BANNARI AMMAN INSTITUTE OF TECHNOLOGY
An Autonomous Institution Affiliated to Anna University - Chennai, Accredited by NAAC with A+ Grade
Sathyamangalam - 638401 Erode District, Tamil Nadu, India
Stay Ahead

FULL STACK DEVELOPMENT – PROJECT WORKFLOW

Name	Ragul M
Department	Computer Science and Engineering
Year	III
Seat No	224
Reg. no	7376221CS273
Project ID	11
Project Name	Event Management System
Domain	IQAC - Events

Problem Statement

In our college, there is a specific department called 'Events team'. Their job is to allocate the venues for the events that would take place in our college like technical symposium, national or international level competition, project competition, tech talk, seminar, webinar and internal events like cultural events, dance and music event, games and some official purposes like placement training, administrative meeting, exam venue, etc.

For this purpose, the respective coordinator will approach the events team for venue allotment for their events. The 'Events team' will check for the availability of the venue, avoidance of date clashing, strength of the event, required space and looks for various combinations. After these, they will approve / reject the requisition and allot the venue. Additionally, they intend to provide accommodation, food and refreshments, vehicles for the guest and participants based on the requisition from coordinator.

In past, they have used a paper for filling these requirements and submit it to the 'Events team' physically. Now the event team has approached us to create them an end-to-end portal for this with suitable user authentication and authorization. The stacks to be used for this is given below:

Front - End	• HTML • CSS • JavaScript
Back - End	• Django • Python
Database	• PostgreSQL • MySQL
API	• OpenAPI • RESTful API • SoapAPI

User Roles

Event Coordinators:

Submit event requests, specify requirements such as venue, accommodation, food, transportation.

Events Team:

Approve / reject event requests, manage venues, schedule events, and allocate resources.

Guests / Participants:

May require a portal to view event details and potentially register for events.

Key Features

Authentication and Authorization:

Role-based access control (e.g., coordinators VS events team).

Event Request Submission:

Coordinators can submit requests for events, including details about the event.

Venue Management:

Events team can manage and allocate venues based on availability.

Schedule Management:

Avoid date clashes, manage calendar.

Resource Allocation:

Manage accommodation, food, transportation for events.

Notifications:

Notification via E-mail and SMS to the coordinator on the status.

Reporting:

Generate reports on events, venue usage, etc.

Technology Stacks

Front End:

HTML, CSS and JavaScript for building the user interface.

Frameworks / Libraries:

Using Bootstrap for styling and jQuery for easier DOM manipulation.

Back End:

Python and Django for server-side processing and handling business logic.

Database:

PostgreSQL or MySQL for storing events, venue and user data, etc.

APIs:

- RESTful API for communication between front-end and back-end.
- OpenAPI for API documentation.
- SoapAPI for integrating with other systems.

Detailed Architecture

Front End:

Creating user interfaces for:

- Event coordinators to submit requests and check status.
- Events team to manage requests, allocate venues, and resources.
- Admin dashboard for reports and system management

Back End:

Creating Django projects for:

- Models such as User, Event, Venue, Request, Resource (Accommodation, Food, Transport).
- Views to handle request submission, approval, resource allocation.
- Forms for input handling.
- Django REST Framework for API endpoints.
- Authentication using Django's built-in auth system with role-based access control.

Database Schema:

- Tables for Users, Roles, Events, Venues, Requests, Resources, etc.
- Relationships between tables (e.g., one-to-many between Venue and Events).

API Design:

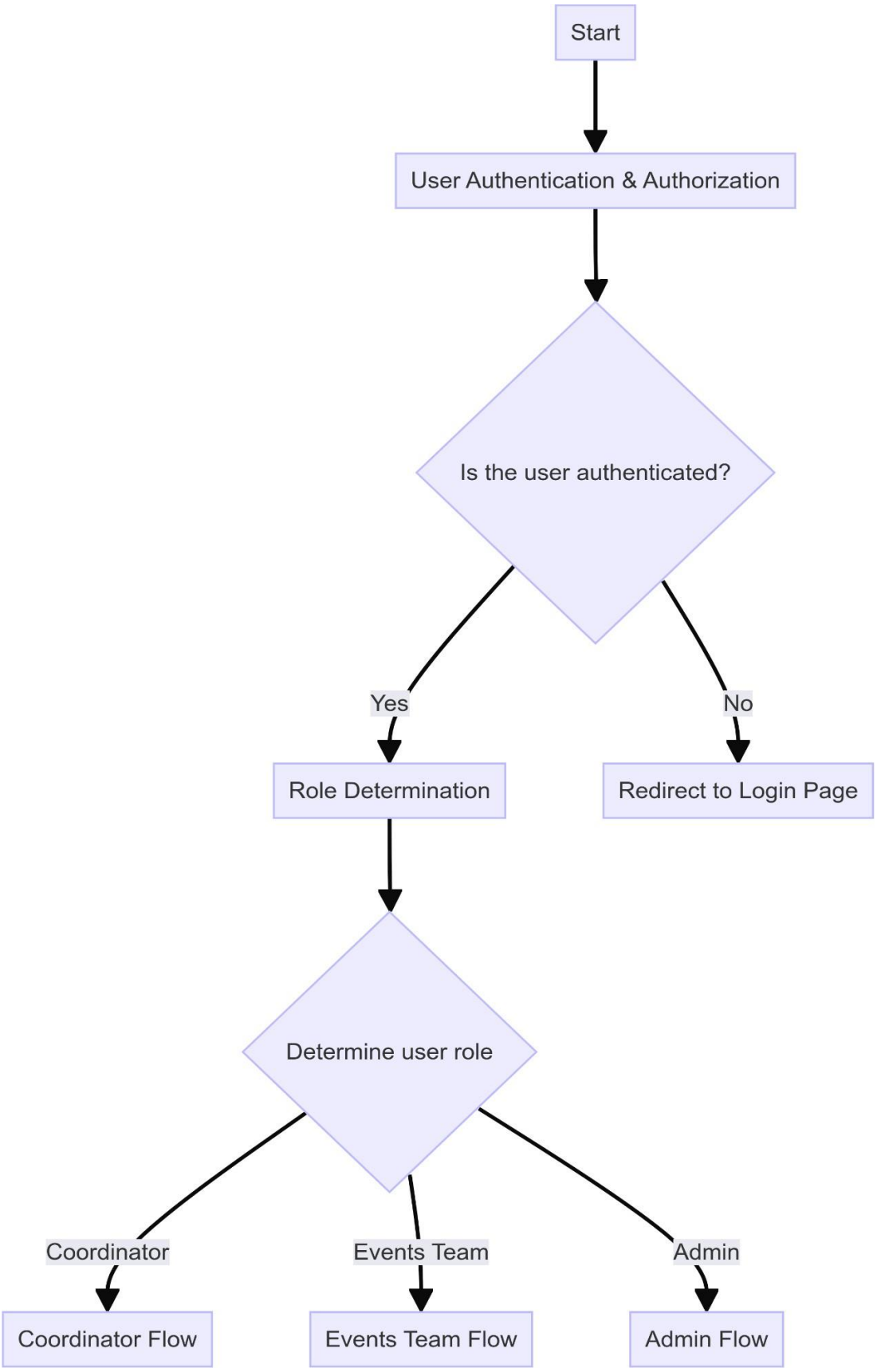
RESTful API Endpoints:

