**ASSIGNMENT 3- BACKEND API DEVELOPMENT WITH DRF**

**Group-9 (Sachin Gaurav Srivastava, Shiva Krishna K, Avijit Nandi)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code:

* + Code is available in github repo: <https://github.com/avijitn1/railman_django>

**App UI:**

RailMan App is a food catering service app for Indian Railways. The app is designed for both users and Restaurant administrator with different set of features based on the roles. The Main UI has navigation bar with various menu items.

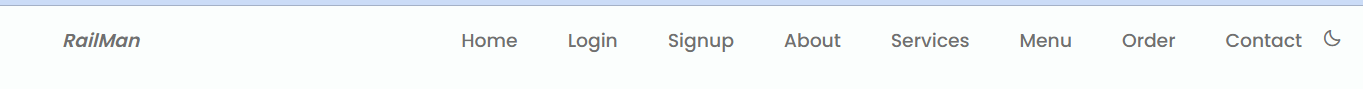
* **Home**:

Graphical user interface

Description automatically generated

* **Layout elements**:
  + Navbar
  + Content
  + Footer
* **NavBar** consists of Menus:

Home, login, Signup, About, Services, Menu and Contact.



* **About**:

Graphical user interface, application

Description automatically generated

* **Order**:

Graphical user interface, application

Description automatically generated

* **Contact**:

Graphical user interface, text, application

Description automatically generated

* **Footer**:

Graphical user interface, text, application

Description automatically generated

**App Backend:**

**RailMan** backend is implemented with Django Rest Framework. The project name is Railman\_backend. The project contains an app **railman.** Virtual environment **venv** is created and packages installed as required. Database used in sqlite3 for ease of use and portability.

Text

Description automatically generated

**Model:**

ER diagram of the model is depicted below. The backend has 5 models: User, RestaurantOwner, Restaurant, Menu and Order.

Diagram

Description automatically generated

Model.py

Text

Description automatically generatedText

Description automatically generated

Admin access is created, and data is input as needed through Django admin page.

Graphical user interface, application, website

Description automatically generated

Serializers:

Model serializers are created to facilitate API responses:

Text

Description automatically generated

**EPIC 1: Login and Registration:**

**1.1 Registration UI:**

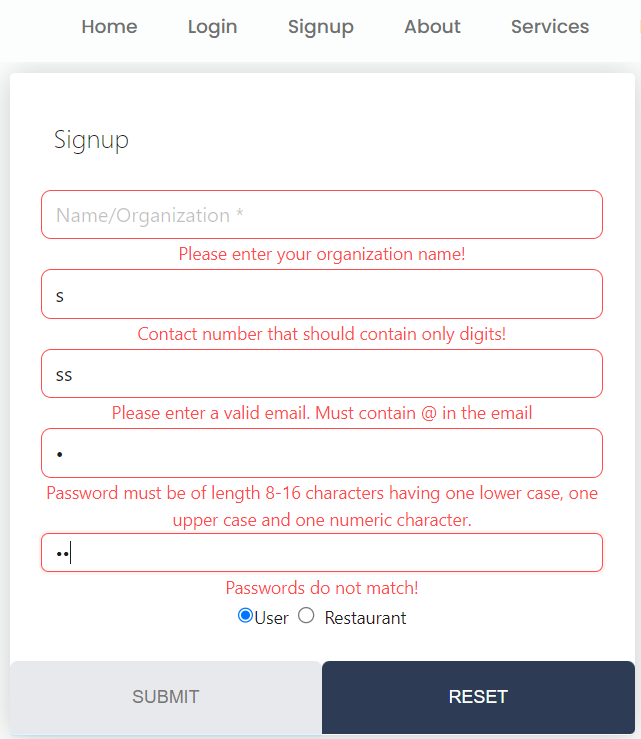
Graphical user interface, application

Description automatically generated

The Signup component consists of:

1. Name/Organization field: [required]
2. Contact: [required, validated for 10-digit phone number]
3. Email: [required, validated]
4. Password: [required, validated – minimum 8-character, one uppercase, and numeric]
5. Confirm password: [required, validated matching password]
6. Role: User/Restaurant owner selection radio [default: user]

**Validation:**



**Backend Implementation and integration with frontend:**

Rest APIs for Registration is implemented in views\_authentication.py. The register API is handled as a POST request with essential fields as input.

