LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

SECUREME-

“PLATFORM TO SECURE YOUR DAILY LIFE”

DATE : 11 NOV 2025

AMRITANSHU KUMAR (12413909)

RITESH SINGH KUSHWAHA (12410009)

MASTER’S IN COMPUTER APPLICATION
SESSION : (2024-26)

<u>**Students's Declaration**</u>

I hereby declare that the work presented in this report entitled "**SecureMe**" in partial fulfillment of the requirement for the award of the degree of Master of Computer Applications (MCA), submitted to the Department of Computer Applications, Lovely Professional University, Phagwara, is an authentic work carried out over the period from August-2025 to Nov-2025.

This project work has been completed under the guidance of **Mr.Anirban Das** as part of the CA.

The content of this report is the result of my own efforts and has not been submitted for the award of any other degree or diploma in this or any other institution.

Ritesh Singh Kushwaha
Reg. No. : 12410009

Amritanshu Kumar
Reg. No. : 12413909

Student's Signature

Acknowledgement

I would like to express my sincere gratitude to **Lovely Professional University** for providing me the opportunity to undertake this project titled **“SecureMe – Platform to Secure Your Daily Life.”**

I extend my heartfelt thanks to my mentor **Mr. Anirban Das**, **School of Computer Applications**, for their constant guidance, encouragement, and valuable feedback throughout the development of this project. Their expert advice and motivation helped me overcome challenges and complete this work successfully.

I am also grateful to my teachers and friends who offered their time, assistance, and constructive suggestions during different stages of the project. Their support played a significant role in enhancing the quality and functionality of this system.

Finally, I wish to thank my family for their continuous support, patience, and understanding during the entire duration of this project. Their encouragement has been my greatest strength and inspiration.

This project has been a great learning experience, allowing me to apply theoretical knowledge to practical implementation and explore new concepts in data security, cryptography, and web development.

Amritanshu Kumar
Ritesh Singh Kushwaha

Abstract

In today’s digital era, the security of personal data has become a major concern due to the increasing risk of identity theft, data breaches, and unauthorized access. The project **“SecureMe – Platform to Secure Your Daily Life”** aims to provide a unified, lightweight, and user-friendly solution to ensure local data protection through encryption and controlled access.

This system integrates three major functionalities within a single platform — **Folder Lock**, **Password Vault**, and **Secure Notes** — each designed to safeguard sensitive files, credentials, and personal information. Developed using **Python (Flask Framework)**, the application leverages advanced **cryptographic algorithms** such as AES and Fernet to encrypt data, ensuring confidentiality and integrity. The platform also maintains **metadata logs** for all encryption and decryption operations, enabling auditability and transparency in data security.

Unlike cloud-based solutions, SecureMe operates completely on a **local environment**, ensuring that no user data is transmitted or stored externally. The use of Excel files and encrypted Word documents as storage mechanisms makes the system lightweight and easy to deploy without the need for complex database configurations.

This project demonstrates the successful integration of web technology, cryptography, and local storage security, resulting in a practical and effective personal security platform. The system’s modular architecture also allows for future scalability, including the integration of **biometric authentication**, **role-based access control**, and **secure file sharing**.

1. Introduction

1.1 Purpose

The primary aim of the project **[1]** is to create and build a platform that secures sensitive data like personal files, stored passwords and notes. Data security has taken on yet another level of importance in the digital era and identity theft, loss of money or worse of all misuse of confidential information can all happen due to loss of data security. The proposed project will offer a unity based system that incorporates folder locking system, password management, and notes secured with encryption into single software solution.

The platform has been realized using **Flask (Python framework)** and using **[2] cryptographic algorithms, hashing techniques, and secure system (passkey-based authentication)** with regard to the strong protection of user data. The system goes beyond encrypting files and data and actually records metadata entries, which assist in tracking encryption information like algorithm type, hashes and times.

1.2 Scope

The system aims to provide a lightweight yet effective security solution that can be used by any individual user on their personal computer. It provides four core modules:

- **User Login Module**, which authenticates users with a passkey before accessing the system.
- **Folder Lock Module,** which restricts access to selected directories using encryption keys and system-level permissions.
- **Password Vault Module**, which stores login credentials for websites or applications in an encrypted Excel file.
- **Secret Notes Module,** which enables users to create personal notes that are saved as encrypted .docx files.

The system also generates and maintains supporting metadata files (**OFile.xlsx, OPass.xlsx, and ONotes.xlsx)**, which store details of encryption operations performed on the data. Unlike enterprise-level cloud-based solutions, this project is lightweight, completely local, and ensures that no data is exposed outside the user's own device.

2. Overall Description

2.1 Product Perspective

The proposed system functions as a **standalone desktop web application** powered by **Flask.** It runs on the local machine and is accessed through a browser interface. Instead of depending on heavy relational databases, the system uses **Excel-based storage for metadata and vault files**. This decision makes the application lightweight and easier to deploy without requiring advanced setups. The system also integrates directly with the operating system to lock or unlock folders, which means it is not just a web app but also interacts with system-level file permissions.

2.2 Product Features

The key features of the application are:

- Offer a **user-friendly web interface** built with Flask, where all features are easily accessible.
- Login system using a **passkey to ensure that only authorized users** can access the platform.
- Securely **lock and unlock** folders at the system level with unique encryption keys.
- Maintain a password vault, where sensitive credentials are stored in encrypted Excel files.
- Provide a **secret notes feature**, where private notes are encrypted and stored securely.
- Manage metadata for every **encryption or decryption operation**, ensuring transparency and traceability.
- Offer a **user-friendly web interface** built with Flask, where all features are easily accessible.

2.3 User Characteristics

The system is designed for **non-technical** and **semi-technical** users who require secure data storage on their personal machines. Users only need basic knowledge of operating a web browser and understanding of passwords or folder paths. No prior technical expertise in cryptography or system security is necessary, since the application automates these operations in the backend.

2.4 Constraints

The project has some limitations, such as dependency on Python installation and its libraries. The system is initially designed for local use and is not cloud-based. Folder locking and unlocking rely on Windows/Linux permissions and commands, which means cross-platform compatibility must be carefully handled. Another constraint is that the security of the platform depends on the strength of the user's passkey.

2.5 Assumptions and Dependencies

It is assumed that the user will run this application on their personal machine with full administrator rights. It is also assumed that the user will remember their passkey, since it is required for login. The application depends heavily on external Python libraries such as **Flask, Cryptography, OpenPyXL,** and **Python-Docx**, which must all be installed properly for the system to function as expected.

3. System Requirements and Specification

2.6 Functional Requirements

The functional requirements define the essential tasks that the system must be capable of performing:

- User Authentication: The system must provide a secure login mechanism where users can access the application using their passkey. Without successful authentication, no further modules or operations should be accessible.

- Folder Lock/Unlock: Users should be able to select a folder from their local file system and apply encryption (lock) or decryption (unlock) through the web interface. The locked folder should not be accessible directly unless unlocked through the application.

- Password Vault Management: The platform must allow users to securely store, encrypt, and retrieve login credentials inside an Excel file (Pass.xlsx). Users can add, update, or remove credentials, and the system should encrypt all sensitive data.

- Secure Notes: The system must provide functionality for creating, saving, and encrypting personal notes into Word documents (.docx). These notes should be accessible only after decryption.

- Metadata Logging: Every operation (e.g., folder encryption, vault modification, note creation) must be automatically logged into metadata files (OFile.xlsx, OPass.xlsx, ONotes.xlsx). Logs should include details such as operation type, algorithm used, date-time, and unique identifiers.

- Web-Based Interaction: All operations must be performed through the Flask-based web interface. Users should not require any direct command-line interaction to use the system.

2.7 Non-Functional Requirements

The non-functional requirements describe the quality attributes and constraints of the system:

- Security: The system must ensure confidentiality and integrity of sensitive data by implementing strong encryption algorithms (e.g., AES). All passkeys should be hashed before verification to prevent plain-text storage.

- Reliability: Reliability is achieved by maintaining detailed metadata logs, which act as a verification and auditing mechanism. Even if data corruption occurs, metadata logs can help trace back operations.

- Usability: The Flask interface must be simple, clear, and intuitive. Users should be able to navigate through modules easily and perform operations without requiring technical expertise. Forms and buttons must have clear labels and error-handling messages.

- Performance: The system must perform critical operations like folder locking/unlocking, vault encryption, and note management in real time, typically completing within a few seconds. Heavy operations must be optimized to minimize lag.

- Portability: Since the platform is built with Python and Flask, it should be cross-platform and run on Windows, Linux, or macOS with minimal configuration.

- Maintainability: The modular design of separate Python scripts for each functionality (folder lock, vault, notes) ensures easy maintenance and debugging. Updates to one module should not break the functionality of others.

- Scalability: Though initially designed for single-user use, the framework should be scalable to support multi-user logins or role-based access in the future.

3. Overall Framework of Application

The overall framework of the application has been structured in a layered manner to ensure modularity, security, and ease of maintenance. Each layer has its own defined responsibilities, and all layers interact seamlessly to provide the required functionalities. The framework is designed to avoid the use of traditional databases and instead leverages file-based storage through Excel sheets and Word documents, making it lightweight and easy to handle.

- **Login Layer:**
  The very first layer of the system is the login mechanism, where users authenticate themselves using a passkey. This passkey ensures that only authorized individuals gain access to the application. The login process acts as the first level of defense, preventing any unauthorized access to the modules. User credentials and access metadata can be maintained in a secure file, with hashing applied to the passkey for validation.

- **Frontend Layer (Flask Web UI):**
  The application uses Flask as both frontend and backend, providing a web-based interface where users interact with the system. Instead of relying on separate HTML/CSS/JS frontends, Flask's built-in templating and forms are used to design simple and functional interfaces. Through this layer, users can perform three main tasks:

  Select a root folder to lock or unlock.

  Enter and manage their credentials in the Password Vault.

  Write, edit, and save secure notes.

- **Backend Layer (Python Modules):**
  The backend layer forms the core of the application's functionality. It consists of independent Python modules **[3]**, each responsible for one major feature. The folder locking module handles restricting access to specific folders by encryption or permission alteration. The password vault module encrypts and manages sensitive login details in Excel format. The notes module allows creation of encrypted Word documents. Each of these modules implements cryptographic functions using libraries such as cryptography or pyAesCrypt. In addition, the backend includes metadata handling functions to log all security operations, such as the algorithm used, the hash of the key, and the date-time of the operation.

- **Storage Layer (Excel and File System):**
  The storage layer ensures persistence of all encrypted data and associated logs. Instead of using SQL databases, the system stores credentials and notes in files. Password details are stored in an encrypted Excel file (Pass.xlsx), while notes are saved as encrypted Word files (.docx). Metadata for each module is stored in separate Excel sheets like OFile.xlsx for folder locks, OPass.xlsx for password vault changes, and ONotes.xlsx for secure notes. These metadata sheets hold information such as folder paths, encryption algorithms used, generated keys, hashes, and timestamps, making them essential for auditing and recovery.

4. Design of Individual Modules

4.1 User Login Module

To ensure that only authorized users can access the application. Design Details:

- The user is prompted to enter a master passkey when accessing the system.
- The entered passkey is not stored in plain text. Instead:
  It is hashed using a secure hashing algorithm (e.g., SHA-256 or bcrypt).
  The hashed version is compared with the stored hash value for validation.
- This ensures that even system developers or attackers cannot retrieve the original passkey.
- The login mechanism acts as a first line of defense, preventing unauthorized access to critical modules.

Key Features:

- Input validation to prevent brute-force or injection attacks.
- Secure passkey hashing and comparison.
- Error handling for incorrect or repeated login attempts.
- Optional logging of login attempts in metadata for auditing.

4.2 Folder Lock Module

To protect sensitive folders by locking/unlocking them through encryption. Design Details:

- The user selects a folder through the Flask web interface.
- A unique encryption key is generated for that session (using libraries like cryptography.Fernet).
- The folder contents are either:
  - Encrypted, OR
  - Hidden and access permissions modified using OS commands.
- Metadata about the operation is recorded in OFile.xlsx, including:
  - Folder path
  - Encryption algorithm used
  - Generated hash of the key
  - Timestamp of the operation

Key Features:

- One-click lock/unlock mechanism via the web app.
- Cross-platform folder permission handling.
- Secure encryption with key rotation.
- Audit trail maintained in metadata logs.

4.3 Folder Lock Module

To securely store and manage user credentials for websites or applications.

- Users can enter website name, username, and password through the Flask form.
- Credentials are stored in an Excel file (Pass.xlsx).
- The entire file is encrypted using AES (Advanced Encryption Standard) or Fernet encryption.

- A metadata file (OPass.xlsx) stores details such as:
    - Hash of the encryption key
    - Algorithm used
    - Timestamp of storage/update

- When required, the user can decrypt the vault (after successful login) to view credentials.

Key Features:

- Centralized secure storage for passwords.

- Encryption with AES/Fernet to prevent data leaks.

- Easy addition, update, and deletion of credentials.

- Metadata ensures full traceability of changes.

4.4 Notes Module

To allow users to create and store personal or confidential notes securely.

- The user writes text through a form in the Flask app.

- Notes are saved as .docx files for compatibility and formatting.

- Each .docx is encrypted using AES/Fernet before storage.

- Metadata of notes (file name, timestamp, encryption details) is stored in ONotes.xlsx.

Key Features:

- Encrypted Word files prevent unauthorized access.

