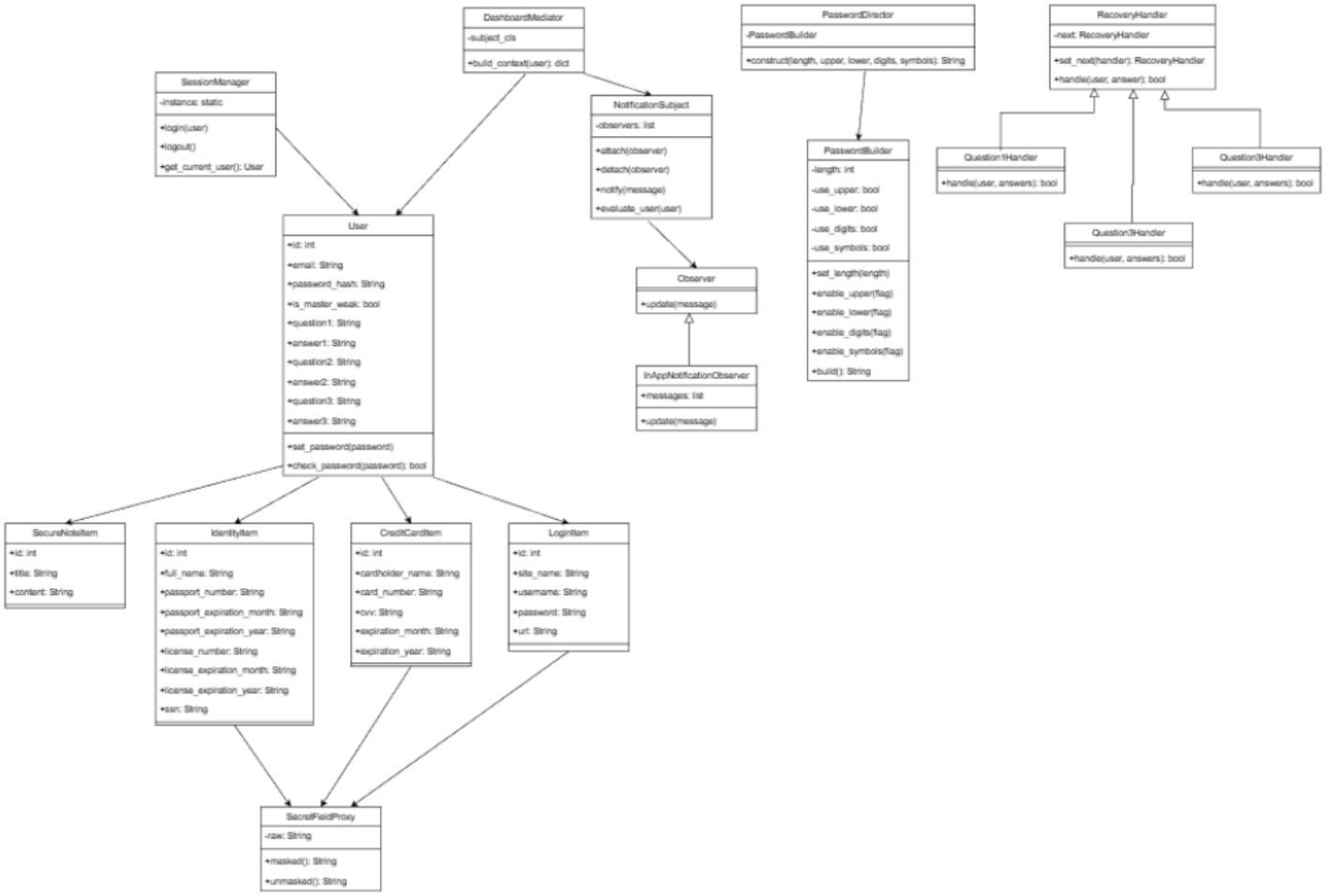
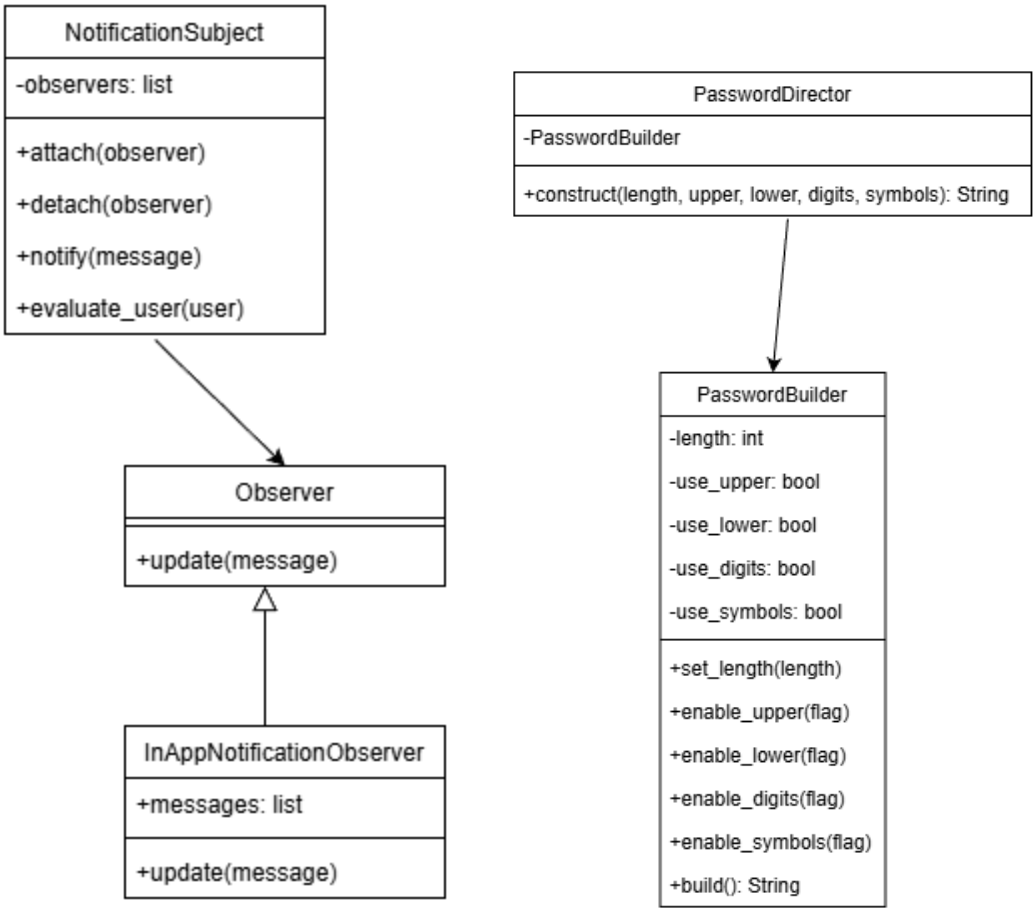
