Task Manager with User Authentication

Course-End Project Problem Statement
(Python Fundamentals)

Submitted By
Nazrul Islam Choudhury

Advanced Executive Program In Applied Generative AI

Description:

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:

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

2. Create a task management system that allows users to:

    a. Add, view, mark as completed, and delete tasks

3. Use file handling to store user credentials and tasks persistently

4. Create an interactive menu-driven interface to manage tasks

Step-by-Step Implementation:

Step 1: Set Up Project Structure

1. Create a folder for the project, e.g., Proj2
2. Inside the folder, create:
    - users.txt: To store user credentials.
    - Task-specific files will be created dynamically for each user (tasks_<username>.txt).

Step 2: User Authentication System

2.1 Registration

- Collect a username and password from the user.
- Hash the password using a hashing algorithm (e.g., SHA-256) to ensure security.
- Check if the username already exists in users.txt.
- If it doesn't exist, store username:hashed_password in users.txt.

2.2 Login

- Prompt the user for their username and password.
- Compare the username and hashed password with the data in users.txt.
- If the credentials match, grant access. Otherwise, show an error message.

Step 3: Task Management System

3.1 Add Tasks

- Create a user-specific tasks file (tasks_<username>.txt) if it doesn't exist.
- Store tasks as a JSON object containing:
    - Task ID
    - Title
    - Description
    - Status (Pending or Completed)

3.2 View Tasks

- Load tasks from the user's task file.
- Display tasks with their ID, title, description, and status.

3.3 Mark Task as Completed

- Identify the task by its ID.
- Update the task's status to Completed.

3.4 Delete Tasks

- Identify the task by its ID.
- Remove it from the task list and save the updated list.

Step 4: File Handling

4.1 User Credentials (users.txt)

- Use plain text for simplicity, storing each user as username:hashed_password.

4.2 User Tasks

- Store tasks for each user in a separate file (tasks_<username>.txt).
- Use JSON format for ease of reading, updating, and saving.

Step 5: Interactive Menu-Driven Interface

5.1 Main Menu

1. Display options for registration, login, and exit.
2. Perform actions based on user input.

5.2 Task Menu

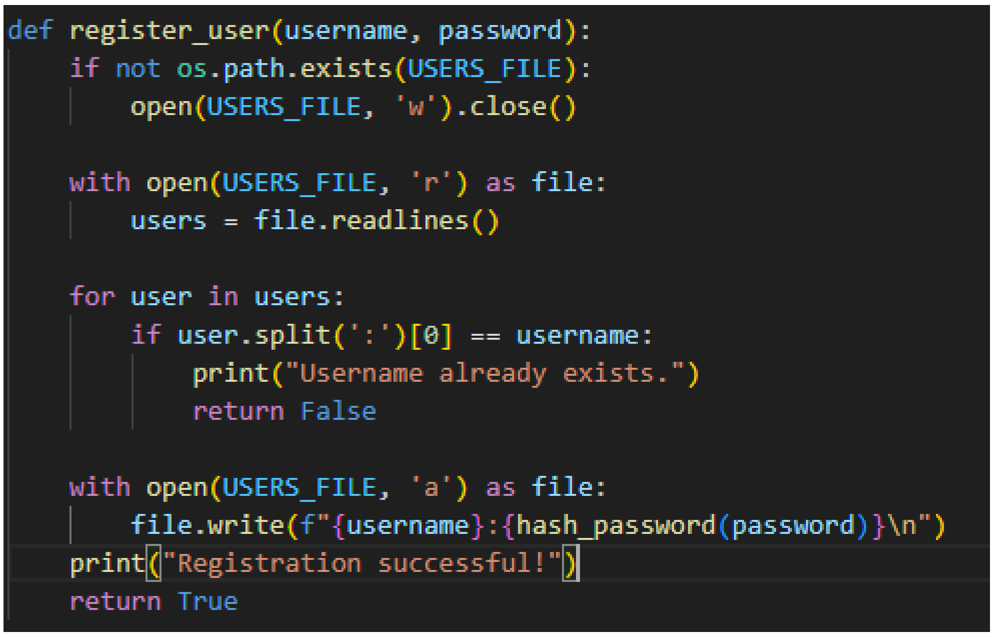
1. Once logged in, show options for:
    - Adding a task.
    - Viewing tasks.
    - Marking a task as completed.
    - Deleting a task.
    - Logging out.
2. Implement error handling for invalid choices or empty inputs.

