Software Architecture and Design Patterns Term Project Report

Hayden Johnson and Matthew Haloostock

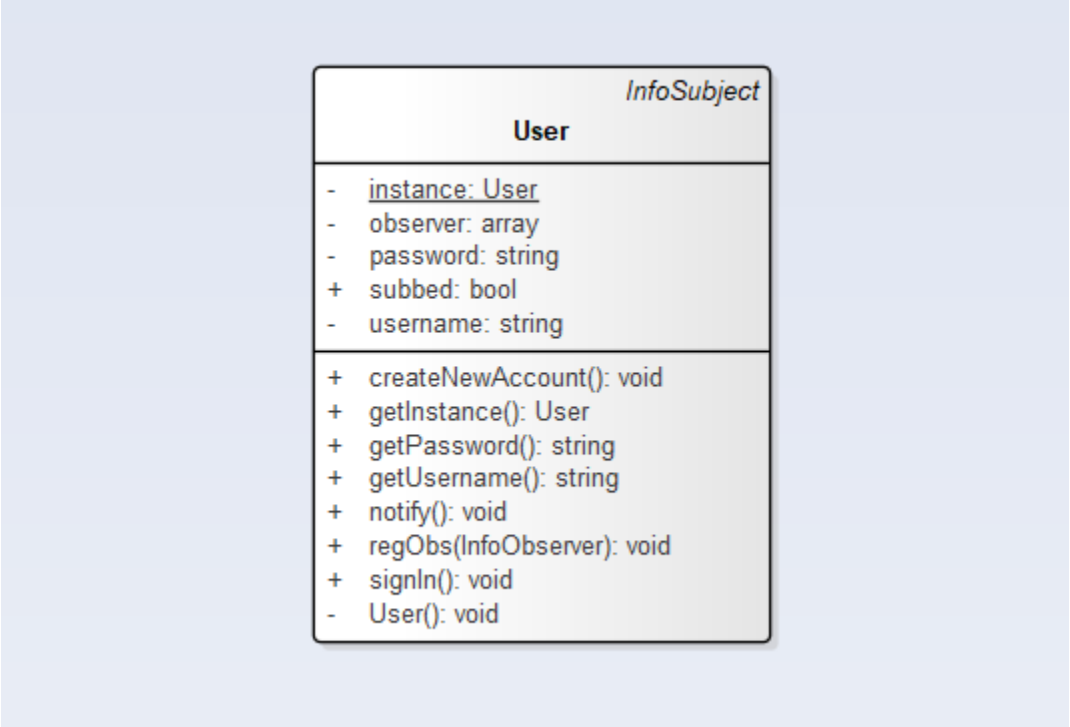
University of Michigan Dearborn

CIS 476 - Professor Tommy Xu

Fall 2025

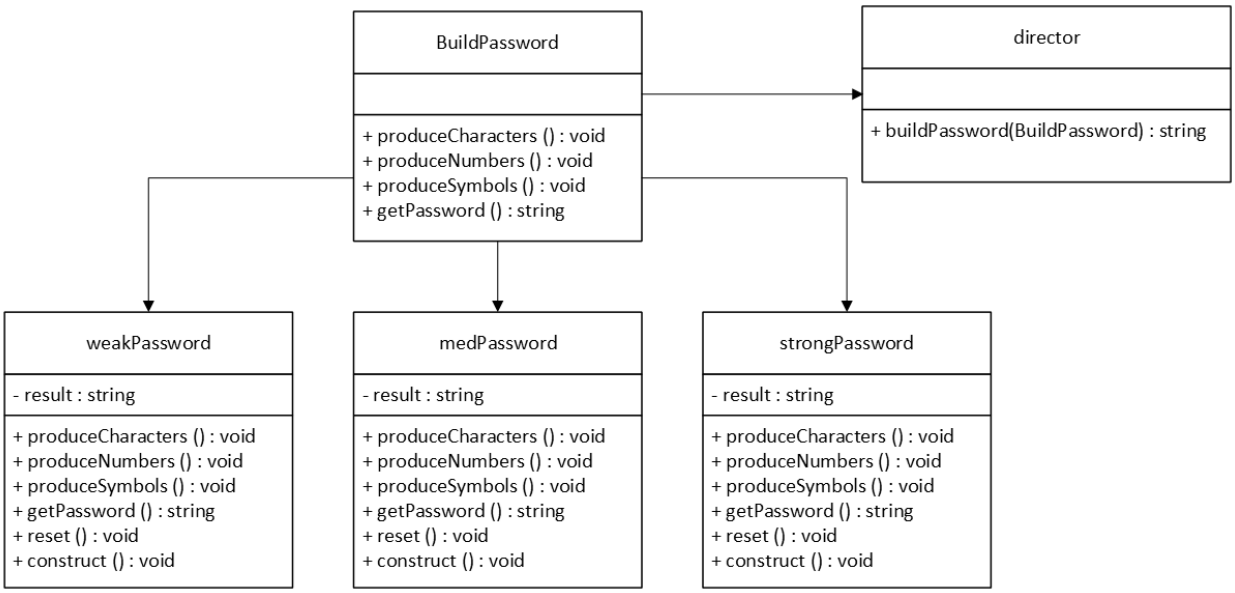
UML Class Diagrams

Singleton:



Our user class is implemented as a Singleton pattern. It is used to make sure that only one instance of the User class is used at a time so that conflicts are not created per request. It stores the user's username, password and data relevant to the Observer pattern, which it is a subject in. The constructor is private so that creating instances can be controlled through the getInstance function. It also handles logging in and creating new accounts. When a user wants to make a change to the database, the User class returns their username and password where appropriate. Their username is used in every table for identifying what data belongs to the user.

Builder:

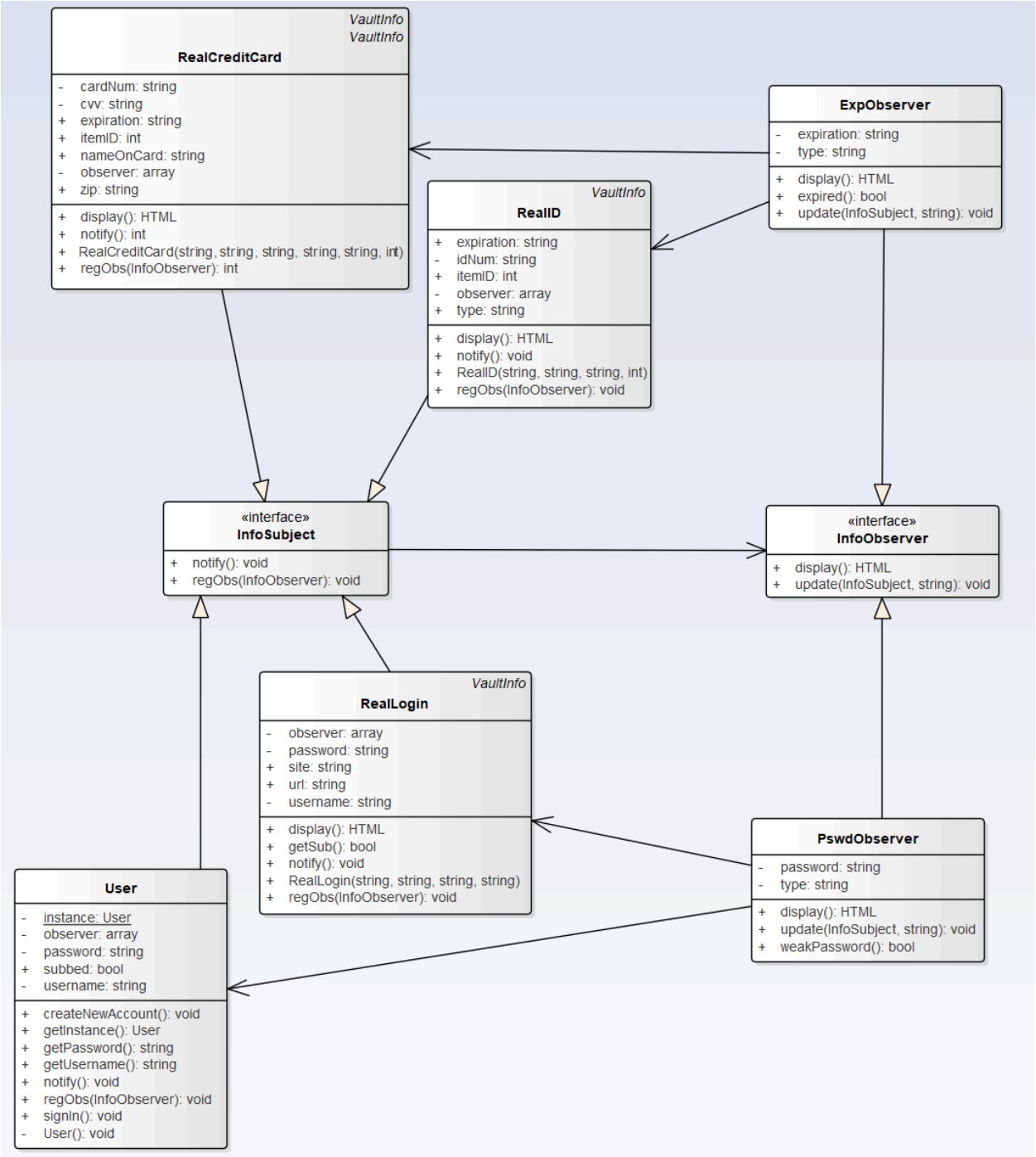


The builder pattern is implemented for password generation. Depending on if the user selects weak, medium, or hard password, the director class will invoke a different call for which password to build. When first arriving on the homepage, a weak password will be suggested.

The abstract class *BuildPassword* has 4 methods that get passed down: *produceCharacters, produceNumbers, produceSymbols,* and *getPassword*. For the *weakPassword* concrete builder, *produceCharacters* will add 10 random uppercase and lowercase characters to a string. Then, *produceNumbers* will append 10 random numbers to the end of the string. *produceSymbols* does nothing in this class. *getPassword* will return the current build and then reset the build to an empty string.

mediumPassword differs from *weakPassword* by appending 10 random symbols to the end of the string in the *produceSymbols* method. *strongPassword* will use 15 random characters, numbers, and symbols, rather than 10.

