**Project Plan: Attendance Management System**

**Frontend (React.js)**

1. **User Panel**
   * **Registration Page**:
     + Register new users.
     + Form validation.
     + Integration with backend API to save user data.
   * **Login Page**:
     + Authenticate users.
     + Secure login with JWT tokens.
   * **Dashboard**:
     + Access to attendance records, leave requests.
     + Overview of user’s attendance summary.
   * **Mark Attendance**:
     + Allow students to mark daily attendance.
     + Ensure single attendance entry per day.
   * **Mark Leave**:
     + Submit leave requests to the admin.
     + Form validation and submission to backend.
   * **View Attendance**:
     + Display all attendance records.
     + Pagination and filtering options.
   * **Edit Profile**:
     + Update user profile and profile picture.
     + Validate and save changes to backend.
2. **Admin Panel**
   * **Login Page**:
     + Authenticate admin users.
     + Secure login with JWT tokens.
   * **Dashboard**:
     + View all student records.
     + Summary of attendance statistics.
   * **Manage Attendance**:
     + CRUD operations on attendance records.
     + View and edit student attendance.
   * **Reports**:
     + Generate user-specific and system-wide attendance reports.
     + Export options (e.g., PDF, CSV).
   * **Leave Approval**:
     + Manage and approve leave requests.
     + Update leave request status.
   * **Grading System**:
     + Set and manage grading criteria based on attendance.
     + Dynamic grading configuration.

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

1. **API Endpoints**
   * **Authentication**:
     + Register and login users.
     + Secure endpoints with JWT.
   * **User Routes**:
     + Manage user profiles.
     + Handle attendance marking and retrieval.
     + Process leave requests.
   * **Admin Routes**:
     + Manage student records.
     + Handle attendance CRUD operations.
     + Generate reports.
     + Approve leave requests.
     + Define and update grading criteria.

**Database (MongoDB)**

1. **Collections**
   * **Users**:
     + Store user information (username, password, profile details).
     + Attendance data linked to users.
   * **Attendance**:
     + Record daily attendance.
     + User ID, date, and attendance status.
   * **LeaveRequests**:
     + Track leave requests.
     + User ID, leave details, and approval status.
   * **Grades**:
     + Define grading criteria based on attendance.
     + Store grading rules and user grades.

**Implementation Details**

1. **User Registration and Login**:
   * Secure user authentication with JWT.
   * Password hashing for secure storage.
2. **Attendance Marking**:
   * Prevent multiple daily attendances.
   * Time-based checks to ensure single entry.
3. **Leave Requests**:
   * Separate collection for leave requests.
   * Track status (pending, approved, rejected).
4. **Admin Management**:
   * Role-based access control for admin functions.
   * Secure admin routes and operations.
5. **Reporting**:
   * Generate attendance reports using MongoDB aggregation queries.
   * Provide summary and detailed views.
6. **Frontend**:
   * Implement React Router for smooth navigation.
   * State management for authentication state.
   * Use context or state management libraries (e.g., Redux) for global state handling.

**Timeline**

**Day 1-2: Setup and Initial Development**

* Setup project structure for frontend (React.js) and backend (Node.js, Express.js).
* Configure MongoDB and create initial collections.
* Develop user registration and login pages (frontend and backend integration).
* Basic dashboard structure for both user and admin panels.

**Day 3-4: Core Functionality**

* Implement attendance marking and leave requests (frontend and backend).
* Create admin dashboard with view and manage student records.
* Develop attendance CRUD operations for admin panel.

**Day 5: Advanced Features**

* Implement reporting functionality with aggregation queries.
* Develop grading system based on attendance.
* Leave approval management in the admin panel.

**Day 6: Final Touches**

* Complete user profile editing and picture upload.
* Ensure navigation and state management (React Router, state management library).
* Thorough testing of all functionalities.

**Day 7: Testing and Deployment**

* Final testing and bug fixing.
* Deployment setup (e.g., hosting frontend on Netlify, backend on Heroku or similar).
* Documentation and user guide preparation.