

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:

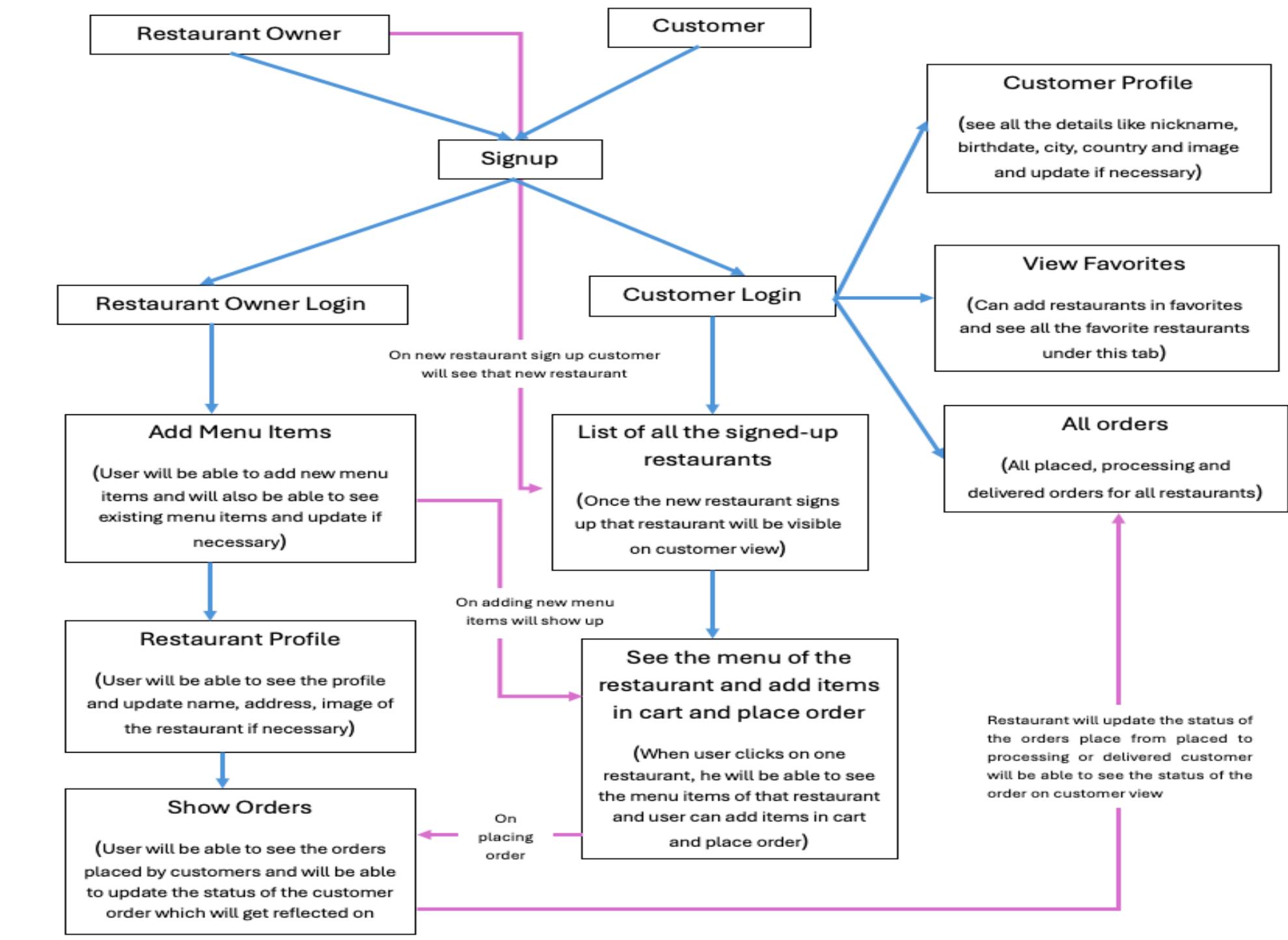
- a. **Login.js** and **Signup.js:** These both handle the initial steps which is the user authentication, providing forms for login and registration. We have **Login.css** and **Signup.css** style for these respective forms.

- 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.

