PROJECT REPORT DOCUMENT

"Cloud Based Storage System"

UE20CS352 – OOAD-J PROJECT REPORT

Submitted by: Team 2

Abhay Hiremath	**PES2UG20CS008**
Adarsh Liju Abraham	**PES2UG20CS017**
Amisha Mathew	**PES2UG20CS038**
Apoorva Naronikar	**PES2UG20CS062**

Under the guidance of

Prof. Sudeepa Roy Dey

Associate Professor, Dept. of CSE

PES University

January - May 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PROBLEM STATEMENT

- Cloud based storage is a way of storing digital data, which can be accessed from any internet connected device.
- It is secure, as it is stored in multiple locations, making it hard to access.
- It is great for businesses, as it can be scaled up as the business grows and is more cost-effective than other types of data storage.

We aim to build a Cloud-based Storage System which ideally includes the features mentioned in the upcoming section below.

We would utilize Java concepts as well as HTML and CSS for front-end and MySQL for back-end, in order to develop and work on this project.

USE CASES

1. Cloud based storage is useful for both large and small businesses, as it offers flexibility, scalability, and security.
2. It can be used to back up sensitive information, store customer data, or facilitate collaboration between multiple users.
3. It is also an ideal solution for those who need to access their data from multiple devices, as it can be accessed from anywhere with an internet connection.
4. Cloud based storage can be used for backups, making sure data is secure and readily available in case of a disaster.
5. Create, Upload and Delete files. User Isolation is done here as it provides more security.

TASKS

1. Upload a file
2. Update a file
3. Backup a file
4. Delete a file
5. Download a file
6. Search a file
7. Design the database + Basic UI implementation

SYNOPSIS

Purpose: To provide a cloud-based storage system that allows users to store and access their files securely and efficiently from anywhere on a web browser

Scope: The cloud-based storage system will allow users to upload, download, and manage their files using a web browser. The system will also provide features such as file sharing, collaboration, version control, and backup.

Users: The primary users of the cloud-based storage system are individuals and organizations who need a secure and reliable way to store and access their files.

FUNCTIONAL REQUIREMENTS

1. User registration and authentication: Users must be able to create an account and log in to the system using a username and password.
2. File upload and download: Users must be able to upload and download files to and from the system using a web browser.
3. File management (Backup): Users must be able to organize their files into folders, rename, delete, and move files between folders.
4. File sharing: Users must be able to share files with other users by providing access to specific files or folders.
5. Backup and Recovery: The system must automatically backup all files stored in the cloud-based storage system to prevent data loss.

