**Django-Preorder System and Disaster Preparedness Hub**

**Final Report**

***Submitted by***

**BHAVANI PRASAD GUMMADALA**

***to***

**DHARANI GOWTHAM**

*In a partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

*in*

*COMPUTER SCIENCE AND ENGINEERING*

**LOVELY PROFESSIONAL UNIVERSITY**

**JULY – 2025**

Project 1:

**Preordering System for College Stalls**

**Introduction**:

A web-based platform allowing college students to **preorder meals** from campus food stalls, reducing wait times, featuring authentication, role-based access and eliminating the need to queue. Students can schedule orders, receive pick-up alerts, and collect meals without missing classes.

**Problem Identification**

* Long queues at peak hours lead to wasted time.
* Students miss classes or meals due to delays.
* Manual order-taking is inefficient for vendors.

**Proposed Solution**

* Scheduled Orders: Students book meals in advance.
* Queue Skipping: Priority pick-up for preorders.
* Real-time Alerts: Notifications when orders are ready.
* Vendor Dashboard: Manage orders.

**Features Implemented**

**User Roles**:

Students (place orders, view history).

Vendors (manage menus, confirm orders).

Admin (oversee system, resolve issues).

**Core Functionality**:

Menu browsing

Pick-up time selection.

**Tech Stack**:

**Backend**: Django (Python)

**Frontend**: HTML/CSS, JavaScript.

**Database**: SQLite.

**User Authentication & Authorization**:

**Login/Register Pages**:

Students/vendors register with email/phone verification.

**Role-Based Access Control**:

Students: Place orders, view history.

Vendors: Manage menus, update order status.

Admin: Approve vendors, audit logs.