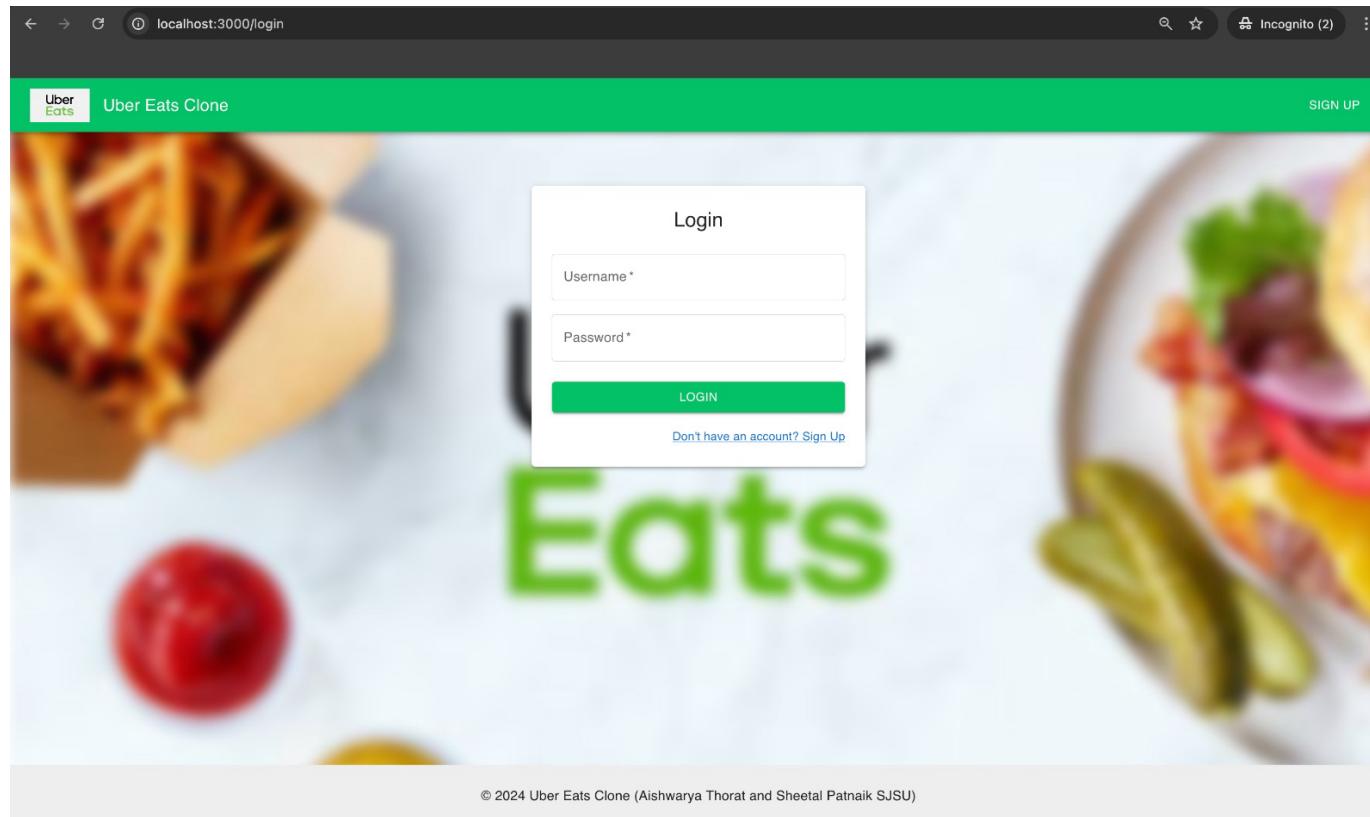
3. Results

a. Customer and Restaurant Registration and Login:

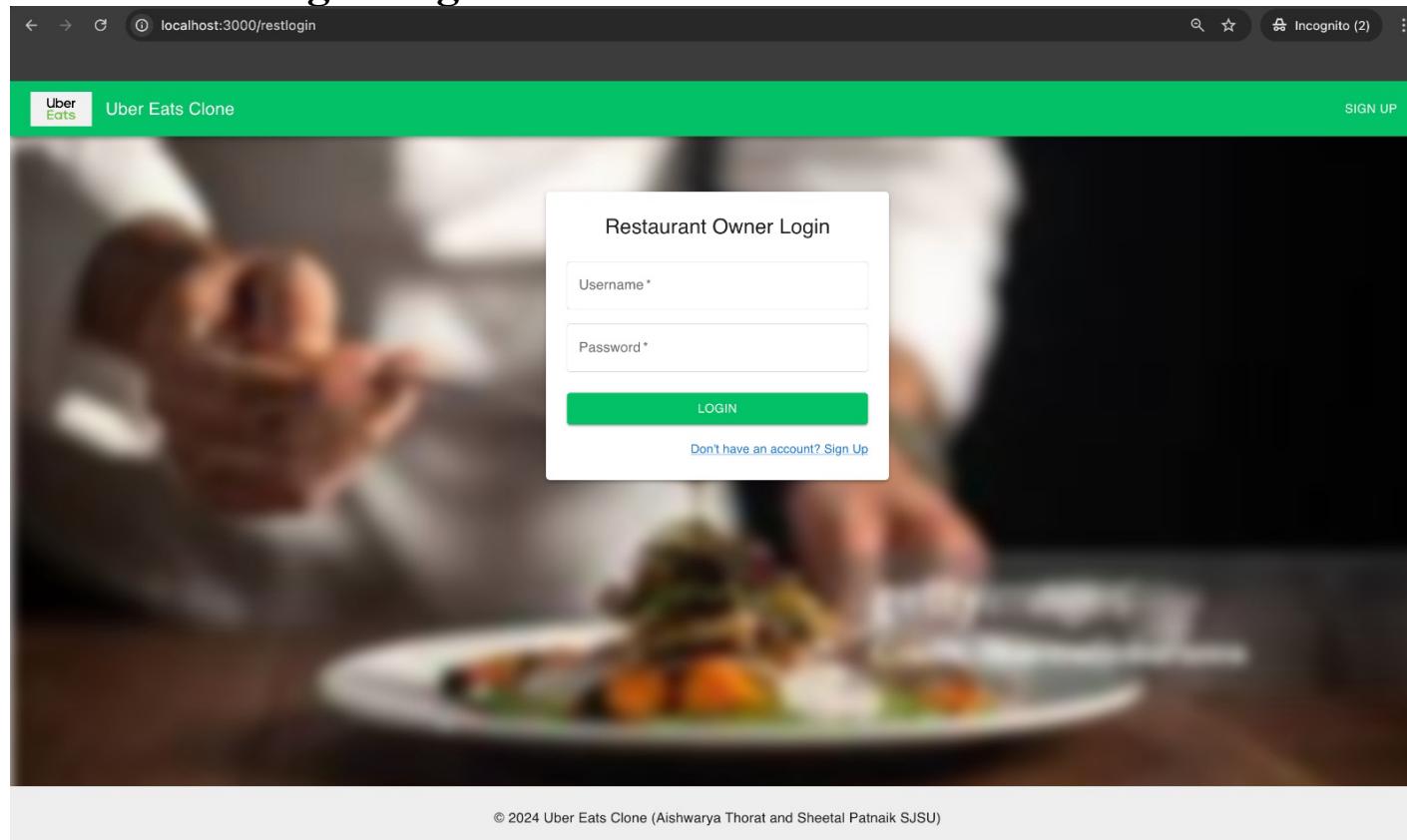
Sign Up Page:

