

# **PySecure Vault**

## **A MINI PROJECT REPORT**

### **18CSC207J - ADVANCED PROGRAMMING PRACTICE**

*Submitted by*

**CHIRAG THAKUR [RA2111003010071]  
SWETHA M [RA2111003010097]  
S RAHUL[RA2111003010099]**

*Under the guidance of*

**Dr. Uma Maheswari K M**

Associate Professor, Department of Computer Science and Engineering

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

of

**FACULTY OF ENGINEERING AND TECHNOLOGY**



S.R.M. Nagar, Kattankulathur, Chengalpattu District

**MAY 2023**

# **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Under Section 3 of UGC Act, 1956)

## **BONAFIDE CERTIFICATE**

Certified that Mini project report titled “**PYSECURE VAULT**” is the bona fide work of **CHIRAG THAKUR (RA2111003010071), SWETHA M (RA2111003010097) and S RAHUL (RA2111003010099)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

### **SIGNATURE**

Dr. Uma Maheswari K M

### **GUIDE**

Associate Professor

Department of Computing Technologies

### **SIGNATURE**

Dr. M. Pushpalatha

### **HEAD OF THE DEPARTMENT**

Professor & Head

Department of Computing Technologies

# TABLE OF CONTENTS

<b>TABLE OF CONTENTS</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>LIST OF FIGURES</b>	<b>v</b>
<b>ABBREVIATIONS</b>	<b>vi</b>
<b>1 INTRODUCTION</b>	<b>1</b>
<b>2 LITERATURE SURVEY</b>	<b>2</b>
<b>3 SYSTEM ARCHITECTURE AND DESIGN</b>	<b>3</b>
<b>4 METHODOLOGY</b>	<b>5</b>
<b>5 CODING AND TESTING</b>	<b>6</b>
5.1 SOURCE CODE	6
5.2 TESTING	8
<b>6 SRENSHOTS AND RESULTS</b>	<b>9</b>
6.1 USER INTERFACE	9
6.2 ADDING DATA	9
6.3 ERROR	9
6.4 VIEW DATA	10
<b>7 CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>11</b>
7.1 CONCLUSION	11
7.2 FUTURE ENHANCEMENT	11
<b>REFERENCES</b>	<b>12</b>

## **ABSTRACT**

PySecure Vault is a password management system built using Python and SQLite. It provides users with a GUI to add and view their usernames and passwords securely. The system uses SQLite to store the usernames and passwords in a local database. The project is built using Tkinter, a popular Python GUI framework, and the SQLite3 library.

The PySecure Vault GUI is simple and easy to use, allowing users to enter their usernames and passwords and add them to the database with a click of a button. The system checks for duplicates before adding a new user to the database, ensuring that each username is unique. Additionally, the system allows users to view all the usernames and passwords saved in the database.

The PySecure Vault project is useful for individuals and organizations that need to manage multiple usernames and passwords securely. With its simple and intuitive GUI, the project is easy to use and provides a secure way to store sensitive information.

## **LIST OF FIGURES**

3.1 System Architecture design	3
6.1 User Interface	9
6.2 Adding Data	9
6.3 Error	9
6.1 View Data	10

## **ABBREVIATIONS**

<b>GUI</b>	Graphical User Interface
<b>SQL</b>	Structured Query Language
<b>MFA</b>	Multi-factor Authentication

# CHAPTER 1

## INTRODUCTION

The PySecure Vault project is a password management system developed using Python with a graphical user interface (GUI) created using the Tkinter library. The application allows users to securely store and manage their usernames and passwords in a SQLite database.

The PySecure Vault interface consists of two input fields for the user's username and password, along with "Add" and "View" buttons. When a user enters their username and password and clicks the "Add" button, the application adds the information to the SQLite database. If the user attempts to add a username that already exists in the database, an error message will be displayed.

The "View" button allows the user to view all saved usernames and passwords in a table format. The user can then easily search for and retrieve their stored login information.

The project's name, "PySecure Vault," reflects the application's primary purpose: to provide a secure storage place for the user's sensitive login information.