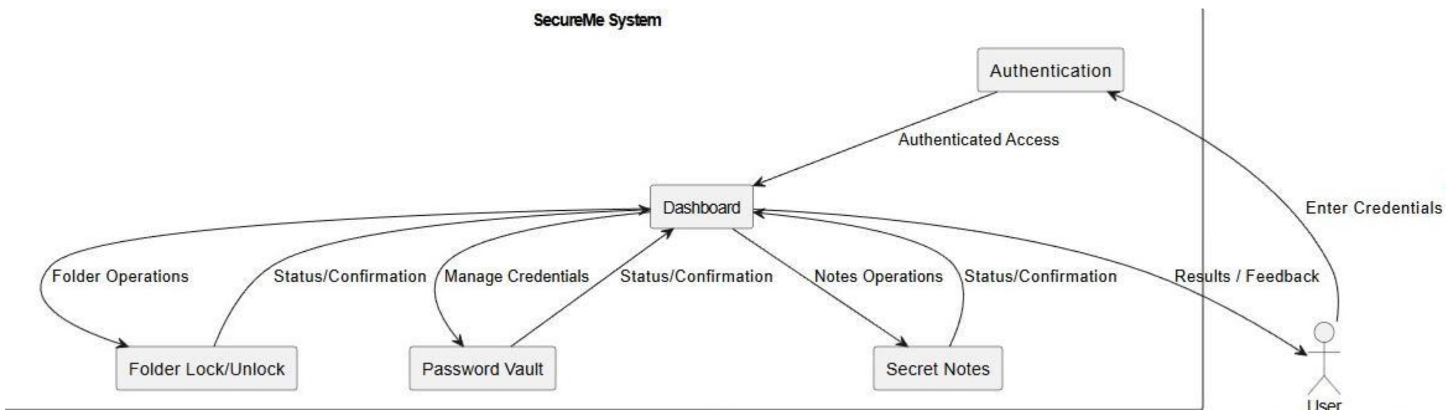
- Automatic logging of encryption/decryption details.

- Ensures user privacy and confidentiality.

5. Data Flow Diagrams (DFD)

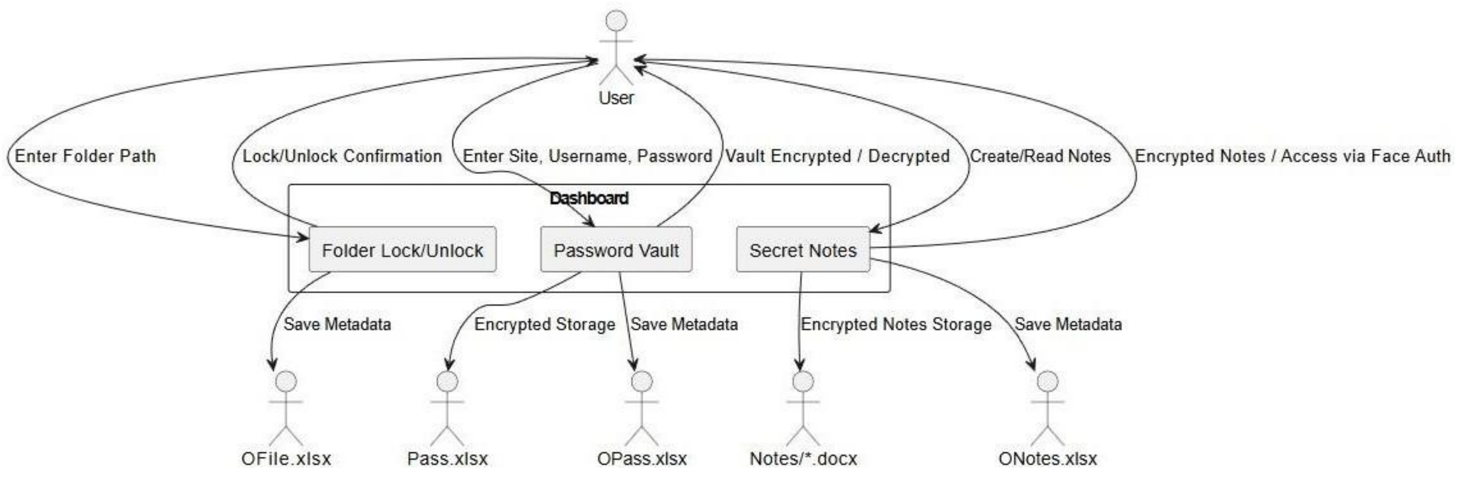
5.1 0-Level DFD

The Level 0 DFD represents the system as a single process, showing how the user interacts with the application. The user provides input (folder path, credentials, notes) and receives output (locked folder, encrypted vault, or secured notes).



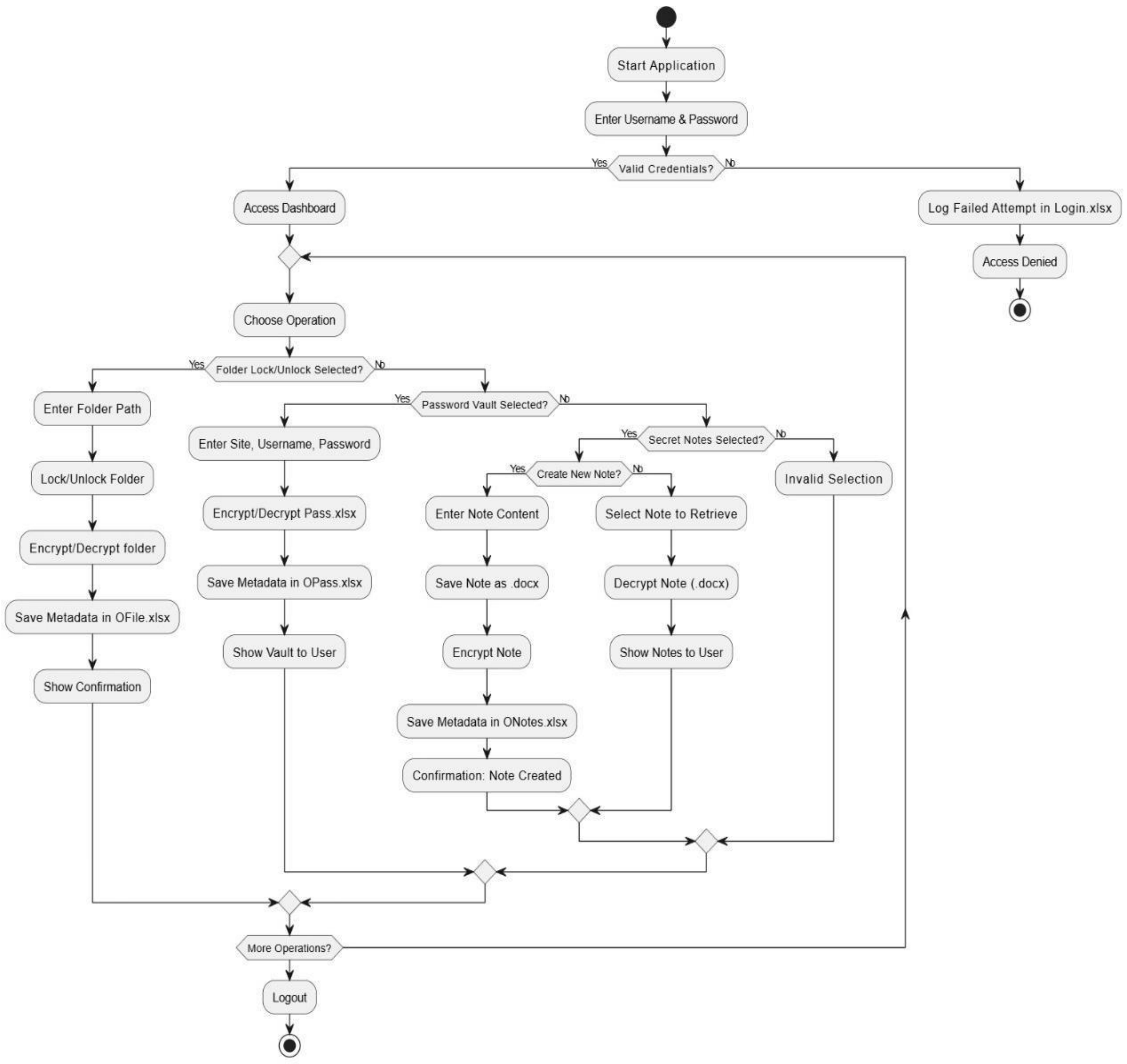
5.2 Level-1 DFD

The Level 1 DFD breaks down the main process into three sub-processes: Folder Locking, Password Vault Management, and Notes Management. Each sub-process interacts with the storage layer (Excel files and Word documents) to securely store and retrieve data, while the login process validates the user before accessing any module.



6. FlowChart

The flowchart illustrates the step-by-step workflow of the Secure Platform. It starts with the user login via the passkey, showing the decision process for authentication. Once the user is verified, the diagram depicts how the system directs the user to the dashboard, from where they can access the Folder Lock, Password Vault, and Notes modules. Each module's interaction with encryption operations, metadata handling, and secure storage is represented, providing a clear visual representation of the system's data flow, module interconnections, and decision points.



7. Source Code

i. Landing.html

```
{% block content %}

<link rel="icon" type="image/png" sizes="32x32"
  href="https://images.icon-icons.com/1402/PNG/512/care-for-security_96913.png">
<title>SecureMe - Offline Security</title>
<style>
  .main-card {
    margin-top: 90px;
    max-width: 900px;
    width: 100%;
    min-height: 450px;
    display: flex;
    flex-direction: column;
    border-radius: 20px;
    overflow: hidden;
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.2);
  }
```

```
  @media (min-width: 768px) {
    .main-card {
      flex-direction: row;
    }
  }
```

```
  .left-panel {
    background-color: #181818;
    color: #fff;
    padding: 2rem;
    flex: 1;
    display: flex;
    flex-direction: column;
    justify-content: center;
  }
```

```
  .left-panel h2 {
    font-size: 1.75rem;
    font-weight: bold;
    margin-bottom: 1.5rem;
  }
```

```
  .left-panel ul {
    list-style: none;
    padding-left: 0;
  }
```

```
  .left-panel li {
    margin-bottom: 0.75rem;
    font-size: 1.1rem;
  }
```

```
  /* Right panel: action area */
  .right-panel {
    background-color: #fff;
    color: #181818;
    padding: 2rem;
    flex: 1;
    display: flex;
    flex-direction: column;
    justify-content: center;
  }
  .right-panel h2 {
    font-size: 1.75rem;
    font-weight: bold;
    margin-bottom: 0.75rem;
  }
```

```
  .right-panel p {
    color: #6b7280;
    /* gray text */
    margin-bottom: 1.5rem;
  }
```

```
  .btn {
    display: inline-block;
    padding: 0.75rem 1.5rem;
    border-radius: 12px;
    text-decoration: none;
    font-weight: bold;
    text-align: center;
    transition: background 0.3s, color 0.3s;
  }
```

```
  .btn-dark {
    background-color: #181818;
    color: #fff;
    border: none;
  }
```

```
  .btn-dark:hover {
    background-color: #2c2c2c;
  }
```

```
  .btn-outline {
    background-color: #fff;
    color: #181818;
    border: 2px solid #181818;
  }
```

```
  .btn-outline:hover {
    background-color: #181818;
    color: #fff;
  }
```

```
  .main-wrapper {
    display: flex;
    justify-content: center;
    align-items: start;
    padding: 0.5rem;
  }
</style>
```

```
<div class="main-wrapper">
  <div class="main-card">
```

```
    <!-- Left Panel -->
    <div class="left-panel">
      <h2>Why SecureMe?</h2>
      <ul>
        <li>  Lock and protect folders</li>
        <li>  Store credentials securely</li>
        <li>  Take private encrypted notes</li>
        <li>  Works 100% offline</li>
        <li>⚡  Simple & lightweight</li>
      </ul>
    </div>
    <div class="right-panel">
      <h2>SecureMe Access</h2>
      <p>Manage your data securely — everything runs locally.</p>
```

```
      {% if session.get('authenticated') %}
      <a href="{{ url_for('dashboard') }}" class="btn btn-dark" style="margin-bottom:0.75rem;">Open Dashboard</a>
      <a href="{{ url_for('Logout') }}" class="btn btn-outline">Logout</a>
      {% else %}
      {% if master_set %}
      <a href="{{ url_for('Login') }}" class="btn btn-dark">Login</a>
      {% else %}
      <a href="{{ url_for('CreatePasskey') }}" class="btn btn-dark">Create Master Passkey</a>
      {% endif %}
      {% endif %}
```

```
    </div>
```

```
  </div>
</div>
```

```
{% endblock %}
```

