Project Documentation

Introduction:

A password management system is a system that aims to manage and secure passwords in such a way that no unauthorized user or hacker can get access to that information by any means. It will have a user-friendly interface that will help a user easily understand how to work with this software. This software will be a superior option to use because of its robust encryption and the best-protected database.

Functional Requirements:

- Allow users to login or sign up for the system first. During SignUp, a user profile will be created
- User Login and other details will be encrypted and securely stored in the system
- The user can create, update, and delete password information entries.
- Strong encryption mechanisms must be implemented for storing all the password information.

Non Functional Requirements:

- The system must be responsive, even with a large number of stored passwords.
- Maintain high-level security standards to prevent unauthorized access to the system.
- Make the UI more user friendly so that users can operate this system more easily.

Design:

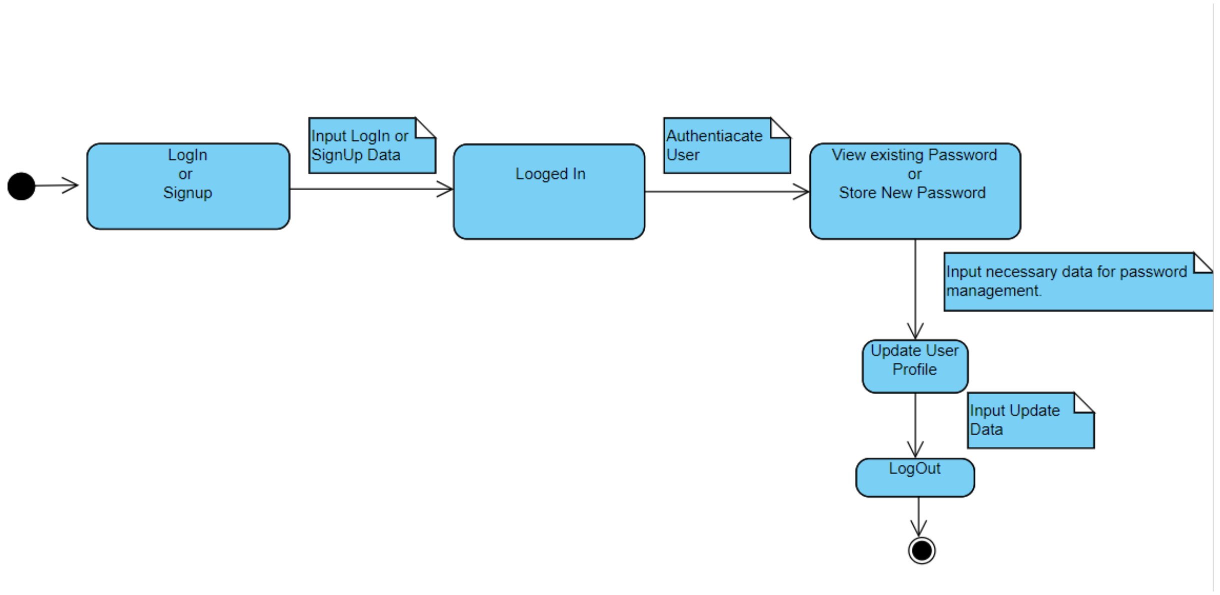
Clint Server Architecture Diagram:

```
+------------------------------+          +------------------------------+
| Desktop Clint                |          | Password Server              |
+------------------------------+          +------------------------------+
|                              |          |                              |
| GUI Interface for User       |←-------→| Database                     |
| Input and Operations         |          | (Password Storage)           |
|                              |←-------→|                              |
|                              |          |                              |
|                              |          |                              |
+------------------------------+          +------------------------------+
```

Description:

- The desktop, which is our client side, will have the user interface that the user will interact with.
- Through the GUI, users will be able to input, update, and remove data.
- Clint will interact with the password server, which maintains a database containing all of the password information.
- The database that is in charge of securely managing and saving passwords is located on this password server.
- Clint and the server will communicate with one another via a secure network.
- Clients will send requests to the server, asking it to store new password information or retrieve old passwords, among other things.
- Secure data movement and storage are ensured by the use of defined protocols to connect the client and server components. (CHATGPT, 2023)

UML State Diagram:

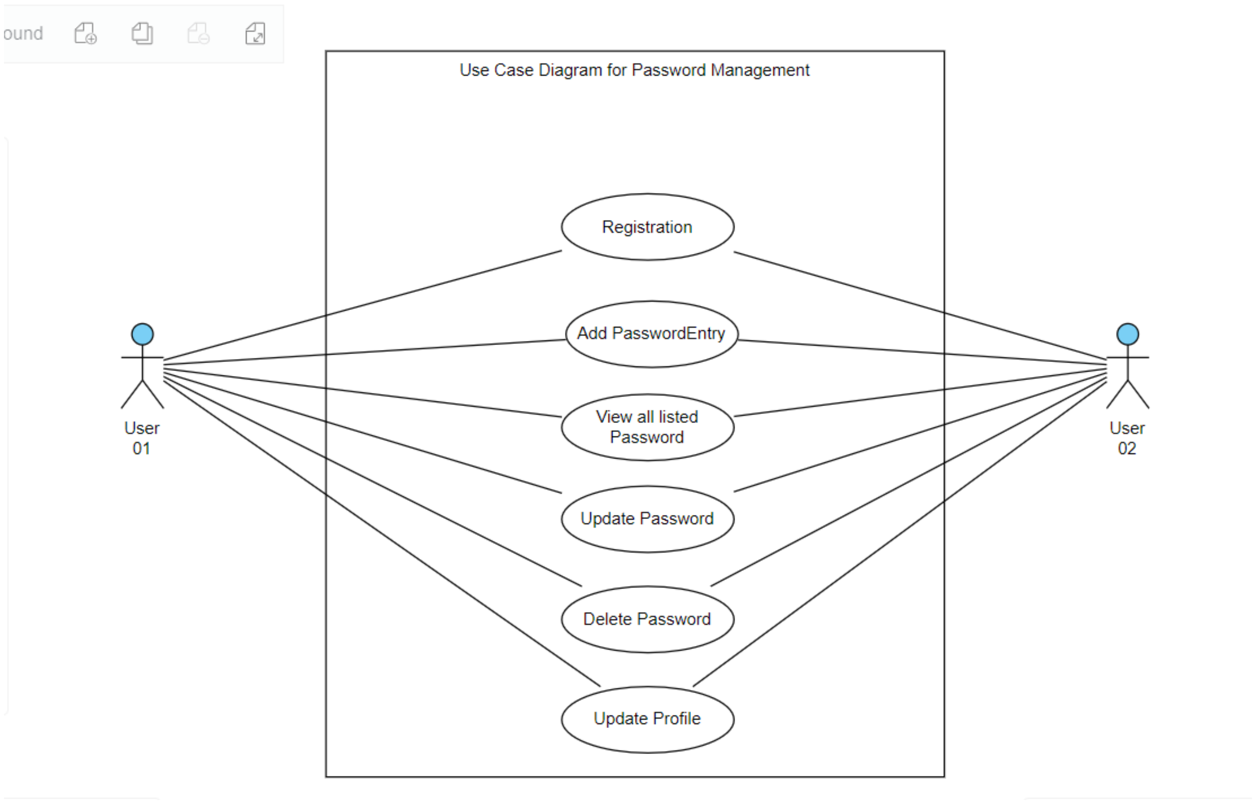

LogIn or Signup
Input LogIn or SignUp Data
Logged In
Authentiacate User
View existing Password or Store New Password
Input necessary data for password management.
Update User Profile
Input Update Data
LogOut

Description:

- The initial state, all users should need to sign-up or logged in to the system
- Transitioned state after successful login or signup where the user can perform actions like storing necessary information.
- In next state, a user can view the stored all password and it's related all information. Also a user can also store new password and its related information in this state.

- In update profile state, a user can view or update logged in details of our password management system.
- Close the current session by switching back to the user who is not logged in.

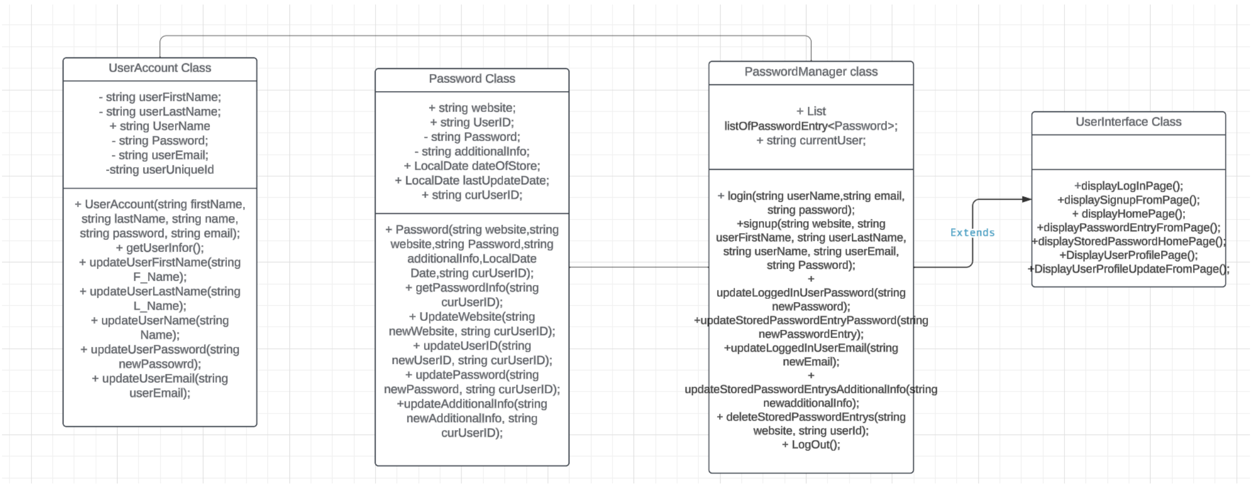
UML Use Case Diagram:



Description:

- In the above use-case diagram, there are two actors. Both are end users. Here are six use cases in this diagram. Each actor may need to register for their first log-in to the system. So they need to store their name, password, and other details in the system, and an authenticated process will authenticate them to log in to the password management software. After logging in, a user can view all the credential details of different websites and systems that they have stored in our password management system and also add new credentials to this system. At the same time, this user can perform some specific function, like updating or deleting those stored credentials on this system. Users can also update their log-in details on this system from their login screen in our password management software.

