DATA 236 Sec 11 Distributed Systems
LAB-1
UBER EATS CLONE

Group-5
Sheetal Patnaik-017526678
Aishwarya Gulab Thorat- 017557579

1.Introduction

- Purpose: The main aim of this project is to build a prototype that is similar to UberEats as a food ordering and delivery platform. This involves building RESTful services with Django(backend) and React(frontend).
- The main goal here is that it simulates two primary roles: Customer and Restaurant.
- For **Customer** we have creating their own profile (account management), browsing restaurants, viewing menus, adding items to a cart, and placing orders.
- Similarly for **Restaurants** we have account management, adding menu items, viewing and managing customer orders, and updating order statuses.

2.System Design

Architecture:

- **Backend**: For backend we have used Django, we divided into multiple apps like accounts and ubereats_clone. The **accounts** app had functionalities which handles user authentication and profiles, while **ubereats_clone** handles the main project configuration.
  a. In backend, we have components as models.py, serializers.py, views.py, forms.py, urls.py , signals.py, settings.py which contains all the project settings, including installed apps, middleware, and database configurations. We also have **urls.py** which is the main routing file that connects to the **accounts** app and all other apps.
- **Frontend**: The React frontend is used in order to handle the UI, state management, and API interactions with Django.

Customer-Related Components:

a. **CustomerOrdersPage.js:** It displays orders placed by the customer along with the status updates. We have used CustomerOrdersPage.css for styling.
b. **DishesList.js:** This is the primary component in the customer's ordering process. It shows us all the dishes that are available for the selection in a restaurant's menu.
c. **OrdersPage.js:** This manages the order summary and checkout process. For styling we use **OrdersPage.css** styles.
d. **Profile.js:** This is the customer profile management which displays all the user information and also has options to edit the details too.
e. **RestaurantList.js:** This displays the list of all the restaurants which are available which helps the customers to select where they want to order from. We have styled it with RestaurantList.css.

Restaurant-Related Components:

a. **RestaurantDashboard.js:** This displays the key management options for the restaurant users which includes the menu and most importantly the order management.
b. **RestaurantOwnerProfile.js:** This allows all the restaurant owners to manage their profile information, and we have styled it with RestaurantOwnerProfile.css.
c. **RestaurantList.js:** As we did for the customers, it is similar, but it enables restaurants in order to view their public profile and menu items.
d. **RestLogin.js:** This manages the restaurant login page, and we have used RestLogin.css for style.

Common Components:

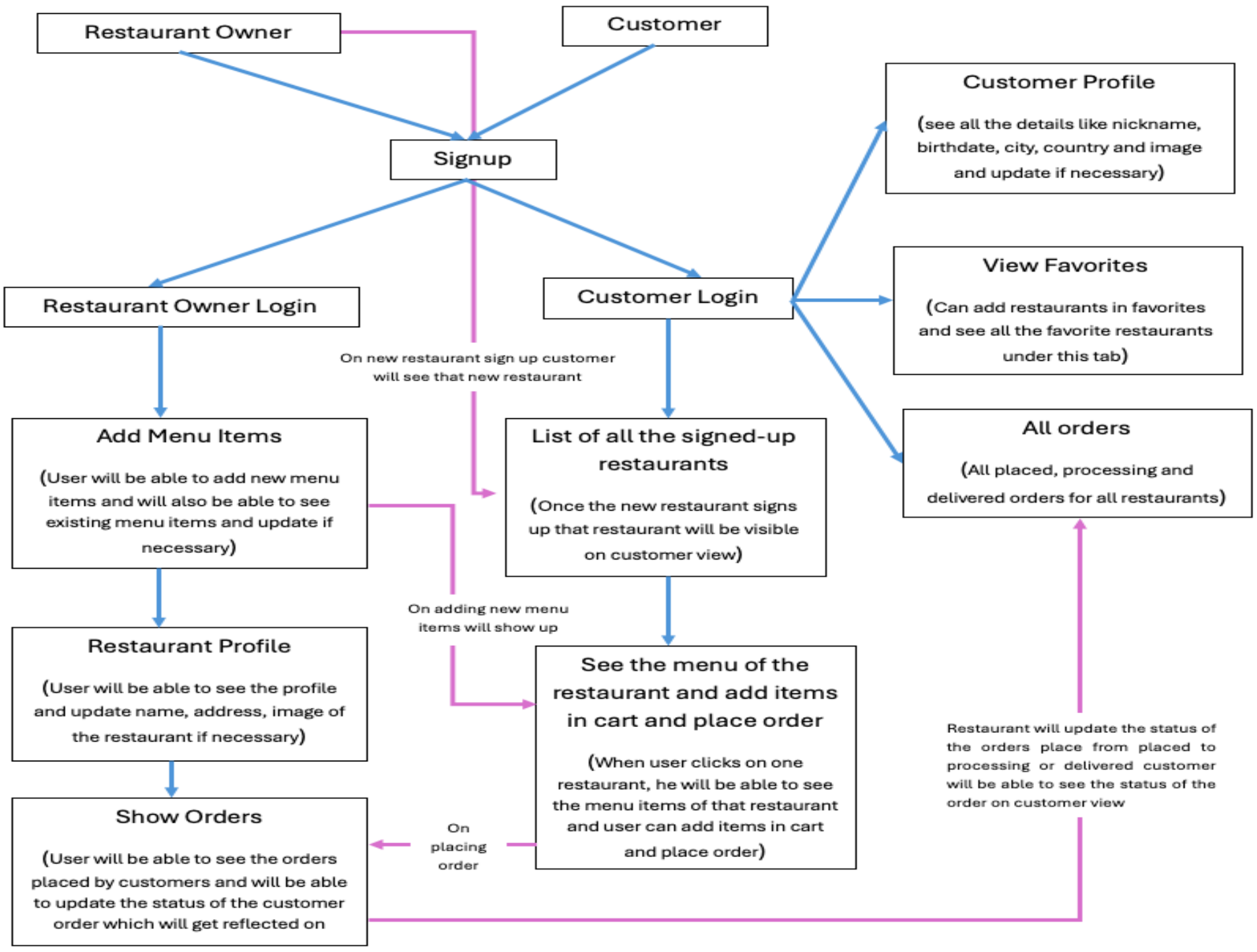
a. **Login.js** and **Signup.js**: These both handle the initial steps which is the user authentication, providing forms for login and registration. We have **Login.css** and **Signup.css** style for these respective forms.

b. **App.js:** This is the root component which wraps all the pages and handles the global navigation throughout. We have used App.css for the main application styling.
c. **index.js:** This is the entry point for the React application, rendering the App component into the root element in index.html. We have **index.css** which manages the global styles such as body font, color schemes, and spacing, applied across the application.

Static Assets:

a. **menu3.png, login_cust.jpeg, rest_login.jpeg, order3.jpeg, and rest_profile1.jpeg:** In order to display the images and the placeholders or few of the example images within the interface we have used.
b. **axios.js:** This handles the HTTP requests to the backend, defines base URL, headers, and any interceptors for authorization tokens or error handling.

Flow of Application



Restaurant Flow

1. **Restaurant Signup:** As we land on our page, we have to select either customer or restaurant from the dropdown and by clicking on restaurant we begin. The restaurant signing up provides information such as the restaurant name, email, password, and location. This creates an account for the restaurant on the platform.
2. **Restaurant Login:** After signing up, the restaurant logs in using the required credentials. Successful login will grant the access to the restaurant's dashboard.
3. **Add Menu Items**: After logging in, the restaurant can add items to its menu, specifying details like dish name, description, price, and category (e.g., appetizers, main course). As soon as we add the items here in the menu of restaurant even the customer can view the same in their dashboard under particular restaurant name.
4. **Restaurant Profile**: This allows the restaurant to manage its profile details like contact information, location etc.
5. **Show Orders**: The restaurant can view incoming orders placed by customers with unique Order ID.
6. **Order Processing and Delivery Status Updates**: Here the restaurant owner who has signed up has the authority to update the status of the order associated to its own restaurant as various

stages which is Received, Preparing and Delivered. This gets reflected in the customer dashboard also.

