**Functional Requirements**

1. **User Management**
   * Users can register and log in.
   * Users can have roles: Author, Moderator, and Admin.
   * Users can update their profile information.
2. **Event Management**
   * Admins can create, update, and delete events.
   * Events have a title, description, start date, end date, and status (upcoming, ongoing, finished).
3. **Section Management**
   * Admins can create, update, and delete sections within events.
   * Sections have a title and description.
4. **Submission Management**
   * Authors can choose a section and upload their research file.
   * Submissions have a title, description, file upload, and status (pending, accepted, rejected).
5. **Review System**
   * Moderators can choose a section to review submissions.
   * Moderators can accept or reject submissions with comments.
6. **Winner Selection and Certificate Issuance**
   * Moderators can select first, second, and third-place winners in each section.
   * The system generates and issues certificates to all participants.

**Non-Functional Requirements**

1. **Security**
   * Implement authentication and authorization.
   * Secure file uploads and user data.
2. **Performance**
   * Optimize for fast load times and efficient data processing.
3. **Scalability**
   * Design to handle an increasing number of users and submissions.
4. **Usability**
   * Ensure the interface is user-friendly and accessible.
5. **Maintainability**
   * Write clean, modular, and well-documented code.

**Individual Tasks**

**1. Setup**

1. **Initialize Project**
   * Set up a new React.js project.
   * Set up a new Node.js project with Express.
   * Set up Prisma for database management.
2. **Database Design**
   * Design the database schema with tables for users, events, sections, submissions, reviews, and certificates.

**2. User Management**

1. **Authentication and Authorization**
   * Implement user registration and login with JWT.
   * Set up role-based access control.
2. **Profile Management**
   * Create endpoints for updating user profiles.
   * Create frontend components for profile management.

**3. Event Management**

1. **Event CRUD Operations**
   * Create backend endpoints for creating, reading, updating, and deleting events.
   * Create frontend components for managing events.
2. **Section Management**
   * Create backend endpoints for creating, reading, updating, and deleting sections within events.
   * Create frontend components for managing sections.

**4. Submission Management**

1. **File Upload**
   * Implement file upload functionality.
   * Ensure files are securely stored and accessible.
2. **Submission CRUD Operations**
   * Create backend endpoints for submitting research.
   * Create frontend components for authors to submit their research.

**5. Review System**

1. **Review Submissions**
   * Create backend endpoints for moderators to review submissions.
   * Create frontend components for the review process.
2. **Comment System**
   * Implement a system for moderators to leave comments on submissions.

**6. Winner Selection and Certificate Issuance**

1. **Winner Selection**
   * Create backend endpoints for selecting winners.
   * Create frontend components for moderators to select winners.
2. **Certificate Generation**
   * Implement a system to generate certificates for participants.
   * Create frontend components to display and download certificates.

**7. Miscellaneous**

1. **UI/UX Design**
   * Design a user-friendly interface for all user roles.
   * Ensure the design is responsive and accessible.
2. **Testing**
   * Write unit and integration tests for backend and frontend.
   * Perform user testing to gather feedback and make necessary adjustments.
3. **Deployment**
   * Set up a production environment.
   * Deploy the application to a cloud service like AWS, Heroku, or Vercel.

**Detailed Task Breakdown**

**1. Initialize Project**

1. **Frontend Setup**
   * Create a new React.js project using Create React App.
   * Install necessary dependencies (React Router, Axios, etc.).
2. **Backend Setup**
   * Create a new Node.js project.
   * Install Express and necessary middleware (CORS, body-parser, etc.).
   * Set up Prisma and connect to your database (e.g., PostgreSQL).

**2. Database Design**

1. **Define Prisma Schema**

prisma

Copy code

model User {

id Int @id @default(autoincrement())

email String @unique

password String

role Role

profile Profile?

submissions Submission[]

reviews Review[]

}

model Profile {

id Int @id @default(autoincrement())

userId Int @unique

user User @relation(fields: [userId], references: [id])

name String

bio String?

}

model Event {

id Int @id @default(autoincrement())

title String

description String

startDate DateTime

endDate DateTime

status EventStatus

sections Section[]

}

model Section {

id Int @id @default(autoincrement())

title String

description String

eventId Int

event Event @relation(fields: [eventId], references: [id])

submissions Submission[]

}

model Submission {

id Int @id @default(autoincrement())

title String

description String

fileUrl String

status SubmissionStatus

sectionId Int

section Section @relation(fields: [sectionId], references: [id])

authorId Int

author User @relation(fields: [authorId], references: [id])

reviews Review[]

}

model Review {

id Int @id @default(autoincrement())

comment String

status SubmissionStatus

submissionId Int

submission Submission @relation(fields: [submissionId], references: [id])

reviewerId Int

reviewer User @relation(fields: [reviewerId], references: [id])

}

model Certificate {

id Int @id @default(autoincrement())

userId Int

user User @relation(fields: [userId], references: [id])

sectionId Int

section Section @relation(fields: [sectionId], references: [id])

position Position

}

enum Role {

AUTHOR

MODERATOR

ADMIN

}

enum EventStatus {

UPCOMING

ONGOING

FINISHED

}

enum SubmissionStatus {

PENDING

ACCEPTED

REJECTED

}

enum Position {

FIRST

SECOND

THIRD

}

**3. Implement Features**

1. **Authentication and Authorization**
   * Use JWT for token-based authentication.
   * Implement middleware to check user roles.
2. **File Uploads**
   * Use a library like multer for handling file uploads in Express.
   * Store files securely and link them in the database.
3. **Review System**
   * Create endpoints for moderators to fetch submissions, review them, and leave comments.
   * Update submission status based on reviews.
4. **Winner Selection and Certificate Generation**
   * Create endpoints for selecting winners.
   * Use a library like pdf-lib or an API for generating certificates.
   * Store certificates and provide download links for users.

**Example Implementation Tasks**

**Frontend (React.js)**

1. **User Authentication Pages**
   * Register, Login, and Profile pages.
   * Implement form validation and API integration.
2. **Event Management Pages**
   * Create, update, and delete events.
   * Display a list of events with status filters.
3. **Section and Submission Pages**
   * Display sections within an event.
   * Allow authors to submit research to sections.
4. **Review Pages**
   * Display submissions for moderators.
   * Allow moderators to accept/reject submissions with comments.
5. **Winner Announcement and Certificate Pages**
   * Display winners in each section.
   * Provide download links for certificates.

**Backend (Node.js/Express)**

1. **User Management Routes**
   * Implement routes for registration, login, and profile updates.
   * Secure routes with JWT and role-based middleware.
2. **Event and Section Routes**
   * Implement CRUD operations for events and sections.
   * Ensure only admins can manage events and sections.
3. **Submission and Review Routes**
   * Implement routes for submitting research.
   * Implement routes for reviewing submissions and updating statuses.
4. **Certificate Generation Routes**
   * Implement routes for selecting winners.
   * Generate and store certificates, and provide download links.

A screenshot of a computer screen

Description automatically generated