UML Class Diagram:


UserAccount Class
- string userFirstName;
- string userLastName;
+ string UserName
- string Password;
- string userEmail;
-string userUniqueId
+ UserAccount(string firstName, string lastName, string name, string password, string email);
+ getUserInfor();
+ updateUserFirstName(string F_Name);
+ updateUserLastName(string L_Name);
+ updateUserName(string Name);
+ updateUserPassword(string newPassowrd);
+ updateUserEmail(string userEmail);
Password Class
+ string website;
+ string UserID;
- string Password;
- string additionalInfo;
+ LocalDate dateOfStore;
+ LocalDate lastUpdateDate;
+ string curUserID;
+ Password(string website,string website,string Password,string additionalInfo,LocalDate Date,string curUserID);
+ getPasswordInfo(string curUserID);
+ UpdateWebsite(string newWebsite, string curUserID);
+ updateUserID(string newUserID, string curUserID);
+ updatePassword(string newPassword, string curUserID);
+updateAdditionalInfo(string newAdditionalInfo, string curUserID);
PasswordManager class
+ List listOfPasswordEntry<Password>;
+ string currentUser;
+ login(string userName,string email, string password);
+signup(string website, string userFirstName, string userLastName, string userName, string userEmail, string Password);
+ updateLoggedInUserPassword(string newPassword);
+updateStoredPasswordEntryPassword(string newPasswordEntry);
+updateLoggedInUserEmail(string newEmail);
+ updateStoredPasswordEntrysAdditionalInfo(string newadditionalInfo);
+ deleteStoredPasswordEntrys(string website, string userId);
+ LogOut();
Extends
UserInterface Class
+displayLogInPage();
+displaySignupFromPage();
+ displayHomePage();
+displayPasswordEntryFromPage();
+displayStoredPasswordHomePage();
+DisplayUserProfilePage();
+DisplayUserProfileUpdateFromPage();

Description:

- **UserAccount class**: Represent user information with attributes like userFirstName, userLastName, userName, password, userEmail, and userUniqueId. Also, some methods will manipulate those attributes.
- **Password class**: Represent password information with attributes website, UserID, Password, additionalInfo, additionalInfo, dateOfStore, lastUpdateDate, and curUserID. With those attributes, we can simply store and manage the credentials and log-in details of different systems for a user.
- **PasswordManager class**: Manage user entries and the current user with methods like LogIn, signup, updateLoggedInUserPassword, updateStoredPasswordEntryPassword, updateStoredPasswordEntrysAdditionalInfo, and deleteStoredPasswordEntrys. With those methods, all the core functionality is managed by this system.
- **UserInterface Class**: This is our GUI-controlled class that handles GUI-related functions for displaying login/signup forms, main app screens, all credentials listing home screens, credentials entry forms, and user profile screens of this system.

<u>Implementation</u>:

Tools: Java for backend, Java Swing for the GUI, Eclips, PostgreSQL.
Encryption: SHA (Secure Hash Algorithm) will be used for encrypting our password for this system.

Explanation

I shall implement the Java file structure using the Eclipse IDE. The file structure should be such that one package will have all the class code. Another package will contain the GUI Java class. I shall also try to maintain a separate package where all the database-related connections will be kept in a Java file. First, I will implement the classes that we need for this project. Then do the GUI design. After that, I will connect all the Java classes to this GUI. Last, I shall write all the data retrieval codes in a separate Java class. Then I will add an object of the Data Retrieve class to our Java main class for storing and deleting data from the database. Next, I shall implement an encryption algorithm to securely store and retrieve data from the database. For this, I shall implement a hashing algorithm for hashing passwords and other important data that we want to store in our database. Here, the MD5 hash algorithm will be implemented. MD5 (Message Digest Method 5) is a cryptographic hash algorithm used to generate a 128-bit digest from a string of any length. It represents the digests as 32-digit hexadecimal numbers. Ronald Rivets designed this algorithm in 1991 to provide the means for digital signature verification. (M, 2023)

Testing:

During developing this software, two types of testing will be conducted. Firstly, Unite testing that will cover the individual functionalities during its development. Secondly integration testing that will ensure seamless interaction between GUI and backend.

References

CHATGPT, 2023. *chat.openai.* [Online]
Available at: https://chat.openai.com/
M, S., 2023. *Cyber Security Tutorial: A Step-by-Step Guide.* [Online]
Available at: https://www.simplilearn.com/tutorials/cyber-security-tutorial/md5-algorithm#:~:text=MD5%20(Message%20Digest%20Method%205,means%20for%20digital%20signature%20verification.
[Accessed 23 12 2023].