National University of Sciences & Technology

School of Electrical Engineering and Computer Science

Faculty of Computing

CS 833: Cloud Computing (3+0) MSCS Spring 2024

|  |  |  |
| --- | --- | --- |
| Assignment 1: Development of a monolithic cloud-centric data storage application | | |
| Full Name:  Full Name:  Full Name:  Full Name: | Qalam ID:  Qalam ID:  Qalam ID:  Qalam ID: | **Class:** MSCS |
| Maximum Marks: 10 | Instructor: Dr. Qaiser Riaz | **Maks Obtained:** |

Date of submission: 11.03.2024

**Motivation**

The cloud computing assignments constitute a vital initiative aimed at equipping students with the skills to develop practical, cloud-centric solutions aligned with modern software design and analysis paradigms. This encompasses the utilization of modern tools and frameworks, problem formulation, design, development, evaluation, testing, and deployment of diverse modules and services embodying distinct features of the application. The application, adopting a multiuser/multi-tenant role-based approach, caters to various levels such as presentation, business logic, and persistence.

(**Note:** This is a collaborative group activity (4-5 members in each group), and equal contribution from all group members is expected.)

**Description:**

Create a web-based data storage application, drawing inspiration from Dropbox, utilizing PaaS and deploying it as a SaaS application on a public cloud AWS. The application will be monolithic in nature, and it will allow end-users to upload single or multiple files, replace existing files, create and manage folders and subfolders, browse files and folders, and perform bulk deletiphtoons. The key features representing different components of the application are outlined below.

**Key Components:**

1. **Identity and access management:** (using relational databases or NoSQL as per your choice). The objective of this module is to let the end users to signup, log-in and access their data. The administrators should be able to control the access of a user e.g., activating/deactivating a user, deleting a user, updating a user etc. Implement it as a separate module namely **IdntyAccMgmtServ** with exposed API so that other parts of the system can interact with it when required.

2. **Storage management:** Each new user upon signup will get 10MBs of cloud storage to store files and folders. Keep track of the user's storage in the database and generate an alert to the user when 80% of the storage is consumed. The user should not be able to upload any new files or folders once 100% of the storage is consumed (unless the user deletes some of the existing files to create additional space). Create a corresponding module namely **StorageMgmtServ** with exposed API.

3. **Usage monitoring:** Keep usage track of user’s stored data in terms of data volume. Although the user can only use 10MBs of allocated storage, the user can delete & create new content resulting in utilizing higher bandwidth. Define a threshold and alert the user when he exceeds a specific limit (e.g. 25MBs per day). Create a separate module namely **UsageMntrServ** with exposed API.

4. **View** will be based on HTML, CSS, or any other client/server-side libraries (javascript, jquery etc.) to display uploaded data in the folders/sub-folders hierarchy. Display additional properties of the folders/files (e.g., size, create/update date, owner, options to edit/delete etc.) in the browser in a presentable form. Create a module **ViewGeneratorServ** which will interact with the above-mentioned modules in the display.

5. You will need additional modules e.g. **Controller, Model** for the proper functioning of the application. Think about which additional modules will be required and implement them accordingly.

6. **Load test** your data storage app to see how many concurrent users can access the website without compromising the quality of service. Keep a log of load at any specific time. During the load test, keep the auto-scaling feature off so that you can find a realistic number. Automated load testing tools can be used. Generate graphs of load testing using automated tools and include them in the report.

7. **Logging:** Keep general logging files to monitor activities performed by the user using appropriate logging module.

8. **Report:** Write a short report which should include the architecture diagram of the application showing different modules, the APIs and other components along with the important features of your data storage app. Make sure to include the working URL of your application.

**Submission:** Create a single zip file which should include report and source code (under src folder) and only one member per group should submit it on LMS.

**Good Luck!**

**Plagiarism Policy:**

Strictly adhere to NUST's plagiarism policy, ensuring that all submitted work is original and appropriately referenced in the report. Any plagiarized submissions, including those generated with ChatGPT or similar applications, will be treated in accordance with the department's policy, resulting in severe consequences.