ii. CreatePasskey.html

```
{% extends "Nav.html" %}
{% block title %}Setup Master Passkey{% endblock %}
{% block content %}

<style>
  .center-container {
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 3rem 1rem;
  }
```

```
  .passkey-card {
    margin-top: 85px;
    background: #fff;
    border-radius: 16px;
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);
    padding: 2.5rem 2rem;
    max-width: 450px;
    width: 100%;
  }
```

```
  .passkey-card h2 {
    text-align: center;
    font-size: 1.75rem;
    font-weight: 700;
    color: #181818;
    margin-bottom: 0.75rem;
  }
```

```
  .passkey-card p {
    text-align: center;
    color: #555;
    margin-bottom: 2rem;
    font-size: 0.95rem;
    line-height: 1.4;
  }
  .passkey-card input[type="password"] {
    width: 100%;
    padding: 0.85rem 1rem;
    border-radius: 10px;
    border: none;
    background-color: #181818;
    color: #fff;
    font-size: 1rem;
    margin-bottom: 1.5rem;
    box-shadow: 0 3px 6px rgba(0, 0, 0, 0.1);
  }
```

```
  .passkey-card input[type="password"]::placeholder {
    color: #ccc;
  }
```

```
  .btn-primary {
    background-color: #181818;
    color: #fff;
    border: none;
    padding: 0.85rem 1.5rem;
    border-radius: 10px;
    cursor: pointer;
    font-weight: 600;
    transition: all 0.3s ease;
  }
```

```
  .btn-primary:hover {
    background-color: #333;
  }
```

```
  .btn-cancel {
    color: #181818;
    font-weight: 600;
    text-decoration: none;
    transition: all 0.3s ease;
  }
```

```
  .btn-cancel:hover {
    text-decoration: underline;
  }
```

```
  .button-row {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-top: 1rem;
  }
```

```
  @media (max-width: 500px) {
    .passkey-card {
      padding: 2rem 1.5rem;
    }
```

```
    .button-row {
      flex-direction: column;
      gap: 1rem;
    }
  }
</style>
```

```
<div class="center-container">
  <div class="passkey-card">
    <h2>Create Master Passkey</h2>
    <p>Create a strong master passkey — this will control access to <strong>SecureMe</strong>.</p>
    <form method="post">
      <input type="password" name="pass1" required minlength="8" placeholder="Master passkey (min 8 chars)" />
      <input type="password" name="pass2" required minlength="8" placeholder="Confirm passkey" />
```

```
      <div class="button-row">
        <button type="submit" class="btn-primary">Create</button>
        <a href="{{ url_for('Landing') }}" class="btn-cancel">Cancel</a>
      </div>
    </form>
  </div>
</div>
```

```
{% endblock %}
```

iii. Login.html

```
{% extends "Nav.html" %}
{% block title %}Login{% endblock %}
{% block content %}

<style>
  .login-container {
    max-width: 450px;
    margin: 12rem auto;
    padding: 2rem;
    background-color: #fff;
    border-radius: 1rem;
    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
  }
```

```
  .login-container h2 {
    text-align: center;
    color: #181818;
    margin-bottom: 1rem;
    font-weight: bold;
  }
```

```
  .login-container label {
    display: block;
    font-weight: 600;
    color: #181818;
    margin-bottom: 0.5rem;
  }
```

```
  .login-container input {
    width: 100%;
    padding: 0.75rem 1rem;
    border-radius: 0.5rem;
    border: none;
    margin-bottom: 1.5rem;
    background-color: #181818;
    color: #fff;
    font-size: 1rem;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
  }
```

```
  .login-container button {
    width: 100%;
    padding: 0.75rem;
    border-radius: 0.5rem;
    border: none;
    background-color: #181818;
    color: #fff;
    font-weight: bold;
    font-size: 1rem;
    cursor: pointer;
    transition: background 0.3s ease;
  }
```

```
  .login-container button:hover {
    background-color: #2c2c2c;
  }
```

```
  .login-container .link-btn {
    display: block;
    margin-top: 1rem;
    text-align: center;
    font-weight: 600;
    color: #181818;
    text-decoration: none;
    transition: color 0.3s ease;
  }
```

```
  .login-container .link-btn:hover {
    color: #0369a1;
  }
</style>
```

```
<div class="login-container">
  <h2>Login</h2>
  <form method="post">
    <label for="pass">Master passkey</label>
    <input id="pass" type="password" name="pass" required placeholder="Enter your master passkey">
```

```
    <button type="submit">Login</button>
```

```
    {% if not master_set %}
    <a class="link-btn" href="{{ url_for('CreatePasskey') }}">Create Master Passkey</a>
    {% endif %}
  </form>
</div>
```

```
{% endblock %}
```

iv. Nav.html

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <link rel="icon" type="image/png" sizes="32x32"
    href="https://images.icon-icons.com/1402/PNG/512/care-for-security_96913.png">
  <title>SecureMe - {% block title %}{% endblock %}</title>
```

```
  <style>
    * {
      box-sizing: border-box;
      margin: 0;
      padding: 0;
      font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
    }
```

```
    ::-webkit-scrollbar {
      display: none;
    }
```

```css
body {
  background-color: #f9f9f9;
  color: #181818;
  line-height: 1.6;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

```css
.navbar {
  width: 100%;
  background-color: #fff;
  border-bottom: 1px solid #e5e5e5;
  padding: 0.75rem 2rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
  position: sticky;
  top: 0;
  z-index: 10;
}
```

```css
.navbar .logo {
  font-size: 0.8rem;
  font-weight: 700;
  color: #181818;
  display: flex;
  align-items: center;
  gap: 0.5rem;
}
```

```css
.logo img {
  width: 10%;
}
```

```css
.nav-links {
  display: flex;
  gap: 1rem;
  align-items: center;
}
```

```css
.nav-links a {
  text-decoration: none;
  padding: 0.6rem 1.2rem;
  border-radius: 8px;
  font-weight: 600;
  transition: all 0.3s ease;
}
```

```css
.nav-links .btn-home {
  border: 2px solid #181818;
  color: #181818;
  background-color: #fff;
}
```

```css
.nav-links .btn-home:hover {
  background-color: #181818;
  color: #fff;
}
```

```css
.nav-links .btn-logout {
  background-color: #181818;
  color: #fff;
  border: none;
}
```

```
    .nav-links .btn-logout:hover {
      background-color: #333;
    }
```

```
    .container {
      width: 95%;
      max-width: 1100px;
      margin: 2rem auto;
      flex: 1;
    }
```

```
    .flash-message {
      padding: 1rem;
      border-radius: 8px;
      margin-bottom: 1rem;
      background-color: #e0f2fe;
      color: #0369a1;
      border: 1px solid #bae6fd;
      text-align: center;
    }
```

```
    footer {
      text-align: center;
      padding: 1rem 0;
      background-color: #fff;
      border-top: 1px solid #e5e5e5;
      font-size: 0.9rem;
      color: #6b7280;
      width: 100%;
    }
  </style>
</head>
```

```
<body>
```

```
  {% if session.get('authenticated') %}
  <nav class="navbar">
    <div class="logo">
      <img src="https://images.icon-icons.com/1402/PNG/512/care-for-security_96913.png">
      <h1>SecureMe</h1>
    </div>
    <div class="nav-links">
      <a href="{{ url_for('Landing') }}" class="btn-home">Home</a>
      <a href="{{ url_for('Logout') }}" class="btn-logout">Logout</a>
    </div>
  </nav>
  {% endif %}
```

```
  <div class="container">
    {% with messages = get_flashed_messages() %}
    {% if messages %}
    <div>
      {% for m in messages %}
      <div class="flash-message">{{ m }}</div>
      {% endfor %}
    </div>
    {% endif %}
    {% endwith %}
```

```
    {% block content %}{% endblock %}
  </div>
```

```
  <footer>
    <small>SecureMe — local-only demo</small>
  </footer>
```

```
</body>
```

```
</html>
```

v. Dashboard.html

```
{% extends "Nav.html" %}
{% block title %}Dashboard{% endblock %}
{% block content %}

<style>
  .dashboard-container {
    padding: 3rem 2rem;
    background-color: #fff;
    font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
    color: #181818;
    max-width: 1200px;
    margin: 80px auto;
  }
```

```
  .dashboard-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: wrap;
    margin-bottom: 2rem;
  }
```

```
  .dashboard-header h2 {
    font-size: 2.2rem;
    font-weight: 700;
    margin: 0;
    color: #111;
  }
```

```
  .stats-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(260px, 1fr));
    gap: 2rem;
  }
```

```
  .stat-card {
    background: #fafafa;
    border: 1px solid #e5e5e5;
    border-radius: 16px;
    padding: 1.75rem 1.5rem;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
    position: relative;
    transition: all 0.3s ease;
  }
```

```
  .stat-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.08);
  }
```

```
  .stat-card h6 {
    color: #6b7280;
    font-weight: 600;
    margin-bottom: 0.5rem;
    font-size: 0.9rem;
  }
```

```
  .stat-card h3 {
```

```css
    font-size: 2.5rem;
    font-weight: 700;
    color: #111;
    margin-bottom: 0.25rem;
  }
```

```css
  .stat-card p {
    font-size: 0.9rem;
    color: #888;
    margin: 0;
  }
```

```css
  .card-arrow {
    position: absolute;
    bottom: 1rem;
    right: 1rem;
    font-size: 1.5rem;
    color: #0369a1;
    text-decoration: none;
    transition: transform 0.2s ease;
  }
```

```css
  .card-arrow:hover {
    transform: translateX(4px);
  }
```

```css
  .table-container {
    margin-top: 3rem;
  }
```

```css
  .login-table {
    width: 100%;
    border-collapse: collapse;
    border-radius: 10px;
    overflow: hidden;
    background: #fff;
    border: 1px solid #e5e5e5;
    box-shadow: 0 5px 20px rgba(0, 0, 0, 0.05);
  }
```

```css
  .login-table th,
  .login-table td {
    padding: 1rem;
    text-align: left;
    font-size: 0.95rem;
  }
```

```css
  .login-table th {
    background-color: #181818;
    color: #fff;
  }
```

```css
  .login-table tr:nth-child(even) {
    background-color: #f4f4f4;
  }
```

```css
  .login-table tr:hover {
    background-color: #e0f2fe;
  }
</style>
```

```html
<div class="dashboard-container">
```

```html
  <div class="dashboard-header">
    <h2>Dashboard Overview</h2>
  </div>
  <!-- === Stats Grid === -->
  <div class="stats-grid">
    <div class="stat-card">
      <h6>Locked Folders</h6>
      <h3>{{ stats.locked_folders }}</h3>
      <p>Folders secured via ACL</p>
      <a href="{{ url_for('FolderLocker') }}" class="card-arrow">&#8594;</a>
    </div>
```

```
    <div class="stat-card">
      <h6>Vault Entries</h6>
      <h3>{{ stats.vault_entries }}</h3>
      <p>Encrypted credentials</p>
      <a href="{{ url_for('PasswordVault') }}" class="card-arrow">&#8594;</a>
    </div>
```

```
    <div class="stat-card">
      <h6>Notes</h6>
      <h3>{{ stats.notes }}</h3>
      <p>Encrypted personal notes</p>
      <a href="{{ url_for('notes_home') }}" class="card-arrow">&#8594;</a>
    </div>
  </div>
</div>
```

```
{% endblock %}
```

vi. FolderLocker.html

```
{% extends "Nav.html" %}
{% block title %}Folder Locker{% endblock %}
{% block content %}
<style>
    body {
        background-color: #ffffff;
        color: #000000;
        font-family: "Poppins", "Segoe UI", sans-serif;
    }

    .locker-container {
        max-width: 900px;
        margin: 4rem auto;
        background: #f8f8f8;
        border-radius: 10px;
        padding: 3rem 3.5rem;
        box-shadow: 0 0 25px rgba(0, 0, 0, 0.08);
        border: 1px solid #e0e0e0;
    }
```

```
    h2 {
        text-align: center;
        color: #000;
        font-weight: 600;
        font-size: 1.8rem;
        margin-bottom: 2rem;
    }
```

```
    input[type="text"] {
        width: 100%;
        padding: 14px 16px;
        border-radius: 8px;
        border: 1px solid #ccc;
        background: #fff;
        color: #000;
        font-size: 15px;
        margin-bottom: 1.5rem;
        transition: all 0.2s ease-in-out;
        box-sizing: border-box;
    }
```

```
    input[type="text"]:focus {
        border-color: #000;
        outline: none;
    }
```

```
    .btn {
        display: inline-block;
        width: 48%;
        text-align: center;
        padding: 12px 0;
        border: none;
        border-radius: 8px;
        cursor: pointer;
        font-weight: 600;
        font-size: 15px;
        transition: all 0.3s ease;
    }
```

```
    .lock-btn {
        background-color: #000;
        color: #fff;
    }
```

```
    .unlock-btn {
        background-color: transparent;
        border: 2px solid #000;
        color: #000;
    }
```

```
    .lock-btn:hover {
        background-color: #333;
    }
```

```
    .unlock-btn:hover {
        background-color: #000;
        color: #fff;
    }
```

```
    .btn-container {
        display: flex;
        justify-content: space-between;
        gap: 10px;
    }
```

```
    p.message {
        text-align: center;
        color: #000;
        background: #eaeaea;
        border-radius: 6px;
        padding: 10px 15px;
        font-size: 14px;
        margin-top: 1.2rem;
        border: 1px solid #ccc;
    }
```

```
    table {
        width: 100%;
        border-collapse: collapse;
        margin-top: 2.5rem;
        font-size: 14px;
        border-radius: 6px;
        overflow: hidden;
    }

    th,
    td {
        padding: 12px 15px;
        border-bottom: 1px solid #ddd;
        text-align: left;
    }

    th {
        background: #f0f0f0;
        color: #000;
        text-transform: uppercase;
        font-size: 13px;
        font-weight: 600;
    }

    tr:hover td {
        background-color: #fafafa;
    }

    .status {
        font-weight: 600;
        text-transform: capitalize;
    }

    .status.locked {
        color: #d60000;
    }

    .status.unlocked {
        color: #2b8a3e;
    }
</style>

<div class="locker-container">
    <h2>  Folder Locker</h2>

    <form method="POST">
        <input type="text" name="folder_path" placeholder="Enter full folder path..." required />
        <div class="btn-container">
            <button type="submit" name="action" value="lock" class="btn lock-btn">
                  Lock Folder
            </button>
            <button type="submit" name="action" value="unlock" class="btn unlock-btn">
                  Unlock Folder
            </button>
        </div>
    </form>

    {% if message %}
    <p class="message">{{ message }}</p>
    {% endif %}

    <h3 style="margin-top:2.5rem; font-weight:600; font-size:1.2rem;">  Locked Folder Log</h3>
    <table>
        <thead>
            <tr>
                <th>Folder Path</th>
                <th>Status</th>
                <th>Date-Time</th>
            </tr>
        </thead>
        <tbody>
            {% for f in folders %}
            <tr>
                <td>{{ f.path }}</td>
                <td class="status {% if f.status == 'Locked' %}locked{% else %}unlocked{% endif %}">
                    {{ f.status }}
                </td>
                <td>{{ f.datetime }}</td>
            </tr>
            {% else %}
            <tr>
                <td colspan="3" style="color:#777; text-align:center;">No entries found.</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</div>
```

```
{% endblock %}
```

