

CIS 476 Term Project

MyPass Password Manager

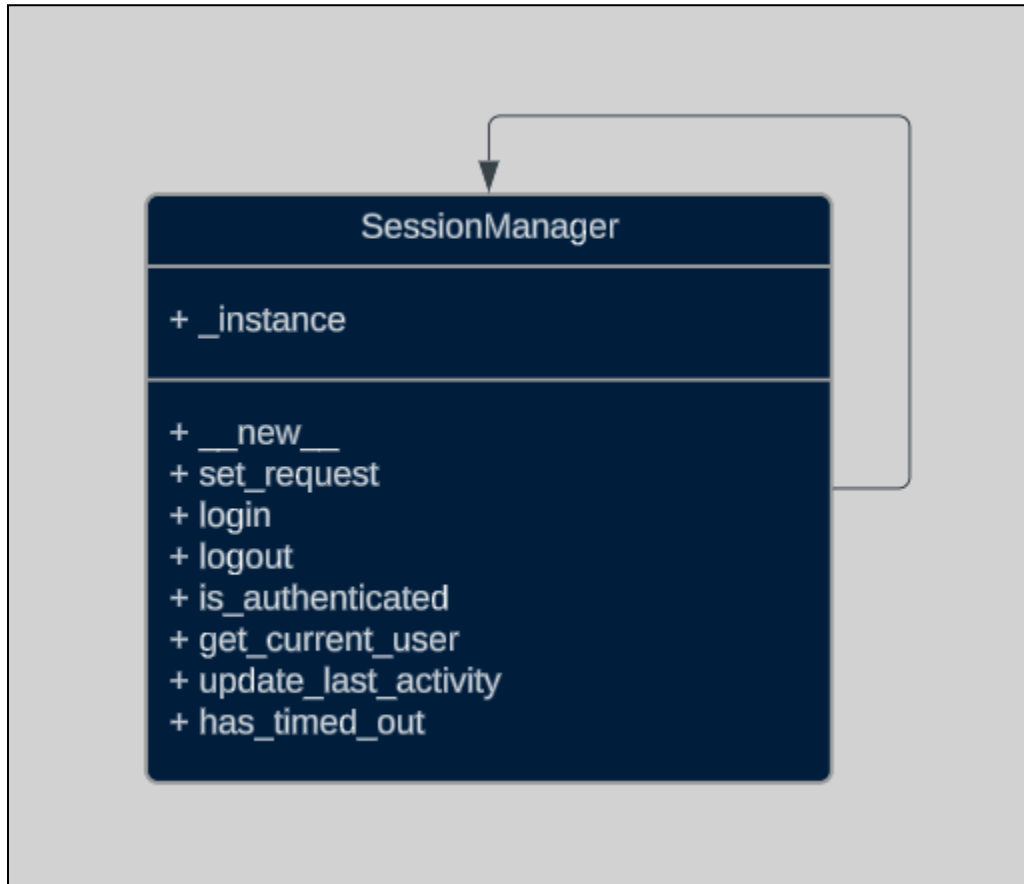


Nicholas Harrison, Yuliya Wickens, Michael Sepsey

Fall 2024

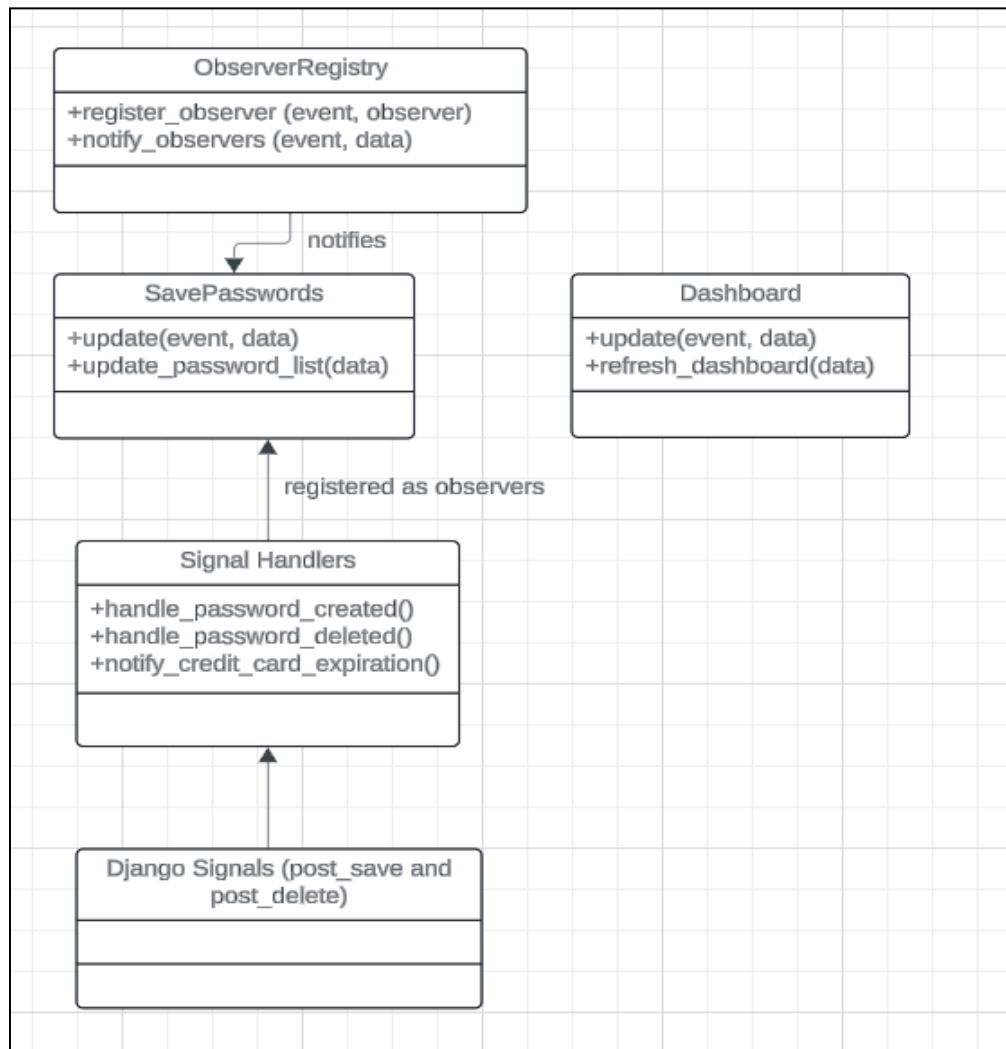
Design Pattern Class Diagrams

Singleton:



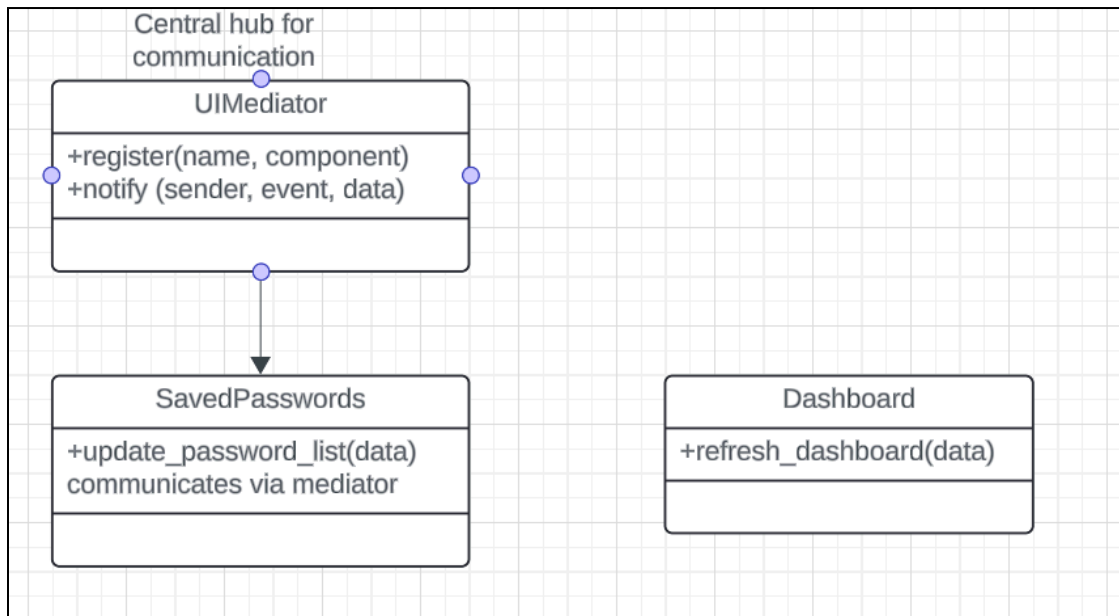
This Singleton design pattern class ensures that only one instance of the session manager exists throughout the life cycle of a user being logged into the application. This is achieved through the `__new__` method which checks to see if there's an instance that exists and either returns the existing instance if it does exist or creates a new one if there is not an instance that exists. This prevents the creation of multiple instances making sure that only one session can be active at a time. This class also manages the logging in and logging out of specific users and authenticating them.

Observer:



1. ObserverRegistry:
 - Registers observers like SavedPasswords and Dashboard.
 - Notifies observers of events triggered by signal handlers.
2. Observers:
 - SavedPasswords and Dashboard implement the `update(event, data)` method to handle specific events.
3. Signals:
 - Trigger events such as `password_created` or `password_deleted`.
 - Signal handlers use ObserverRegistry to notify registered observers.

Mediator:



- **UIMediator:**
 - Acts as the communication hub between components like SavedPasswords and Dashboard.
 - Handles interactions via register() and notify() methods.
- **Components:**
 - SavedPasswords and Dashboard rely on the mediator to coordinate their actions.
 - They are loosely coupled and do not communicate directly with each other.

Differences between the two patterns used in the app:

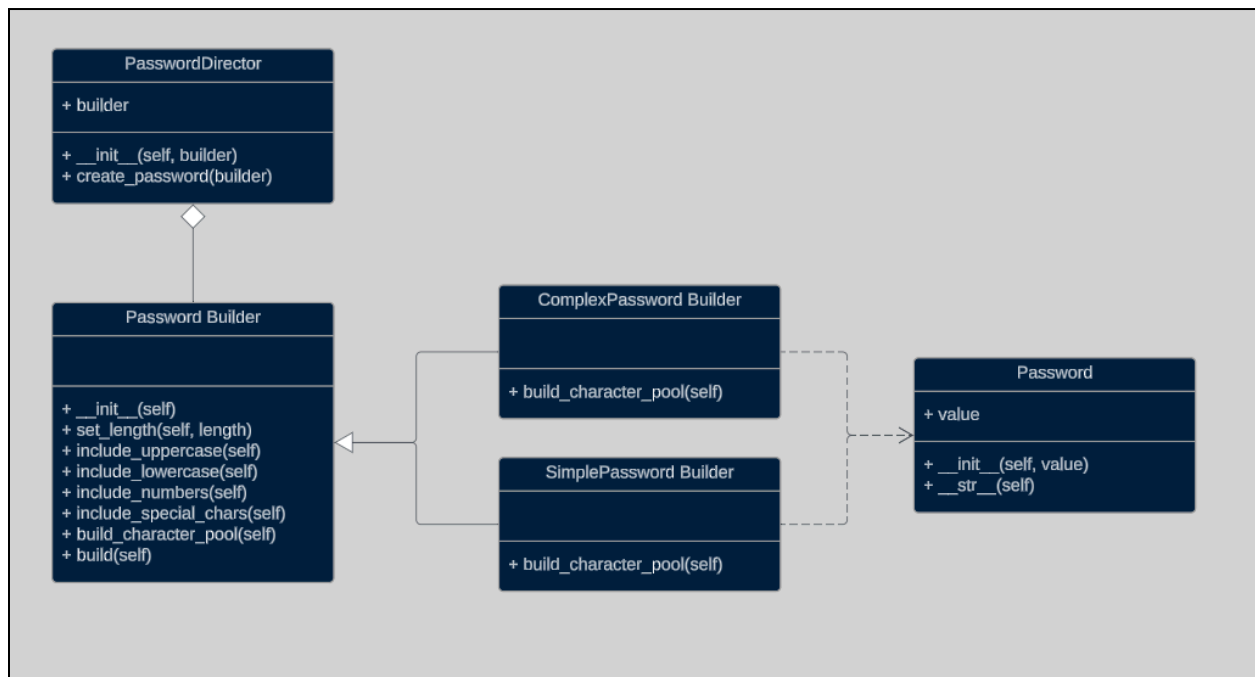
Observer:

It focuses on event-based communication, uses ObserverRegistry to notify observers of system-wide events like password_created, and acts independently based on the events they are registered for.

Mediator:

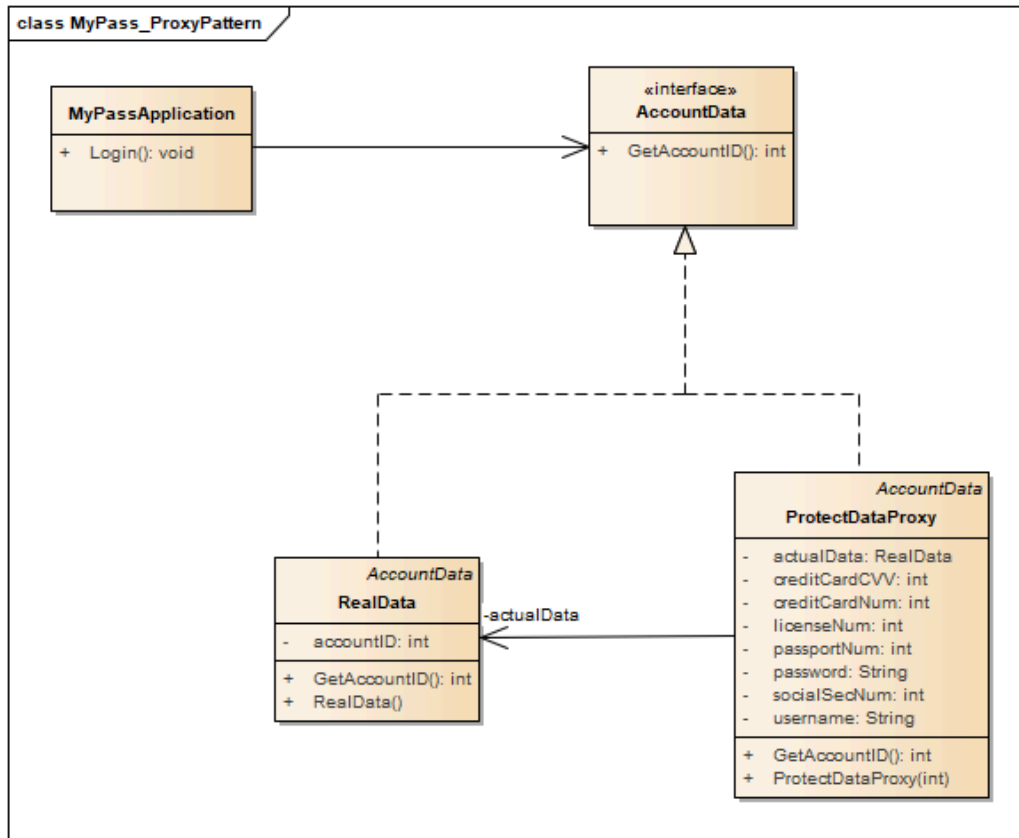
It focuses on component coordination. The mediator, UIMediator, manages interactions between components (SavedPasswords, Dashboard). The components are unaware of each other and rely on the mediator for communication.