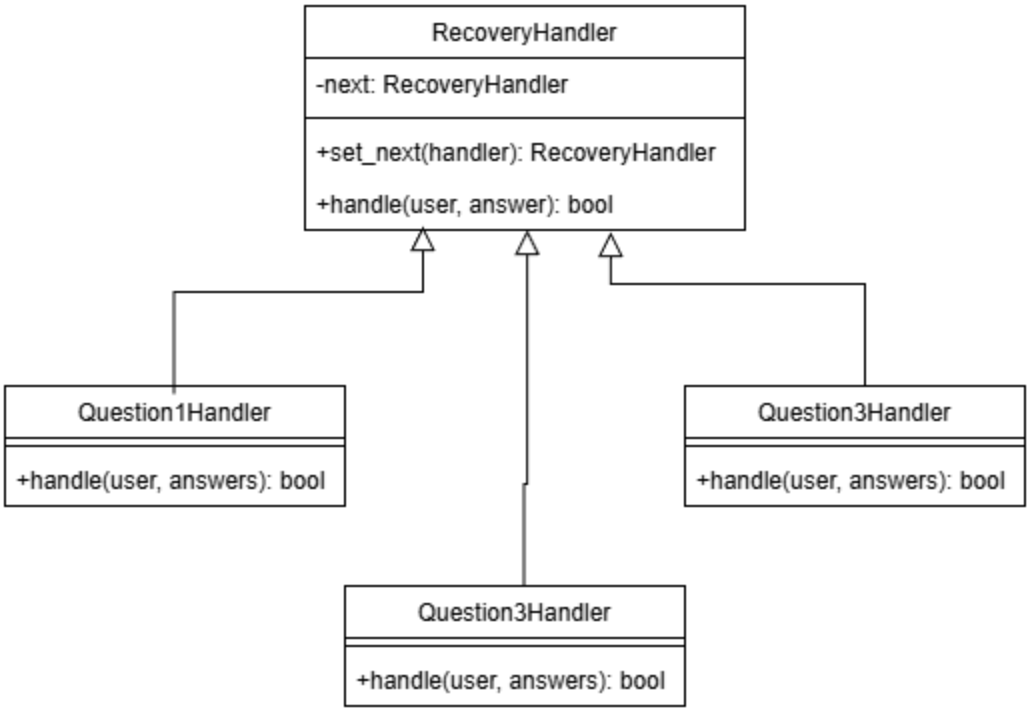