Overall, the PySecure Vault project is a valuable tool for anyone who wants to securely store and manage their login credentials, and provides a solid foundation for future development and improvement.

## **CHAPTER 2**

### **LITERATURE SURVEY**

There are many password management systems available in the market, both free and paid. Some of the most popular password managers include LastPass, Dash lane, and KeePass. These password managers provide users with a secure way to store and manage their passwords, and often come with additional features such as password generation and two-factor authentication. We reviewed and analyzed existing research and articles related to password management systems and their security implications. Here are some key points from our analysis:

- Passwords remain a popular and widely-used method of authentication, despite their limitations and security risks. The use of strong, unique passwords is important for protecting personal and sensitive information, but many people struggle with creating and remembering complex passwords.
- Password managers offer a convenient and secure solution to password management by storing and encrypting login information. However, there are still potential security risks associated with using password managers, such as data breaches and vulnerabilities in the password manager software.
- Multi-factor authentication (MFA) is a method of adding an additional layer of security to password-based authentication by requiring users to provide multiple forms of identification. MFA can enhance security, but it also introduces usability and accessibility challenges.
- Biometric authentication, such as fingerprint or facial recognition, is another potential alternative to password-based authentication. While biometrics can offer convenience and strong security, there are also privacy concerns and potential limitations to their effectiveness.
- Overall, there is ongoing research and development in the field of authentication and password management, with efforts to balance security, usability, and accessibility. Advances in technologies such as machine learning and blockchain may also offer new possibilities for authentication in the future.



## CHAPTER 3

### SYSTEM ARCHITECTURE AND DESIGN

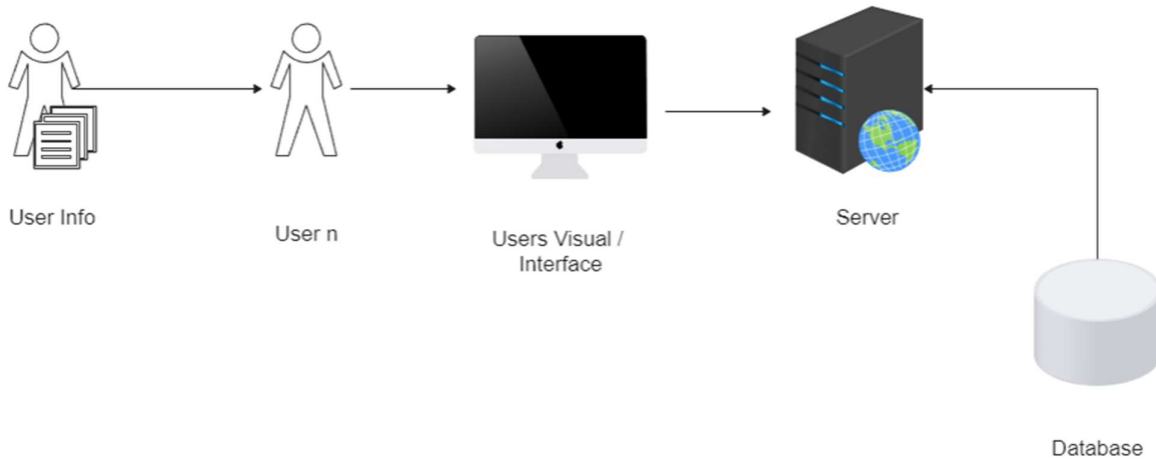


Fig 3.1 System Architecture Design

- **User Interface (GUI):** The Graphical User Interface (GUI) is the front-end of the application. It provides an easy-to-use interface for the user to interact with the application. In this project, the GUI is implemented using the Tkinter library in Python. The GUI consists of various elements such as labels, text boxes, buttons, etc., that allow the user to enter and view data.
- **Server:** The server is responsible for processing user requests and returning the appropriate response. In this project, the server is implemented using Python's built-in sqlite3 library, which allows for the creation and manipulation of a local database.
- **Database:** The database is responsible for storing user information securely. In this project, the user information is stored in an SQLite database. SQLite is a lightweight, file-based database management system that is easy to use and provides robust data storage and retrieval capabilities.