**Security**:

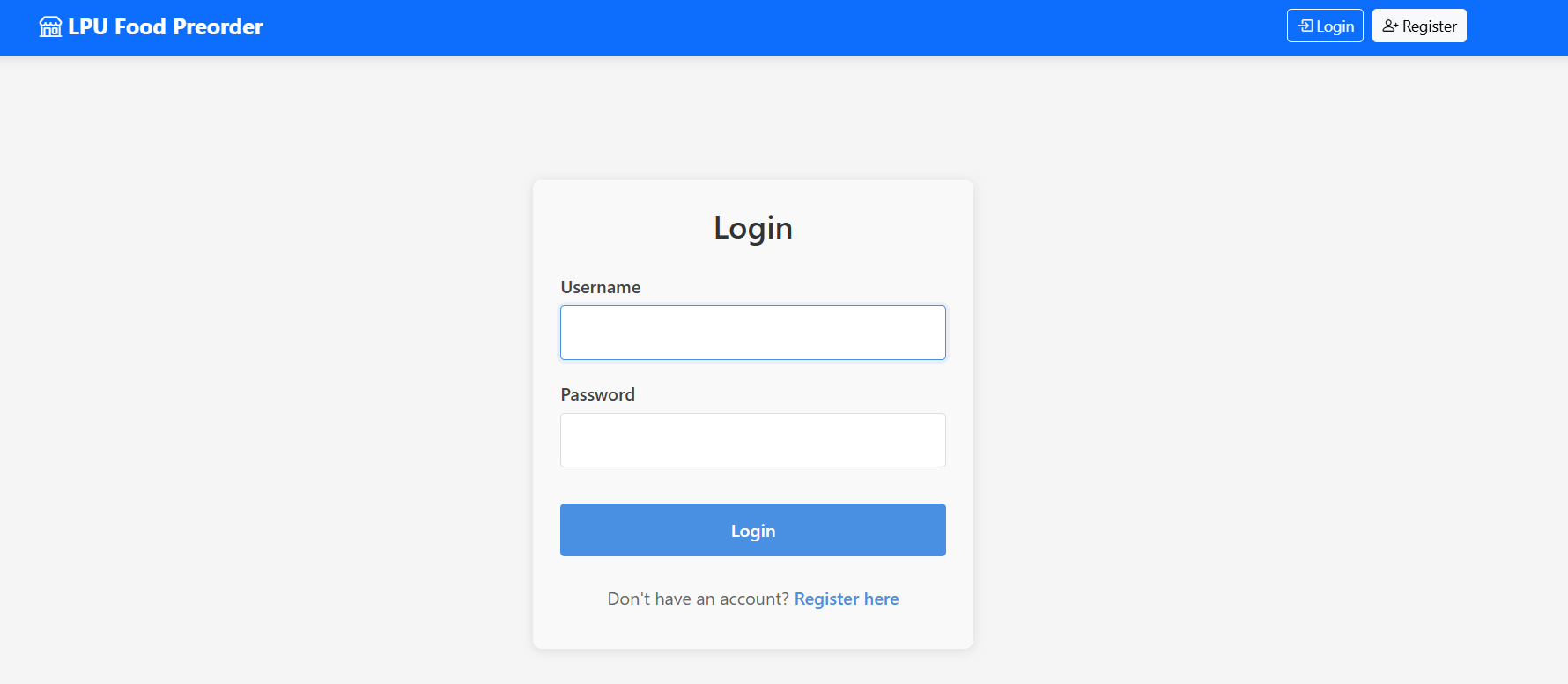
Password hashing

CSRF protection for forms.

Session-based authentication (@login\_required decorators).

