**Software Requirement Specification (SRS) Document**

**-Secure Password Manager**

**1. Brief Description of Project**

The Secure Password Manager is a simple yet effective console-based application developed in C that allows users to securely store and retrieve their passwords. The system encrypts stored passwords and protects access using a master password.

**2. Purpose / Goal**

The main goal of this project is to provide a **secure and efficient way** to manage passwords while helping learners understand basic encryption techniques, file handling, and user authentication in C.

**3. Usefulness / Benefit**

* **For Users:** Ensures safe storage and retrieval of sensitive credentials.
* **For Learners:** Covers file handling, encryption, and user authentication in C.
* **For Security Awareness:** Encourages good password management habits.

**4. Hardware / Software Involved**

**Hardware Requirements:**

* Standard PC/Laptop
* Minimum 2GB RAM
* 500MB disk space

**Software Requirements:**

* Programming Language: C
* Compiler: GCC / Turbo C / Code::Blocks
* Operating System: Windows/Linux/MacOS

**5. Detailed Feature List**

**User Module:**

* Set up a master password for authentication.
* Store new passwords securely (encrypted).
* Retrieve passwords after authentication.
* Update or delete stored passwords.
* View a list of stored accounts (without showing passwords).

**Security Module:**

* Encrypt passwords before storing them.
* Decrypt passwords when retrieving them.
* Protect stored passwords using a master password.

**6. Test / Demonstration Plan**

* **Unit Testing:** Validate encryption, decryption, and file storage.
* **Integration Testing:** Ensure proper user authentication and data handling.
* **Security Testing:** Verify that passwords are not stored in plaintext.
* **User Acceptance Testing:** Test with users for ease of use.

**7. Expected Interaction Interface and Sample Use Cases**

**Interaction Interface:**

* Console-based UI with menu-driven navigation.
* Secure input fields to prevent password leaks.

**Sample Use Cases:**

1. **User Sets Up Password Manager**
   * User sets a master password → System encrypts and stores it.
2. **User Stores a New Password**
   * User logs in → Enters account details and password → System encrypts and saves it.
3. **User Retrieves a Password**
   * User logs in → Selects an account → System decrypts and displays the password.
4. **User Updates or Deletes a Password**
   * User logs in → Chooses an account → Updates or deletes the password entry.