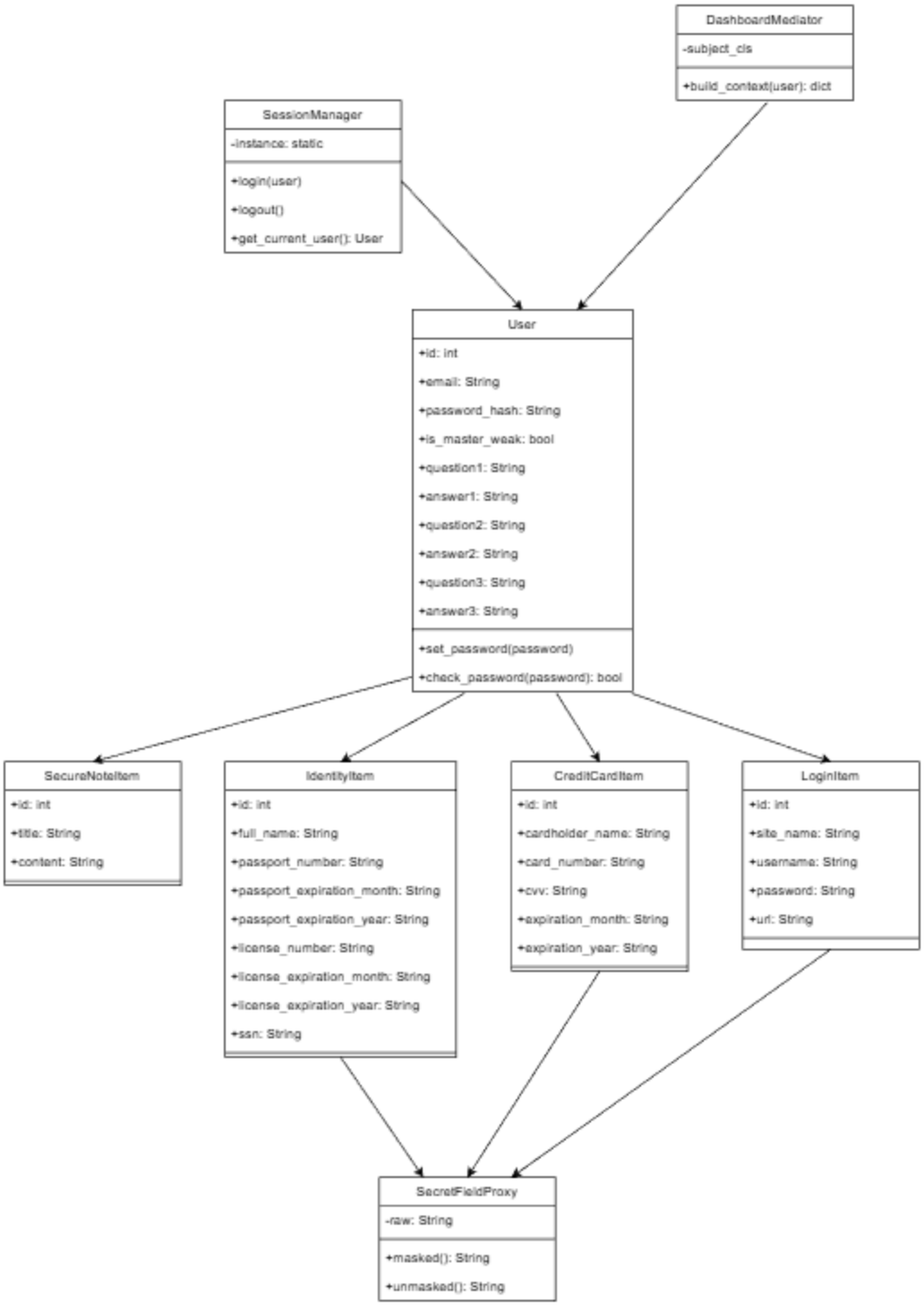
Full Class Diagram:




NotificationSubject
-observers: list
+attach(observer)
+detach(observer)
+notify(message)
+evaluate_user(user)
Observer
+update(message)
InAppNotificationObserver
+messages: list
+update(message)
PasswordDirector
-PasswordBuilder
+construct(length, upper, lower, digits, symbols): String
PasswordBuilder
-length: int
-use_upper: bool
-use_lower: bool
-use_digits: bool
-use_symbols: bool
+set_length(length)
+enable_upper(flag)
+enable_lower(flag)
+enable_digits(flag)
+enable_symbols(flag)
+build(): String


RecoveryHandler
-next: RecoveryHandler
+set_next(handler): RecoveryHandler
+handle(user, answer): bool
Question1Handler
+handle(user, answers): bool
Question3Handler
+handle(user, answers): bool
Question3Handler
+handle(user, answers): bool



Classes description:

User: This class represents a registered user of the system. It stores the authentication details, master password, and all answers to the security questions.

LoginItem: A saved username and password for a website that includes the site name, username, password, and URL. The Proxy pattern is used to display sensitive data.

CreditCardItem: Represents a stored credit card, and includes the cardholder name, number, CVV, and expiration date. Sensitive fields including the card number and CVV can be masked and unmasked using the Proxy, and the expiration date is assessed by the Observer pattern to send notifications to the user.

IdentityItem: This class represents a stored personal identity document such as a passport, driver's license, or SSN. It uses the Proxy pattern for masking and unmasking sensitive data, and the Expiration fields are evaluated to send warnings to the user through the observer pattern.

SecureNoteItem: A text note that may contain private information stored securely in the mypass application by the user.

SessionManager: Implements the Singleton pattern to ensure that only one session exists. This class contains the current logged-in user, handles login and logout functionality, and executes an automatic logout after a period of inactivity

SecretFieldProxy: This class wraps a sensitive string value and provides two controlled access methods: masked() returns a partially hidden value, and unmasked() shows the full value. This class is meant to hide potentially sensitive data in the UI.

PasswordBuilder: A builder class that can be configured by a user and is used to construct complex passwords. Length and character sets can be selected and its build() method produces the final password based on these options.

PasswordDirector: This class is responsible for coordinating the builder to assist in password construction. It invokes the appropriate builder configuration methods and returns a final password. This allows the password generation to be decoupled from actual construction logic.

Observer: This is an interface defined for all observers in the Observer pattern. All of the concrete observers have to implement the update method in order to receive notifications.

InAppNotificationObserver: This represents the concrete implementation of the Observer class. It stores a list of incoming notifications, which show up on the dashboard view as alert notifications for the user.

NotificationSubject: Maintains a list of observers and creates notifications. It evaluates the user's stored data for the following: weak master password, weak login passwords, expired credit cards, and expired passport/license. It then notifies all of the attached observers when an issue is found.

DashboardMediator: Implements the Mediator pattern to handle the dashboard’s communication with the system. It creates the NotificationSubject, attaches the observers, triggers evaluation, and generates the dashboard context.

