Task Manager with User Authentication Writeup

Problem Statement:

In today’s world, individuals often need to keep track of various tasks in a structured way. You are tasked with building a Task Manager that allows users to manage their tasks. The system should include user authentication, meaning each user has to log in with a username and password. Once logged in, users can create, view, update, and delete their tasks. Each user’s tasks should be stored separately, and only the authenticated user can access their tasks.

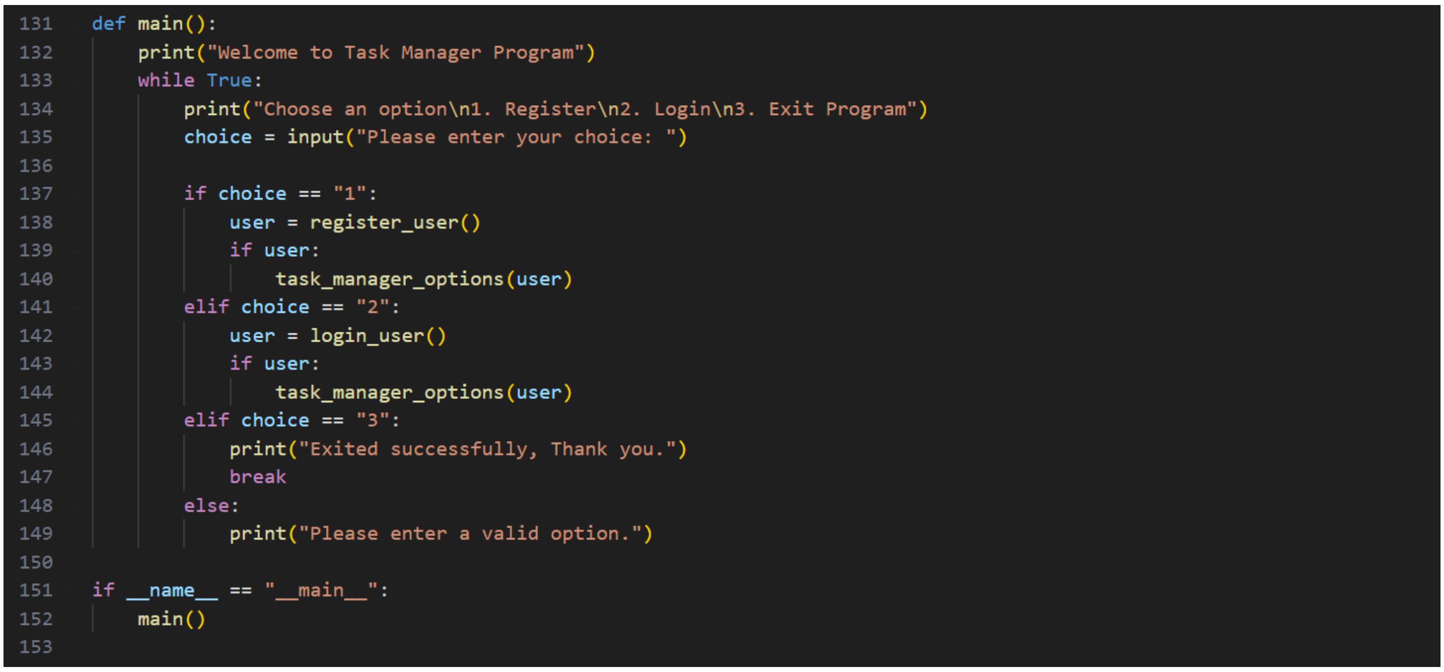
Objectives:

Design and implement a user authentication system (login and registration)

- Create a task management system that allows users to: Add, view, mark as completed, and delete tasks
- Use file handling to store user credentials and tasks persistently
- Create an interactive menu-driven interface to manage tasks

Steps:

Main Function:



```
def main():
    print("Welcome to Task Manager Program")
    while True:
        print("Choose an option\n1. Register\n2. Login\n3. Exit Program")
        choice = input("Please enter your choice: ")

        if choice == "1":
            user = register_user()
            if user:
                task_manager_options(user)
        elif choice == "2":
            user = login_user()
            if user:
                task_manager_options(user)
        elif choice == "3":
            print("Exited successfully, Thank you.")
            break
        else:
            print("Please enter a valid option.")

if __name__ == "__main__":
    main()
```

The main block allows the users to select a option to register, login or exit the program. If the choice is 1, register user function will be called and. If it is option 2, user login function is called. In either case, task manager options function is called i.e options of the task manager are displayed once the users are validated. Option 3 is to exit from the task manager program.

User Authentication:

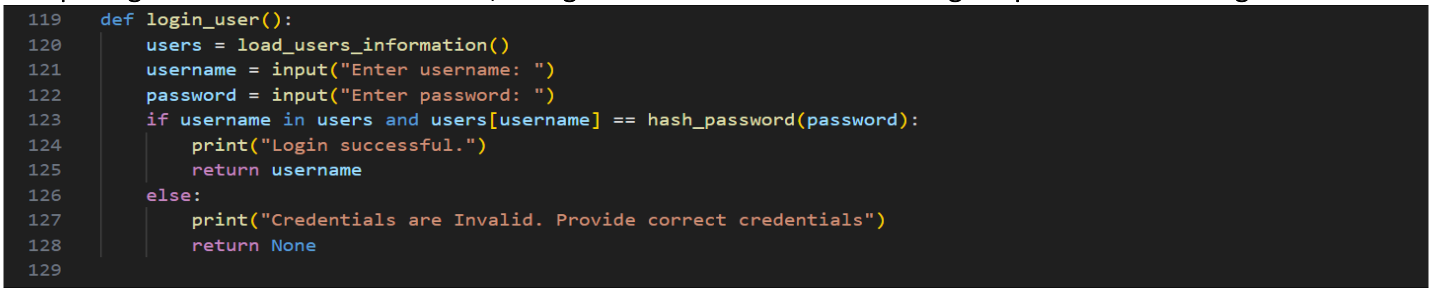
Registration: Create a function to prompt the user to enter a username and password. Ensure that the username is unique, and hash the password for security before storing it in a file

```
USERS_FILE = "users.json"
TASKS_DIR = "tasks"

if not os.path.exists(TASKS_DIR):
    os.makedirs(TASKS_DIR)

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def load_users_information():
    if not os.path.exists(USERS_FILE):
        return {}
    with open(USERS_FILE, "r") as f:
        return json.load(f)

def save_users_information(users):
    with open(USERS_FILE, "w") as f:
        json.dump(users, f)
def register_user():
    users = load_users_information()
    username = input("Enter username: ")
    if username in users:
        print("Username already exists.")
        return None
    password = input("Enter password: ")
    users[username] = hash_password(password)
    save_users_information(users)
    print("Registration successful.")
    return username
```

This function performs the following:

- The load_user_information() function is used to read all the user information from the file.
- Once the user provides the username, it will validate with the already existing usernames. If the username is already used then it prompts that the username already exists.
- If the username is available to use, the user can then enter the password. Then the password is hashed and the password is saved along with the username in the file.
- The save_user_information() function is used to write the information into the file.
- Finally after the data is saved, Registration successful is prompted.

