Highly confidential Security System

Objective:

Highly confidential Security System is an online application to be built as a product that the system will help user in logging in to the client system for which it is holding/storing the password, either by the software interface or directly by hardware interface.

Users of the System:

1. Admin
2. User

Functional Requirements:

- Build an application that user can use the Security System (Software and hardware).
- The application should have a mail id and password locker.
- This application should have a Bank account information locker.
- This application should have a Video, Audio, Image locker.
- Also, an integrated platform required for admin and customer.
- **Maximum 1 Account per email**
- **The filename should be variant from other files**

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

- ➢ Build such that it is difficult to hack through.
- ➢ Multi-factor authentication for the sign-in process

Output/ Post Condition:

- ➢ Admin report
- ➢ Viewable and downlable reports with password protection
- ➢ Standalone application / Deployed in an app Container

Non-Functional Requirements:

Security	• App Platform –UserName/Password-Based Credentials • Sensitive data has to be categorized and stored in a secure manner • Secure connection for transmission of any data
Performance	• Peak Load Performance • Highly confidential Security System -< 3 Sec • Admin application < 2 Sec • Non Peak Load Performance
Availability	• 99.99 % Availability
Standard Features	• Scalability • Maintainability • Usability • Availability • Failover

Logging & Auditing	• The system should support logging(app/web/DB) & auditing at all levels
Monitoring	• Should be able to monitor via as-is enterprise monitoring tools
Cloud	• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	• IE 7+ • Mozilla Firefox Latest – 15 • Google Chrome Latest – 20 • Mobile Ready

Technology Stack

Front End	React Google Material Design Bootstrap / Bulma
Server Side	Spring Boot Spring Web (Rest Controller) Spring Security Spring AOP Spring Hibernate
Core Platform	OpenJDK 11
Database	MySQL or H2

Platform Pre-requisites (Do’s and Don’ts):

1. The React app should run in port 8081. Do not run the React app in the port: 3000.
2. Spring boot app should run in port 8080.

Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
2. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
4. Adhere strictly to the endpoints given below.

Application assumptions:

1. The login page should be the first page rendered when the application loads.

2. Manual routing should be restricted by using AuthGaurd by implementing the canActivate interface. For example, if the user enters as http://localhost:3000/signup or http://localhost:3000/home the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.
2. Basic mobile validation should be performed.

Project Tasks:

API Endpoints:

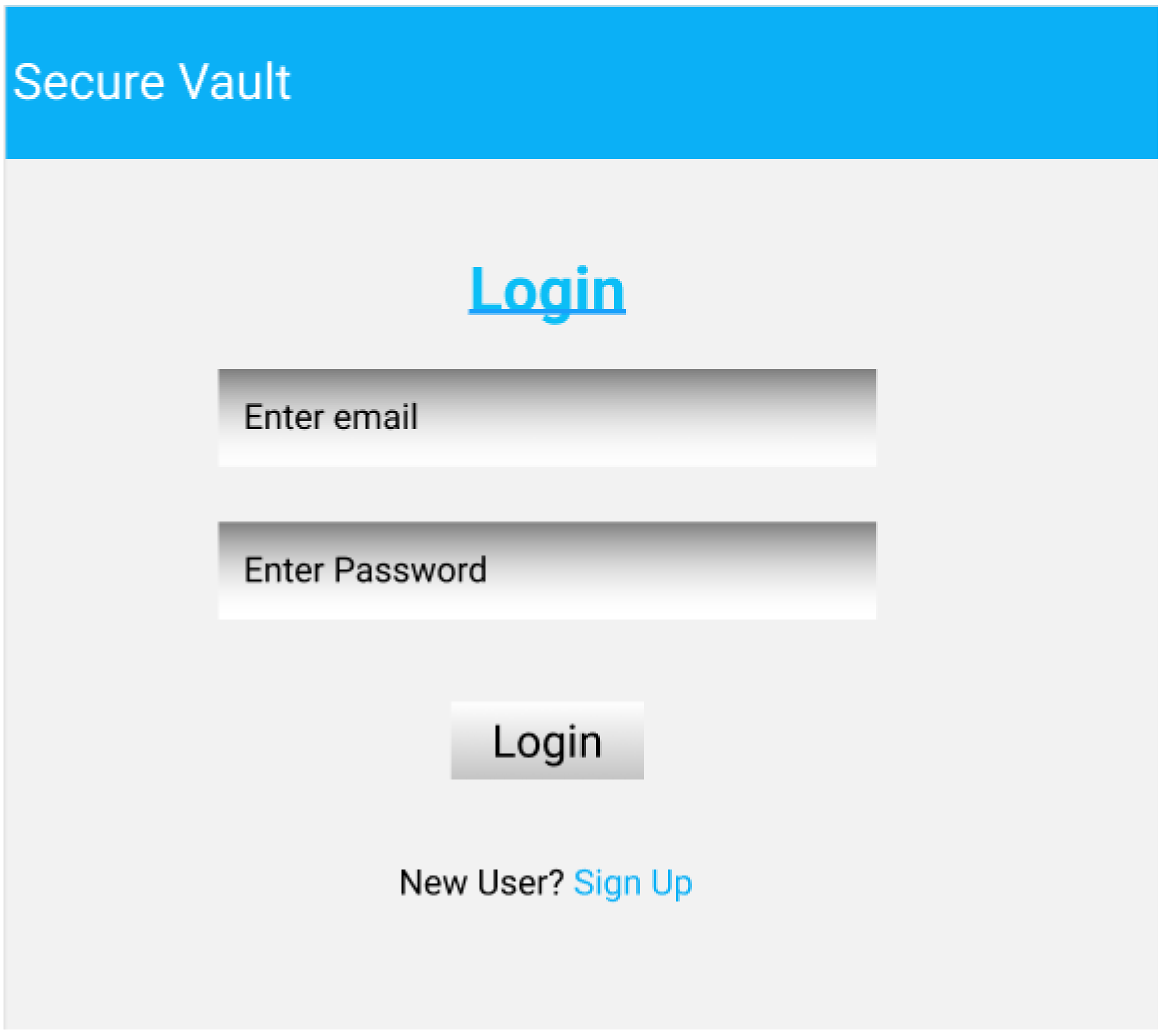
USER			
Action	URL	Method	Response
Login	/login	POST	true/false
Signup	/signup	POST	true/false
Get Bank Information	/bank	GET	Array of Bank details
Add Bank Information	/bank/{id}	POST	Information Added Successfully
Update Bank Information	/bank/{id}	PUT	Information Updated
Delete Bank Information	/bank/{id}	DELETE	Information Deleted
Get Media Information	/media	GET	Array of Media Details
Add Media	/media/{id}	POST	Media Added
Update Media	/media/{id}	PUT	Media Updated
Delete Media	/media/{id}	DELETE	Media Removed
Get Credentials Information	/credentials	GET	Array of Credentials details
Add Credentials Information	/credentials/{id}	POST	Credentials Added Successfully
Update Credentials Information	/credentials/{id}	PUT	Credentials Updated
Delete Credentials Information	/credentials/{id}	DELETE	Credentials Deleted
ADMIN			
Action	URL	Method	Response
Get All Users	/admin/user	GET	Array of users
Approve User	/admin/approveUser	POST	Approved Successfully
Remove User	/admin/delete/{id}	DELETE	User Removed
Update User	/admin/update /{id}	UPDATE	User Updated
Get Specific User	/admin/user /{id}	GET	Particular User Detail

Frontend:

User:

Login:

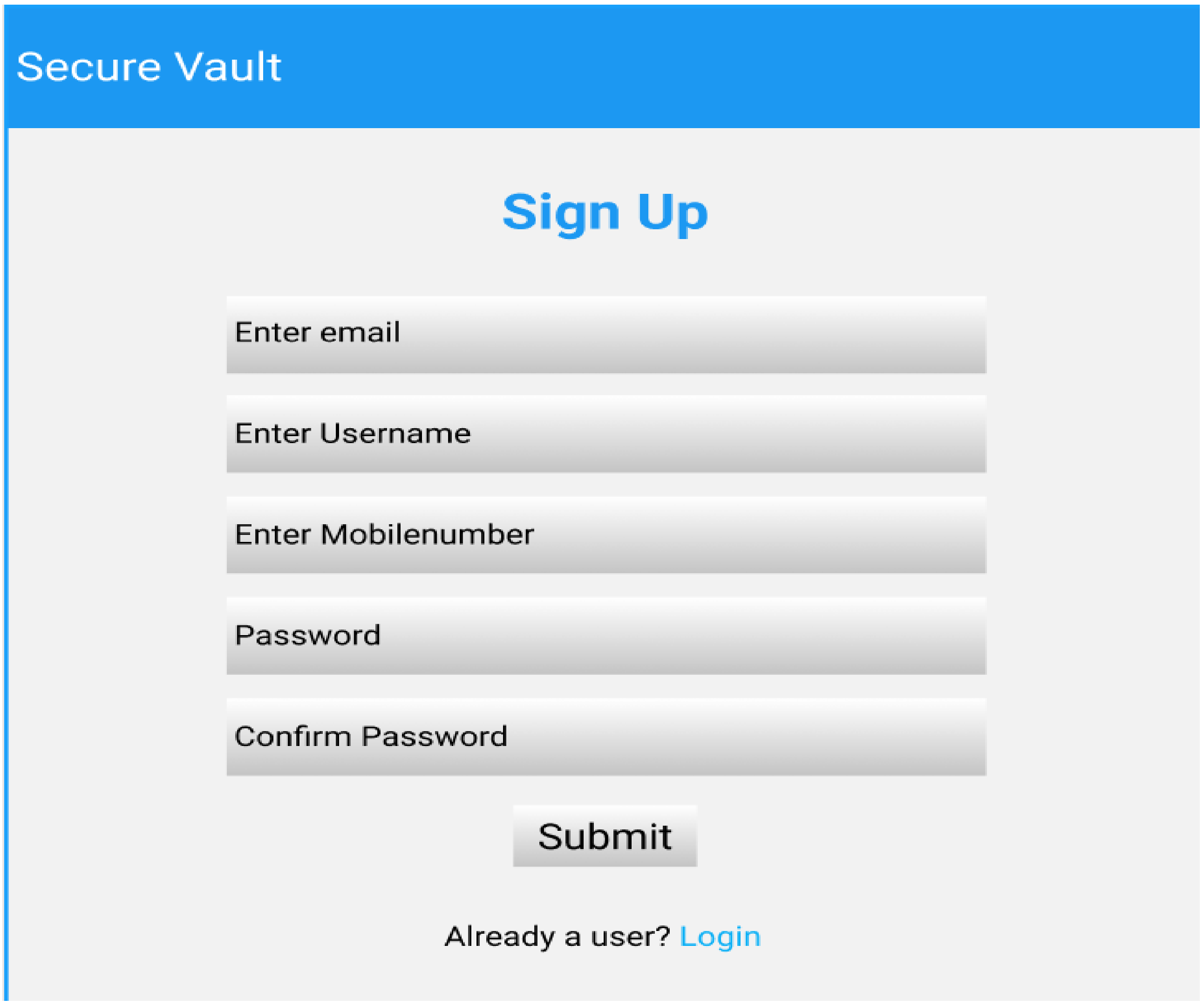
Output Screenshot:


Secure Vault

Login

Signup:

Output Screenshot:


Secure Vault

Sign Up

Enter email

Enter Username

Enter Mobilenumber

Password

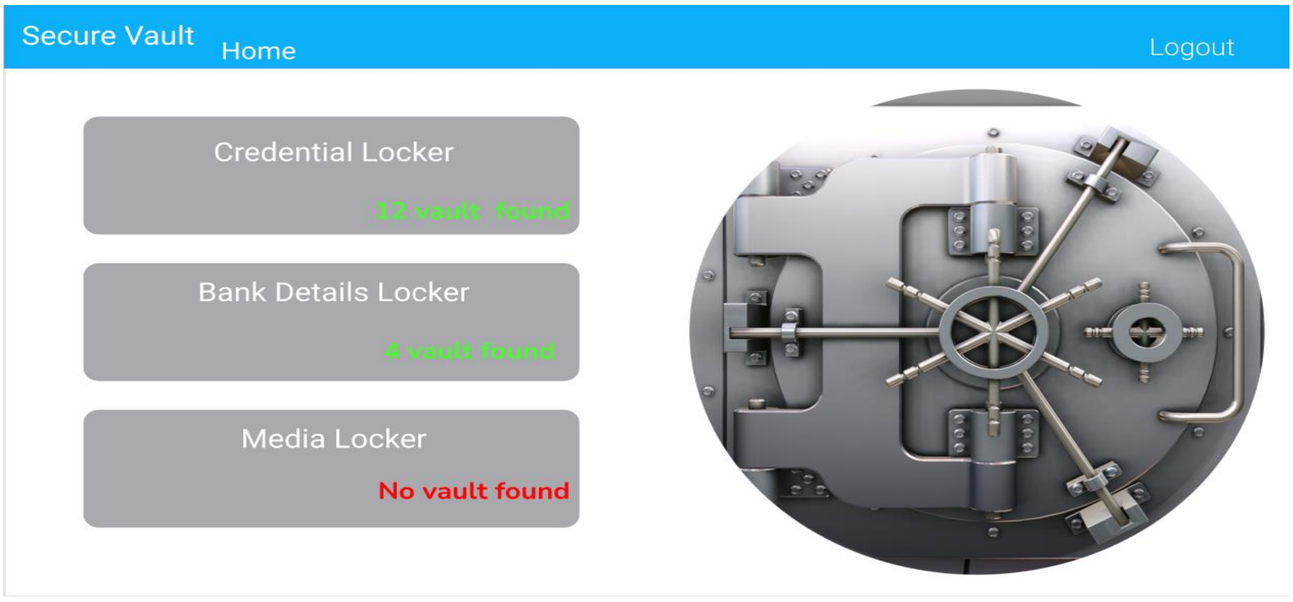
Confirm Password

Submit

Already a user? Login

Home:

Output Screenshot:


Secure Vault Home Logout

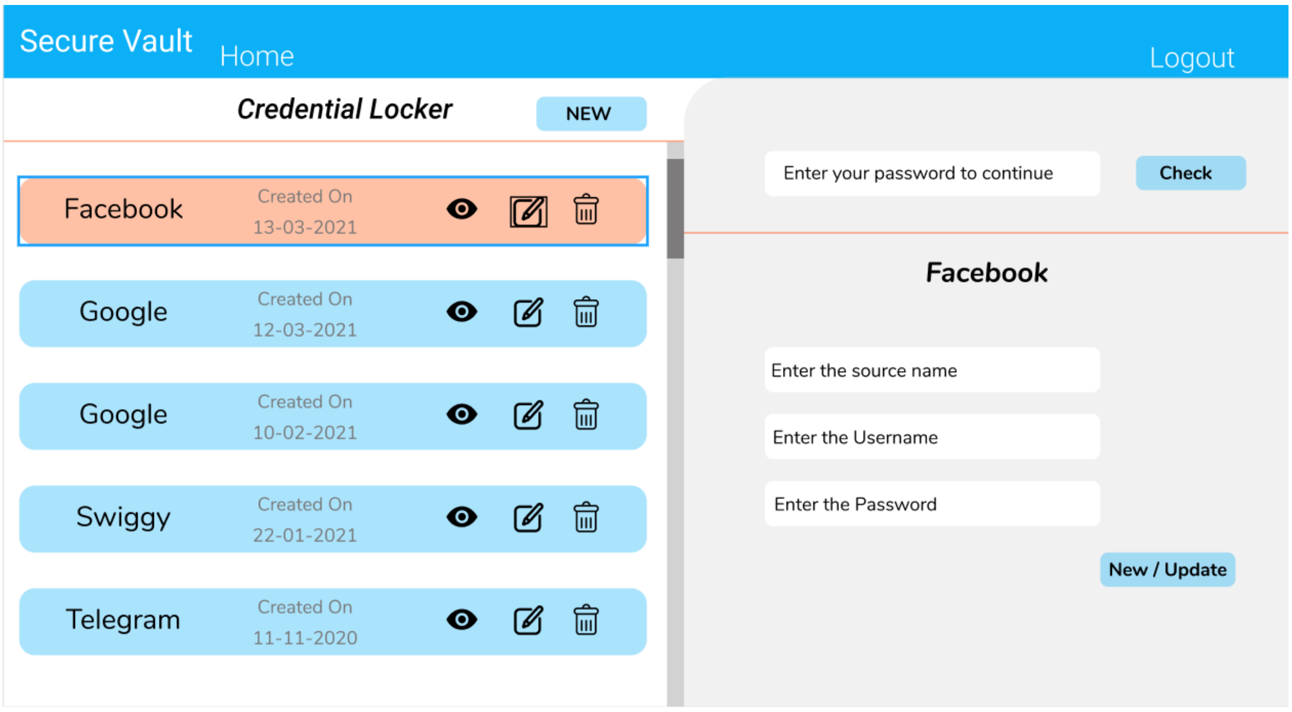
Credential Locker
12 vault found

Bank Details Locker
4 vault found

Media Locker
No vault found

Credential Locker:

Output Screenshot:



Secure Vault Home Logout

Credential Locker NEW

Facebook Created On 13-03-2021

Google Created On 12-03-2021

Google Created On 10-02-2021

Swiggy Created On 22-01-2021

Telegram Created On 11-11-2020

Enter your password to continue Check

Facebook

Enter the source name

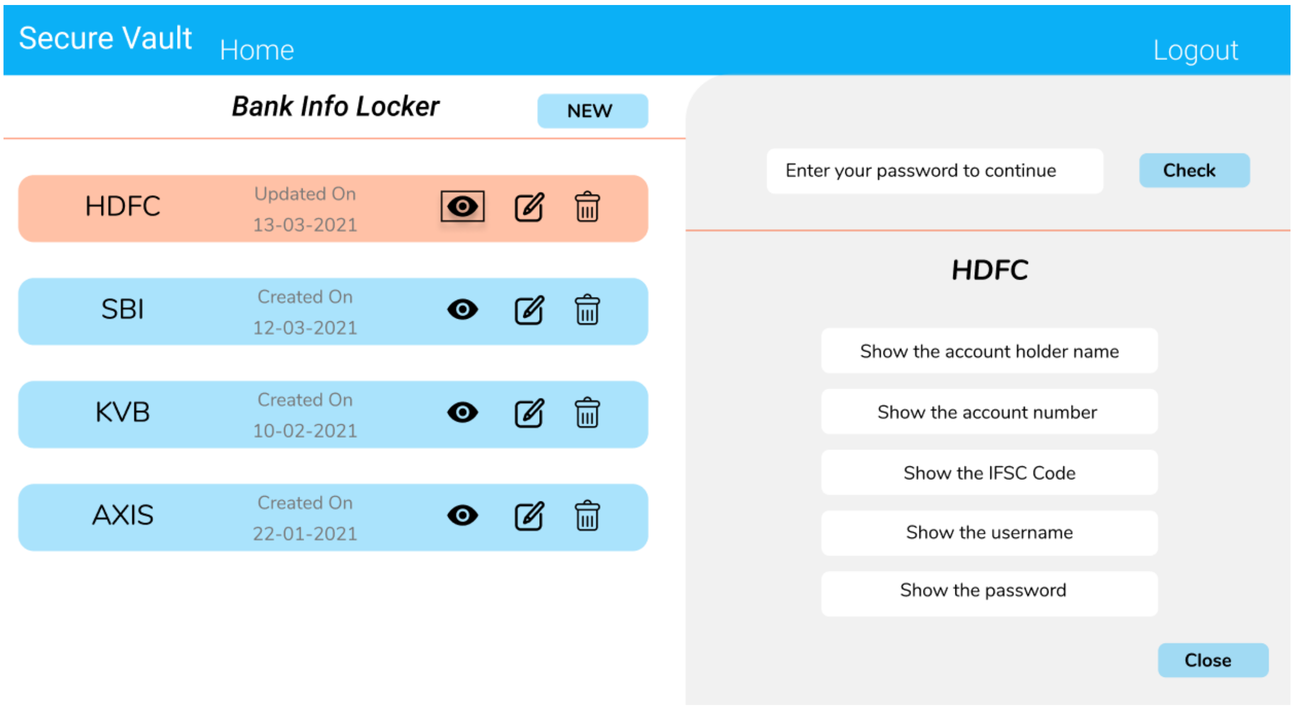
Enter the Username

Enter the Password

New / Update

Bank Info Locker:

Output Screenshot:



Secure Vault Home Logout

Bank Info Locker NEW

HDFC Updated On 13-03-2021

SBI Created On 12-03-2021

KVB Created On 10-02-2021

AXIS Created On 22-01-2021

Enter your password to continue Check

HDFC

Show the account holder name

Show the account number

Show the IFSC Code

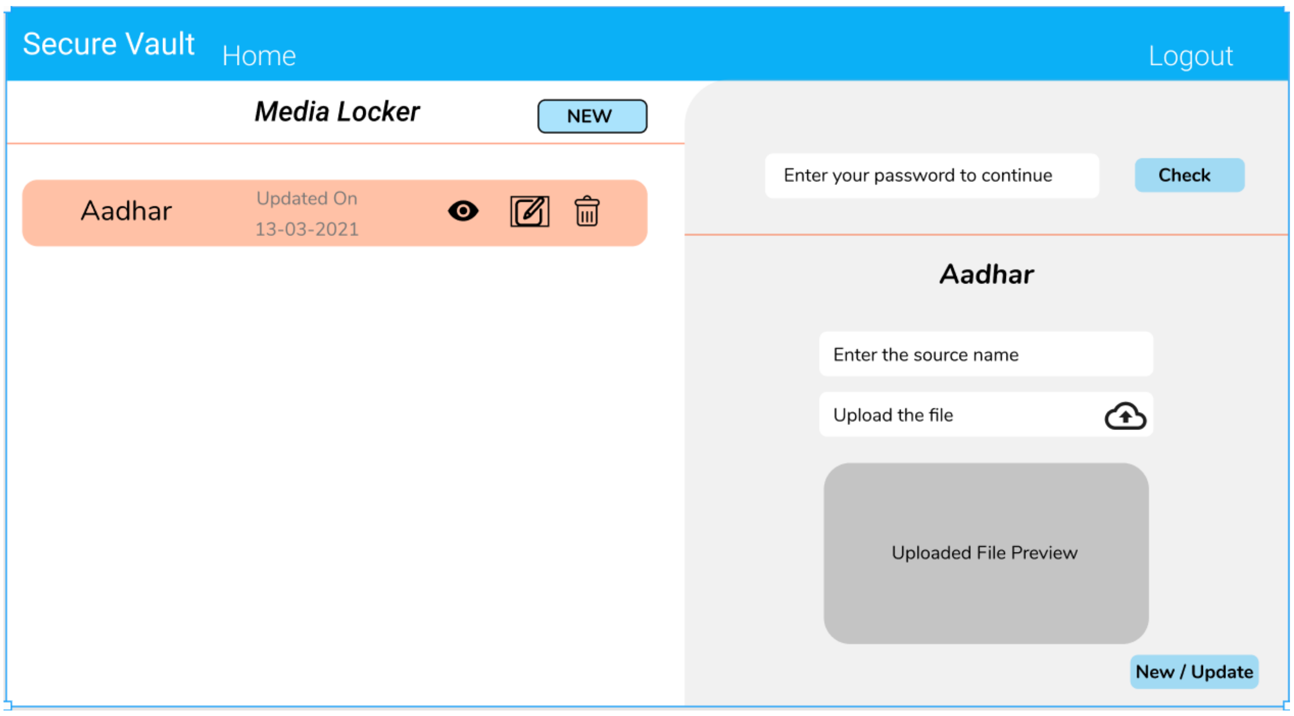
Show the username

Show the password

Close

Media Locker:

Output Screenshot:


Secure Vault Home Logout

Media Locker NEW

Aadhar	Updated On 13-03-2021	

Enter your password to continue — Check

Aadhar

Enter the source name

Upload the file

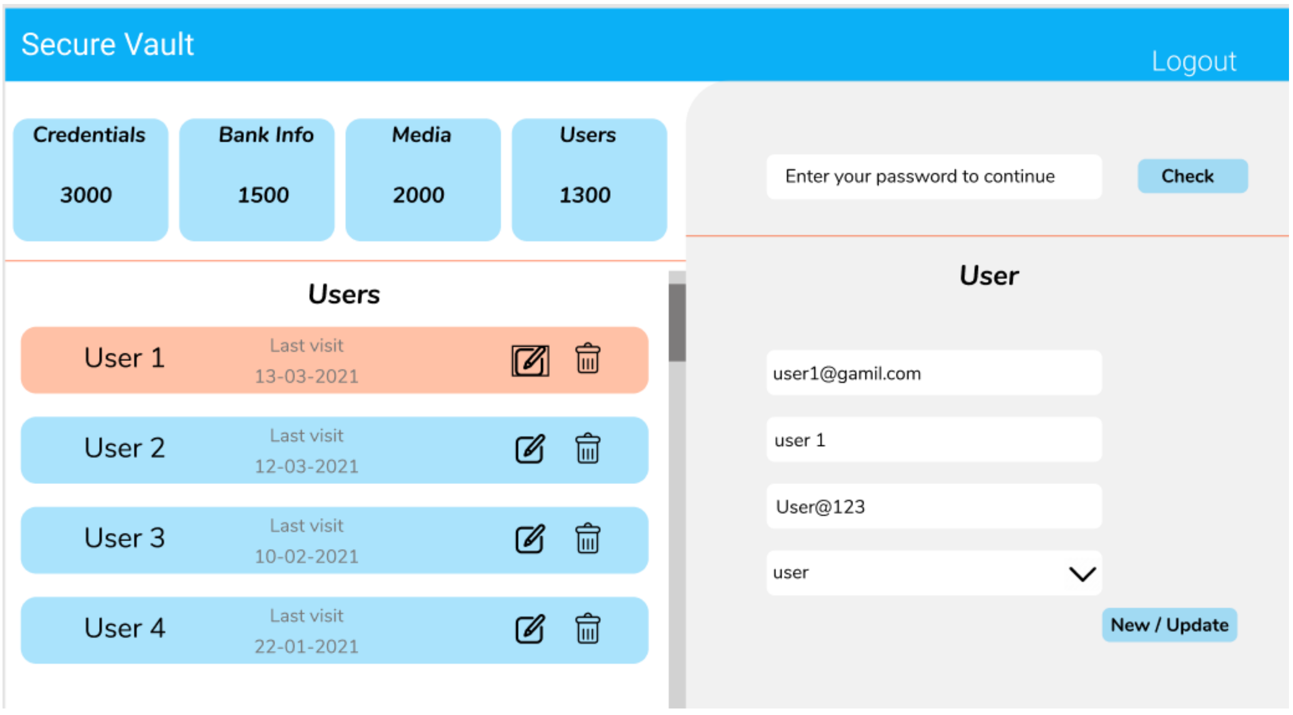
Uploaded File Preview

New / Update

Admin:

Home:

Output Screenshot:


Secure Vault Logout

Credentials	Bank Info	Media	Users
3000	1500	2000	1300

Users

User 1	Last visit 13-03-2021
User 2	Last visit 12-03-2021
User 3	Last visit 10-02-2021
User 4	Last visit 22-01-2021

Enter your password to continue — Check

User

user1@gamil.com

user 1

User@123

user

New / Update

Backend:

Class and Method description:

Model Layer:

1. UserModel: This class stores the user type (admin or the User) and all user information.
    a. Attributes:
        i. email: String
        ii. password: String
        iii. mobileNumber: String
        iv. active: Boolean
        v. role: String
    b. Methods: -
2. LoginModel: This class contains the email and password of the user.
    a. Attributes:
        i. email: String
        ii. password: String
    b. Methods: -
3. BankValutModel: This class stores the encrypted Bank information.
    a. Attributes:
        i. valutId: String
        ii. userId: UserModel
        iii. accountNumber: Long
        iv. accountName: String
        v. IFSC: String
        vi. userName: String
        vii. password: String
    b. Methods: -
4. MediaValutModel: This class stores the encrypted media information.
    a. Attributes:
        i. valutId: String
        ii. userId: UserModel
        iii. mediaName: String
        iv. image: Blob

        v. video: Blob
        vi. audio: Blob
    b. Methods: -

<u>Controller Layer:</u>

5. SignupController: This class control the user signup
    a. Attributes: -
    b. Methods:
        i. saveUser(UserModel user): This method helps to store users in the database and return true or false based on the database transaction.
6. LoginController: This class controls the user login.
    a. Attributes: -
    b. Methods:
        i. checkUser(LoginModel data): This method helps the user to sign up for the application and must return true or false
7. BankValutController: This class controls the add/edit/update/view Bank information.
    a. Attributes: -
    b. Methods:
        i. List< BankValutModel > getBankInfo(): This method helps the User to fetch their all bank information from the database.
        ii. BankValutModel bankInfoById(String id): This method helps to retrieve a Bank information from the database based on the valut id.
        iii. bankInfoEditSave(BankValutModel data): This method helps to edit a Bank information and save it to the database.
        iv. bankInfoSave(BankValutModel data): This method helps to add a new Bank information to the database.
        v. bankInfoDelete (String id): This method helps to delete a Bank information from the database.
8. MediaValutController: This class controls the add/edit/update/view Media information.
    a. Attributes: -
    b. Methods:
        i. List< MediaValutModel > getMediaInfo(): This method helps the User to fetch their all Media information from the database.
        ii. MediaValutModel mediaInfoById(String id): This method helps to retrieve a Media information from the database based on the valut id.
        iii. mediaInfoEditSave(MediaValutModel data): This method helps to edit a Media information and save it to the database.

iv. mediaInfoSave(MediaValutModel data): This method helps to add a new Media information to the database.

v. MediaInfoDelete (String id): This method helps to delete a Media information from the database