CLI Password Manager - Code and Sample Output

This document contains the full implementation code for the CLI Password Manager along with sample executions and their outputs. The tool uses AES-256 encryption, a master password, and JSON storage to securely manage credentials.

Python Code Implementation

```
import argparse
import json
import os
import base64
import getpass
import tempfile
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives.ciphers.aead import AESGCM

VAULT_FILE = "vault.json"

def derive_key(password: str, salt: bytes) -> bytes:
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt,
        iterations=480000
    )
    return kdf.derive(password.encode())

def encrypt_data(key: bytes, data: dict) -> dict:
    aesgcm = AESGCM(key)
    nonce = os.urandom(12)
    plaintext = json.dumps(data).encode()
    ciphertext = aesgcm.encrypt(nonce, plaintext, None)
    return {
        "nonce": base64.b64encode(nonce).decode(),
        "ciphertext": base64.b64encode(ciphertext).decode()
    }

def decrypt_data(key: bytes, enc: dict) -> dict:
    aesgcm = AESGCM(key)
    nonce = base64.b64decode(enc["nonce"])
    ciphertext = base64.b64decode(enc["ciphertext"])
    plaintext = aesgcm.decrypt(nonce, ciphertext, None)
    return json.loads(plaintext.decode())

def load_vault(master_password: str) -> tuple:
    if not os.path.exists(VAULT_FILE):
        salt = os.urandom(16)
        key = derive_key(master_password, salt)
        vault = {"salt": base64.b64encode(salt).decode(), "entries": {}}
        save_vault(vault)
        return vault, key
    with open(VAULT_FILE, "r") as f:
        vault = json.load(f)
    salt = base64.b64decode(vault["salt"])
    key = derive_key(master_password, salt)
    return vault, key

def save_vault(vault: dict):
    with tempfile.NamedTemporaryFile("w", delete=False) as tmp:
        json.dump(vault, tmp, indent=2)
        temp_name = tmp.name
    os.replace(temp_name, VAULT_FILE)

def add_entry(master_password: str):
    vault, key = load_vault(master_password)
    website = input("Website: ")
    username = input("Username: ")
    password = getpass.getpass("Password: ")
    entry = {"username": username, "password": password}
```

```python
    encrypted_entry = encrypt_data(key, entry)
    vault["entries"][website] = encrypted_entry
    save_vault(vault)
    print(f"[+] Entry for {website} added.")

def get_entry(master_password: str):
    vault, key = load_vault(master_password)
    website = input("Website to retrieve: ")
    if website not in vault["entries"]:
        print("[-] No such entry found.")
        return
    try:
        entry = decrypt_data(key, vault["entries"][website])
        print(f"Username: {entry['username']}")
        print(f"Password: {entry['password']}")
    except Exception:
        print("[-] Failed to decrypt. Wrong master password or corrupted data.")

def list_entries(master_password: str):
    vault, _ = load_vault(master_password)
    if not vault["entries"]:
        print("[*] Vault is empty.")
        return
    print("Stored websites:")
    for site in vault["entries"].keys():
        print(f"- {site}")

def delete_entry(master_password: str):
    vault, _ = load_vault(master_password)
    website = input("Website to delete: ")
    if website in vault["entries"]:
        del vault["entries"][website]
        save_vault(vault)
        print(f"[+] Entry for {website} deleted.")
    else:
        print("[-] No such entry found.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Secure CLI Password Manager")
    subparsers = parser.add_subparsers(dest="command")
    subparsers.add_parser("add", help="Add a new credential")
    subparsers.add_parser("get", help="Retrieve a credential")
    subparsers.add_parser("list", help="List all stored websites")
    subparsers.add_parser("delete", help="Delete a credential")
    args = parser.parse_args()
    if not args.command:
        parser.print_help()
        exit(1)
    master_password = getpass.getpass("Master Password: ")
    if args.command == "add":
        add_entry(master_password)
    elif args.command == "get":
        get_entry(master_password)
    elif args.command == "list":
        list_entries(master_password)
    elif args.command == "delete":
        delete_entry(master_password)
```

Sample Program Output

Add Entry

```
$ python password_manager.py add
Master Password: ****
Website: facebook.com
Username: john_doe
Password: ****
[+] Entry for facebook.com added.
```

Get Entry

```
$ python password_manager.py get
Master Password: ****
Website to retrieve: facebook.com
Username: john_doe
Password: mySecurePass123
```

List Entries

```
$ python password_manager.py list
Master Password: ****
Stored websites:
- facebook.com
- gmail.com
- github.com
```

Delete Entry

```
$ python password_manager.py delete
Master Password: ****
Website to delete: facebook.com
[+] Entry for facebook.com deleted.
```