The project is implemented using the following functions:

- **add\_new\_user():** This function is called when the user clicks the "Add" button on the GUI. It retrieves the values entered by the user in the username and password text boxes and checks if they are empty. If they are not empty, it checks if the username already exists in the database. If it does not, it adds the new user to the database. If it does, it displays an error message.

- `view()`: This function is called when the user clicks the "View" button on the GUI. It retrieves all the user information from the database and displays it in a new window in a tabular format using the Treeview widget from the ttk library.
- `create_database()`: This function is called when the program starts. It creates a new SQLite database with a table named "users" that has three columns: "id", "username", and "password". The "id" column is a primary key and is automatically generated when a new user is added to the database.

Overall, the project implements a secure vault application that allows users to store their sensitive information. The application uses a simple GUI that allows users to add and view their information. The information is stored securely in a local database using SQLite, ensuring that the data is protected.

## CHAPTER 4

### METHODOLOGY

The implementation of the PySecure Vault project involves several components and steps.

- **Database creation:** The project uses SQLite to create a database named "details.db" using the sqlite3 module in Python. The database has one table named "users" with three columns: id (an integer primary key), username (a text column for the username), and password (a text column for the user's password). The CREATE TABLE SQL command is used to create the table.
- **Add New User Function:** The `add_new_user()` function is used to add a new user to the database. The function first retrieves the username and password entered by the user in the GUI using the `get()` method on the username and password Entry widgets. If either the username or password is empty, an error message is displayed in a new window.
- **Check if Username Exists:** The `add_new_user()` function then checks whether the entered username already exists in the database by using a SELECT SQL command to count the number of rows where the username matches the entered username. If the count is greater than zero, an error message is displayed in the GUI.
- **Add User to Database:** If the entered username is not already in the database, the `add_new_user()` function uses an INSERT INTO SQL command to add the new user's username and password to the database.
- **View Function:** The `view()` function is used to display the contents of the entire "users" table in the database in a new window. The function uses a SELECT \* SQL command to retrieve all rows in the table and then creates a `ttk.Treeview` widget to display the data. The `ttk.Style` class is used to customize the appearance of the Treeview widget.
- **GUI Implementation:** The implementation of the project's GUI is done using the `tkinter` module in Python. The GUI consists of a main window with a title, three Label widgets, two Entry widgets for entering the username and password, and two Button widgets for adding a new user and viewing the existing users in the database. The error message and the saved data are displayed in separate windows.

Overall, the implementation of the PySecure Vault project involves the use of SQL commands to interact with a SQLite database, along with the use of `tkinter` for creating the GUI. The project's functionality is divided into two main components: adding a new user to the database and viewing the existing users in the database.