Text

Description automatically generated

On submit click user gets an alert “Successful registration”

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Testing with Postman application:

Graphical user interface, text, application, email

Description automatically generated

**1.2 Login UI and Backend:**

On clicking Login menu Login UI is presented:

Graphical user interface, application, website

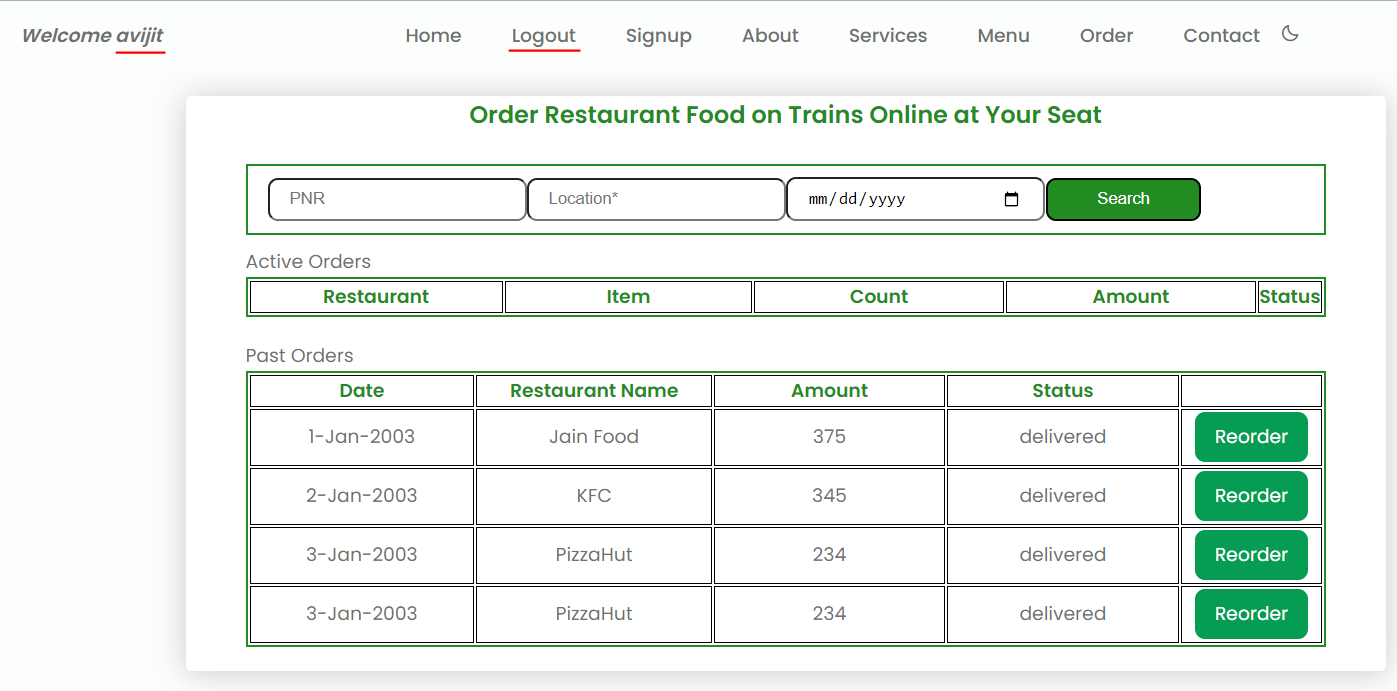
Description automatically generated

Integration with backend through DRF:

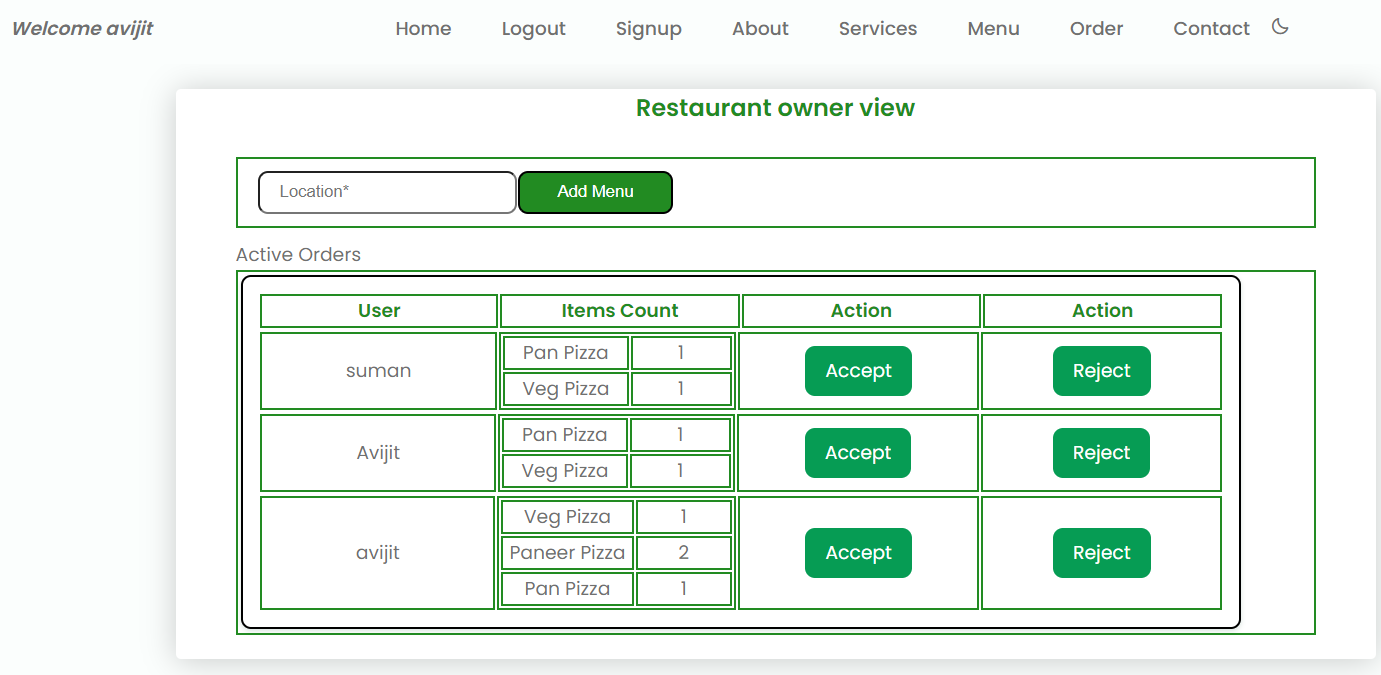
Text

Description automatically generated

1. Correct credentials takes user to user dashboard.



* 1. Restaurant owner to Restaurant owner dashboard.



1. Correct credentials save data in Redux Store (Reducer: userData)
   1. NavBar menu updated to logout
   2. NavBar welcome message: “Welcome UserName”

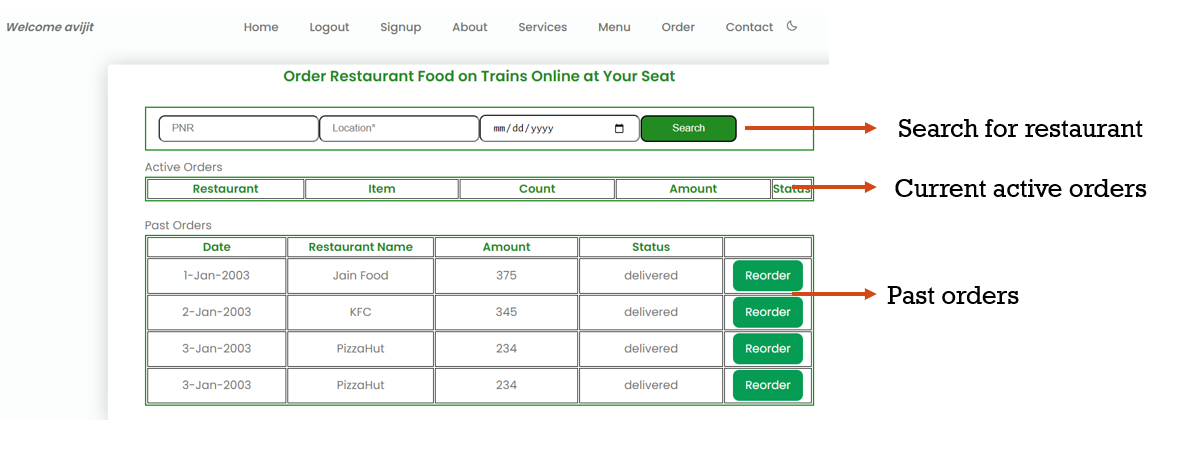
Graphical user interface, text, application

Description automatically generated

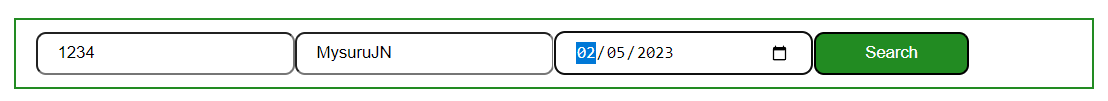
**EPIC 2: CUSTOMER FOOD ORDER**

* 1. **User DashBoard:**

1. On Successful login user is presented with User Dashboard: No current order displays empty active orders table.



* 1. **View Restaurant:** 
     1. Search with PNR, Location (Station Name), Date



* + 1. OnClick Search button takes to Restaurant exploration page. Displays only restaurant matching search criteria. Multiple pages are displayed with 3 restaurants in each page.

