=== SecurePass - Password Manager Project Report ===

Author: Fadzai Chitsinde
Project: SecurePass - Password Manager
Date: January 2026

---

1. Project Overview

SecurePass is a Python-based desktop application designed to enhance personal cybersecurity by generating, checking, and storing secure passwords. It is a demonstration of practical Python programming, cryptography, API integration, and GUI design skills.

1.1 Purpose

- To generate strong, complex passwords for various websites and applications.
- To check whether passwords have been exposed in known data breaches.
- To safely store passwords locally in an encrypted format for personal access.
- To provide a clean and intuitive user interface for managing credentials.

1.2 Target Audience

- Students, professionals, and anyone seeking secure password management.
- Demonstrates portfolio-relevant software engineering, cybersecurity, and Python skills.

---

2. Key Features

1. **Password Generation:**
2. Users can generate strong passwords with customizable length and complexity.
3. Complexity levels include letters only, letters + digits, or letters + digits + symbols.
4. **Leak Checking:**
5. Integration with the `Have I Been Pwned` API to verify whether a password has been compromised.
6. Warns users if the password appears in known breaches and prevents saving unsafe passwords.
7. **Password Storage:**
8. Securely saves passwords locally using `cryptography.Fernet` encryption.
9. Allows retrieval of saved passwords in a separate, scrollable window.

10. **Viewing Saved Passwords:**

11. Users can search through saved credentials by site or application name.

12. Provides a read-only, scrollable view of stored credentials.

13. **Copy to Clipboard:**

14. Users can copy generated passwords directly to the clipboard for quick use.

15. **User-Friendly Interface:**

16. Built with Tkinter for a clean, minimal, and intuitive layout.
17. Buttons grouped logically to reduce clutter: Generate, Check & Save, View Saved, Exit.

---

3. Technical Implementation

3.1 Environment

- **Language:** Python 3.13
- **Libraries:**
- `tkinter` for GUI
- `requests` for API calls
- `cryptography` for encryption
- `hashlib` for SHA1 hashing
- `json` and `os` for file handling

3.2 Architecture

- **Encryption:** Each password is stored locally in a JSON structure, encrypted with a Fernet symmetric key.
- **Leak Checking:** Passwords are hashed with SHA1 and checked against the Have I Been Pwned API range endpoint.
- **GUI:** Single main window for generation and leak checking, separate Toplevel window for viewing saved passwords.

3.3 Data Flow

1. User enters site/app name and optionally username/email.
2. User generates a password using the generator.
3. User clicks 'Check Leak & Save':
4. Password is checked via API.
5. If leaked → user is warned, password is NOT saved.
6. If safe → password is encrypted and stored locally.
7. Users can view saved passwords via a search-enabled, scrollable window.

---

4. Demonstrated Skills

- **Python Programming:** GUI design, modular code, exception handling.
- **Cryptography:** Symmetric encryption and secure password storage.
- **API Integration:** Have I Been Pwned password breach checking.
- **Data Management:** JSON file handling and structured local storage.
- **UI/UX Design:** Clean and user-friendly Tkinter interface with grouped buttons and scrollable display.
- **Cybersecurity Awareness:** Implemented leak checks and secure storage to prevent unsafe password reuse.

---

5. Installation & Usage

1. Create a Python virtual environment and activate it:

```
python -m venv venv
venv\Scripts\activate
```

2. Install required packages:

```
pip install -r requirements.txt
```

3. Run the application:

```
python main.py
```

4. Use the GUI to generate, check, save, and view passwords.

---

6. Future Enhancements

- Optional: Integrate cloud syncing for cross-device password access.
- Add multi-user support with separate encrypted profiles.
- Implement password strength visualization.
- Include auto-fill capabilities for web browsers.

Conclusion: SecurePass demonstrates practical application development skills in Python, cybersecurity best practices, and user interface design. It is a strong portfolio piece showcasing problem-solving, security awareness, and software engineering proficiency.