Builder:



This builder pattern is used to generate passwords with different levels of complexity. The class called **PasswordBuilder** is the abstract class providing methods to set parameters like length and whether to include uppercase, lowercase, numbers, or special characters. The actual implementation of how the characters are selected is handled by the two subclasses which are called **SimplePasswordBuilder** and **ComplexPasswordBuilder** and they define specific character pools based on the complexity desired. The password director class manages the entire building process and ensures that the password is created with the appropriate concrete builder. In the process of adding a new password to the password manager vault, the user is able to select whether they want a simple or complex password. From there the builder is selected based on their choice, then a director is created with the specific concrete builder, and finally, the password is created by calling `create_password` with the director who is associated with the specific concrete builder.

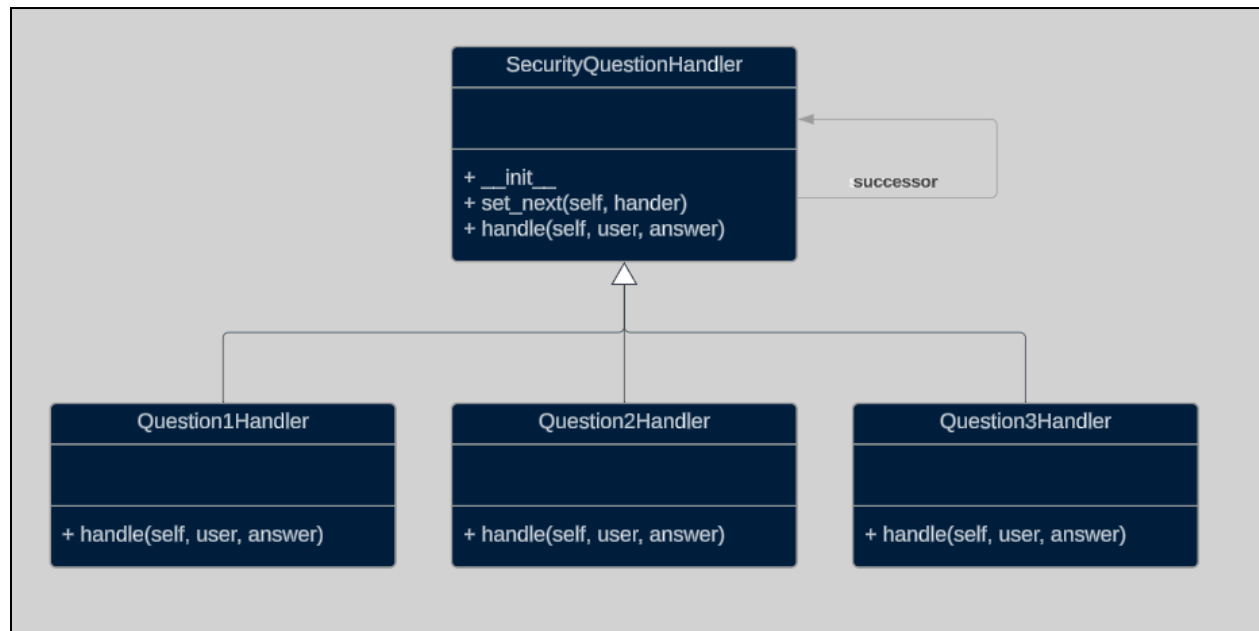
Proxy:



The proxy pattern is used to mask sensitive information (i.e. login information, passwords, passport numbers, social security numbers, and credit card information) from the user's view. By default, the `ProtectDataProxy` class will be invoked by the `AccountData` interface with all sensitive information masked (hidden from view). When the user clicks on a button, it will invoke the `RealData` class that will unmask the sensitive information for a brief moment (for security reasons). Users will be able to (within that moment) copy the information to their clipboards to paste on other websites requiring that information. Unmasking sensitive information requires the account ID (basically whoever is logged in at the moment) to display the correct information, not information from someone else on the system.

For the proxy pattern to be invoked, the user must be logged on to the `MyPassApplication` website and be situated on a webpage where the sensitive information is stored (i.e. Vault).

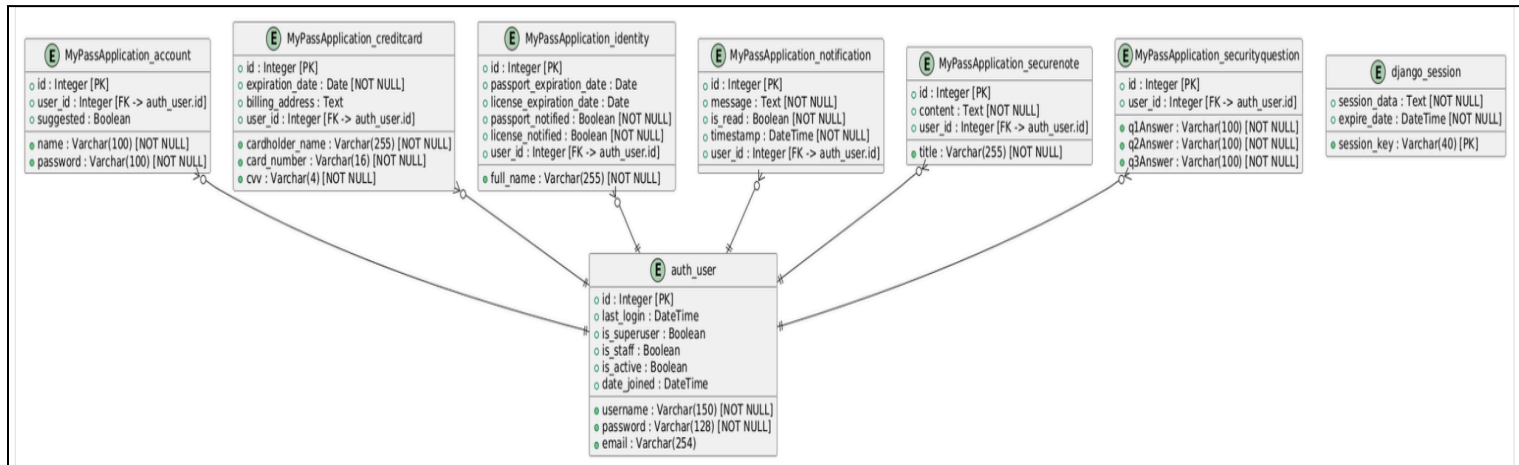
Chain of Responsibility:



This chain of responsibility pattern is used to handle security question handlers. There is a base class that includes the `set_next` function as well as the `handle` function. From there the sub-classes specify how to handle each individual request, which in this case is individual security questions. When the program is running, three handlers are created and a chain is created from questions one to two to three. Each specific handler has a question as well as a corresponding answer. The `handle` function will ensure that the answer given matches the answer that is stored for the user for that specific question. After that is handled properly the next handler can handle the next request because of the chain that was created. Once the final handler has handled its request To verify that the given answer matches the correct answer, the chain is complete.

Database Schema

Structure from SQLite:



Explanation:

1. **Entities:** Each table is represented as an entity.
2. **Attributes:** Attributes are listed with their name, data type, and constraints (e.g., [PK] for primary key, [NOT NULL] for non-nullable).
3. **Relationships:** Foreign key constraints are shown using lines between entities.
 - o || represents a many-to-one relationship (e.g., multiple accounts belong to one user).

User Interface Screenshots

Account Creation

Create an Account

Username:

Email:

Password:

Password confirmation:

What is your favorite color?

What city were you born in?

What is the name of your first employer?

Already have an account? [Login here](#)

Login

• Your account has been locked due to inactivity.

Username or Email:

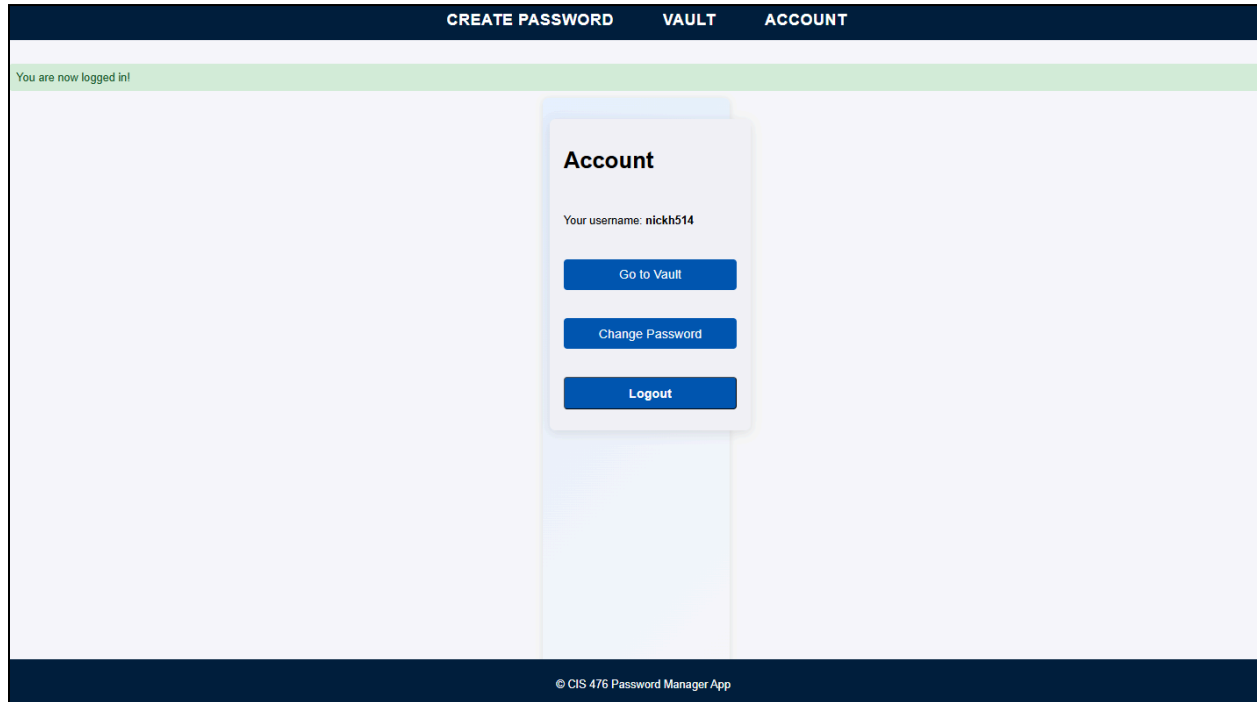
Password:

Don't have an account? [Register here](#)

[Forgot your password?](#)