RecoveryHandler: Base class for the password recovery chain; keeps a reference to the next handler and defines the abstract handle() method. Each handler checks one security answer and forwards the request along the chain if successful.

Question1Handler, Question2Handler, Question3Handler: Each of these classes checks a single security question answer during password recovery. If the answer is correct, the next handler in the chain is called and if it is incorrect, the chain stops and the recovery attempt fails.

Database Schema:

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY,
    email VARCHAR(255) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    is_master_weak BOOLEAN NOT NULL DEFAULT 0,
    question1 VARCHAR(255) NOT NULL,
    answer1 VARCHAR(255) NOT NULL,
    question2 VARCHAR(255) NOT NULL,
    answer2 VARCHAR(255) NOT NULL,
    question3 VARCHAR(255) NOT NULL,
    answer3 VARCHAR(255) NOT NULL
);

CREATE TABLE login_items (
    id INTEGER PRIMARY KEY,
    site_name VARCHAR(255) NOT NULL,
    username VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    url VARCHAR(255),
    user_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE credit_cards (
    id INTEGER PRIMARY KEY,
    cardholder_name VARCHAR(255) NOT NULL,
    card_number VARCHAR(255) NOT NULL,
    cvv VARCHAR(10) NOT NULL,
    expiration_month VARCHAR(2) NOT NULL,
    expiration_year VARCHAR(4) NOT NULL,
    user_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

```
CREATE TABLE identities (
    id INTEGER PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    passport_number VARCHAR(255),
    passport_expiration_month VARCHAR(2),
    passport_expiration_year VARCHAR(4),
    license_number VARCHAR(255),
    license_expiration_month VARCHAR(2),
    license_expiration_year VARCHAR(4),
    ssn VARCHAR(255),
    user_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE secure_notes (
    id INTEGER PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    content TEXT NOT NULL,
    user_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Description:

Users Table:

All authentication and security information, along with password recovery, is stored in the users table. Each row represents a registered MyPass user. It is also the parent table for vault items through one-to-many relationships.

- **id**: Primary key identifying a user
- **email**: the email address used as the username for login
- **password_hash**: Bcrypt-hashed master password used to decrypt the vault
- **is_master_weak**: A boolean flag indicating whether the master password fails strength checks
- **question1, question2, question3**: The three security questions chosen during registration
- **answer1, answer2, answer3**: Answers to the security questions, stored securely

Login_items Table:

Stores entries of websites or services saved by the user. It provides login credential storage with masking and unmasking through the Proxy pattern and weak password detection through the Observer pattern.

- **id**: Primary key for the login item.
- **site_name**: The name of the website
- **username**: The login username for that site

- **password**: The password stored in the vault, masked when displayed on the UI
- **url**: The website URL
- **user_id**: Foreign key that links to the owner in the users table.

Credit_cards Table:

This table stores sensitive credit card information belonging to the user and allows secure storage and management of credit card information. The Observer pattern checks these entries for expired cards and shows the corresponding notifications.

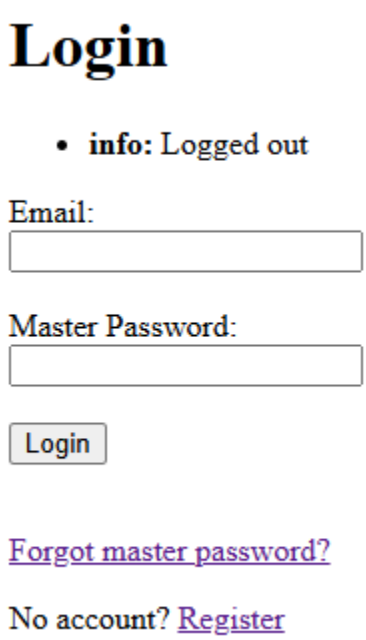
- **id**: Primary key for the card entry
- **cardholder_name**: name on card
- **card_number**: Full credit card number; masked and unmasked using Proxy
- **cvv**: Security code masked and unmasked with Proxy
- **expiration_month**, **expiration_year**: Used to determine expiration status
- **user_id**: Foreign key that links the card to a specific user

Identities Table:

Stores personal identity documents.Expired passports and licenses trigger the Observer notifications that are then displayed on the dashboard.