The screenshot shows the sign-up interface for the Uber Eats Clone. At the top, there's a navigation bar with a back arrow, forward arrow, refresh icon, and a search bar. The URL 'localhost:3000/signup' is visible. On the right side of the bar are icons for Incognito mode and more. Below the bar, the main header reads 'Uber Eats Clone' and 'LOGIN'. The background of the page is a blurred image of various food items like pizza and salad. A central modal window is open, titled 'Sign Up'. It has a dropdown menu for 'User Type' with 'Customer' selected. Below the dropdown are fields for 'Username*', 'Email Address*', and 'Password*'. At the bottom of the modal is a green 'SIGN UP' button, and at the very bottom, a link 'Already have an account? Sign in'.

Customer Login Page:

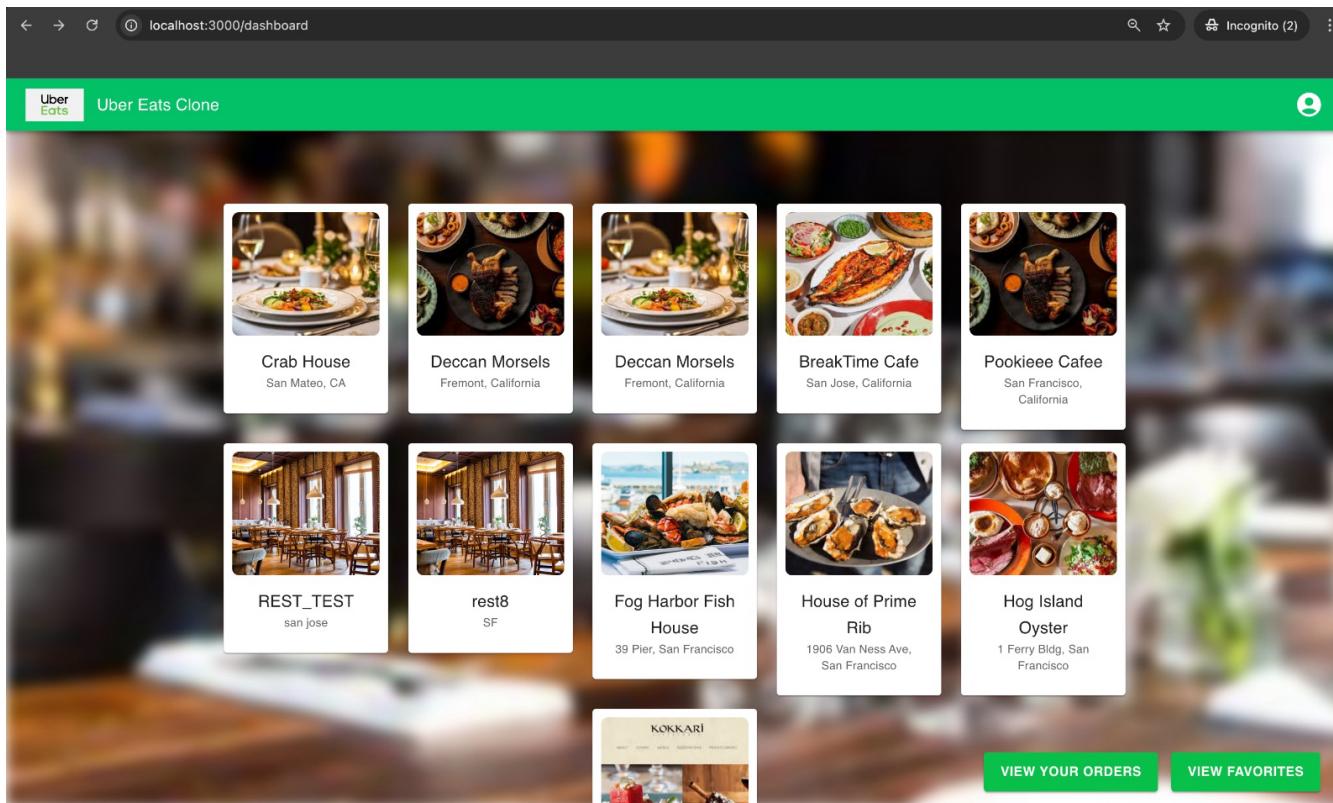


Restaurant Login Page:



b. Customer Restaurant Browsing and Ordering:

○ Restaurant List:



○ Menu Page:

localhost:3000/menu/25_rest

Incognito (2)

