RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE
FACULTÉ D'INFORMATIQUE

Project Report: Password Manager Web Extension

Created by :

BAGHDALI ABDELMADJID
222231347703

MERZOUK MOUHAMED ISLAM
222231378419

Framed by :

ABDELHADI

Année universitaire :
2025-2026

1. Project Overview

This project is a ***Password Manager Web Extension*** designed to securely store, manage, and automatically fill user credentials. The main objective of the project is to provide a ***secure, reliable, and user-friendly solution*** for password management while respecting privacy principles.

The project is based on:

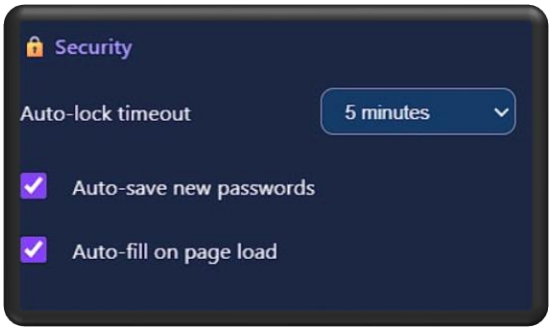
- A Web Extension interface for user interaction.
- A ***Python-based backend*** responsible for security, logic, and data handling.
- A local encrypted database for storing sensitive credentials.

The system is designed so that all sensitive data remains on the user's device, ensuring a high level of trust and security.

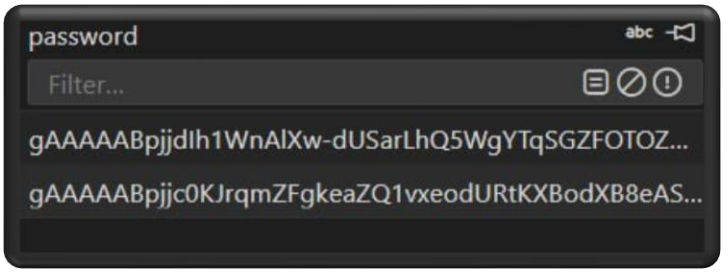
2. Features

2.1 Core Features

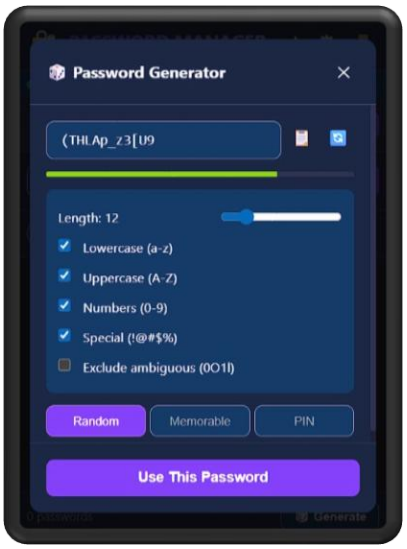
- Automatic saving of login credentials.
- Automatic filling of login forms.



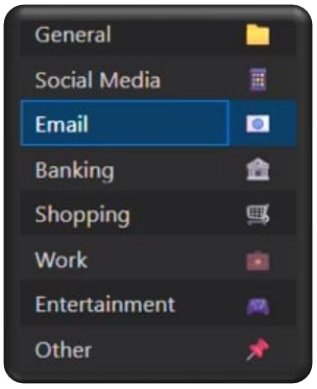
- Secure storage of passwords using strong encryption.



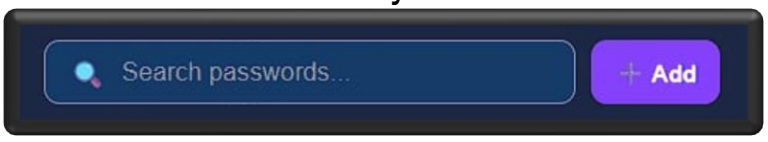
- Password generator for creating strong passwords.



