University of Michigan Dearborn

# MyPassProject Report

Moria Almadrahi, Layal Saad, Sara Kassem

CIS-476-001

Dr.Zhiwei Xu

November 28, 2025

# Table of Contents

# 1. Introduction

The MyPass Password Manager is a secure, web-based application designed to help users safely store, organize, and manage their sensitive information. This includes passwords, credit cards, secure notes, identity documents, and recovery information. The goal of this project is to demonstrate strong knowledge of software engineering design patterns, secure programming principles, data abstraction, modular architecture, and real-world application design using PHP, MySQL, HTML/CSS, and JavaScript.

MyPass addresses the real-world problem of unsafe credential storage. Many users reuse passwords, store them in unencrypted text files, or lose access to accounts due to weak organization and poor security practices. MyPass solves these issues through:

- Strong encryption
- Secure vault access
- Clean data validation
- Multiple design patterns to ensure modularity
- Structured recovery workflows
- Password generation tools
- Expiration alert systems

This report documents the complete design, architecture, security implementation, and testing of the MyPass system, as well as detailed diagrams and feature descriptions.

# 2. System Overview

MyPass provides a full password vault system with the following features:

Core Features

- User registration and login
- Secure session handling
- Add, edit, delete vault items

- Store passwords, notes, IDs, cards
- Password generator
- Password strength validation
- Security question recovery
- Sensitive data masking / unmasking
- Detailed error handling
- Database-backed storage
- Real-time warnings via observer alerts

Design Goals

- Maintain high security using strong coding practices
- Demonstrate software engineering design pattern mastery
- Build a scalable, modular architecture
- Provide clean separation of logic
- Emphasize usability while maintaining safety
- Follow industry best practices

MyPass closely resembles real password managers (LastPass, 1Password, Bitwarden) but is implemented using minimal tools to showcase pure software engineering logic.

## 3. System Architecture

The system uses a modular MVC-like architecture with clear separation between:

Core Components

- SessionManager (Singleton)

Controls authentication states, login, logout, and validation.

- DBConnection

Provides PDO-based connection to MySQL, ensuring secure communication with the database.

- VaultItem

Represents a generic stored item (login, card, note, ID).

- Encryption Utility

Handles hashing and verification of passwords.

- Controllers

Handle CRUD operations and route interaction.

- Validation Handlers

Used through the mediator & chain patterns.

How the Architecture Works

1 A request enters (login, create item, edit item).

2 SessionManager validates authentication.

3 DBConnection retrieves/stores data.

4 Proxy screens sensitive fields.

5 Mediator coordinates input validation.

6 Chain of Responsibility checks password or recovery rules.

7 Observer checks for weak or expired data.

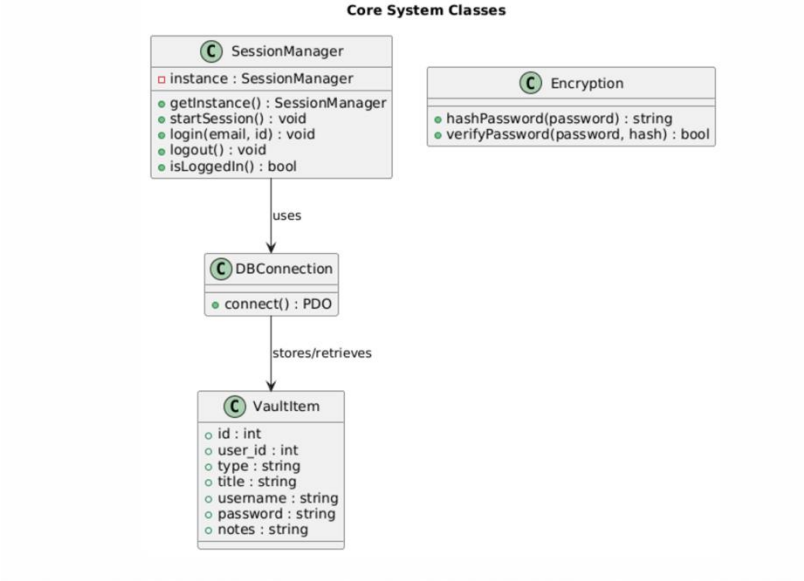
8 Results are displayed through the UI.

## 4. UML Diagrams

The system includes multiple UML diagrams that represent system interaction and structure.

Class Diagrams-

Core Systems Classes:



Description-

SessionManager (Singleton)

Controls all login/logout operations and ensures only one session handler instance exists.

DBConnection

Handles PDO database connection for all CRUD actions.

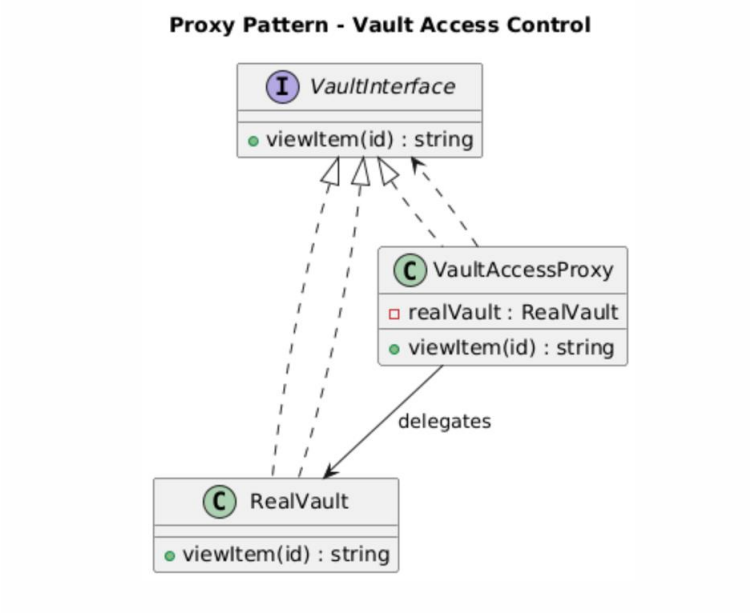
Encryption

Provides password hashing and verification functions.

VaultItem

Represents stored passwords, notes, or cards inside the user's vault.