vii. PasswordVault.html

```
{% extends "Nav.html" %}
{% block title %}Password Vault{% endblock %}
{% block content %}

<style>
    body {
        background: #fff;
        color: #111;
        font-family: "Poppins", sans-serif;
    }
```

```
    .vault-container {
        max-width: 1100px;
        margin: 40px auto;
        padding: 20px;
    }
```

```
    .vault-header {
        display: flex;
        justify-content: space-between;
        align-items: center;
        margin-bottom: 25px;
    }
```

```
    .vault-header h2 {
        font-weight: 600;
        color: #111;
    }
```

```
    .filter-bar {
        display: flex;
        gap: 10px;
        margin-bottom: 25px;
        align-items: center;
    }
```

```
    .filter-bar select {
        padding: 8px 12px;
        border-radius: 6px;
        border: 1px solid #ccc;
        font-size: 14px;
    }
```

```css
    .add-btn {
        background: #111;
        color: #fff;
        border: none;
        padding: 8px 18px;
        border-radius: 6px;
        cursor: pointer;
        font-size: 14px;
        transition: 0.2s;
    }
```

```css
    .add-btn:hover {
        background: #333;
    }
```

```css
    .vault-grid {
        display: grid;
        grid-template-columns: repeat(auto-fill, minmax(280px, 1fr));
        gap: 20px;
    }
```

```css
    .vault-card {
        background: #f8f9fa;
        border: 1px solid #ddd;
        border-radius: 10px;
        padding: 18px;
        transition: 0.3s;
        cursor: pointer;
        position: relative;
    }
```

```css
    .vault-card:hover {
        box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    }
```

```css
    .vault-card h4 {
        font-size: 18px;
        font-weight: 600;
        margin-bottom: 8px;
        color: #111;
    }
```

```css
    .vault-card p {
        margin: 5px 0;
        font-size: 14px;
        color: #333;
    }
```

```css
    .category-tag {
        display: inline-block;
        padding: 3px 8px;
        background: #e0e0e0;
        border-radius: 6px;
        font-size: 12px;
        color: #444;
        margin-bottom: 10px;
    }
```

```css
    .eye-icon {
        cursor: pointer;
        margin-left: 6px;
    }
```

```css
    /* Edit mode styling */
    .vault-card form {
        display: flex;
        flex-direction: column;
        gap: 8px;
    }

    .vault-card input,
    .vault-card select {
        width: 100%;
        padding: 8px 10px;
        border: 1px solid #ccc;
        border-radius: 5px;
        font-size: 14px;
    }

    .save-btn {
        background: #111;
        color: #fff;
        border: none;
        padding: 8px 14px;
        border-radius: 6px;
        cursor: pointer;
        font-size: 14px;
        align-self: flex-end;
        transition: 0.2s;
    }

    .save-btn:hover {
        background: #333;
    }

    .no-data {
        text-align: center;
        color: #666;
        margin-top: 30px;
    }

    /* Modal styles */
    .modal {
        display: none;
        position: fixed;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background: rgba(0, 0, 0, 0.5);
        justify-content: center;
        align-items: center;
        z-index: 1000;
    }

    .modal-content {
        background: #fff;
        padding: 25px;
        border-radius: 10px;
        width: 350px;
        position: relative;
        animation: fadeIn 0.3s ease-in-out;
    }

    @keyframes fadeIn {
        from {
            opacity: 0;
            transform: scale(0.95);
        }

        to {
            opacity: 1;
            transform: scale(1);
        }
    }
```

```
    .modal-content h3 {
        margin-bottom: 15px;
        font-size: 20px;
    }
```

```
    .modal-content input,
    .modal-content select {
        width: 100%;
        padding: 8px 10px;
        border: 1px solid #ccc;
        border-radius: 6px;
        margin-bottom: 10px;
        font-size: 14px;
    }
```

```
    .modal-actions {
        display: flex;
        justify-content: space-between;
    }
```

```
    .cancel-btn {
        background: #999;
        color: #fff;
        border: none;
        padding: 8px 14px;
        border-radius: 6px;
        cursor: pointer;
    }
</style>
```

```
<div class="vault-container">
```

```
    <div class="vault-header">
        <h2>  My Password Vault</h2>
        <button class="add-btn" id="addBtn">+ Add Password</button>
    </div>
```

```
    <div class="filter-bar">
        <label for="filter">Filter by Category:</label>
        <select id="filter">
            <option value="All">All</option>
            <option value="Social">Social</option>
            <option value="Work">Work</option>
            <option value="Finance">Finance</option>
            <option value="Entertainment">Entertainment</option>
            <option value="Other">Other</option>
        </select>
    </div>
```

```
    {% if vault_entries and vault_entries|length > 0 %}
    <div class="vault-grid" id="vaultGrid">
        {% for entry in vault_entries %}
        <div class="vault-card" data-category="{{ entry.category }}" data-id="{{ loop.index }}">
            <span class="category-tag">{{ entry.category }}</span>
            <h4>{{ entry.name }}</h4>
            <p><strong>Website:</strong> <a href="{{ entry.website }}" target="_blank">{{ entry.website }}</a></p>
            <p><strong>Email/User:</strong> {{ entry.contact }}</p>
            <p><strong>Password:</strong>
                <span class="password-value" data-password="{{ entry.password }}">••••••••</span>
                <span class="eye-icon" onclick="togglePassword(this)"> </span>
            </p>
            <p><small>Added: {{ entry.date }}</small></p>
        </div>
        {% endfor %}
    </div>
    {% else %}
    <p class="no-data">No passwords saved yet. Click “Add Password” to create one.</p>
    {% endif %}
</div>
```

```
<div id="addModal" class="modal">
    <div class="modal-content">
        <h3>Add New Password</h3>
        <form method="POST" action="/vault/add">
            <input type="text" name="website" placeholder="Website URL" required>
            <input type="text" name="name" placeholder="Name" required>
            <input type="text" name="contact" placeholder="Email / Username / Phone">
            <input type="password" name="password" placeholder="Password" required>
            <select name="category" required>
                <option value="Social">Social</option>
                <option value="Work">Work</option>
                <option value="Finance">Finance</option>
                <option value="Entertainment">Entertainment</option>
                <option value="Other" selected>Other</option>
            </select>
            <div class="modal-actions">
                <button type="submit" class="save-btn">  Save</button>
                <button type="button" id="closeModal" class="cancel-btn">✖ Cancel</button>
            </div>
        </form>
    </div>
</div>
```

```
<script>
    const filter = document.getElementById("filter");
    const vaultGrid = document.getElementById("vaultGrid");
    let activeEditCard = null;
```

```
    filter.addEventListener("change", () => {
        const category = filter.value;
        document.querySelectorAll(".vault-card").forEach(card => {
            card.style.display = (category === "All" || card.dataset.category === category) ? "block" : "none";
        });
    });
```

```
    function togglePassword(el) {
        const pwSpan = el.previousElementSibling;
        const realPw = pwSpan.getAttribute("data-password");
        if (pwSpan.textContent === "••••••••") {
            pwSpan.textContent = realPw;
            el.textContent = " ";
        } else {
            pwSpan.textContent = "••••••••";
            el.textContent = " ";
        }
    }
```

```
    function restoreCard(card, originalHTML) {
        card.innerHTML = originalHTML;
        activeEditCard = null;
    }
```

```
    document.querySelectorAll(".vault-card").forEach(card => {
        card.addEventListener("click", function (e) {
            if (e.target.classList.contains("eye-icon")) return;
            if (activeEditCard && activeEditCard !== this) return;
```

```
            activeEditCard = this;
            const originalHTML = this.innerHTML;
            const id = this.dataset.id;
            const name = this.querySelector("h4").textContent;
            const website = this.querySelector("a").textContent;
            const contact = this.querySelector("p:nth-of-type(2)").textContent.replace("Email/User:", "").trim();
            const category = this.dataset.category;
            const password = this.querySelector(".password-value").getAttribute("data-password");
```

```
            this.innerHTML = `
                <form method="POST" action="/password-vault/edit/${id}">
                    <input type="text" name="name" value="${name}" placeholder="Name" required>
                    <input type="text" name="website" value="${website}" placeholder="Website URL" required>
                    <input type="text" name="contact" value="${contact}" placeholder="Email / Username">
                    <input type="password" name="password" value="${password}" placeholder="Password" required>
                    <select name="category" required>
                        <option ${category === "Social" ? "selected" : ""}>Social</option>
                        <option ${category === "Work" ? "selected" : ""}>Work</option>
                        <option ${category === "Finance" ? "selected" : ""}>Finance</option>
                        <option ${category === "Entertainment" ? "selected" : ""}>Entertainment</option>
                        <option ${category === "Other" ? "selected" : ""}>Other</option>
                    </select>
                    <button type="submit" class="save-btn">  Save</button>
                </form>
            `;
```

```
            function handleOutsideClick(event) {
                if (!card.contains(event.target)) {
                    restoreCard(card, originalHTML);
                    document.removeEventListener("click", handleOutsideClick);
                }
            }
            setTimeout(() => document.addEventListener("click", handleOutsideClick), 50);
        });
    });
```

```
    const addBtn = document.getElementById("addBtn");
    const addModal = document.getElementById("addModal");
    const closeModal = document.getElementById("closeModal");
```

```
    addBtn.addEventListener("click", () => addModal.style.display = "flex");
    closeModal.addEventListener("click", () => addModal.style.display = "none");
    window.addEventListener("click", (e) => { if (e.target === addModal) addModal.style.display = "none"; });
</script>
```

```
{% endblock %}
```

viii. NotesManager.html

```
{% extends "Nav.html" %}
{% block title %}Notes{% endblock %}
{% block content %}

<style>
    body {
        margin: 0;
        background-color: #fafafa;
        color: #1e1e1e;
        font-family: "Inter", "Segoe UI", sans-serif;
    }
```

```
    .notes-container {
        position: relative;
        display: flex;
        height: 90vh;
        border: 1px solid #ddd;
        border-radius: 8px;
```

```css
        overflow: hidden;
        box-shadow: 0 4px 20px rgba(0, 0, 0, 0.05);
        margin: 2rem;
    }
```

```css
    /* Sidebar */
    .sidebar {
        width: 280px;
        background-color: #fff;
        border-right: 1px solid #e0e0e0;
        padding: 1rem;
        display: flex;
        flex-direction: column;
        overflow-y: auto;
        z-index: 1;
    }
```

```css
    .sidebar h3 {
        font-size: 1rem;
        color: #333;
        font-weight: 600;
        margin-bottom: 1rem;
        padding-left: 4px;
    }
```

```css
    .create-btn {
        display: block;
        background-color: #1e1e1e;
        color: white;
        text-align: center;
        font-weight: 600;
        padding: 10px;
        border-radius: 6px;
        text-decoration: none;
        margin-bottom: 1rem;
        transition: all 0.2s ease;
    }
```

```css
    .create-btn:hover {
        background-color: #444;
    }
```

```css
    .note-item {
        padding: 10px 12px;
        border-radius: 6px;
        cursor: pointer;
        color: #333;
        text-decoration: none;
        display: block;
        font-size: 0.95rem;
        transition: all 0.2s ease;
    }
```

```css
    .note-item:hover {
        background-color: #f5f5f5;
    }
```

```css
    .note-item.active {
        background-color: #1e1e1e;
        color: #fff;
    }
```

```css
    /* Editor Section */
    .editor-section {
        flex: 1;
        display: flex;
        flex-direction: column;
        background-color: #fff;
        padding: 2rem;
        overflow-y: auto;
        position: relative;
    }
```

```css
.editor-section form {
    display: flex;
    flex-direction: column;
    height: 100%;
}
```

```css
.title-input,
.text-area {
    border: 1px solid #ccc;
    border-radius: 6px;
    padding: 12px;
    font-size: 1rem;
    transition: border-color 0.2s;
}
```

```css
.title-input {
    margin-bottom: 1rem;
}
```

```css
.title-input:focus,
.text-area:focus {
    outline: none;
    border-color: #1e1e1e;
}
```

```css
.text-area {
    flex: 1;
    resize: none;
}
```

```css
.button-row {
    margin-top: 1rem;
    display: flex;
    gap: 10px;
}
```

```css
.btn {
    background-color: #1e1e1e;
    border: none;
    padding: 10px 18px;
    color: #fff;
    font-weight: 600;
    border-radius: 6px;
    cursor: pointer;
    transition: all 0.2s ease;
}
```

```css
.btn:hover {
    background-color: #333;
}
```

```css
.btn-secondary {
    background-color: #e0e0e0;
    color: #333;
}
```

```css
.btn-secondary:hover {
    background-color: #d5d5d5;
}
```

```
    .empty-message {
        text-align: center;
        color: #999;
        margin-top: 20%;
    }
```

```
    .key-panel {
        position: absolute;
        top: 0;
        right: 0;
        width: 765px;
        height: 100%;
        padding-top: 240px;
        background-color: #1e1e1e;
        color: #fff;
        box-shadow: -4px 0 12px rgba(0, 0, 0, 0.2);
        transform: translateX(100%);
        transition: transform 0.3s ease;
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
        text-align: center;
        z-index: 3;
    }
```

```
    .key-panel.visible {
        transform: translateX(0);
    }
```

```
    .key-input {
        background-color: #fafafa;
        border: 1px solid #ccc;
        color: #333;
        padding: 10px;
        border-radius: 6px;
        font-size: 1rem;
        width: 250px;
        text-align: center;
    }
```

```
    .submit-btn {
        margin-top: 1rem;
        background-color: #fff;
        border: none;
        color: #1e1e1e;
        padding: 8px 20px;
        border-radius: 6px;
        cursor: pointer;
        font-weight: 600;
    }
```

```
    .submit-btn:hover {
        background-color: #f0f0f0;
    }
</style>
```

```
<div class="notes-container">
    <div class="sidebar">
        <a href="{{ url_for('create_new_note') }}" class="create-btn">+ New Note</a>
        <h3>My Notes</h3>
        {% for note in notes %}
        <a class="note-item {% if note == active_note %}active{% endif %}"
            href="{{ url_for('open_note', filename=note) }}">{{ note }}</a>
        {% else %}
        <p style="color:#999; font-size:0.9rem;">No notes yet</p>
        {% endfor %}
    </div>

    <div class="editor-section">
        {% if new %}
        <form method="POST" action="{{ url_for('create_new_note') }}">
            <input type="text" name="title" placeholder="Enter new file name..." class="title-input" required />
            <textarea name="content" placeholder="Write your content..." class="text-area" required></textarea>
            <div class="button-row">
                <button type="submit" class="btn">  Save Note</button>
            </div>
        </form>

        {% elif active_note %}
        <form method="POST" action="{{ url_for('edit_note', filename=active_note) }}">
            <input type="hidden" name="current_key" value="{{ current_key }}" />
            <input type="text" name="title" value="{{ title }}" class="title-input" id="titleInput" readonly />
            <textarea name="content" class="text-area" id="contentArea" readonly>{{ content }}</textarea>
            <div class="button-row">
                <button type="button" id="editBtn" class="btn">  Edit</button>
                <button type="submit" id="saveBtn" class="btn" style="display:none;">  Save Note</button>
            </div>
        </form>

        <script>
            const editBtn = document.getElementById("editBtn");
            const saveBtn = document.getElementById("saveBtn");
            const titleInput = document.getElementById("titleInput");
            const contentArea = document.getElementById("contentArea");

            editBtn.addEventListener("click", () => {
                titleInput.removeAttribute("readonly");
                contentArea.removeAttribute("readonly");
                contentArea.focus();
                editBtn.style.display = "none";
                saveBtn.style.display = "inline-block";
            });
        </script>

        {% else %}
        <div class="empty-message">
            <h3>Select a note from the left to view or create a new one</h3>
        </div>
        {% endif %}

        {% if active_note %}
        <div class="key-panel {% if show_key_prompt %}visible{% endif %}">
            <h2>  Enter Encryption Key</h2>
            <form method="POST" action="{{ url_for('open_note', filename=active_note) }}">
                <input type="password" name="key" maxlength="6" class="key-input" placeholder="Enter 6-character key"
                    required />
                <button type="submit" class="submit-btn">Unlock</button>
            </form>
        </div>
        {% endif %}
    </div>
</div>

{% endblock %}
```