Step 6: Future Enhancements

1. Encryption: Encrypt task files for added security.
2. Web or GUI Interface: Use a framework like Flask for better usability.
3. Search and Filter: Allow searching for tasks or filtering by status.

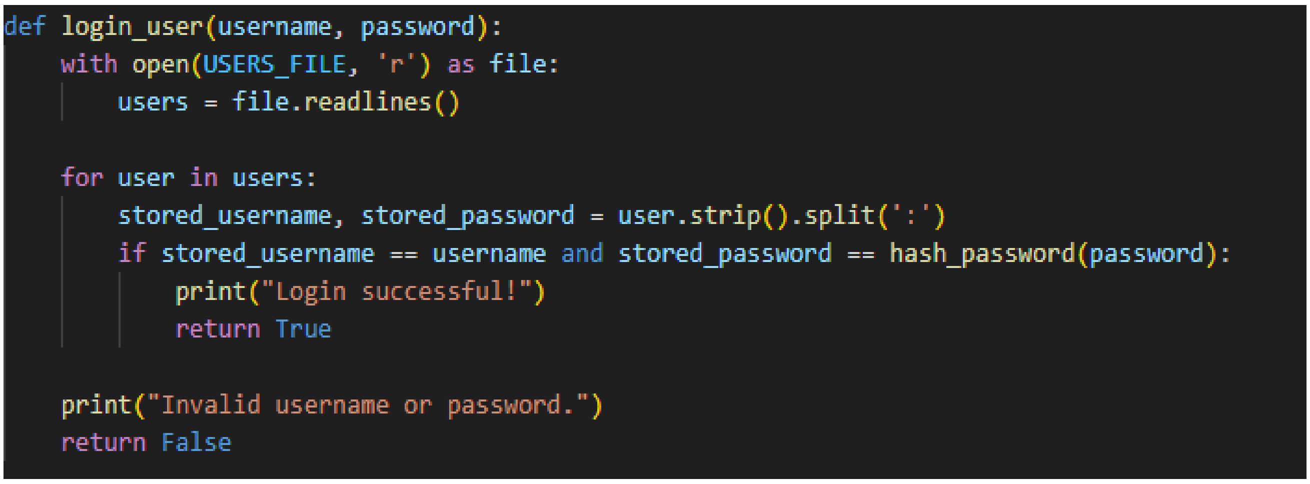
Sample Code Outline

- **Registration**



```python
def register_user(username, password):
    if not os.path.exists(USERS_FILE):
        open(USERS_FILE, 'w').close()

    with open(USERS_FILE, 'r') as file:
        users = file.readlines()

    for user in users:
        if user.split(':')[0] == username:
            print("Username already exists.")
            return False

    with open(USERS_FILE, 'a') as file:
        file.write(f"{username}:{hash_password(password)}\n")
    print("Registration successful!")
    return True
```

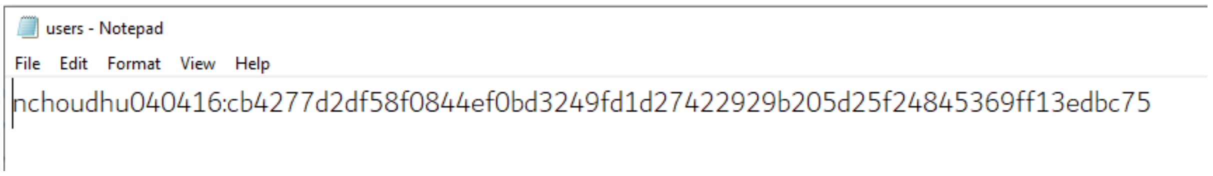
- **Login**



```python
def login_user(username, password):
    with open(USERS_FILE, 'r') as file:
        users = file.readlines()

    for user in users:
        stored_username, stored_password = user.strip().split(':')
        if stored_username == username and stored_password == hash_password(password):
            print("Login successful!")
            return True

    print("Invalid username or password.")
    return False
```