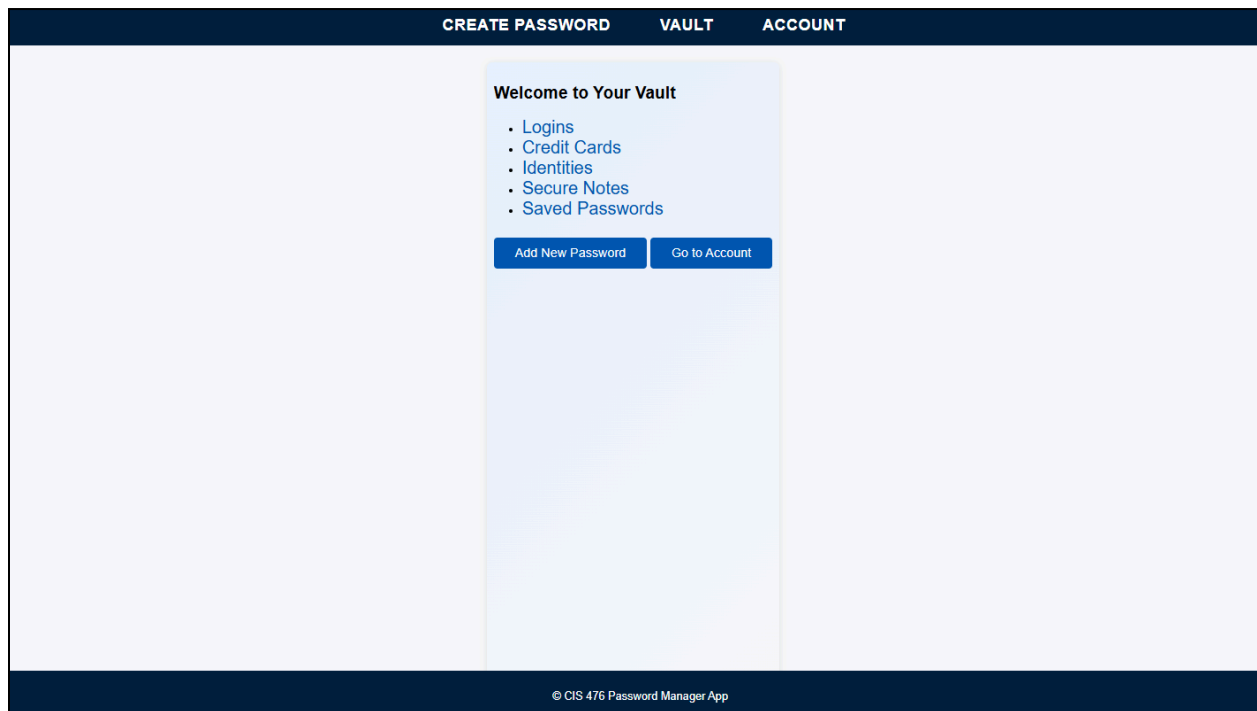
The account creation page enables users to set up their accounts by providing a username, password, and answers to three security questions. This form will validate the user's inputs and ensure no empty answers and a secure password is provided. The submitted data will store the account with the security questions in the database. Security questions are stored so that later on if the user forgets their password, they will be able to recover their account by using the answer to the security questions. This session management is implemented for user and account security such as an automatic logout mechanism which is triggered after one minute of inactivity.

Account Page



After a successful login, the users will be directed to their account page. This page retrieves the user's information and gives options such as accessing the vault, changing their password, or logging out. Since the user is authenticated via their session token, security questions are not required for the changing of their password. This page serves as the central point for navigating the app's main features.

Vault Page



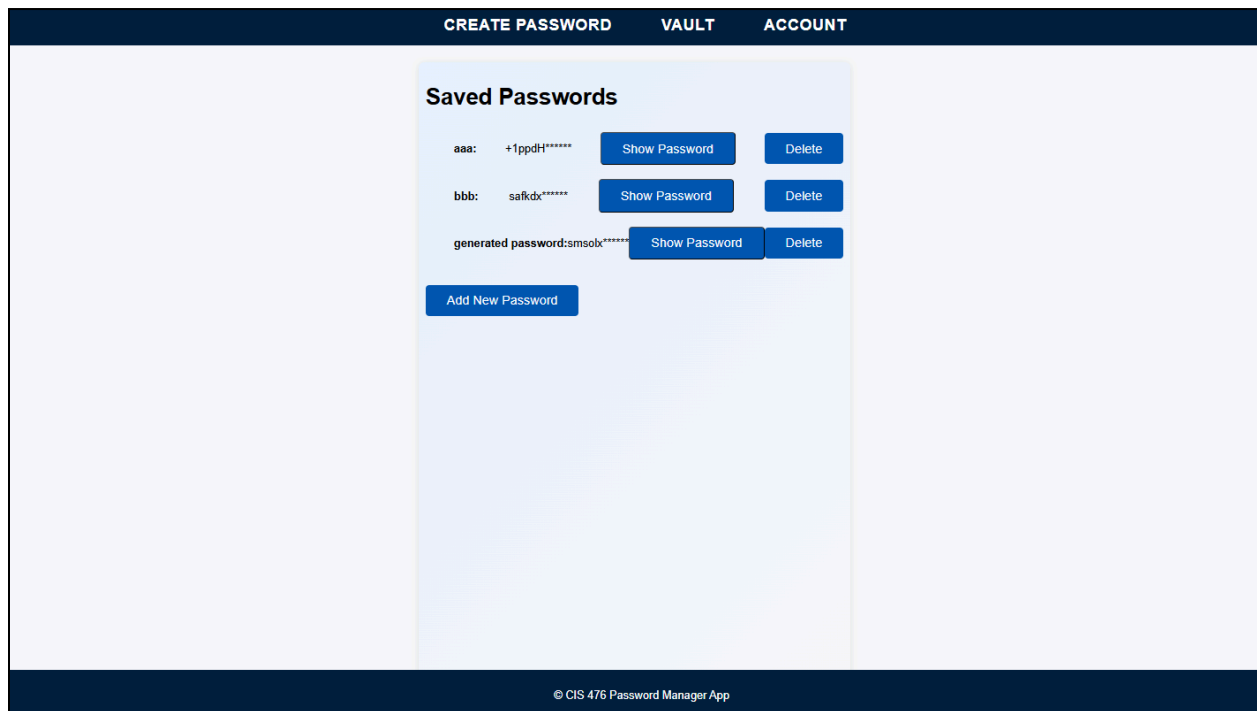
The vault page is the main hub for storing and managing sensitive user data such as passwords, logins, identities, and notes. This is broken up into specific categories which are stored using different models and data types for the specific data being stored. Each item in the vault will have the option to view, edit, or delete it. Creation of new entries is handled through forms which will save the data to the database to be later viewed in the vault.

