Task Description:

Develop a simple Student Management System using the MERN stack. The system should allow registration of students, courses, and teachers. Students can enroll for courses, and each course should have one assigned teacher. All forms should have proper validations. Additionally, implement optional features such as linking a payment gateway to course enrollment, profile picture upload and allowing students to upload their homework for enrolled courses.

Requirements:

1. **Registration:**
    - Students, courses, and teachers should be able to register with basic information such as name, email, and password.
    - Students and teachers should be able to upload a profile picture during registration.
2. **Course Enrollment:**
    - Students should be able to enroll for available courses.
    - Each course should have one assigned teacher.
3. **Optional: Payment Gateway Integration:**
    - Integrate a payment gateway for course enrollment.
    - Upon successful payment, allow students to enroll for the course.
4. **Optional: Homework Upload:**
    - Implement a feature for students to upload their homework for enrolled courses.
    - The uploaded homework should be associated with the respective course and student.

Optional Questions (Additional Marks):

1. **Payment Gateway Integration:**
    - Discuss the choice of payment gateway and justify your selection.
    - Discuss the flow of payment process.
    - Explain the security measures implemented to ensure safe transactions.
2. **Homework Upload:**
    - Discuss the file upload mechanism used for homework submissions.
    - How would you ensure the security and integrity of uploaded files?
    - Implement a backup mechanism to save homework outside the server (Eg. Upload to google drive)
3. **Profile Picture Upload:**
    - Describe the approach used for storing and retrieving profile pictures.
    - How would you optimize the performance of profile picture retrieval?

Backend Components:

1. **Multer for File Uploads:**
    - Use Multer middleware in Express.js for handling file uploads.
2. **Mongoose for Database:**
    - Utilize Mongoose ODM library to model application data and interact with MongoDB.
3. **Node.js and Express for Server:**
    - Develop the server-side application using Node.js and Express.js.
4. **JWT Tokens and Bcrypt:**
    - Implement JWT tokens for authentication and authorization.
    - Use Bcrypt for secure password hashing.

Frontend Components:

1. **React.js for Frontend:**
    - Develop the user interface using React.js library.
2. **Material-UI (MUI) for UI Components:**
    - Utilize Material-UI components for designing the user interface.
3. **Redux for State Management:**
    - Manage application state using Redux library.
4. **React Hook Form for Form Handling and Validations**:
    - Utilize React Hook Form library for managing form state, handling form submissions, and performing validations.
    - React Hook Form offers a simple and efficient way to manage form data and validations using React hooks.

Integration:

- **Multer with Express.js**: Integrate Multer middleware into Express.js for handling file uploads (You can save uploaded files in the server with proper directories).
- **Mongoose with Node.js**: Use Mongoose in Node.js applications for interaction with MongoDB.
- **JWT Tokens and Bcrypt with Authentication**: Implement JWT tokens and Bcrypt for secure authentication and authorization.
- **React.js with Material-UI and Redux**: Develop the frontend using React.js with Material-UI components and manage application state using Redux.
- **React Hook Form with Form Management**: Integrate React Hook Form alongside other frontend libraries for managing forms in React.js.

Evaluation Criteria:

1. **Functionality:** Ensure all core features are implemented correctly.
2. **Code Quality:** Maintain clean and organized code following best practices.
3. **Optional Features Implementation:** Evaluate the integration of payment gateway and homework upload feature.
4. **Optional Questions:** Assess the understanding and reasoning behind the optional features.

Note:

- Provide clear instructions on how to run the project locally.
- Include relevant comments and documentation for easy understanding.
- Consider scalability and performance aspects in your design and implementation.
- Feel free to use packages outside the list mentioned above.

END