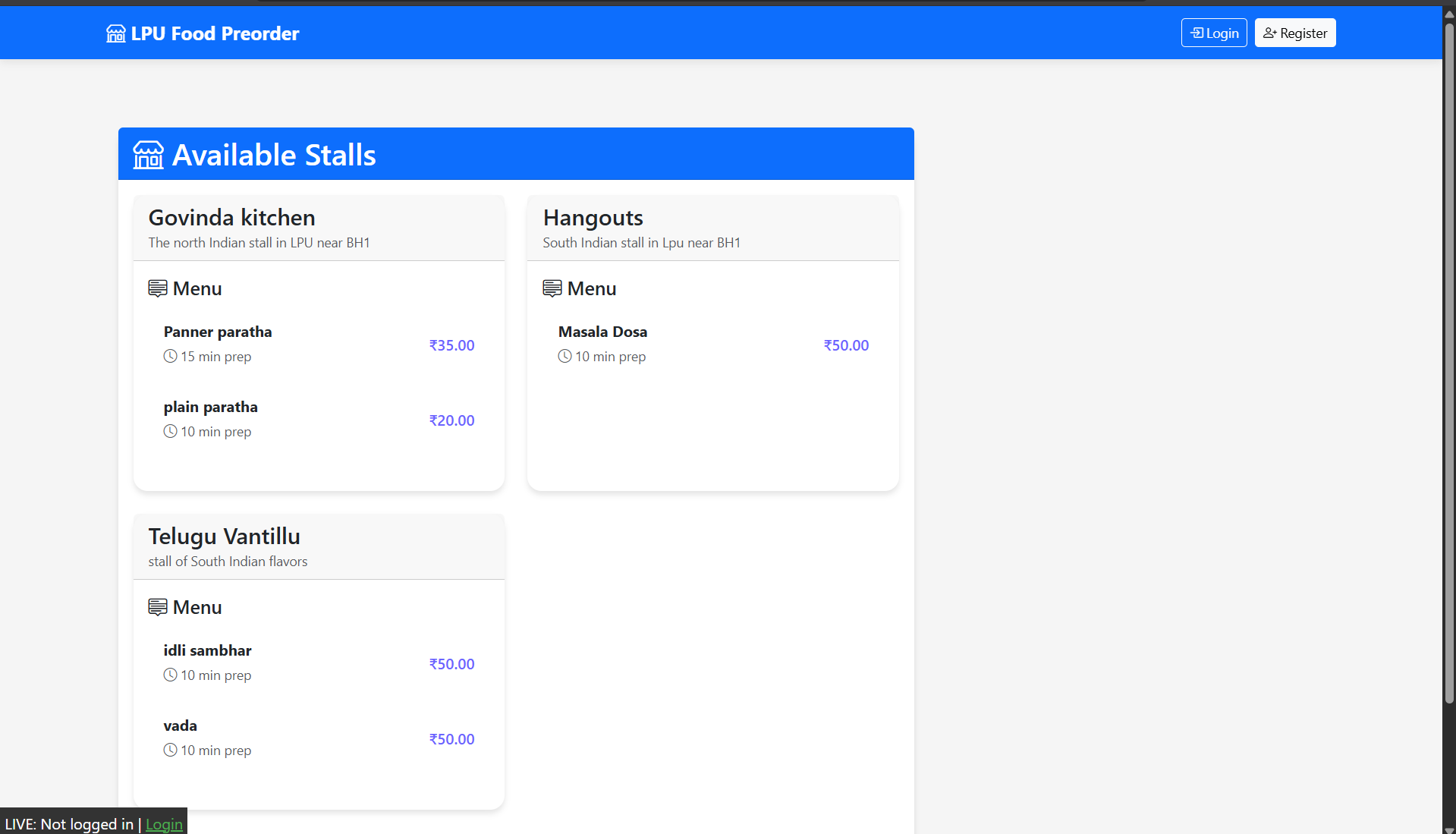
**Screenshots (Mock Description)**

**Login Page**: Email/Password