Proxy Patterns-

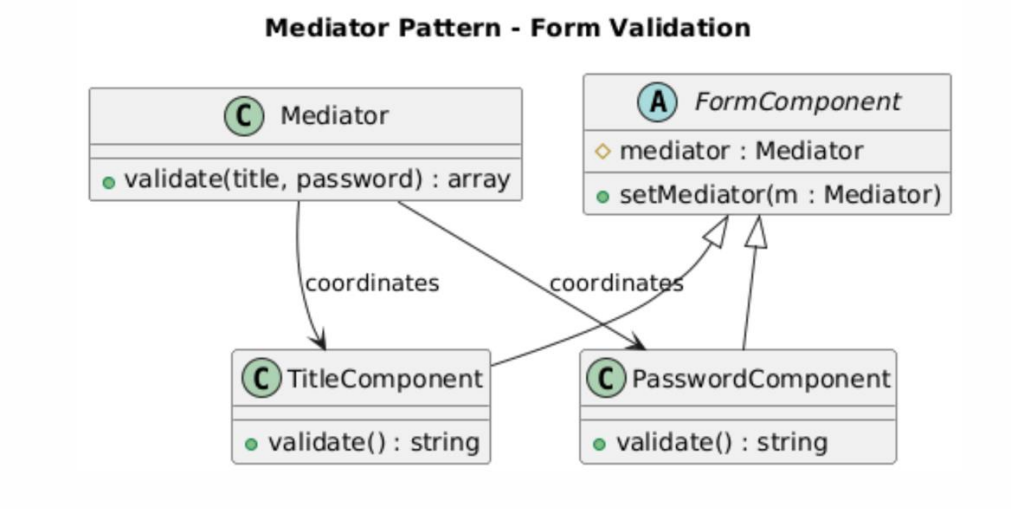


Description-

Proxy Pattern (VaultAccessProxy)

The proxy restricts access to vault items by checking whether the logged-in user is authorized to view the item. If allowed, it forwards the request to RealVault. If not, it blocks the request with an error. VaultAccessProxy is also responsible for masking and unmasking sensitive fields. When the UI first requests a vault item, the proxy returns a version where passwords, credit card numbers, CVV, and identity numbers are replaced with "••••" placeholders. Only when the user explicitly clicks "Show" and the active session is still valid does the proxy fetch and return the unmasked values.

Mediator Pattern-

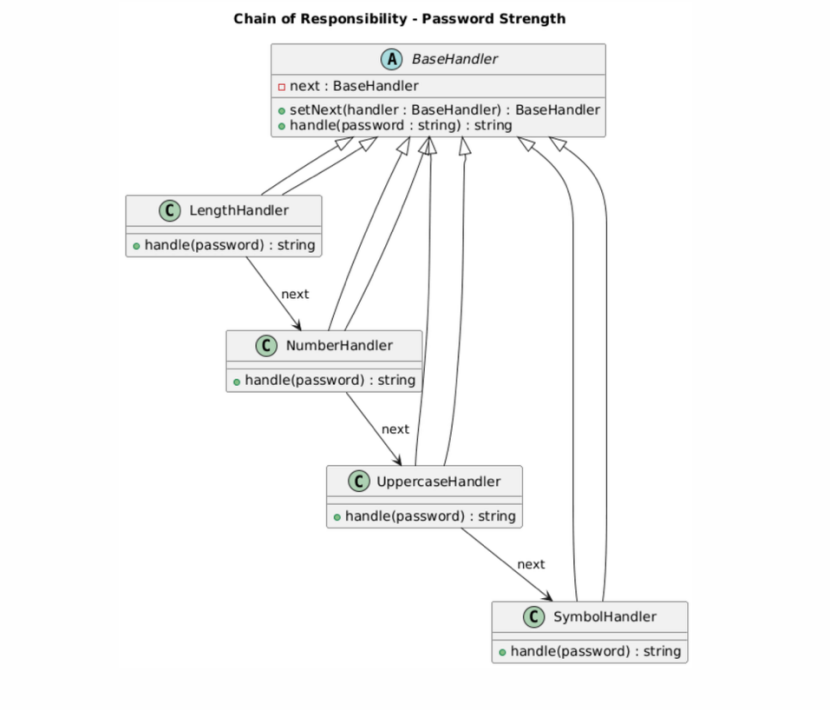



Description-

Mediator Pattern

The mediator centralizes all validation logic.
TitleComponent checks for empty titles, while PasswordComponent checks formatting. The mediator gathers all errors and sends them to the UI.

Chain of Responsibility-




Description-

Chain of Responsibility Pattern

Each handler checks one password requirement:

LengthHandler → 8+ characters

NumberHandler → at least one digit

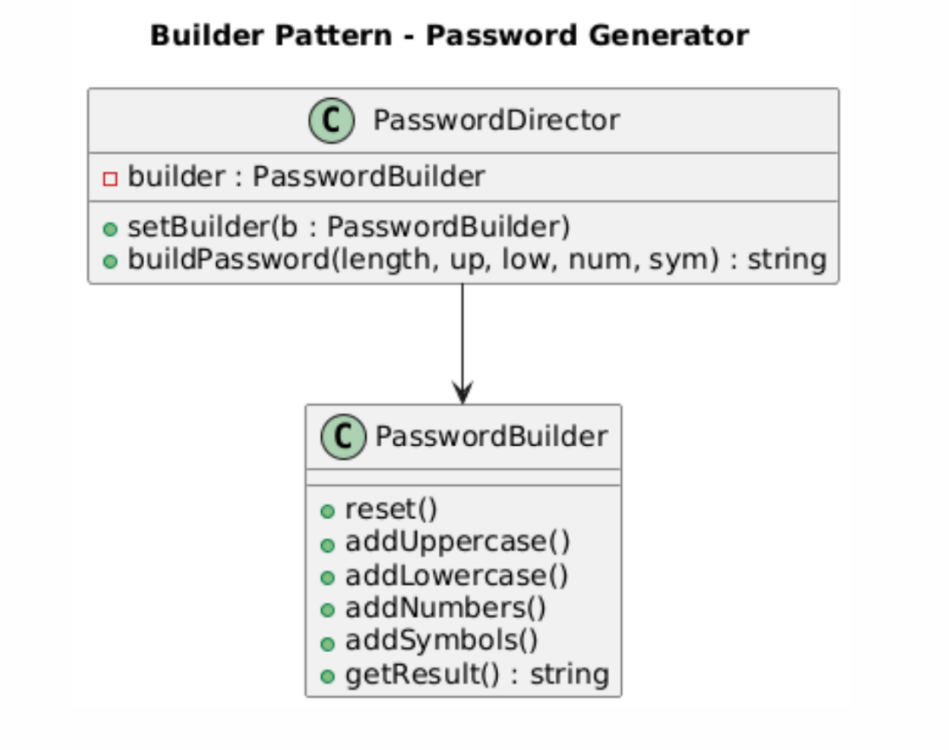
UppercaseHandler → at least one uppercase letter

SymbolHandler → at least one special symbol

If one fails, the chain stops and reports that error to the UI.

A second chain is used for master password recovery, built from Question1Handler → Question2Handler → Question3Handler. Each handler validates one of the three stored security question answers. If all three handlers succeed, the chain allows the user to reset their master password; if any handler fails, the chain stops immediately and returns a recovery error. This directly satisfies the requirement of using the Chain of Responsibility pattern to secure master password recovery based on the three security questions in the users table.