Password Generator Page

The screenshot shows a web application interface for a password generator. At the top, there is a dark blue navigation bar with three white text links: "CREATE PASSWORD", "VAULT", and "ACCOUNT". The main content area has a light blue background. In the center, there is a white card with a light blue header that reads "Create a New Password". Below the header, the card contains the following elements: a text input field labeled "Account Name:"; a dropdown menu labeled "Select Password Complexity:" with "Simple" selected; a text input field labeled "Or enter your own password:" with the placeholder text "Enter your password"; a checkbox labeled "Save this password in my Vault"; and a blue button labeled "Generate and Save Password". At the bottom of the page, there is a dark blue footer bar with the text "© CIS 476 Password Manager App" in white.

The password generator page offers functionality to the user to create secure passwords. Users can manually enter their desired password or select either a simple or complex generated password using the builder pattern. If the check box is selected to save the password to the vault, the password is encrypted and stored in the database.

Saved Passwords Page



The saved password page displays passwords in a masked format. The page will generate a table of all of the passwords with their correct title, with options to unmask or delete each individual password. These passwords are stored securely using encryption. Adding new passwords will allow the user to manually enter a password or generate a password as seen on the page previous.

Forgot Password Chain of Responsibility

Forgot Password

What is your favorite color?

Enter your answer

Submit

Forgot Password

- Correct! Next question.

What city were you born in?

Enter your answer

Submit

Forgot Password

- Correct! Next question.

What is the name of your first employer?

Enter your answer

Submit

Password Reset

Username:

New Password:

Reset Password

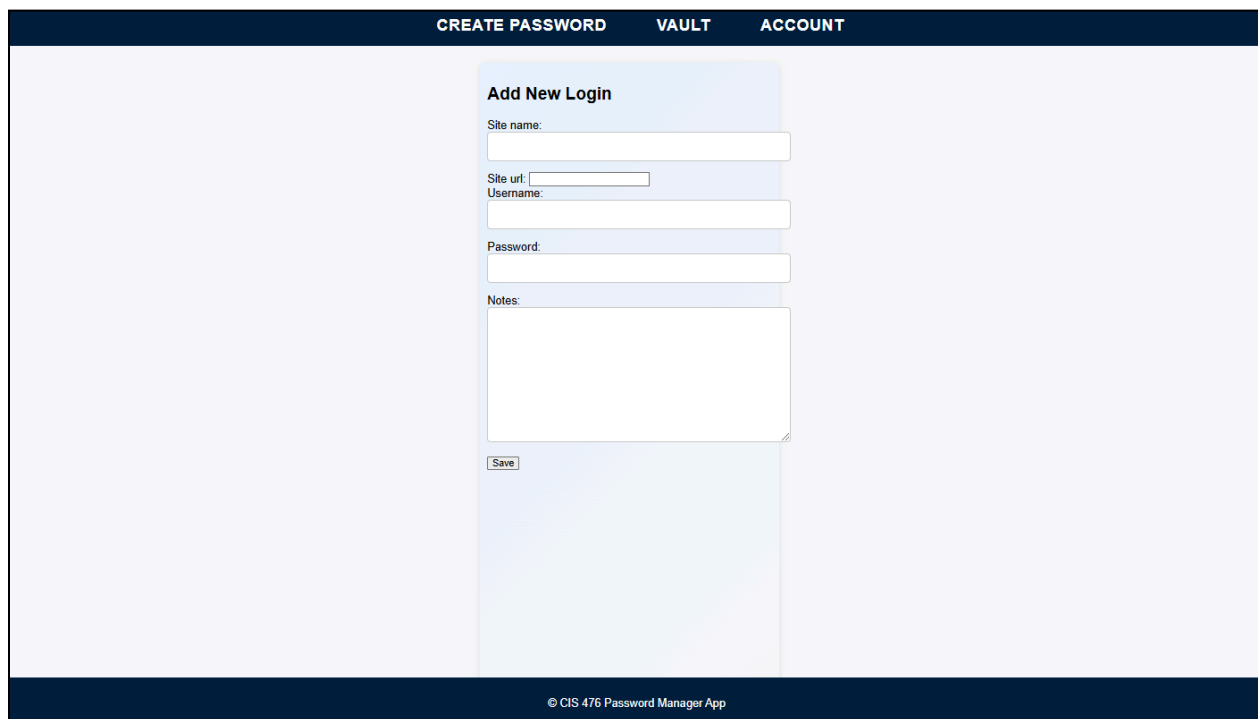
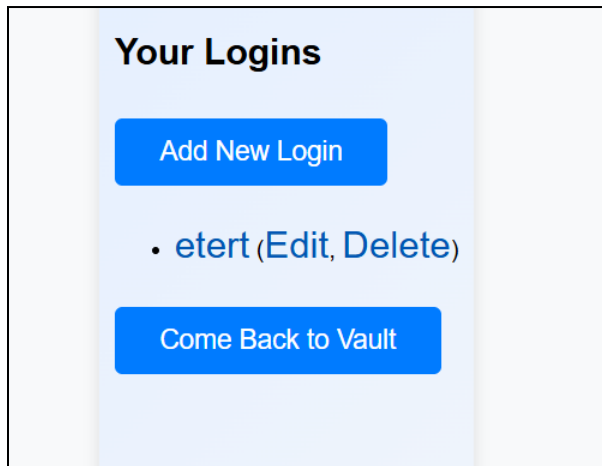
- All answers correct! You can now reset your password.