ix. encryption_utils.py

```
import os
import hashlib
```

```python
import random
import string
import base64
from datetime import datetime
from cryptography.fernet import Fernet
from openpyxl import Workbook, load_workbook
from pathlib import Path
from config import BASE_DIR

# -----------------------------------------------------------------
# Paths
# -----------------------------------------------------------------
META_FILE = BASE_DIR / "data" / "ONotes.xlsx"
VAULT_KEY_FILE = BASE_DIR / "data" / "vault_master.key"
META_FILE.parent.mkdir(parents=True, exist_ok=True)
```

```python
# -----------------------------------------------------------------
# Utilities for file-based encryption (used by Notes)
# -----------------------------------------------------------------
def generate_random_key():
    return ''.join(random.choices(string.ascii_letters + string.digits, k=6))
```

```python
def derive_fernet(user_key: str) -> Fernet:
    digest = hashlib.sha256(user_key.encode("utf-8")).digest()
    b64key = base64.urlsafe_b64encode(digest[:32])
    return Fernet(b64key)
```

```python
def ensure_meta_exists():
    if not META_FILE.exists():
        wb = Workbook()
        ws = wb.active
        ws.title = "NotesKeys"
        ws.append(["File Name", "Algorithm", "Hash of Key", "Date-Time", "Key"])
        wb.save(META_FILE)
```

```python
def upsert_metadata(file_name: str, algo: str, key: str):
    ensure_meta_exists()
    wb = load_workbook(META_FILE)
    ws = wb.active
    key_hash = hashlib.sha256(key.encode("utf-8")).hexdigest()
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

```python
    found_row = None
    for r in range(2, ws.max_row + 1):
        if ws.cell(row=r, column=1).value == file_name:
            found_row = r
            break
```

```python
    if found_row:
        ws.cell(row=found_row, column=2).value = algo
        ws.cell(row=found_row, column=3).value = key_hash
        ws.cell(row=found_row, column=4).value = now
        ws.cell(row=found_row, column=5).value = key
    else:
        ws.append([file_name, algo, key_hash, now, key])
```

```python
    wb.save(META_FILE)
```

```python
def encrypt_file(file_path: Path, user_key: str):
    fernet = derive_fernet(user_key)
    with open(file_path, "rb") as f:
        data = f.read()
    token = fernet.encrypt(data)
    with open(file_path, "wb") as f:
        f.write(token)
    upsert_metadata(file_path.name, "Fernet", user_key)
```

```
def decrypt_file(file_path: Path, user_key: str) -> bool:
    fernet = derive_fernet(user_key)
    with open(file_path, "rb") as f:
        token = f.read()
    try:
        data = fernet.decrypt(token)
    except Exception:
        return False
    with open(file_path, "wb") as f:
        f.write(data)
    return True
```

```
# -----------------------------------------------------------------
# Text encryption (used by Password Vault)
# -----------------------------------------------------------------
def _get_vault_key() -> bytes:
    if not VAULT_KEY_FILE.exists():
        key = Fernet.generate_key()
        VAULT_KEY_FILE.write_bytes(key)
        return key
    return VAULT_KEY_FILE.read_bytes()
```

```
def encrypt_text(plaintext: str) -> str:
    if not plaintext:
        return ""
    fernet = Fernet(_get_vault_key())
    encrypted = fernet.encrypt(plaintext.encode("utf-8"))
    return encrypted.decode("utf-8")
```

```
def decrypt_text(ciphertext: str) -> str:
    if not ciphertext:
        return ""
    fernet = Fernet(_get_vault_key())
    decrypted = fernet.decrypt(ciphertext.encode("utf-8"))
    return decrypted.decode("utf-8")
```

x. folder_locker.py

```
import os
import subprocess
from datetime import datetime
from pathlib import Path
from openpyxl import Workbook, load_workbook
from config import BASE_DIR

LOCK_LOG = BASE_DIR / "data" / "LockedFolders.xlsx"
LOCK_LOG.parent.mkdir(parents=True, exist_ok=True)
```

```
def ensure_log_exists():
    if not LOCK_LOG.exists():
        wb = Workbook()
        ws = wb.active
```

```python
        ws.title = "LockedFolders"
        ws.append(["Folder Path", "Status", "Date-Time"])
        wb.save(LOCK_LOG)


def update_log(folder_path, status):
    ensure_log_exists()
    wb = load_workbook(LOCK_LOG)
    ws = wb.active

    found = False
    for row in ws.iter_rows(min_row=2, values_only=False):
        if row[0].value == folder_path:
            row[1].value = status
            row[2].value = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            found = True
            break

    if not found:
        ws.append([folder_path, status, datetime.now().strftime("%Y-%m-%d %H:%M:%S")])

    wb.save(LOCK_LOG)


def lock_folder(folder_path):
    if not os.path.exists(folder_path):
        return False, "❌ Folder not found."

    try:
        username = os.getlogin()
        cmd = f'icacls "{folder_path}" /deny {username}:(RX)'
        result = subprocess.run(cmd, shell=True, capture_output=True, text=True)

        if result.returncode == 0:
            update_log(folder_path, "Locked")
            return True, f"✅ Folder locked: {folder_path}"
        else:
            return False, f"  Failed: {result.stderr.strip()}"

    except Exception as e:
        return False, f"  Error: {str(e)}"


def unlock_folder(folder_path):
    if not os.path.exists(folder_path):
        return False, "❌ Folder not found."

    try:
        username = os.getlogin()
        cmd = f'icacls "{folder_path}" /remove:d {username}'
        result = subprocess.run(cmd, shell=True, capture_output=True, text=True)

        if result.returncode == 0:
            update_log(folder_path, "Unlocked")
            return True, f"  Folder unlocked: {folder_path}"
        else:
            return False, f"  Failed: {result.stderr.strip()}"

    except Exception as e:
        return False, f"  Error: {str(e)}"


def list_locked_folders():
    ensure_log_exists()
    wb = load_workbook(LOCK_LOG)
    ws = wb.active
    folders = []
    for row in ws.iter_rows(min_row=2, values_only=True):
        if not row[0]:
            continue
        folders.append({
            "path": row[0],
            "status": row[1],
            "datetime": row[2],
        })
```

```python
    return folders
```

```python
def count_locked_folders():
    folders = list_locked_folders()
    return sum(1 for f in folders if f["status"] == "Locked")
```

xi. login_manager.py

```python
import json, base64, secrets, hashlib, hmac
from pathlib import Path
from functools import wraps
from flask import session, redirect, url_for, flash
from config import PASS_HASH_FILE

SALT_LEN = 16
ITERATIONS = 200_000
```

```python
def is_master_set() -> bool:
    return PASS_HASH_FILE.exists()
```

```python
def create_master_passkey(passphrase: str) -> bool:
    salt = secrets.token_bytes(SALT_LEN)
    hash_bytes = hashlib.pbkdf2_hmac("sha256", passphrase.encode("utf-8"), salt, ITERATIONS)
    data = {
        "salt": base64.b64encode(salt).decode(),
        "hash": hash_bytes.hex(),
        "iterations": ITERATIONS
    }
    PASS_HASH_FILE.write_text(json.dumps(data))
    return True
```

```python
def verify_master_passkey(passphrase: str) -> bool:
    if not PASS_HASH_FILE.exists():
        return False
    data = json.loads(PASS_HASH_FILE.read_text())
    salt = base64.b64decode(data["salt"])
    expected = bytes.fromhex(data["hash"])
    iters = data.get("iterations", ITERATIONS)
    candidate = hashlib.pbkdf2_hmac("sha256", passphrase.encode("utf-8"), salt, iters)
    return hmac.compare_digest(candidate, expected)
```

```python
def login_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        if not session.get("authenticated"):
            flash("You must be logged in to view that page.")
            return redirect(url_for("Login"))
        return f(*args, **kwargs)
    return decorated
```

xii. notes_manager.py

```python
from pathlib import Path
from datetime import datetime
from docx import Document
from config import BASE_DIR
from backend.encryption_utils import generate_random_key, encrypt_file, decrypt_file

NOTES_DIR = BASE_DIR / "data" / "notes"

def ensure_notes_dir():
    NOTES_DIR.mkdir(parents=True, exist_ok=True)


def save_note(title: str, content: str):
    ensure_notes_dir()

    safe = "".join(c for c in title if c.isalnum() or c in (" ", "_", "-")).rstrip()
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    file_name = f"{safe.replace(' ', '_')}_{timestamp}.docx"
    file_path = NOTES_DIR / file_name


    doc = Document()
    doc.add_heading(title, level=1)
    doc.add_paragraph(content)
    doc.save(file_path)


    key = generate_random_key()
    encrypt_file(file_path, key)

    return file_name, key


def list_notes():
    ensure_notes_dir()
    notes = [f.name for f in NOTES_DIR.glob("*.docx")]
    return sorted(notes, reverse=True)


def load_note_content(filename: str, user_key: str):
    ensure_notes_dir()
    file_path = NOTES_DIR / filename

    if not file_path.exists():
        return None, None

    ok = decrypt_file(file_path, user_key)
    if not ok:
        return None, None

    doc = Document(file_path)
    title = doc.paragraphs[0].text if doc.paragraphs else filename
    content = "\n".join(p.text for p in doc.paragraphs[1:])
    encrypt_file(file_path, user_key)

    return title, content


def update_note(filename: str, current_key: str, new_title: str, new_content: str):
    ensure_notes_dir()
    file_path = NOTES_DIR / filename

    if not file_path.exists():
        raise FileNotFoundError(f"Note not found: {file_path}")

    ok = decrypt_file(file_path, current_key)
    if not ok:
        raise ValueError("Incorrect current key")

    doc = Document()
    doc.add_heading(new_title, level=1)
    doc.add_paragraph(new_content)
    doc.save(file_path)

    new_key = generate_random_key()
    encrypt_file(file_path, new_key)

    return new_key

def count_notes():
    ensure_notes_dir()
    return len(list(NOTES_DIR.glob("*.docx")))
```

xiii. vault_manager.py

```python
import os
from datetime import datetime
from openpyxl import Workbook, load_workbook
from config import BASE_DIR
from backend.encryption_utils import encrypt_text, decrypt_text

