Project Report:

Password Generator and Manager

Course: Vityarthi Project
Name: Tarushi Taanvi
Registration No.: 25BAI10343
Institution: VIT Bhopal University
Year: 2025

1. Introduction

This project describes the design of a user-friendly, secure **Password Generator and Manager**. The tool is designed in Python with the Tkinter GUI library to address the pressing need of users to generate strong and unique passwords and store them securely for multiple websites. The application also has the functionality of generating strong passwords automatically and storing credentials locally.

2. Problem Statement

The approach to security in the modern digital landscape largely depends upon password strength and uniqueness. Users tend to reuse simple passwords, thereby exposing themselves to credential-stuffing and brute-force attacks. This problem has twofold aspects: a difficulty in creating truly random, complex passwords and a lack of having a simple, local mechanism to store them safely.

This project solves this by automating the creation of secure passwords and providing a user-friendly interface for immediate local logging of website credentials.

3. Functional Requirements

ID	Requirement	Description
FR-01	Strong Password Generation	The password generated should be a string of a balanced mix of letters, both upper and lower case, with numbers and symbols.
FR-02	Password Storage	Must save the generated website, email, and password credentials to a local text file.
FR-03	Graphical User Interface (GUI)	The interface should clearly provide a graphical, interactive means of data input and function execution using Tkinter.
FR-04	Input Validation	Should be checked does Website and Password fields are empty before attempting to save.
FR-05	User Confirmation	Must show a confirmation prompt (ask ok cancel) before writing data into the file permanently.

4. Non-Functional Requirements

ID	Requirement	Description
NFR-01	Usability	The GUI should be intuitive and user-friendly so that saving and generating passwords is effortless.
NFR-02	Reliability	Bias-free random strings must be generated by the password generator on each run.
NFR-03	Portability	The application should be executable on different operating systems, including anywhere Python and Tkinter are available.

5. System Architecture

The architecture is simple, represented as a Client-Side Desktop Application. A single Python script, password generator.py, embodies all the code and logic.

- **GUI Layer (Tkinter):** Responsible for the user input (Website, Email,

Password fields) and button events.
- **Logic Layer:** Python Functions include generate_password() for randomization and save() for data handling and file I/O.
- **Data Layer (Local File):** A simple text file used for the persistent of saved credentials.

6. Design Diagrams

Workflow Diagram

The primary workflow has two separate processes involved: Password Generation and Data Saving.

1. **Generate Password:** User clicks "Generate Password" → Python's random module selects character lengths → shuffle() mixes the list → String is formed → Password Entry field is updated.
2. **Save Credentials:** User clicks "Add" → GUI validates inputs → Confirmation dialog appears → If confirmed, format data as (Website | Email | Password) → Adds data to the local text file: python project 2 updated.txt.

7. Implementation Details

A. Technologies Used

- **Language:** Python
- **GUI Framework:** tkinter
- **Core Libraries:** random, tkinter.messagebox, PIL (Pillow) for image handling.

B. Password Generation Logic (FR-01)

The password creation ensures complexity by defining separate pools for letters, numbers, and symbols.

- **Length:** The length of the final password would range from 12 to 18 characters, given by 8-10 letters, 2-4 symbols, and 2-4 numbers.

- **Randomization:** List comprehensions and random.choice/random.randint are applied to choose the characters. random.shuffle() plays an important role in enhancing randomness and scrambling the character types within the string.

C. Data Persistence and Security (FR-02)

Here, data is written in plain text to the path: C:\Users\Tarushi Tanavi\Desktop\New folder\python project 2 updated.txt.

Note on Security: Currently, the data rests locally in plain text. For a production application, this file would have to be encrypted using either hashing or symmetric encryption methods to meet security best practices.

8. Challenges Encountered

1. **Image Dependency:** The existing code relies on hard-coded local paths for loading the logo image. This makes the application non-portable and very likely to fail when used on other machines.
2. **External Library:** Using PIL/Pillow to handle images introduces one additional dependency beyond the standard Python install.
3. **Plain Text Storage:** The current mechanism of storing plain text files is a serious security vulnerability that will need future enhancements in encryption.

9. Future Enhancements

1. **Encryption:** Implement AES-256 or similar encryption for the local storage file.
2. **Search Functionality:** Add a function and GUI element dedicated solely to searching the stored file for existing credentials based on the entered website name.
3. **JSON/CSV Storage:** This would involve moving away from a simple text file to a JSON or CSV format for readable and easily programmatically accessible data.

4. **Platform-Agnostic Image Loading:** The image asset should be in the project folder, and the code should use relative paths so the application can work on any computer.

10. Contact Information

- **Name:** Tarushi Taanvi
- **Registration No.:** 25BAI10343
- **Email:** tarushi.25bai10343@vitbhopal.ac.in
- **Institution:** VIT Bhopal University