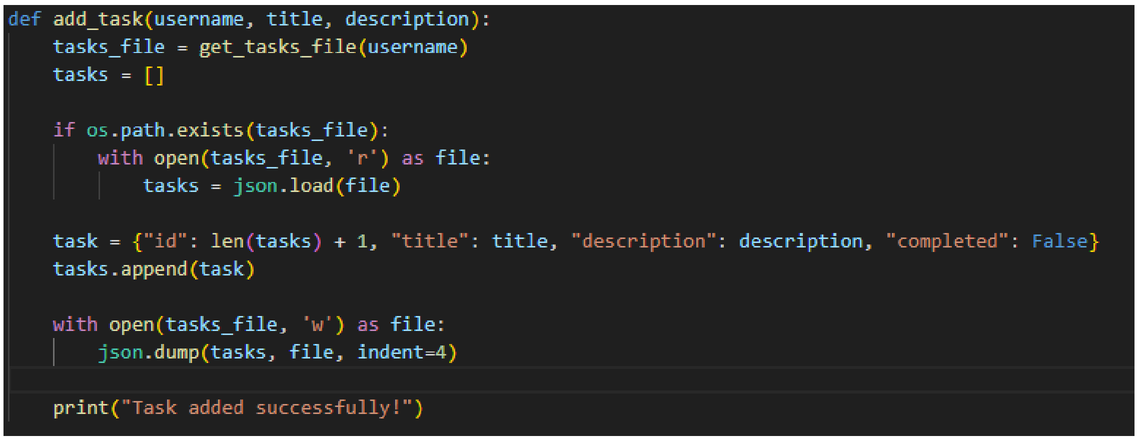
USERS_FILE = "D:\\IITM\\IITM AG\\Python Fundamentals\\Python_Project\\Proj2\\users.txt"

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



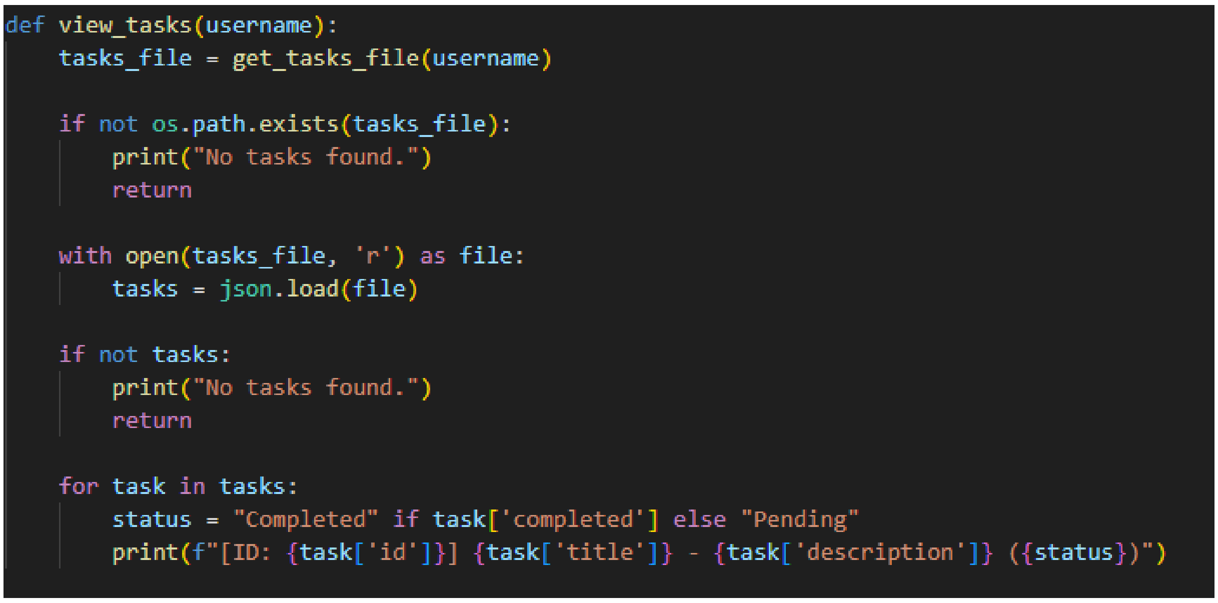
```
nchoudhu040416:cb4277d2df58f0844ef0bd3249fd1d27422929b205d25f24845369ff13edbc75
```