Login: Create a function to prompt the user for their username and password, validate the credentials by comparing them with the stored data, and grant access to the task manager upon successful login

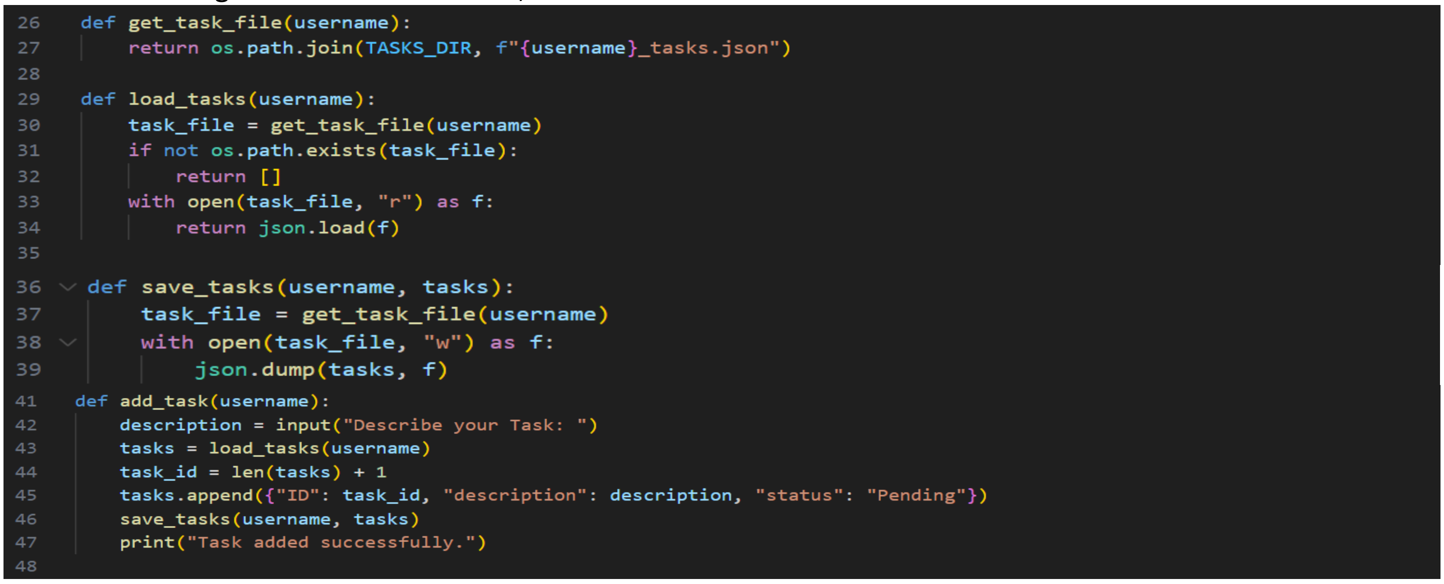


```
def login_user():
    users = load_users_information()
    username = input("Enter username: ")
    password = input("Enter password: ")
    if username in users and users[username] == hash_password(password):
        print("Login successful.")
        return username
    else:
        print("Credentials are Invalid. Provide correct credentials")
        return None
```

This function performs the following:

- The load_user_information() function is used to read all the user information from the file.
- Once the user provides the username and password, it validates if the username and hashed password combination is present in the data obtained from the file.
- If present it prompt Login successful, else it prompts that the credentials are invalid and the user should again perform the login task.

Add a Task: Create a function that prompts the user for a task description. Assign a unique task ID and set the status to Pending Store the task in a file, and confirm that the task was added