Builder Pattern-

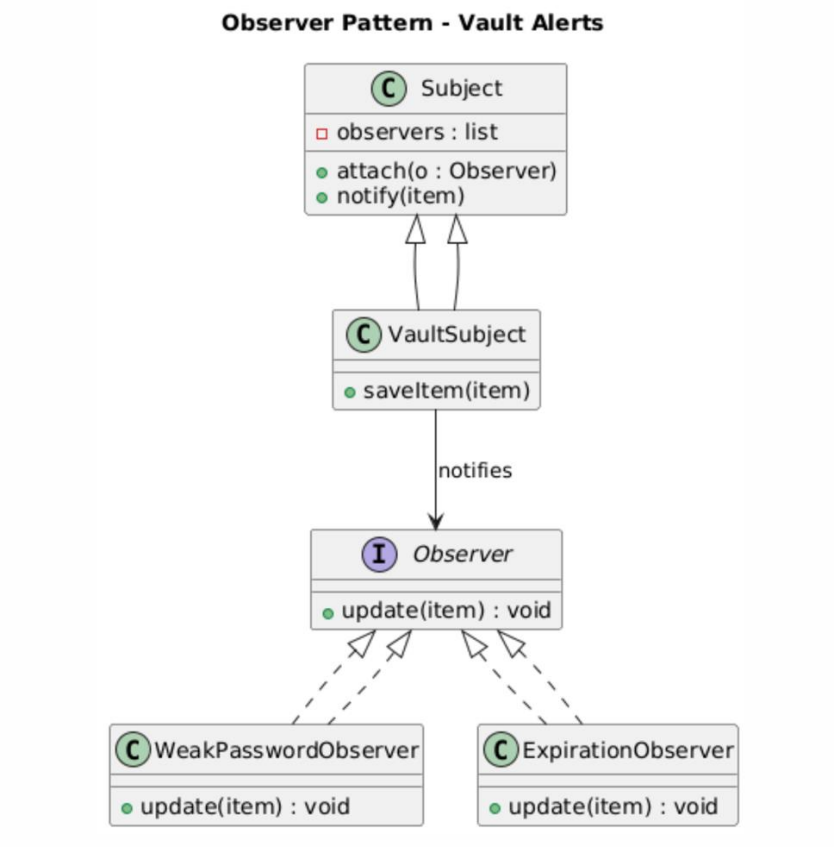



Description-

Builder Pattern

Allows users to generate a customizable strong password.
The director chooses which components to include (uppercase, numbers, symbols), while the builder assembles the final password.

Observer Pattern-




Description-

Observer Pattern

The vault subject notifies observers whenever a new item is stored. WeakPasswordObserver checks the strength of each new password and shows a warning banner when it is weak. ExpirationObserver reads stored expiration dates for credit cards, passports, and driver's licenses; when an item is close to or past its expiration date, it triggers a notification inside the vault so the user can update it.

## 5. Design Patterns

MyPass implements five major software design patterns. Each pattern supports a critical part of the system.

5.1 Proxy Pattern – Sensitive Data Access

The Proxy protects sensitive information such as:

- Passwords
- Credit card numbers
- ID numbers

Behavior

- Vault values appear masked (•••••).
- Only when the user clicks "Show" does the Proxy verify:
  - Active session
  - Correct user ownership
  - No expired session
  - No unauthorized access

If valid → Proxy reveals the accurate decrypted value to the UI.

5.2 Mediator Pattern – Input Validation

Instead of scattering validation logic, the Mediator coordinates:

- Title validation
- Password formatting
- Card number rules
- Identity field checks
- Note-length validation

Each component (TitleComponent, PasswordComponent, etc.) reports to the mediator, which

5.3 Chain of Responsibility – Password Strength & Account Recovery

A. Password Strength Chain

Each handler validates one rule:

1. LengthHandler – minimum 8 characters
2. NumberHandler – must include digit
3. UppercaseHandler – must include uppercase
4. SymbolHandler – must include symbol

If a rule fails → chain stops and returns error.

B. Recovery Question Chain

3 questions must be answered correctly in order.

If any answer fails → user cannot reset password.

5.4 Builder Pattern – Password Generator

Users can select:

- Length
- Uppercase letters
- Lowercase letters
- Symbols
- Numbers

The Builder Pattern constructs the final password through the PasswordDirector combining modules.

5.5 Observer Pattern – Security Alerts

Two observers monitor all vault items:

WeakPasswordObserver

Warns users when stored passwords:

- Are too short
- Lack digits
- Lack uppercase
- Lack symbols

ExpirationObserver

Alerts when identity or card information is:

- Already expired
- Expiring soon

## 6. Database Design

MyPass uses a two-table relational database.

6.1 users Table

Stores:

- user_id
- email

- hashed password
- security question 1
- security question 2
- security question 3

This ensures secure login and recovery.

6.2 vault_items Table

Stores:

- item_id
- user_id
- type (login, card, note, ID)
- title
- username / note / card number / fields
- masked/unmasked password
- timestamps

## 7. Security Implementation

Key Security Features

- Strong hashing for passwords
- Secure session handling
- Proxy-protected data access
- Input validation using mediator
- Recovery verification using chains
- Alert system via observers
- SQL-safe prepared statements
- No plaintext passwords stored
- Masking/unmasking system

Hashing

Uses strong hashing (password_hash, password_verify).

Session Hardening

- Regenerated IDs
- No data exposed without login
- Expired sessions force logout

## 8. Testing Strategy

Testing included manual testing + automated logic testing.

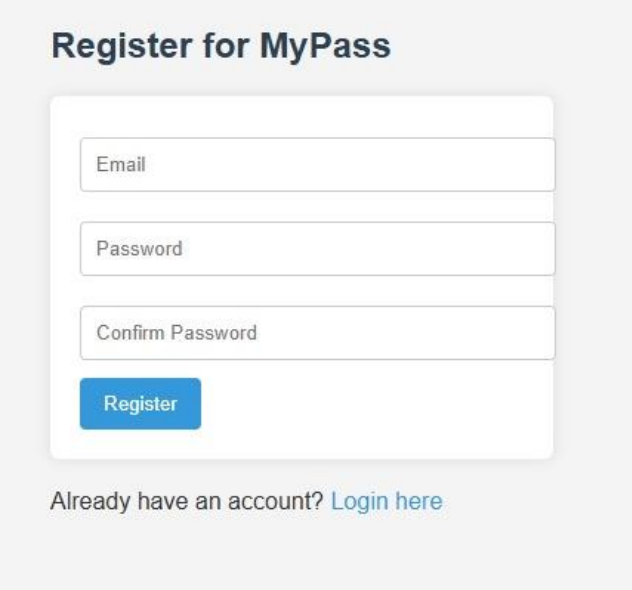
Manual Testing Included

- Registration edge cases
- Login failures
- Vault CRUD activities
- Masked/unmasked fields
- Weak password warnings
- Expired ID warnings
- Recovery question errors
- Password generator output

