# Design Document: Patient Document Upload Portal

## 1. Tech Stack Choices

**Q1. What frontend framework did you use and why?**

React (with Vite and Tailwind CSS). React offers a modular, component-based structure and excellent community support. Vite ensures fast dev builds, while Tailwind allows rapid UI styling.

**Q2. What backend framework did you choose and why?**

Spring Boot. It provides robust support for RESTful APIs, easy integration with databases, and file storage support.

**Q3. What database did you choose and why?**

MySQL. It is widely used, reliable, and integrates well with Spring Boot via JPA/Hibernate.

**Q4. If you were to support 1,000 users, what changes would you consider?**

Move file storage to AWS S3, introduce user authentication, paginate file lists, use PostgreSQL, and deploy backend/frontend on scalable platforms like AWS/GCP. Add caching and rate-limiting for performance and security.

## 2. Architecture Overview

Frontend (React) → Backend (Spring Boot REST API) → Database (MySQL) + File System (uploads/)

Flow:

- User uploads PDF via frontend.
- Frontend sends it to backend using POST /documents/upload.
- Backend stores file in uploads/ folder, saves metadata in database.
- Files can be listed, downloaded, or deleted via API.

## 3. API Specification

| Endpoint | Method | Description |
|---|---|---|
| /documents/upload | POST | Upload a PDF |
| /documents | GET | List all documents |
| /documents/:id | GET | Download a file |
| /documents/:id | DELETE | Delete a file |

## 4. Data Flow Description

**File Upload Flow:**

1. User selects PDF file and clicks 'Upload'.

2. Frontend sends POST request to /documents/upload.

3. Backend stores file in uploads/ folder.

4. Metadata (filename, size, created_at) is stored in the MySQL database.

**File Download Flow:**

1. User clicks 'Download'.

2. Frontend sends GET request to /documents/{id}.

3. Backend reads file and responds with file blob.

4. Browser downloads file with original filename.

## 5. Assumptions

- File type restricted to PDF (validated client and server-side).

- Max file size assumed <10MB.

- No user authentication (single user assumed).

- Files stored locally in uploads/ folder.

- Metadata stored in MySQL using Spring JPA.

- Concurrency not handled for simultaneous upload/download (single-threaded assumption).