Observer:

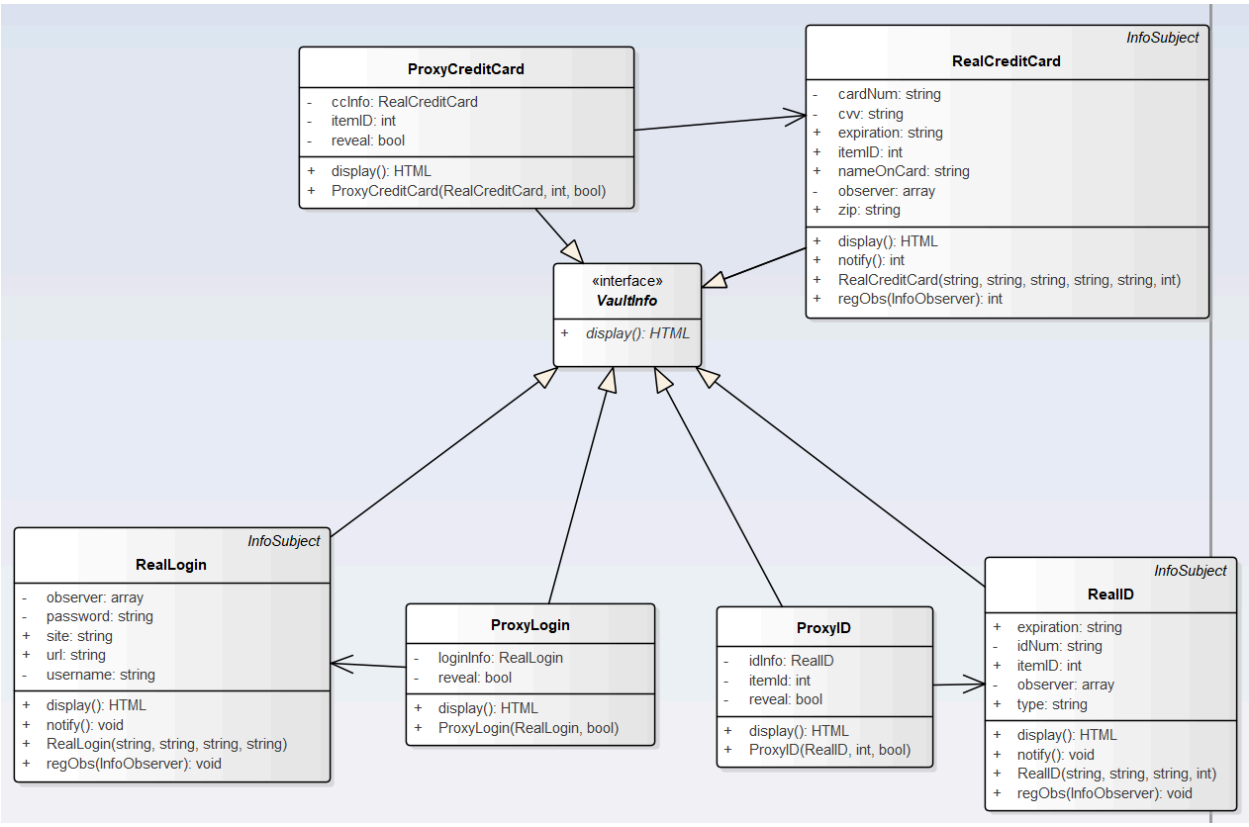


An Observer pattern is used to handle notifications for the user about objects that contain the user's stored data. The observer checks the state of the subject it is subscribed to whenever the subject calls its notify function. An interface is used to define the methods that the subject needs to have. The function regObs is used to add observers to the subject. We do not use a removeObserver function since the server is stateless and every request made by the user resets the variables. Instead we use a

database attribute for subjects to determine if they do or do not want observer notifications on a per item basis. The subjects are User, RealLogin, RealCreditCard and RealID.

The InfoObject interface defines the methods for our concrete observer classes. The concrete observers must be able to receive the current state of the subject and what type of subject it is through the update method. The display function displays a notification to the user depending on a condition determined by the observer. The PswdObserver checks whether the subject has a weak password while the ExpObserver determines whether the expiration date for a subject has passed. User and RealLogin get checked for password integrity while RealCreditCard and RealID get checked for expired credit cards or IDs.

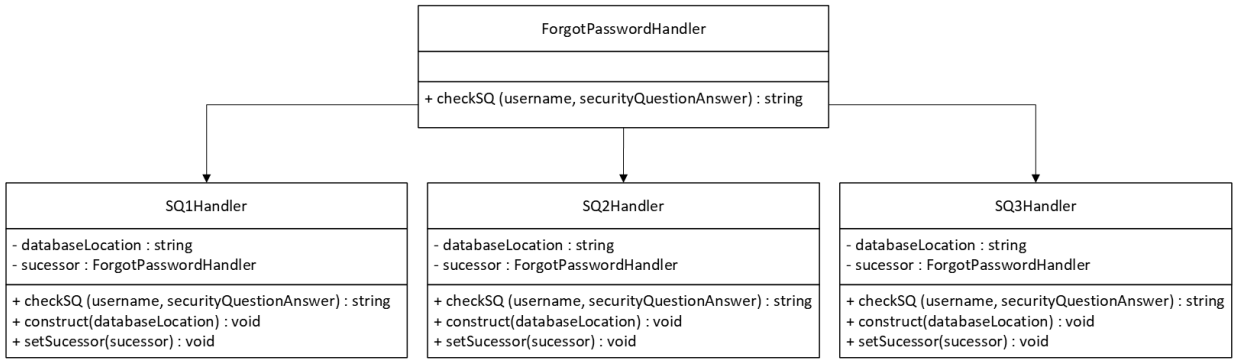
Proxy:



The Proxy pattern is used to hide sensitive information about different data items the user stored. The interface VaultInfo is used to define that all the classes that inherit from it need a display function. A proxy class stores an instance of a real class item as one of its private data members. All of the proxy classes also have a reveal attribute that tells its display function whether it should display sensitive data or not. ProxyID and ProxyCreditCard also keep an itemID attribute for helping the user identify the different items in the Credit Card and ID tables. RealLogin, RealCreditCard and RealID are used to store the relevant information stored in the vault for each category. They also function

as the subject classes which the observers monitor. There are no getter functions for real object classes, so the only way to display their private information is by calling its display function. The proxy classes thus properly protect the data inside the real objects and determine when to display them.

Chain of Responsibility:



The abstract class *ForgotPasswordHandler* has one method: checkSQ (check security question).

When *SQ1Handler->checkSQ* is invoked, it will check if its answer is correct. If correct, it invokes and returns the response given by its successor (which would be an *SQ2Handler* in this case). If incorrect, it returns “Incorrect security question 1”.

SQ2Handler->checkSQ does the same thing, except its successor is *SQ3Handler* and the incorrect statement is “Incorrect security question 2”.

SQ3Handler->checkSQ checks if the answer for its question is correct. If correct, it returns the master password, and if incorrect it returns “Incorrect security question 2”.

Database Schema

Tables:

User:

- username: VARCHAR(12) (P.K.)
- password: VARCHAR(32)
- SQ1A (Security Question 1 Answer): VARCHAR(30)
- SQ2A VARCHAR(30)
- SQ3A: VARCHAR(30)

Login:

- siteName: VARCHAR(30) (P.K.)
- u_User: VARCHAR(12) (P.K.) (F.K.)
- username: VARCHAR(20)
- password: VARCHAR(20)
- url: VARCHAR(75)
- notify: CHAR(1)

Credit_Card:

- cardNum: VARCHAR(16) (P.K.)
- cvv: VARCHAR(4)
- nameOnCard: VARCHAR(26)
- expiration: CHAR(10)
- zip: CHAR(5)
- u_User: VARCHAR(12) (F.K.)
- notify: CHAR(1)

Identification:

- idNum: VARCHAR(19) (P.K.)
- type: VARCHAR(15)
- expiration: CHAR(10)
- u_User: VARCHAR(12) (F.K.)
- notify: CHAR(1)

Secure_Notes :

- u_User: VARCHAR(12) (P.K.) (F.K.)
- noteName: VARCHAR(10) (P.K.)
- note: VARCHAR(1000)

Description:

The User table holds the master login credentials for the user that gives them access to their information stored in the vault. It holds their username, which must be unique since it is the primary key, and their password. It also stores their answers for security questions 1, 2 and 3.

The Login table keeps track of the various usernames and passwords each user stores in the vault. Each tuple in the table will also have attributes for the name of the website the logins are associated with and the URL for the website. The attribute u_User (the user's username) is a foreign key that references User. This way we can retrieve only the login credentials for the user to whom they belong. Here, notify is used to determine whether this tuple should be checked by the concrete observer for password strength. The primary key is a composite key between siteName and u_User, allowing the user to have a single login for each website.

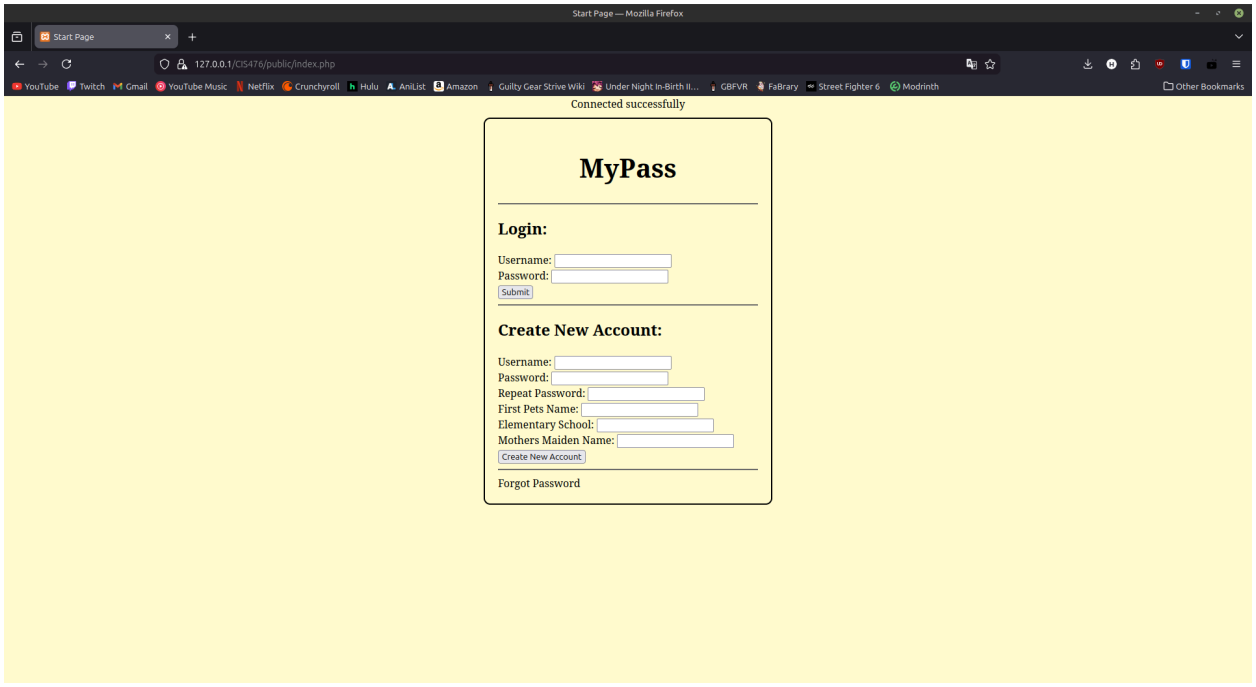
The table for Credit_Card stores all the information the user will need to manage their credit cards. Since credit card numbers are unique, our table will use it as the primary key, cardNum. They can also be 15 or 16 digits in length typically, so we are using VARCHAR(16) for their data type. Similarly, a CVV is typically 3 or 4 digits long, so we will give it VARCHAR(4). We also need to store the expiration date and the zip code of the card. The notify attribute is used to determine whether this object is subscribed to the observer class or not. The foreign key is u_User since it can reference the associated user and the system can get all cards for that user only.