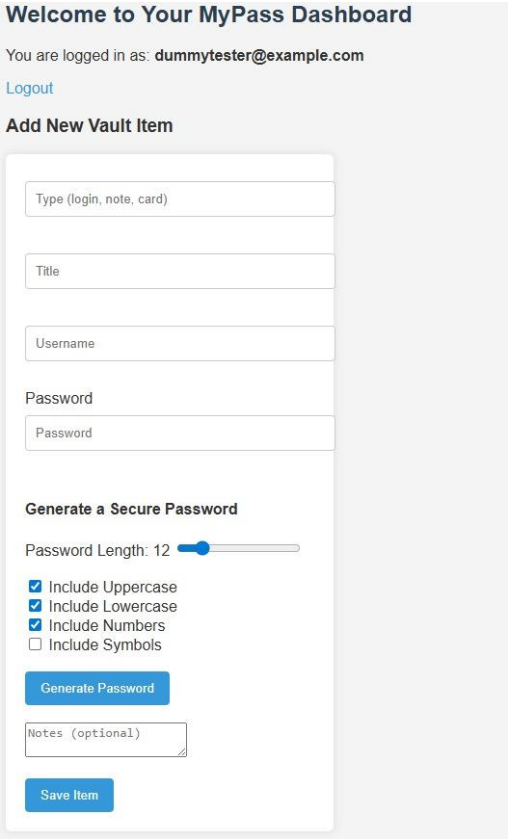
Automated Testing

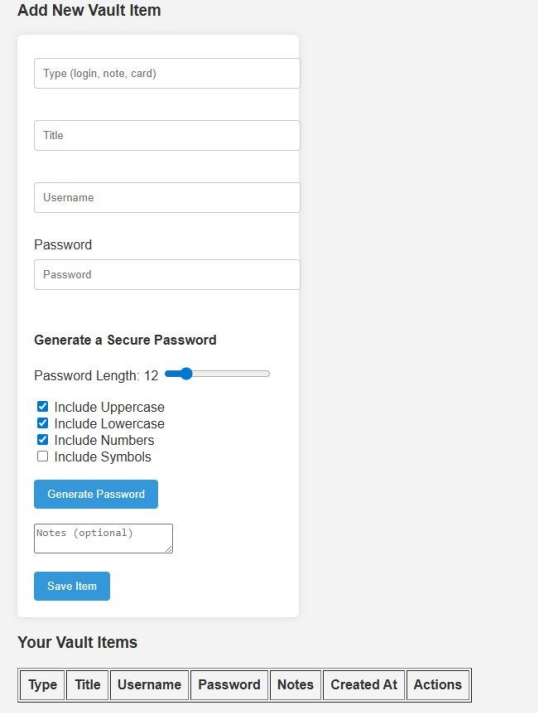
- Validator tests
- Chain tests
- Pattern integrity tests
- Database insert/select tests

# 9. Screenshots

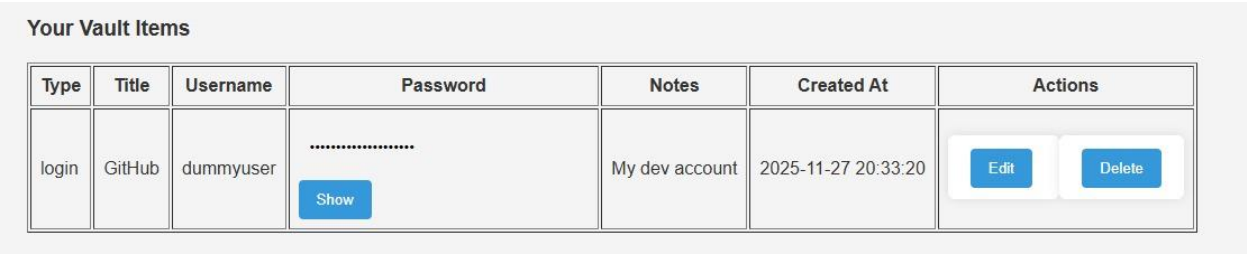



-Register Page(Empty State)

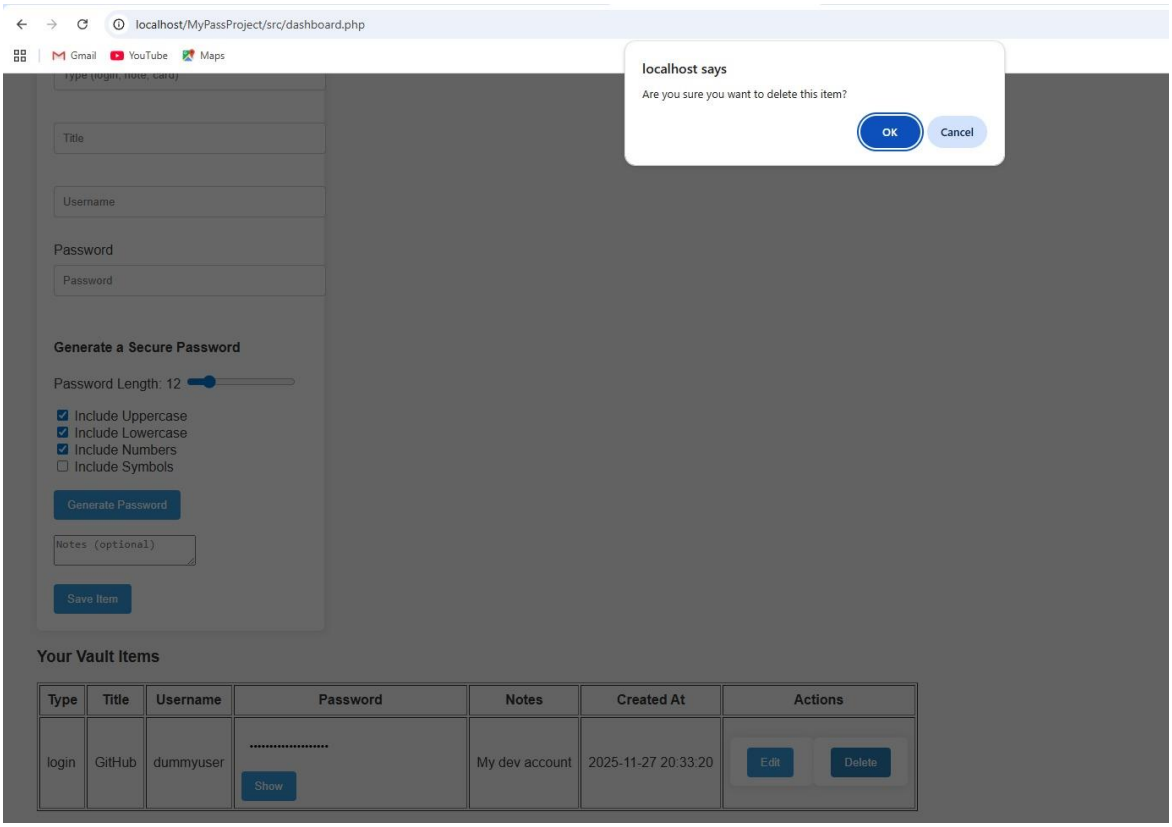


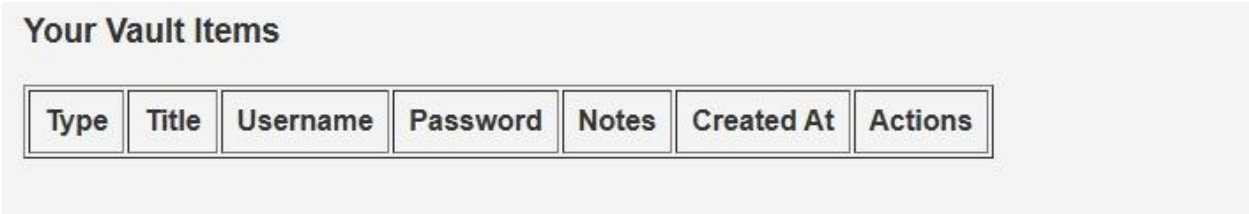



- (these two picture back to back): Dashboard View (After Login, Before Data Entry)

