# Term Project: MyPass

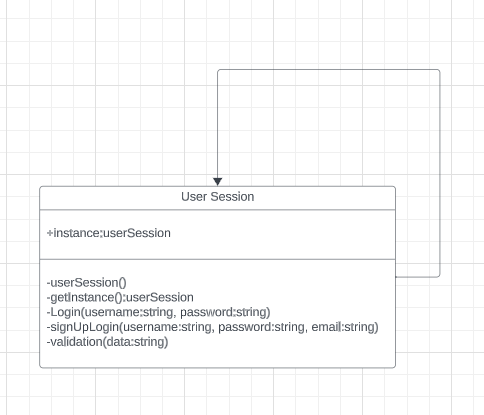
# CIS 476 - 001

By: Mohamed Muhsin , Yehya altamimi , Deniz Acikbas

# 

# Implementation

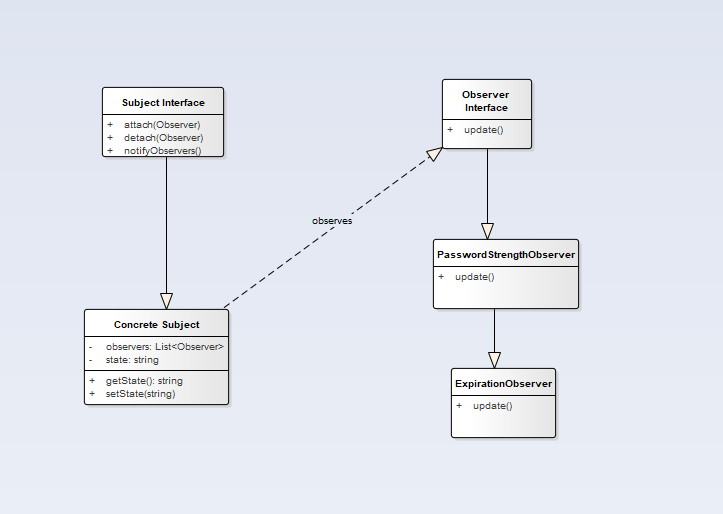
* User Authentication and Encryption:



UML Description :

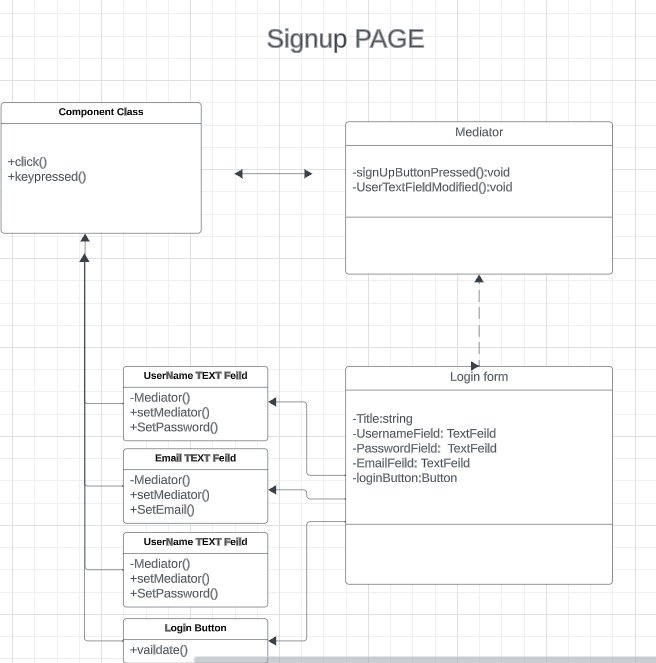
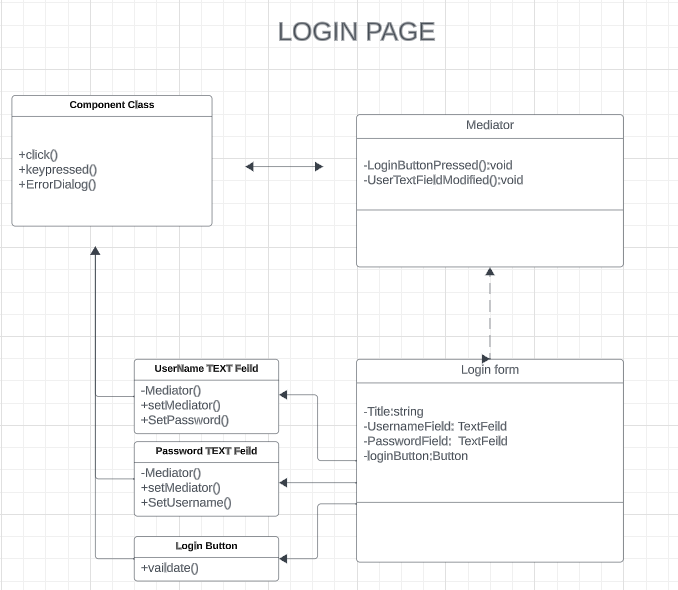
The class diagram provided outlines a User Session class that enforces the Singleton pattern for secure session management in an application. The Singleton pattern ensures a single User Session instance exists, providing a unified point of access. This class controls session initiation, user authentication, and session validation. The private constructor restricts instantiation to within the class itself, while the static getInstance() method manages controlled access to the session, ensuring thread safety and consistency. The login() and signUpLogin() methods manage user credentials, while validation() checks the integrity of the session data. This design centralizes session management, offering both security and simplicity by ensuring that user interactions are managed consistently across the application.