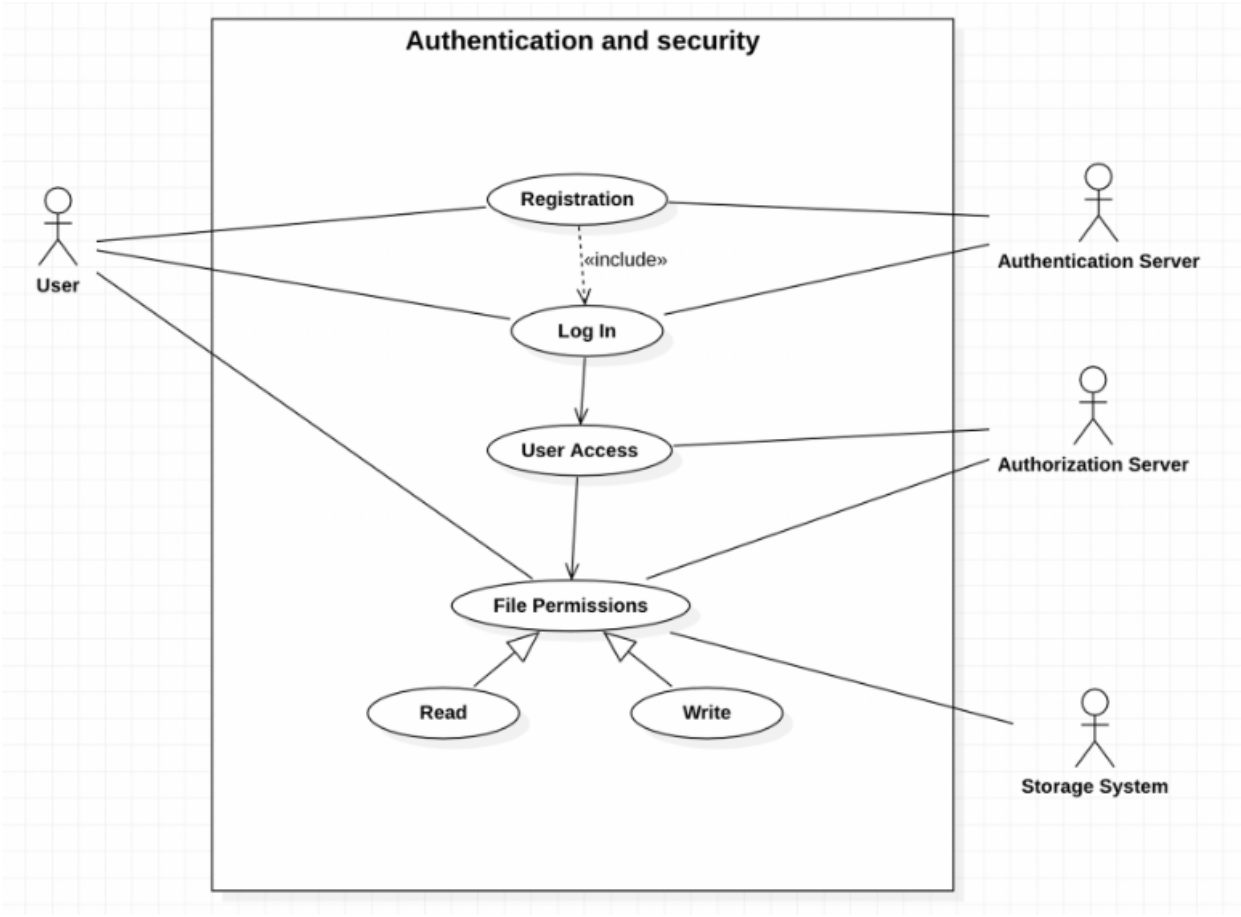
NON FUNCTIONAL REQUIREMENTS

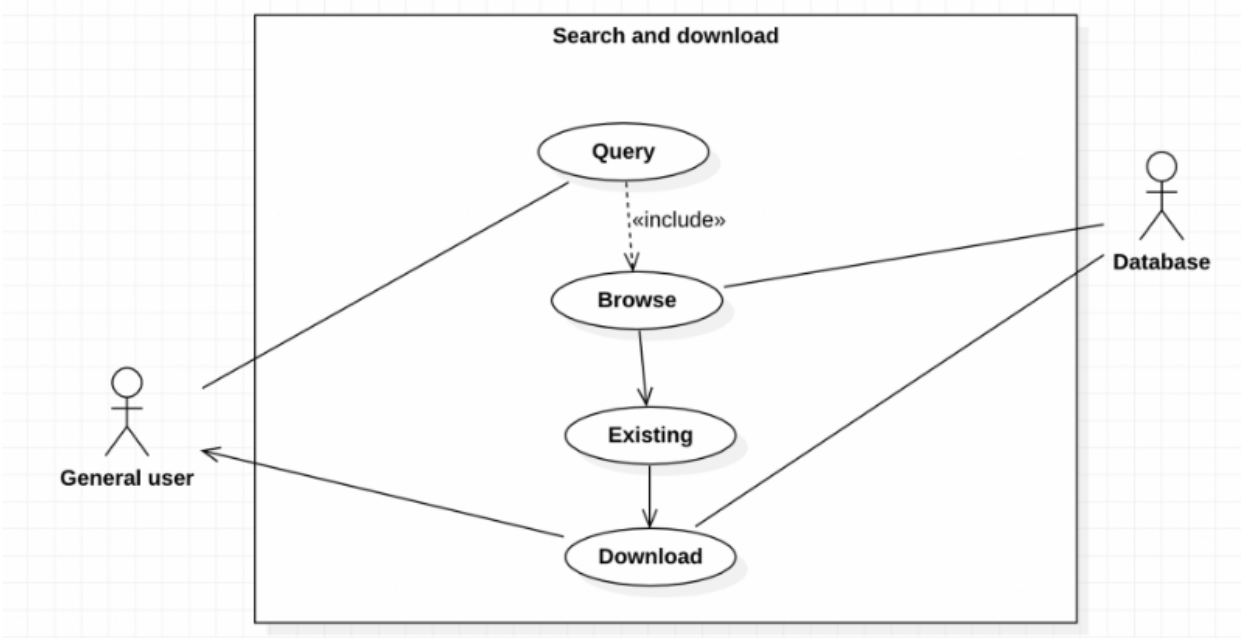
1. Security: The system must provide strong encryption for all data in transit and at rest, and use secure authentication and authorization mechanisms to prevent unauthorized access.
2. Scalability: The system must be able to handle a large number of users and a large volume of data.
3. Performance: The system must provide fast and responsive access to files, even when accessed by multiple users simultaneously.
4. Reliability: The system must have high availability and be able to recover quickly from any failures.
5. Usability: The system must be easy to use and provide a user-friendly interface that is intuitive and accessible to all users.