VAULT_FILE = BASE_DIR / "data" / "PasswordVault.xlsx"
VAULT_FILE.parent.mkdir(parents=True, exist_ok=True)

def ensure_vault_exists():
    if not VAULT_FILE.exists():
        wb = Workbook()
        ws = wb.active
        ws.title = "Vault"
        ws.append(["Website", "Name", "Email/Username/Phone", "Password", "Category", "Date"])
        wb.save(VAULT_FILE)

def save_entry(website: str, name: str, contact: str, password: str, category: str) -> tuple[bool, str]:
    ensure_vault_exists()
    try:
        encrypted_pass = encrypt_text(password)
        wb = load_workbook(VAULT_FILE)
        ws = wb.active
        ws.append([website, name, contact, encrypted_pass, category, datetime.now().strftime("%Y-%m-%d %H:%M:%S")])
        wb.save(VAULT_FILE)

        return True, "✅ Password entry saved successfully."
    except Exception as e:
        return False, f"❌ Error saving entry: {e}"

def list_entries():
    ensure_vault_exists()
    wb = load_workbook(VAULT_FILE)
    ws = wb.active

    entries = []
    for row in ws.iter_rows(min_row=2, values_only=True):
        website, name, contact, enc_pass, category, date = row
        try:
            decrypted_pass = decrypt_text(enc_pass)
        except Exception:
            decrypted_pass = "  Error"
        entries.append({
            "website": website,
            "name": name,
            "contact": contact,
            "password": decrypted_pass,
            "category": category,
            "date": date
        })
    return entries
```

```
def count_vault_entries() -> int:
    ensure_vault_exists()
    wb = load_workbook(VAULT_FILE)
    ws = wb.active
    return ws.max_row - 1
```

```
def update_entry(entry_id: int, website: str, name: str, contact: str, password: str, category: str) -> tuple[bool, str]:
    """Update an existing vault entry by row index (1-based excluding header)."""
    ensure_vault_exists()
    try:
        wb = load_workbook(VAULT_FILE)
        ws = wb.active
```

```
        # +1 to skip header row
        row_idx = entry_id + 1
        if row_idx > ws.max_row:
            return False, "❌ Entry not found."
```

```
        encrypted_pass = encrypt_text(password)
        ws.cell(row=row_idx, column=1).value = website
        ws.cell(row=row_idx, column=2).value = name
        ws.cell(row=row_idx, column=3).value = contact
        ws.cell(row=row_idx, column=4).value = encrypted_pass
        ws.cell(row=row_idx, column=5).value = category
        ws.cell(row=row_idx, column=6).value = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

```
        wb.save(VAULT_FILE)
        return True, "✅ Entry updated successfully."
    except Exception as e:
        return False, f"❌ Error updating entry: {e}"
```

xiv. app.py

```
from flask import Flask, render_template, request, redirect, url_for, flash, session
from backend.folder_locker import lock_folder, unlock_folder, list_locked_folders
from backend.notes_manager import count_notes
from backend.vault_manager import save_entry, list_entries, count_vault_entries, update_entry
from config import SECRET_KEY
from backend.login_manager import (
    is_master_set,
    create_master_passkey,
    verify_master_passkey,
    login_required,
)
from backend.notes_manager import (
    save_note,
    list_notes,
```

```python
    load_note_content,
    update_note,
)



app = Flask(__name__, template_folder="templates", static_folder="static")
app.secret_key = SECRET_KEY
app.config["SESSION_COOKIE_HTTPONLY"] = True
app.config["SESSION_COOKIE_SAMESITE"] = "Lax"
```

```python
# ---------------------------
# Inject global master key flag
# ---------------------------
@app.context_processor
def inject_master_flag():
    return {"master_set": is_master_set()}
```

```python
# ---------------------------
# Landing page
# ---------------------------
@app.route("/", endpoint="Landing")
def Landing():
    return render_template("Landing.html")
```

```python
# ---------------------------
# Setup master key
# ---------------------------
@app.route("/setup", methods=["GET", "POST"], endpoint="CreatePasskey")
def CreatePasskey():
    if is_master_set():
        flash("Master passkey already created — please login.")
        return redirect(url_for("Login"))
```

```python
    if request.method == "POST":
        p1 = request.form.get("pass1", "")
        p2 = request.form.get("pass2", "")
        if not p1 or p1 != p2:
            flash("Passkeys empty or do not match.")
            return redirect(url_for("CreatePasskey"))
```

```python
        create_master_passkey(p1)
        flash("Master passkey created. Please log in.")
        return redirect(url_for("Login"))
```

```python
    return render_template("CreatePasskey.html")
```

```python
# ---------------------------
# Login
# ---------------------------
@app.route("/login", methods=["GET", "POST"], endpoint="Login")
def Login():
    if request.method == "POST":
        passphrase = request.form.get("pass", "")
        if verify_master_passkey(passphrase):
            session.clear()
            session["authenticated"] = True
            flash("Login successful.")
            return redirect(url_for("dashboard"))
        else:
            flash("Invalid passkey.")
            return redirect(url_for("Login"))
    return render_template("Login.html")
```

```python
# --------------------------
# Logout
# --------------------------
@app.route("/logout", endpoint="Logout")
def Logout():
    session.clear()
    flash("Logged out.")
    return redirect(url_for("Landing"))
```

```python
# --------------------------
# Dashboard
# --------------------------
@app.route("/dashboard")
@login_required
def dashboard():
    locked_folders = list_locked_folders()
    notes_count = count_notes()

    stats = {
        "locked_folders": sum(1 for f in locked_folders if f["status"] == "Locked"),
       "vault_entries": count_vault_entries(),
        "notes": notes_count,
    }

    return render_template("dashboard.html", stats=stats)
```

```python
# --------------------------
# Folder Locker
# --------------------------
@app.route("/folder-locker", methods=["GET", "POST"], endpoint="FolderLocker")
@login_required
def FolderLocker():
    message = None
    if request.method == "POST":
        folder_path = request.form.get("folder_path", "").strip()
        action = request.form.get("action", "")

        if not folder_path:
            flash("Folder path is required.")
            return redirect(url_for("FolderLocker"))

        if action == "lock":
            success, message = lock_folder(folder_path)
        elif action == "unlock":
            success, message = unlock_folder(folder_path)
        else:
            message = "Invalid action."

        flash(message)

    folders = list_locked_folders()
    return render_template("FolderLocker.html", folders=folders, message=message)
```

```python
# --------------------------
# Notes Home
# --------------------------
@app.route("/notes", methods=["GET"], endpoint="notes_home")
@login_required
def notes_home():
    notes = list_notes()
    return render_template(
        "NotesManager.html",
        notes=notes,
        active_note=None,
        title="",
        content="",
        new=False,
        show_key_prompt=False,
    )
```

```python
# ---------------------------
# Open a note
# ---------------------------
@app.route("/notes/open/<filename>", methods=["GET", "POST"], endpoint="open_note")
@login_required
def open_note(filename):
    notes = list_notes()
```

```python
    if request.method == "POST":
        key = request.form.get("key", "")
        title, content = load_note_content(filename, key)
```

```python
        if title is None:
            flash("Invalid key or decryption failed.")
            return redirect(url_for("notes_home"))
```

```python
        return render_template(
            "NotesManager.html",
            notes=notes,
            active_note=filename,
            title=title,
            content=content,
            editable=False,
            current_key=key,
            show_key_prompt=False,
        )
    return render_template(
        "NotesManager.html",
        notes=notes,
        active_note=filename,
        show_key_prompt=True,
        key_prompt_for=filename,
    )
```

```python
# ---------------------------
# Edit and save note
# ---------------------------
@app.route("/notes/edit/<filename>", methods=["POST"], endpoint="edit_note")
@login_required
def edit_note(filename):
    current_key = request.form.get("current_key", "")
    new_title = request.form.get("title", "")
    new_content = request.form.get("content", "")
```

```python
    try:
        new_key = update_note(filename, current_key, new_title, new_content)
    except ValueError:
        flash("Current key incorrect. Could not save.")
        return redirect(url_for("open_note", filename=filename))
    except Exception as e:
        flash(f"Error saving note: {e}")
        return redirect(url_for("open_note", filename=filename))
```

```python
    flash(f"Note updated and re-encrypted with a new key: {new_key}")
    return redirect(url_for("notes_home"))
```

```python
# ---------------------------
# Create new note
# ---------------------------
@app.route("/notes/new", methods=["GET", "POST"], endpoint="create_new_note")
@login_required
def create_new_note():
    if request.method == "POST":
        title = request.form.get("title", "").strip()
        content = request.form.get("content", "").strip()

        if not title or not content:
            flash("Title and content required.")
            return redirect(url_for("create_new_note"))

        file_name, key = save_note(title, content)
        flash(f"Note created and encrypted successfully. File: {file_name}, Key: {key}")
        return redirect(url_for("notes_home"))

    notes = list_notes()
    return render_template(
        "NotesManager.html",
        notes=notes,
        new=True,
        active_note=None,
        title="",
        content="",
        editable=True,
        show_key_prompt=False,
    )
# ---------------------------
# Password Vault Page
# ---------------------------
@app.route("/password-vault", methods=["GET", "POST"])
@login_required
def password_vault():
    """Password Vault – Add and View Passwords."""
    if request.method == "POST":
        website = request.form.get("website", "").strip()
        name = request.form.get("name", "").strip()
        contact = request.form.get("contact", "").strip()
        password = request.form.get("password", "").strip()
        category = request.form.get("category", "Other")

        if not website or not password:
            flash("  Website and Password fields are required.")
            return redirect(url_for("password_vault"))

        success, message = save_entry(website, name, contact, password, category)
        flash(message)
        return redirect(url_for("password_vault"))

    # GET – list all vault entries
    vault_entries = list_entries()
    return render_template("PasswordVault.html", vault_entries=vault_entries)


@app.route("/vault", methods=["GET"], endpoint="PasswordVault")
@login_required
def PasswordVault():
    """Display all saved encrypted passwords."""
    entries = list_entries()
    return render_template("PasswordVault.html", vault_entries=entries)


@app.route("/vault/add", methods=["POST"], endpoint="AddVaultEntry")
@login_required
def AddVaultEntry():
    """Handle new password form submission."""
    website = request.form.get("website", "").strip()
    name = request.form.get("name", "").strip()
    contact = request.form.get("contact", "").strip()
    password = request.form.get("password", "").strip()
    category = request.form.get("category", "Other")
```

```
    if not (website and name and password):
        flash("Website, Name, and Password are required.")
        return redirect(url_for("PasswordVault"))
```

```
    success, msg = save_entry(website, name, contact, password, category)
    flash(msg)
    return redirect(url_for("PasswordVault"))
```

```
# --------------------------
# Edit Vault Entry
# --------------------------
@app.route("/password-vault/edit/<int:entry_id>", methods=["POST"])
@login_required
def edit_vault_entry(entry_id):
    """Update an existing password entry."""
    website = request.form.get("website", "").strip()
    name = request.form.get("name", "").strip()
    contact = request.form.get("contact", "").strip()
    password = request.form.get("password", "").strip()
    category = request.form.get("category", "Other")
```

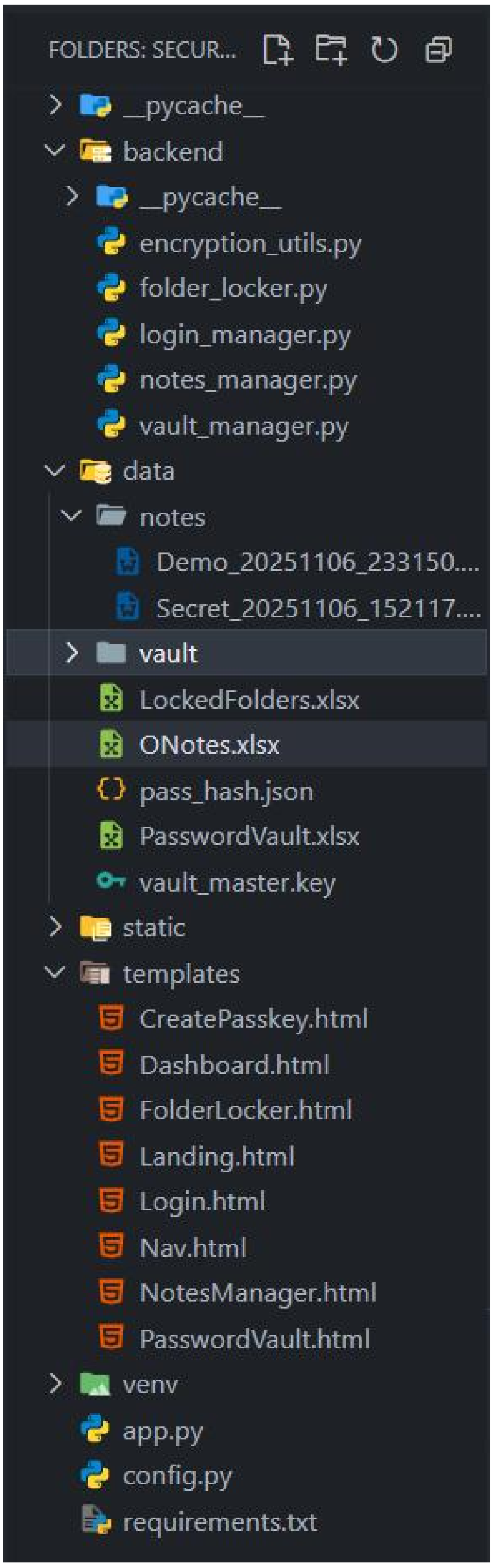
```
    success, msg = update_entry(entry_id, website, name, contact, password, category)
    flash(msg)
    return redirect(url_for("password_vault"))
```

```
# --------------------------
# Debug route list
# --------------------------
@app.cli.command("routes")
def list_routes():
    """List all registered routes — run with `flask routes`."""
    import urllib
    output = []
    for rule in app.url_map.iter_rules():
        methods = ",".join(rule.methods)
        url = urllib.parse.unquote(f"{rule}")
        output.append(f"{rule.endpoint:20s} {methods:25s} {url}")
    for line in sorted(output):
        print(line)
```

