Vault Engine

**Problem Statement:**

Create a Vault Engine

**My Understanding:**

The objective is to create a Secret Engine that provides a secure way to store passwords. The Vault Engine should be designed to ensure the confidentiality and integrity of the stored data , while allowing authorized users to access the data when needed.

**Vault Engine have the following features:**

User Sign-Up : Any unregistered user can create an account.

Secret Message storage : User can able to store multiple secret messages securely in the Vault Engine.

Access Control : The Vault Engine enforce access control to ensure that only authoried users can access to the stored secret messages.

Encryption : Vault Engine uses ascii values for encryption and to protect the confidentiality of stored messages.

**Implementation:**

**Encryption:**

In this project, we use the Ascii values to encode the user’s secret message. We have created a class with name “EncodeAndDecode” **.** In that class we created two methods one is to encode the secret message using the ascii values and another method is to decode the secret message when required. When the user enters the password , the password is encrypted by the password encoder. The encrypted password is stored in the database.

**User profile creation:**

When user creates a new profile. User service sends a request to store the user details in the database. This includes username and password.

We validate the user using user id from the database. If the user tries to access with the id which is not present in the database, the application deny access to application.

**Password Strength:**

While the user is being created the application validates the password strength only after the password is identified as the strong password user can be able to create an account. We are validating the password strength using the pattern. Password must include the following characters atleast one capital letter, one special character, one number, and length of the password must be greater than the 8 characters.

**Confidential Message Storage:**

User can store the message of any length like passwords, secret information etc. SecretMessage Service store the message in the database. It stores the information securely and associates with the user profile.

To ensure the confidentiality of the Messages, the application uses encryption and other security measures. For Example , the messages are encoded before being stored in the database, and access to the messages is restricted to authorized users only.

**Access Control :**

To enforce access control, the application verifies whether the user exists or not in the database. So that only authorised users are accessing their data. When user attempts to access their it checks whether the user id is matching with the credentials stored in the database.

**Record maintainance:**

We have created an entity with the name records . It store the information related to the Secret Message like whenever an user creates an secret message an entry will be made in the records table with the details of user, modified date and time and the Name of the file which they have created or altered.

**Database Schema:**

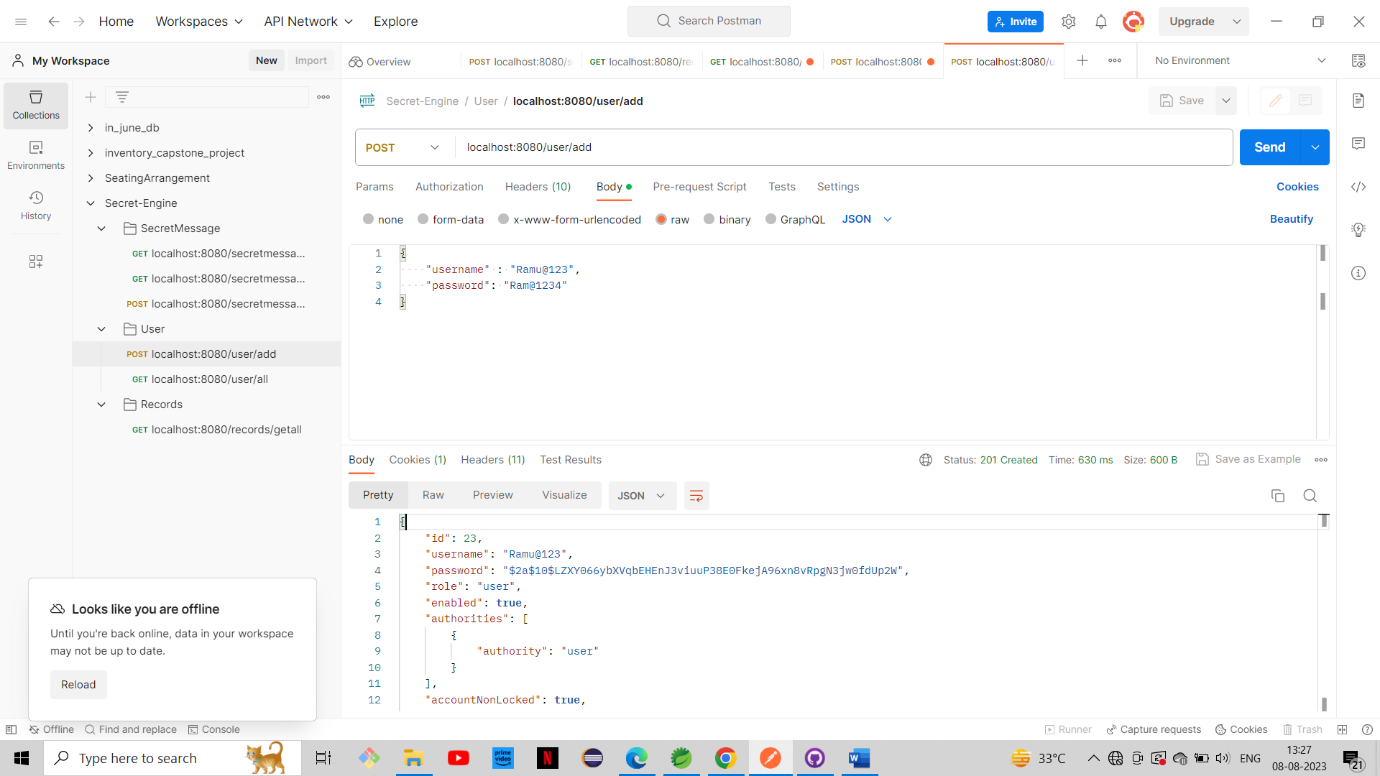
The database structure is the critical aspect of the project. We use a well-designed database structure to store the data in an efficient way.