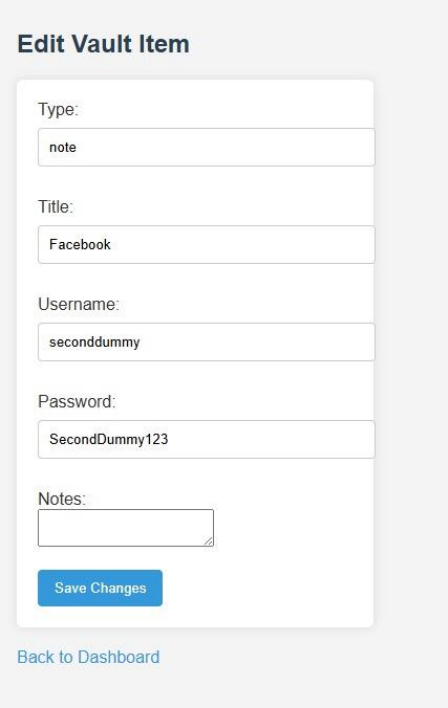


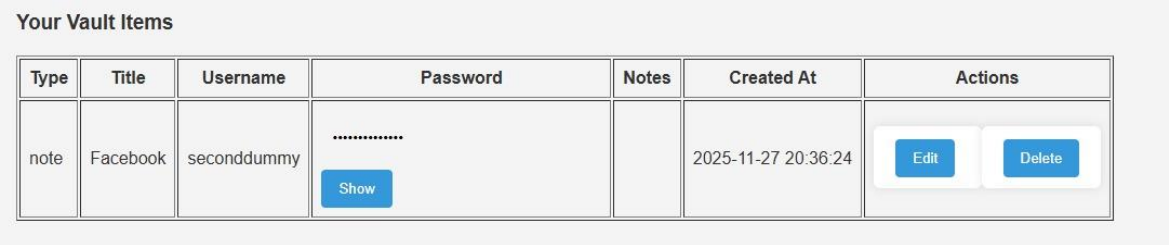

- Create form (success state) and List view with item shown

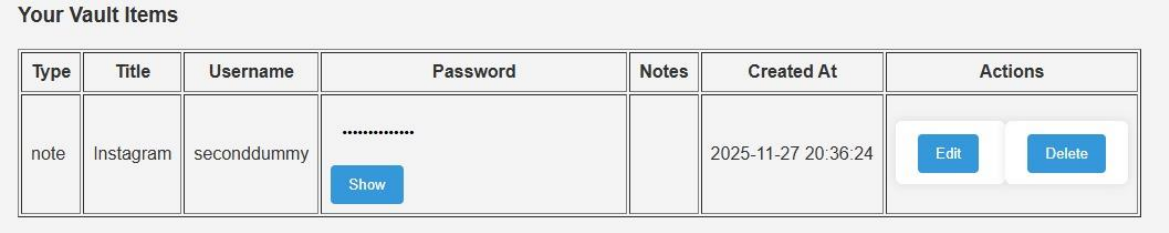


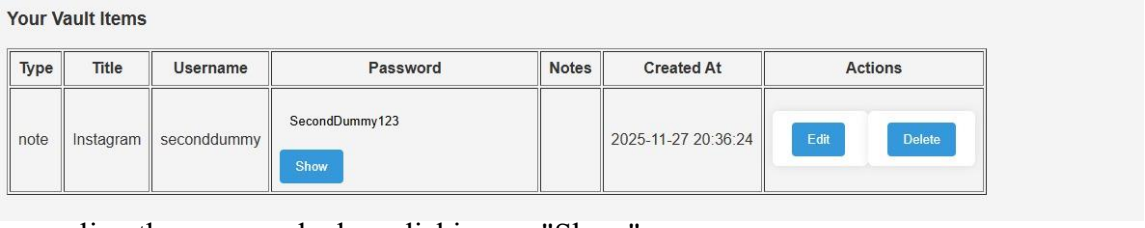

-(these 2 pictures back to back): Delete Functionality and vault item was deleted



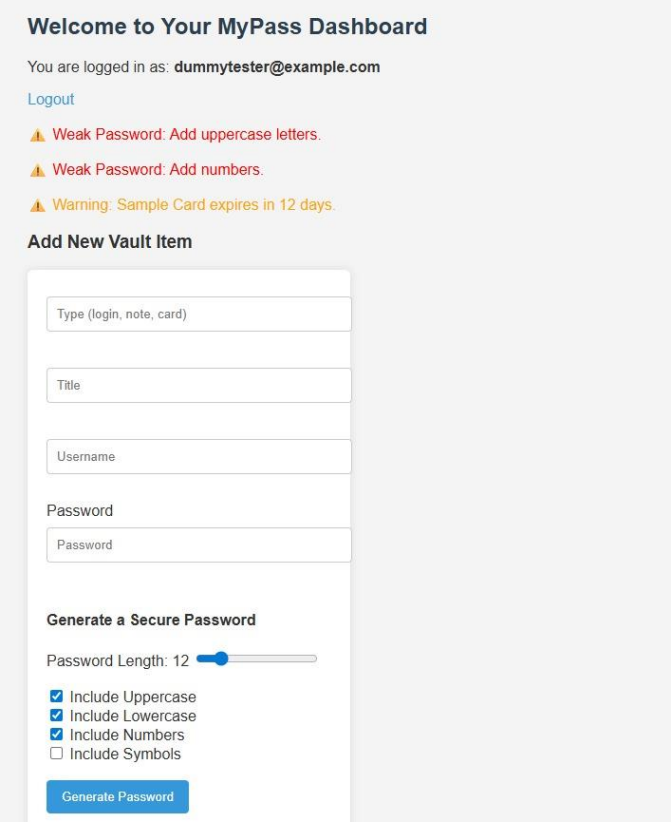



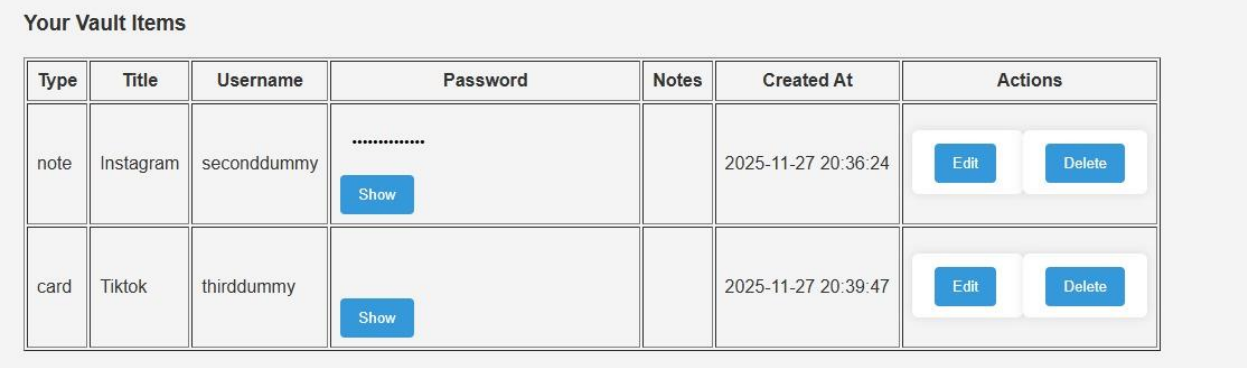




- (these 3 pics back to back): Edit functionality with the form pre-filled with values, then the edited item shows now updated which is Instagram



