. This format is easily queried when I need to display the information about the list of food items donated each week for this program.

6.3.4 Detailed Design Diagrams Section

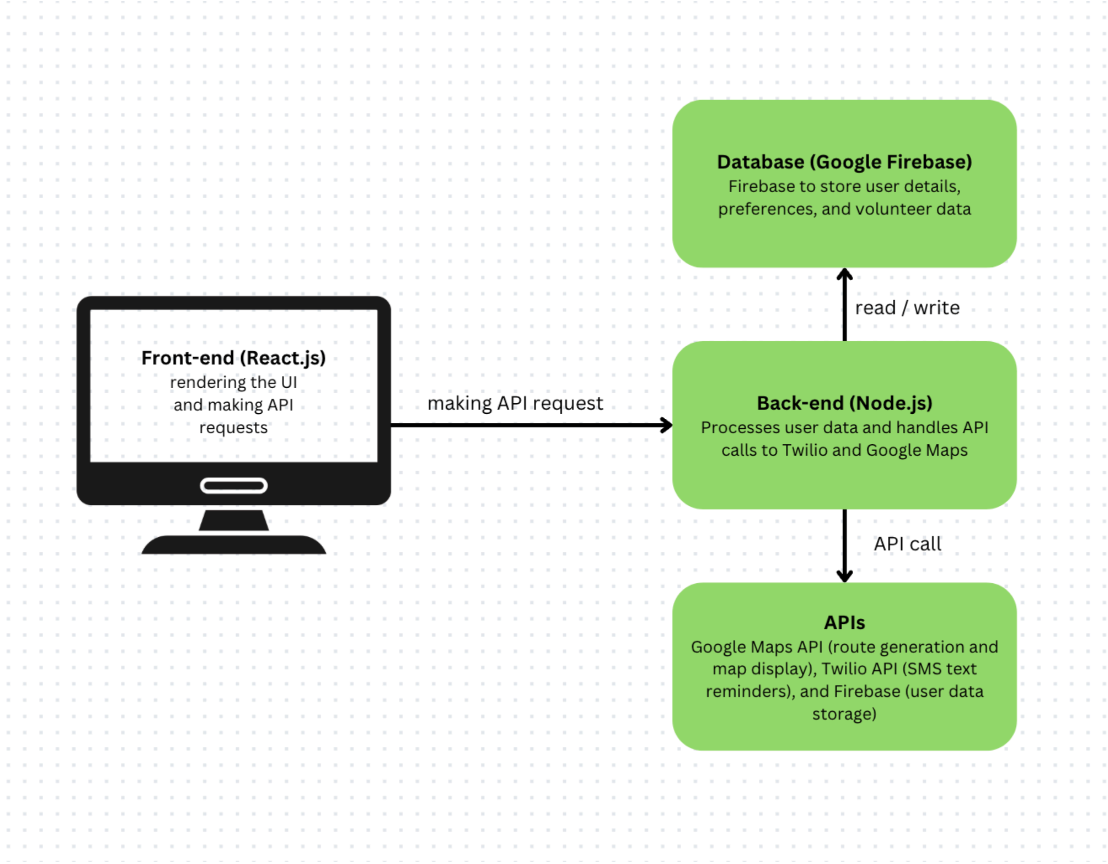
Concept of Operations Diagram


Step 1: User inputs route details or signs up for a shift through the React front-end
Enter start and end destination:
View route
Spring 2025 Schedule:
02/23/25 at 8:00 a.m.
Sign up
Step 2: Front-end sends a request to Node.js back-end with user details
Node.js
Step 3: Back-end retrieves user data from Firebase, calls Google Maps API to generate the route, and uses Twilio to send SMS text reminders
Google Maps API
Google Firebase
Twilio API
Step 4: Node.js back-end processes responses from APIs and sends it to the front-end
Node.js
Step 5: Front-end updates UI with the generated route and sends the user a SMS reminder that they have signed up to volunteer
Route:
You're signed up!

The concept of operations diagram for LMU's Food Recovery Network website outlines how data flows through the system. The process begins when a user interacts with the React front-end, entering route details for food deliveries or signing up to volunteer. This triggers an HTTP request to the Node.js back-end, processing the request and interacting with external APIs and databases. The user first inputs route details, signs up for a shift, or tracks the amount of food delivered through the React front-end. The front-end then sends a request to the Node.js back-end with user data.

The back-end gets user data from Firebase, uses the Google Maps API to calculate the route, and utilizes the Twilio API to send SMS updates to confirm the volunteer's participation. The Node.js back-end processes the responses from both APIs and sends the route and confirmation data back to the front-end. Finally, the front-end updates the UI to display the delivery route and sends users confirmation texts. This ensures a seamless and interactive experience for all members of the Food Recovery Network program at LMU.