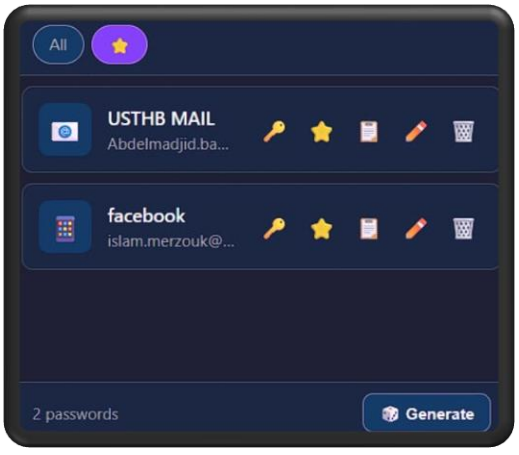
- Categorization of stored credentials.



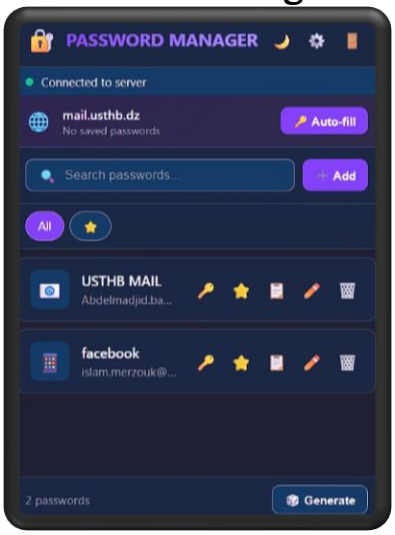
- Fast search functionality.

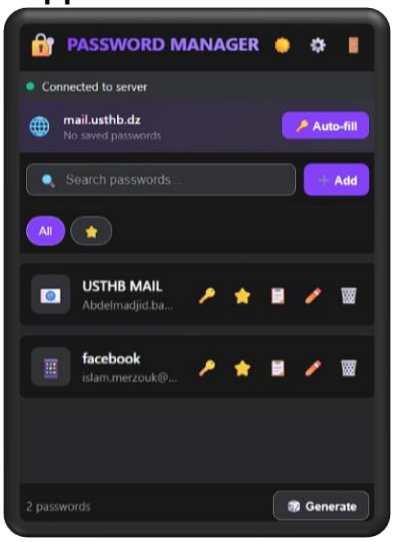


- Favorites system for quick access.

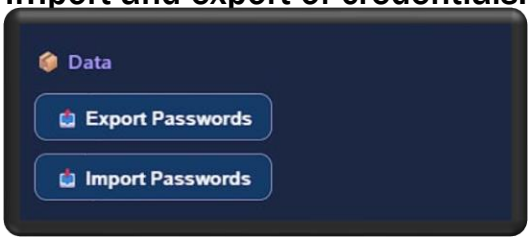


- Dark mode and light mode support.



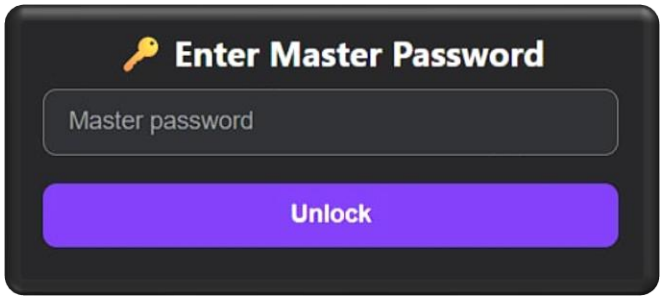


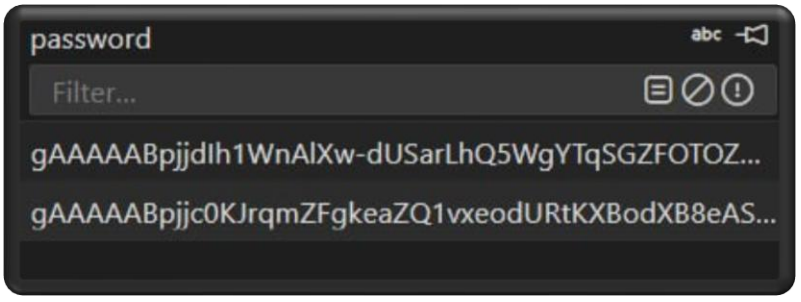
- Import and export of credentials.



