Project Title: PDF & DOCX Document Manager

## 1. Project Overview

This project is a **Document Management System** built using **Spring Boot** for the backend, **React.js** for the frontend, and **Oracle Database 21c** for storing user and document data. Users can upload PDF/DOCX files, search for documents by content, download, and delete them. The system also uses JWT for authentication.

## 2. Technologies Used

* **Backend:** Java, Spring Boot, Spring Security, JWT
* **Frontend:** React.js, Axios, Bootstrap
* **Database:** Oracle 21c XE
* **Libraries:**
  + Apache PDFBox (for PDF text extraction)
  + Apache POI (for DOCX text extraction)
  + BCryptPasswordEncoder (password hashing)
* **Version Control:** Git & GitHub

## 3. Project Structure

### Backend (pdfapp)

pdfapp  
├── src  
│ ├── main  
│ │ ├── java/com/pdfapp/pdfapp  
│ │ │ ├── controller  
│ │ │ │ └── DocumentController.java (Handles file upload, download, search, delete)  
│ │ │ ├── model  
│ │ │ │ ├── Document.java (Document entity with file, text, uploadedAt)  
│ │ │ │ └── User.java (User entity with hashed password and email)  
│ │ │ ├── repository  
│ │ │ │ ├── DocumentRepository.java (DB operations for documents)  
│ │ │ │ └── UserRepository.java (DB operations for users)  
│ │ │ ├── security  
│ │ │ │ └── JwtFilter.java (JWT validation)  
│ │ │ ├── config  
│ │ │ │ └── SecurityConfig.java (Spring Security config)  
│ │ │ ├── dto  
│ │ │ │ └── DocumentDTO.java (Data Transfer Object for documents)  
│ │ │ └── PdfappApplication.java (Main Spring Boot app)  
│ │ └── resources  
│ │ └── application.properties (DB & server config)  
├── pom.xml (Maven dependencies)  
└── mvnw, mvnw.cmd (Maven wrappers)

### Frontend (pdf-frontend)

pdf-frontend  
├── src  
│ ├── components  
│ │ └── DocumentManager.js (React component to manage documents)  
│ ├── App.js (Main React app component)  
│ └── index.js (React entry point)  
├── package.json (Dependencies and scripts)  
└── public (Static assets)

## 4. Database Schema

### Users Table

| Column | Type | Purpose |
| --- | --- | --- |
| ID | NUMBER | Primary key |
| USERNAME | VARCHAR | Login username |
| PASSWORD\_HASH | VARCHAR | Hashed password (BCrypt) |
| EMAIL | VARCHAR | User email |
| CREATED\_AT | DATE | Registration timestamp |

### Documents Table

| Column | Type | Purpose |
| --- | --- | --- |
| ID | NUMBER | Primary key |
| OWNER\_ID | NUMBER | Foreign key to users |
| FILENAME | VARCHAR | Original file name |
| MIME\_TYPE | VARCHAR | File type (PDF/DOCX) |
| PDF\_BLOB | BLOB | Actual file bytes |
| EXTRACTED\_TEXT | CLOB | Extracted text for search |
| UPLOADED\_AT | DATE | Timestamp of upload |

## 5. Key Features

1. **User Registration & Login:**
   * Passwords are securely hashed using BCrypt.
   * JWT token generated for authenticated sessions.
2. **Upload Documents:**
   * Supports PDF and DOCX files.
   * Extracts text content for search.
3. **Search Documents:**
   * Search by filename or extracted text.
4. **Download & Delete:**
   * Download files via browser.
   * Delete documents by owner.
5. **Security:**
   * Spring Security + JWT.
   * Stateless session management.
   * CORS enabled for React frontend.

## 6. Backend Code Highlights

### DocumentController.java

@PostMapping("/upload")  
public ResponseEntity<DocumentDTO> upload(...) { ... }  
  
@GetMapping("/search")  
public List<DocumentDTO> searchDocuments(@RequestParam String keyword) { ... }  
  
@GetMapping("/{id}/download")  
public ResponseEntity<byte[]> download(@PathVariable Long id) { ... }

### SecurityConfig.java

http.csrf().disable()  
 .authorizeHttpRequests()  
 .requestMatchers("/api/users/login", "/api/users/register", "/api/documents/\*\*").permitAll()  
 .anyRequest().authenticated()  
 .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

### DocumentDTO.java

private Long id;  
private String filename;  
private String mimeType;  
private Long ownerId;  
private Date uploadedAt;

## 7. Frontend Code Highlights

### DocumentManager.js

* Handles file selection, upload, search, delete, and download.
* Uses Axios to communicate with backend endpoints.
* Displays uploaded documents with **filename** and **uploaded date**.
* Bootstrap/React styling added for UI.

## 8. Instructions to Run

### Backend

cd pdfapp  
mvn clean install  
mvn spring-boot:run

* Backend runs on http://localhost:8080

### Frontend

cd pdf-frontend  
npm install  
npm start

* Frontend runs on http://localhost:3000 (or alternate port if 3000 is busy)

## 9. GitHub & Deployment

1. **Initialize Git & Commit**

git init  
git add .  
git commit -m "Initial commit"

1. **Push to GitHub**

git remote add origin <your-repo-url>  
git branch -M main  
git push -u origin main

1. **Clone & Run**

git clone <your-repo-url>  
cd pdfapp  
mvn spring-boot:run  
cd ../pdf-frontend  
npm install  
npm start

## 10. Flow Diagram

User -> Frontend -> API Endpoints  
Upload -> Extract Text -> Save to DB  
Search -> Backend Query -> Return Results  
Download/Delete -> File Access / DB Operation

## 11. Screenshots

* Placeholder for login, registration, document list, search, upload, download

This document contains **all details about your project**, including backend, frontend, database schema, key features, and instructions to run the project locally or via GitHub.