- **id**: Primary key for the identity entry
- **full_name**: The user's name.
- **passport_number**: Passport identifier masked using proxy.
- **passport_expiry_month, passport_expiry_year**: Used to detect expiration of passport
- **license_number**: Driver's license number, masked via Proxy.
- **license_expiry_month, license_expiry_year**: Used for expiration detection of license
- **ssn**: Social Security Number, masked using Proxy
- **user_id**: Foreign key linking the identity to the owning user.

UI Screenshots:



Login Screen: used to enter username and password, register as a new user, or recover master password

Register

Email:

Master Password:

Security Questions

Question 1:

Answer 1:

Question 2:

Answer 2:

Question 3:

Answer 3:

Register

Already have an account? Login

Register Screen: used for new users to register. Must input email, master password, and the 3 recovery questions in case master password must be recovered.

Recover Master Password

Email:

Answer your three security questions to reset your master password.

Answer 1:

Answer 2:

Answer 3:

New Master Password:

Reset Password

Back to Login

Recover Master Password Screen: user is prompted to answer the 3 recovery questions chosen in order to set a new master password.

MyPass Dashboard

Notifications

- Your master password appears weak. Consider increasing its length and incorporating mixed characters.

Vault

- Login Items
- Credit Cards
- Identities
- Secure Notes

Tools

- Password Generator

Logout

Main Dashboard: shows all of the links to vault items as well as the password generator. Notifications will appear at the top, as shown in this example with a weak master password.

Login Items

- **info:** Login item deleted.

Add / Update Login

Site Name:

Username:

Password:

URL:

Save Login

Saved Logins

Site	Username	Password	URL	Actions
test	testuser Copy	********* Show Copy	mytest.com Copy URL	Edit | Delete

Back to Dashboard

Login Items: displays all of the sites that the user has saved login sites for. Allows for adding, deleting, and editing existing sites, and password is masked using the proxy pattern.

Credit Cards

- **success:** Credit card saved.

Add Credit Card

Cardholder Name:

Card Number:

CVV:

expiration Month (MM):

expiration Year (YYYY):

Save Card

Saved Cards

Cardholder	Number	CVV	expiration	Actions
user 1	********** Show Copy	*** Show Copy	11/2024	Edit | Delete

Back to Dashboard

Credit Card Vault: securely stores user credit card information, masking sensitive fields. Allows for adding and deleting entries, as well as editing existing entries.

Identities

- **success:** Identity saved.

Add Identity

Full Name:

Passport

Passport Number:
Passport expiration Month (MM):
Passport expiration Year (YYYY):

Driver License

License Number:
License expiration Month (MM):
License expiration Year (YYYY):

Social Security

SSN:

Save Identity

Saved Identities

Name	Passport	License	SSN	Actions
My user	***** Show Copy Exp: 05/2027	******** Show Copy Exp: 05/2010	********* Show Copy	Edit | Delete

Back to Dashboard

Identities Screen: Allows users to input information about passwords, licenses, names, and a SSN. This is securely stored using masking to protect sensitive information.

Secure Notes

- **success:** Secure note saved.

Add Secure Note

Title:

Content:

Save Note

Saved Notes

Title	Content	Actions
Test	This is a test	Edit | Delete

Back to Dashboard

Secure Notes: allows users to save secure notes stored in the mypass application.

Password Generator

- **success:** Password generated.

Length:
12

- [x] Include Uppercase
- [x] Include Lowercase
- [x] Include Digits
- [x] Include Symbols

Generate Password

Generated Password

BhzcSW(dsFhk

Back to Dashboard

Password generation tool: Allows users to generate a random password based on the configuration settings of their choosing.

References:

https://www.sqlalchemy.org/
https://sqlite.org/
https://flask.palletsprojects.com/en/stable/
https://www.python.org/
https://bitwarden.com/
https://www.w3schools.com/html/