Menu for 25_rest

Burger Bread, Chicken Patty, Tomato, Cheese, Mayo, Lettuce Price: \$10.00 Status: available Quantity <input type="text"/> ADD TO CART	Cheese Shots Soda Guava Price: \$20.00 Status: not available Quantity <input type="text"/> ADD TO CART	Pizza Pizza Base, Cheese, Sauce, Chicken Price: \$21.00 Status: available Quantity <input type="text"/> ADD TO CART	Berry Blush Berry, Vanilla Creme, Vanilla Vream Price: \$18.00 Status: available Quantity <input type="text"/> ADD TO CART	Roasted Chicken Roasted Chicken Price: \$12.00 Status: available Quantity <input type="text"/> ADD TO CART
Fruit Punch Berries, Orange, Soda Price: \$10.00 Status: available Quantity <input type="text"/> ADD TO CART	Spicy Guava Soda Guava Price: \$16.00 Status: available Quantity <input type="text"/> ADD TO CART	Biscoff Cheesecake Biscoff Cheesecake Price: \$18.00 Status: available Quantity <input type="text"/> ADD TO CART	Orange Mint Orange, Lemon, Mint Price: \$15.00 Status: available Quantity <input type="text"/> ADD TO CART	Berry Blush Berry Blush Price: \$21.00 Status: available Quantity <input type="text"/> ADD TO CART
Salad Salad Price: \$45.00 Status: available Quantity <input type="text"/> ADD TO CART				

Cart Items

Name	Quantity	Actions
No items in the cart.		

[PLACE ORDER](#)

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

localhost:3000/menu/25_rest

Incognito (2)

3 ADD TO CART	3 ADD TO CART	3 ADD TO CART	3 ADD TO CART	3 ADD TO CART
Fruit Punch Berries, Orange, Soda Price: \$13.00 Status: available Quantity <input type="text"/> 3 ADD TO CART	Spicy Guava Soda Guava Price: \$16.00 Status: available Quantity <input type="text"/> 3 ADD TO CART	Biscoff Cheesecake Biscoff Cheesecake Price: \$21.00 Status: available Quantity <input type="text"/> 3 ADD TO CART	Orange Mint Orange, Lemon, Mint Price: \$18.00 Status: available Quantity <input type="text"/> 3 ADD TO CART	Berry Blush Berry Blush Price: \$21.00 Status: available Quantity <input type="text"/> 3 ADD TO CART
Salad Salad Price: \$45.00 Status: available Quantity <input type="text"/> 3 ADD TO CART				

Cart Items

Name	Quantity	Actions
Salad	1	REMOVE
Spicy Guava	2	REMOVE
Pizza	3	REMOVE

[PLACE ORDER](#)

After placing order, Cart Page:

localhost:3000/customer-orders

Incognito (2)

Your Orders

Placed Orders

Order ID	Restaurant Name	Order Date	Status	Items
3	rest8	22/10/2024, 23:12:10	placed	Spicy Guava (Qty: 1), Burger (Qty: 1), mock (Qty: 2),
11	BreakTime Cafe	26/10/2024, 23:02:44	placed	Salad (Qty: 1),
13	BreakTime Cafe	27/10/2024, 15:50:17	placed	Salad (Qty: 1), Spicy Guava (Qty: 2), Pizza (Qty: 3),

