

SOFTWARE REQUIREMENT SPECIFICATION FOR GD SLOT BOOKING

Name	MURALIDHARAN M
Roll no	7376222IT204
Seat no	360
Project ID	40
Problem Statement	Group Discussion Slot Booking System

PROBLEM STATEMENT:

In educational and professional settings, effectively managing and participating in communication-focused events like group discussions, mock interviews, and self-introductions is essential but challenging. Administrators face difficulties in adjusting event schedules, setting participant limits, and ensuring consistent communication. Meanwhile, participants need a streamlined process for registering and receiving timely updates about event changes. There is a critical demand for an application that simplifies these processes, enabling both administrators and users to interact seamlessly, manage event details dynamically, and improve overall engagement through intuitive notifications and reminders. This application aims to enhance the organization of educational events, ensuring they are accessible, well-coordinated, and beneficial for all involved parties.

PURPOSE OF THE PROJECT:

The primary purpose of this project is to design and implement a specialized event management application tailored for educational and professional development settings. The application aims to facilitate the organization and participation in key communicative events such as group discussions, mock interviews, and self-introductions. By automating scheduling, registration, and notification processes, the application seeks to enhance the effectiveness of these events, improving participants' communication skills in a structured and supportive environment.

SCOPE OF THIS PROJECT:

The scope of the project encompasses the development of web-based software using HTML, CSS, JavaScript for the frontend, Django for the backend, and MySQL for the database. The application will cater to two main types of users: administrators and participants (users).

PROJECT FLOW:

1. Project Planning and Requirements Analysis

- Define Goals: Outline the main objectives for each type of user (admin and user).
- Gather Requirements: List functionalities based on the roles.
- Design Specifications: Create wireframes and data flow diagrams for visual representation.

2. Environment Setup

- Development Tools Installation: Install Python, Django, MySQL, and any necessary libraries or frameworks.
- Version Control Setup: Initialize a Git repository and set up GitHub for code sharing and collaboration.

3. Database Design

- Database Schema: Design tables for users, events, registrations, notifications, etc.

- Relationships: Define primary and foreign keys, along with indexing for optimized queries.
- MySQL Setup: Implement the schema in MySQL.

4. Backend Development (Django)

- Models: Create Django models that correspond to the database schema.
- Admin Interface: Utilize Django's admin functionalities to allow easy management of database entries by the admin.
- APIs/Views: Develop views and RESTful APIs for handling the business logic, such as user registration, event management, and notification dispatch.
- User Authentication: Implement user authentication and authorization for admin and regular users.

5. Frontend Development

- Static Pages: Use HTML and CSS to design the homepage, login, registration, and dashboard pages.
- Interactive Components: Employ JavaScript to add interactivity, such as form validations, dynamic content loading, and event handlers for user actions.
- AJAX: Use AJAX for asynchronous data requests to the backend without refreshing the webpage.

6. Integration

- Connect Backend and Frontend: Ensure that the frontend can successfully communicate with the backend through APIs.
- Session Management: Implement session management to maintain user state across different pages.

7. Notification System

- Email Setup: Integrate an email sending functionality for notifications using Django's email framework.
- Notification Logic: Implement logic to send notifications upon registration, and reminders before an event.

8. Testing

- Unit Testing: Write unit tests for both frontend and backend functions.
- Integration Testing: Ensure all parts of the application work together seamlessly.

- User Acceptance Testing: Conduct testing involving potential users to receive feedback and make necessary adjustments.

9. Deployment

- Deployment Server Setup: Configure a server for hosting the application.
- Deployment: Deploy the application on the server using tools like Gunicorn, Nginx, and Daphne if WebSockets are used.
- Database Migration: Migrate the database setup to the live server environment.

10. Maintenance and Upgrades

- Monitor Performance: Keep track of application performance and user feedback.
- Update Regularly: Apply security patches, update dependencies, and add new features based on user feedback.

