To run the project successfully, where both the frontend and backend are designed to connect to an online server while also being available locally, you would need to set up a local server acting as a middleman. This local server will route requests appropriately: directing the frontend to the online APIs and the backend to the online database.

Here are the steps to execute the project:

Setting Up a Local Server as a Middleman

Backend Configuration:

- Configure the backend to make API calls to the online server's base URL (https://qa2.sunbasedata.com) rather than using a hardcoded URL.
- Modify the backend API endpoints to call the online database for data operations.

Frontend Configuration:

- Configure the frontend to make API calls to the local server (localhost) for backend interactions.
- Adjust the frontend code to utilize local endpoints (e.g., http://localhost:9090) for API requests.

Local Server Setup:

- Set up a local server (e.g., using Node.js with Express) to act as a middleman.
- Create proxy routes to redirect frontend requests to the online server and backend requests to the online database.

Routing Configuration:

- Define routes on the local server that forward API requests:
    - Routes matching the frontend's requests should forward to the online server.
    - Routes from the frontend that pertain to the backend should route to the local backend (localhost:9090).

Testing:

- Start the local server to act as a middleman.
- Run the frontend, configured to communicate with the local server.
- Ensure the backend interacts with the online database and connects to the local server for handling API requests.

Running the Application

Once the local server is up and running as the intermediary between the frontend and backend, follow these steps:

Start the Local Server: Run the local server that manages the routing between the frontend and backend.
Launch the Backend: Run the backend server (configured to use the online API endpoints) on localhost:9090.

Initiate the Frontend: Open the frontend application on a browser, configured to communicate with the local server for API requests.

By following these steps, you can set up a local server to act as an intermediary, enabling both frontend and backend components to interact with their respective online endpoints while being accessible locally.

API Integration: The project involves integrating several APIs provided by SunBase Data. These APIs allow functionalities like user authentication, creating, retrieving, updating, and deleting customer data.

Frontend Development: The primary focus has been on creating a basic frontend using HTML and JavaScript to interact with these APIs. The interface includes three main screens: Login, Customer List, and Add New Customer.

Login Screen: The application starts with a Login screen where users can provide their login credentials. Upon successful authentication, the application fetches a token needed for subsequent API calls.

Customer List Screen: After successful authentication, the application displays a Customer List screen. It fetches a list of customers using the appropriate API and displays the data in a table format.

Add New Customer Screen: Users can add new customers by filling out a form with the required fields. Upon submission, the data is sent to the API for creating a new customer.

CRUD Operations: The application allows CRUD (Create, Read, Update, Delete) operations on the customer data. Users can delete and update existing customers by clicking buttons within the Customer List screen.

Token-Based Authentication: The application uses token-based authentication by fetching a bearer token from the authentication API and including it in the headers of subsequent API requests.

Error Handling: Basic error handling is present in the application, notifying users about authentication failures, missing data, or unsuccessful API calls through simple alert messages.

Note:

However, it's important to note that the backend and frontend are not hosted online. The backend is linked to an online SQL server, and the connection details, such as server address, credentials, and database details, are configured within the application.settings file.

Additionally, the application's basic functionalities, such as user authentication, CRUD operations for customer data, token-based authentication, and error handling, have been implemented in the project.