Graphical user interface, website

Description automatically generated

Backend API Implementation: path('api/restaurants/city/', views.RestaurantList.as\_view()),

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Integration with Frontend:

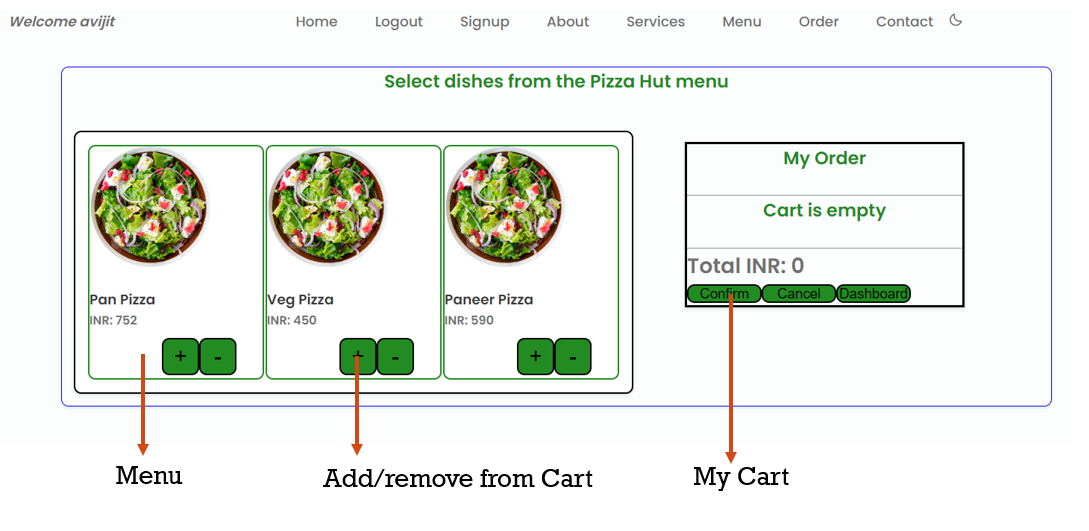
Text

Description automatically generated

* 1. **Select Item and Order**

**Backend API to explore Menu:** path('api/menu/', views.MenuList.as\_view()),

1. Onclick Cart button on any Restaurant takes user to the Restaurant’s Menu page Example: We clicked on PizzaHut cart button:



* + 1. OnClick Add to Cart button  the menu is added to the cart. My Order component shows all selected items and calculates total price of the items. This uses redux store to store data and show in main user dashboard

Graphical user interface, application

Description automatically generated

* + 1. OnClick => confirm button saves the order in the database and takes user to main user dashboard

Table

Description automatically generated

**Redux State:**

Redux actions and Reducers are implemented to create state for each user.

Text

Description automatically generated with medium confidence

* **State before login:**

Text

Description automatically generated

* **State after login:**

Graphical user interface, text, application

Description automatically generated

* **State after adding to cart:**

Text

Description automatically generated with low confidence

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**EPIC 3: RESTAURANT DASHBOARD**

3.1 **Restaurant Dashboard:**

Login with Restaurant role take user to Restaurant owner dashboard:

Graphical user interface, text, application

Description automatically generated

**Edit Menu**

On successful login Restaurant owner’s dashboard is presented with Navigation menu to **Edit menu**. Restaurant owner also get to see a dashboard to accept or reject orders.

Graphical user interface

Description automatically generated

Edit Menu items:

Graphical user interface, text, application

Description automatically generated

Update Menu API: path('api/menu/', views.MenuList.as\_view()),

Updating price of Menu from INR 987 to INR 1000

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Integration with Frontend:

Text

Description automatically generated

Dashboard to Accept/reject orders:

Graphical user interface

Description automatically generated

Backend API for getting orders: path('api/orders/', views.OrderList.as\_view()),

Text

Description automatically generated

Documentation:

path(r'docs/', include\_docs\_urls(title='Helpdesk API')),

Graphical user interface, text, application

Description automatically generated