The Identification table is used for holding various IDs for the user. ID numbers can vary in length depending on the type. Even driver's license numbers between states can differ, with New York's being 19 digits long. Therefore, we chose VARCHAR(19) for attribute idNum, which serves as the primary key since every ID number is unique. We also need a type attribute which keeps track of whether the stored ID is a driver's license, state ID, SSN, passport and any other form of identification. Since most types of ID expire, the table needs an attribute, expiration, to keep track of it. We also use the notify attribute here in order to determine if an identification subject is subscribed to the observer or not. And again, this table will have u_User as a foreign key, referencing User.

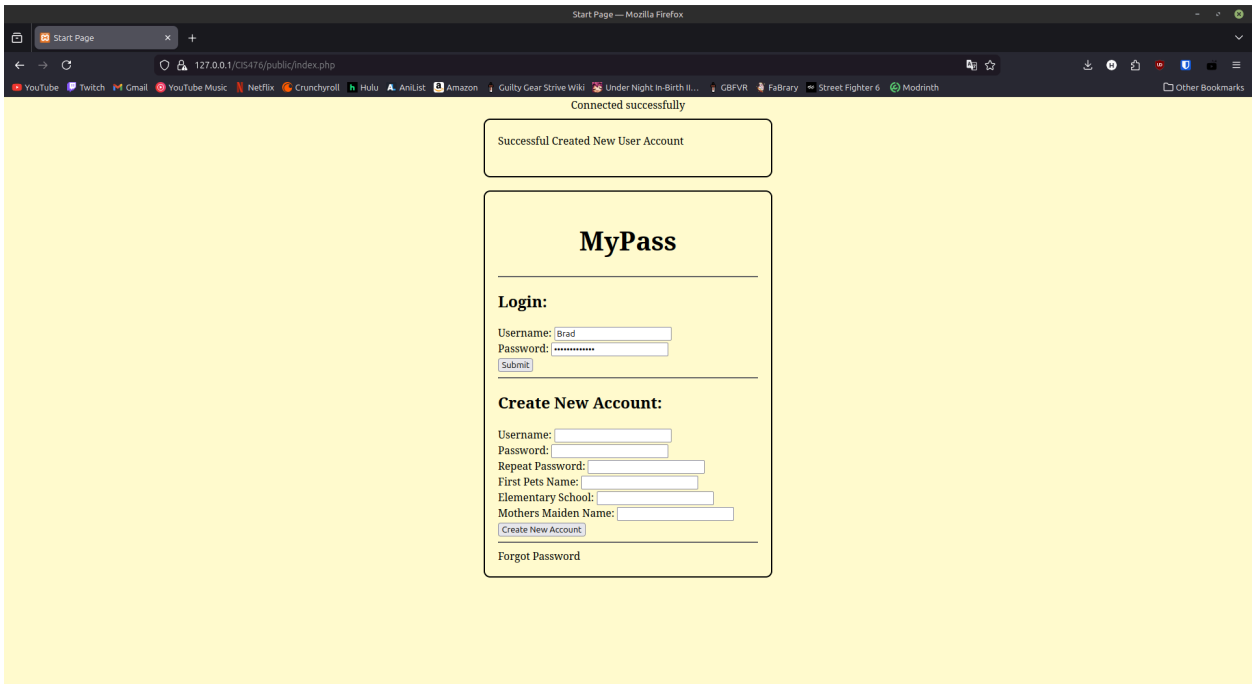
The last table in our database system, Secure_Notes, will store notes in the vault for the user. The primary key is a composite key between attributes u_User and noteName, which is a user given name for the note. The num attribute can be the same between users but will be associated with a different username (u_User), allowing a user to keep multiple notes. Attribute u_User is a foreign key referencing User. Last, the note attribute holds the secure note, which we allow to be 1000 characters long.

UI Screenshots

Login:

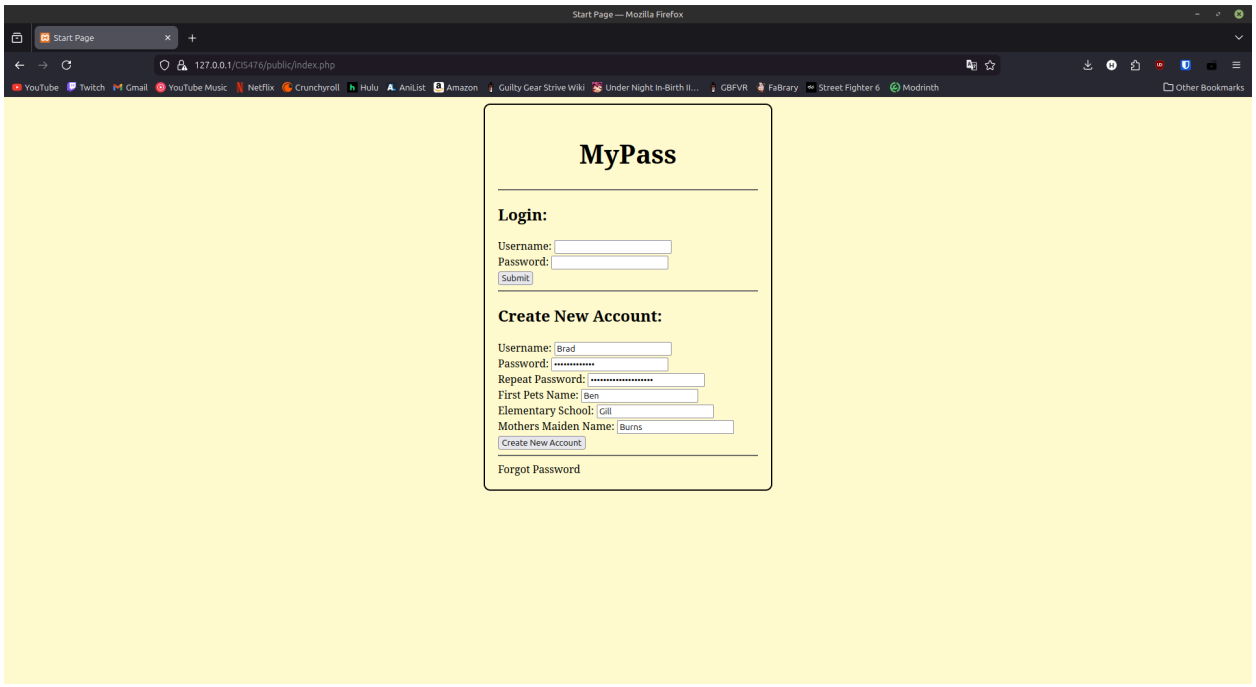


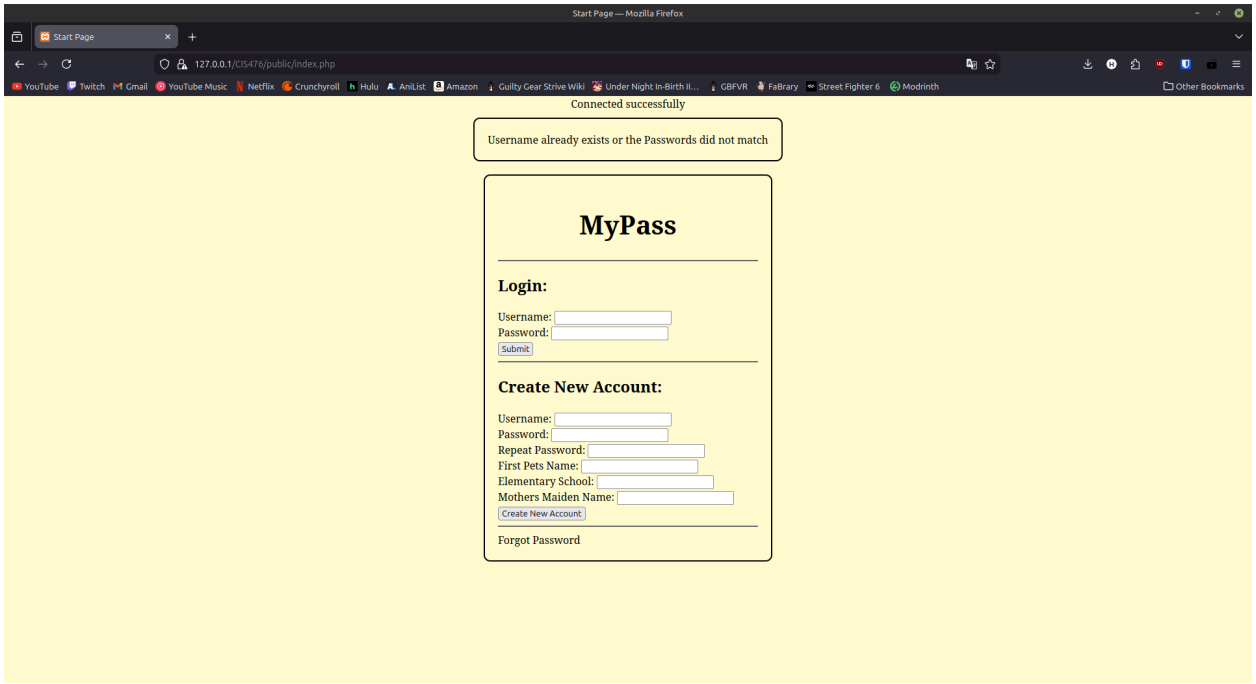
This image showcases the login and create new account forms at the starting page of the MyPass web-application.



