**Product Requirements Document (PRD)**

**Project 1: Preordering System for College Stalls**

**1. Overview**

A Django-based web application that allows students to pre-order items from college stalls, skip queues, and pick up items in designated time slots. It includes secure authentication and role-based dashboards for users, stall owners, and admins.

**2. Goals and Objectives**

* Eliminate long queues in food stalls.
* Enable smooth and timely order pickup.
* Provide separate dashboards for different roles.
* Ensure real-time order updates and stall management.

**3. Key Features**

* **Role-based login** (Student, Stall Owner, Admin)
* **Order placement and scheduling**
* **Stall item management** by owners
* **Admin monitoring panel**
* **Order tracking and confirmation**

**4. User Roles**

* **Student**: Browse stalls, place preorders.
* **Stall Owner**: Manage menu and order queue.
* **Admin**: Manage all users, stalls, and activities.

**5. Technical Stack**

* **Backend**: Django + SQLite
* **Frontend**: HTML, CSS, JS
* **Auth**: CustomUser with user\_type

**6. Success Metrics**

* Reduced student wait time
* Efficient vendor operations
* Admin access to complete system control

**Project 2: Disaster Preparedness Hub**

**1. Overview**

A public-facing, API-integrated platform that provides users with real-time disaster alerts, preparedness checklists, and emergency contact information. It is built using Django REST Framework and includes frontend views in plain HTML/CSS/JS.

**2. Goals and Objectives**

* Provide real-time alerts for disasters like earthquakes or floods.
* Offer safety checklists to help users prepare.
* Display emergency contacts and weather information.

**3. Key Features**

* **Alert System** (severity-based: info, warning, critical)
* **Preparedness Checklist** (earthquake, flood, etc.)
* **Static frontend site** for public access

**4. Technical Stack**

* **Backend**: Django + DRF
* **Frontend**: HTML, CSS, JS
* **API**: Integrating Backend to frontend

**5. Modules & Endpoints**

* /api/alerts/ - Alert List & Creation
* /api/checklist/ - Checklist List & Creation
* /api/weather/ - Weather Alerts (via integration)

**6. User Flow**

* User lands on homepage
* Sees latest alerts and checklist
* Can click to view full alerts or full checklist
* Weather data and emergency contacts are visible

**7. Success Metrics**

* Active user visits during live disasters
* Community adoption in institutions
* Public feedback on usability and preparedness

**Common Considerations**

* **Responsive Design**: All frontends adapt across devices.
* **Security**: Django's in-built security features enabled.
* **Testing**: Manual testing and endpoint verification.

**Prepared by:** Bhavani Prasad Gummadala  
**Date:** July 22, 2025