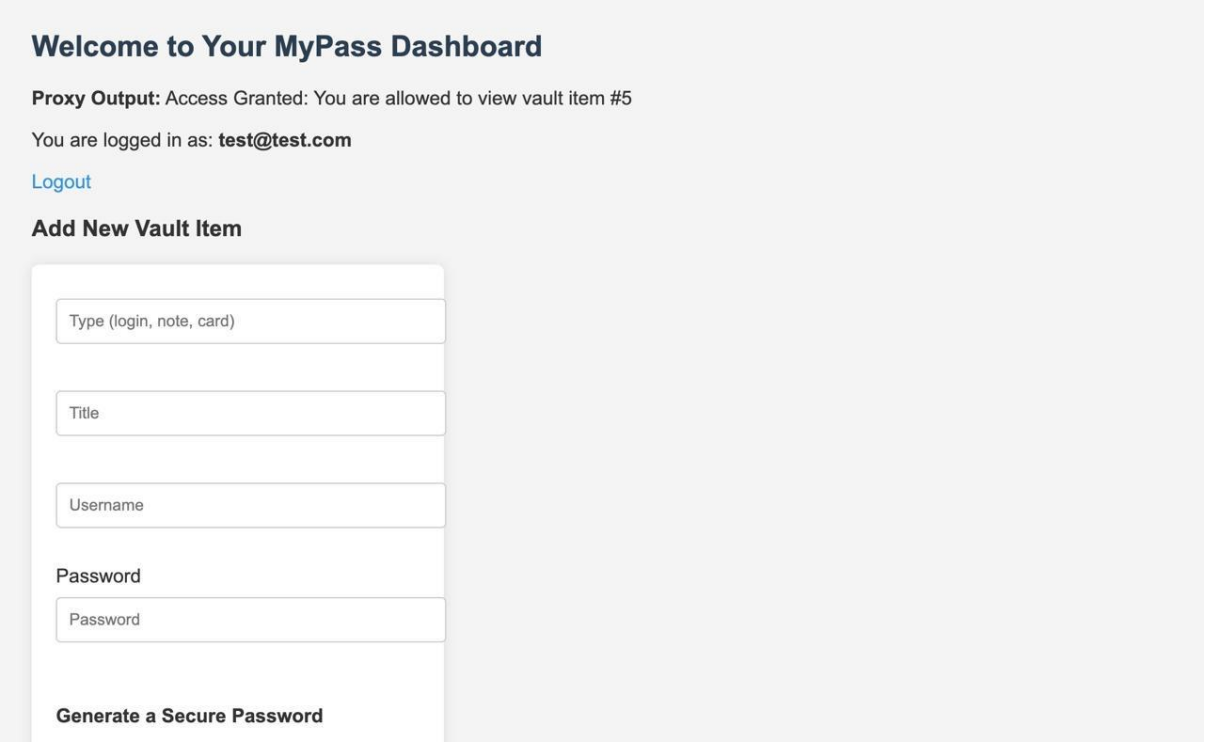

-revealing the password when clicking on "Show"



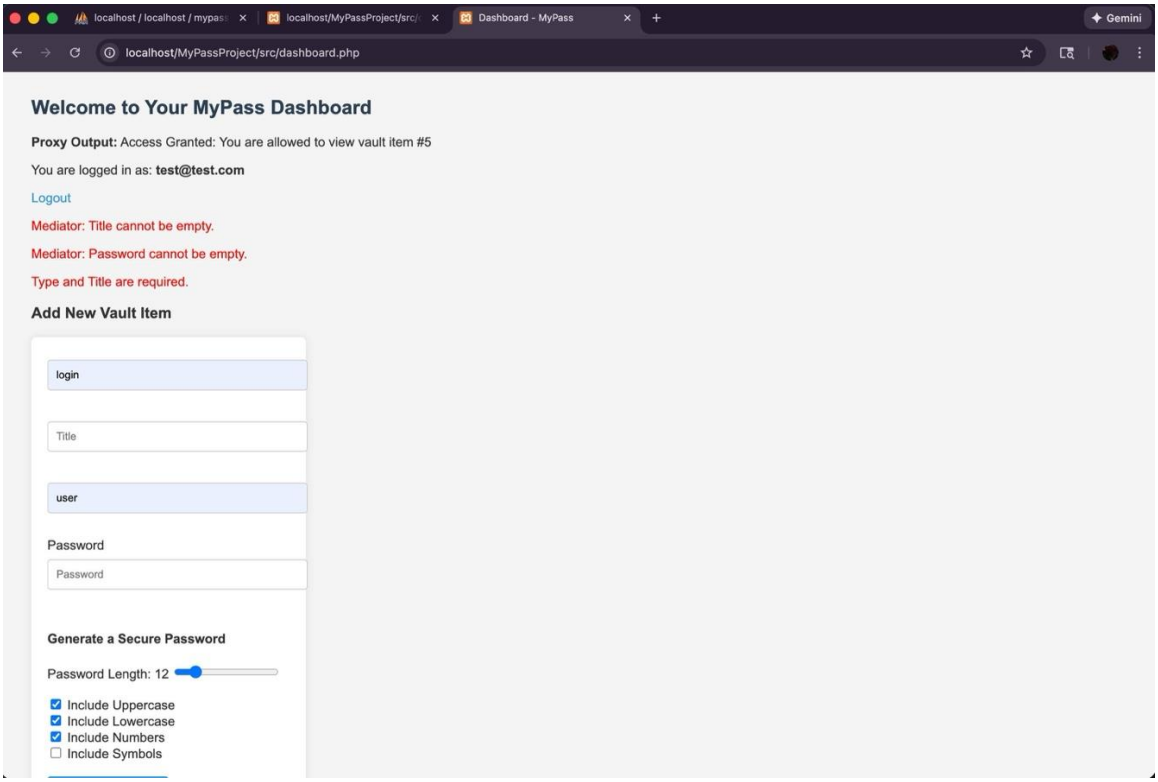

- (2 pics back to back): The first one displays weak password alerts (missing uppercase and numbers), shows expiration warning for a card that will expire in 12 days, shows current user email at the top, confirms that the Observer pattern is actively monitoring vault items and triggering real-time feedback. The second screenshot shows updated vault list with: One "note" type entry (Instagram), One "card" type entry (Tiktok), proper timestamps, password masking, and Edit/Delete buttons are working , confirms new items were added successfully and are being tracked for weak/expiring data.