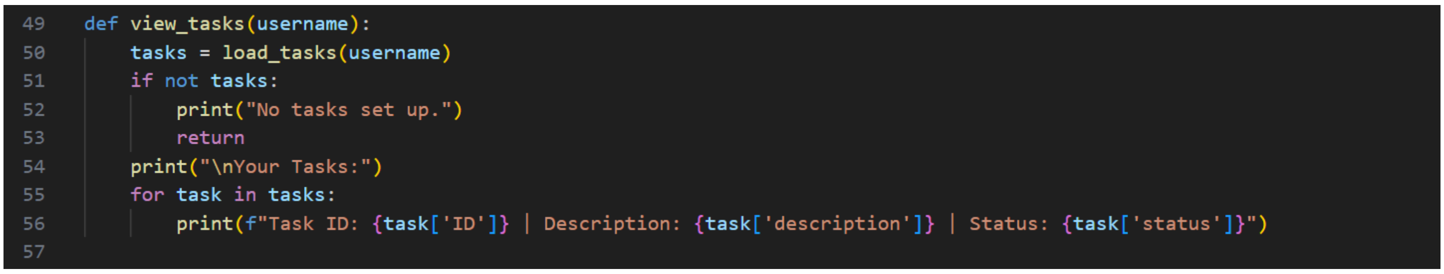


```python
def get_task_file(username):
    return os.path.join(TASKS_DIR, f"{username}_tasks.json")

def load_tasks(username):
    task_file = get_task_file(username)
    if not os.path.exists(task_file):
        return []
    with open(task_file, "r") as f:
        return json.load(f)

def save_tasks(username, tasks):
    task_file = get_task_file(username)
    with open(task_file, "w") as f:
        json.dump(tasks, f)
def add_task(username):
    description = input("Describe your Task: ")
    tasks = load_tasks(username)
    task_id = len(tasks) + 1
    tasks.append({"ID": task_id, "description": description, "status": "Pending"})
    save_tasks(username, tasks)
    print("Task added successfully.")

```

This function performs the following:

- The add_task function allows the users to add a task in the task manager.
- The user is asked to describe the task which the user wants to add.
- All the tasks assigned to the user or provided by the user are fetched from the file.
- Once the tasks are obtained, the task id is assigned for the new task that needs to be added.
- The task is then added to the tasks files which has the users provided tasks.
- Task added successfully is then prompted.

View Tasks: Create a function to retrieve and display all tasks for the logged-in user. Each task should show the task ID, description, and status (Pending or Completed)

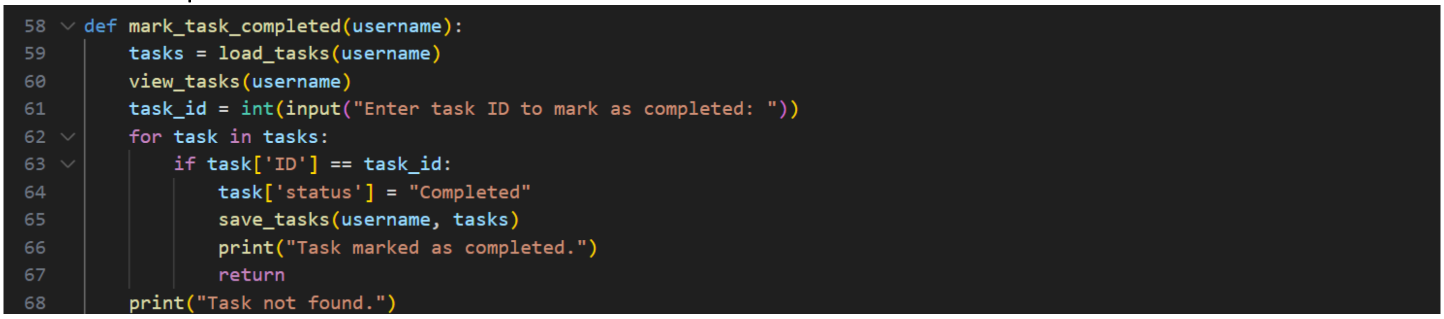


```python
def view_tasks(username):
    tasks = load_tasks(username)
    if not tasks:
        print("No tasks set up.")
        return
    print("\nYour Tasks:")
    for task in tasks:
        print(f"Task ID: {task['ID']} | Description: {task['description']} | Status: {task['status']}")

```

This function performs the following:

- The load_task function fetches all the tasks of the user.
- If the user hasn't set up any tasks yet, it prompts No task set up.
- Else, all the tasks i.e task ID, description and the status of the tasks are displayed.