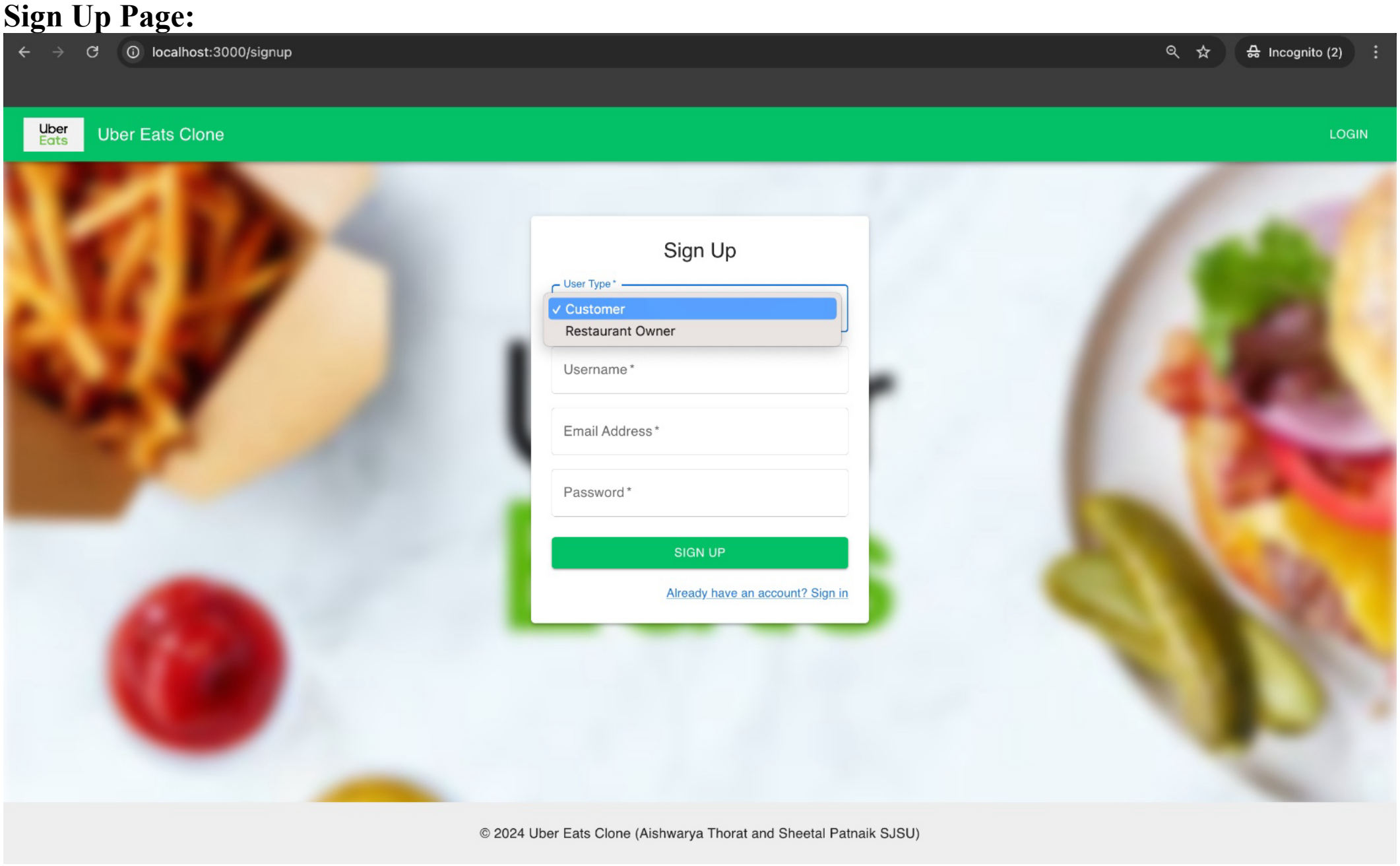
Customer Flow:

1. **Customer Signup:** As soon as the customer lands the opening page and stays on that page, the customer creates an account by signing up with personal information which includes name, email, and password.
2. **Customer Login:** Once the customer logs in using their credentials they can access the main dashboard and begin browsing restaurants.
3. **View Restaurant List:** The customer can now view a list of available restaurants on the platform. Each restaurant listing provides basic information, and the customer can select a restaurant to view its menu.
4. **Select Restaurant to View Menu**: In order to see the menu where all available items along with their details will be mentioned, the customer needs to select that particular restaurant.
5. **Add Items to Cart and Place Order**: In order to place order, the customer has to add items by quantity and upon finalizing, the customer can click on Place Order button to successfully place the order.
6. **Show Order Status**: In order to track the status of the order, the customer can view it on this page. As there are changes in the restaurant order tracker, same progress is showed here also.
7. **Favorites**: In scenario where customer wants to add few restaurants from the dashboard into favorites then there is heart to click on the restaurant name which adds that in the favorites list.

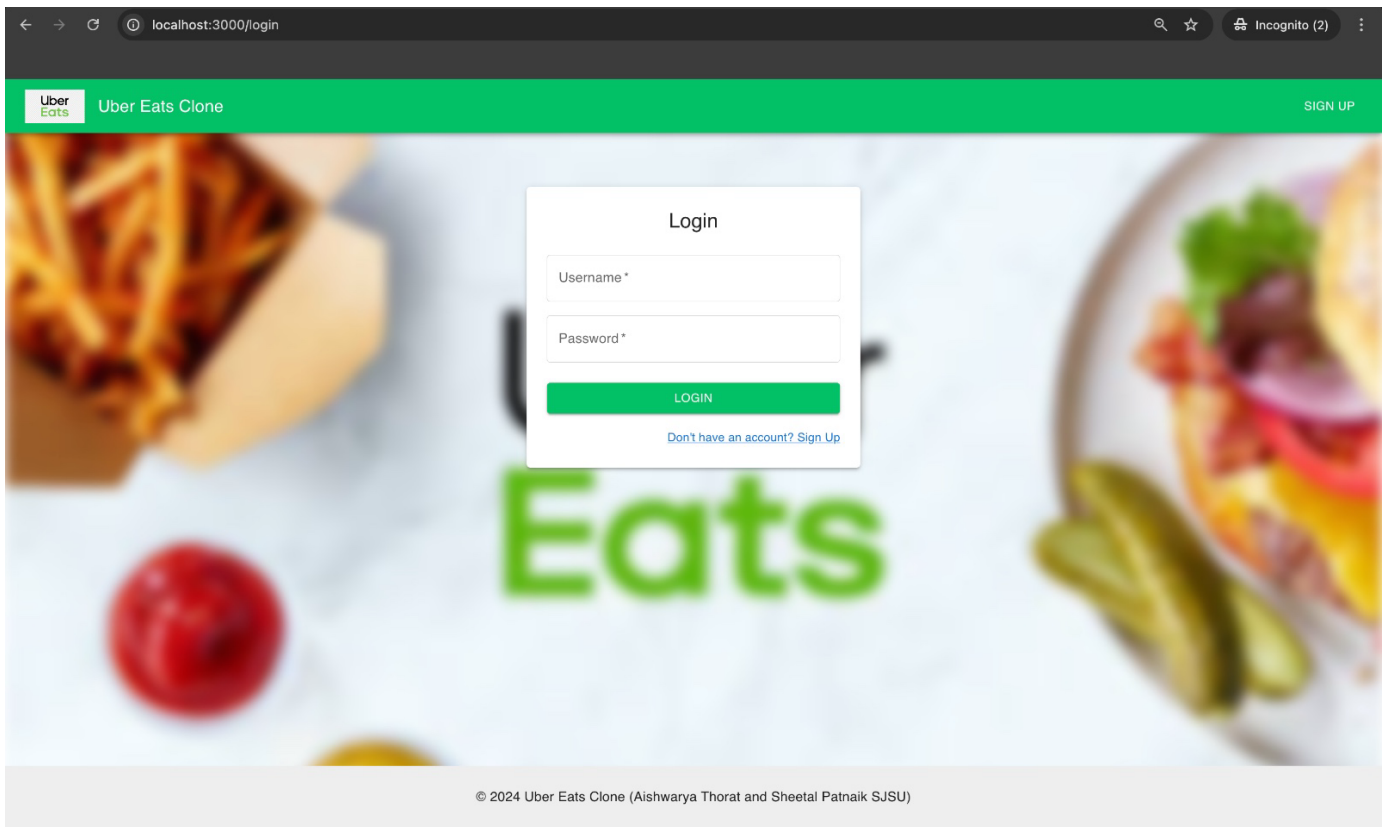
3.Results

a. Customer and Restaurant Registration and Login:

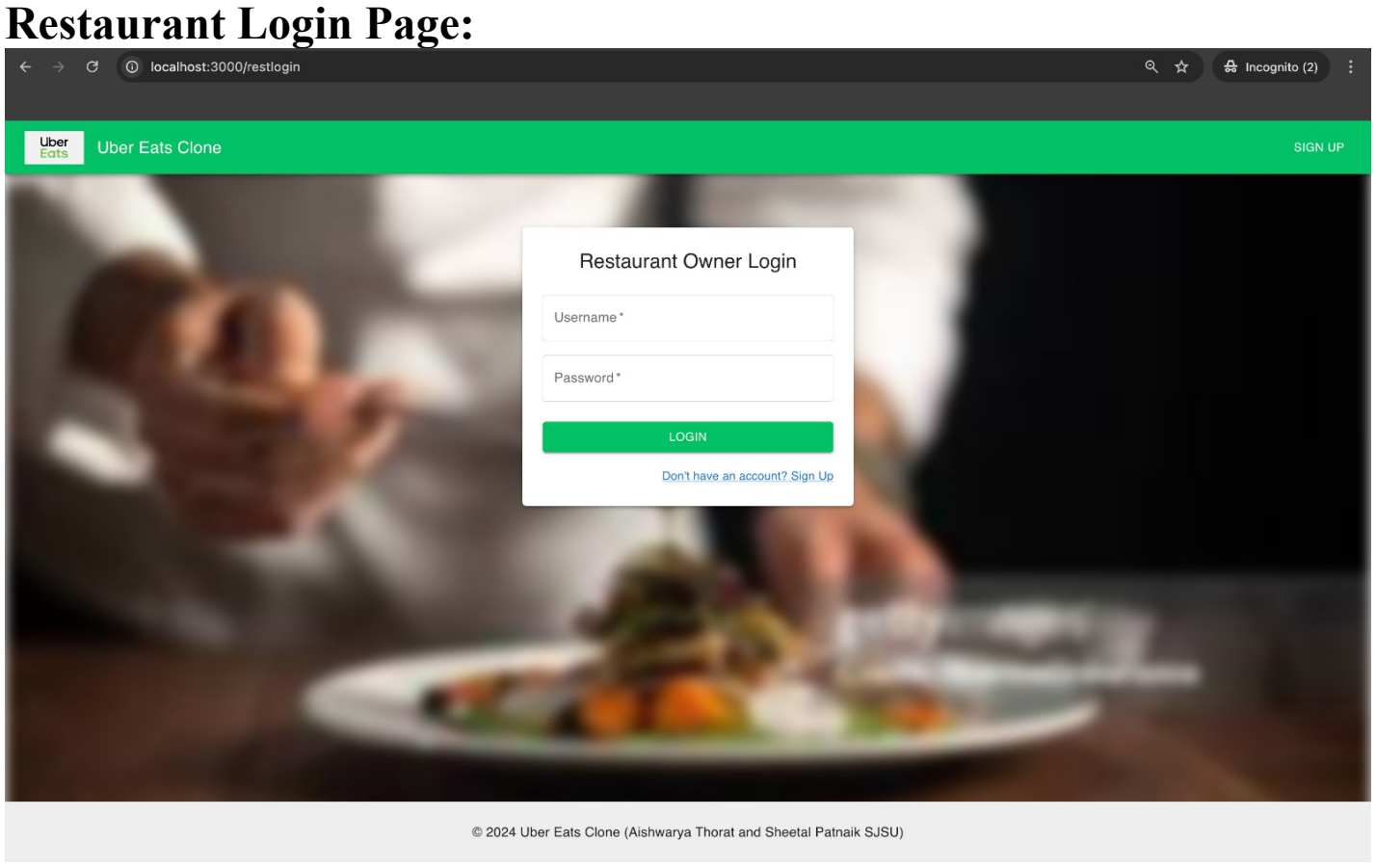
Sign Up Page:



