Here's a roadmap for creating a cloud-based storage system using Java. This guide includes the technologies you can use, features of the project, and how to divide the work into modules.

Project Overview

**Project Name:** Cloud Storage System  
**Description:** A cloud-based storage system allows users to store, manage, and retrieve files securely over the internet.  
**Technologies:**

* **Backend:** Java (using Spring Boot or similar frameworks)
* **Cloud Platform:** Google Cloud Platform (GCP) or Amazon Web Services (AWS)
* **Database:** Relational databases like MySQL or PostgreSQL for user management, and NoSQL databases like MongoDB for metadata storage
* **Frontend:** Optional - HTML/CSS/JavaScript for a web interface or (java swing , javafx)

Features

1. **User Management**
   * User registration and login functionality
   * Password encryption for security
2. **File Management**
   * Upload files to cloud storage
   * Download files from cloud storage
   * Delete files from cloud storage
   * List all uploaded files
3. **Security**
   * Authentication and authorization for access control
   * Data encryption for stored files
4. **Cloud Integration**
   * Use Google Cloud Storage or AWS S3 for storing files
5. **Optional Features**
   * File sharing with other users
   * Versioning of files
   * Search functionality for files

Roadmap

**Step 1: Project Setup**

1. **Choose a Cloud Platform:**
   * Decide whether to use Google Cloud Platform (GCP) or Amazon Web Services (AWS).
   * Set up a project in the chosen platform.
2. **Set Up the Development Environment:**
   * Install Java Development Kit (JDK).
   * Choose an IDE like Eclipse or IntelliJ IDEA.
   * Install necessary plugins or libraries for cloud integration.
3. **Create a New Java Project:**
   * Use Spring Boot for a simpler setup.
   * Add dependencies for cloud storage integration (e.g., Google Cloud Storage Client for Java).

**Step 2: User Management Module**

1. **Design Database Schema:**
   * Use a relational database like MySQL for user data.
   * Define tables for users and their details.
2. **Implement User Registration and Login:**
   * Use Spring Security for authentication.
   * Implement password hashing for security.
3. **Test User Management Functionality:**
   * Ensure registration and login work correctly.

**Step 3: Cloud Storage Integration Module**

1. **Set Up Cloud Storage:**
   * Enable Google Cloud Storage or AWS S3 in your project.
   * Create a bucket for storing files.
2. **Implement File Upload and Download:**
   * Use the Google Cloud Storage Client for Java or AWS SDK for Java.
   * Handle file uploads and downloads securely.
3. **Test Cloud Storage Functionality:**
   * Ensure files can be uploaded and downloaded correctly.

**Step 4: Security Module**

1. **Implement Authentication and Authorization:**
   * Use Spring Security to restrict access to files based on user roles.
2. **Encrypt Stored Files:**
   * Use encryption libraries like Java Cryptography Architecture (JCA).
3. **Test Security Features:**
   * Ensure only authorized users can access files.

**Step 5: Optional Features**

1. **File Sharing:**
   * Implement functionality to share files between users.
2. **Versioning:**
   * Store different versions of files.
3. **Search Functionality:**
   * Implement a search bar to find files by name or metadata.

**Step 6: Frontend**

1. **Create a Web Interface:**
   * Use HTML/CSS/JavaScript to create a user-friendly interface.(java swing or javafx)
   * Integrate with the backend using REST APIs.
2. **Test the Web Interface:**
   * Ensure all features work as expected through the web interface.

**Step 7: Deployment**

1. **Deploy the Application:**
   * Use platforms like Heroku or Google App Engine for deployment.
2. **Test the Deployed Application:**
   * Ensure all features work correctly in the deployed environment.