This series of pages represents the chain of responsibility pattern in the code in the case that the user forgets their password while not logged in. The user will click the forgot password button, enter their username, and then be prompted to answer their 3 security questions in order. Each security question must be answered before moving on to the next question. Once all the questions are answered correctly, the user can enter their username again and enter a new password for their account.

The app does store passwords as encrypted data in DB:

id	name	password	user_id	suggested	
Fil...	Filter	Filter	Filter	Filter	
5	Yulenska152007	gAAAAABnT3mVDSxKtMfRbqbbXwa2o8bYXFbh...	1	1	
6	Yulenska152007	gAAAAABnT4OUyg93XRRRgdW39ry-...	1	1	
7	Yulenska152007	gAAAAABnT4OzOIeDFUDvTzMazrXh8_8_C0BW...	1	1	

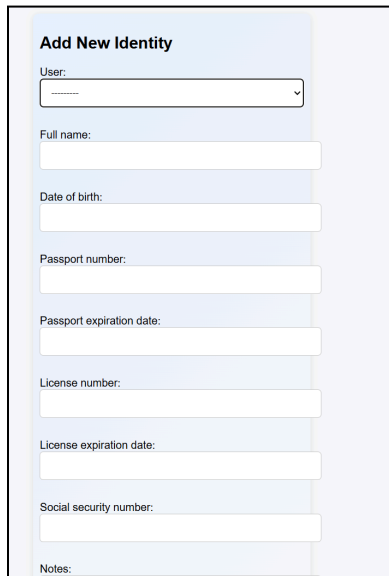
Add New Login Page



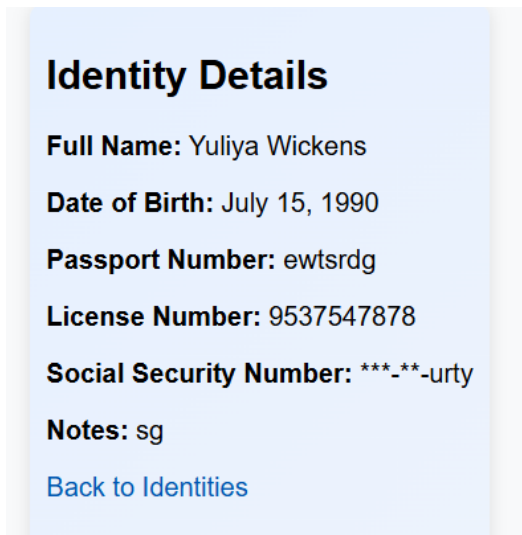
A screenshot of the 'Add New Login' form within a web application. The form is titled 'Add New Login' and is set against a light blue background. It includes the following fields: 'Site name:' with a text input, 'Site url:' with a text input, 'Username:' with a text input, 'Password:' with a text input, and 'Notes:' with a larger text area. A 'Save' button is located at the bottom of the form. The form is part of a larger interface with a dark blue header containing the links 'CREATE PASSWORD', 'VAULT', and 'ACCOUNT'. The footer of the page displays the copyright notice '© CIS 476 Password Manager App'.

The option to add new data to the vault for any specific data type has a page that looks very similar to this. This form will collect input fields specific to each data type. Submission of this form the data is stored in the database for the specific user. These data entries are displayed on the vault page for their specific data type with options to view, edit, or delete the entries. This is the main functionality of this password manager application.

Similarly, it works for credit cards, identities, secure notes, and logins.



A form titled "Add New Identity" with a light blue header. It contains several input fields: a dropdown menu for "User:", text boxes for "Full name:", "Date of birth:", "Passport number:", "Passport expiration date:", "License number:", "License expiration date:", and "Social security number:". A "Notes:" label is at the bottom without an input field.



A card titled "Identity Details" with a light blue background. It displays the following information: "Full Name: Yuliya Wickens", "Date of Birth: July 15, 1990", "Passport Number: ewtsrdg", "License Number: 9537547878", "Social Security Number: ***-**-urty", and "Notes: sg". At the bottom is a blue link "Back to Identities".

Part of the social security number is hidden.