**Full Stack Java Developer**

Capstone Project Problem Statement



**DataMax: A Safe and Secure Data-Sharing Platform**

**Problem statement:**

Design and develop an online application that supports safe file sharing within an organization with added features like end-user authentication when downloading the data

**Scenario:**

**DataMax** is a safe and secure data-sharing platform that allows end users to upload and download data. Uploading data will be constrained by fewer restrictions and authentication requirements when compared to downloading data. Novel features will be implemented to ensure security when downloading data.

**The flow of the application:**

Within an organization, users can register for an account within the DataMax system. With this account, users can upload and download files and documents. Security algorithms (like AES encryption) will be implemented to protect the upload and download functions.

**Expected deliverables:**

**Features of the application:**

* Home page
* About page
* Login page
* Register page
* Upload file page
* Download the file page
* Admin panel

**Recommended technologies:**

* Database management: MySQL
* Backend logic: Java Programming (Spring Boot, JPA, or Hibernate)
* Front-end development: Angular, HTML, CSS, and Bootstrap
* Automation and testing technologies: Selenium and TestNG
* DevOps and production technologies: Git, GitHub, Docker, and Jenkins
* Optional implementation: Kubernetes and AWS

**Project development guidelines:**

* The project will be delivered in three sprints, with every sprint delivering a minimally viable product.
* It is mandatory to perform proper sprint planning with user stories to develop all project components.
* The learner can use the above-mentioned technologies for different layers of the project.
* The web application should be responsive and should fetch or send data dynamically without hardcoded values.
* The learner must maintain the version of the application over GitHub and every new change should be sent to the repository.
* The learner must implement a CI/CD pipeline using Jenkins.
* The learner should also deploy and host the application on an AWS EC2 instance.
* The learner should also implement automation testing before the application enters the CI/CD pipeline.
* The learner should use Git branching to separately perform the basic automation testing on the application.
* The learner should make a rich frontend for the application, which is user-friendly and easy for the user to navigate through the application.
* There will be two portals in the application: the admin and the user portal.
* The learner should implement validation within the backend Spring Boot layer, ensuring data passed by the API meets the system's constraints.
* The learner should host the backend application on a local server, and the hosted API should be used by Angular to communicate with the backend.

**Functionalities of the Admin User:**

It deals with all the backend data generation and product information. The admin user should be able to:

* + Authorize the roles and guidelines for the user
  + Grant access to the user regarding file upload and/or file download
  + Grant access to the user regarding creating files and folders
  + Block the user account in case of any threat to the data
  + Authorize the files marked as sensitive to be downloaded
  + Confirm implementation of data security algorithms during the different operations

**Functionalities of the End User:**

It deals with user activities. The user should be able to:

* + Register or log in to the application
  + Upload files to their account
  + Provide access to their files to others
  + Request other accounts for access as and when needed
  + View uploads and downloads history
  + Copy files between folders/directories/destinations within their account
  + Request other accounts for file access/downloads

**Validations (Frontend/Backend):**

* Password validation
* File size validation
* The end user downloading the file should be an authenticated user
* The end user downloading the file should be part of that organization
* Password creation validation
* User type creation validation with a set of permissions. For example, the user belongs to the organization.
* Additional validations will be added at the time of solution development where appropriate

**Business logic exception scenarios:**

* **FileStorageException:**

An unexpected situation that occurs while storing a file in the system.

* **FileNotFoundException:**

It is thrown when the file that the user is trying to download is not found.

* **IOException:**

It is thrown then there are two files with the same name.

**The following requirements should be met:**

A student must achieve the following steps in each phase:

* Installations (Java JDK, Apache Maven, and Spring Tool Suite (STS))
* Creating a starter project
* Configure file storage properties to enable multipart file loads that will be uploaded for sharing across the organization
* Configure the directory in which the files will be stored
* Define the maximum file size that can be uploaded
* Storing the uploaded files in the target directory
* Binding the properties defined in the properties file to a POJO class
* Enable configuration properties
* Build REST API for uploading and downloading files
* Build a service for storing files in the system and retrieving them as and when needed
* Optional: Build containers using Docker and Spring Cloud
* Optional: Implement Kubernetes-based orchestration

**Screenshots:**

**Admin user**

Login page:

Graphical user interface

Description automatically generated

Admin dashboard:

Graphical user interface, text, application

Description automatically generated

Authorize the goals and guidelines for the user:

Graphical user interface, application

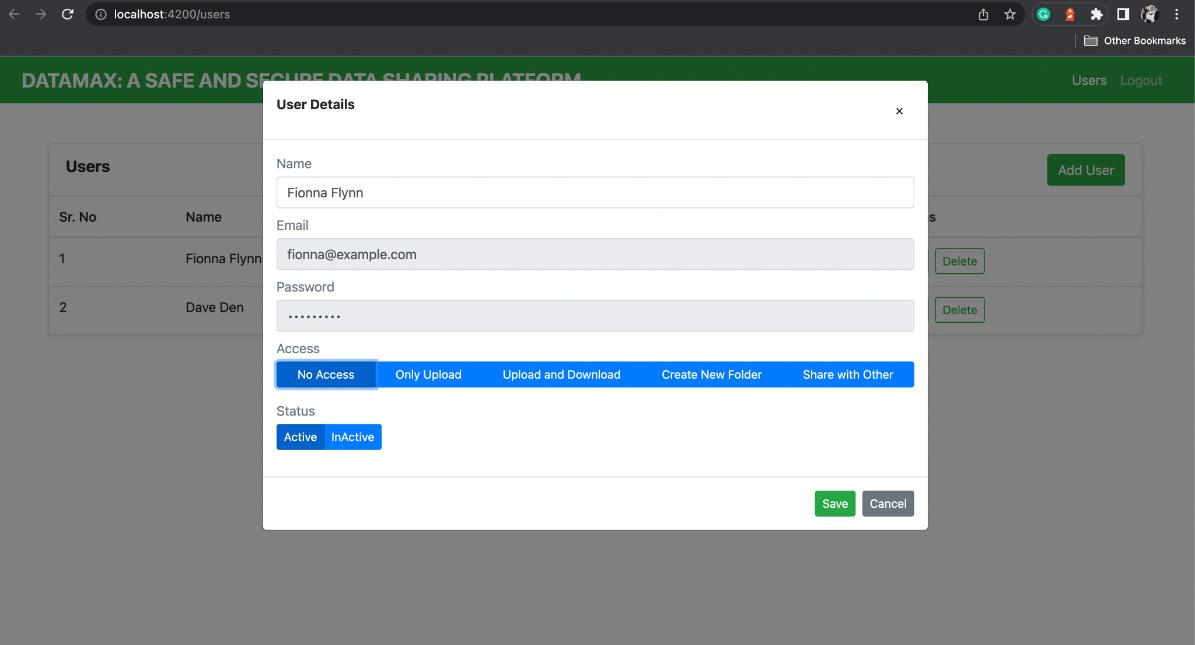
Description automatically generated

Grant access to the user regarding file upload or download:

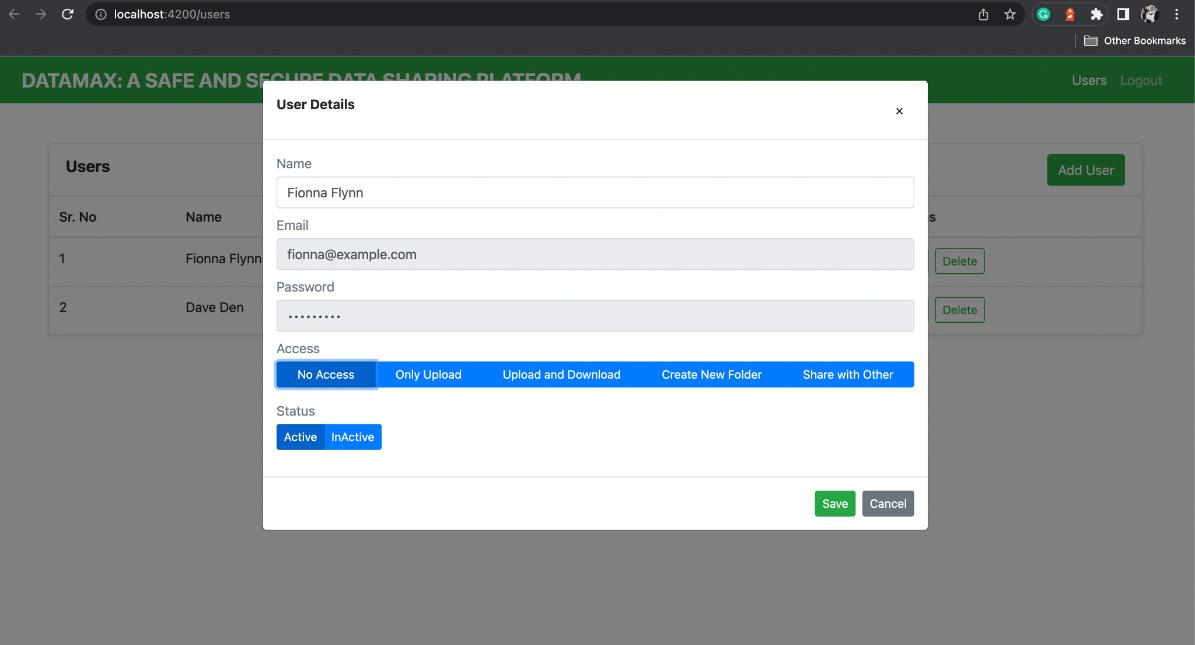
Graphical user interface, application

Description automatically generated

Grant access to the user regarding creating and sharing files and folders:



To block any user, modify the status to inactive:



**End user**

Register/Create user page:

Graphical user interface

Description automatically generated

Login page:

Graphical user interface

Description automatically generated

End-user dashboard after login:

Graphical user interface, website

Description automatically generated

Create folder component o/p:

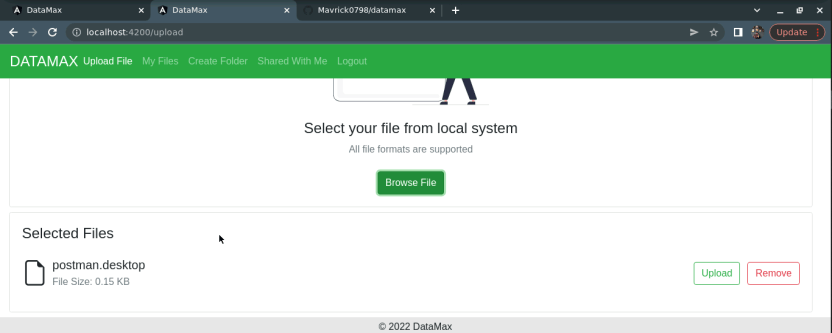
Graphical user interface, application

Description automatically generated

**Graphical user interface, application, Teams

Description automatically generated**

Upload a file:



Share and provide access to their files to others:

Graphical user interface, text, application

Description automatically generated

Sharing files:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, chat or text message

Description automatically generated

Copy and Move files:

Graphical user interface, text, application

Description automatically generated

These options appear when we are copying and moving the files.

Graphical user interface, application

Description automatically generated

My files component o/p:

**Graphical user interface, text, application

Description automatically generated**

Shared with me component o/p:

Graphical user interface, text

Description automatically generated