2.2 Security Features

- Encrypted storage of all sensitive data.
- Master password protection.





- Secure authentication mechanism between the extension and backend

3. Project Structure

```
web_extension/
├── backend/
│   ├── app.py
│   ├── auth_manager.py
│   ├── database_manager.py
│   ├── password_generator.py
│   └── requirements.txt
│
├── passwords.db
├── start_server.py
├── README.md
├── USER_GUIDE.md
└── SETUP_GUIDE.md
```

4. File-Level Description

4.1 app.py

This file represents the ***main entry point of the backend server***. It initializes the server, defines API routes, and connects user requests with authentication and database logic.

Responsibilities:

- Server initialization.
- API routing.
- Request validation.
- Communication with authentication and database modules.

4.2 auth_manager.py

This file is responsible for ***authentication and security management***. It handles user verification and session security.

Responsibilities:

- Token generation.
- Token validation.
- Access control to protected resources.

4.3 database_manager.py

This file manages all interactions with the database.

Responsibilities:

- Database connection.
- Secure storage of credentials.
- Retrieval, update, and deletion of records.

4.4 password_generator.py

This file contains logic related to password generation.

Responsibilities:

- Generating strong passwords.
- Customizing password complexity.

4.5 start_server.py

This file provides a simple way to start the backend server.

Responsibilities:

- Launching the server.
- Preparing the environment for execution.

5. Database Description

- Database file: passwords.db.
- Stores encrypted credentials only.
- No sensitive data is stored in plain text.

6. Detailed Function Explanation

6.1 app.py – Function Explanation

The functions in this file handle the main application logic and API endpoints.

- ***initialize_app()***: Initializes the backend application and prepares required configurations.

- *register_routes()*: Defines and registers all API endpoints used by the extension.
- *handle_login_request()*: Receives login data from the extension and processes authentication.
- *handle_save_password()*: Receives credentials and forwards them to the database manager for secure storage.
- *handle_get_passwords()*: Retrieves stored credentials for auto-fill or user display.
- *handle_update_password()*: Updates existing credentials in the database.
- *handle_delete_password()*: Removes selected credentials from storage.

6.2 auth_manager.py – Function Explanation

The functions in this file ensure secure authentication and authorization.

- *generate_token()*: Creates a secure authentication token after successful verification.
- *verify_token()*: Validates incoming tokens to ensure request authenticity.
- *authenticate_user()*: Verifies user identity using the master password.
- *protect_route()*: Restricts access to sensitive endpoints.

6.3 database_manager.py – Function Explanation

The functions in this file manage encrypted data persistence.

- *connect_database()*: Establishes a secure connection to the database.
- *encrypt_password()*: Encrypts passwords before storage.
- *decrypt_password()*: Decrypts passwords when retrieval is required.
- *store_password()*: Saves new credentials into the database.
- *fetch_passwords()*: Retrieves stored credentials.
- *update_password()*: Updates existing records.

- *delete_password()*: Deletes credentials securely.
- *check_duplicates()*: Prevents storing duplicate credentials.

6.4 password_generator.py – Function Explanation

These functions handle password generation logic.

- *generate_password()*: Generates a strong random password.
- *set_password_length()*: Defines the desired password length.
- *include_special_characters()*: Enables or disables special characters.

6.5 start_server.py – Function Explanation

This file contains utility functions for running the server.

- *start_server()*: Launches the backend server and listens for requests.
- *load_environment()*: Prepares environment variables and configurations.

7. Conclusion

This project demonstrates a structured and secure approach to building a password manager. The clear separation between authentication, database management, and application logic makes the system scalable, maintainable, and suitable for efficient project verification.