# **File Upload with User Details in Spring Boot and React**

## **Overview**

This project demonstrates how to upload a file along with user details in a Spring Boot backend and React frontend. The uploaded file is stored in the src/main/resources/static/uploads/ folder, while the file metadata and user details are saved in MongoDB.

## **1. Spring Boot Backend Implementation**

### ****1.1 Dependencies (pom.xml)****

Ensure you have the following dependencies in your pom.xml:

<dependencies>

<!-- Spring Boot Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- MongoDB -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

<!-- Lombok for reducing boilerplate code -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<scope>provided</scope>

</dependency>

<!-- Multipart File Handling -->

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

</dependency>

</dependencies>

### ****1.2 Model Classes****

#### **User Model**

package com.example.handle\_multi\_media.model;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class User {

private String name;

private String email;

}

#### **FileDocument Model**

package com.example.handle\_multi\_media.model;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Document(collection = "files")

public class FileDocument {

@Id

private String id;

private String fileName;

private String fileType;

private String filePath;

private String userName;

private String userEmail;

}

### ****1.3 Repository****

package com.example.handle\_multi\_media.repository;

import com.example.handle\_multi\_media.model.FileDocument;

import org.springframework.data.mongodb.repository.MongoRepository;

public interface FileRepository extends MongoRepository<FileDocument, String> {

}

### ****1.4 Service Layer****

package com.example.handle\_multi\_media.service;

import org.springframework.stereotype.Service;

import org.springframework.web.multipart.MultipartFile;

import com.example.handle\_multi\_media.model.FileDocument;

import com.example.handle\_multi\_media.model.User;

import com.example.handle\_multi\_media.repository.FileRepository;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.StandardCopyOption;

import java.util.List;

import java.util.Optional;

import java.util.UUID;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Service

public class FileService {

private final FileRepository fileRepository;

private final String UPLOAD\_DIR = "uploads/";

private final String STATIC\_PATH = "src/main/resources/static/" + UPLOAD\_DIR;

private static final Logger logger = LoggerFactory.getLogger(FileService.class);

public FileService(FileRepository fileRepository) {

this.fileRepository = fileRepository;

}

public FileDocument saveFile(User user, MultipartFile file) throws Exception {

logger.info("Received upload request for User: {} - {}", user.getName(), user.getEmail());

// Create directory if not exists

Path uploadDirPath = Path.of(STATIC\_PATH);

if (!Files.exists(uploadDirPath)) {

Files.createDirectories(uploadDirPath);

logger.info("Created directory: {}", uploadDirPath);

}

// Generate unique file name

String fileName = UUID.randomUUID() + "\_" + file.getOriginalFilename();

Path filePath = Path.of(STATIC\_PATH + fileName);

// Copy file to the directory

Files.copy(file.getInputStream(), filePath, StandardCopyOption.REPLACE\_EXISTING);

logger.info("File saved successfully: Name={}, Type={}, Path={}", fileName, file.getContentType(), filePath);

// Store relative path in DB

FileDocument fileDocument = new FileDocument(null, fileName, file.getContentType(), UPLOAD\_DIR + fileName,

user.getName(), user.getEmail());

FileDocument savedFile = fileRepository.save(fileDocument);

logger.info("File document saved to DB: ID={}, User={}", savedFile.getId(), savedFile.getUserName());

return savedFile;

}

public List<FileDocument> getAllFiles() {

return fileRepository.findAll();

}

public Optional<FileDocument> getFileById(String id) {

return fileRepository.findById(id);

}

public boolean deleteFile(String id) {

if (fileRepository.existsById(id)) {

fileRepository.deleteById(id);

return true;

}

return false;

}

}

### ****1.5 Controller****

package com.example.handle\_multi\_media.controller;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.multipart.MultipartFile;

import com.example.handle\_multi\_media.model.FileDocument;

import com.example.handle\_multi\_media.model.User;

import com.example.handle\_multi\_media.service.FileService;

import org.springframework.http.ResponseEntity;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.http.HttpStatus;

import java.util.List;

@CrossOrigin(origins = "http://localhost:5173") // Allow React frontend

@RestController

@RequestMapping("/api/files")

public class FileController {

private final FileService fileService;

private static final Logger logger = LoggerFactory.getLogger(FileController.class);

public FileController(FileService fileService) {

this.fileService = fileService;