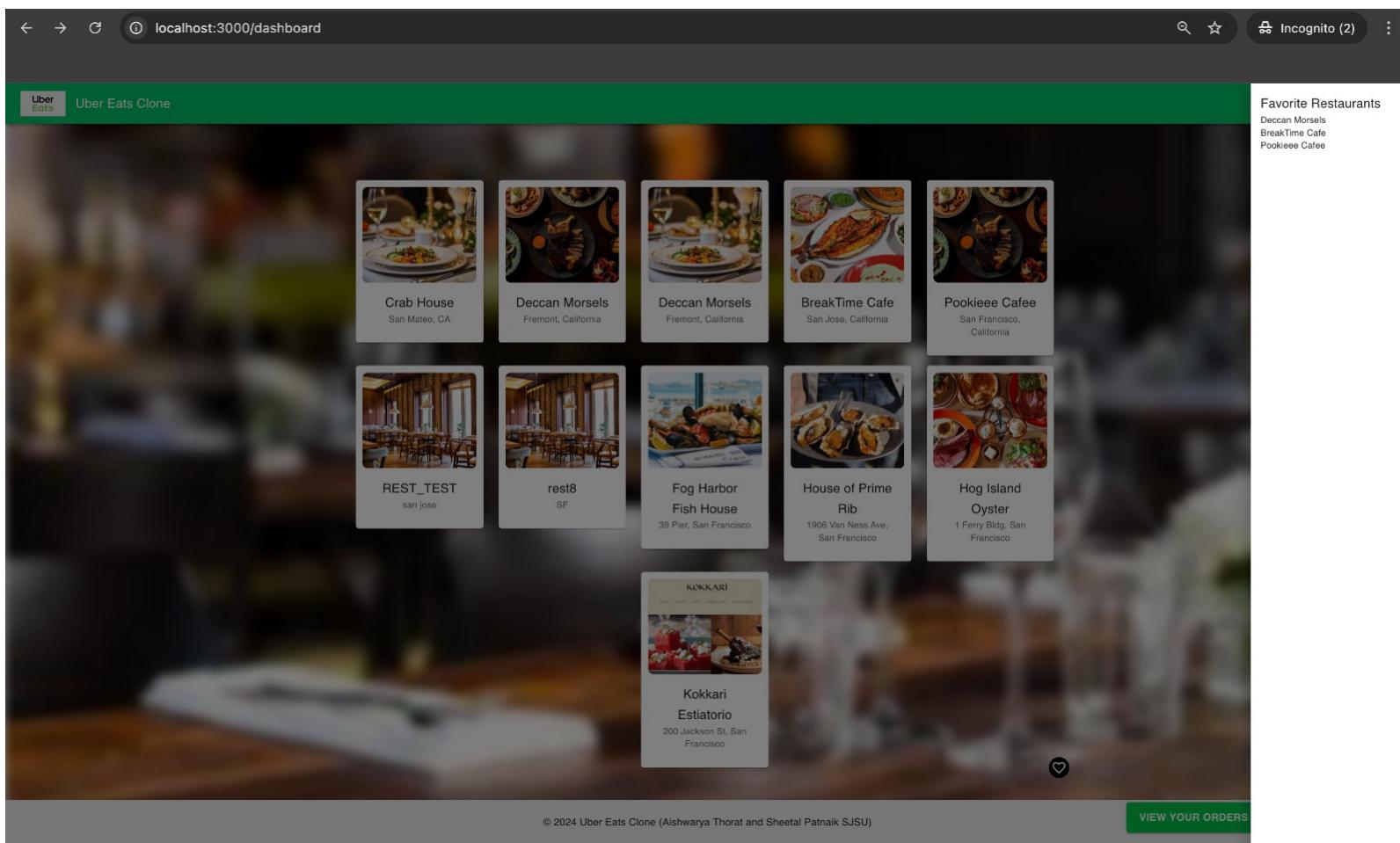
Processing Orders

Order ID	Restaurant Name	Order Date	Status	Items
5	Pookeee Cafe	23/10/2024, 00:50:00	processing	Spicy Guava (Qty: 2), Burger (Qty: 4),

Delivered Orders

Order ID	Restaurant Name	Order Date	Status	Items
4	REST_TEST	23/10/2024, 00:06:43	delivered	mock (Qty: 2), Burger (Qty: 4),
6	Pookeee Cafe	23/10/2024, 02:01:06	delivered	Spicy Guava (Qty: 1), Burger (Qty: 1),
10	BreakTime Cafe	26/10/2024, 21:52:28	delivered	Burger (Qty: 2), Salad (Qty: 4),

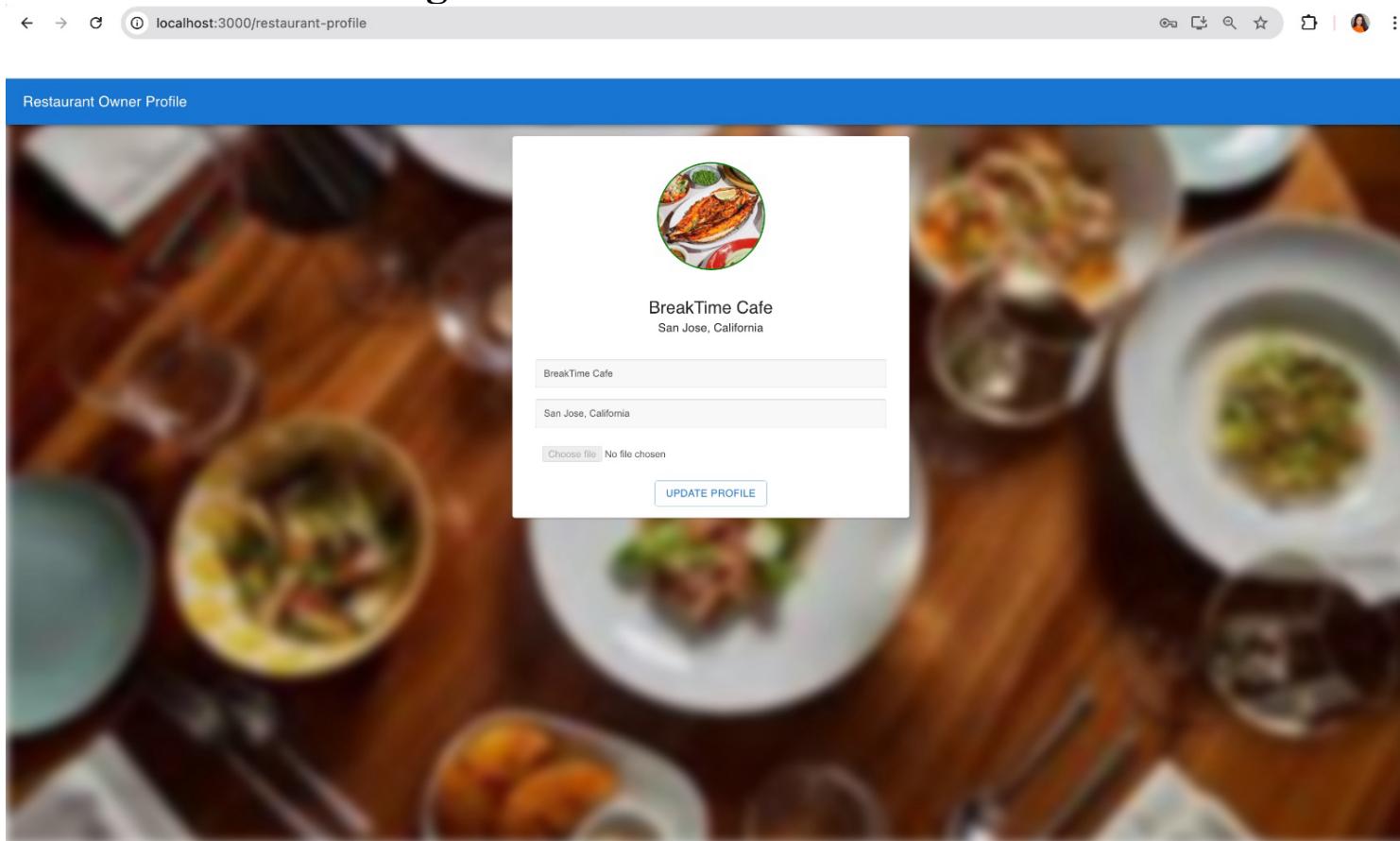
Favorites Page:



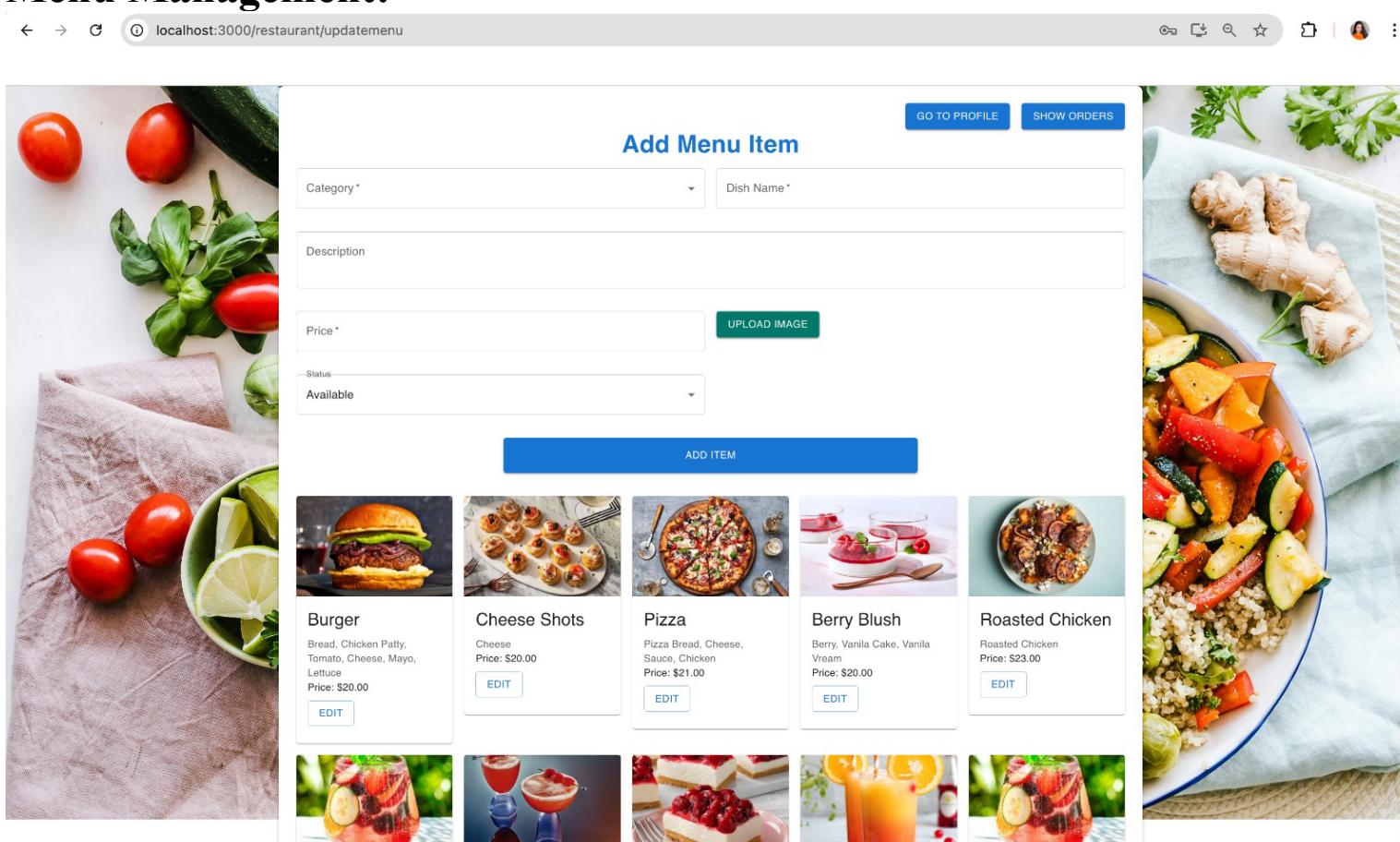
c. Restaurant Dashboard:

- Order Management:

Restaurant Profile Page:



- Menu Management:



d. Order Status Tracking:

The screenshot shows a web application titled "Orders for". It displays three sections: "Upcoming Orders", "Processing Orders", and "Delivered Orders".

- Upcoming Orders:** Shows two orders. Order ID B9010EE5B6 is placed by customer "aishwarya1" with items "Salad (Qty: 1)". Order ID F8A7797B9B is placed by customer "aishwarya1" with items "Salad (Qty: 1), Spicy Guava (Qty: 2), Pizza (Qty: 3)". Both have an "UPDATE STATUS" button.
- Processing Orders:** Shows no orders found.
- Delivered Orders:** Shows one order. Order ID BC4D66C7E1 is delivered by customer "aishwarya1" with items "Burger (Qty: 2), Salad (Qty: 4)". An "UPDATE STATUS" button is present.

A decorative image of a meal is visible at the bottom of the page.