Customer Login Page:

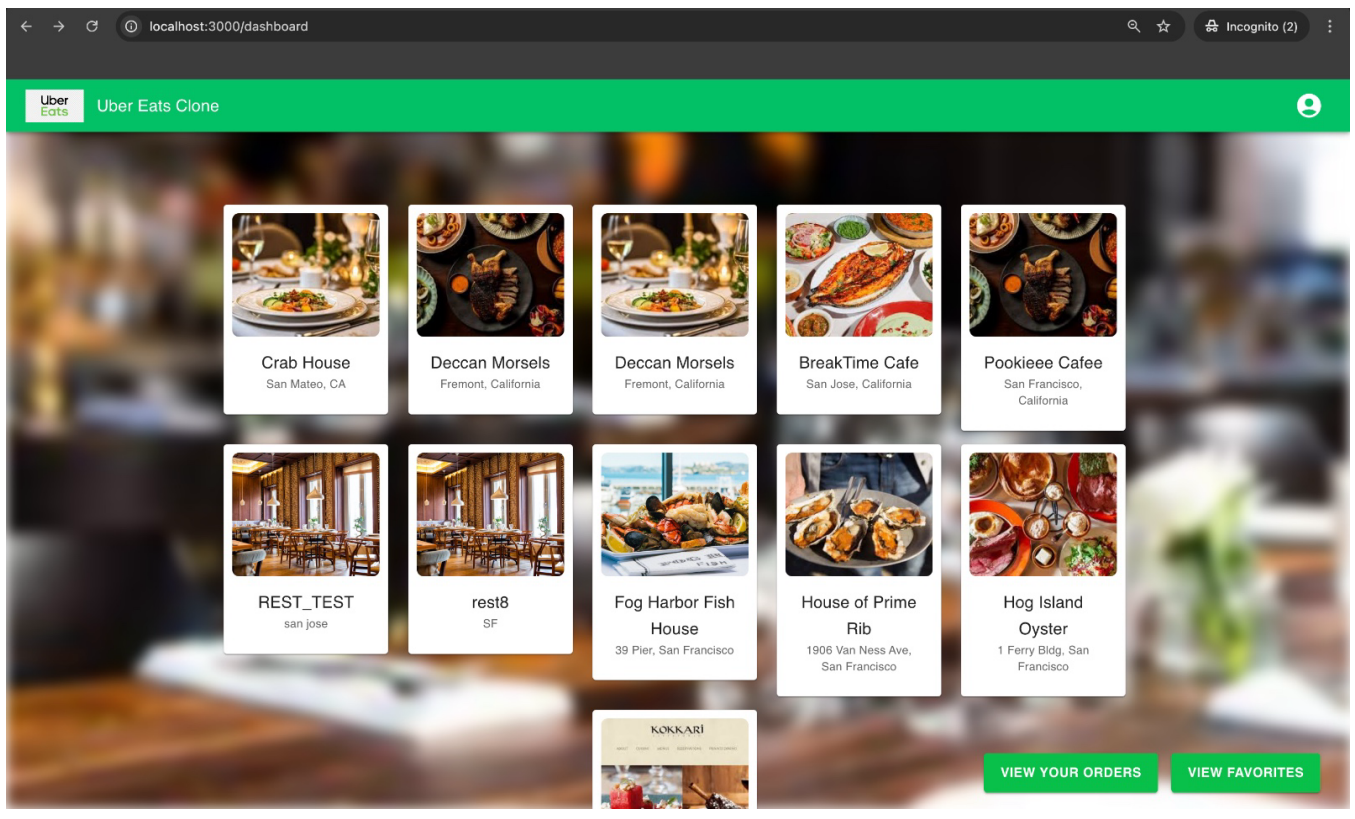


Restaurant Login Page:

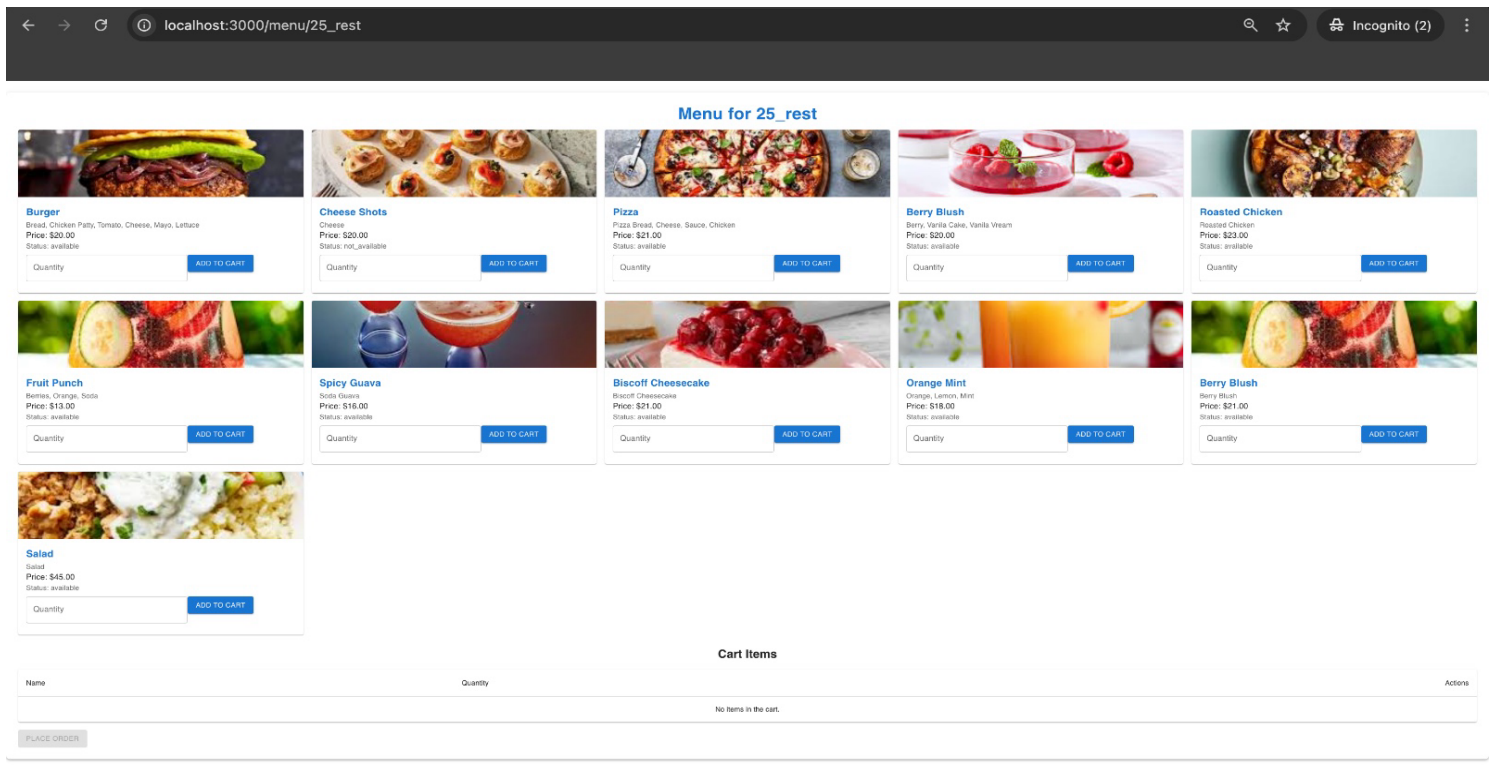


b. Customer Restaurant Browsing and Ordering:

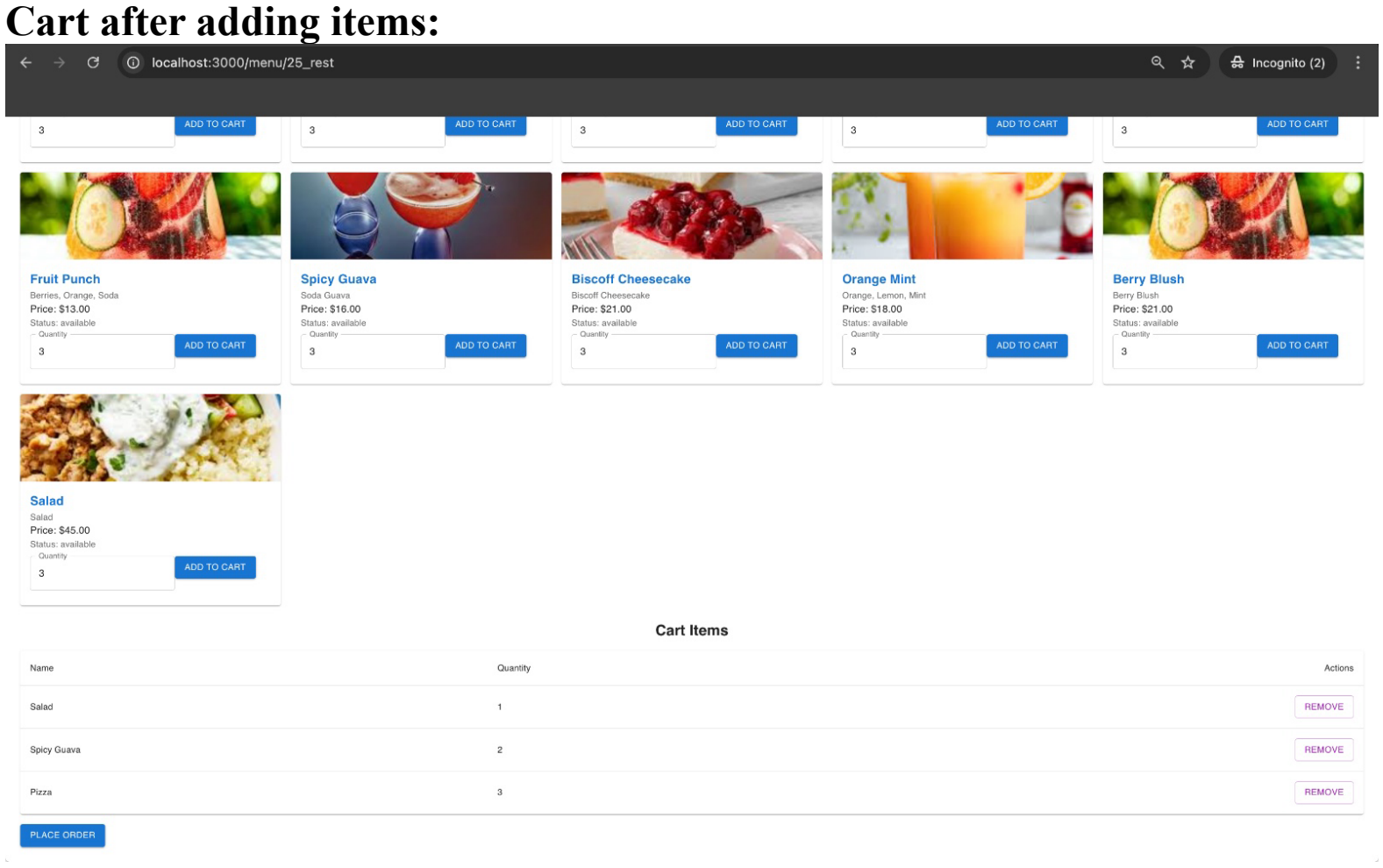
- ## Restaurant List:



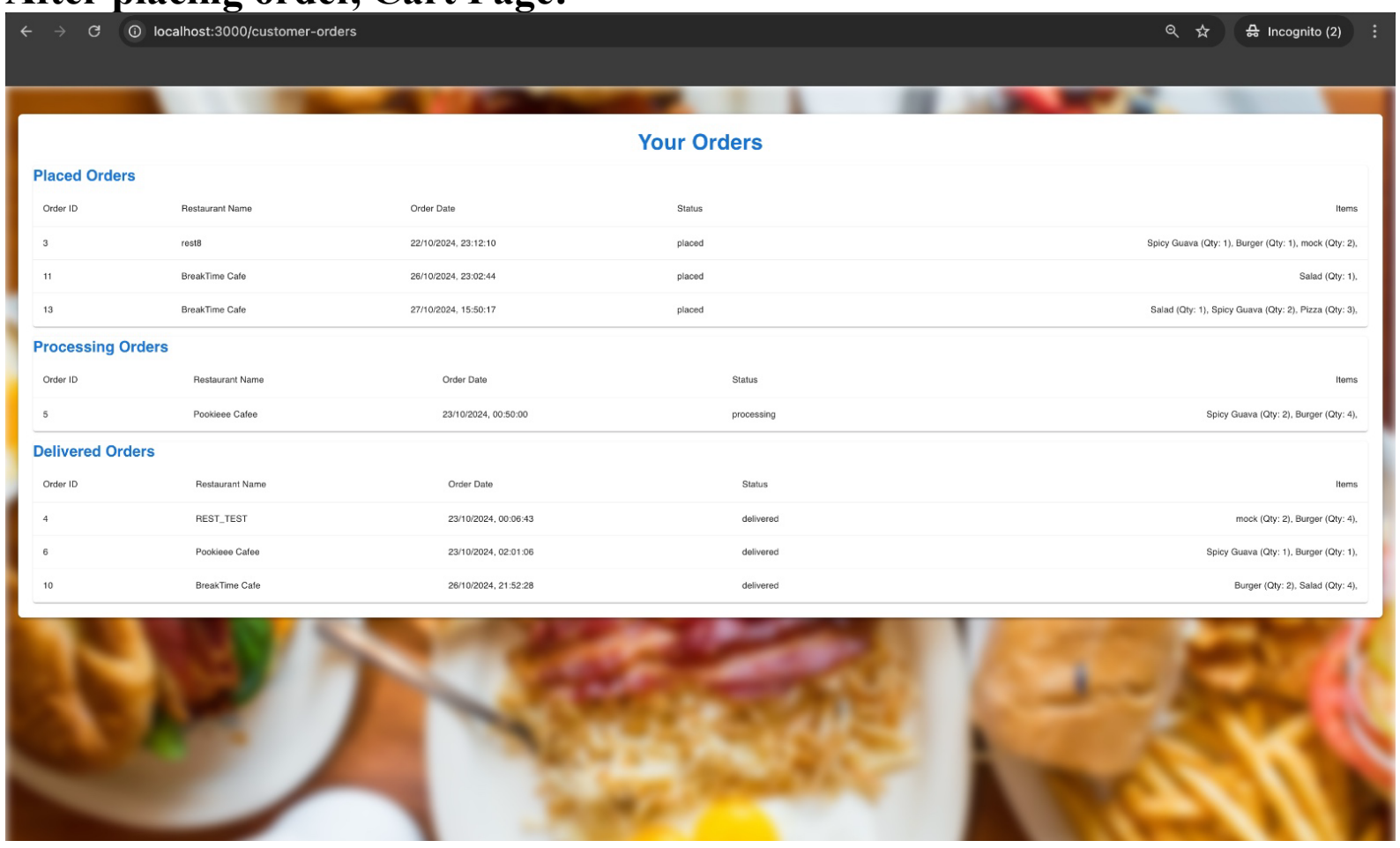
- ## Menu Page:



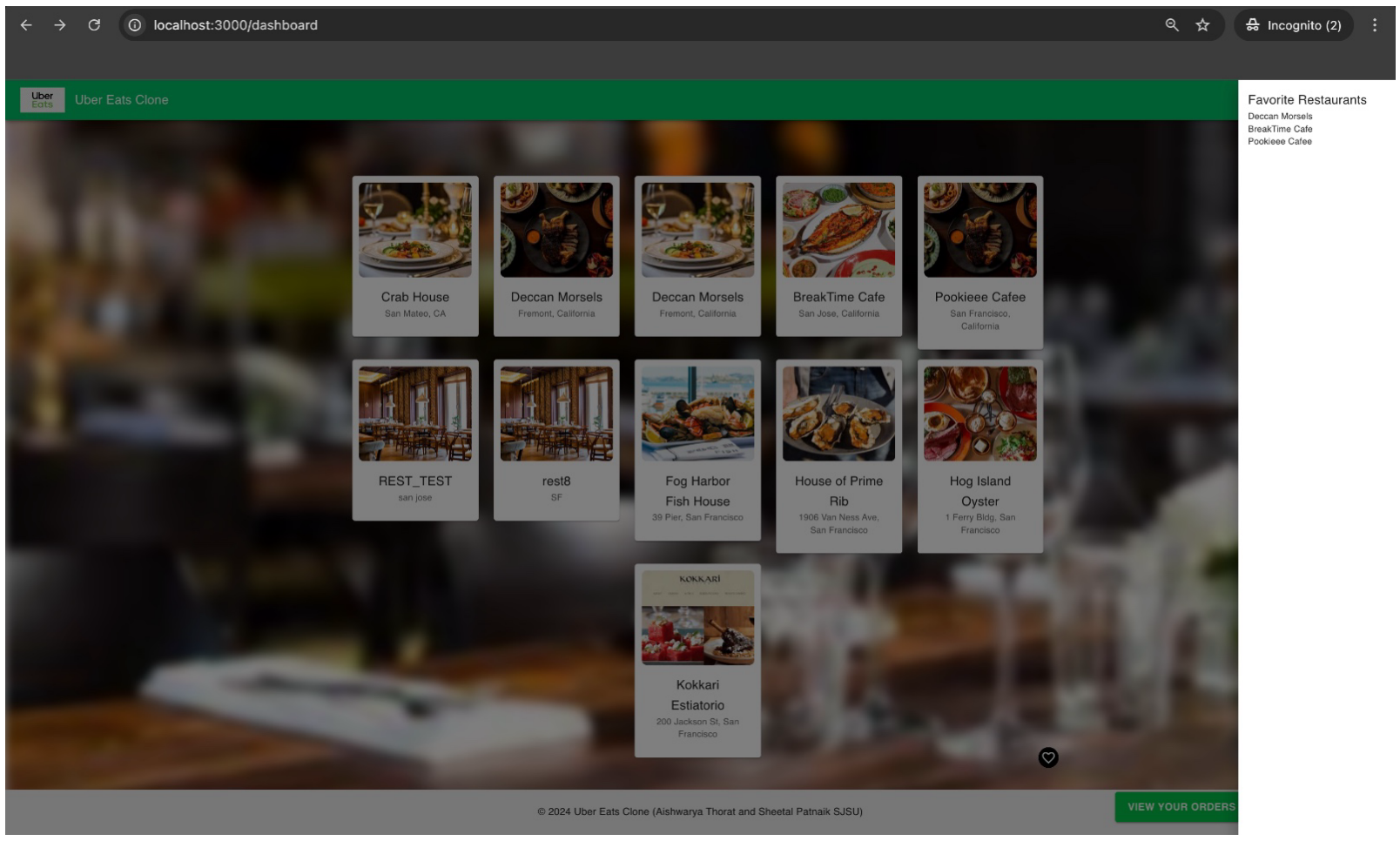
- **Cart and Checkout:**
  Cart after adding items:



After placing order, Cart Page:



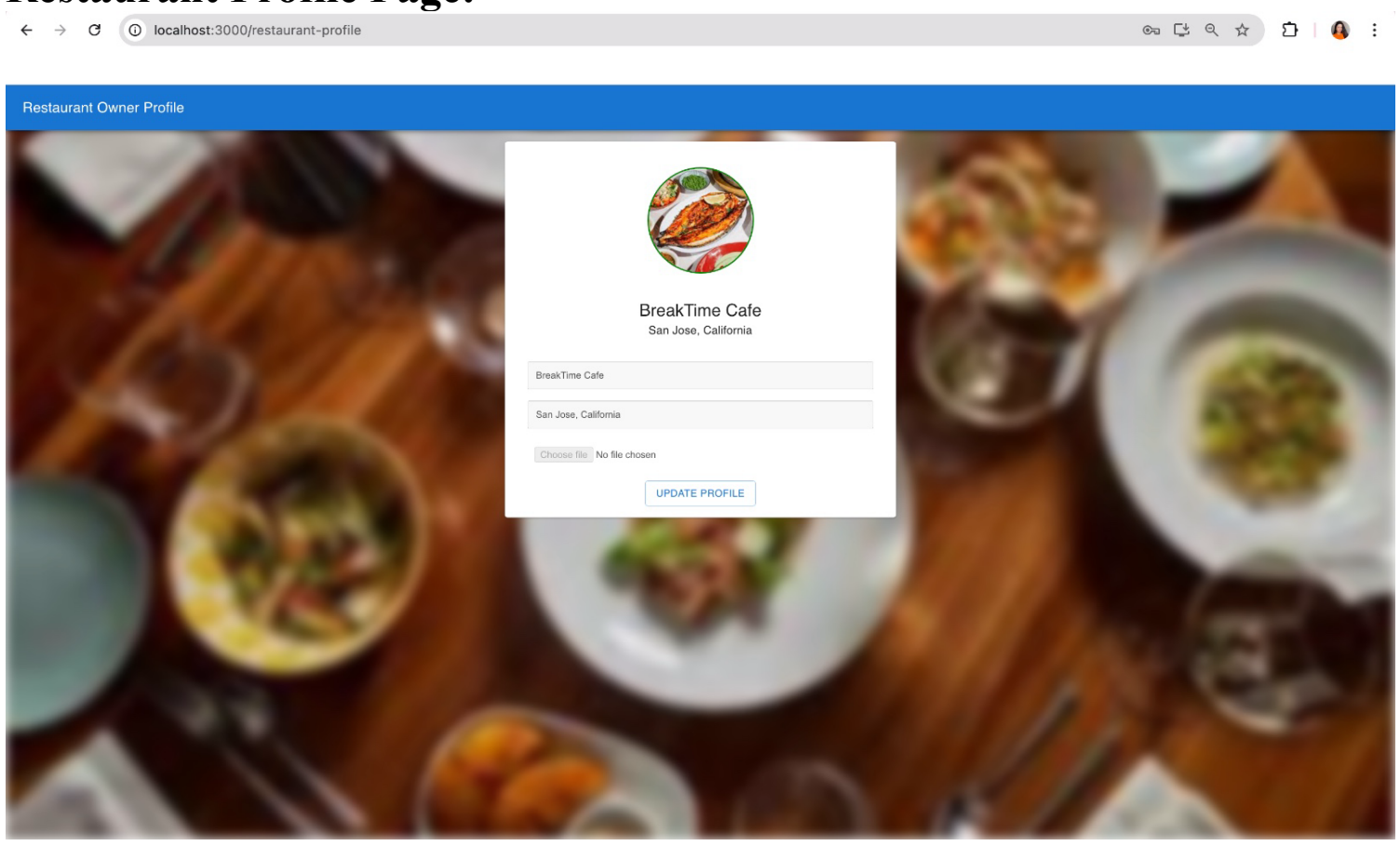
Favorites Page:



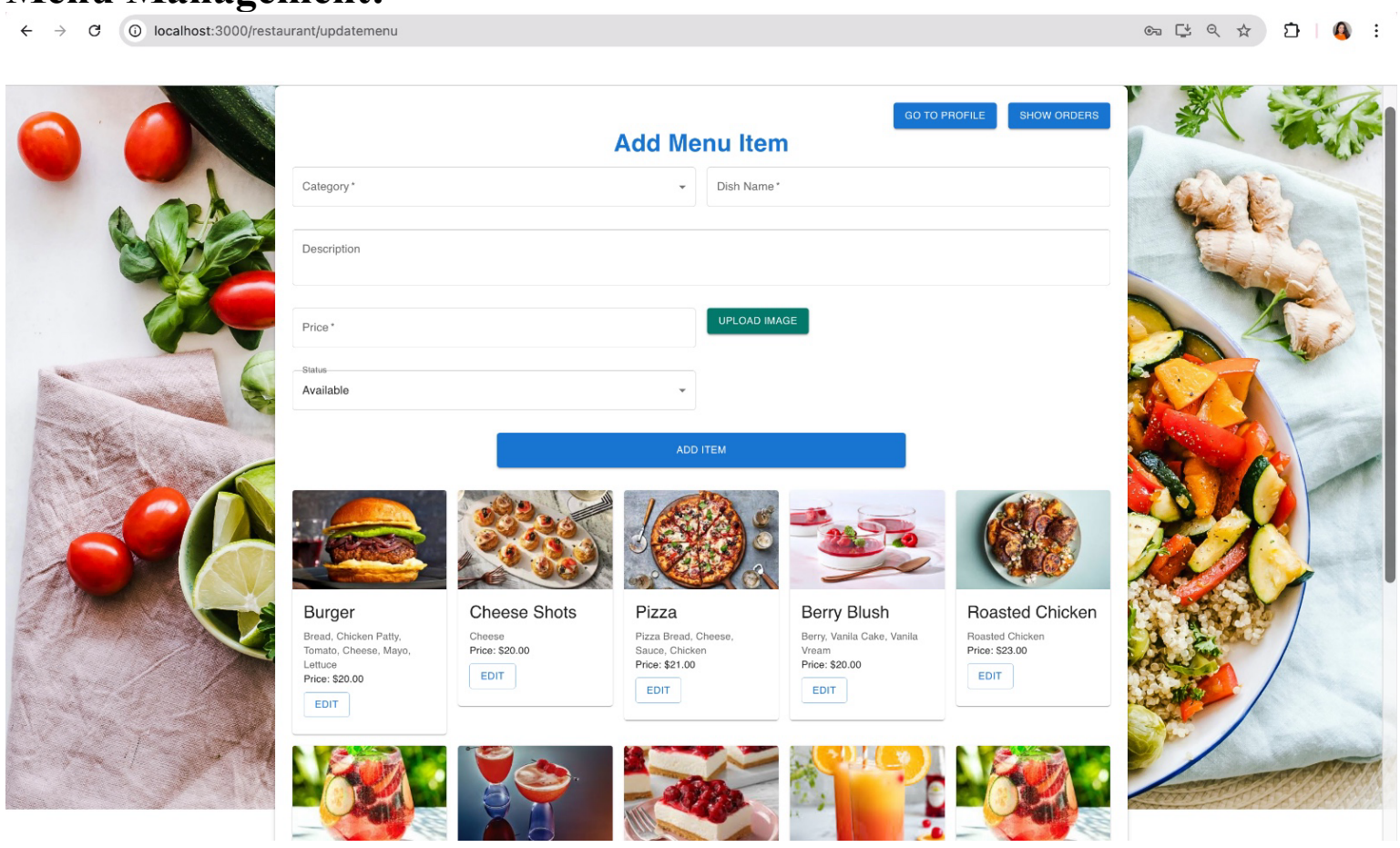
c. Restaurant Dashboard:

- **Order Management:**

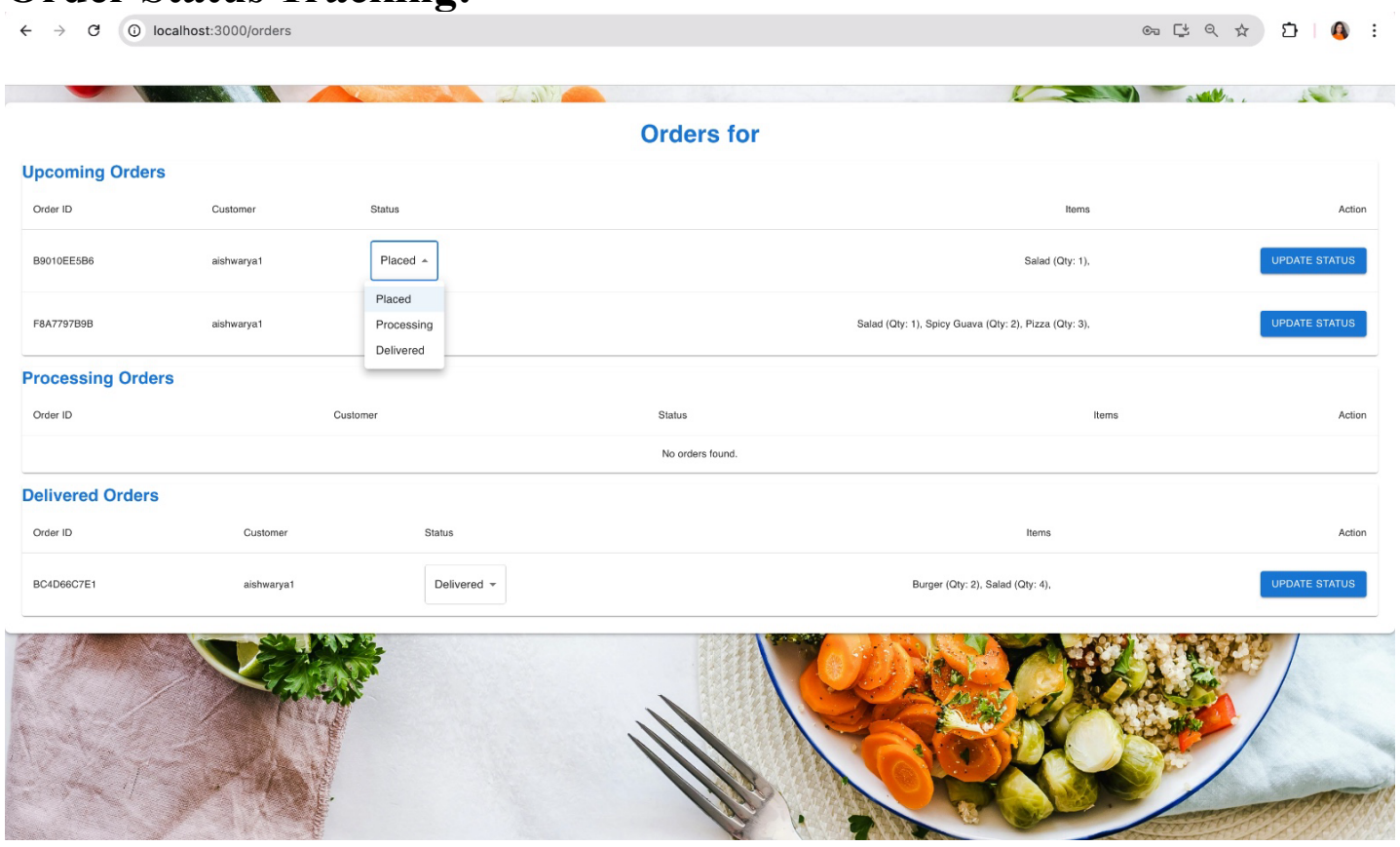
Restaurant Profile Page:



- **Menu Management:**



d. Order Status Tracking:



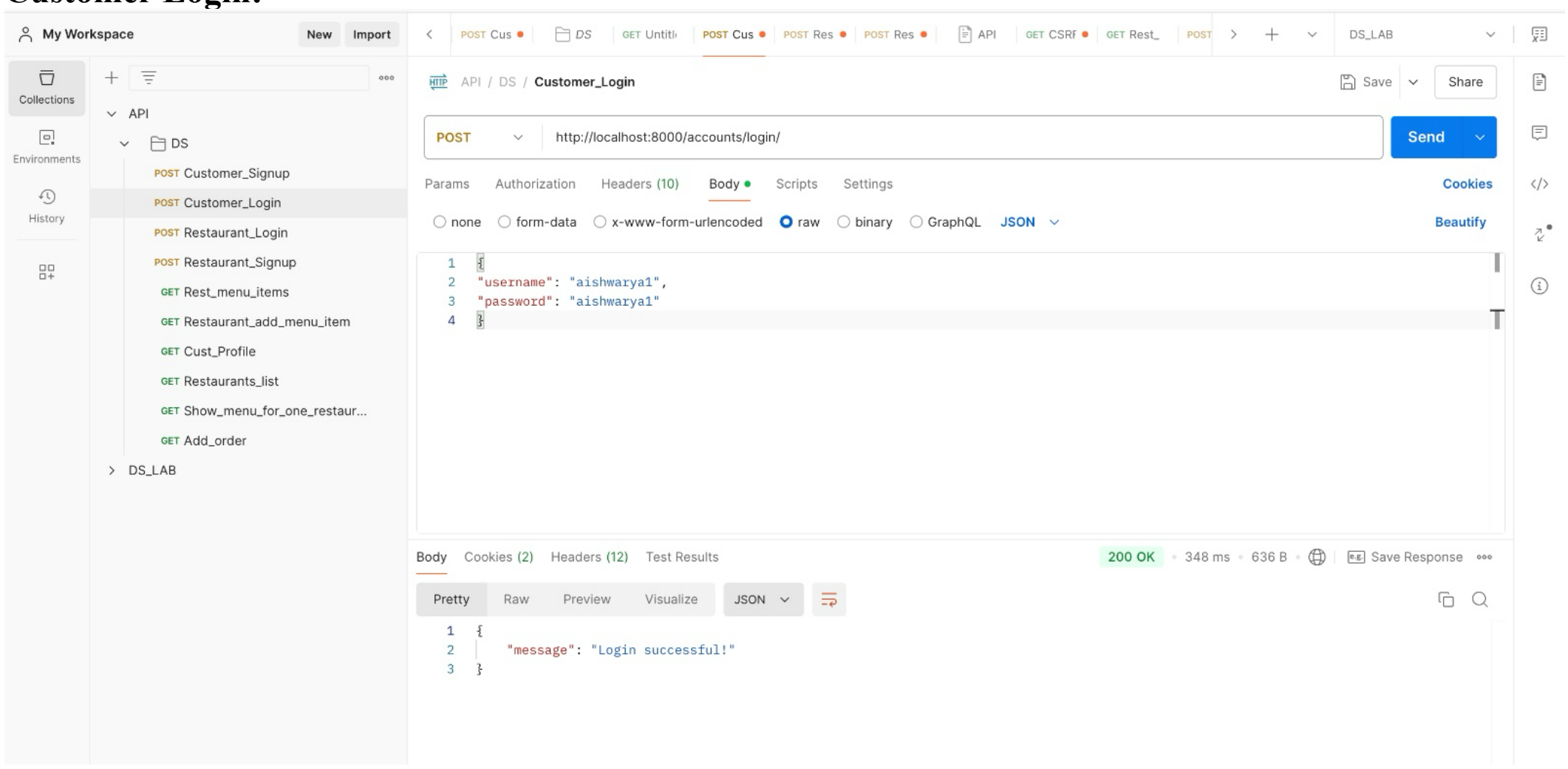
4.Performance

a. **Responsiveness:** The application that we have made is tested across all devices like mobile tablet, and desktop and using the CSS frameworks (like Bootstrap or custom media queries) we have made sure that our website is fully responsive.
b. **Load Testing:** We have tested how it is able to simulate multiple users together and also working on multiple orders simultaneously.
c. The observed response time is also efficient.
d. The performance is optimized because of the usage of React and using techniques like splitting components for better reusability and faster load times.

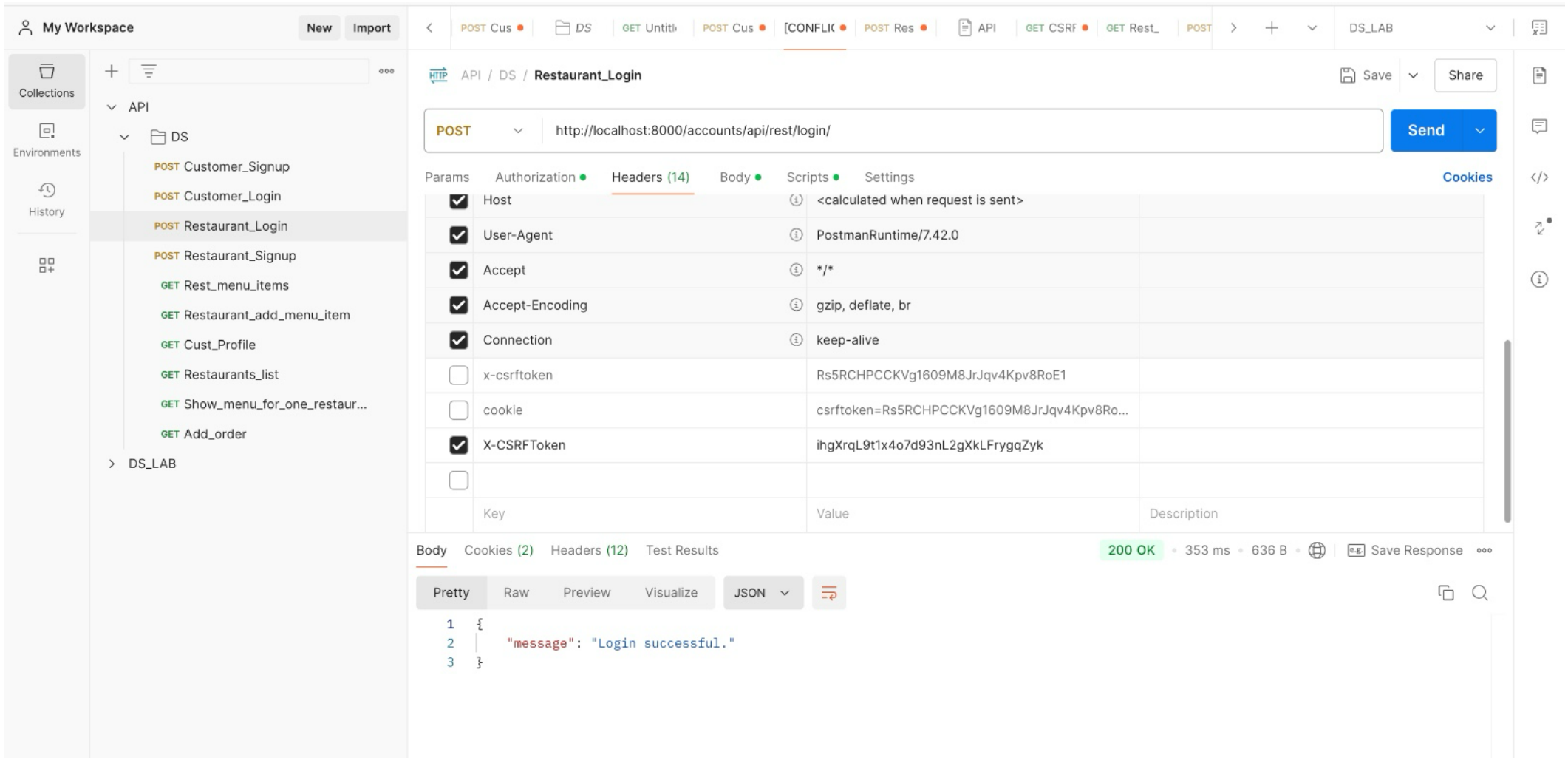
5.API Documentation:

By using POSTMAN, we have connected to each API endpoint and checked the endpoints like etching restaurant lists, placing an order, etc.