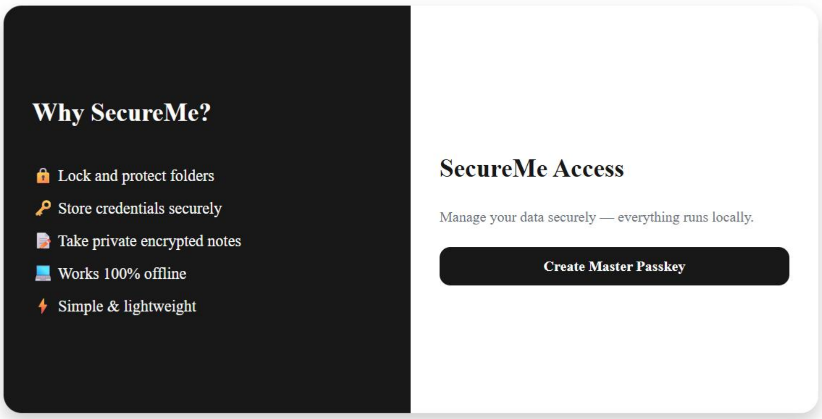
```
# --------------------------
# Run the app
# --------------------------
if __name__ == "__main__":
    app.run(debug=True)
```

8. Screen Shots

Folder Structure:



```
FOLDERS: SECUR...
> __pycache__
v backend
  > __pycache__
    encryption_utils.py
    folder_locker.py
    login_manager.py
    notes_manager.py
    vault_manager.py
v data
  v notes
      Demo_20251106_233150....
      Secret_20251106_152117....
  > vault
    LockedFolders.xlsx
    ONotes.xlsx
    pass_hash.json
    PasswordVault.xlsx
    vault_master.key
> static
v templates
    CreatePasskey.html
    Dashboard.html
    FolderLocker.html
    Landing.html
    Login.html
    Nav.html
    NotesManager.html
    PasswordVault.html
> venv
  app.py
  config.py
  requirements.txt
```

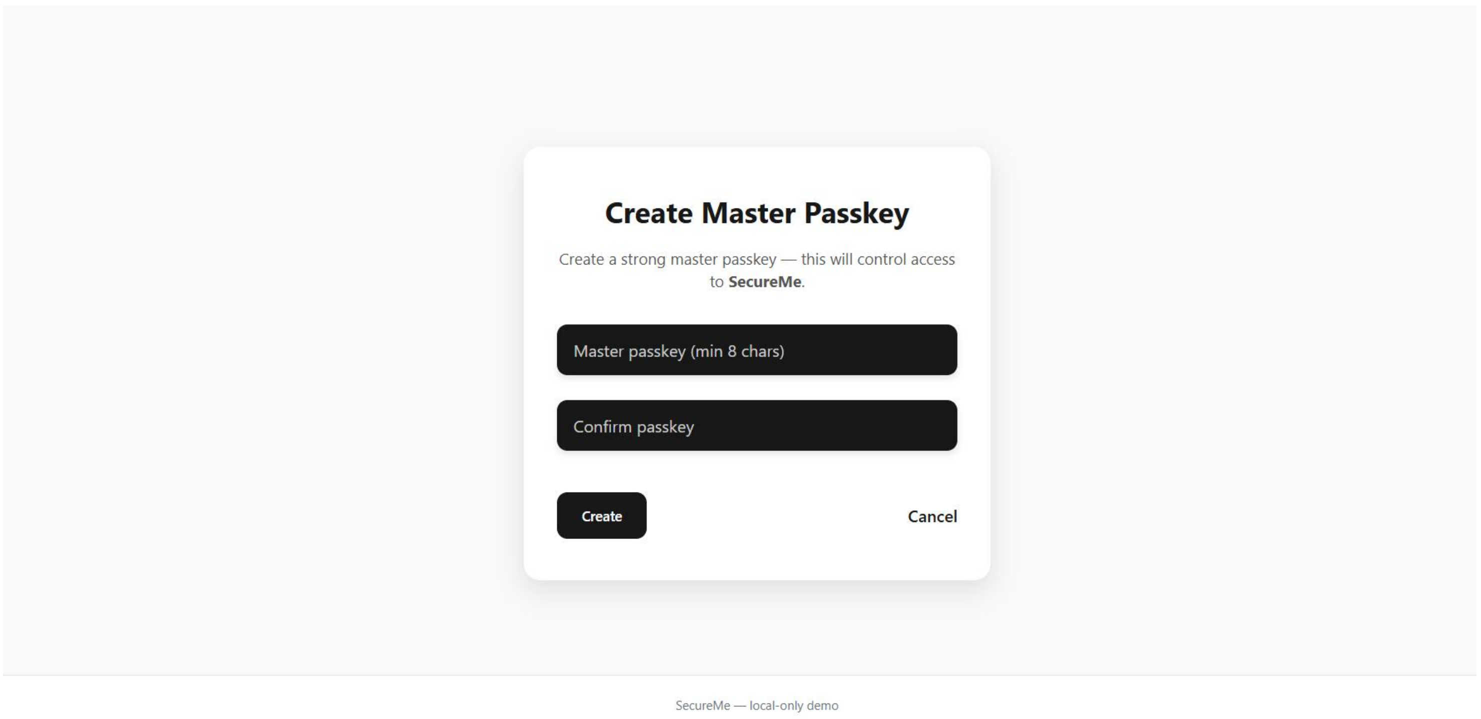


Why SecureMe?

- 🔒 Lock and protect folders
- 🔑 Store credentials securely
- 📝 Take private encrypted notes
- 💻 Works 100% offline
- ⚡ Simple & lightweight

SecureMe Access

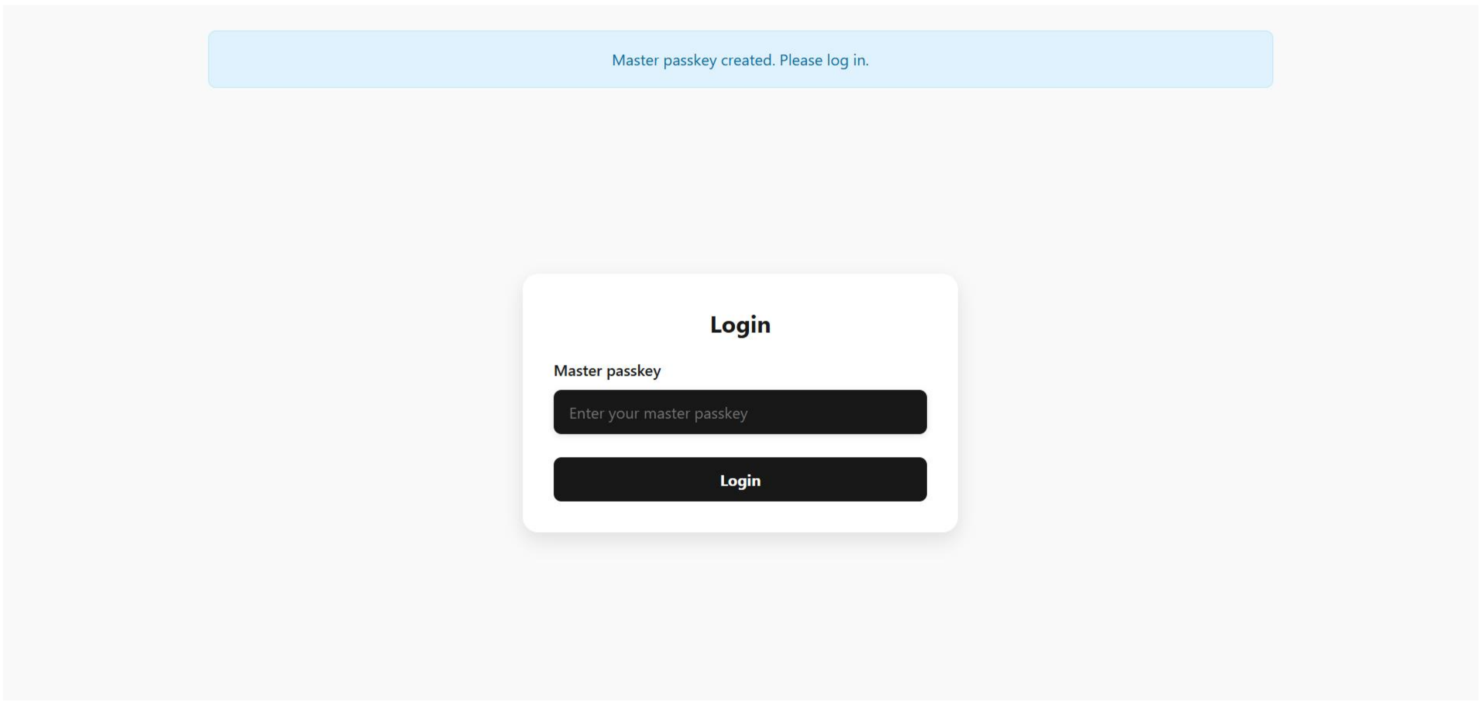
Manage your data securely — everything runs locally.



Create Master Passkey

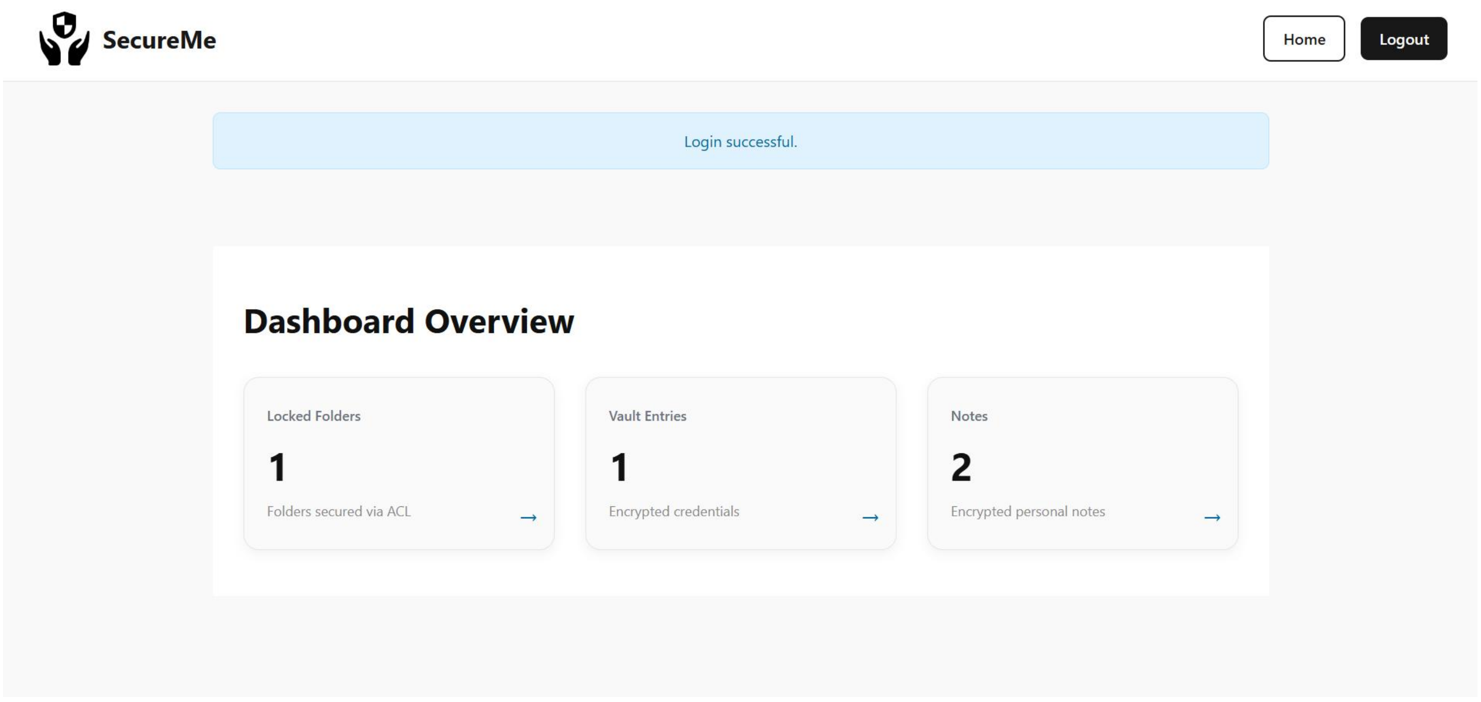
Create a strong master passkey — this will control access to **SecureMe**.

SecureMe — local-only demo


Master passkey created. Please log in.