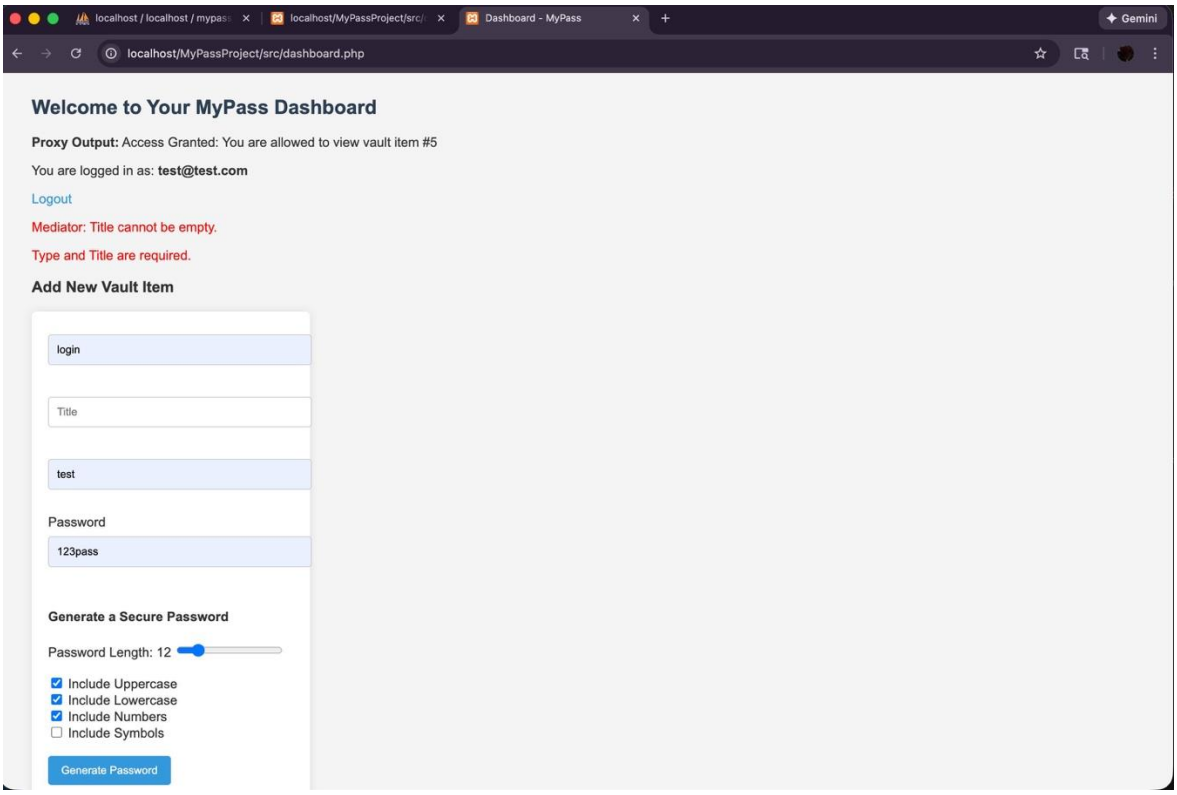
Proxy-




Shows the page confirming access to a vault item. The proxy checks if the user is allowed before showing data.