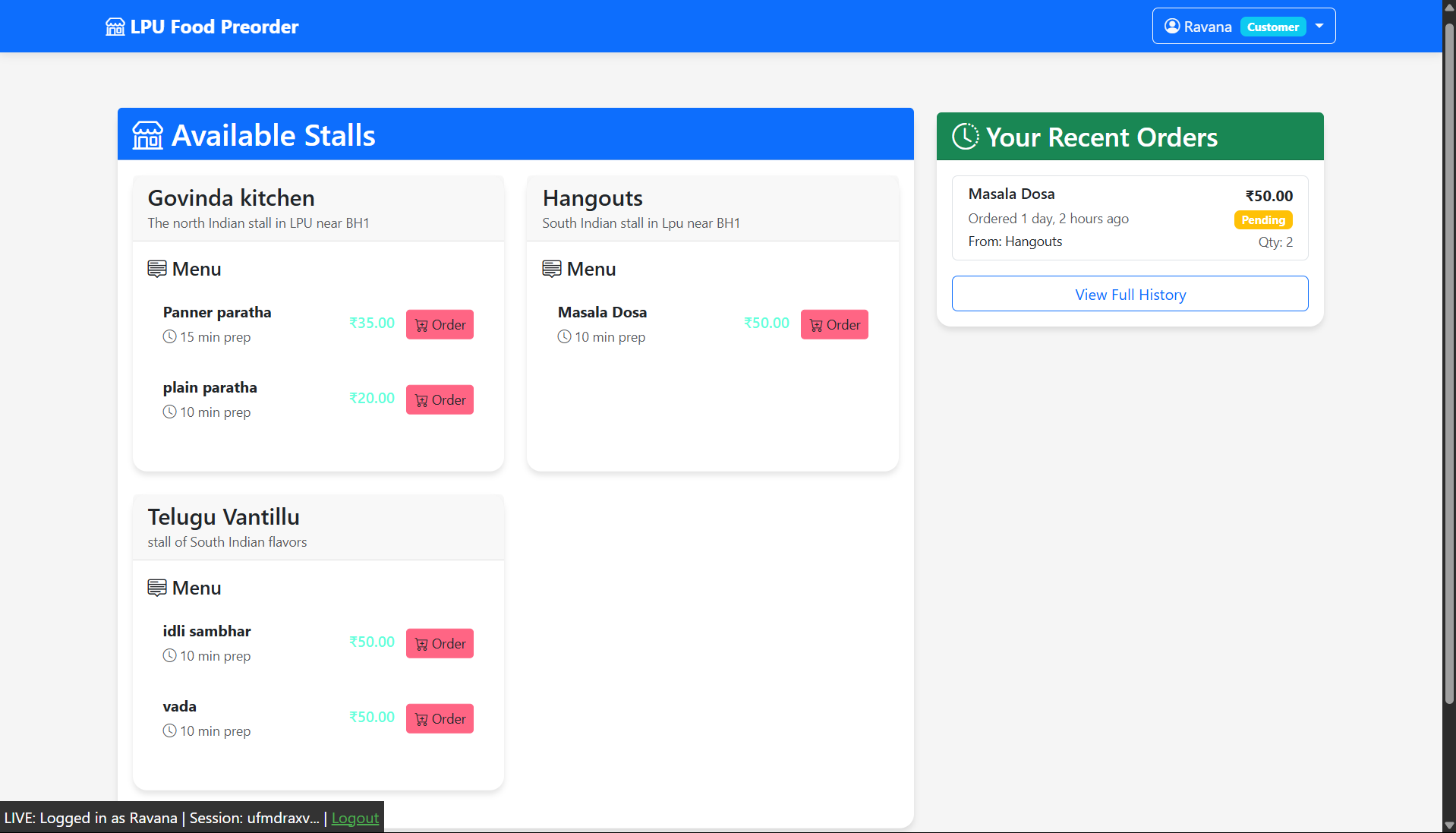


**Dashboard**:

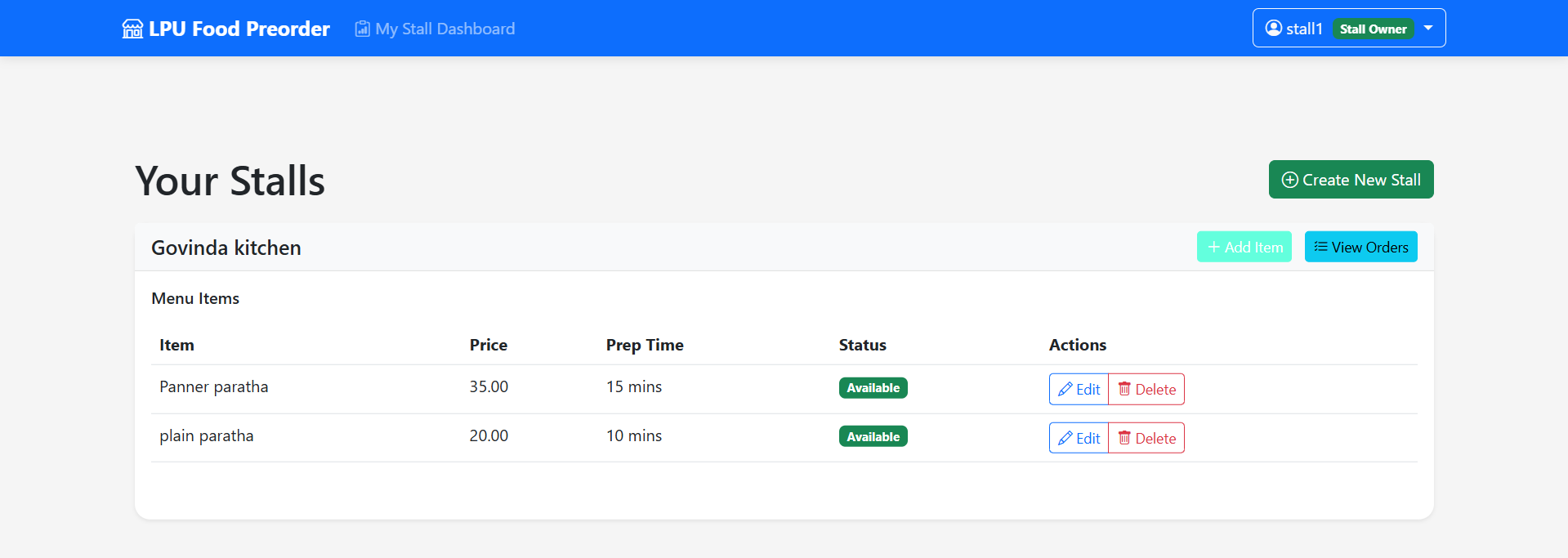
Without login :