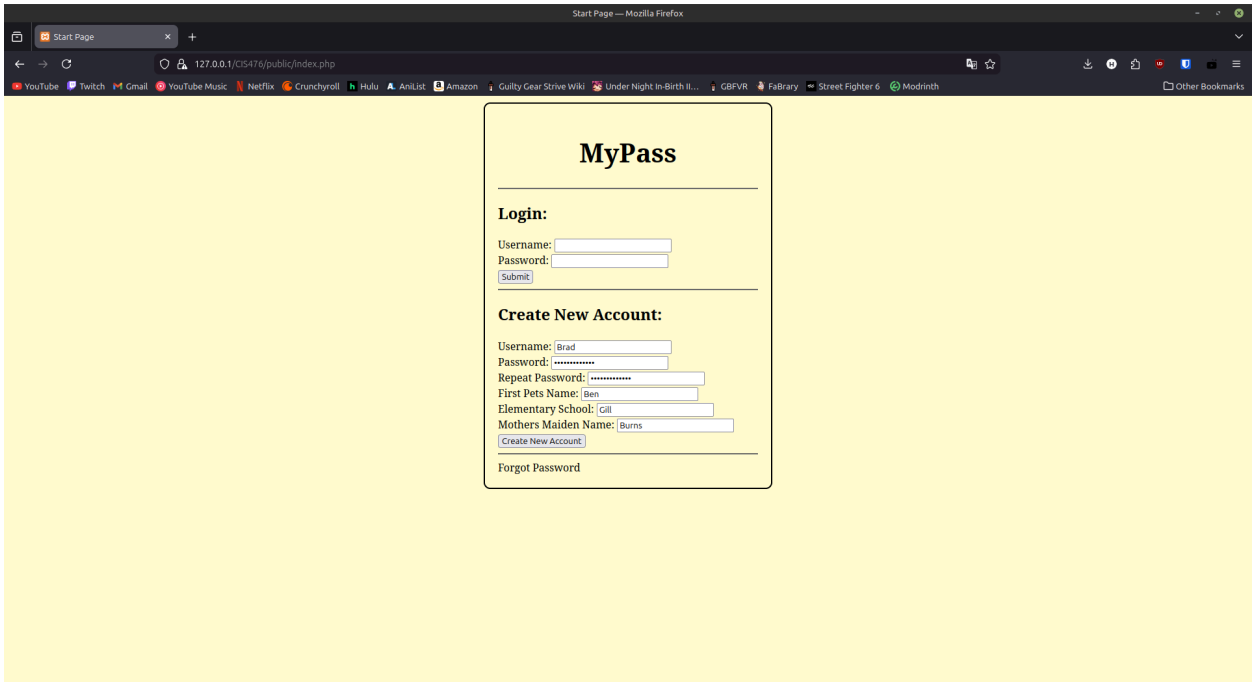
This image shows the message for successfully creating an account

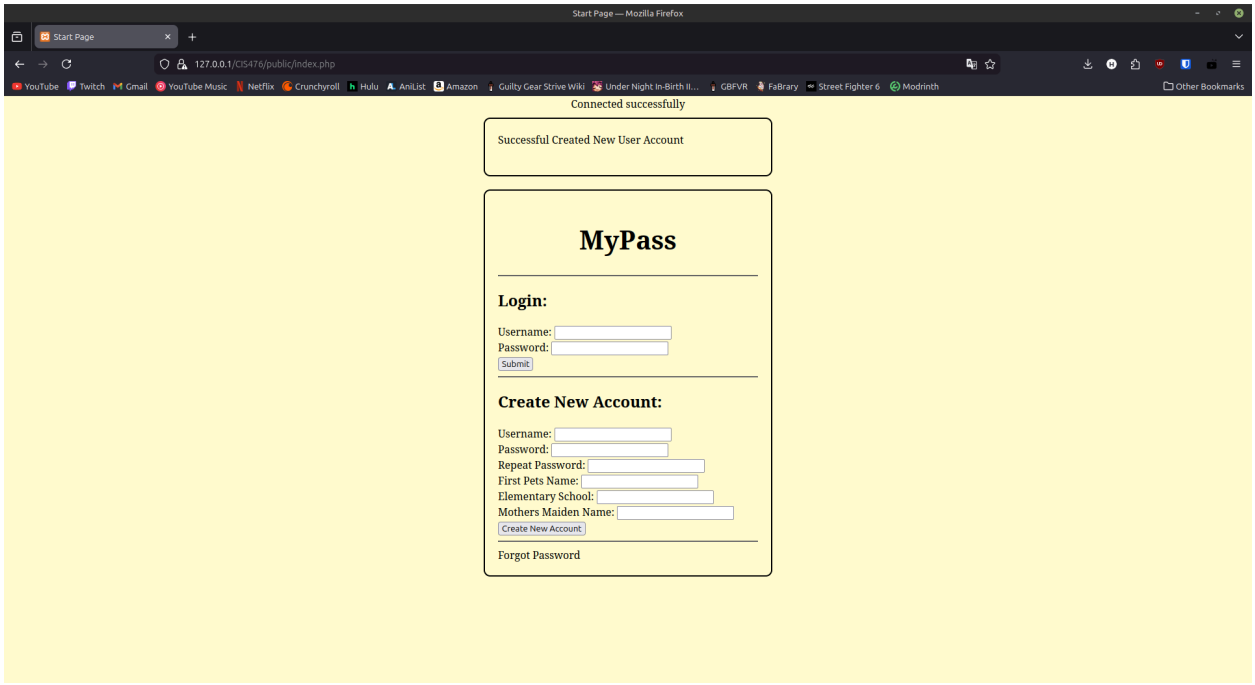
New Account:



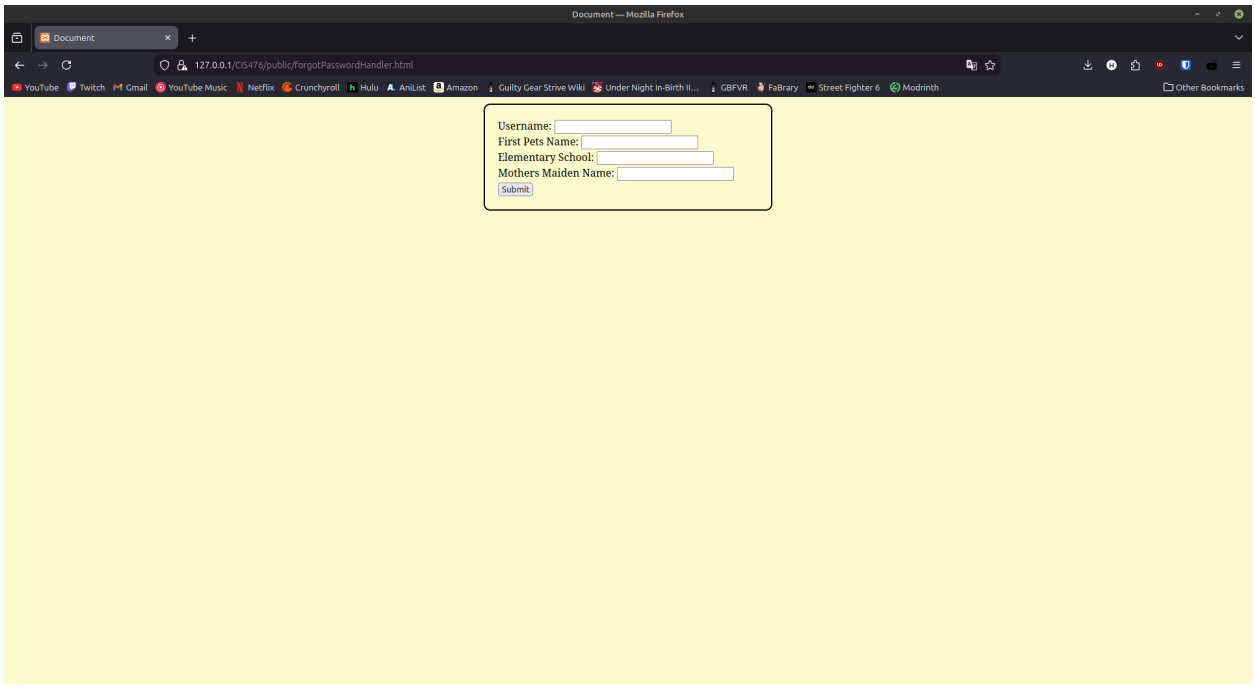


These two images show what happens if a user's passwords do not match when they try to create a new account. An error message is displayed at the top of the page.

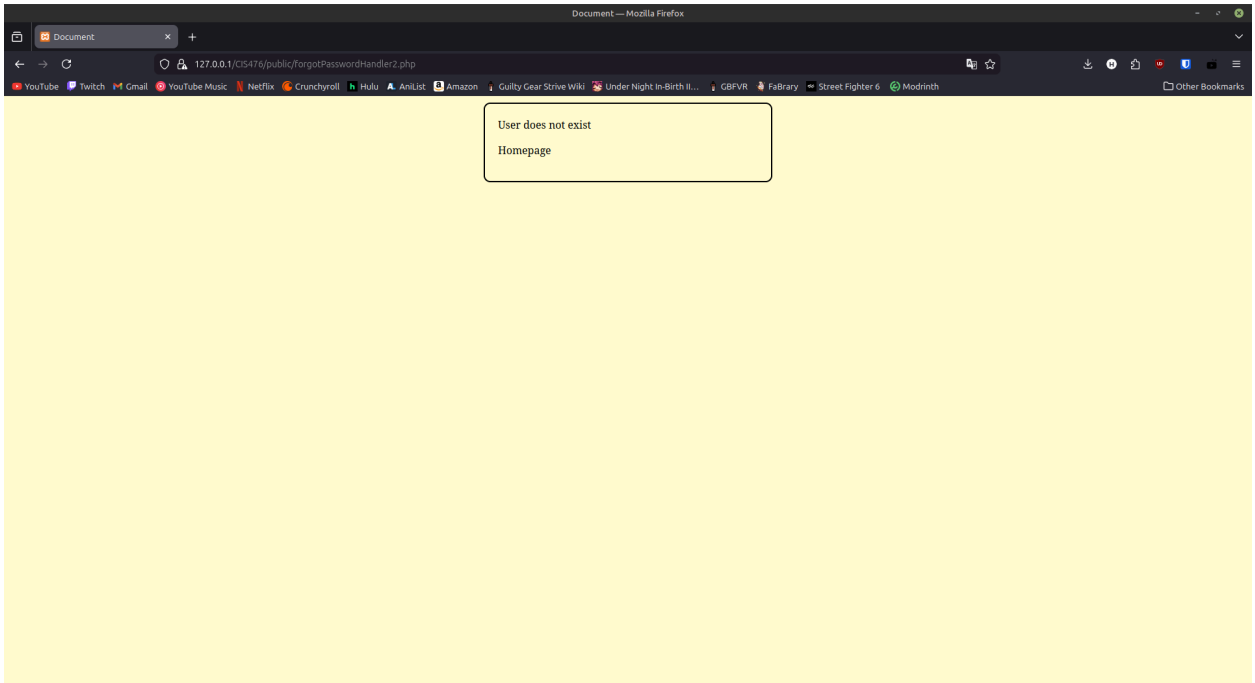

Start Page — Mozilla Firefox
Start Page
127.0.0.1/CIS476/public/index.php
YouTube
Twitch
Gmail
YouTube Music
Netflix
Crunchyroll
Hulu
AniList
Amazon
Guilty Gear Strive Wiki
Under Night In-Birth II...
GBFVR
FaBrary
Street Fighter 6
Modrinth
Other Bookmarks
MyPass
Login:
Username:
Password:
Submit
Create New Account:
Username: Brad
Password:
Repeat Password:
First Pets Name: Ben
Elementary School: Gill
Mothers Maiden Name: Burns
Create New Account
Forgot Password


Start Page — Mozilla Firefox
Start Page
127.0.0.1/CIS476/public/index.php
YouTube
Twitch
Gmail
YouTube Music
Netflix
Crunchyroll
Hulu
AniList
Amazon
Guilty Gear Strive Wiki
Under Night In-Birth II...
GBFVR
FaBrary
Street Fighter 6
Modrinth
Other Bookmarks
Connected successfully
Successful Created New User Account
MyPass
Login:
Username:
Password:
Submit
Create New Account:
Username:
Password:
Repeat Password:
First Pets Name:
Elementary School:
Mothers Maiden Name:
Create New Account
Forgot Password

