PASSWORD MANAGER IN C

Project Report

Surya Prakash Sharma

SAP ID. 590027399

Institution: UPES Dehradun

Course: B.Tech Computer Science

Date: November 2025

1. Abstract

This project presents a console-based password manager developed in C, designed to store, retrieve, search, and delete login credentials locally. It emphasizes simplicity, modularity, and practical use of core programming concepts such as structures, file handling, and string manipulation.

The program uses plain text storage for credentials, making it suitable for personal use and academic demonstration. By avoiding external dependencies, it ensures a lightweight and controllable environment for data management.

2. Problem Definition

2.1 The Issue

Managing multiple login credentials securely and efficiently is a common challenge. Most users rely on browsers or third-party tools, which may be "overkill" for personal or offline use, or lack transparency in how data is stored.

2.2 The Objective

This project aims to build a lightweight, local password manager. It allows users to store and manage credentials without relying on external services, ensuring data ownership and reducing the attack surface associated with cloud storage.

3. System Architecture

The system operates on a cyclic executive model involving the following stages:

1. **Initialization:** Loading credentials from credentials.txt into memory.
2. **User Interface:** Displaying the menu (Add, View, Search, Delete, Exit).
3. **Operation:** Executing the selected function against the data structure.
4. **Persistence:** Writing any modifications back to the file system.
5. **Termination:** Clearing memory and exiting the program.

4. Algorithm: Add Credential

The core logic for adding a new entry involves the following steps:

1. Prompt the user for *Website*, *Email*, and *Password* strings.
2. Sanitize input by stripping newline characters (\n) to preserve file formatting.
3. Populate a Credential structure instance.
4. Append the structure to the in-memory array.
5. Execute the save_to_file() routine immediately to ensure data persistence.

5. Data Structure Implementation

The system utilizes a C typedef struct to encapsulate credential data:

```
4  typedef struct {
5      char website[50];
6      char email[50];
7      char password[50];
8  } Credential;
9
```

Fixed-size character arrays are chosen to simplify memory management and file I/O operations.

6. File Handling

6.1 Save Data

Data is written to disk using a pipe-delimited format:

```c
void save_to_file() {
    FILE *fp = fopen(FILE_NAME, "w");
    if (!fp) {
        printf("Error saving file!\n");
        return;
    }
    for (int i = 0; i < count; i++) {
        fprintf(fp, "%s|%s|%s\n",
                credentials[i].website,
                credentials[i].email,
                credentials[i].password);
    }
    fclose(fp);
```

This ensures that fields containing spaces are handled correctly.

6.2 Load Data

Data is read using formatted input scanning:

```c
void load_from_file() {
    FILE *fp = fopen(FILE_NAME, "r");
    if (!fp) return; // no file yet

    char line[200];
    while (fgets(line, sizeof(line), fp)) {
        Credential c;
        sscanf(line, "%49[^|]|%49[^|]|%49[^\n]",
               c.website, c.email, c.password);
        credentials[count++] = c;
    }
    fclose(fp);
```

This parses the file line-by-line, respecting the delimiters.

7. Testing and Validation

Test Case	Input	Expected Output	Result
1: Add	github	Data appended to file	**Pass**
2: Search	Query: github	"Found matching credential"	**Pass**
3: Delete	Target: github	Entry removed from storage	**Pass**

1. Add

```
PS C:\Users\Surya Prakash\OneDrive\Desktop\my code> cd "c:\Users\Surya Prakash\OneDrive\Desktop\my code\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
Γ£à Loaded 1 credentials from text file.

=== Password Manager (Plain Text Storage) ===
1. Add Credential
2. View All Credentials
3. Search by Website
4. Delete Credential
5. Exit
Enter your choice: 1
Website: Github
Email: suryaprkash778@gmail.com
Password: ryd9rvmhsj
Γ£à Credential saved (in memory).
Γ£à Credentials saved to text file.
```

2. Search

```
PS C:\Users\Surya Prakash\OneDrive\Desktop\my code> cd "c:\Users\Surya Prakash\OneDrive\Desktop\my code\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
Γ£à Loaded 2 credentials from text file.

=== Password Manager (Plain Text Storage) ===
1. Add Credential
2. View All Credentials
3. Search by Website
4. Delete Credential
5. Exit
Enter your choice: 3
Enter website to search: Github

Website: Github
Email: suryaprkash778@gmail.com
Password: ryd9rvmhsj
```

3. delete

```
PS C:\Users\Surya Prakash\OneDrive\Desktop\my code> cd "c:\Users\Surya Prakash\OneDrive\Desktop\my code\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
Γ£à Loaded 2 credentials from text file.

=== Password Manager (Plain Text Storage) ===
1. Add Credential
2. View All Credentials
3. Search by Website
4. Delete Credential
5. Exit
Enter your choice: 4
Enter website to delete: Github
Γ£à Credential deleted.
Γ£à Credentials saved to text file.
```

8. Conclusion and Future Work

Conclusion

This password manager successfully demonstrates how C can be used to build a modular application with persistent storage. It reinforces key skills in file I/O and data structuring.

Future Enhancements

- Implementation of AES Encryption.
- Master Password Authentication.
- SQLite Database Integration.
- GUI Development (GTK).

9. References

- Let us c by Yashavant Kanetkar
- TutorialsPoint. (2024). *File Handling in C*.
- GeeksforGeeks. (2024). *Structures and Pointers in C*.