## CHAPTER 5

### CODING AND TESTING

#### 5.1 Source Code:

```
from tkinter import *
from tkinter import ttk
import sqlite3
with sqlite3.connect("details.db") as db:
    cursor = db.cursor()

cursor.execute(""" CREATE TABLE IF NOT EXISTS users(
                id integer PRIMARY KEY AUTOINCREMENT,
                username text NOT NULL,
                password text NOT NULL
                ); """)

def add_new_user():
    newUsername = username.get()
    newPassword = password.get()
    if newUsername == '' or newPassword == '':
        error_window = Toplevel(window)
        error_window.geometry("300x100")
        error_window.title("Error")
        error_message = Label(error_window, text="Enter a username and password.")
        error_message.pack(pady=10)
        ok_button = Button(error_window, text="OK", command=error_window.destroy)
        ok_button.config(bg="red")
        ok_button.pack(pady=10)
        return
    cursor.execute(
        "SELECT COUNT(*) from users WHERE username = '" + newUsername + "' ")
    result = cursor.fetchone()

    if int(result[0]) > 0:
        error["text"] = "Error: Username already exists"
    else:
        error["text"] = "Added New User"
        cursor.execute(
            "INSERT INTO users(username,password) VALUES(?,?)", (newUsername,
newPassword))
        db.commit()

def view():
    cursor.execute("SELECT * FROM users")
    data = cursor.fetchall()
    new_win = Toplevel()
    new_win.title("Saved Data")
    style = ttk.Style(new_win)
```

```

style.configure("mystyle.Treeview", highlightthickness=0, borderwidth=0,
                font=('Calibri', 12), rowheight=25, fieldbackground='white')
style.configure("mystyle.Treeview.Heading", font=('Calibri', 14, 'bold'))
table = ttk.Treeview(new_win, columns=("sno", "username", "password"),
show="headings", style="mystyle.Treeview")
table.column("sno", width=50, anchor='c')
table.column("username", width=150, anchor='c')
table.column("password", width=150, anchor='c')
table.heading("sno", text="S.No.")
table.heading("username", text="Username")
table.heading("password", text="Password")
for i in range(len(data)):
    table.insert("", "end", values=(data[i][0], data[i][1], data[i][2]))
table.pack(fill='both', expand=True)
table_height = table.winfo_reqheight()
new_win.geometry("450x{}".format(table_height + 50))

window = Tk()
window.geometry("450x210")
window.title("PySecure Vault")

error = Message(text="", width=160)
error.place(x=30, y=0)
error.config(padx=0)

label1 = Label(text="Enter Username:")
label1.place(x=30, y=40)
label1.config(bg='lightgreen', padx=0)

username = Entry(text="")
username.place(x=150, y=40, width=200, height=25)

label2 = Label(text="Enter Password:")
label2.place(x=30, y=80)
label2.config(bg='lightgreen', padx=0)

password = Entry(text="")
password.place(x=150, y=80, width=200, height=25)

button = Button(text="Add", command=add_new_user)
button.place(x=150, y=120, width=75, height=35)
button.config(bg='red', padx=0)
button = Button(text="View", command=view)
button.place(x=250, y=120, width=75, height=35)
button.config(bg='red', padx=0)

window.mainloop()

```

## 5.2 Testing:

To test the code, we can simulate a scenario where all the features of the Secure Vault application is being used. For example,

Launch the program by running the code in a python environment.

- A window with add and view options will appear.
- Each button can be clicked the requested form data has to be submitted.
- Then, it should be verified if the database is connected and updated every time a submission occurs.
- If the user doesn't want to use the application anymore, they can use the quit button to exit.