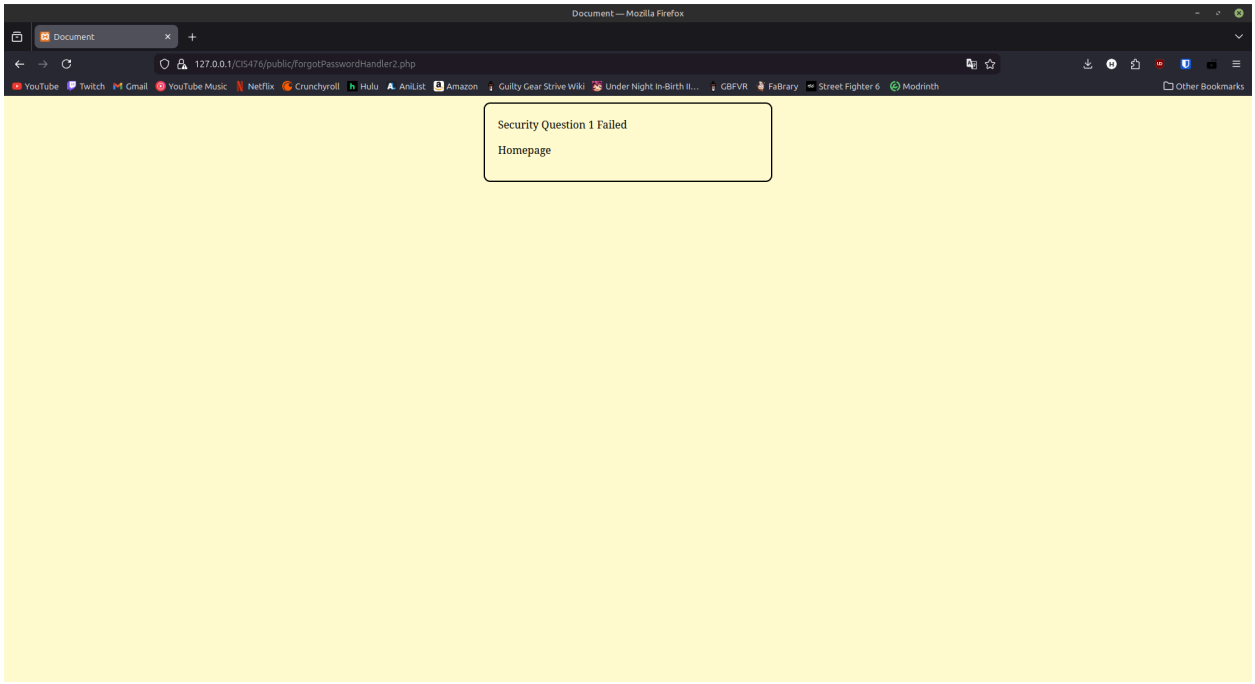
Forgot Password:



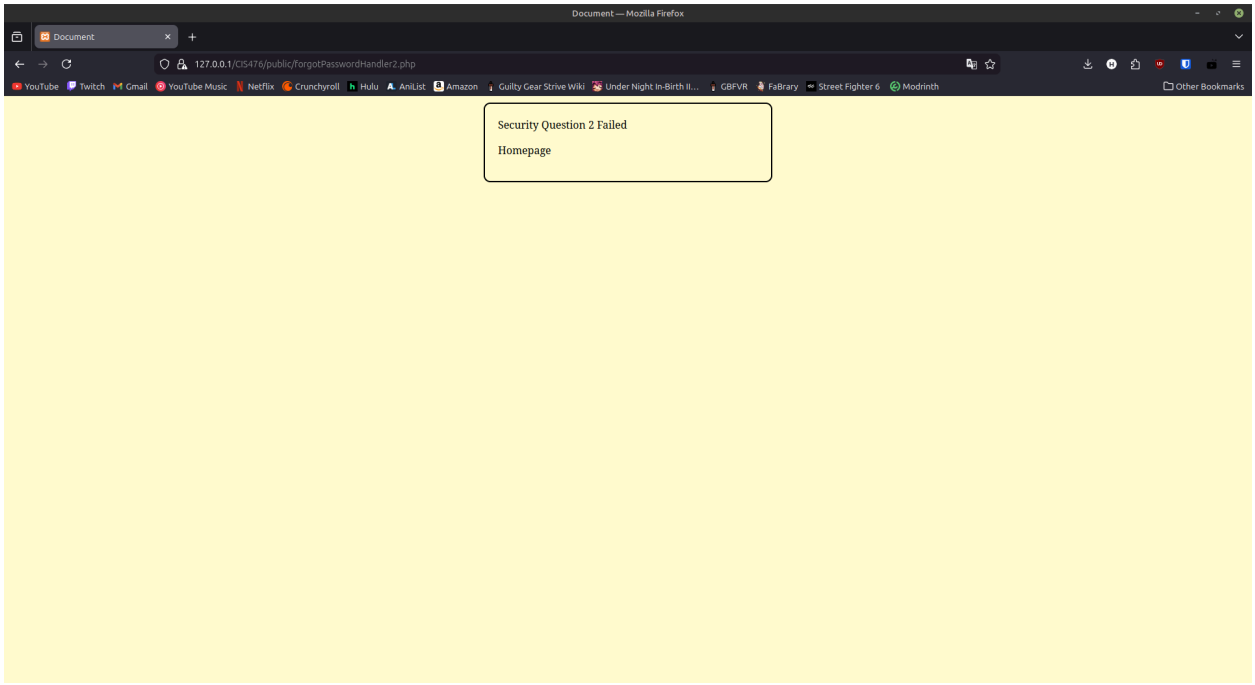
The above image displays the “forgot password” form where the user must answer 3 security questions.



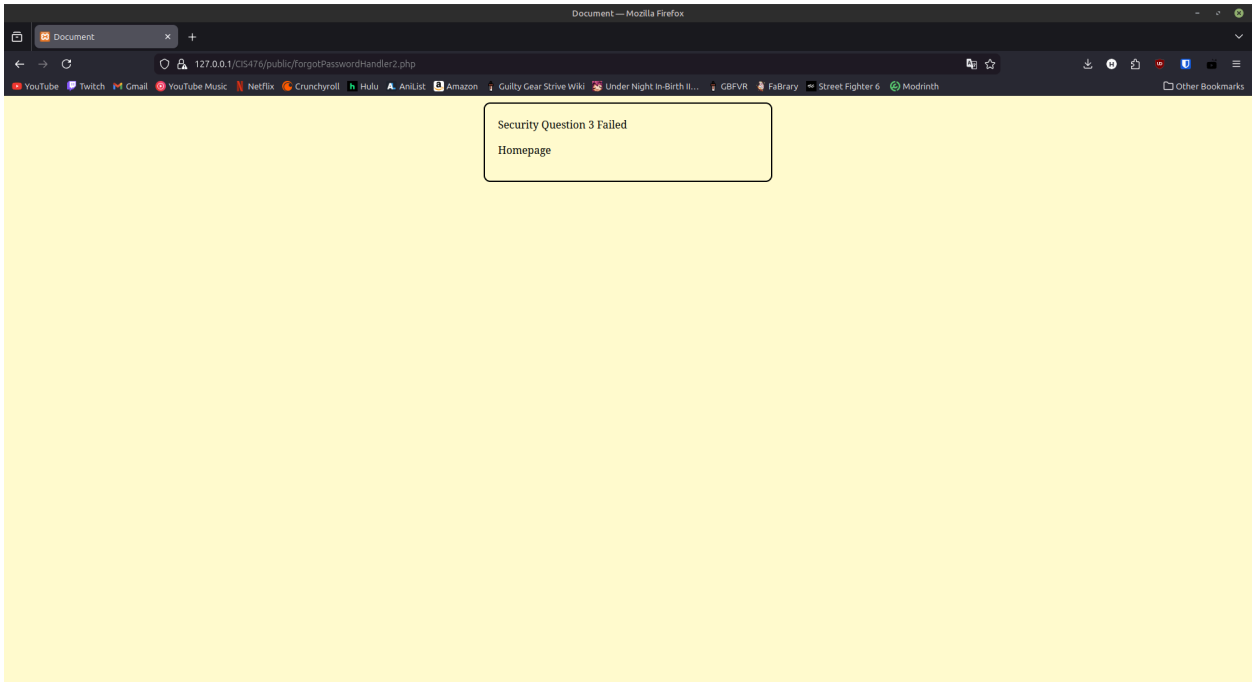
What happens when the user tries to submit the form with an invalid username.



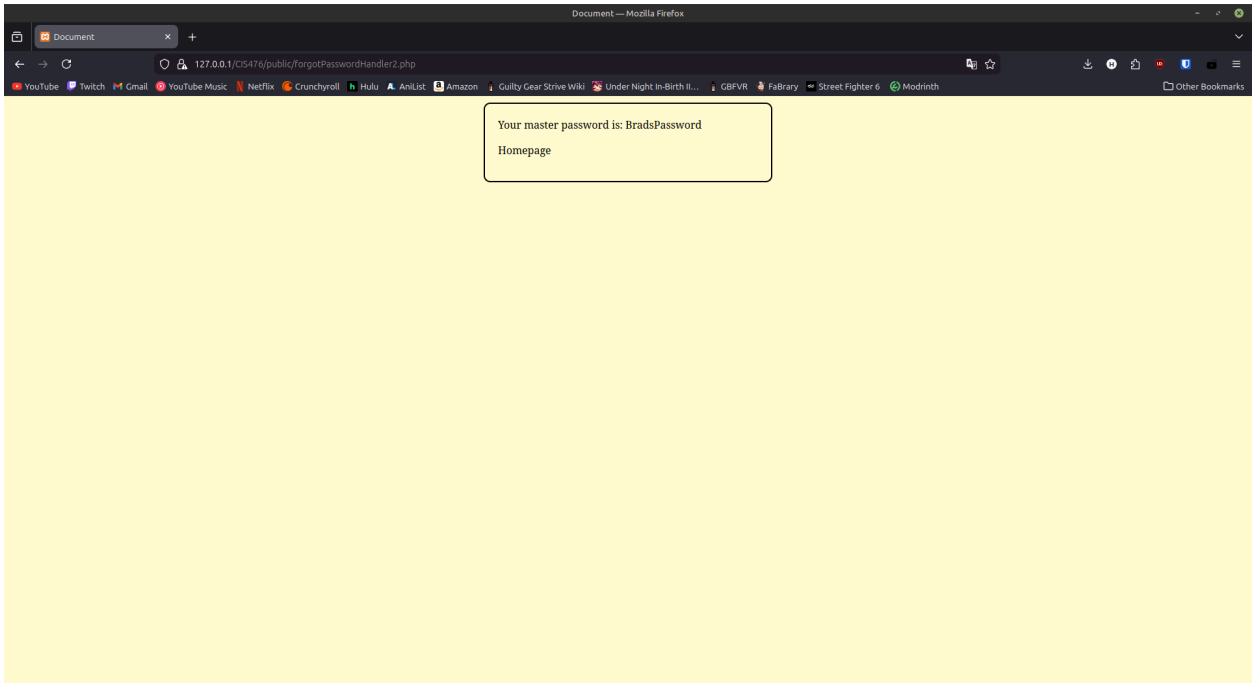
The answer to security question 1 was not correct.



