**Cloud-Based File Storage System**

**Project Overview:**

The Cloud-Based File Storage System is a web application designed to allow users to securely register, log in, and manage their files by uploading, downloading, and deleting them. Files are stored in Google Cloud Storage, and metadata is stored in a MySQL database. The user interface is developed using Java Swing, and all backend operations are handled using Spring Boot with JDBC.

**Tech Stack:**

* **Frontend (UI):** Java Swing
* **Backend:** Spring Boot
* **Database Connection:** JDBC (Java Database Connectivity)
* **Database:** MySQL
* **Cloud Storage:** Google Cloud Storage

**Modules:**

**1. User Authentication**

* User Registration and Login
* Password encryption
* Session handling or token-based authentication

**2. File Management**

* Upload files to Google Cloud Storage
* List all uploaded files by the user
* Download files
* Delete files

**3. Access Control**

* Each user can only manage their own files
* File visibility can be extended to public/private (optional)

**Database Schema:**

**Table: users**

* id (INT, PK, AUTO\_INCREMENT)
* username (VARCHAR)
* email (VARCHAR)
* password (VARCHAR)

**Table: files**

* id (INT, PK, AUTO\_INCREMENT)
* user\_id (INT, FK)
* filename (VARCHAR)
* gcs\_path (TEXT)
* uploaded\_at (TIMESTAMP)

**Backend API Endpoints (Spring Boot):**

**User APIs:**

* POST /auth/register - Register new users
* POST /auth/login - Login and authenticate users

**File APIs:**

* POST /files/upload - Upload a file
* GET /files/list - List uploaded files
* GET /files/download/{id} - Download file
* DELETE /files/{id} - Delete file

**Java Swing Frontend:**

* Login Form
* Registration Form
* Dashboard with:
  + Upload File (browse and upload)
  + Display list of user files (with download/delete)
* API Communication using HttpURLConnection or HttpClient

**JDBC Usage:**

* Use of JdbcTemplate in Spring Boot
* Manual SQL queries for CRUD operations

**Sample JDBC Code:**

String sql = "INSERT INTO users (username, password, email) VALUES (?, ?, ?)";

jdbcTemplate.update(sql, username, password, email);

**Google Cloud Storage Integration:**

* Create a GCS Bucket
* Create a service account and download JSON credentials
* Configure in Spring Boot using environment variable GOOGLE\_APPLICATION\_CREDENTIALS
* Use GCP Storage Client Library to upload/download files

**Sample Code to Upload File:**

Storage storage = StorageOptions.getDefaultInstance().getService();

BlobId blobId = BlobId.of(bucketName, fileName);

BlobInfo blobInfo = BlobInfo.newBuilder(blobId).build();

storage.create(blobInfo, fileBytes);

**Folder Structure:**

project-root/

├── backend/

│ ├── controller/

│ ├── service/

│ ├── model/

│ ├── repository/ ← JDBC logic

│ └── config/

├── frontend/

│ └── SwingApp.java ← Java Swing GUI

├── resources/

│ └── application.properties

├── credentials/

│ └── gcp-key.json ← Service account key

**Future Enhancements:**

* Implement JWT token-based authentication
* Enable public file sharing with access links
* File versioning support
* Drag and drop in Swing UI

**Conclusion:**

This project integrates Java desktop GUI (Swing) with modern backend architecture (Spring Boot and cloud storage) using JDBC for database operations, making it a powerful full-stack application demonstrating real-world cloud and data management capabilities.