Mark a Task as Completed: Create a function that allows the user to select a task by its **ID** and update its **status** to Completed

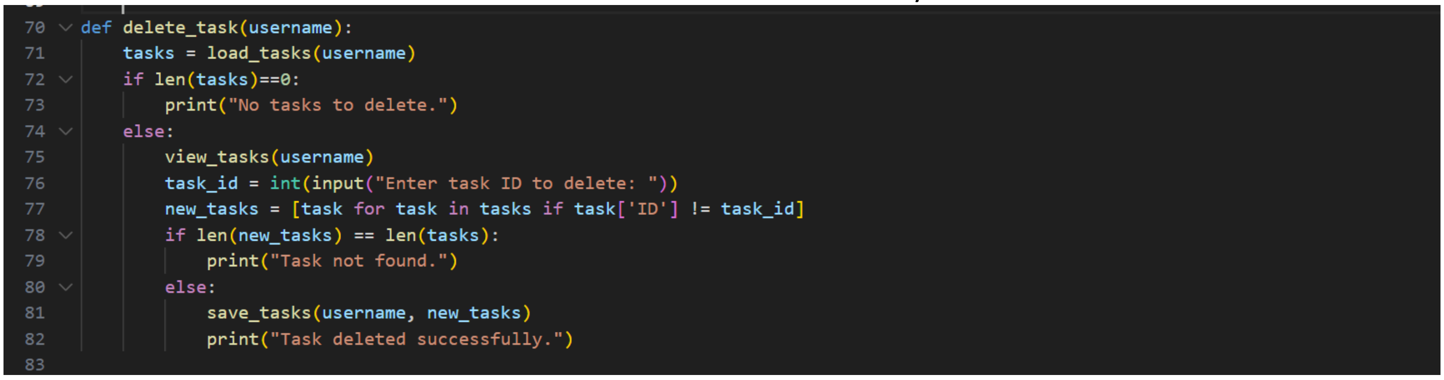


```python
def mark_task_completed(username):
    tasks = load_tasks(username)
    view_tasks(username)
    task_id = int(input("Enter task ID to mark as completed: "))
    for task in tasks:
        if task['ID'] == task_id:
            task['status'] = "Completed"
            save_tasks(username, tasks)
            print("Task marked as completed.")
            return
    print("Task not found.")
```

This function performs the following:

- The load_task function fetches all the tasks of the user.
- All the tasks which the user has set up are displayed along with the status.
- The user can then provide the task id which needs to be marked as completed.
- If the task id is present in the tasks then the status is updated to completed.
- Else, task not found is prompted.

Delete a Task: Create a function that allows the user to select a task by its **ID** and delete it from their task list