Database is designed to store the user details and secret message details like name and secret message. We also store the data of records whenever an secret message is created or altered.

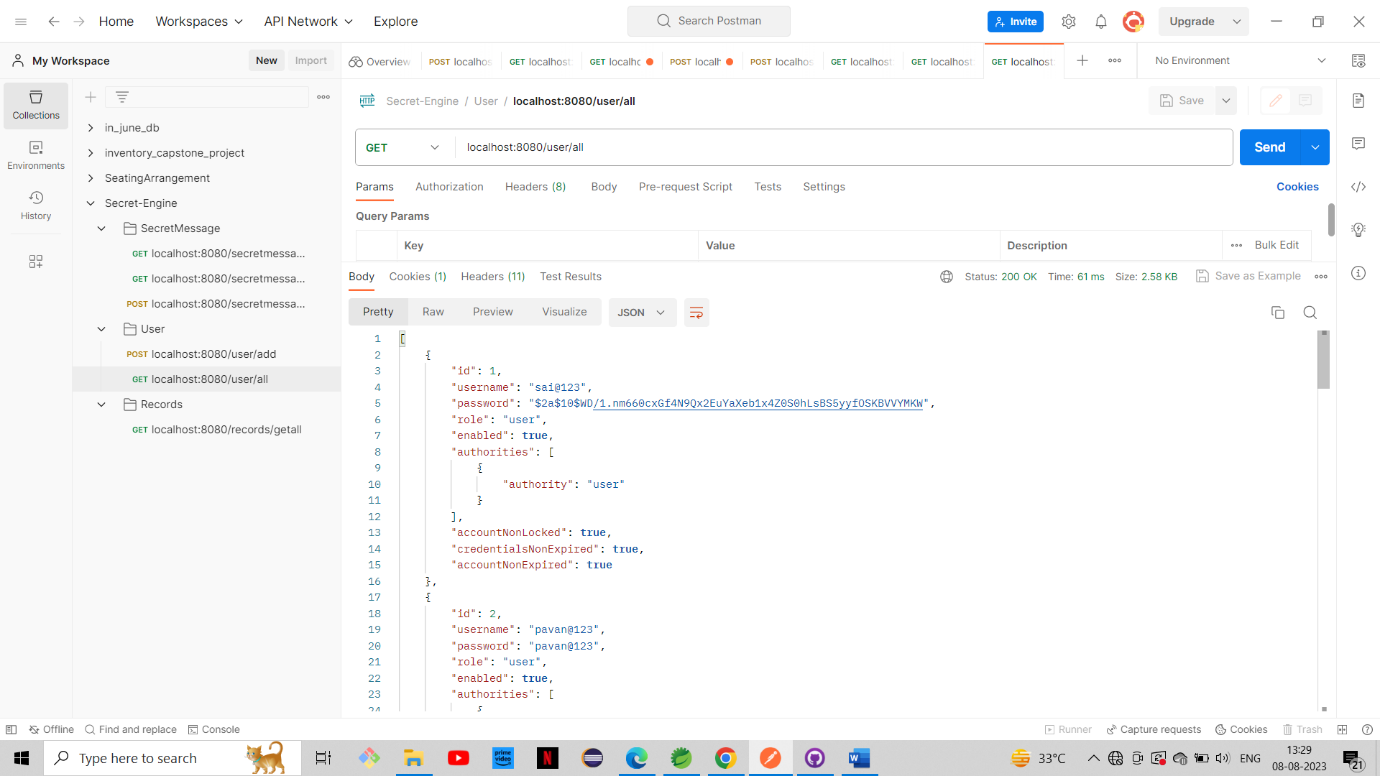