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:

The screenshot shows the POSTMAN interface with the following details:

- Collection:** My Workspace
- Request:** POST /Customer_Login
- Method:** POST
- URL:** http://localhost:8000/accounts/login/
- Body (JSON):**

```
1 {
2   "username": "aishwarya1",
3   "password": "aishwarya1"
4 }
```
- Response Headers:** 200 OK, 348 ms, 636 B
- Response Body (Pretty JSON):**

```
1 {
2   "message": "Login successful!"
3 }
```

Restaurant Login:

The screenshot shows the Postman interface with the following details:

- Workspace:** My Workspace
- Collection:** DS
- Request:**
 - Method:** POST
 - URL:** http://localhost:8000/accounts/api/rest/login/
 - Headers:** (14)
 - Host: <calculated when request is sent>
 - User-Agent: PostmanRuntime/7.42.0
 - Accept: */*
 - Accept-Encoding: gzip, deflate, br
 - Connection: keep-alive
 - x-csrfToken: Rs5RCHPCKVg1609M8JrJqv4Kpv8RoE1
 - cookie: csrfToken=Rs5RCHPCKVg1609M8JrJqv4Kpv8Ro...
 - X-CSRFToken: ihgXrqL9t1x4o7d93nL2gXkLFryggZyk
 - Body:** (Pretty) JSON response:


```

1 {
2   "message": "Login successful."
3 }
```
- Status:** 200 OK
- Time:** 353 ms
- Size:** 636 B

Menu items for one particular restaurant

The screenshot shows the Postman interface with the following details:

- Workspace:** My Workspace
- Collection:** DS
- Request:**
 - Method:** GET
 - URL:** http://localhost:8000/api/menu/25_rest/
 - Headers:** (7)
 - Query Params:** (1)

Key	Value	Description	Bulk Edit
Key	Value	Description	
 - Body:** (Pretty) JSON response:


```

1 [
2   {
3     "name": "Burger",
4     "category": "burger",
5     "description": "Bread, Chicken Patty, Tomato, Cheese, Mayo, Lettuce",
6     "price": "20.00",
7     "status": "available",
8     "image": "/media/menu_images/Burger1.jpeg"
9   },
10  {
11    "name": "Cheese Shots",
12    "category": "starter",
13    "description": "Cheese",
14    "price": "20.00",
15    "status": "not_available",
16    "image": "/media/menu_images/starter3.jpeg"
17  },
18  {
19    "name": "Pizza",
20    "category": "pizza",
21    "description": "Pizza Bread, Cheese, Sauce, Chicken",
22    "price": "21.00",
23    "status": "available",
24  }
```
- Status:** 200 OK
- Time:** 9 ms
- Size:** 2.38 KB

Restaurant List

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Collections', 'Environments', and 'History'. Under 'API', there's a folder named 'DS' containing several requests: 'POST Customer_Signup', 'POST Customer_Login', 'POST Restaurant_Login', 'POST Restaurant_Signup', 'GET Rest_menu_items', 'GET Restaurant_add_menu_item', 'GET Cust_Profile', 'GET Restaurants_list' (which is selected), 'GET Show_menu_for_one_restaur...', and 'GET Add_order'. The main area shows a 'HTTP API / DS / Restaurants_list' request. It's a 'GET' request to 'http://localhost:8000/api/restaurants/'. The 'Body' tab is selected, showing a JSON response with three restaurant objects. The first object has a restaurant name of 'rest8', profile picture 'restaurant_pictures/img_xhPLaIe.jpg', address 'SF', created at '2024-10-09T03:14:36.720Z', and user username 'rest8'. The second object has a restaurant name of 'REST_TEST', profile picture 'restaurant_pictures/img_M2CfEr9.jpg', address 'san jose', created at '2024-10-09T06:11:08.197Z', and user username 'Rest_test'. The third object has a restaurant name of 'Pookeee Cafee', profile picture 'restaurant_pictures/Rest2_WYtvYob.jpeg', address 'San Francisco, California', created at '2024-10-23T07:46:05.661Z', and user username 'Pookeee'. The status bar at the bottom indicates a 200 OK response with 15 ms, 1.62 KB, and a save response button.

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 few commits at the end of project.

The screenshot shows a GitHub repository page for 'Uber-Eats / LAB1 / backend / accounts /'. The left sidebar shows the repository structure with 'Files' and a list of files including 'main', 'migrations', 'templates/accounts', 'admin.py', 'apps.py', 'forms.py', 'models.py', 'serializers.py', 'signals.py', 'tests.py', 'urls.py', 'views.py', 'media', 'ubereats_clone', '.DS_Store', 'manage.py', and 'requirements.txt'. The main area displays a list of recent commits for the 'accounts' folder. The commits are as follows:

Name	Last commit message	Last commit date
..		
migrations	backend for handling cart menu items - order management	5 days ago
templates/accounts	Refactor	3 weeks ago
__init__.py	Refactor	3 weeks ago
admin.py	backend for handling cart menu items - order management	5 days ago
apps.py	Refactor	3 weeks ago
forms.py	Backend of MenuItem page	2 weeks ago
models.py	Frontend Changes for MenuItemForm and Login	2 days ago
serializers.py	Show Orders for a particular customer - backend	3 days ago
signals.py	Refactor	3 weeks ago
tests.py	Refactor	3 weeks ago
urls.py	Show Orders for a particular customer - backend	3 days ago
views.py	Frontend Changes for MenuItemForm and Login	2 days ago

Commits

main

All users All time

Commits on Oct 27, 2024

- Merge pull request #16 from DS_aish [...](#) Verified f7e3fd8
- Freeze Requirements [...](#) aishwaryagulabthorat committed 15 hours ago
- Merge pull request #15 from DS_aish [...](#) Verified a741ce7
- Freeze Requirements [...](#) aishwaryagulabthorat committed 16 hours ago

Commits on Oct 25, 2024

- Merge pull request #13 from DS_aish [...](#) Verified 73622c7
- Favourites Logic [...](#) aishwaryagulabthorat committed 2 days ago
- Merge pull request #12 from DS_aish [...](#) Verified cde7532
- Frontend Changes for ManulItemForm and Login [...](#) aishwaryagulabthorat committed 2 days ago
- Merge pull request #11 from DS_aish [...](#) Verified 777aa42
- Frontend Changes for the pages - making it attractive [...](#) aishwaryagulabthorat committed 3 days ago

github.com/SheetalPatnaik/Uber-Eats

Code Issues Pull requests Actions Projects Security Insights Settings

Uber-Eats Private

Unwatch 1 Fork 0 Star 0

main 7 Branches Tags Go to file Add file Code

About
No description, website, or topics provided.

