**Project Documentation: Multi-Organizational Management System**

**1. Project Structure**

bash

Copy code

multi\_org\_system/

├── multi\_org\_system/

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

│

├── organizations/

│ ├── migrations/

│ ├── templates/

│ │ ├── homepage.html

│ │ ├── login.html

│ │ ├── register.html

│ │ ├── organization\_users.html

│ │ └── assign\_role.html

│ ├── models.py

│ ├── views.py

│ ├── forms.py

│ ├── admin.py

│ └── urls.py

│

├── db.sqlite3

├── manage.py

└── requirements.txt

**2.URL Structure:**

* + /: Homepage listing all organizations.
  + /organization/<org\_id>/: Displays the user list for the selected organization.
  + /organization/<org\_id>/assign\_role/: Form for assigning roles.
  + /register/: User registration page.
  + /login/ and /logout/: Authentication routes.

### 3. Assumptions and Challenges

1. **Assumptions:**
   * Each user belongs to only one organization.
   * Only the main organization admin (superuser) can create and manage sub-organizations.
   * Role logic assumes three roles: **Admin**, **Editor**, and **Viewer**. However, only **Admin** and **Viewer** roles are implemented due to time constraints.
   * Users with the **Admin** role can manage users and assign roles within their organization, while **Viewer** users have read-only access.
2. **Challenges:**
   * The functionality for the **Editor** role (e.g., limited editing rights without full admin privileges) was planned but not implemented due to time limitations.
   * Ensuring seamless access control and enforcing security boundaries between organizations required careful testing and refinement.

.