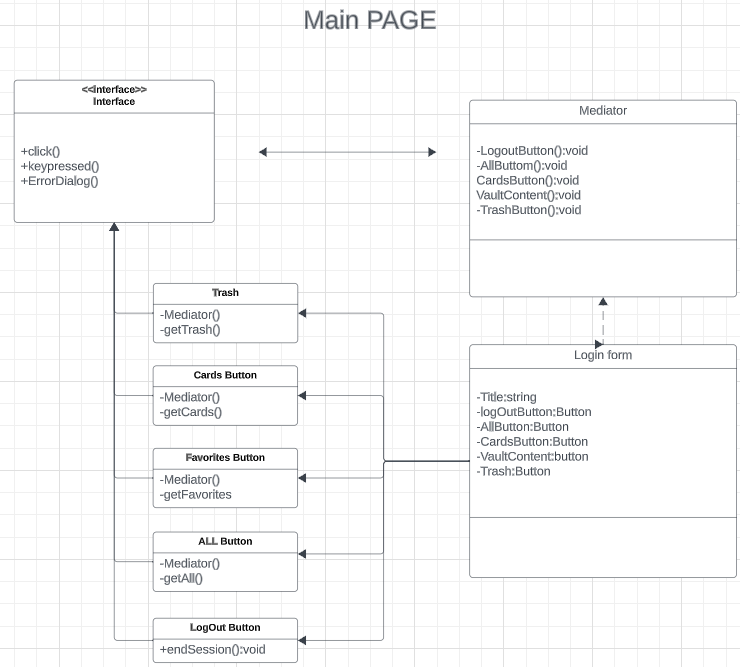
* Password Storage and Management:



UML Description :

In the password storage system, the Observer Design Pattern is used for responsive password management. The central component (`ConcreteSubject`) tracks password changes and notifies specific observers (`PasswordStrengthObserver` and `ExpirationObserver`) about these updates. Each observer then independently assesses and updates relevant aspects like password strength and expiration, ensuring the system's responsiveness and efficiency in handling password-related changes.

