**Project Plan**

**Technology Stack**

* **Front-End:** React.js with React Router for SPA navigation, Axios for API requests, possibly Material UI for styling.
* **Back-End:** Node.js with Express.js for RESTful API endpoints.
* **Database:** MongoDB (Atlas or local), Mongoose for schema definitions and queries.
* **Authentication:** JWT-based authentication, bcrypt for password hashing.
* **File Handling:** multer (Node.js middleware) for handling file uploads.

**Development Steps**

**Step 1: Setup & Boilerplate**

* Initialize GitHub repository and add project plan and technical specifications
* Set up basic project structure:
  + /backend (Node/Express)
  + /frontend (React)
* In the frontend: create a simple React app with a “Hello World” page.

**Step 2: User & Authentication**

* Define User model in Mongoose.
* Implement POST /api/auth/login and basic JWT authentication middleware.
* Seed database with a professor user and a student user.
* On the frontend, create a login page and store JWT in localStorage after login.

**Step 3: Sessions Management**

* Define Session model in Mongoose.
* Implement POST /api/sessions, GET /api/sessions, PUT /api/sessions/:id, DELETE /api/sessions/:id.
* Add logic to prevent overlapping sessions for the same professor.
* On the frontend, if the user is a professor, show a Sessions Management page:
  + List existing sessions.
  + Form to create a new session.

**Step 4: Requests Management**

* Define Request model.
* Implement POST /api/requests for students to create requests.
  + Validate session is active and no previous approval exists.
* Implement GET /api/requests to list:
  + For a student: their requests.
  + For a professor: requests made to their sessions.
* Frontend:
  + Student Dashboard: display active sessions and a button to send request.
  + Professor Dashboard: list incoming requests.

**Step 5: Approvals & Rejections**

* Implement PUT /api/requests/:id/approve and PUT /api/requests/:id/reject.
  + Check capacities.
  + Check that request is pending.
* Frontend:
  + Professor can approve or reject from the requests list.
  + Show messages upon success or error.

**Step 6: File Uploads**

* Integrate multer for file uploads.
* Implement:
  + PUT /api/requests/:id/upload-student-file
  + PUT /api/requests/:id/upload-professor-file
  + PUT /api/requests/:id/reject-student-file (to request new upload)
* Frontend:
  + Student: If request approved, show upload form.
  + Professor: If student file is not acceptable, click “Request re-upload.”

**Step 7: Validation, Error Handling, and UI Polish**

* Add input validations in back-end (e.g., check empty fields, date ranges).
* Improve error handling: return standardized error responses.
* Frontend: show user-friendly error and success messages.
* Add loading states and improve UI/UX.

**Step 8: Final Testing & Deployment**

* Test all flows: login, session creation, request submission, approval/rejection, file upload.
* Prepare production build of React frontend.
* Deploy back-end to a service (e.g. Render, Heroku, or AWS) and frontend to Netlify/Vercel.