# Password Manager Documentation

1. Introduction:

This document explains the design and functionality of a simple Password Manager written in C. The program demonstrates basic file operations, user input handling, and CRUD (Create, Read, Update, Delete) operations. It uses a master password for authentication and stores credentials in a text file named 'passwords.txt'.

2. Features:

- Add Credentials: Allows the user to store a website, username, and password.
- View Credentials: Reads and displays all stored credentials from the file.
- Delete Credential: Removes a specific credential (by website name) from the file.
- Update Credential: Modifies the username and password for an existing website entry.
- Search Credential: Searches and displays credentials for a specific website.
- Master Password Protection: Ensures that only authorized users can access the manager.

3. Code Overview:

a. Main Function:

- Prompts the user for a master password.
- If the password is correct, it displays a menu for the various operations.
- Depending on the user's choice, it calls the corresponding function.

b. addCredentials Function:

- Opens the file 'passwords.txt' in append mode.
- Prompts the user to enter a website, username, and password.
- Writes these details to the file in a space-separated format.

c. viewCredentials Function:

- Opens the file in read mode.
- Reads and displays each line (credential) from the file.

d. deleteCredential Function:

- Opens the existing file and a temporary file.
- Reads each credential and writes it to the temporary file unless it matches the specified website for deletion.
- Replaces the original file with the temporary file after deletion.

e. updateCredential Function:

- Similar to delete, it reads credentials and writes them to a temporary file.
- When the target website is found, it prompts for new username and password values and writes the updated entry.

f. searchCredential Function:

- Opens the file and searches for a credential matching the specified website.
- Displays the credential if found.

4. Compilation and Execution:

- Compile the C code using a command such as:

```
gcc password_manager.c -o password_manager
```

- Run the executable:

```
./password_manager
```

# Password Manager Documentation

5. Security Considerations:

- The program stores credentials in plain text. For real-world applications, consider using proper encryption techniques.
- The master password is hardcoded. In a production environment, it should be stored securely or derived from a secure source.

6. Conclusion:

This C password manager is a simple demonstration of file I/O and basic CRUD operations. It serves as a starting point for understanding how to build command-line applications in C that manage sensitive information. Further improvements can include better encryption, error handling, and a more robust user interface.