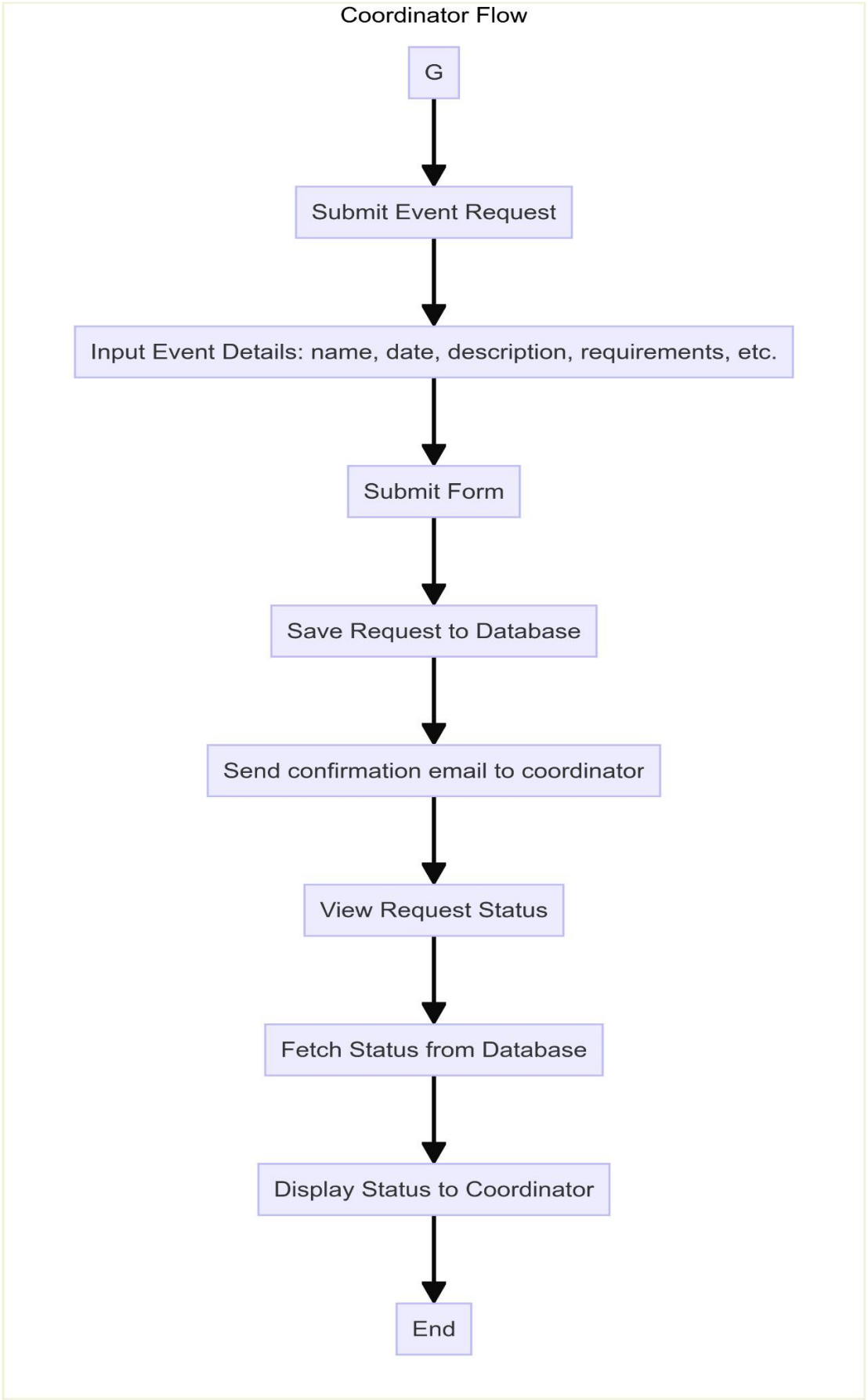
- POST /api/events: Create a new event request.
- GET /api/events: List all events.
- PUT /api/events/{id}: Update an event.
- DELETE /api/events/{id}: Delete an event.
- Other endpoints for venue management, resource allocation, etc.