FUNCTIONAL REQUIREMENTS:

FUNCTIONAL REQUIREMENTS FOR ADMIN:

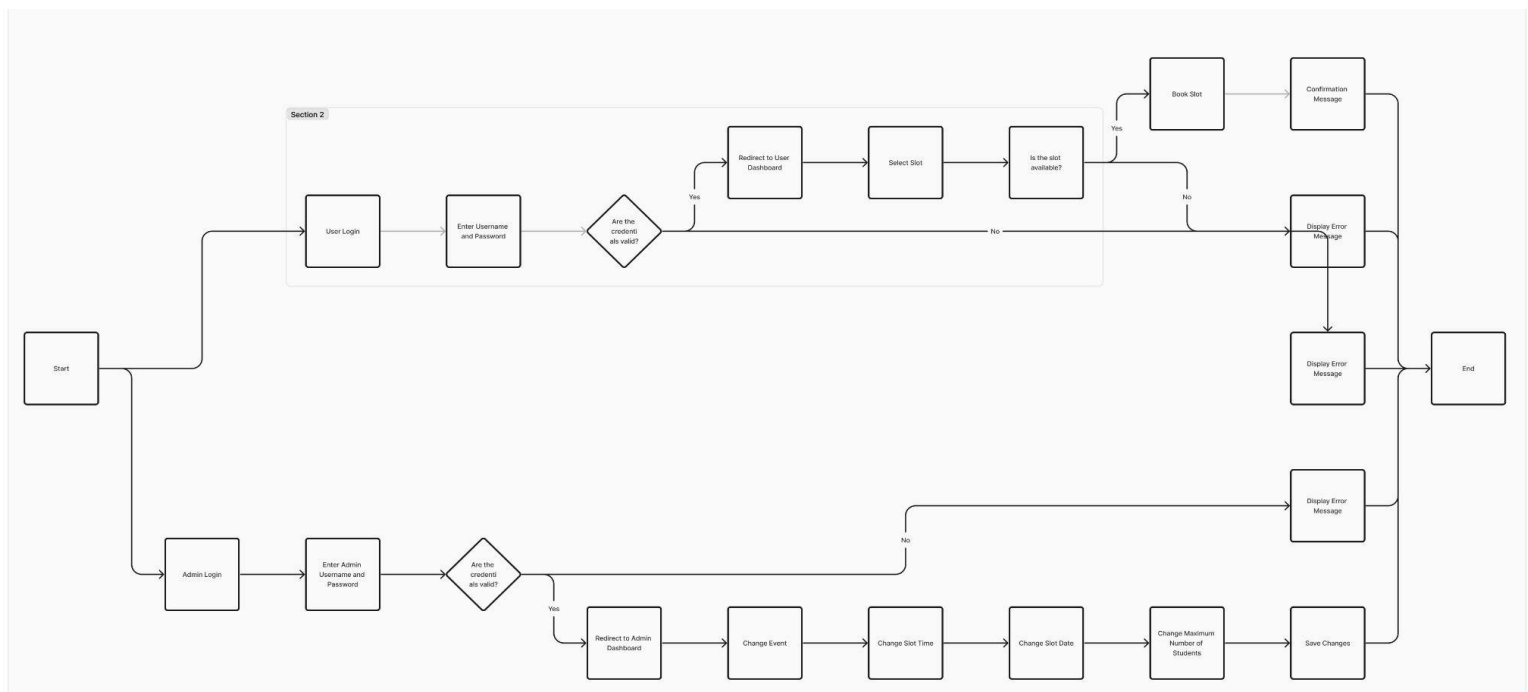
- Login/Logout Functionality: Secure access to an admin dashboard where they can manage events, users, and other settings.
- Event Management: Ability to create, update, and delete events. Set or modify the date, time, and capacity for events like group discussions, mock interviews, and self-introductions. Update event details once a week or as required.
- User Management: view and manage user registrations and attendance. Access to user profiles, including personal details and uploaded resumes. Ability to activate or deactivate user accounts.
- Batch Management: Create, update, and delete batches for different events. Assign users to specific batches based on their registration details.

- Faculty Management: Assign and manage faculty or moderators for each event. Update faculty assignments as necessary.
- Reporting: Generate reports on user registrations, attendance, and event status. Access to analytics for better decision-making regarding future events.

