

SHARE2TEACH Technical Documentation

© 2024 CodeCrafters. All Rights Reserved.



Septermber, 2014

CodeCrafters

Share2Teach is a web-based platform fostering a global learning community, aimed at enabling collaborative knowledge sharing. The platform offers four user roles: Admin/Moderator, Educator, and Open Access User, each with varying levels of access and functionality. This document provides a detailed explanation of the implementation, including REST API, file storage, database schemas, user roles, and system workflows.

**System Architecture**

The application is a Spring Boot-powered web app, interfacing with users via a web-based user interface. It provides secure login, document management and analytics functionalities.

Technology Stack

* Backend – Spring Boot, Java
* Database – MySQL (Hosted on Azure)
* Frontend – HTML, CSS, JavaScript (To be confirmed in the future)
* File Storage – Cloud (Azure)
* Security – JWT (JSON Web Tokens), BCrypt for password encryption

**System Components**

**User Roles and Actions**

| **User Role** | **Actions** |
| --- | --- |
| Admin/Moderator | Login, search documents, view documents, rate documents, use FAQ, contribute documents, moderate documents, use analytics |
| Educator | Login, search documents, view documents, rate documents, use FAQ, contribute, moderate |
| Open Access User | Search documents, view documents, rate documents, use FAQ |

**REST API Design**

The backend API is designed using REST principles, facilitating easy communication between the frontend and the server.

The system implements secure user authentication using JWT for session handling.

* Login:

POST /auth/login – Allows users to login with a username and password.

Input: { "username": "user",

"password": "pass" }

Output: { "token": "JWT\_TOKEN" }

* Register:

POST /auth/register – Allows new users to register.

The file handling system provides upload, view, search and moderation functions.

* File Upload:

POST /api/files/upload: Upload a new file.

Input: MultipartFile file, String uploadedBy

Output: File uploaded successfully

* File Approval:

PUT /api/files/approve/{fileId}: Approve a pending file.

**Implementation Details**

**File Upload Service**

Sample code in Java

@Service

public class FileStorageService {

private final Path fileStorageLocation =

Paths.get("uploads").toAbsolutePath().normalize();

private final FileRepository fileRepository;

@Autowired

public FileStorageService(FileRepository fileRepository) throws IOException {

this.fileRepository = fileRepository;

Files.createDirectories(this.fileStorageLocation);

}

public FileEntity storeFile(MultipartFile file, String uploadedBy) throws IOException {

FileEntity fileEntity = new FileEntity();

fileEntity.setFileName(file.getOriginalFilename());

fileEntity.setFileType(file.getContentType());

fileEntity.setData(file.getBytes());

fileEntity.setUploadDate(new Date());

fileEntity.setUploadedBy(uploadedBy);

fileEntity.setStatus("Pending");

return fileRepository.save(fileEntity);

}

}

**API Controller for File Upload**

@RestController

@RequestMapping("/api/files")

public class FileUploadController {

private final FileStorageService fileStorageService;

@Autowired

public FileUploadController(FileStorageService fileStorageService) {

this.fileStorageService = fileStorageService;

}

@PostMapping("/upload")

public ResponseEntity<String> uploadFile(@RequestParam("file") MultipartFile file, @RequestParam("uploadedBy") String uploadedBy) {

try {

FileEntity savedFile = fileStorageService.storeFile(file, uploadedBy);

return ResponseEntity.ok("File uploaded successfully: " + savedFile.getFileName());

} catch (IOException e) {

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body("Failed to upload file");

}

}

}

**Security Considerations**

* JWT Authentication – All sensitive endpoints (such as file uploads and user registration) are protected using JWT authentication.
* Encryption – Passwords are encrypted using BCrypt.
* File Upload Security – The system checks for file types and limits sizes to ensure malicious files cannot be uploaded.

**Database Schema:**