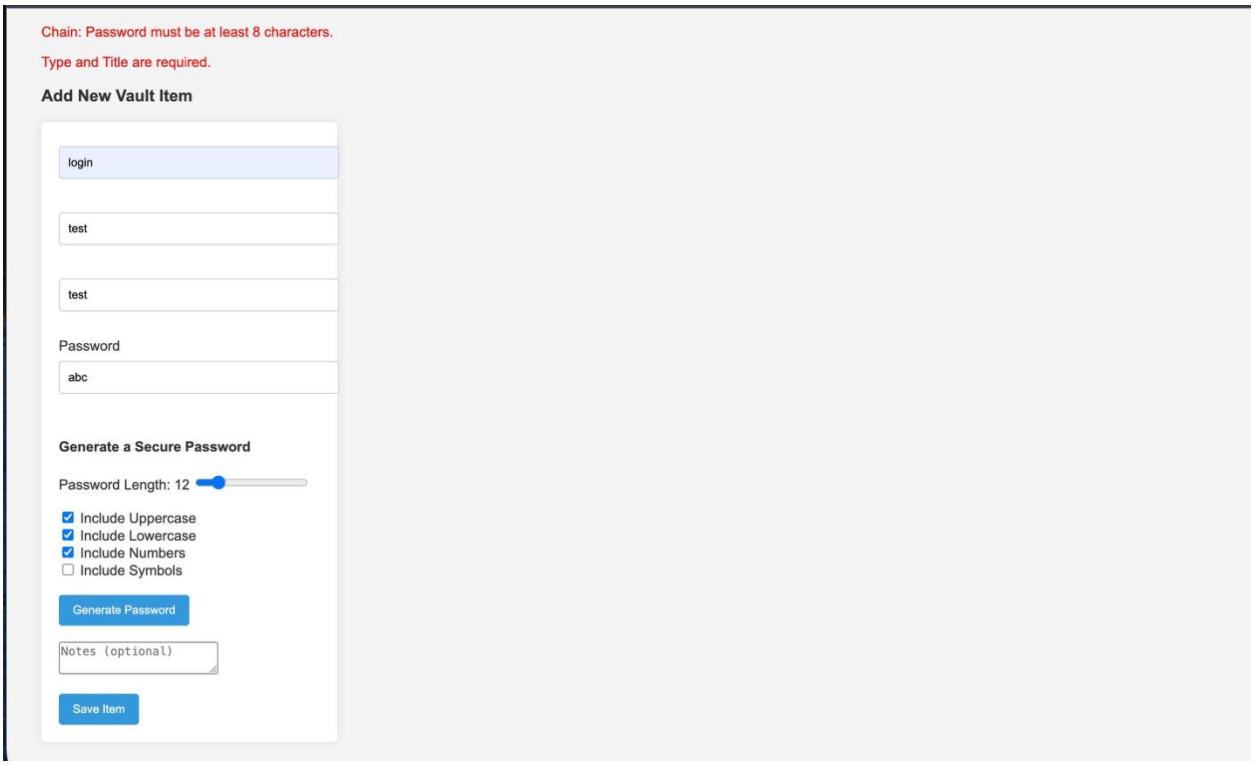
Mediator-

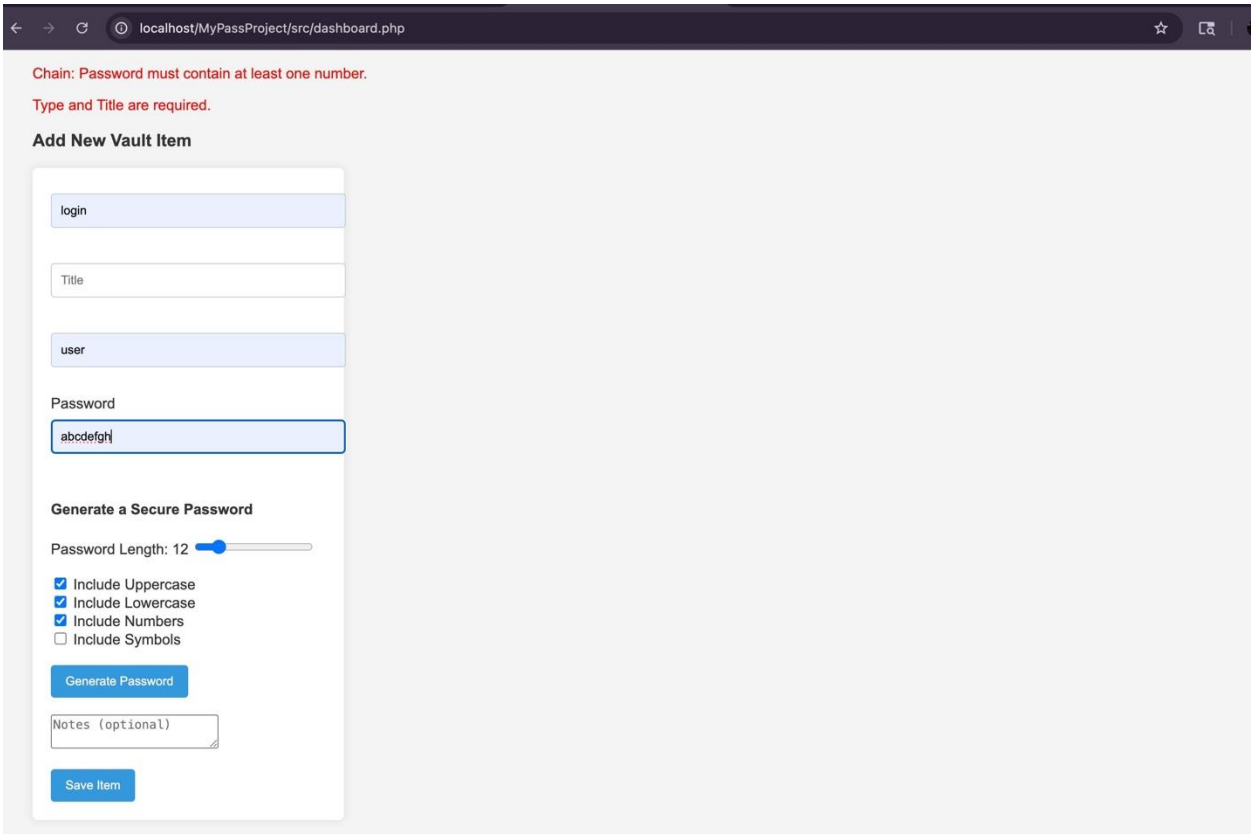


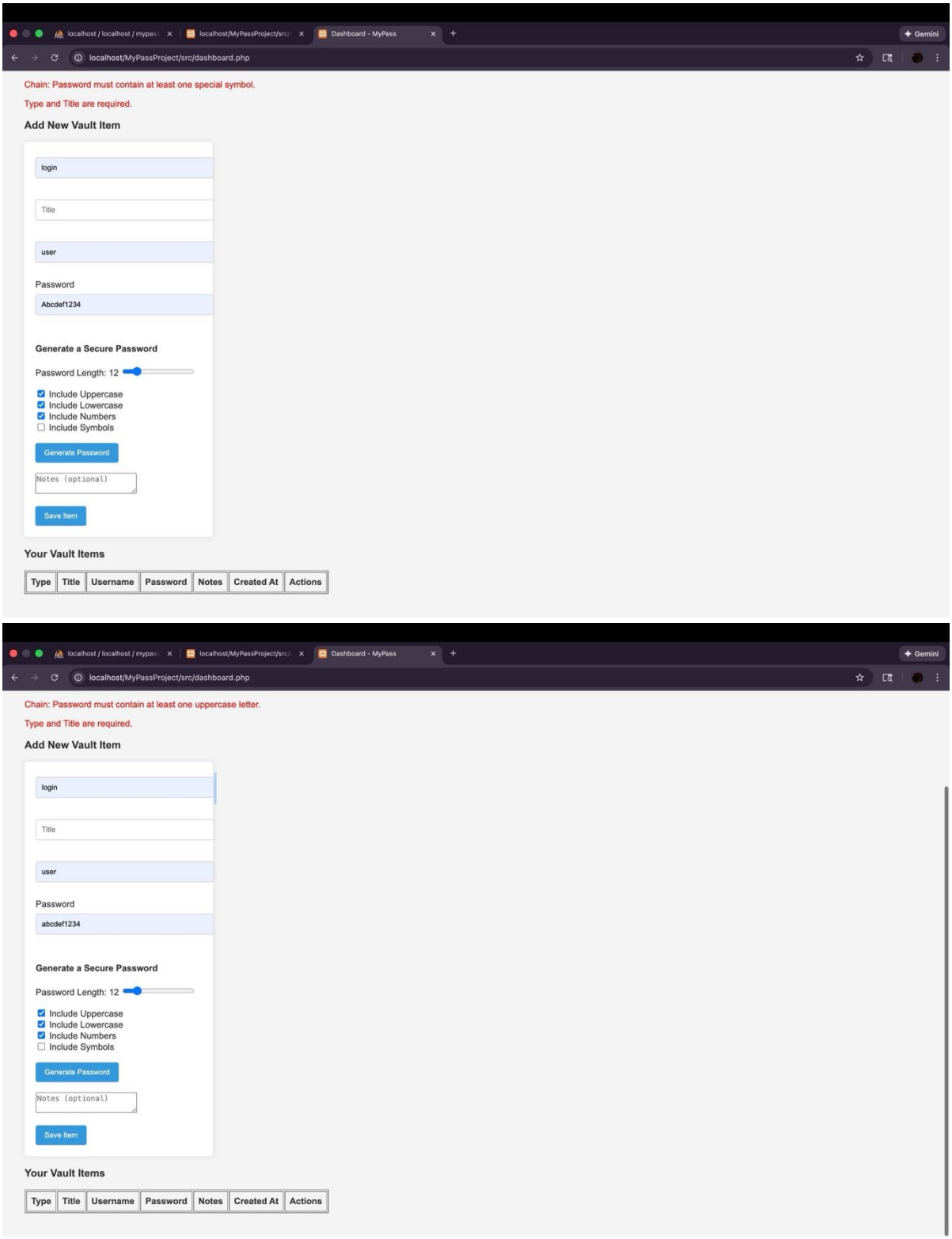




Shows form validation errors generated by the mediator, such as empty title or empty password.

Chain of Responsibility-






Shows password errors triggered by the chain validator, each message coming from a different rule (length, number, uppercase, symbol).

## 10. Conclusion

MyPass successfully demonstrates mastery of software engineering principles, secure system design, and pattern-driven development. The system uses five major design patterns, each applied thoughtfully to solve real security and architecture problems. It implements secure storage, encrypted authentication, and dynamic validation workflows while ensuring clean modular code and maintainability.

The final system is a functional, secure password manager that reflects industry practices and academic rigor.

## 11. References

Design Patterns: Elements of Reusable Object-Oriented Software, Gamma et at.,

Richard Helm, Ralph Johnson, and John Vlissides Addison-Wesley,