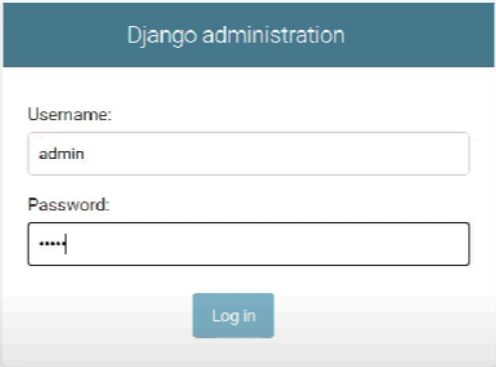
FUNCTIONAL REQUIREMENTS FOR USERS:

- Registration/Login/Logout: Secure registration and login system. Users can manage their profiles, including updating personal information and resumes.
- Event Registration: View available events, including date, time, and remaining slots. Register for events within the set limits (10 for group discussions, 2 for self-introductions, and 1 for mock interviews). Receive confirmation of registration via email or in-app notification.
- Notifications: Receive reminders for upcoming events they are registered for. Get notified if an event they are registered for is rescheduled or cancelled. Receive notifications about new events or changes in existing events.

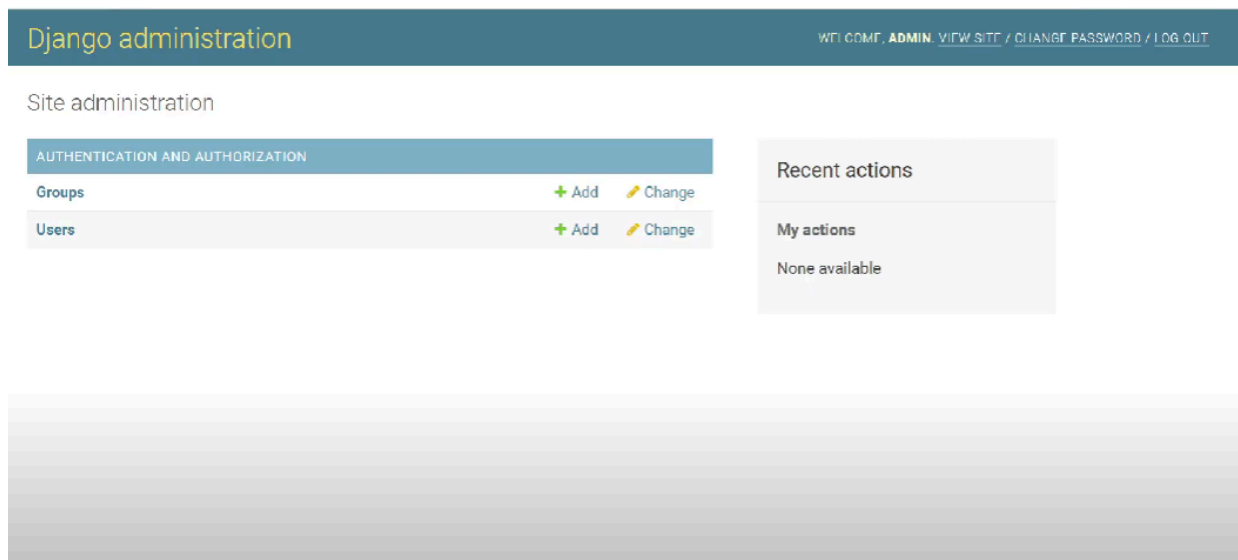
FLOW CHART :



ADMIN INTERFACE :



The image shows the Django administration login interface. It features a dark teal header with the text "Django administration". Below the header, there are two input fields: "Username:" with the value "admin" and "Password:" with masked characters "....". A blue "Log in" button is positioned below the password field. The entire form is centered on a light gray background.



The image shows the Django administration dashboard. At the top, there is a dark teal header with the text "Django administration" on the left and a navigation bar on the right containing links: "WELCOME", "ADMIN", "VIEW SITE", "CHANGE PASSWORD", and "LOG OUT". Below the header, the main content area is titled "Site administration". On the left, there is a section titled "AUTHENTICATION AND AUTHORIZATION" with two sub-sections: "Groups" and "Users". Each sub-section has a "+ Add" link and a "Change" link with a pencil icon. On the right, there is a section titled "Recent actions" with a sub-section "My actions" that displays "None available".