The answer to security question 2 was not correct.

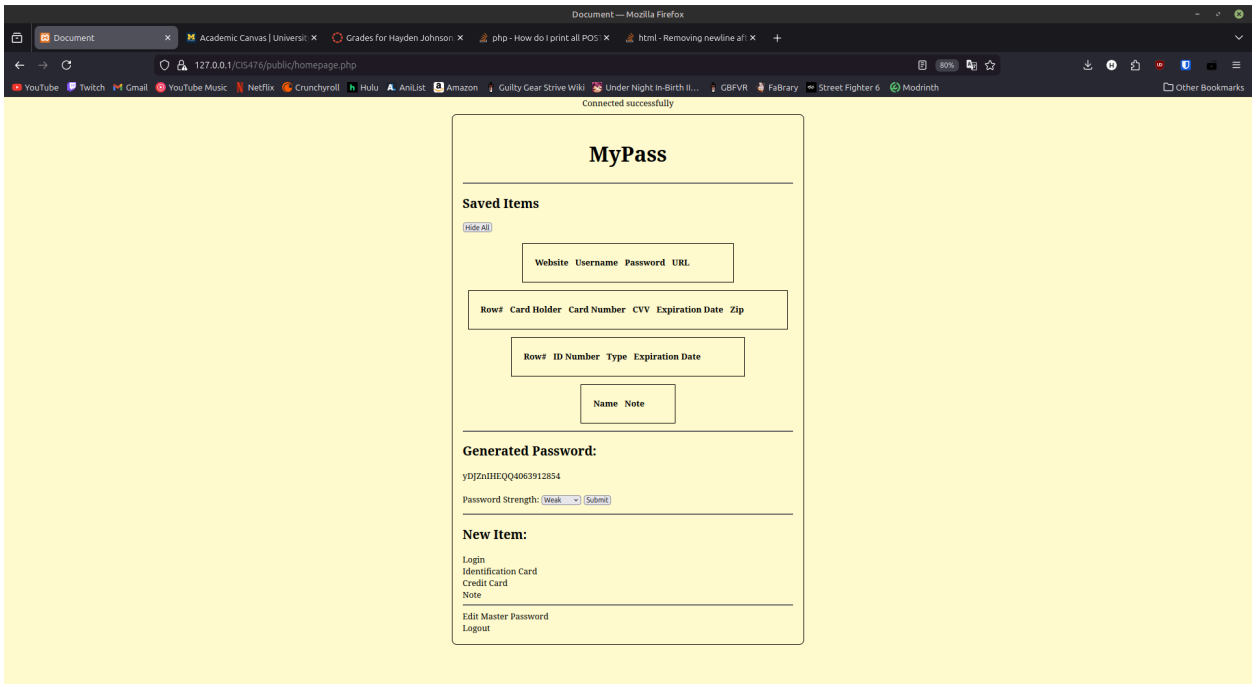


The answer to security question 3 was not correct.

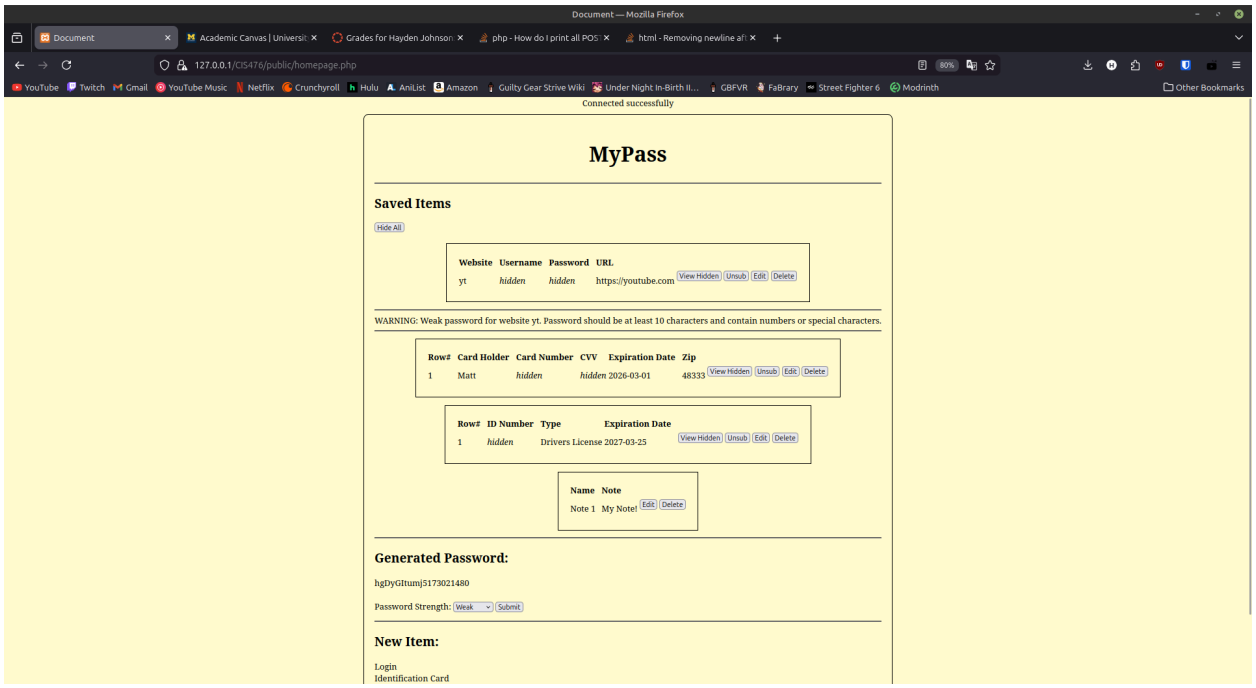


The entered username was valid and all of the security questions were answered correctly so the user is displayed their password. They can copy it before returning to the Homepage.

Homepage:

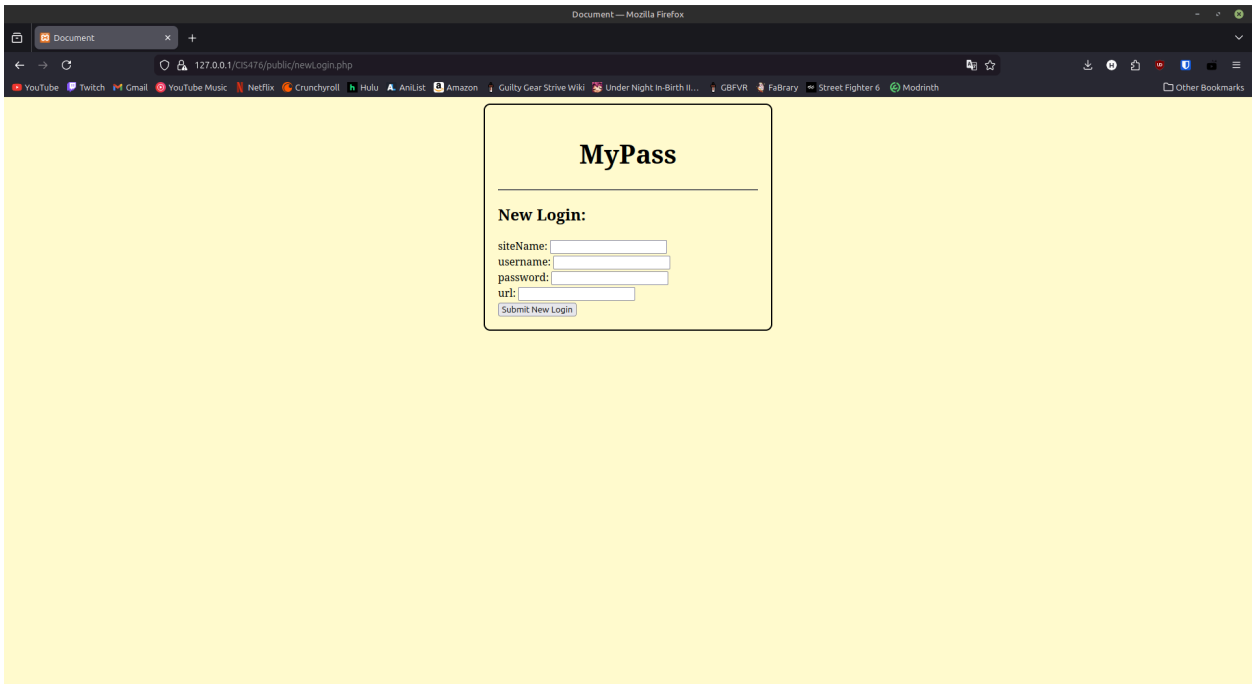


The above image displays a successful login where the user has no data stored in the vault. Passwords are automatically generated with varying strengths.

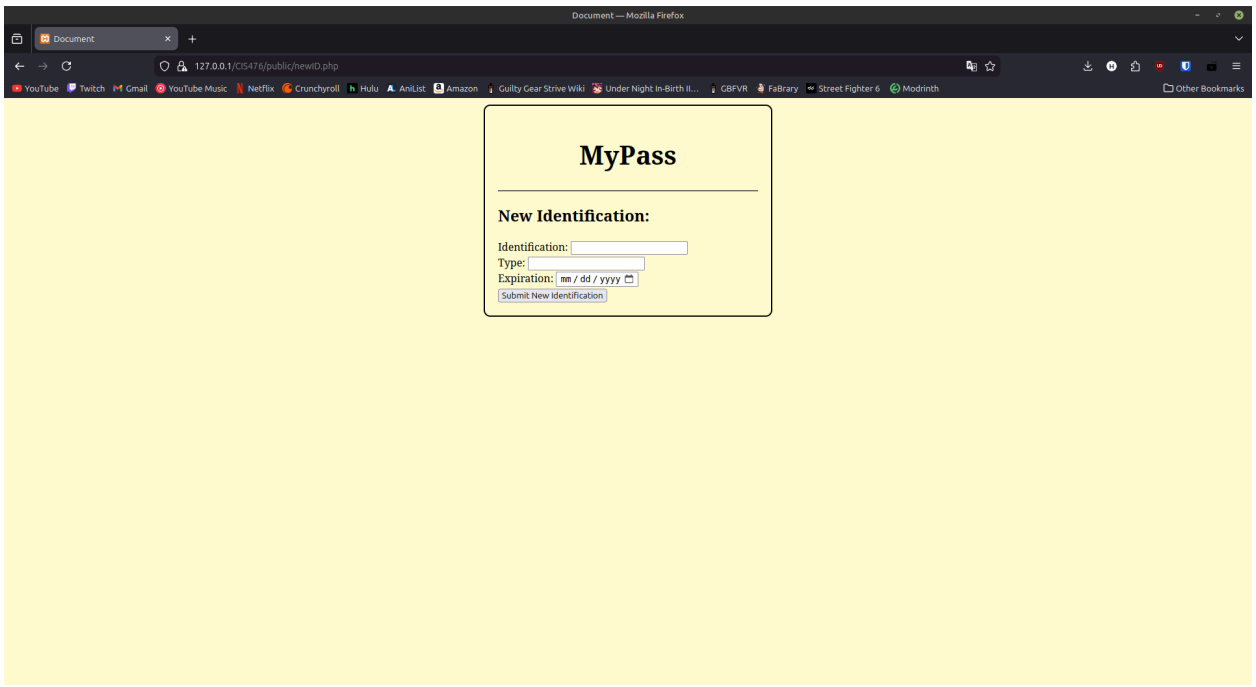


This image displays the user's Homepage when there are items stored in the vault. The top table is for Logins, the second is for Credit Cards, the third is for IDs and the last is for Secure Notes. There is a warning under Logins states that the password for “yt” is too weak.