**Screenshots:**

1. Saving the user details into the database.

URL: localhost:8080/user/add

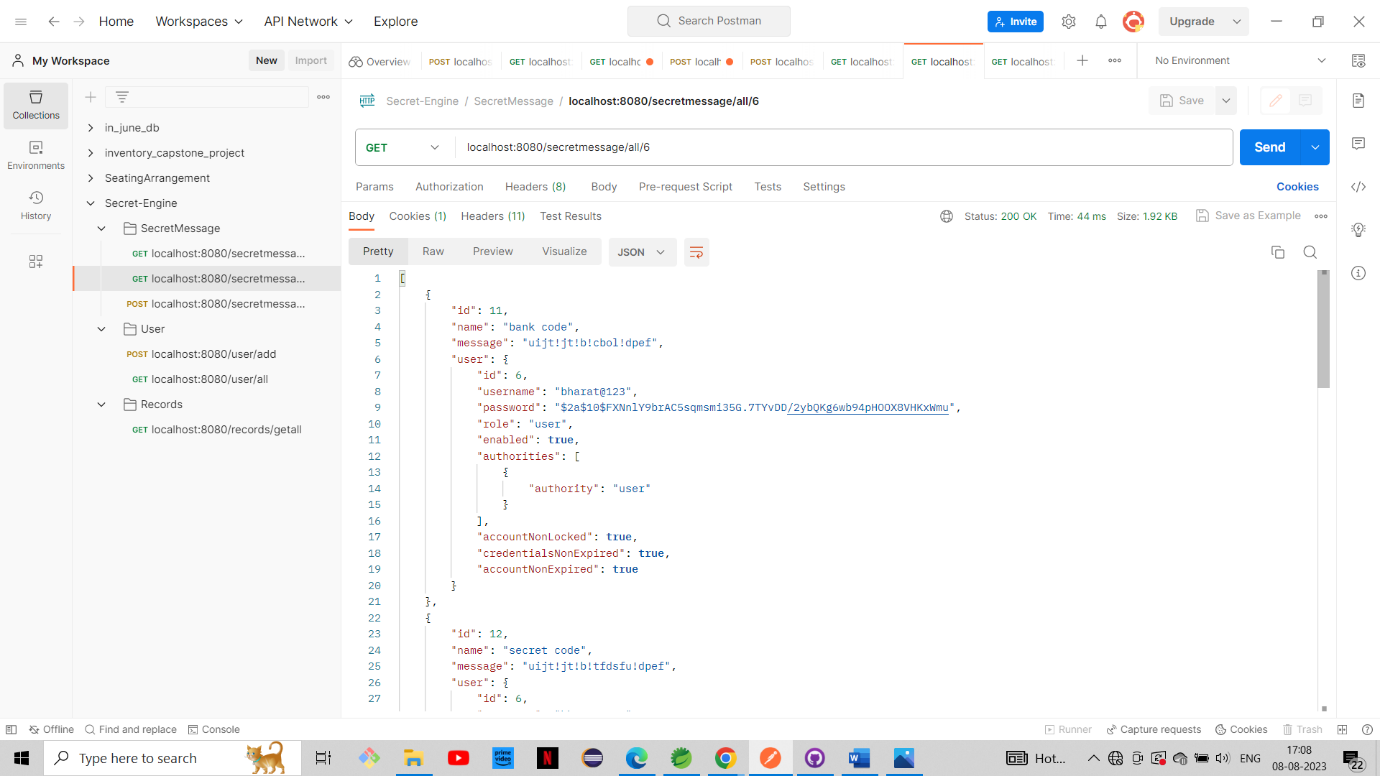


1. Getting all the users from the database  
   URL: localhost:8080/user/all



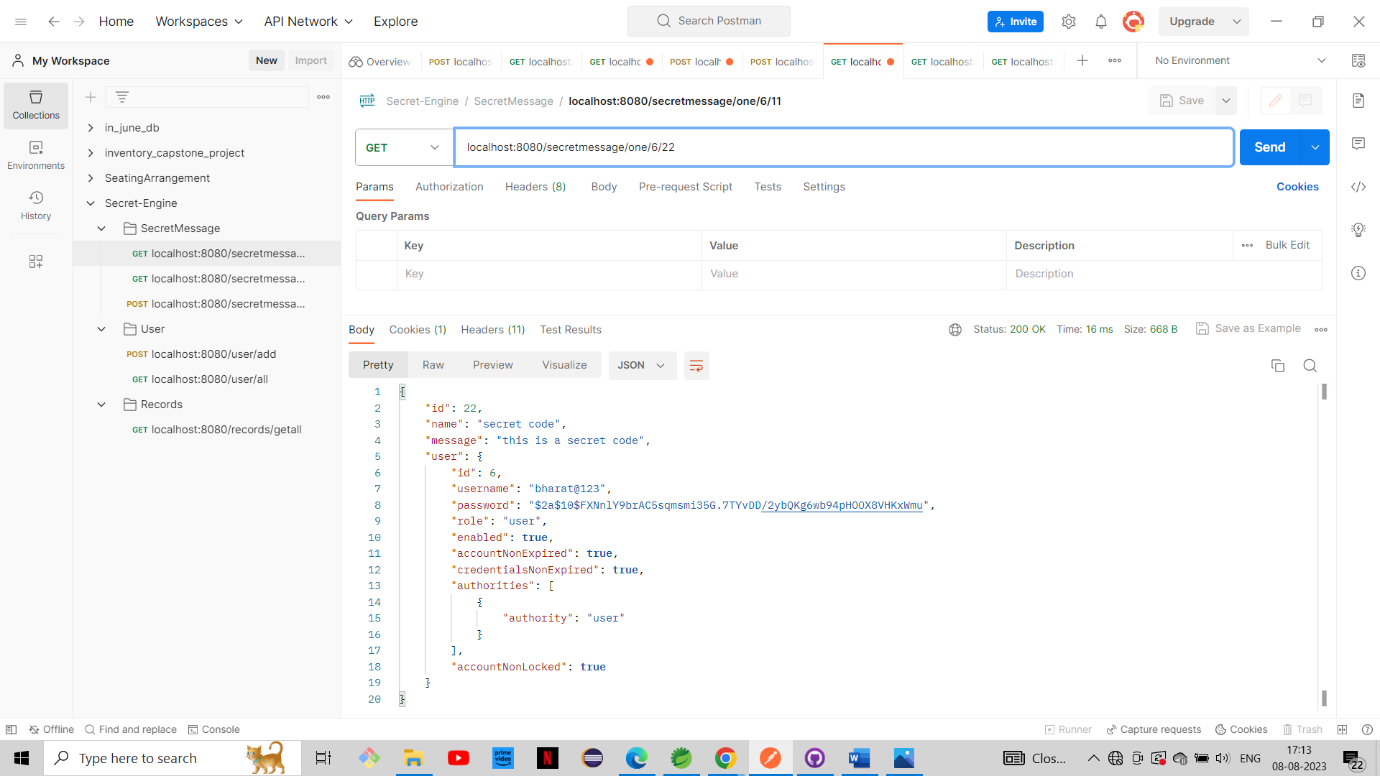
1. Getting all the Secret Messages with the particular user.

URL: localhost:8080/secretmessage/all/6



1. Get one secret message

URL: localhost:8080/secretmessage/one/6/20



1. List of Records

URL: localhost:8080/records/getall

