SecureVault

An Object-Oriented Password Manager with a Graphical User Interface

1. Objective & Problem Statement

Objective

This project proposes the design and implementation of SecureVault, a desktop password management application. The primary goal is to apply and demonstrate core Object-Oriented Programming (OOP) principles to build a secure, reliable, and user-friendly system. Using C++, the project will feature a fully graphical user interface (GUI) rendered with OpenGL, a lightweight XOR encryption module for data security, and file-based persistence to ensure user data is saved between sessions.

Problem Statement

In the digital age, users manage a vast number of online accounts, each requiring a unique and strong password. The common pitfalls of password management—using weak passwords, reusing passwords, or storing them insecurely (e.g., in plain text files)—create significant security risks. SecureVault aims to solve this by providing a centralized and encrypted repository for credentials, accessed through an intuitive graphical interface, thereby promoting better security hygiene.

2. Core OOP Concepts in Practice

SecureVault is fundamentally an object-oriented project and is built on the four pillars of OOP.

Encapsulation

Data and the methods that operate on that data are bundled together into classes. For example, a Credential object holds the website, username, and password, and only allows access through public methods like getters and setters. This prevents direct, uncontrolled access to sensitive data and hides the internal complexity.

Encapsulation means keeping data safe inside objects and only letting it be changed in controlled ways.

Abstraction

The system hides complex implementation details behind simple interfaces. The UIManager manages a collection of UIElement objects. When the manager calls the `draw` method on an element, it does not need to know how to draw a button versus a text field; it only needs to know that the element is drawable. This simplifies the high-level logic.

Abstraction means showing only the important features and hiding the details.

Inheritance

To reduce code duplication and create a logical hierarchy, inheritance is used. A base UIElement class defines common properties like position and size and behaviors like `draw` and `onClick`. Concrete classes such as Button, TextField, and Panel inherit from UIElement and implement their specific functionalities.

Inheritance lets new classes use features from existing classes.

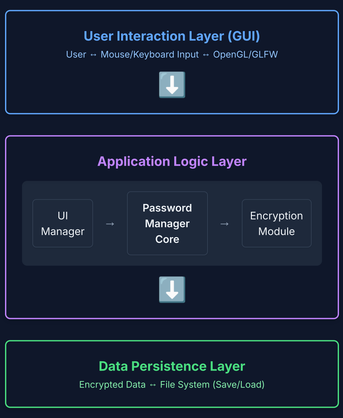
Polymorphism

This allows objects of different classes to be treated in a uniform way. The UIManager can hold a list of UIElement pointers. When it iterates through this list and calls `draw` on each element, the correct `draw` method for a Button, TextField, or other element is executed at runtime. This makes the UI rendering engine flexible and easily extensible.

Polymorphism means using one interface to work with different types of objects.

3. System Architecture

A visual representation of SecureVault's component layers and data flow.



4. Key Features & Scope

Key Features

Master Authentication
Secure login screen to authenticate the user.

Full CRUD Functionality
Create, Read, Update, and Delete password entries.

Multi-Field Entries
Stores website, username, password, and backup code.

XOR Encryption
Encrypts all credentials stored in the data file.

Persistent Storage
Saves credentials to a local file for persistence.

Fully Interactive GUI
All interactions handled through a custom OpenGL interface.

Scope

In Scope

- ✔ Complete GUI using OpenGL
- ✔ XOR encryption for data at rest
- ✔ File-based data persistence
- ✔ Master login mechanism
- ✔ Full credential lifecycle management (CRUD)
- ✔ Comprehensive error handling via GUI feedback

Out of Scope

- ✖ Network synchronization
- ✖ Multi-user support
- ✖ Advanced encryption (e.g., AES, RSA)
- ✖ Cloud storage integration
- ✖ Browser or mobile integration
- ✖ Biometric authentication

5. Project Progress

You can follow the development of SecureVault and review the source code by visiting my project repositories on GitHub.

GitHub Repository
github.com/tijulkabir

Personal Website
tijulkabir.me

6. Submission Details

Proposer (Student)

Name: Tijul Kabir Toha

Student ID: 240113

Email: toha.240113@s.pust.ac.bd

GitHub: tijulkabir

Website: tijulkabir.me

Instructor

Name: Dr. Md. Toukir Ahmed

Email: toukirahmedreal@pust.ac.bd

Assistant Professor
Department of Computer Science and Engineering
Pabna University of Science and Technology
Pabna, Bangladesh