Term Project: MyPass

CIS 476 -001

By: Athira Nakkara Radhakrishnan

Presentation video link: <https://youtu.be/9jGkLTTN7Sk>

GitHub: <https://github.com/AthiraCode/TP476.git>

**Implementation**

* User Authentication and Encryption:

A screenshot of a computer

Description automatically generated

UML Description:

The provided class diagram illustrates a User Session class that implements the Singleton pattern for secure session handling within an application. By using the Singleton pattern, it ensures only one instance of the User Session exists, offering a single point of access. This class handles the initiation of sessions, user authentication, and session validation. The private constructor prevents external instantiation, while the static getInstance() method provides controlled access to the session, ensuring both thread safety and consistency. The login() and signUpLogin() methods handle user credentials, while validation() ensures the integrity of session data. This design centralizes session management, providing both security and simplicity by maintaining consistent handling of user interactions throughout the application.

* Password Storage and Management:

A diagram of a server

Description automatically generated

UML Description:

The password storage system employs the Observer Design Pattern to manage passwords dynamically. The main component (ConcreteSubject) monitors password changes and notifies designated observers, such as the PasswordStrengthObserver and ExpirationObserver, about these updates. Each observer then independently evaluates and adjusts specific aspects, such as password strength and expiration, ensuring the system remains responsive and efficient in managing password changes.

* User Interface and Interaction:

A diagram of a login page

Description automatically generated A diagram of a signup

Description automatically generated

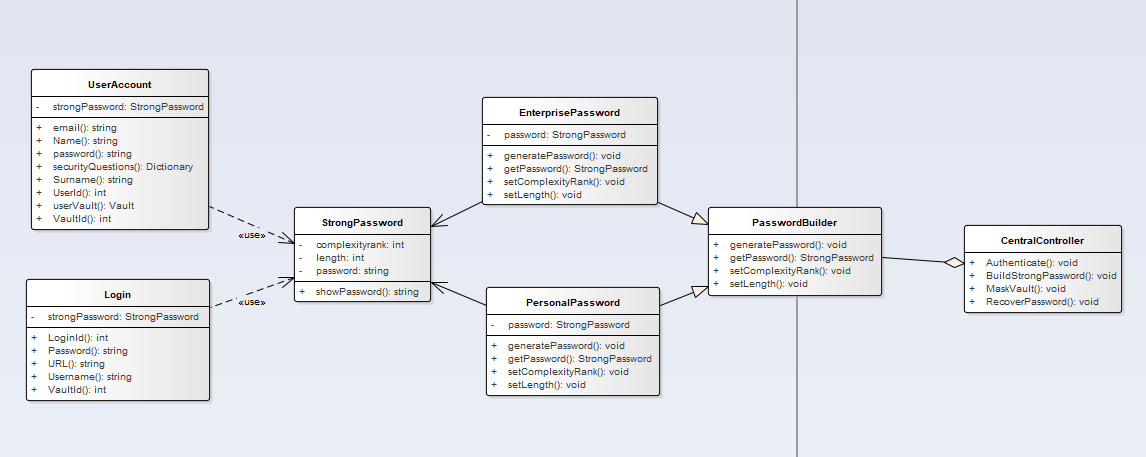
A diagram of a software

Description automatically generated

UML Description:

The UML diagrams demonstrate the use of the Mediator design pattern to facilitate communication between UI components, such as text fields and buttons, across different pages. The Mediator class manages interactions, minimizing direct dependencies between components and simplifying the handling of user inputs and system responses. This pattern is applied on each page to ensure efficient and maintainable communication between components.

* Password Generation:



UML Description:

The Password Generation classes are designed using the builder pattern. In this setup, the PasswordBuilder class includes methods like setLength (defines the password length), setComplexityRank (indicates the desired complexity, with score 1 for simple and score 2 for advanced), and generatePassword (creates the password based on the length and complexity). There are two types of passwords: Enterprise and Personal. Enterprise passwords are 16 characters long with a complexity score of 2 (advanced), while personal passwords are 8 characters with a complexity score of 1 (basic). The StrongPassword object is used for password generation. The process is initiated by the CentralController, and the resulting password string is returned to be assigned to the user account from the database.

* Data Mask and Unmask:

A diagram of a computer

Description automatically generated

UML Description:

The masking process is implemented using the proxy pattern. The IMask interface defines the MaskOn and MaskOff methods. The RealMask class holds the default mask values, with the initial MaskOff set to false and MaskOn set to true (indicating that vaults are initialized without a mask). The MaskProxy class is the sole handler of data from the CentralController. It determines whether the vaults should be masked or unmasked and interacts with the RealMask class when necessary. The process begins with the CentralController triggering the change in mask status. Once completed, the mask value is applied to the vault, and EF Core updates the vault information in the database.

* Maste Password Recovery:

A diagram of a recovery

Description automatically generated

UML Description:

The Password Recovery Process follows the Chain of Responsibility Pattern. The RecoverPassword method is abstract and implemented in each question class. The RecoveryAssistant class first sets the question using the SetQuestion method, then triggers the RecoverPassword methods in each question class. Each question class retrieves data from the database to present questions to the user. As the user answers, hash tables are used to match the answer with the correct question. If the answer matches a question, the method returns true. Once at least three true values are obtained, the RecoveryAssistant resets the password to "12345" and advises the user to create a master password for future logins. The user can then use the generated simple password to access their account. The entire process is managed by the CentralController, which is invoked by the form class.

**Database Schema**

**users Table**

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

email VARCHAR(255) NOT NULL UNIQUE,

master\_password\_hash VARCHAR(255) NOT NULL,

security\_question\_1 TEXT NOT NULL,

security\_question\_2 TEXT NOT NULL,

security\_question\_3 TEXT NOT NULL,

security\_answer\_1\_hash VARCHAR(255) NOT NULL,

security\_answer\_2\_hash VARCHAR(255) NOT NULL,

security\_answer\_3\_hash VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Vault\_items Table**

CREATE TABLE vault (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

item\_type ENUM('Login', 'Credit Card', 'Identity', 'Secure Note') NOT NULL,

data TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

**clipboard\_logs Table**

CREATE TABLE clipboard\_logs (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

data\_type ENUM('Login', 'Credit Card', 'Identity', 'Secure Notes') NOT NULL,

data\_value TEXT NOT NULL,

copied\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

expiration\_time TIMESTAMP DEFAULT NULL,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

**activity\_logs Table**

CREATE TABLE activity\_logs (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

last\_activity TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

**Screenshots of application**

* + Login page

A screenshot of a login screen

Description automatically generated

* + Register page

A screenshot of a login form

Description automatically generated

UI Mockups

A screenshot of a computer

Description automatically generated

A screenshot of a login screen

Description automatically generated