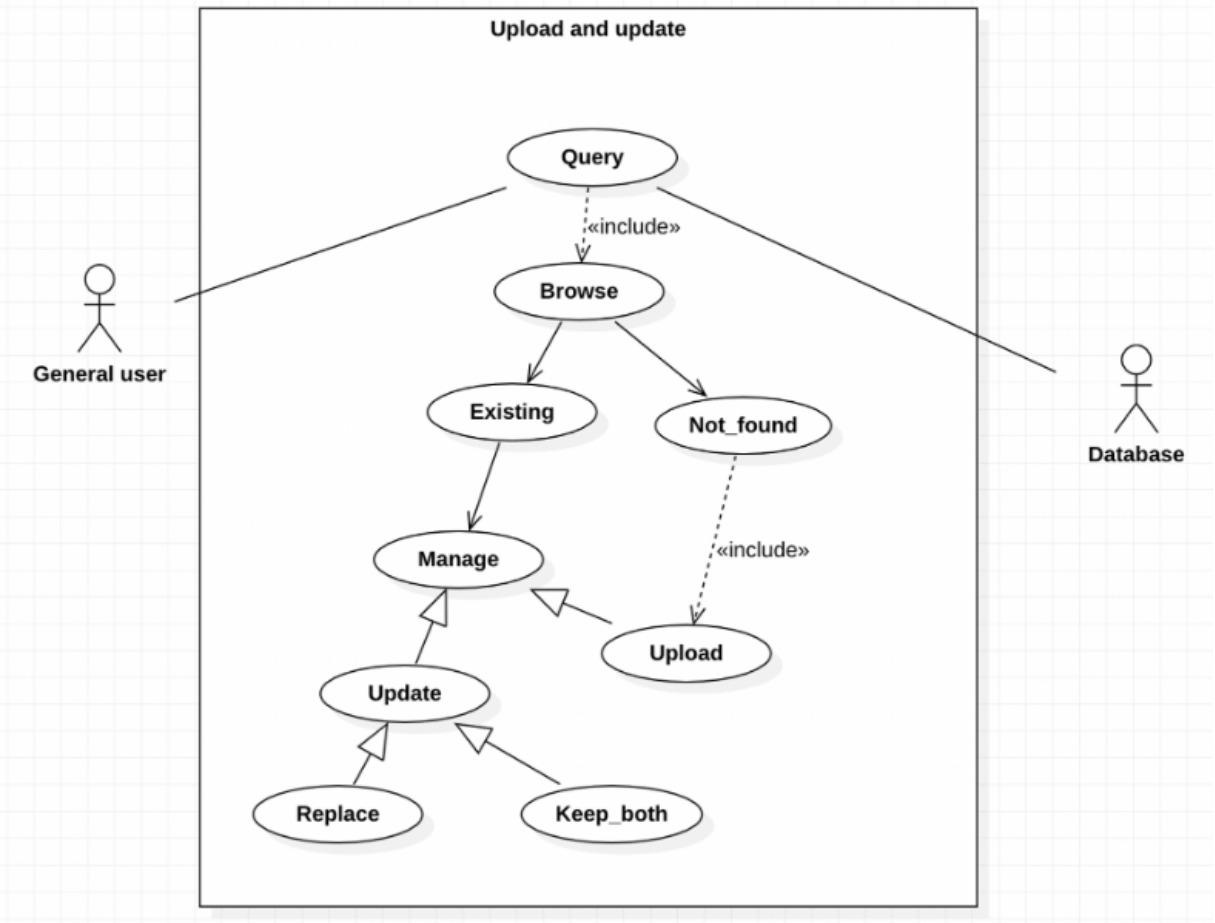
ASSUMPTIONS AND DEPENDENCIES

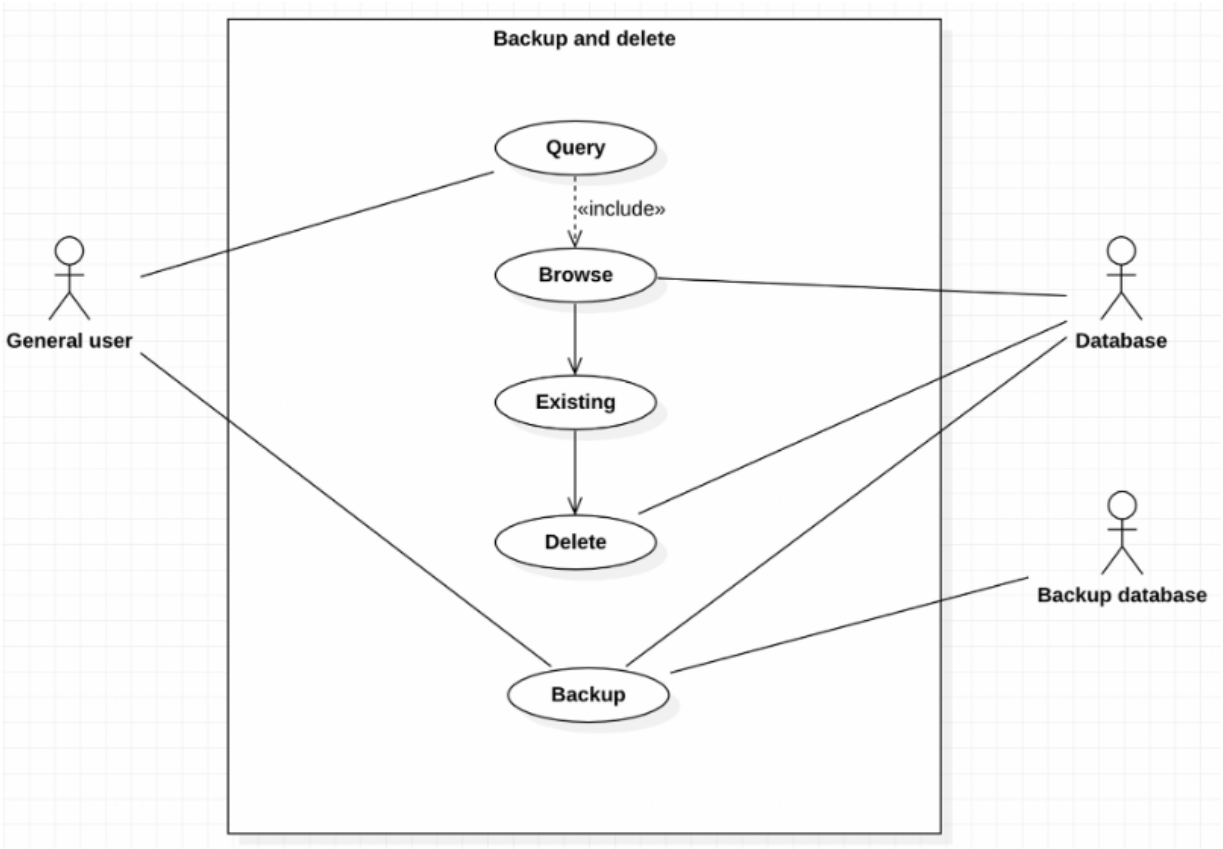
1. The system will use industry-standard encryption and security mechanisms to ensure the safety and privacy of user data.
2. The system will be compatible with a wide range of file types, including text documents, images, videos, and audio files.

USE CASE DIAGRAMS


Authentication and security
User
Registration
«include»
Log In
User Access
File Permissions
Read
Write
Authentication Server
Authorization Server
Storage System


Search and download
Query
«include»
Browse
Existing
Download
General user
Database


Upload and update
Query
«include»
Browse
Existing
Not_found
Manage
«include»
Upload
Update
Replace
Keep_both
General user
Database



DESIGN PATTERNS USED

1. Single Responsibility Principle (SRP) : The Controller class has a single responsibility of handling HTTP requests and returning responses. It does not contain any business logic or database access code, which are delegated to separate classes.

2. Dependency Inversion Principle (DIP) : The Controller class depends on the Connection object, which is passed to it via constructor injection. This allows the code to be more easily tested, as it can be mocked in unit tests.

3. Open-Closed Principle (OCP) : The code follows OCP by extending the functionality of the system without modifying existing code. The Controller class can be extended with additional endpoints and functionality without modifying the existing code.

4. Loose Coupling and High Cohesion (GRASP) : The code follows the GRASP principle of loose coupling by decoupling the Controller from the database access code, which is located in separate classes. The code also follows the principle of high cohesion by grouping related methods and data fields in the Controller class.

5. Factory Method Pattern : The DriverManager.getConnection method is an example of the Factory Method Pattern, which provides a centralized mechanism for creating objects.

6. Data Access Object (DAO) Pattern : The code demonstrates the use of the DAO pattern by separating database access code into a separate class. The Model class encapsulates data and the UserDao class encapsulates database access code.

7. Builder Pattern : The Model class has a constructor with multiple parameters, which can make it difficult to read and maintain the code. The Builder Pattern can be used to simplify the creation of objects with many parameters.