Releases
No releases published [Create a new release](#)

Packages
No packages published [Publish your first package](#)

Contributors 2

- aishwaryagulabthorat Aishwarya Gul...
- SheetalPatnaik Sheetal Patnaik

README

Uber-Eats

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, along. The accounts app had functionalities which handles user authentication and

7. Deployment:

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

Personal > ubereats-main

GitHub SheetalPatnaik/Uber-Eats main

Overview Resources Deploy Metrics Activity Access Settings

Metrics (last 24hrs) All Metrics

No Metrics data in the last 24 hours

Installed add-ons -\$0.00/hour Configure Add-ons

There are no add-ons for this app
You can add add-ons to this app and they will show here. [Learn more](#)

Dyno formation -\$0.00/hour Configure Dynos

This app has no process types yet
Add a Procfile to your app in order to define its process types. [Learn more](#)

Latest activity All Activity

- sheetal.patnaik@sjtu.edu: Build failed Today at 7:58 PM · [View build log](#)
- sheetal.patnaik@sjtu.edu: Build failed Today at 7:55 PM · [View build log](#)
- sheetal.patnaik@sjtu.edu: Build failed Today at 7:53 PM · [View build log](#)
- sheetal.patnaik@sjtu.edu: Build failed Today at 7:47 PM · [View build log](#)
- sheetal.patnaik@sjtu.edu: Enable Logplex Today at 7:23 PM · v2
- sheetal.patnaik@sjtu.edu: Initial release Today at 7:23 PM · v1

Collaborator activity Manage Access

There is no recent activity on this app
Collaborator activity will be shown when there are recent deploys

us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-004d4e3330f19ca30&osUser=ubuntu&sshPort=22&addressFamily=ipv4#/
 Search [Option+S] Close permanently

```
Using cached setuptools-75.2.0-py3-none-any.whl (1.2 MB)
Downloaded six-1.16.0-py2.py3-none-any.whl (11 kB)
Downloaded sortedcontainers-2.4.0-py2.py3-none-any.whl (29 kB)
Downloaded text_unidecode-1.3-py2.py3-none-any.whl (78 kB)
    78.2/78.2 kB 9.6 MB/s eta 0:00:00
Downloaded tqdm-4.66.6-py3-none-any.whl (78 kB)
    78.3/78.3 kB 11.1 MB/s eta 0:00:00
Downloaded ujson-5.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (53 kB)
    53.7/53.7 kB 6.3 MB/s eta 0:00:00
Downloaded Werkzeug-2.2.3-py3-none-any.whl (233 kB)
    233.6/233.6 kB 28.1 MB/s eta 0:00:00
Downloaded cloudpickle-3.1.0-py3-none-any.whl (22 kB)
Downloaded huggingface_hub-0.26.2-py3-none-any.whl (447 kB)
    447.5/447.5 kB 33.3 MB/s eta 0:00:00
Downloaded jmespath-1.0.1-py3-none-any.whl (20 kB)
Downloaded numba-0.60.0-cp312-cp312-manylinux2014_x86_64.manylinux2014_x86_64.whl (3.8 MB)
    3.8/3.8 kB 62.8 MB/s eta 0:00:00
Downloaded numpy-1.20.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (19.2 MB)
    19.2/19.2 kB 56.3 MB/s eta 0:00:00
Downloaded pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.7 MB)
    12.7/12.7 kB 48.2 MB/s eta 0:00:00
Downloaded requests_toolbelt-1.0.0-py2.py3-none-any.whl (54 kB)
    54.5/54.5 kB 6.6 MB/s eta 0:00:00
Downloaded safetensors-0.4.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (434 kB)
    434.8/434.8 kB 46.7 MB/s eta 0:00:00
Downloaded scikit_learn-1.5.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.9 MB)
    12.9/12.9 kB 65.5 MB/s eta 0:00:00
Downloaded scipy-1.14.1-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (40.8 MB)
    40.8/40.8 kB 33.3 MB/s eta 0:00:00
Downloaded torch-2.5.0-cp312-cp312-manylinux_x86_64.whl (906.4 MB)
    906.4/906.4 kB ? eta 0:00:00
Downloaded nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
    363.4/363.4 kB 495.1 kB/s eta 0:00:00
Downloaded nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
    13.8/13.8 kB 75.0 MB/s eta 0:00:00
Downloaded nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
    24.6/24.6 kB 61.9 MB/s eta 0:00:00
Downloaded nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
    883.7/883.7 kB 65.2 MB/s eta 0:00:00
Downloaded nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
    664.8/664.8 kB 387.7 kB/s eta 0:00:00
ERROR: Could not install packages due to an OSError: [Errno 28] No space left on device
(myenv) ubuntu@ip-172-31-34-73:~/Uber-Eats/LAB1/backend$ i-004d4e3330f19ca30 (Lab1)
```

Private IP: 172.31.34.73 Private IP: 172.31.34.73

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