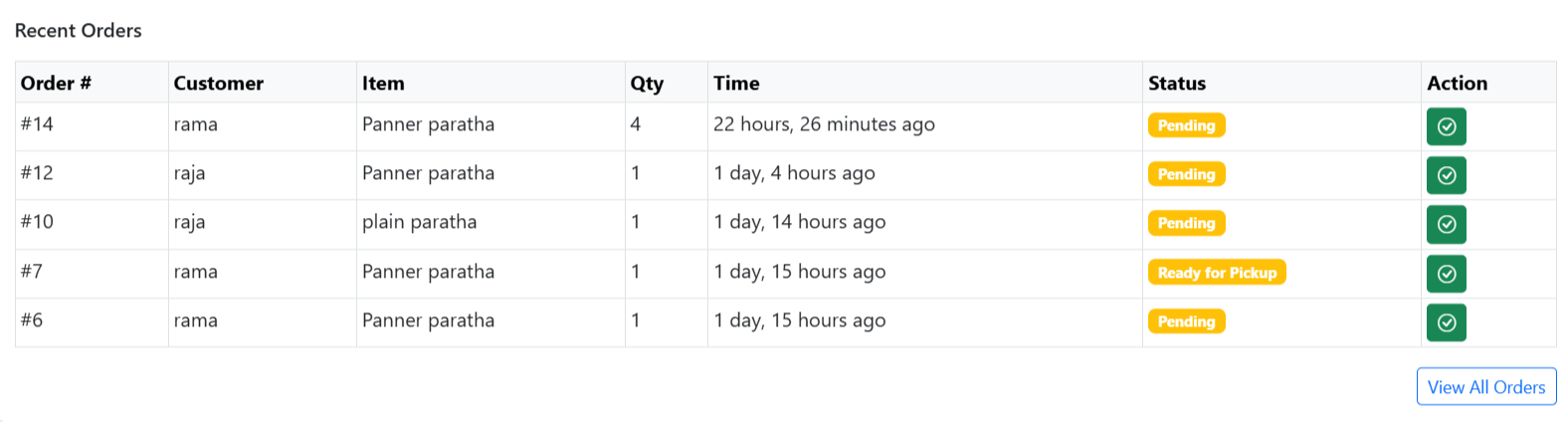
After login as user :



After login as stall vendor :



Vendors can see orders:



Project 2:

**Disaster Preparedness Hub with API Integration**

**Introduction**

A centralized platform aggregating real-time disaster alerts (floods, earthquakes, etc.) via APIs, providing preparedness checklists, and emergency contacts.

**Problem Identification**

* Lack of localized, real-time disaster updates.
* No standardized preparedness guides for communities.

**Proposed Solution**

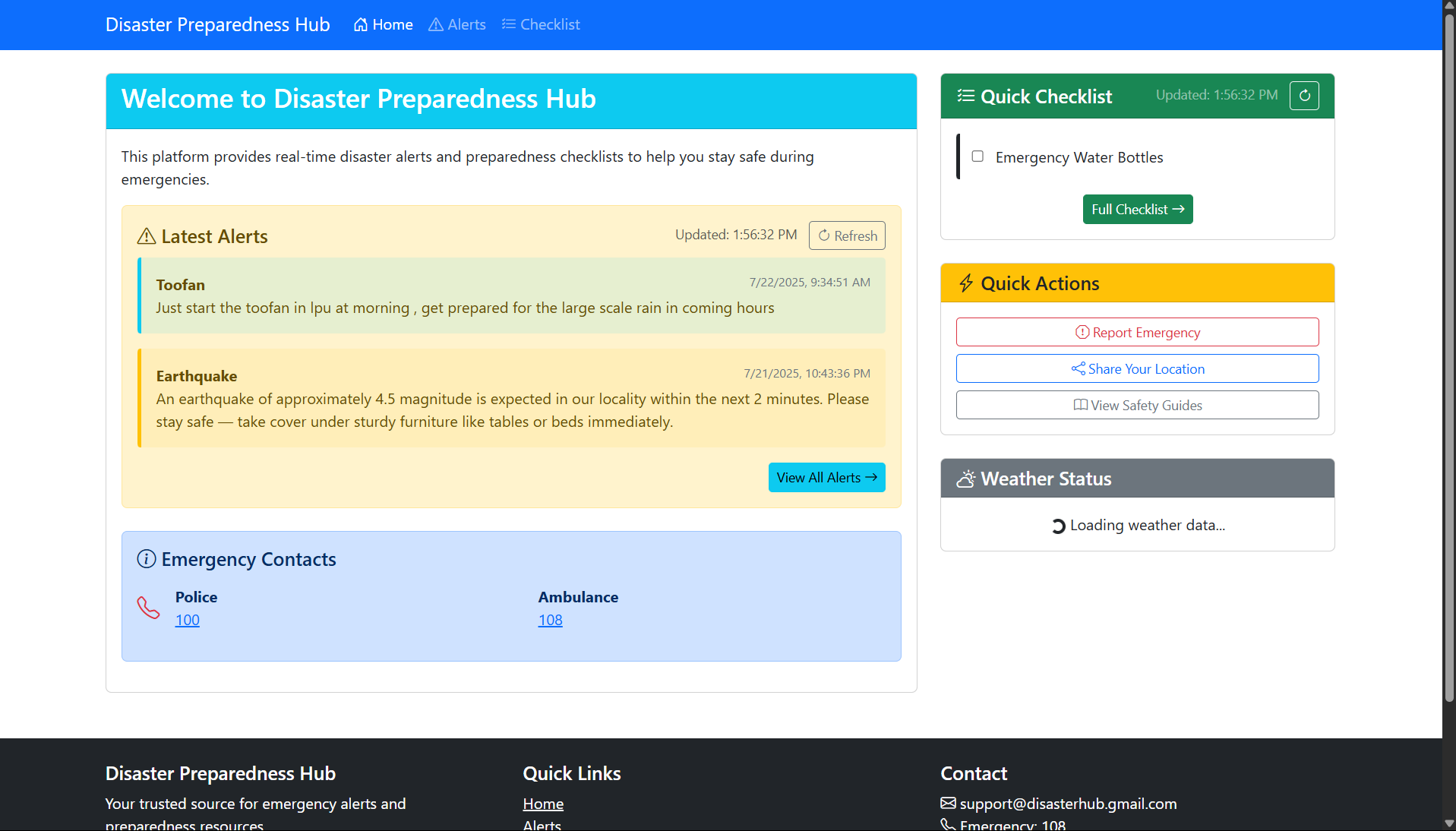
* **API Integration**: Fetch alerts from sources.
* **Checklists**: Customizable preparedness plans (e.g., "Earthquake Kit").
* **User Subscriptions**: Email/SMS alerts for specific regions.
* **Frontend**: HTML/CSS/JavaScript (vanilla or lightweight framework like Alpine.js).
* **Backend**: Django REST Framework (DRF) for building custom APIs.
* **Communication**: Fetch/Axios calls from frontend to Django endpoints.