Login

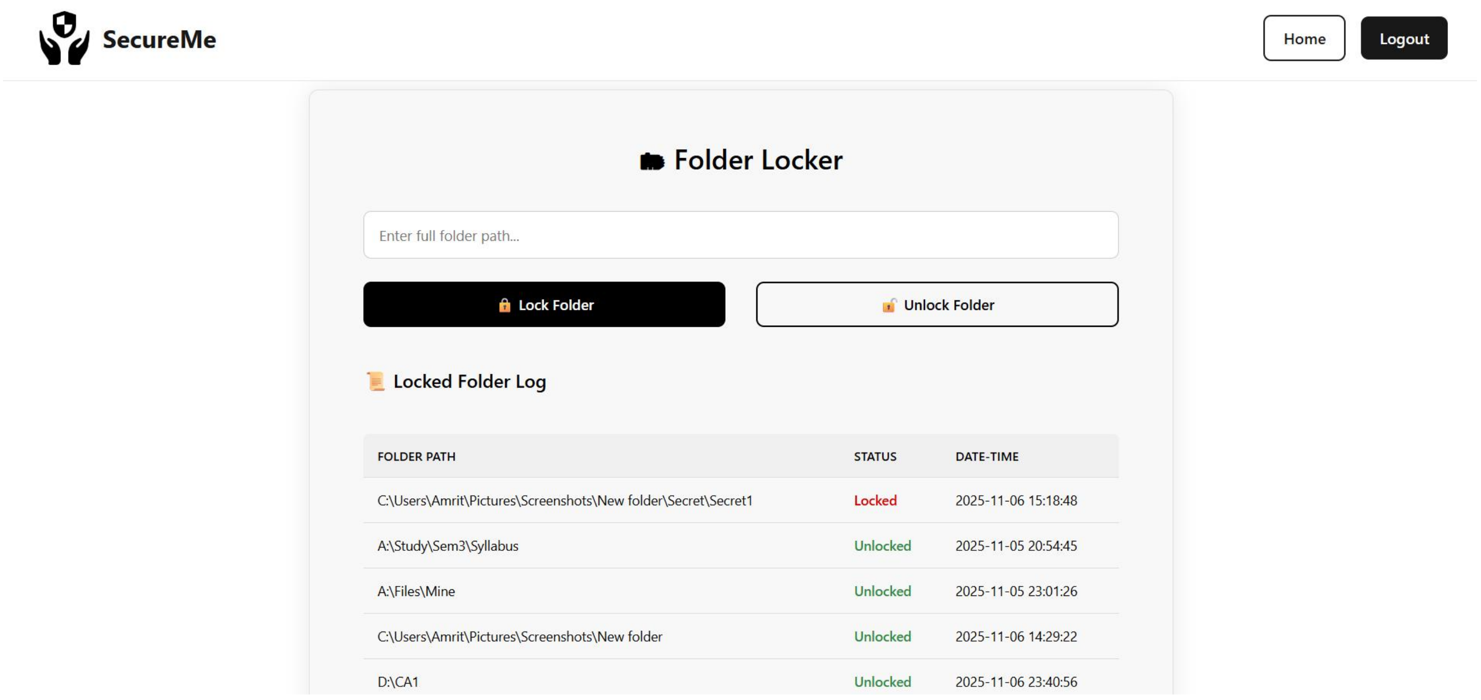
Master passkey



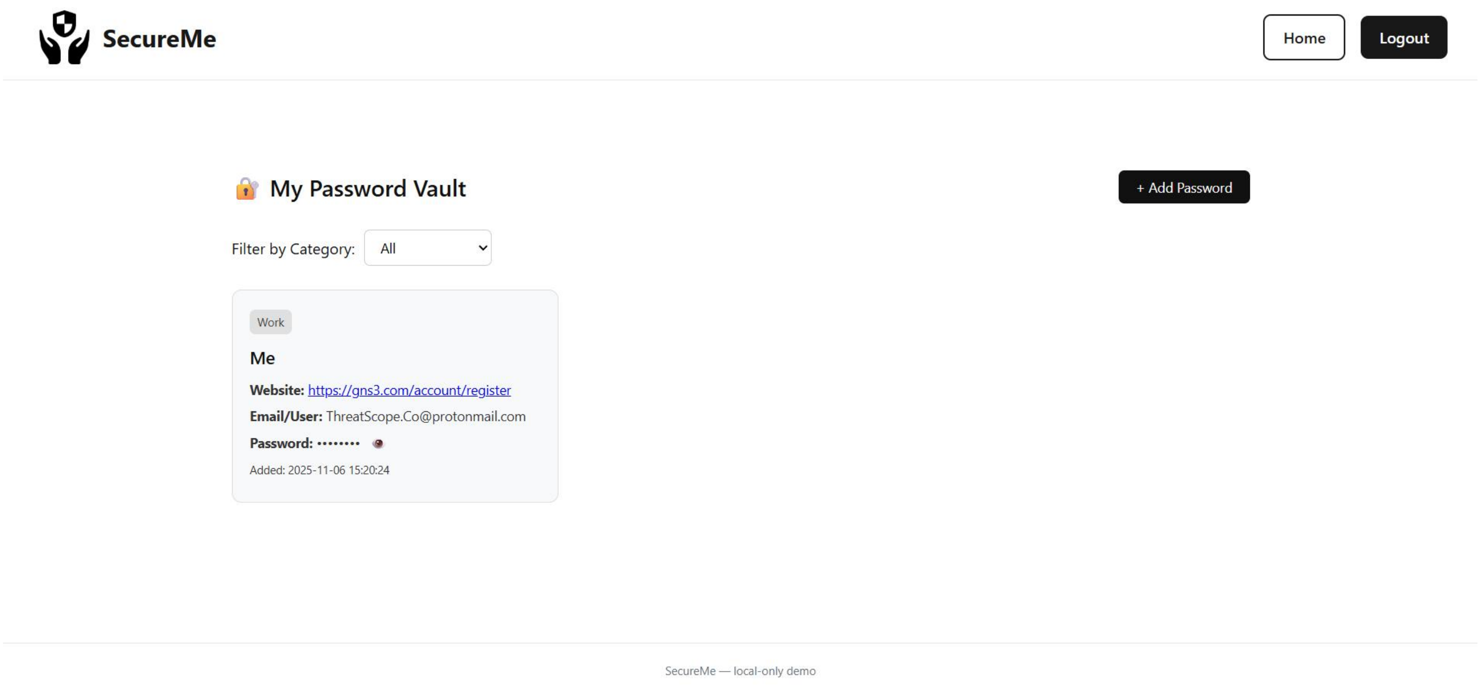
Login successful.

Dashboard Overview

Locked Folders	Vault Entries	Notes
1	1	2
Folders secured via ACL →	Encrypted credentials →	Encrypted personal notes →



FOLDER PATH	STATUS	DATE-TIME
C:\Users\Amrit\Pictures\Screenshots\New folder\Secret\Secret1	Locked	2025-11-06 15:18:48
A:\Study\Sem3\Syllabus	Unlocked	2025-11-05 20:54:45
A:\Files\Mine	Unlocked	2025-11-05 23:01:26
C:\Users\Amrit\Pictures\Screenshots\New folder	Unlocked	2025-11-06 14:29:22
D:\CA1	Unlocked	2025-11-06 23:40:56



Me

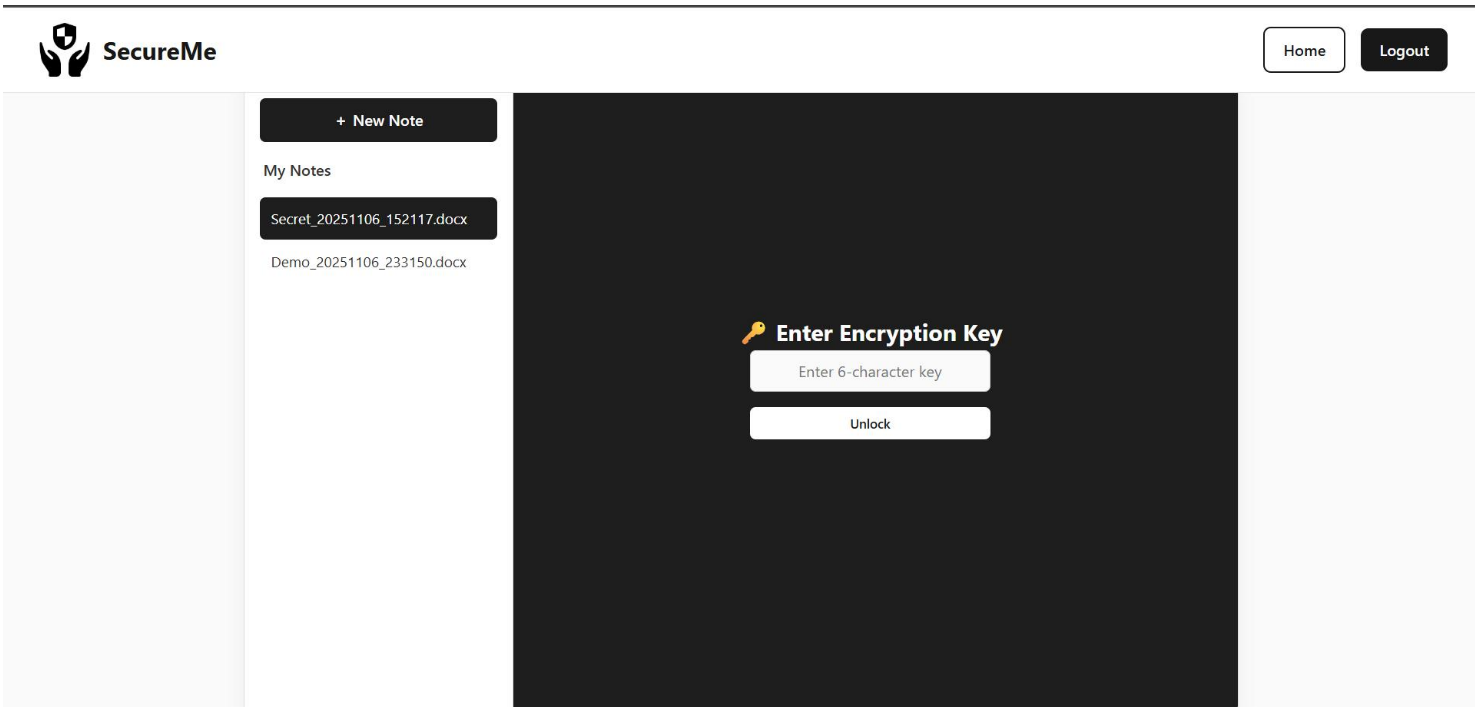
Website: https://gns3.com/account/register

Email/User: ThreatScope.Co@protonmail.com

Password: ••••••••

Added: 2025-11-06 15:20:24

SecureMe — local-only demo



9. Future Scope

The SecureMe Platform is designed as a robust foundation for personal data security, and there are several ways it can be enhanced to expand functionality, usability, and security:

- Desktop Application Development:
  Transforming the web-based Flask platform into a full-fledged desktop application using frameworks like PyQt, Tkinter, or Electron will allow offline usage without relying on a browser.
  This will also provide better folder locking and direct access to system files.

- Improved User Interface (UI):
  Designing a modern, intuitive, and responsive interface with dashboards, icons, and guided workflows will improve user experience.
  Drag-and-drop functionality for files, richer note-editing tools, and easier password vault management can be included.

- Biometric Authentication:
  In addition to passkey login, fingerprint, facial recognition, or voice-based authentication can be implemented for stronger security.

Multi-factor authentication combining passkey + biometrics can prevent unauthorized access.

- Role-Based Access Control (RBAC):
  For multi-user environments, implementing role-based permissions will allow different access levels. This can be useful for small teams, families, or shared computers, ensuring sensitive data is accessed only by authorized roles.

- Secure File Sharing:
  Adding encrypted file sharing between users (locally or via secure cloud links) can allow collaborative usage without compromising security.
  Files can be shared with temporary access keys or self-expiring links for added protection.

- Advanced Cryptographic Algorithms:
  Upgrading encryption from standard AES/Fernet to hybrid cryptography or post-quantum resistant algorithms can further strengthen data protection.
  Periodic key rotation, salt updates, and stronger hashing methods can also enhance security resilience.

10. Conclusion

The SecureMe Platform files, passwords and notes manager is an overall tool to protect the risk of data loss in modern times with technologies of this sort. The project makes a strong emphasis on robust security, usability, and modularity which are the key properties of this multifunctional application that incorporates locking folders, managing password vaults, and storing notes in an encrypted form in them. The system is constructed with use of Flask as a web-based interface, which employs the use of strong cryptographic algorithms that will keep all user data confidential and tamper-immune. With metadata tracking, an added level of accountability is created, enabling users to check on what encryption activities are taking place and ensuring that the users retain control of their data.

All the modules of the platform help in ensuring the general security as well as the usability of the system. The User Login Module has the functionality to make sure that only authorized users will get access to the platform as they should have a passkey. The Folder Lock Module locks down folders using encryption and system-based levels of permissions whereas the Password Vault Module encrypts and locks the login credentials in excel files. Secret Notes Module permits users to create and remember personal notes. This combination of modules is what gives users the experience of a connected ecosystem as they are able to control how their digital assets are handled and at the same time feel safe against loss through unauthorized access.

In the future, the platform has a great potential to be improved and scaled. There is also the possibility of upgrade in a single desktop client, user interface enhancement to ensure easy usage, the use of bio-metric authentication, the incorporation of the use of role-based user access control where multiple individuals may use the platform and the feature of secure file sharing and high cryptography algorithm integration. Such improvements would widen the platform making it meet the individual needs as well as collaborative or organizational applications. The platform shows that combining the rigidity of security with the agility of future development, a stable, scalable, and user-friendly platform can be developed that protects digital data.

References

[1] *S. Author, "Title of the Paper," International Journal of Research Publication and Reviews, vol. 6, no. 4, pp. xxx–xxx, 2024. [Online]. Available: https://ijrpr.com/uploads/V6ISSUE4/IJRPR42315.pdf*

[2] *TechReviewAdvisor, "Encryption vs Decryption," 2024. [Online]. Available: https://techreviewadvisor.com/encryption-vs-decryption/*

[3] *CodeRivers, "Python Encryption Library," 2024. [Online]. Available: https://coderivers.org/blog/python-encryption-library/*