# Personal Journal App (MERN Stack)

## 🧠 Objective

Build a basic Personal Journal App using the MERN stack (MongoDB, Express, React, Node.js).

This task will help us evaluate your understanding of:

* Authentication & session management
* Full-stack development practices
* Backend logic
* UI design & API integration
* Problem-solving in real-world scenarios

## ✅ Functional Requirements

1. User Authentication

* Implement register and login functionality
* Use JWT-based authentication
* Tokens should be stored securely (e.g., in localStorage or cookies)

1. Session Management (Limit to 2 Devices)

* A user can be logged in from a maximum of 2 devices at any time
* If the user logs in from a third device, the oldest active session must be invalidated automatically
* Use the provided backend logic (or your own if preferred) for managing sessions:
* Store session metadata (device info, IP, user-agent)
* Track accessToken, createdAt, lastUsedAt, and isActive
* Implement logout (current session) and optionally logout all devices

1. Journal Entries

* Users should be able to:
* Create new journal entries (title, content, date)
* Edit and delete existing entries
* View a list of their journal entries (latest first)

1. UI

* Simple React-based frontend
* Use any UI framework (Bootstrap, Tailwind, MUI) or write plain CSS
* Must include:
* Login/Register forms
* Journal dashboard
* Logout button
* (Optional) View current active sessions

## 🧪 Technical Requirements

* MongoDB: Store users, sessions, journal entries
* Express.js: REST API (auth, journal, session management)
* React.js: Frontend (functional components preferred)
* Node.js: Backend server
* Use dotenv for environment configs
* Bonus: Add device detection using user-agent string

## 🔐 Session Enforcement Guidelines

* On login, generate a JWT and save it along with device info in a Session collection
* Before saving a new session:
  + Check how many active sessions already exist for that user
  + If 2 already exist, delete/invalidate the oldest session
* On protected routes (like fetching journal entries), validate token and ensure it matches an active session

## 📦 Deliverables

* GitHub (or similar) repo with:
* README.md explaining:
  + Setup instructions
  + How the session-limiting works
  + Any assumptions you made
* Clear project structure for backend and frontend
* Well-formatted, clean code (modular, readable)
* Working implementation of login/session control and journal functionality