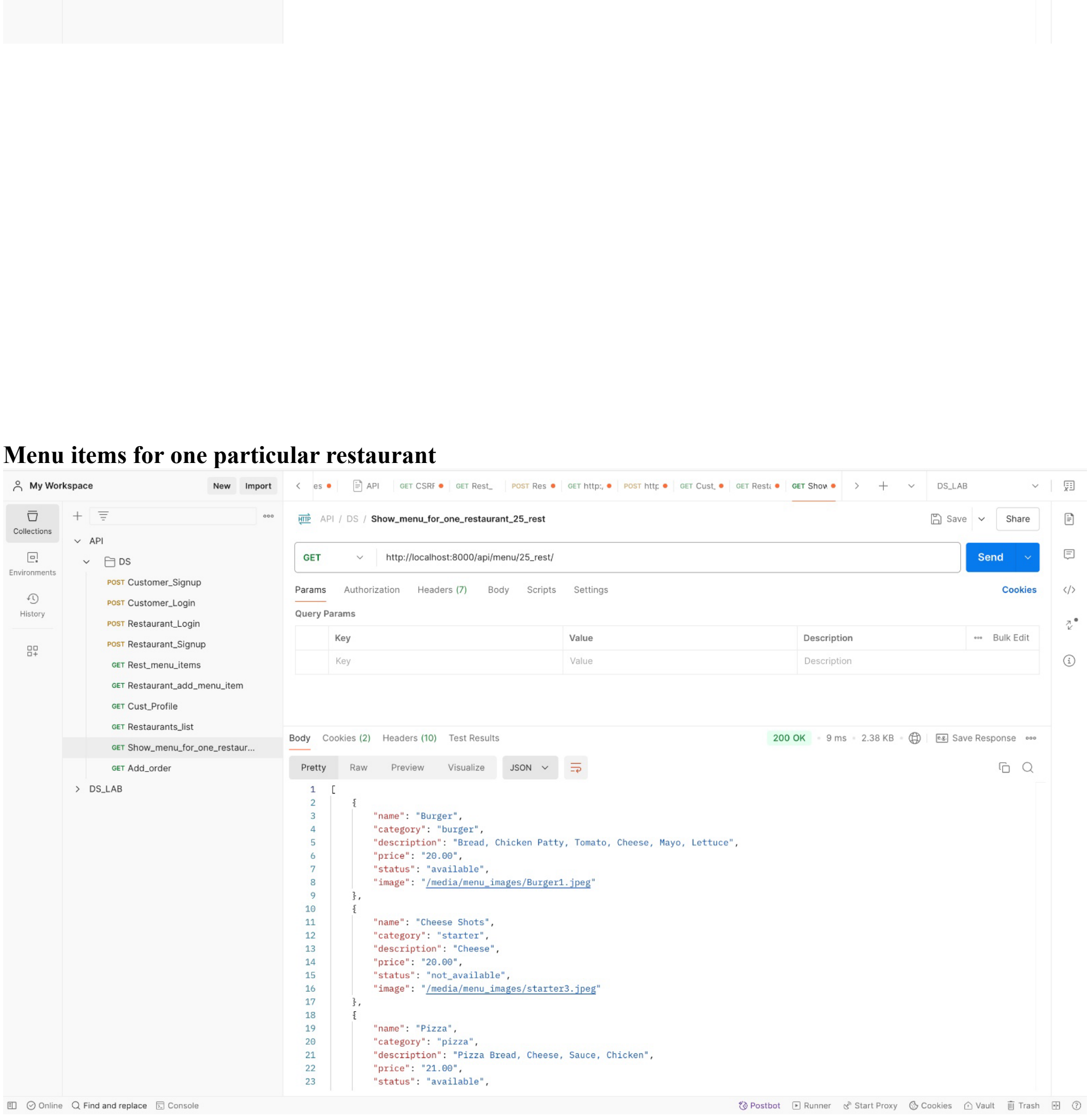
Customer Login:



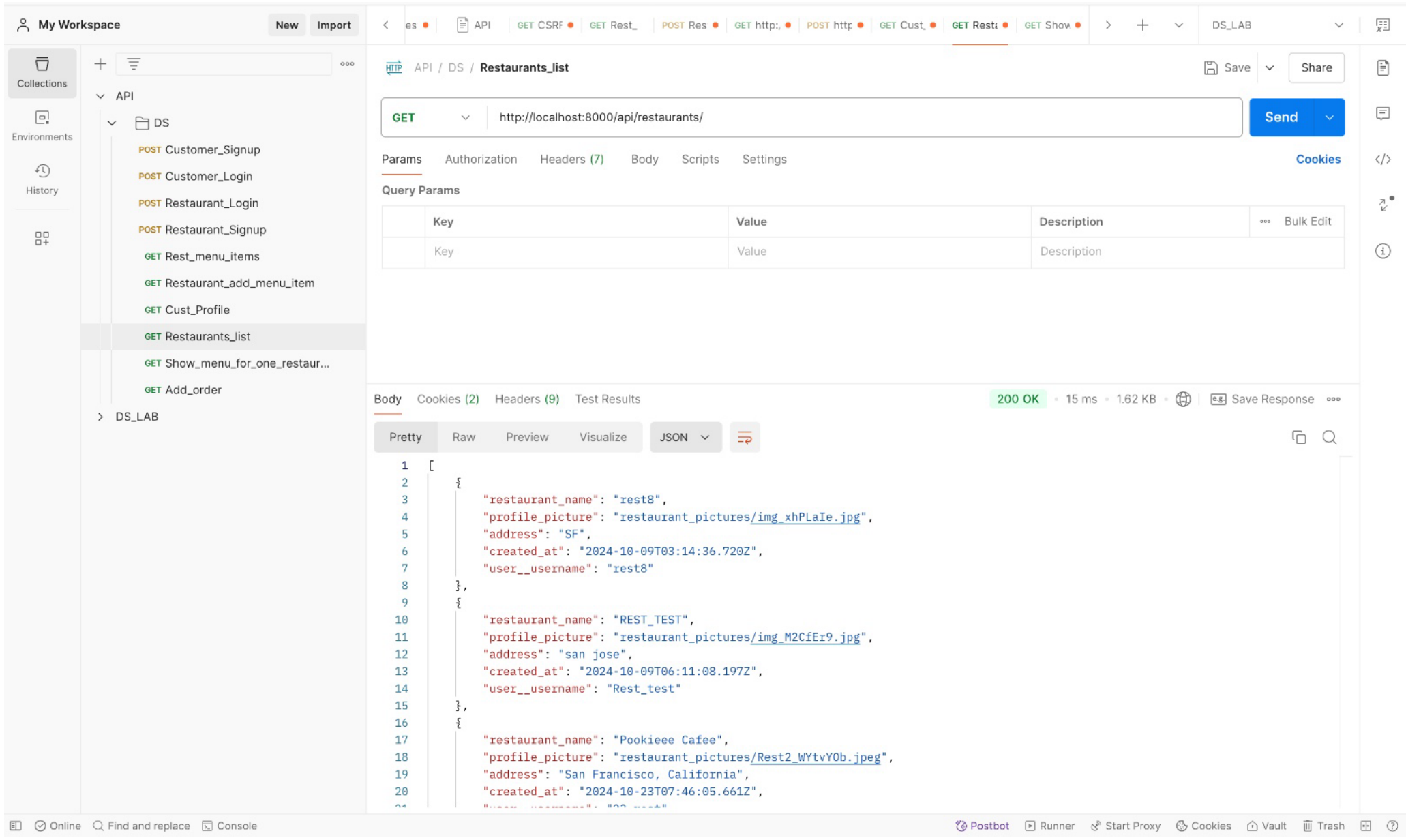
Restaurant Login:



Menu items for one particular restaurant

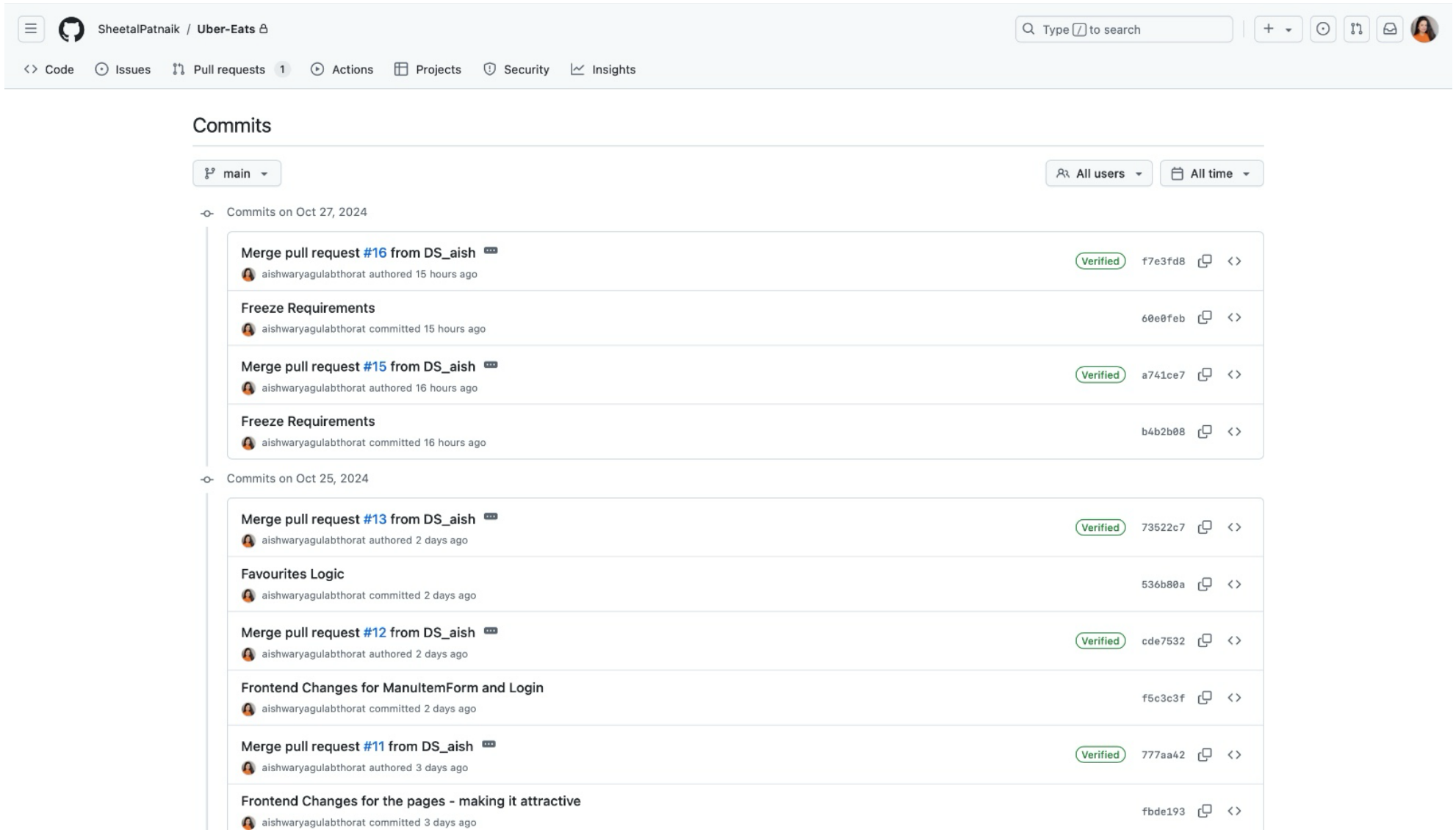


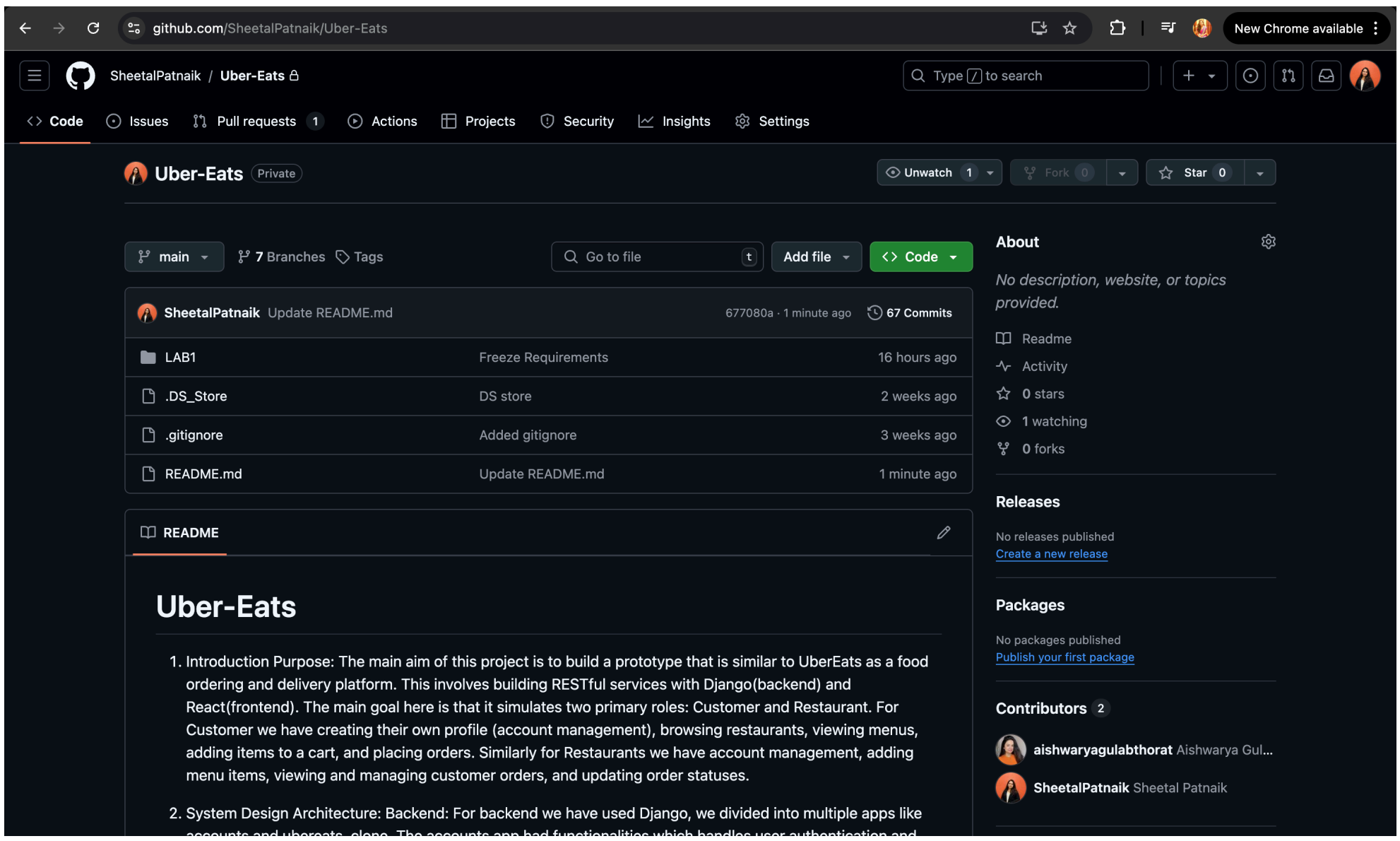
Restaurant List



6.**Git Repository Organization**:

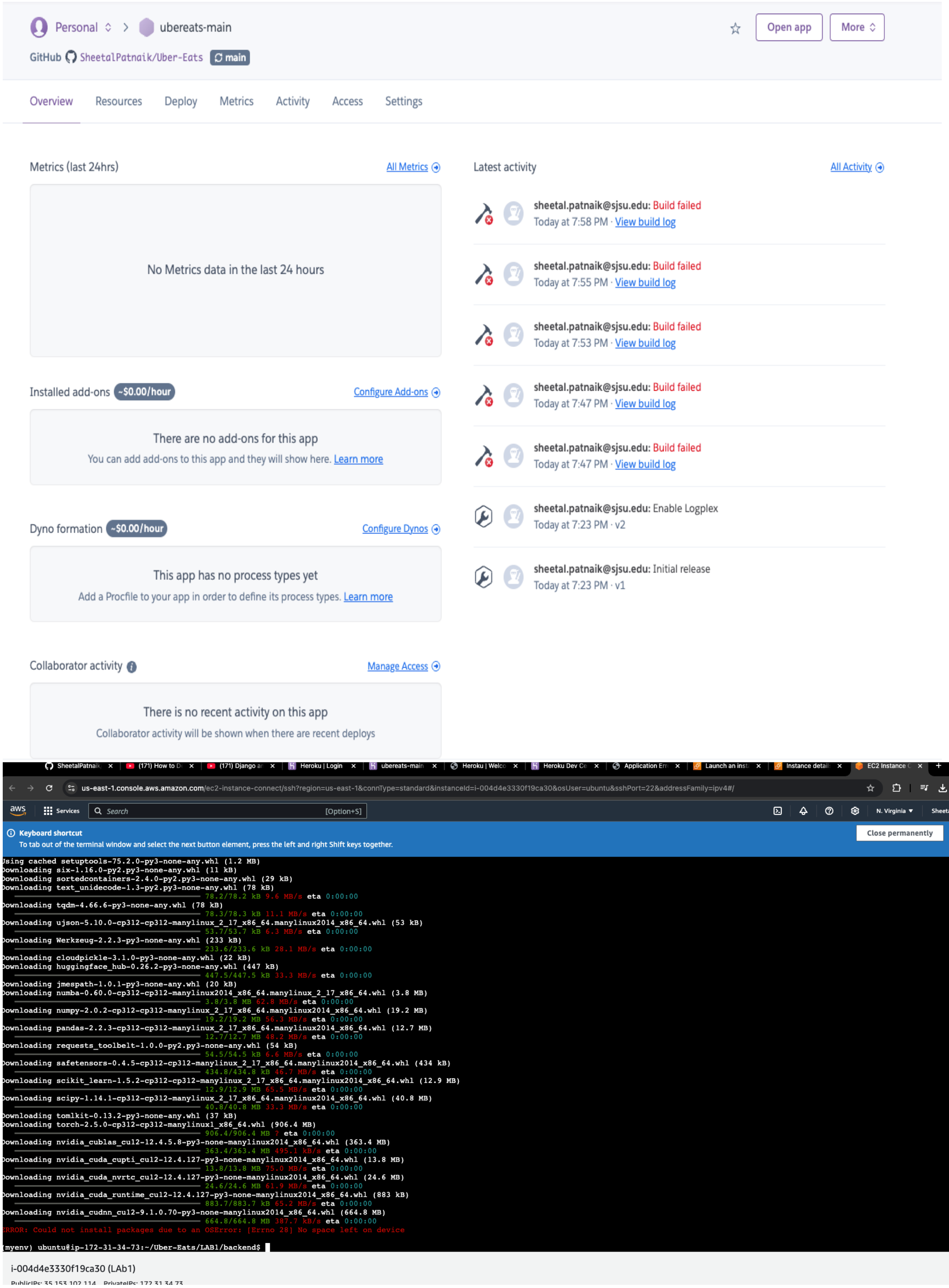
This is our GitHub Repository Link: https://github.com/SheetalPatnaik/Uber-Eats
We have created backend and frontend folder where we upload all our files and commit changes regularly. At the beginning we were doing it separately, but then faced a few commit issues hence we decided to do it from one system together. Hence there are a fewer commits at the end of project.





7. Deployment:

We tried to deploy in both the AWS and Heroku but due to the space issue we couldn’t.



We tried our best but couldn’t get enough space although we tried using t2.medium and t2.large as instance sizes.