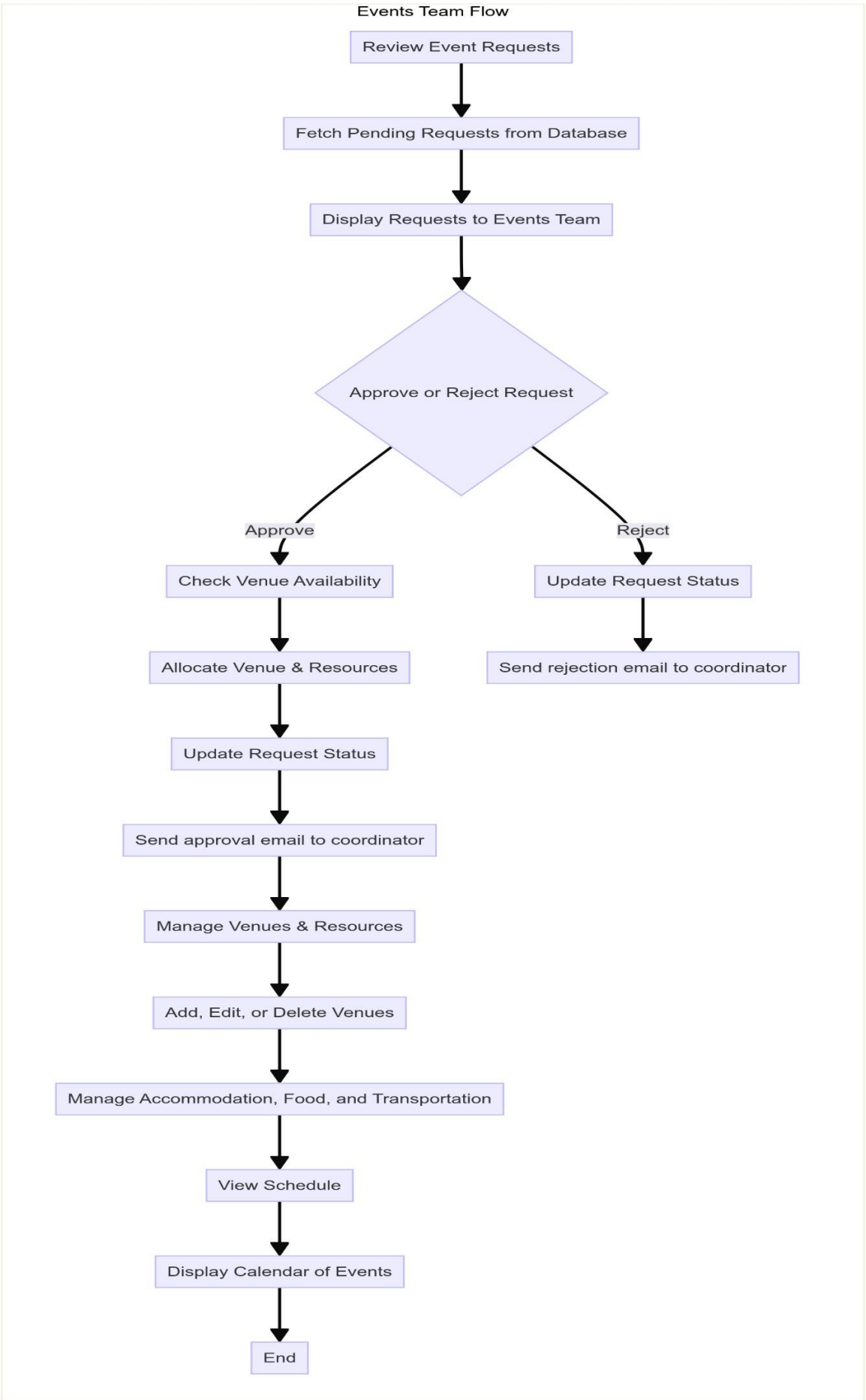
OpenAPI:

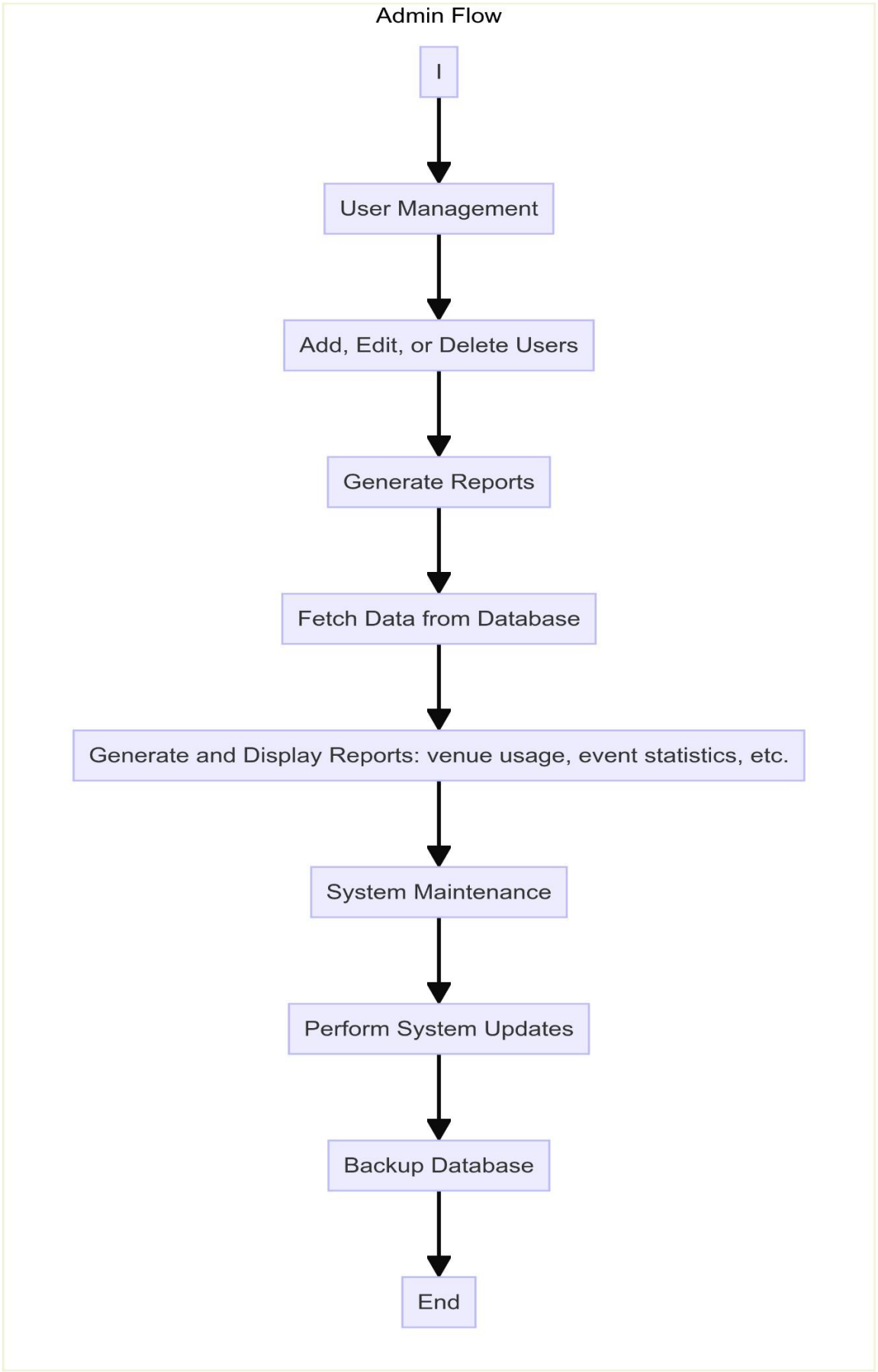
- Document these APIs for easy understanding and usage.

