EMS

Introduction

EMS (Event Management System) is a platform that provides users to Create and Publish Events and allow General Public to come and join the Organizer. An Event can be any General Public Gathering like Party, Concert, Celebrity Event or anything like that.
Organizer will create an event with General details like venue, capacity, ticket price, date and time. Visitors can go for that event after buying the ticket of that particular event for a fixed price. After the Event, the Organizer can edit the Status and upload the event images. Similarly, an Event Visitor can post a review for that event for transparency and creating a profile for that Organization as well.

Modules

Mainly there are two users in the Platform

1. User (**Visitor**)
2. Admin (**Organizer**)

1. Auth Module:

The Auth Module is responsible for user authentication and authorization. It includes functionalities for user registration, login, password management, and role-based access control. Both Visitors and Organizers will need to authenticate themselves to access the platform's features.

Key features:
- User registration (Visitor and Organizer)
- Login/logout functionality with cookies and localstorage
- Role-based access control to differentiate between Visitors and Organizers
- Maintaining Profile of both User Types and storing profile images

2. Events Module:

The Events Module allows Organizers to create, manage, and publish events. Visitors can browse, search, and purchase tickets for these events. After the event, Organizers can update event details and status, and Visitors can post reviews.

Key features:
- Event creation and management (venue, capacity, ticket price, date, time)
- Ticket purchasing
- Event updates (status changes, image uploads to cloud)
- Event details page accessible to both Visitors and Organizers

3. Organizations Module:

The Organizations Module manages the profile and details of the event Organizers. It allows Organizers to maintain their profiles and showcase their past events and reviews. Visitors can view Organizer profiles to gain insights into their reliability and event quality.

Key features:
- Organizer profile creation and management
- Display of past events organized
- Display of reviews and ratings received from Visitors

4. Searching Module:

The Searching Module enables Visitors to find events based on various criteria. It provides search functionality by event type, date, location, and other filters. This module ensures that Visitors can easily discover events that match their interests.

Key features:
- Advanced search options (by date, location, event type, time, organization, etc.)
- Filtering and pagination of search results
- Keyword-based search functionality

5. Review Module:

The Review Module allows Visitors to post reviews and ratings for events they have attended. This feature fosters transparency and helps build the reputation of Organizers. Reviews include written feedback and star ratings, contributing to the Organizer's profile.

Key features:
- Posting reviews and ratings for attended events
- Viewing reviews on event and Organizer profiles
- Moderation tools for managing inappropriate reviews
- Aggregate ratings for events and Organizers

6. Analysis Module:

The Analysis Module provides analytics to admins at two levels: single event level and overall platform level.

Single Event Level:

- Revenue and ticket sales analysis
- Review and rating summaries

Overall Platform Level:

- Revenue and financial summaries
- Common event categories of all of your events
- Number of Events Created By Organization

Database Schema

1. User Schema

```
const userSchema = new mongoose.Schema(
   {
       name: {
           type: String,
           required: [true, "Please enter your name"],
       },
       email: {
           type: String,
           required: [true, "Please enter your email"],
           validate: {
               validator: function (value) {
                   return emailRegexPattern.test(value);
               },
               message: "please enter a valid email",
           },
           unique: true,
       },
       password: {
           type: String,
           required: [true, "Please enter password"],
           validate: {
               validator: function (value) {
                   return passwordRegexPattern.test(value);
               },
               message: "please enter a valid password (min 6 characters)
with at least one uppercase letter, one lowercase letter, and one number",
           },
           select: false,
       },
       role: {
           type: String,
           enum: ["user", "admin"],
           default: "user",
       },
       avatar: {
           type: String,
           default:
"https://images.rawpixel.com/image_png_social_square/cHJpdmF0ZS9sci9pbWFnZ
XMvd2Vic2l0ZS8yMDIzLTAxL3JtNjA5LXNvbGlkaWNvbi13LTAwMi1wLnBuZw.png",
       },
       bio: {
           type: String,
           required: [true, "Please enter your bio"],
       },
       tickets: [{
           type: mongoose.Schema.Types.ObjectId,
           ref: "Ticket",
       }],
       events: [{
           type: mongoose.Schema.Types.ObjectId,
           ref: "Event",
       }],
   },
   { timestamps: true }
);
```

2. Event Schema

```
const eventSchema = new mongoose.Schema(
    {
        title: {
            type: String,
            required: [true, "Please enter event title"],
        },
        description: {
            type: String,
            required: [true, "Please enter event description"],
        },
        date: {
            type: Date,
            required: [true, "Please enter event date"],
        },
        startTime: {
            // type: String,
            type: Number,
            required: [true, "Please enter event start time"],
        },
        endTime: {
            // type: String,
            type: Number,
            required: [true, "Please enter event end time"],
        },
        location: {
            type: String,
            required: [true, "Please enter event location"],
        },
        organizer: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "User",
            required: true,
        },
        capacity: {
            type: Number,
            required: [true, "Please enter event capacity"],
        },
        category: {
            type: String,
            required: [true, "Please enter event category"],
        },
        tickets: [{
            type: mongoose.Schema.Types.ObjectId,
            ref: "Ticket",
        }],
        reviews: [{
            type: mongoose.Schema.Types.ObjectId,
            ref: "Review",
        }],
        ticketPrice: {
            type: Number,
            required: [true, "Please enter ticket price"],
        },
        status: {
            type: String,
            enum: ["upcoming", "ongoing", "past"],
            default: "upcoming",
        },
        media: [{
            type: String,
        }],
        numberOfTicketsSold: {
            type: Number,
            default: 0,
        },
    },
    { timestamps: true }
);
```

3. Ticket Schema

```
const ticketSchema = new mongoose.Schema(
    {
        event: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "Event",
            required: true,
        },
        user: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "User",
            required: true,
        },
        price: {
            type: Number,
            required: [true, "Please specify ticket price"],
        },
        purchaseDate: {
            type: Date,
            default: Date.now,
        },
        status: {
            type: String,
            enum: ["purchased"],
            default: "purchased",
        },
    },
    { timestamps: true }
);
```

4. Review Schema

```
const reviewSchema = new mongoose.Schema(
   {
       event: {
           type: mongoose.Schema.Types.ObjectId,
           ref: "Event",
           required: true,
       },
       user: {
           type: mongoose.Schema.Types.ObjectId,
           ref: "User",
           required: true,
       },
       rating: {
           type: Number,
           required: true,
           min: 1,
           max: 5,
       },
       comment: {
           type: String,
           required: true,
       },
   },
   { timestamps: true }
);
```