**Tech Stack:**

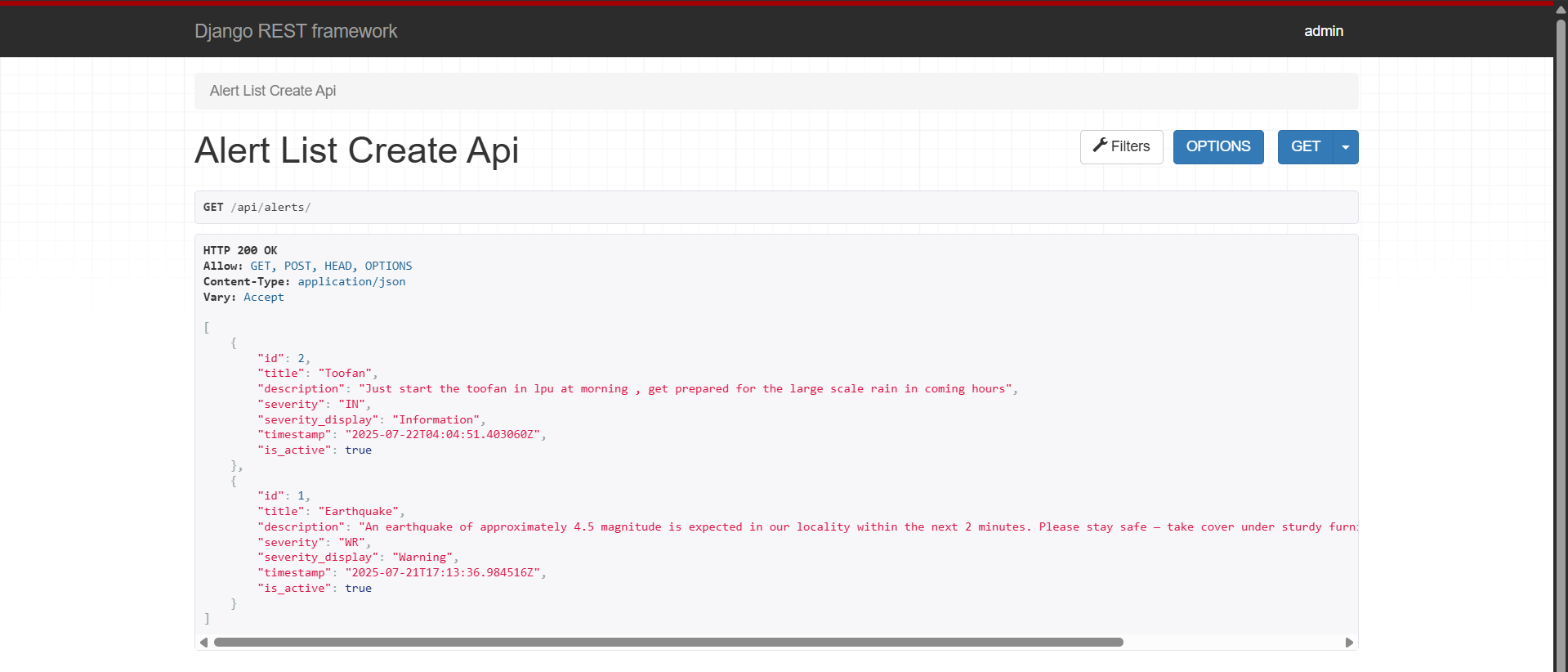
* Frontend: HTML/CSS/JavaScript
* Backend: Django REST Framework (DRF) for building custom APIs.
* Communication: Fetch