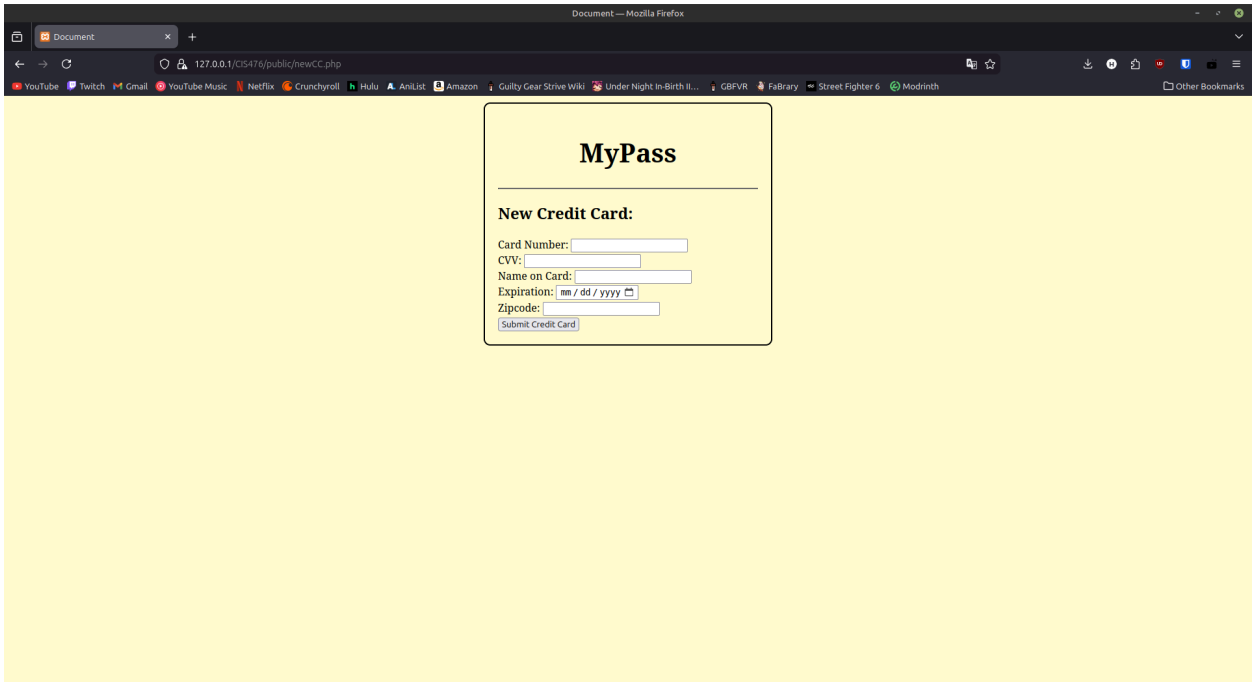
New Item:



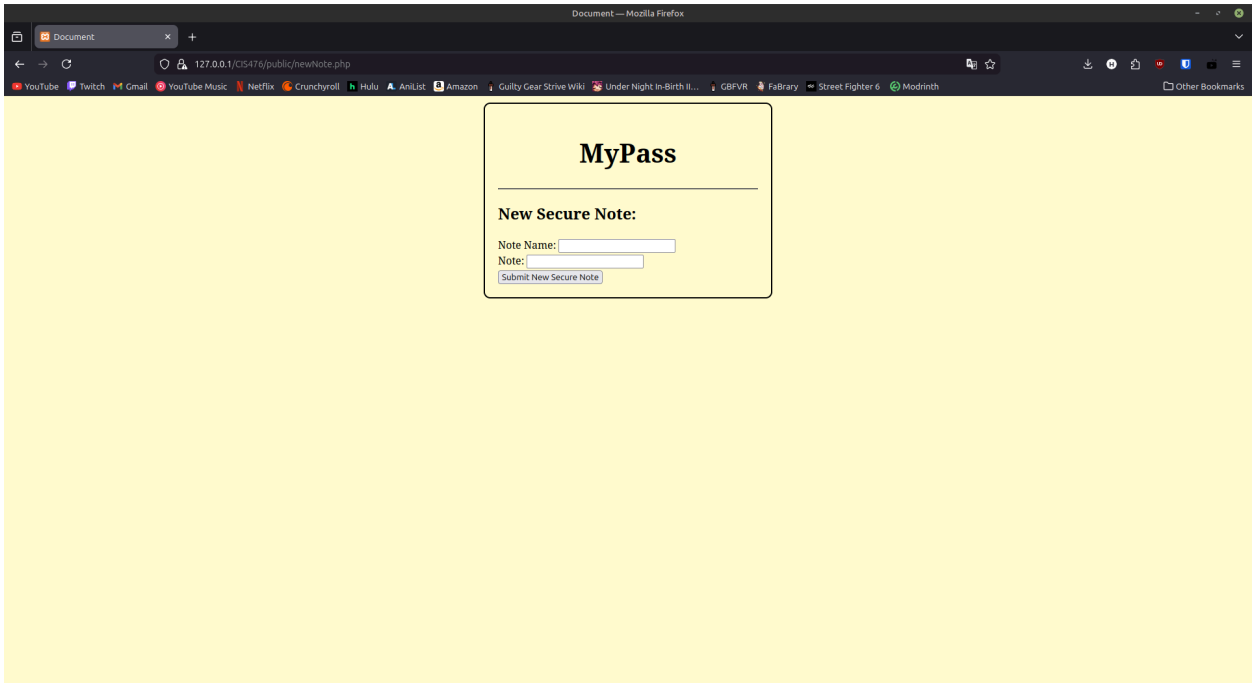
Above is the Create New Login page. The user gets here by clicking “Login” under the “New Item” section.



Above is the Create New Identification page. The user gets here by clicking “Identification Card” under the “New Item” section.

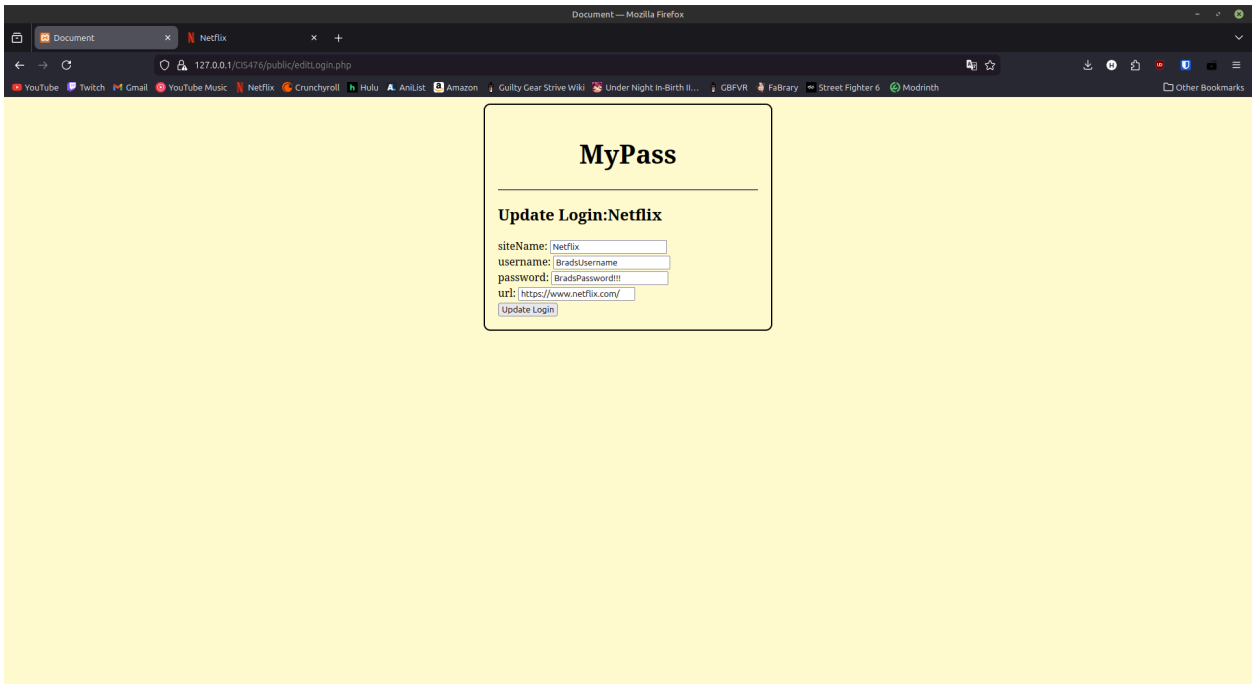


Above is the Create New Credit Card page. The user gets here by clicking “Credit Card” under the “New Item” section.