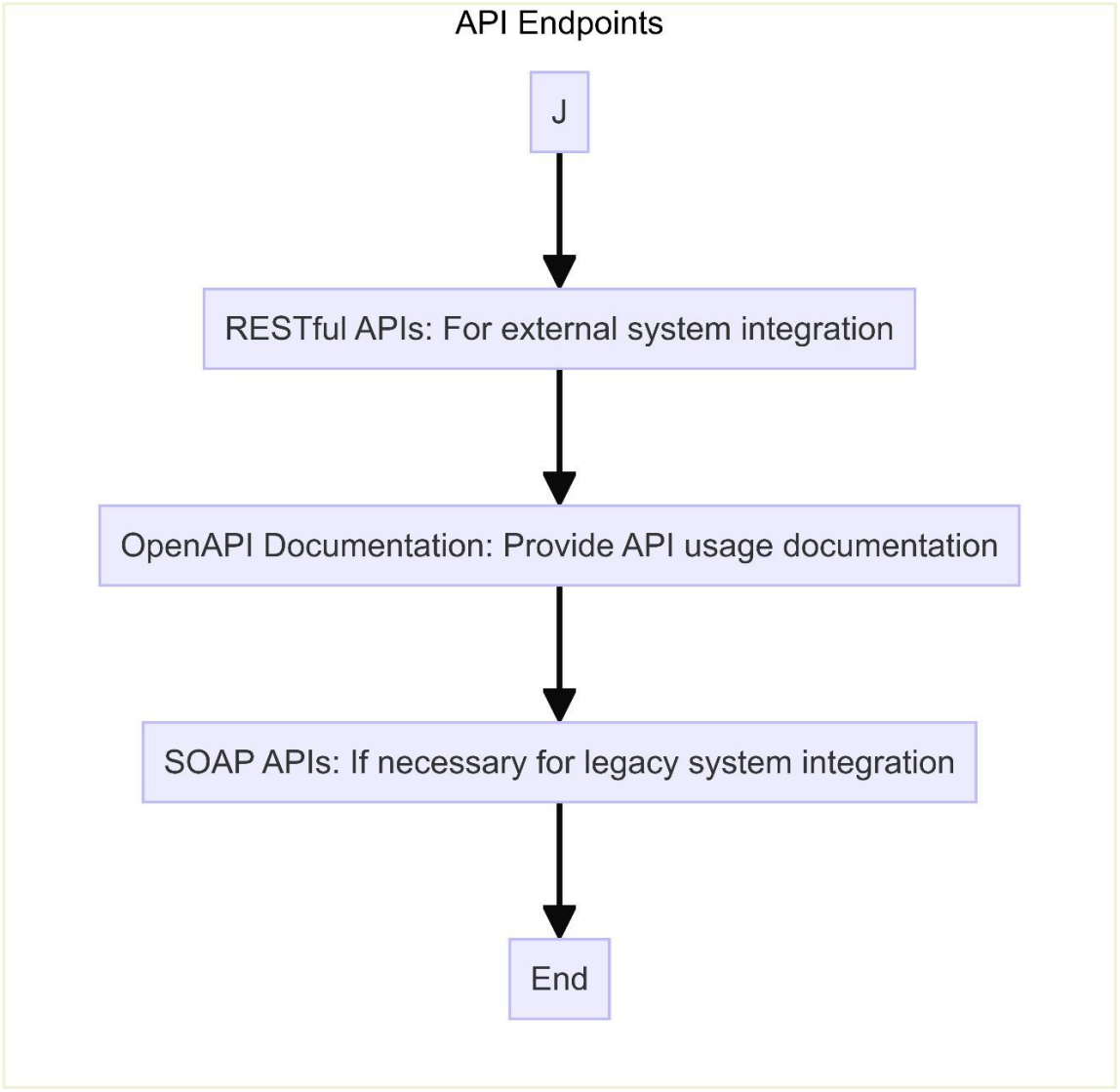
Flowchart


Start
User Authentication & Authorization
Is the user authenticated?
Yes
No
Role Determination
Redirect to Login Page
Determine user role
Coordinator
Events Team
Admin
Coordinator Flow
Events Team Flow
Admin Flow


Coordinator Flow
G
Submit Event Request
Input Event Details: name, date, description, requirements, etc.
Submit Form
Save Request to Database
Send confirmation email to coordinator
View Request Status
Fetch Status from Database
Display Status to Coordinator
End


Events Team Flow
Review Event Requests
Fetch Pending Requests from Database
Display Requests to Events Team
Approve or Reject Request
Approve
Reject
Check Venue Availability
Update Request Status
Allocate Venue & Resources
Send rejection email to coordinator
Update Request Status
Send approval email to coordinator
Manage Venues & Resources
Add, Edit, or Delete Venues
Manage Accommodation, Food, and Transportation
View Schedule
Display Calendar of Events
End


Admin Flow
I
User Management
Add, Edit, or Delete Users
Generate Reports
Fetch Data from Database
Generate and Display Reports: venue usage, event statistics, etc.
System Maintenance
Perform System Updates
Backup Database
End


API Endpoints
J
RESTful APIs: For external system integration
OpenAPI Documentation: Provide API usage documentation
SOAP APIs: If necessary for legacy system integration
End

Implementation Plan

Setting up Development Environment:

- Set up a Django project with PostgreSQL / MySQL database.
- Install necessary libraries and frameworks (Django REST Framework, etc.).

Developing Models:

- Define Django models for Users, Roles, Events, Venues, Requests, Resources.

Creating API Endpoints:

- Use Django REST Framework to create RESTful APIs.
- Document APIs using OpenAPI.

Developing Frontend:

- Create HTML templates, CSS styles, and JavaScript for interactive UIs.
- Implement forms for event request submission and management.

Implementation of Authentication and Authorization:

- Use Django's auth system to manage user authentication.
- Implement role-based access control for different user types.

Testing:

- Write unit tests for backend logic.
- Perform integration testing for API endpoints.
- Conduct user acceptance testing with the events team as well as the event coordinators.

Deployment:

- Deploy the application on a web server.
- Ensure database migrations are handled.
- Set up a production-ready environment with necessary configurations.

Additional Considerations

Scalability:

- Ensure the system can handle large number of events and users.

Security:

- Implementing security best practices (e.g., input validation, secure authentication).

Usability:

- Focusing on a user-friendly interface for coordinators and events team.

Maintenance:

- Finding a low-traffic window for regular maintenance and updates.