Project Report: Portfolio Management Platform

Project Overview

The Portfolio Management Platform is a web-based application designed to help users manage their portfolios, which can include various types of media (e.g., images) and information related to their work, achievements, and projects. The platform features user authentication, the ability to create and manage portfolio items, and integration with third-party APIs for retrieving sports events and world news. The application allows users to register, log in, and access their portfolios, while administrators and editors have distinct roles that govern what actions they can take within the platform.

Key Features and Functionalities

1. **User Authentication**:
    - **Registration**: Users can create an account by providing their email, password, first name, last name, age, and gender. Passwords are securely hashed using bcrypt before storing them in the database.
    - **Login**: Registered users can log in using their email and password. Upon successful login, they are redirected to their portfolio page.
    - **Role-based Access Control**: The application defines two user roles: **Admin** and **Editor**. Admin users have full control over the platform, including the ability to manage users, while editors can only manage their own portfolio items.
2. **Portfolio Management**:
    - **Create Portfolio Items**: Users (editors and admins) can create portfolio items by providing a title, description, and images (up to three images per item).
    - **View Portfolio**: All portfolio items are displayed in a carousel format. Users can view, edit, and delete items if they have appropriate permissions.
    - **Delete Portfolio Items**: Users with appropriate privileges (e.g., admins) can delete portfolio items from the platform.
3. **Email Notifications**:
    - Upon successful registration, users receive a welcome email via Nodemailer. The email contains a personalized message thanking the user for registering and welcoming them to the platform.
4. **Third-party API Integration**:
    - **Sports Data**: The platform integrates with the SportScore API to display upcoming sports events.
    - **News Data**: The platform integrates with News API to display the latest world news. These data sources are accessible via dedicated routes and are rendered dynamically on the platform.
5. **Responsive Design**:

- The platform has been designed to be responsive, ensuring a seamless user experience across both desktop and mobile devices.

6. **Session Management**:
    - The platform uses **express-session** for session management, allowing users to stay logged in during their session and ensuring that session data is securely encrypted.

Technologies Used

1. **Node.js**:
    - The backend server is built using **Node.js**, a runtime environment that allows JavaScript to be run on the server-side.
2. **Express.js**:
    - The web framework used for handling HTTP requests, routing, and middleware in the application.
3. **MongoDB**:
    - The platform uses **MongoDB**, a NoSQL database, to store user data, portfolio items, and session information. The database is accessed using **Mongoose**.
4. **Passport.js**:
    - **Passport.js** is used to manage user authentication. It handles login sessions using a local strategy that verifies the user's credentials.
5. **Bcrypt.js**:
    - **Bcrypt** is used to hash user passwords before storing them in the database, ensuring that passwords are securely stored.
6. **Multer**:
    - **Multer** is used for handling image uploads. It stores the uploaded images in the server's file system and saves the file paths to the database.
7. **Nodemailer**:
    - **Nodemailer** is used to send email notifications to users after they register on the platform.
8. **Axios**:
    - **Axios** is used for making HTTP requests to third-party APIs (such as the SportScore API and News API) to fetch sports events and world news.
9. **EJS (Embedded JavaScript)**:
    - **EJS** is the templating engine used to render dynamic content on the server side. It allows JavaScript code to be embedded in HTML views, enabling the display of personalized user data and dynamically fetched content.

Database Design

1. **User Model**:
    - username: String (required, unique) - User’s email address (used for login).
    - password: String (required) - Hashed password for user authentication.
    - firstName: String - User's first name.
    - lastName: String - User's last name.
    - age: Number - User's age.
    - gender: String - User's gender.
    - role: String (enum: 'admin', 'editor', default: 'editor') - User role (admin or editor).
2. **Portfolio Item Model**:
    - title: String (required) - Title of the portfolio item.
    - description: String (required) - Description of the portfolio item.
    - image1: String (required) - Path to the first image.
    - image2: String (required) - Path to the second image.
    - image3: String (required) - Path to the third image.
    - createdAt: Date (default: current date) - Date the portfolio item was created.

API Integration

1. **Sports Data**:
    - The application uses the **SportScore API** to fetch upcoming sports events. The data is displayed on the /api/sports route, where users can view a list of upcoming events, including event names and dates.
2. **News Data**:
    - The platform integrates with the **News API** to fetch the latest world news. The articles are displayed on the /api/news route, showing headlines and links to the full articles.

Challenges and Solutions

1. **Handling Image Uploads**:
    - One of the key challenges was configuring **Multer** to handle multiple image uploads per portfolio item. The solution was to use a storage engine that saves the images to the local server's file system and stores the image paths in the database.
2. **Email Notifications**:
    - Ensuring that email notifications were properly sent to users after registration required setting up **Nodemailer** with Gmail's SMTP service. The email contains a personalized welcome message and a link to the login page.

3. **API Integration**:
    - Integrating external APIs such as **SportScore** and **News API** required handling asynchronous requests and rendering the data dynamically. This was achieved using **Axios** to make GET requests and pass the data to the EJS templates for rendering.
4. **Session Management**:
    - Managing user sessions and ensuring that they persist across requests while keeping user data secure was another challenge. The application uses **express-session** to handle sessions, with encryption ensured through a secret session key.

Conclusion

The Portfolio Management Platform successfully provides a robust solution for managing portfolios, viewing and editing portfolio items, and integrating with third-party APIs for additional features like sports events and news. It utilizes modern web technologies, including **Node.js**, **Express.js**, **MongoDB**, and **Passport.js**, to provide a secure, responsive, and scalable application. With a user-friendly interface and role-based access control, the platform offers a great experience for both regular users and administrators. Future improvements could include additional user features, cloud storage for images, and enhanced user roles with granular permissions.