**Screenshots (Mock Description):**

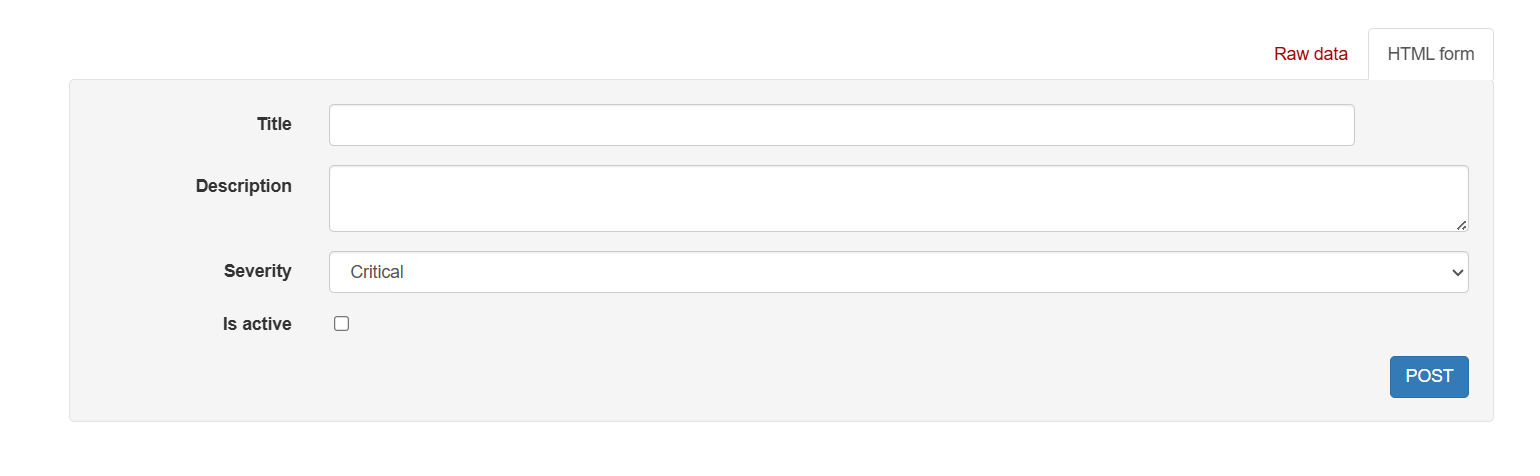
**Main Dashboard:**



Creating the post via Admin



\



**Conclusion:**

The **Preordering System** successfully addressed the key issues of long queues and inefficient manual order-taking in college stalls. By implementing role-based access, students can now **schedule meal pickups**, vendors can **efficiently manage orders**, and administrators can **oversee the system holistically**.

The project introduced a **user-friendly interface**, backed by Django, allowing **secure logins, real-time notifications, and seamless order workflows**. The solution not only improves operational efficiency but also enhances the overall student experience by saving time and reducing class disruptions.

The **Disaster Preparedness Hub** project culminated in a centralized platform that integrates real-time disaster alerts, emergency checklists, and weather information. By using the Django REST Framework and frontend technologies like HTML, CSS, and JavaScript, it ensures **users receive up-to-date alerts**, while also offering **preparedness guides for different disaster types**.

Through a clean and accessible interface, users can view **checklists for earthquakes, floods, etc., see critical alerts**, and access **emergency contact details**, enhancing personal and community safety.