**Online Event Management – Design Doc**

To design an Online Event Management System with the outlined features, here's a structured approach:

System Architecture:

1. Frontend: Not implemented

2. Backend:

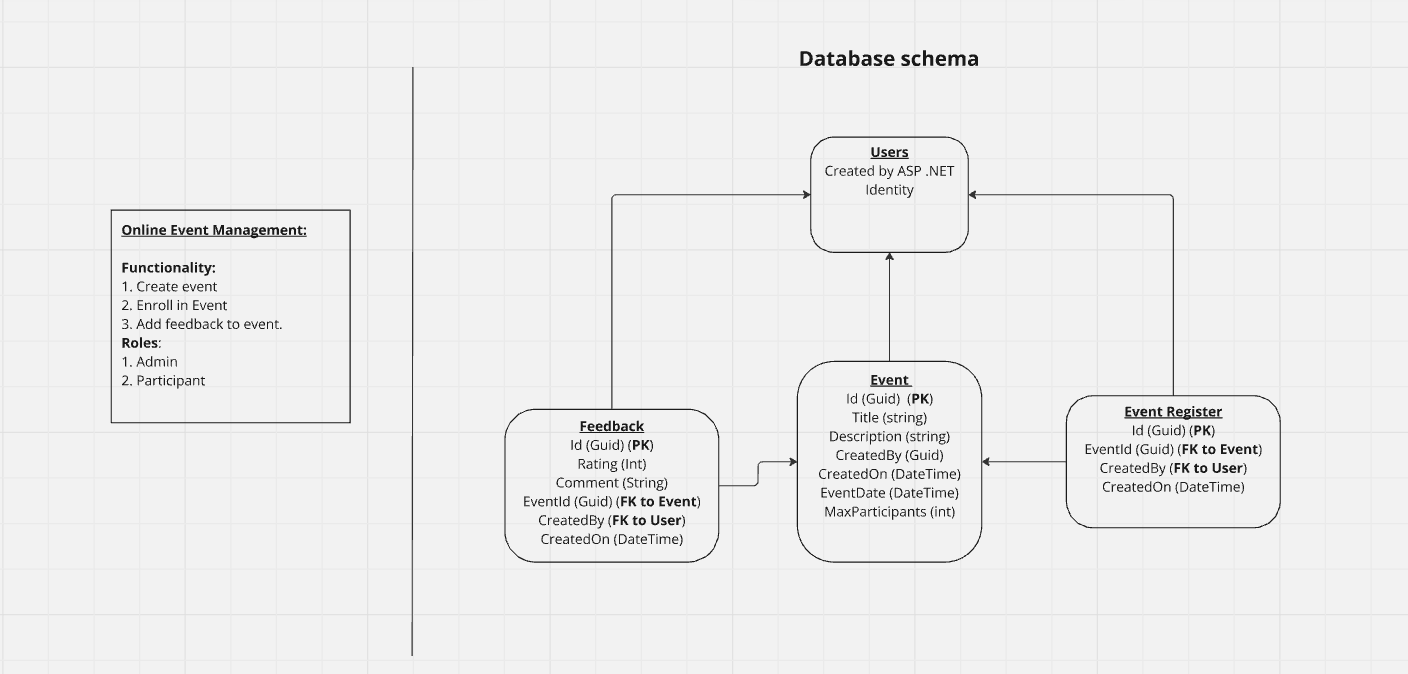
- .NET/.NET Core with C#: For handling business logic, data manipulation, and API endpoints.

- Data Access Layer: ADO.NET & Identity EF Core (Authentication)

- Authentication & Authorization: Implement roles (Admin, Participant) with ASP.NET Identity.

3. Database:

- SQL Server: Store user data, event data, participant data, etc.

Detailed Design  
  


1. User Roles & Authentication

- Admin:

- Create and manage events.

- View participants and their feedback.

- Set a maximum limit for event participants.

- Participant:

- View, filter, and sort events.

- Register for events (subject to availability).

- Provide feedback on events they’ve attended.

Implementation:

- Use ASP.NET Identity for managing user authentication and roles.

2. Event Management

- Event Entity:

- `EventId` (int, Primary Key)

- `Title` (string)

- `Description` (string)

- `Date` (DateTime)

- `MaxParticipants` (int)

- `CurrentParticipants` (int)

- `AdminId` (string, Foreign Key to Users table)

CRUD Operations:

- Create Event: Admin users can create events, setting the maximum participants limit.

- Read Events: All users can view event details.

- Update Event: Only the admin of the event can modify its details.

- Delete Event: Admins can delete events they created.

Overbooking Prevention:

- Implement a check in the event registration process to prevent more participants than allowed.

3. Event Registration & Feedback

- Registration Entity:

- `RegistrationId` (int, Primary Key)

- `EventId` (int, Foreign Key)

- `UserId` (string, Foreign Key)

- `RegistrationDate` (DateTime)

- Feedback Entity:

- `FeedbackId` (int, Primary Key)

- `EventId` (int, Foreign Key)

- `UserId` (string, Foreign Key)

- `Rating` (int)

- `Comments` (string)

Implementation:

- Use ADO.NET handle these entities.

- Only allow feedback from participants who attended the event.

4. Event Filtering & Sorting

- Filtering:

- Allow users to filter events by date, category, or availability.

- Sorting:

- Default: Sort events by the earliest date.

- Users can choose to sort by popularity, registration deadline, etc.

Implementation:

- Use LINQ in C# to perform filtering and sorting operations on the event list.

5. Hierarchy Visualization

- Create a React component that visualizes the event hierarchy as a tree structure.

- This can represent relationships between events or sub-events.

Sample API Endpoints

1. User Authentication:

- `POST /api/auth/register`: Register a new user.

- `POST /api/auth/login`: Authenticate user and generate JWT.

2. Event Management:

- `GET /api/events`: Get a list of events (filtering and sorting via query parameters).

- `GET /api/events/{id}`: Get details of a specific event.

- `POST /api/events`: Create a new event (Admin only).

- `PUT /api/events/{id}`: Update event details (Admin only).

- `DELETE /api/events/{id}`: Delete an event (Admin only).

3. Event Registration and Feedback:

- `POST /api/events/{id}/register`: Register a user for an event.

- `POST /api/events/{id}/feedback`: Submit feedback for an event.