Role-Based Task Management System

-Neeha Shirin

1. Introduction

This Django-based project is a simple Role-Based Task Management System. It enables three types of users: Super Admin, Admin, and User. The system allows admins to assign tasks to users, track progress, and receive task completion reports.

2. Features

- User Authentication and Role-Based Login
- Super Admin Dashboard
- Admin Dashboard with Task Assigning Capability
- User Dashboard to View and Update Task Status
- Task Status Tracking (Pending, In Progress, Completed)
- Validation for Task Completion Reports

3. User Roles

Super Admin

- Highest authority.
- Can log in and view the Super Admin dashboard.

Admin

- Can log in and manage users.
- Can assign and update tasks.

User

- Can view tasks assigned to them.
- Can mark tasks as completed with a completion report and worked hours.

4. Models

4.1 CustomUser

Inherits from AbstractUser. Adds a `role` field with choices: User, Admin, Super Admin.

Automatically sets `is_staff` to True if the role is Admin.

4.2 Task

- `title`: Title of the task
- `description`: Details of the task
- `assigned_to`: Foreign key to the user
- `due_date`: Deadline for the task
- `status`: Task progress status (Pending, In Progress, Completed)
- `completion_report`: Description after task is completed
- `worked_hours`: Number of hours worked
- Validation: Completion report and hours required when status is 'Completed'

5. Views & Functionalities

5.1 Common

- home
- role_login

5.2 User

- user_login
- user_dashboard
- update_task

5.3 Admin

- admin_login
- admin_dashboard
- admin_assign_task
- admin_update_task

5.4 Super Admin

- superadmin_login
- superadmin_dashboard

6. Templates Overview

- login_user.html
- login_admin.html
- login_superadmin.html
- user_dashboard.html
- admin_dashboard.html
- superadmin_dashboard.html
- home.html

7. How to Run the Project

Clone the repository

Create a virtual environment and activate it:

```
python -m venv env
source env/bin/activate  # Windows: env\Scripts\activate
```

Install dependencies:

```
pip install -r requirements.txt
```

Run migrations:

```
python manage.py makemigrations
python manage.py migrate
```

Create a superuser:

```
python manage.py createsuperuser
```

Run the server:

```
python manage.py runserver
```

8. Conclusion

This project demonstrates a simple yet functional task management system with role-based access control. It showcases clean code structure and a real-world use case of Django's class-based views, model relationships, and authentication system.

SUBMITTED BY: NEEHA SHIRIN