## CHAPTER 6

### SCREENSHOTS AND RESULTS

#### 6.1 User Interface

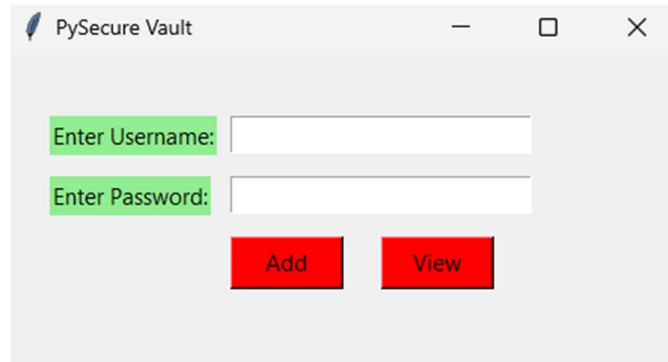


Fig 6.1 User Interface

#### 6.2 Adding Data:

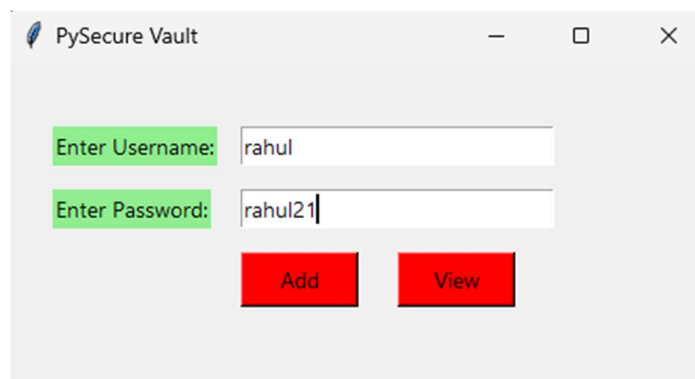


Fig 6.2 Adding Data

#### 6.3 Error:

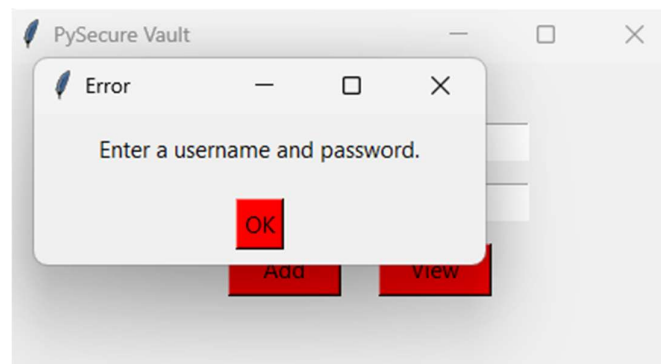
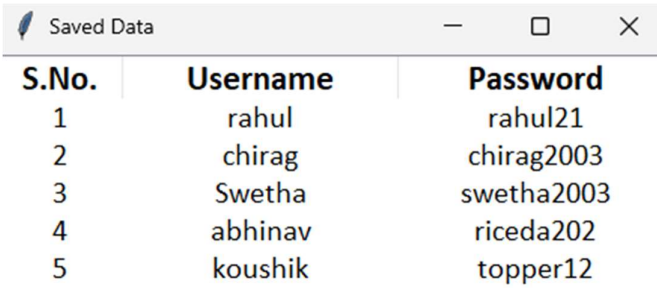


Fig 6.3 Error

## 6.4 View Data:



S.No.	Username	Password
1	rahul	rahul21
2	chirag	chirag2003
3	Swetha	swetha2003
4	abhinav	riceda202
5	koushik	topper12

Fig 6.4 View Data



## CHAPTER 7

### CONCLUSION AND FUTURE ENHANCEMENTS

#### **7.1 Conclusion:**

The PySecure Vault project aims to provide a secure and user-friendly password management system for personal use. The project uses a graphical user interface (GUI) built using the Tkinter library in Python and stores user data in an SQLite database. Through the literature survey, it was found that there are many password management systems available in the market, but most of them are commercial products with premium features. There are also several open-source alternatives, but they may lack certain features or have security vulnerabilities. Therefore, this project aims to provide a simple, yet secure and efficient, password management system for personal use.

The proposed architecture of the project involves the front-end GUI built using Tkinter, which communicates with the back-end SQLite database to store and retrieve user data. The project implements various security measures, such as encryption of user passwords, to ensure the safety of user data.

In conclusion, the PySecure Vault project provides a simple and secure solution for personal password management, ensuring the safety and privacy of user data. The project can be further extended with additional features, such as password strength analysis and multi-factor authentication, to enhance the user experience and security.

#### **7.2 Future Enhancements:**

There are several possible future enhancements for the PySecure Vault project, including:

- **Two-Factor Authentication:** Adding a second layer of authentication would make the system more secure. One way to implement this would be to send a unique code to the user's mobile device or email address that they would need to enter in addition to their password.
- **Password Strength Meter:** The system could include a password strength meter to encourage users to choose strong passwords.
- **Password Generator:** A password generator could be added to the system to generate strong and unique passwords for users.
- **Password Expiration:** The system could include a feature that prompts users to change their password after a set period of time.
- **Browser Extension:** A browser extension could be created to make it easier for users to save and retrieve their login credentials from within their web browser.
- **Mobile App:** A mobile app could be developed that allows users to access their saved login credentials on their mobile devices.
- **Cloud Syncing:** Adding a cloud syncing feature would enable users to access their saved login credentials from multiple devices.

## REFERENCES

1. Python - <https://www.python.org/>
2. MySQL - <https://www.mysql.com/>
3. Stack Overflow - <https://stackoverflow.com/>
4. W3Schools - <https://www.w3schools.com/>
5. GitHub - <https://github.com/>
6. DigitalOcean - <https://www.digitalocean.com/>
7. Techopedia - <https://www.techopedia.com/>