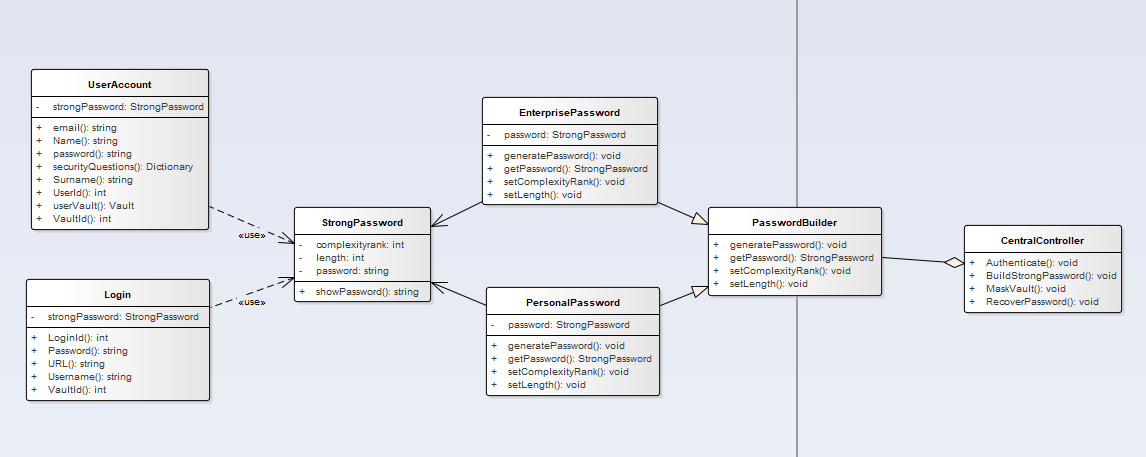
* User Interface and Interaction:



UML Description :

The UML diagrams show the Mediator design pattern , which helps communication between UI components like text fields and buttons across various pages. The Mediator class handles interactions and reduces direct dependencies among components, streamlining the management of user inputs and system responses. Each page utilizes this pattern for efficient and maintainable component communication.

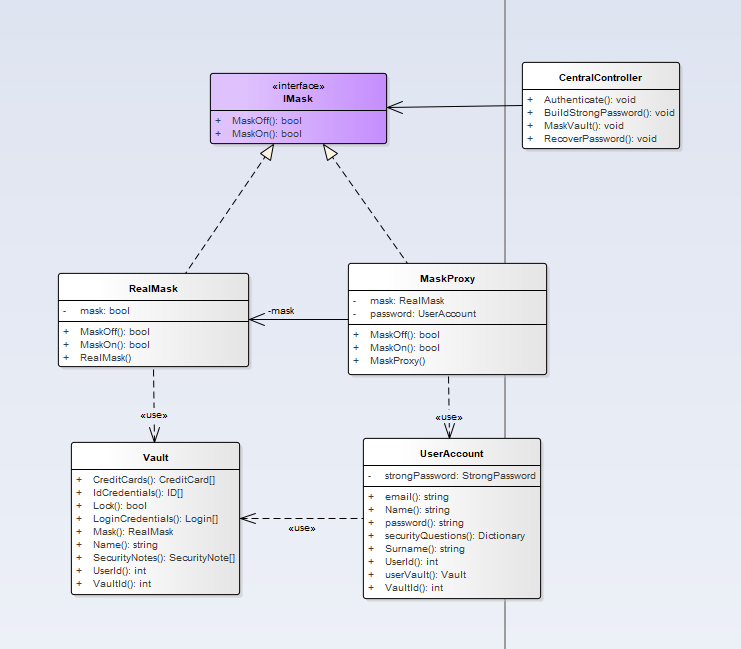
* Password Generation



UML Description :

The Password Generation classes are built using a builder pattern. In this diagram, the builder class is PasswordBuilder which contains setLength (represents the length of the password), setComplexityRank (tells how complex the password should be, which score 1 is simple complexity and score 2 is the advanced complexity), generatePassword method generates the password based on the length and complexity rank. There are two types of passwords: Enterprise and Personal Passwords. Enterprise passwords have 16 characters with score 2 complexity (advanced) whereas personal passwords have 8 characters and score 1 complexity (basic). The strong password is the object type which will be used for password generation. The process shall be initiated by CentralController and the entire process will return a string to be assigned to the user account received from the database.

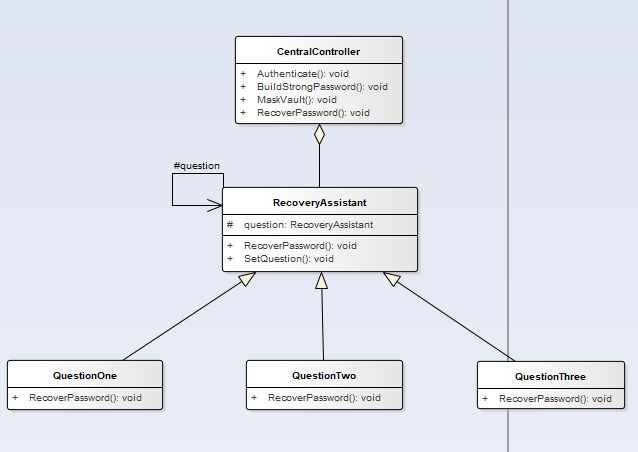
* Data Mask and Unmask:



UML Description :

Masking process is designed using a proxy pattern. IMask is the interface which contains MaskOn and MaskOff methods. RealMask contains the default values that need to be assigned to the mask. In RealMask, the initial MaskOff value is false and MaskOn value is true (This is because the vaults are initialized without a mask). MaskProxy is the only class that handles the data received from the CentralController. MaskProxy determines whether the vaults should be masked or unmasked and calls the RealMask class when necessary. This process is initiated by CentralController to change the mask status. Once the process is finished, it assigns mask value to the vault. Then EF Core updates the vault value on the database.

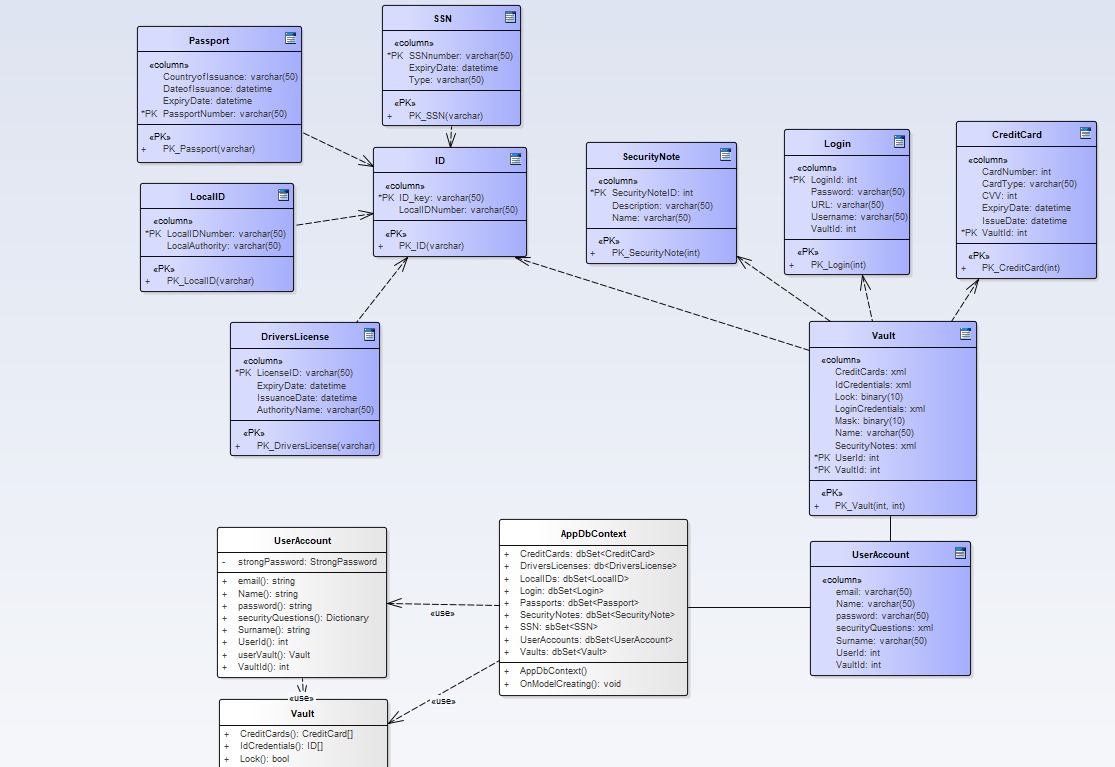
* Master Password Recovery:

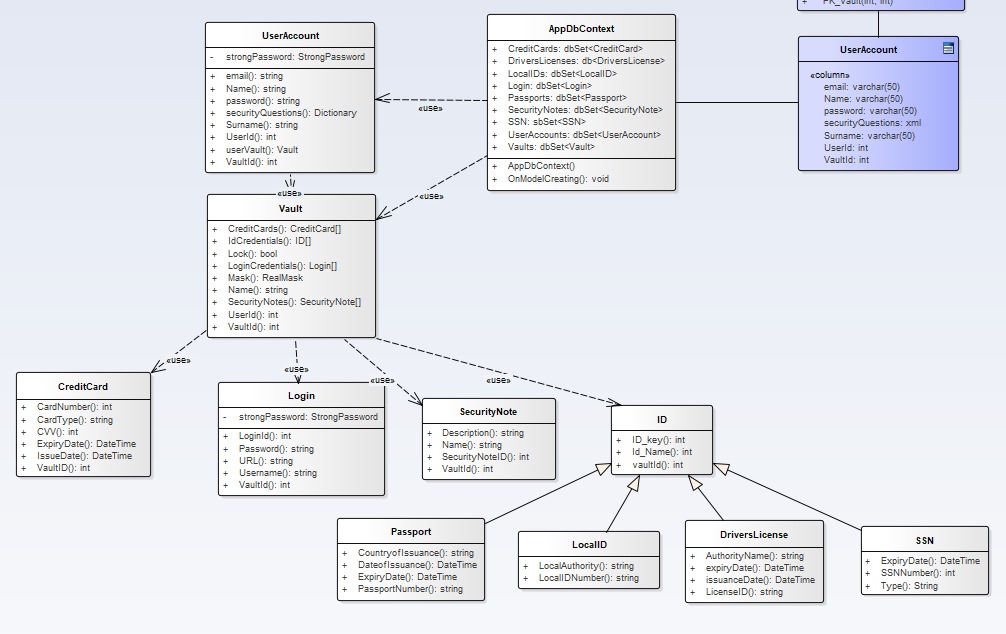


UML Description :

The Password Recovery Process is based on the Chain of Responsibility Pattern. RecoverPassword is an abstract method which will be implemented in each question class. The RecoveryAssistant class first sets the question using the SetQuestion method, then triggers RecoverPassword methods in each question class. Each question class retrieves data from the database to ask questions to the user. As the user enters the answer, it uses hash tables to access the question. If the question is being found based on the user answer, the return value will be true. If at least three true values are received, RecoveryAssistant resets the password to “12345” and prompts the user to generate a master password in the future. The user enters the generated simple password to login to their account. This process is entirely handled by the CentralController which is called by the form class.

# DataBase ER diagram



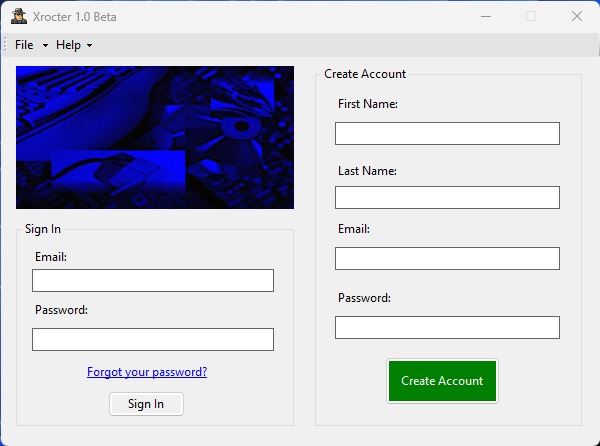


Database ER Diagram Description :