}

@PostMapping("/upload")

public ResponseEntity<FileDocument> uploadFile(

@RequestPart("user") User user,

@RequestParam("file") MultipartFile file) {

try {

logger.info("Received upload request for user: {}", user.getName());

FileDocument savedFile = fileService.saveFile(user, file);

return ResponseEntity.ok(savedFile);

} catch (Exception e) {

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).build();

}

}

@GetMapping

public List<FileDocument> getAllFiles() {

return fileService.getAllFiles();

}

@GetMapping("/{id}")

public ResponseEntity<FileDocument> getFile(@PathVariable String id) {

return fileService.getFileById(id)

.map(ResponseEntity::ok)

.orElseGet(() -> ResponseEntity.notFound().build());

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteFile(@PathVariable String id) {

return fileService.deleteFile(id) ? ResponseEntity.noContent().build() : ResponseEntity.notFound().build();

}

}

## **2. Testing with Postman**

### ****2.1 Upload File****

* **Method:** POST
* **URL:** http://localhost:8080/api/files/upload
* **Body:**
  + Select form-data
  + Key: user, Value: { "name": "John Doe", "email": "johndoe@example.com" } (as text/plain)
  + Key: file, Value: Select a file (as file)

### ****2.2 Get All Files****

* **Method:** GET
* **URL:** http://localhost:8080/api/files

### ****2.3 Get File by ID****

* **Method:** GET
* **URL:** http://localhost:8080/api/files/{id}

### ****2.4 Delete File****

* **Method:** DELETE
* **URL:** http://localhost:8080/api/files/{id}

# **React:**

import { useState, useEffect } from "react";

import axios from "axios";

const FileUpload = () => {

  const [user, setUser] = useState({ name: "", email: "" });

  const [file, setFile] = useState(null);

  const [files, setFiles] = useState([]);

  useEffect(() => {

    fetchFiles();

  }, []);

  const fetchFiles = async () => {

    try {

      const res = await axios.get("http://localhost:8080/api/files");

      setFiles(res.data);

    } catch (error) {

      console.error("Error fetching files:", error);

    }

  };

  const handleChange = (e) => {

    setUser({ ...user, [e.target.name]: e.target.value });

  };

  const handleFileChange = (e) => {

    setFile(e.target.files[0]);

  };

  const handleUpload = async (e) => {

    e.preventDefault();

    if (!user.name || !user.email || !file)

      return alert("All fields are required!");

    const formData = new FormData();

    formData.append("file", file);

    formData.append(

      "user",

      new Blob([JSON.stringify(user)], { type: "application/json" })

    );

    console.log(formData.get("file"));

    console.log(formData.get("user"));

    try {

      await axios.post("http://localhost:8080/api/files/upload", formData, {

        headers: { "Content-Type": "multipart/form-data" },

      });

      setUser({ name: "", email: "" });

      setFile(null);

      fetchFiles();

    } catch (error) {

      console.error("Upload error:", error);

    }

  };

  const handleDelete = async (id) => {

    try {

      await axios.delete(`http://localhost:8080/api/files/${id}`);

      fetchFiles();

    } catch (error) {

      console.error("Delete error:", error);

    }

  };

  return (

    <div className="file-upload-container">

      <h2>File Upload</h2>

      <form onSubmit={handleUpload}>

        <input

          type="text"

          name="name"

          value={user.name}

          onChange={handleChange}

          placeholder="Enter Name"

          required

        />

        <input

          type="email"

          name="email"

          value={user.email}

          onChange={handleChange}

          placeholder="Enter Email"

          required

        />

        <label className="file-drop-zone">

          Drag & Drop your file here or

          <label className="file-label">

            Click to Select

            <input

              type="file"

              className="file-input"

              onChange={handleFileChange}

              required

            />

          </label>

        </label>

        {file && <p className="selected-file">📁 Selected File: {file.name}</p>}

        <button type="submit" className="upload-btn">

          Upload

        </button>

      </form>

      <h3>Uploaded Files</h3>

      <ul className="file-list">

        {files.map((file) => (

          <li key={file.id} className="file-item">

            <span>

              {file.fileName} ({file.fileType})

            </span>

            <span className="file-actions">

              <a href={`http://localhost:8080/${file.filePath}`} download>

                Download

              </a>

              <button

                className="delete-btn"

                onClick={() => handleDelete(file.id)}

              >

                Delete

              </button>

            </span>

          </li>

        ))}

      </ul>

    </div>

  );

};

export default FileUpload;