

Password Manager - Generate & Check passwords (Console)

Problem Memorizing and generating new passwords can be annoying. Users tend to repeat rather than randomize. Also, there is no feedback on whether the password is good or bad. This is why users need a safe, local way to create and manage passwords. The tool should list available commands, let users customize password generation, keep records with usernames and platforms, and provide strength feedback — all without relying on external services.

Scenario The password generator solves part of the problem where the user is dependent on third party password generators. The user needs locally generated passwords. Complementary to the existing password manager a password generator is needed.

User stories:

1. As a user, I want to see the list of commands to see what the app is capable of.
2. As a user, I want to select generation options of the password.
3. As a user, I want to view the records of generated passwords along with their associated usernames and platforms.
4. As a user, I want to get feedback on the passwords strength.

Use cases:

- Show possible commands
- Generate password (possible options)
- Show password's strength
- Encrypt and decrypt passwords
- Access User and Password information (from `passwords.db`)
- Managing passwords

☒ Project Requirements

Each app must meet the following three criteria in order to be accepted (see also the official project guidelines PDF on Moodle):

1. Interactive app (console input)
2. Data validation (input checking)
3. File processing (read/write)

1. Interactive App (Console Input)

The application interacts with the user via the console. Users can:

- View the possible commands
- Generate passwords

- Manage users and passwords
 - Receive feedback on the strength of the passwords
-

2. Data Validation

The application validates all user input to ensure data integrity and a smooth user experience. This is implemented in `main.py` as follows:

- **Generate password (possible options):** When the user wants to create a password, the program checks if the input is a digit and within the valid menu range:

```
if not length.isdigit() or int(length) < MIN_LENGTH:
    print(f"⚠ Invalid password length. Password length must be at least
{MIN_LENGTH} characters length and fully digit.")
    continue
```

This ensures only valid menu items can be generated.

These checks prevent crashes and guide the user to provide correct input, matching the validation requirements described in the project guidelines.

3. File Processing

The application reads and writes data using files:

- **Input/Output file:** `vault.db` — Contains the entries, one item per line in the format `username, platform, password`.
 - The application interacts with this file depending on the command specified.
 - The application writes data to file when needed.

⚙ Implementation

Technology

- Python 3.x
- Environment: GitHub Codespaces
- Some PIP libraries

📁 Repository Structure

```
Password-Generator/
├── main.py           # main program logic (console application)
├── datastore.py      # SQLite handling
├── password_utils.py # Password encryption
├── input_utils.py    # Utils for inputs (integers, booleans, passwords, etc.)
├── datastore.py      # SQLite handling
└── vault.db          # database with passwords (the name of the file is an
```

```
environment variable)
├─ docs/           # optional screenshots or project documentation
└─ README.md       # project description and milestones
```

Installation (first time run)

`python3 -m pip install -r requirements.txt`

1. Install pip (if not installed):

```
python3 -m ensurepip --upgrade
```

2. Install dependencies:

```
python3 -m pip install requirements.txt
```

How to Run

1. Open the repository in **GitHub Codespaces**
2. Open the **Terminal**
3. Run:

```
python3 main.py
```

Libraries Used

- **os**: Used for file and path operations, such as checking if the menu file exists and creating new files.
- **dotenv**: Used to load local .env file.
- **cryptography**: Used to encrypt/decrypt passwords.

Team & Contributions

Name	Contribution
Andrii Vlasov	Data storage implementation and encryption
Aaron Casula	Data presentation and view logic
José Gédance	Password generation functionality

Contributing

- Use this repository as a starting point by importing it into your own GitHub account.
- Work only within your own copy — do not push to the original template.
- Commit regularly to track your progress.



License

This project is provided for **educational use only** as part of the Programming Foundations module.

[MIT License](#)