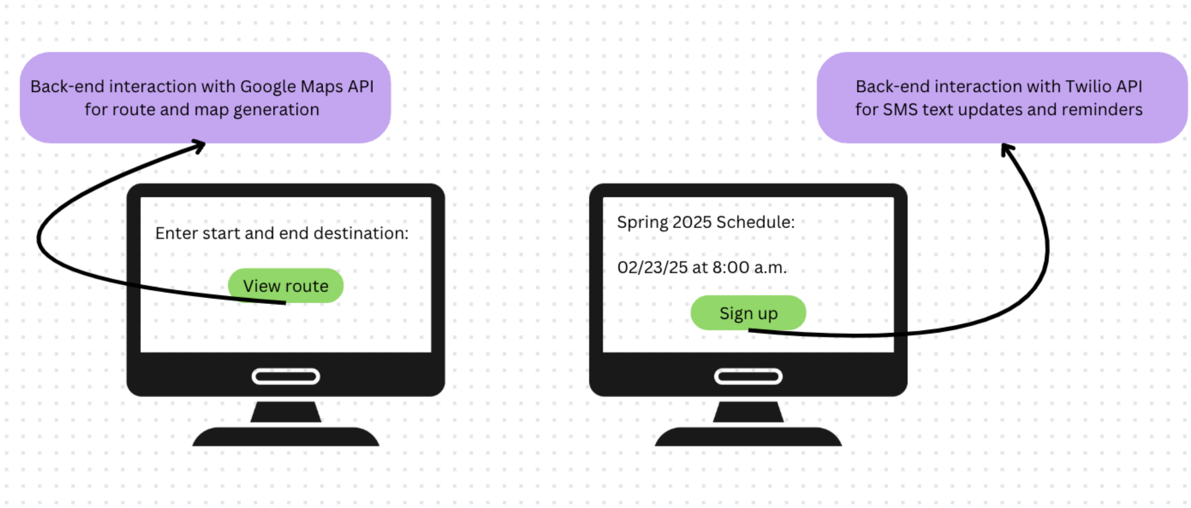
Interface Flow Diagram


Database (Google Firebase)
Firebase to store user details, preferences, and volunteer data
read / write
Front-end (React.js)
rendering the UI and making API requests
making API request
Back-end (Node.js)
Processes user data and handles API calls to Twilio and Google Maps
API call
APIs
Google Maps API (route generation and map display), Twilio API (SMS text reminders), and Firebase (user data storage)

The interface flow diagram for LMU's Food Recovery Network website begins with a computer representing the front-end (React), which is responsible for rendering the UI and making API requests. This front-end communicates with the backend (Node.js), indicated by an arrow labeled "making API request." The back-end processes user data, handles API calls to external services like Google Maps and Twilio, in addition to structuring JSON outputs.

From the back-end, there is an arrow marked "API call" pointing to a box representing external APIs, including the Google Maps API (for route generation and map display), the Twilio API (for SMS updates and reminders), and Firebase (for user data storage). Additionally, the backend connects to a separate database box labeled Google Firebase via an arrow called "read / write," used for storing user details, preferences, and volunteer data.

Front-End and Back-End Responsibilities Diagram

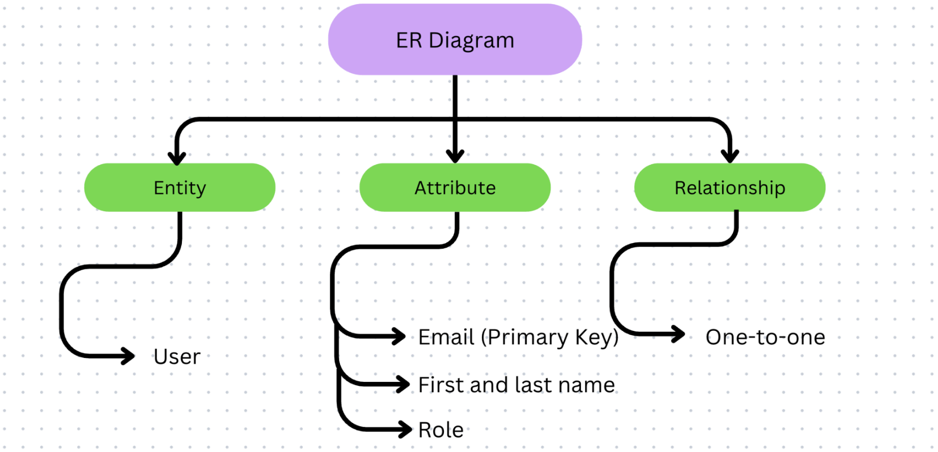

Back-end interaction with Google Maps API for route and map generation
Enter start and end destination:
View route
Back-end interaction with Twilio API for SMS text updates and reminders
Spring 2025 Schedule:
02/23/25 at 8:00 a.m.
Sign up

This diagram for the Food Recovery Network (LMU chapter) website shows a computer interface where volunteers are able to input route details by entering the pick-up and drop-off locations and observe the generated route after clicking on the "View Route" button. This triggers back-end interaction with the Google Maps API to calculate and display the optimal route.

The second computer seen in this diagram has a "Sign Up" button where volunteers can register for upcoming shifts by selecting the available dates. Once they sign up, the back-end interacts with the Twilio API to send SMS text reminders, confirming their registration and sending brief instructions on what to do. This setup ensures that volunteers can easily plan their food recovery shifts.

6.4 Database Design and Description Section

6.4.1 Database Design ER Diagram Section



6.4.2 Database Access Section

Users will have access once they either log in or sign up and create an account with the Google Firebase database. When users input their availability or track the amount of food donated on a FRN run, they will be able to access the trip data and can also access data from previous trips.

6.4.3 Database Security Section

Firebase will be used for authentication to control user access in Voyage. Only authenticated users will be able to view their trip information. Firebase automatically encrypts data and will be used to find any errors or unauthorized access attempts.