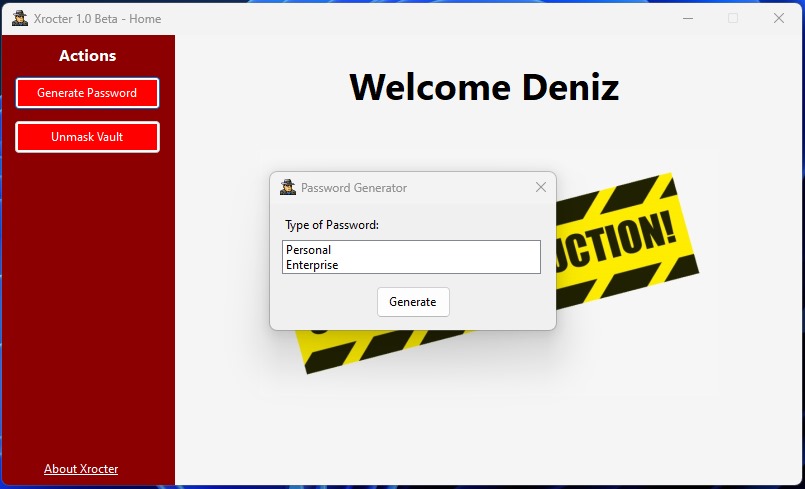
The database management system of MyPass is based on Entity Framework Core as the object relational mapper which converts C# classes into SQL queries to create a database on the Microsoft SQL Server. In this diagram there are 10 database model classes created as C# classes and 10 database contexts. The middle class that transforms C# data to SQL queries is AppDbContext. AppDbContext, maps the relations between classes and populates the database. However, this type of database manipulation is not done automatically through AppDbContext. Everytime a change happens to C# classes (including AppDbContext), the user needs to run migrations on .NET CLR (abbreviated for Common Language Runtime). This can be done using the built -in Visual Studio terminal or Microsoft Powershell. After the developer successfully creates a migration (a document that contains SQL queries to manipulate table data), the user updates the database using either the same .NET CLR or Microsoft Powershell. Once the database is successfully created or updated, the user will be able to retrieve data using the connection string generated by Microsoft SQL Server. Additionally, developers can run Azure or Amazon Web Services to implement this application for commercial purposes where the data is stored in a company's data center

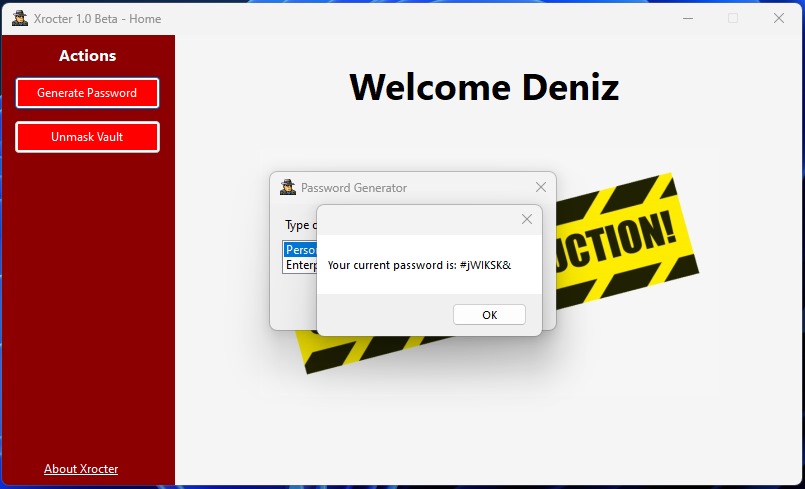
# Screenshots Of application

#Login / Signup Pages

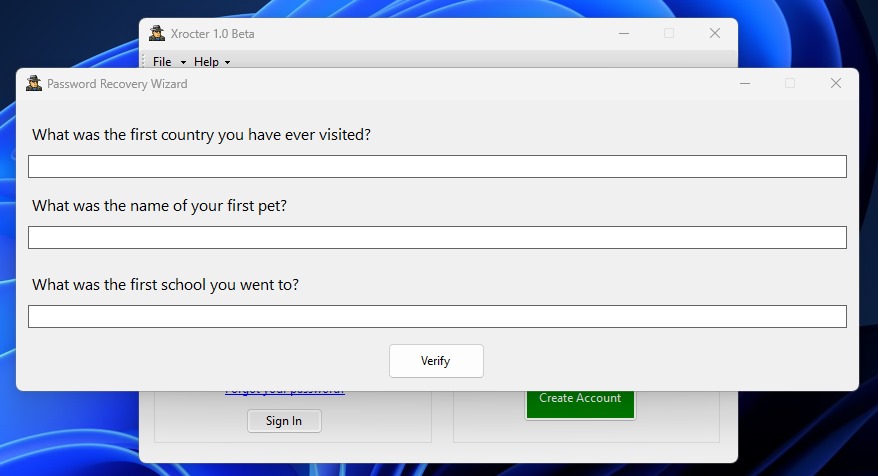


#Password Generator





#Password Recovery



#Successful account creation