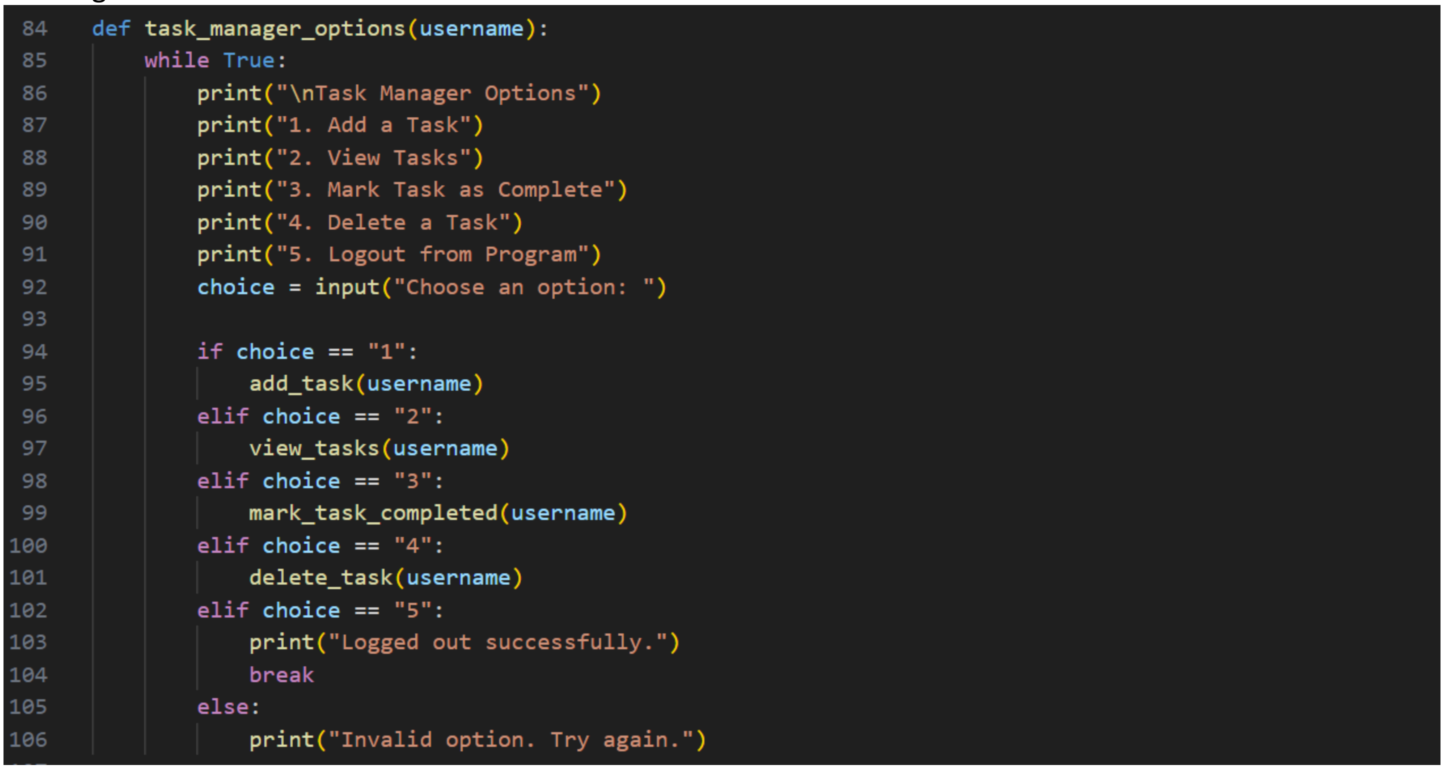
```python
def delete_task(username):
    tasks = load_tasks(username)
    if len(tasks)==0:
        print("No tasks to delete.")
    else:
        view_tasks(username)
        task_id = int(input("Enter task ID to delete: "))
        new_tasks = [task for task in tasks if task['ID'] != task_id]
        if len(new_tasks) == len(tasks):
            print("Task not found.")
        else:
            save_tasks(username, new_tasks)
            print("Task deleted successfully.")
```

This function performs the following:

- The load_task function fetches all the tasks of the user.
- If there are no tasks set up by the user, it displays No tasks to delete.
- Else, all the tasks set up by the user are displayed. Then the user can provide the task id.
- All the tasks which are not with the task id are stored in a new list and that is saved in the tasks file. Task deleted successfully is prompted.

Create an Interactive Menu:

Build a menu that allows users to choose between: Add a Task
View Tasks
Mark a Task as Completed
Delete a Task
Logout
For each option the user selects, the corresponding function is called, and loop back to the menu until the user logs out.



```python
def task_manager_options(username):
    while True:
        print("\nTask Manager Options")
        print("1. Add a Task")
        print("2. View Tasks")
        print("3. Mark Task as Complete")
        print("4. Delete a Task")
        print("5. Logout from Program")
        choice = input("Choose an option: ")

        if choice == "1":
            add_task(username)
        elif choice == "2":
            view_tasks(username)
        elif choice == "3":
            mark_task_completed(username)
        elif choice == "4":
            delete_task(username)
        elif choice == "5":
            print("Logged out successfully.")
            break
        else:
            print("Invalid option. Try again.")
```