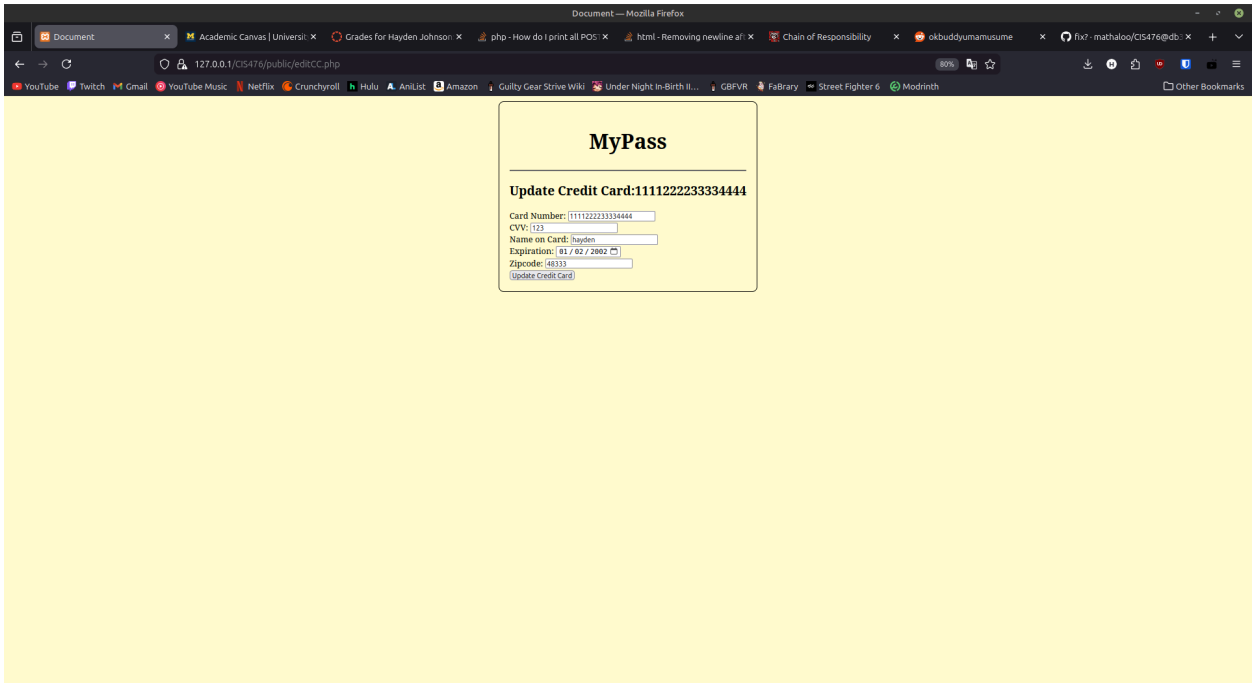


Above is the Create new Secure Note page. The user gets here by clicking “Note” under the “New Item” section.

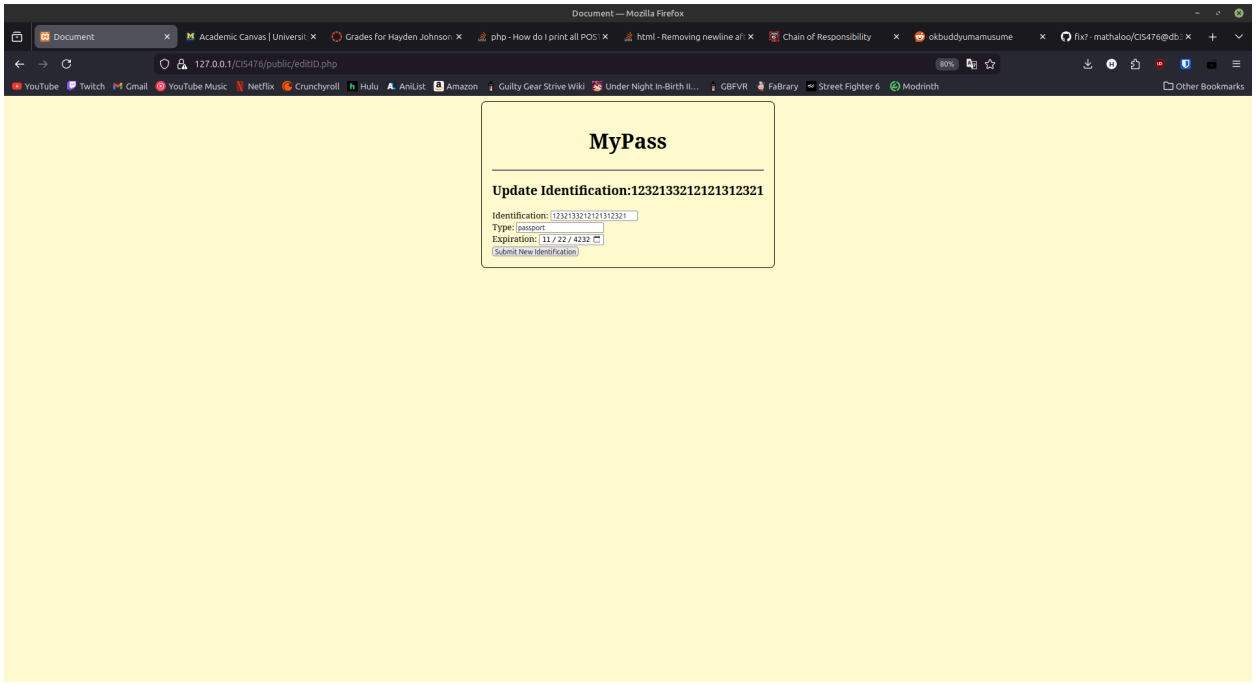
Update Item:



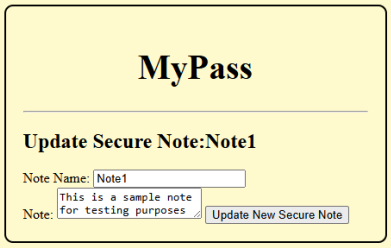
This image shows the Update Login page. The user can get here by clicking “Edit” next to the item that they want to modify. The current data fills the inputs for user convenience.



This image shows the Update Credit Card page. The user can get here by clicking “Edit” next to the item that they want to modify. The current data fills the inputs for user convenience.



This image shows the Update Identification page. The user can get here by clicking “Edit” next to the item that they want to modify. The current data fills the inputs for user convenience.

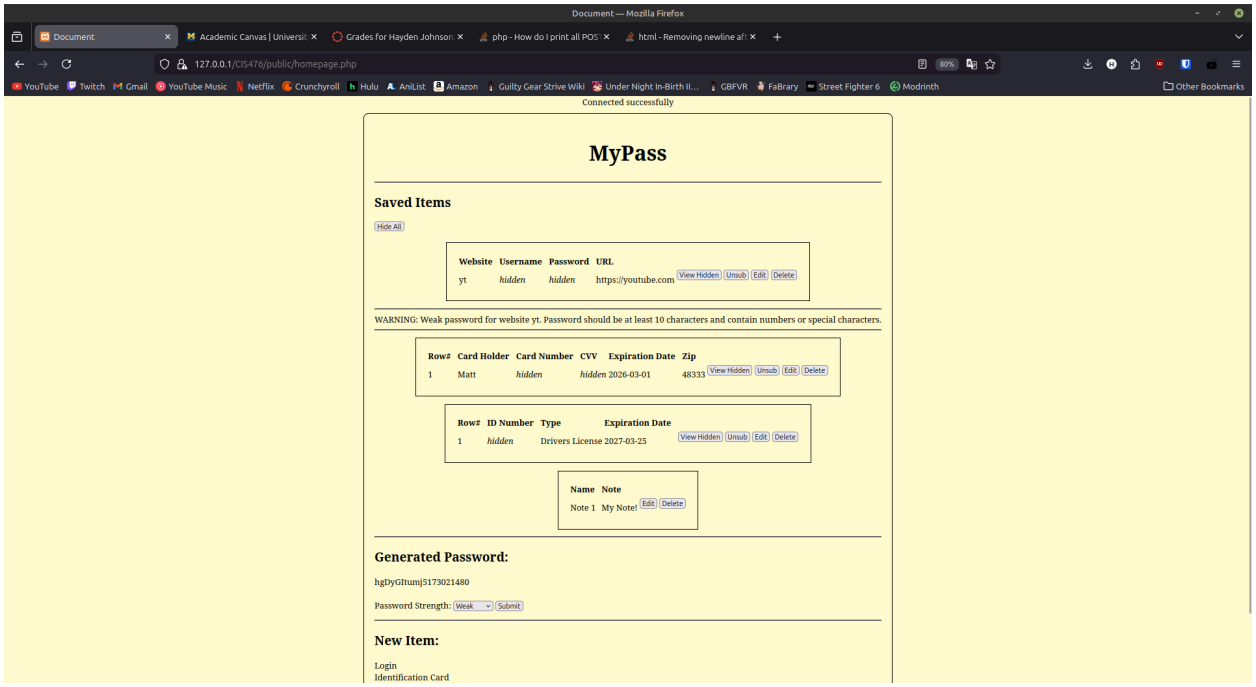


This image shows the Update Secure Note page. The user can get here by clicking “Edit” next to the item that they want to modify. The current data fills the inputs for user convenience.

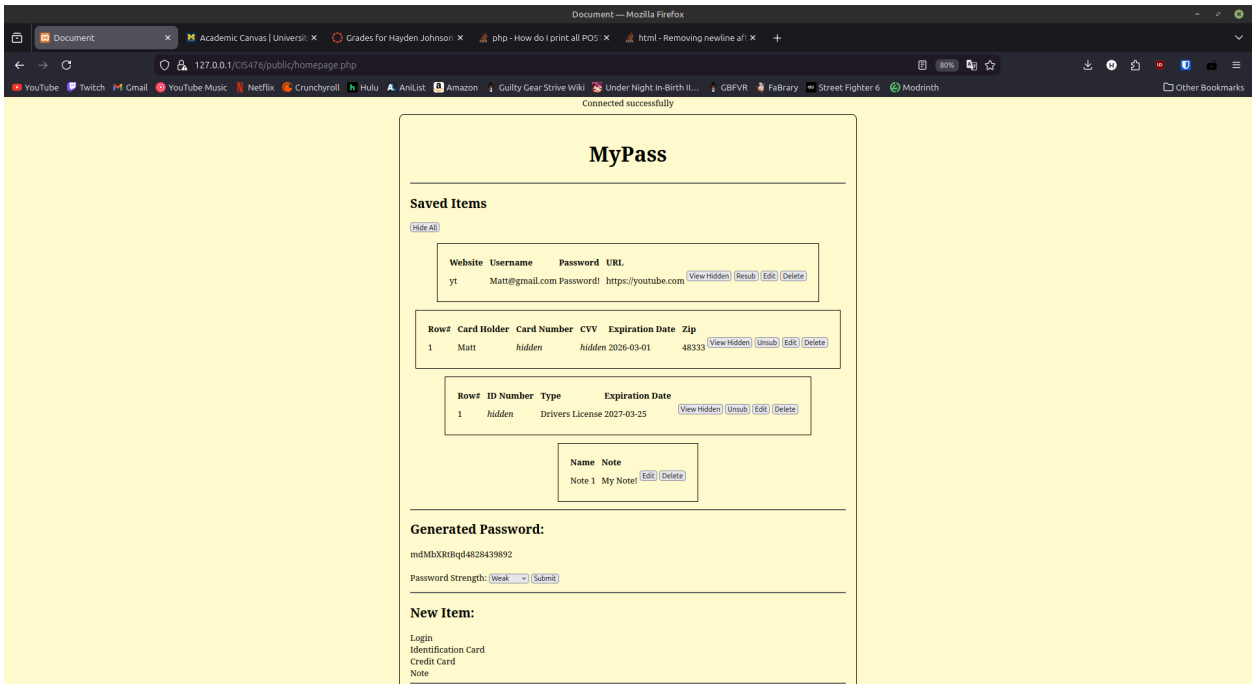
This image shows the Update Master Password page. The user can get here by clicking “Edit Master Password” near the bottom of the page. Here the user can change their password by clicking update or turn on/off notifications about the strength of their master password (vault login) with the Unsub/Resub buttons. There is also a link to return to the Homepage.

Unhide Data and Observe Weak Password:

Before:



After hitting “View Hidden” and “Unsub”:



References

https://refactoring.guru