**Description:**

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.