Tech Stack Choices

Q1. What frontend framework did you use and why? (React, Vue, etc.)

I used **React** because it provides a fast, component-based architecture ideal for building dynamic UIs like file upload dashboards.
It integrates easily with REST APIs from the backend.
Tailwind CSS helps build clean, responsive UI quickly with utility classes.

Q2. What backend framework did you choose and why? (Express, Flask, Django, etc.)

I used **Spring Boot** because it offers robust support for file uploads, REST APIs, and integrates smoothly with MySQL.
It is scalable, secure, and ideal for enterprise applications like healthcare portals.
Spring Data JPA simplifies storing and retrieving file metadata.

Q3. What database did you choose and why? (SQLite vs PostgreSQL vs others)

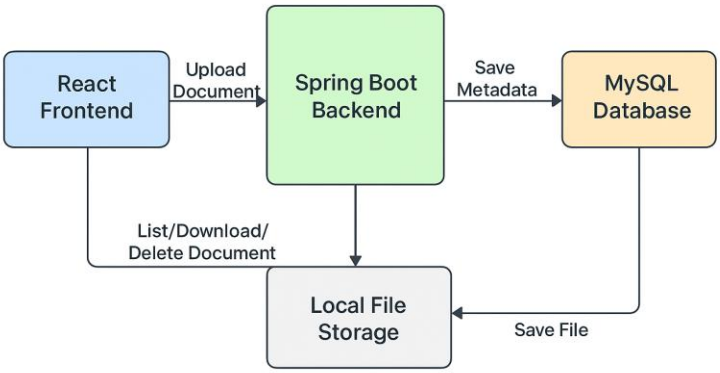
I chose **MySQL** because it is reliable, fast, and ideal for structured data like file metadata.
It supports multiple concurrent users, unlike SQLite which is file-based.
MySQL integrates very well with Spring Boot and Hibernate.

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

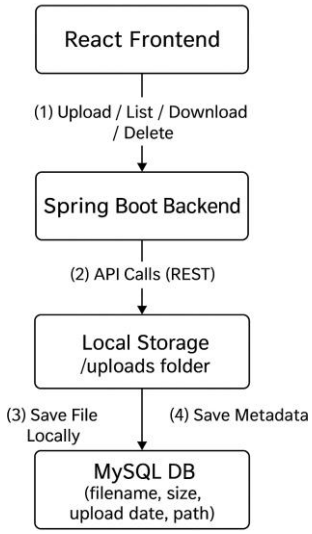
- I would move file storage to Amazon S3 for scalable and reliable storage.
- For faster delivery on the frontend, I would use CloudFront as a CDN.
- For the backend, I would use AWS Lambda with API Gateway to handle requests.
- For the database, I would use Amazon RDS for better performance and automatic backups.
- I would choose Lambda because it automatically scales and has built-in load balancing, making it easy to handle increased traffic without managing servers.

Architecture Overview

1. Draw or describe the flow between frontend, backend, database, and file storage.



2. You can use a simple diagram or bullet points.



API Specification

POST /api/files/upload
Uploads a PDF file, validates it, stores it locally, and saves metadata in MySQL.

GET /api/files
Returns a list of all uploaded PDF metadata stored in the database.

GET /api/files/{id}/download
Downloads the actual PDF file associated with the given file ID.

DELETE /api/files/{id}
Deletes the PDF from local storage and removes its metadata from MySQL.

Q5. Describe the step-by-step process of what happens when a file is uploaded and when it is downloaded.

When a File Is Uploaded

- User selects a PDF in the React frontend.
- Frontend sends the file to the backend using POST /api/files/upload.
- Backend validates that the file is a PDF.
- Backend saves the file locally in the uploads/ folder.
- Backend stores metadata in MySQL (file name, size, upload date, path).
- Backend sends success response back to the frontend.
- Frontend updates the UI, showing the newly uploaded document.

When a File Is Downloaded

- User clicks download on a specific file in React.
- Frontend calls GET /api/files/{id}/download.
- Backend finds metadata in MySQL using the file ID.
- Backend reads the actual file from local storage.
- Backend returns the file to the frontend as a PDF.
- Browser prompts the user to save or open the downloaded PDF.

Q6. What assumptions did you make while building this? (e.g., file size limits, authentication, concurrency)

- Users must be authenticated before uploading, listing, downloading, or deleting files.
- Each user can only access their own uploaded documents.
- The system stores files in a local uploads/ folder.
- Metadata (filename, size, upload date, path) is saved in MySQL.