- **Add Task:**



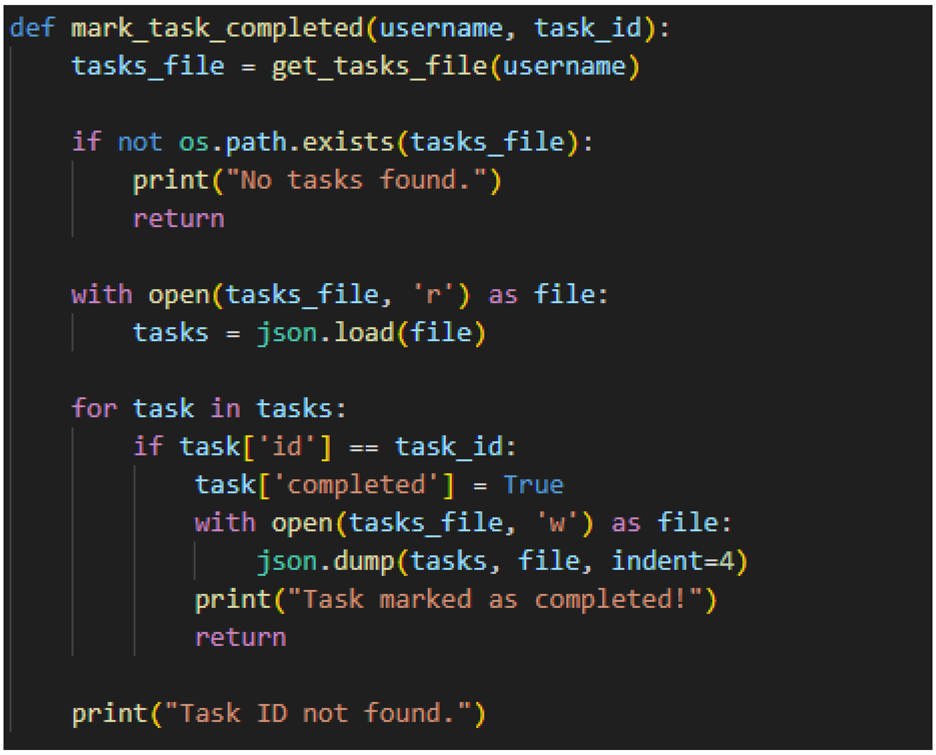
```python
def add_task(username, title, description):
    tasks_file = get_tasks_file(username)
    tasks = []

    if os.path.exists(tasks_file):
        with open(tasks_file, 'r') as file:
            tasks = json.load(file)

    task = {"id": len(tasks) + 1, "title": title, "description": description, "completed": False}
    tasks.append(task)

    with open(tasks_file, 'w') as file:
        json.dump(tasks, file, indent=4)

    print("Task added successfully!")
```

- **View Task:**



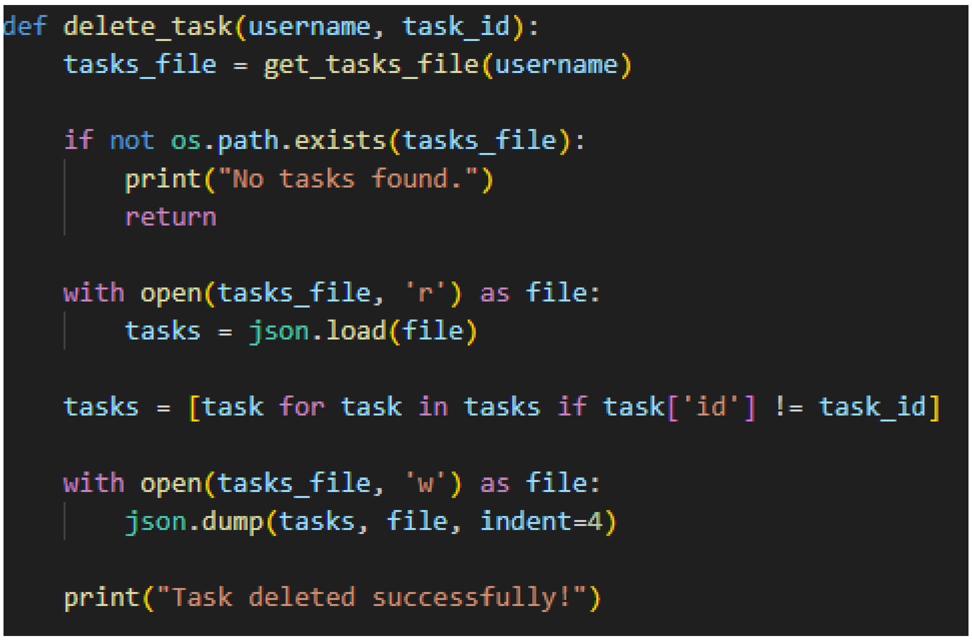
```python
def view_tasks(username):
    tasks_file = get_tasks_file(username)

    if not os.path.exists(tasks_file):
        print("No tasks found.")
        return

    with open(tasks_file, 'r') as file:
        tasks = json.load(file)

    if not tasks:
        print("No tasks found.")
        return

    for task in tasks:
        status = "Completed" if task['completed'] else "Pending"
        print(f"[ID: {task['id']}] {task['title']} - {task['description']} ({status})")
```

- **Mark Task as Completed:**



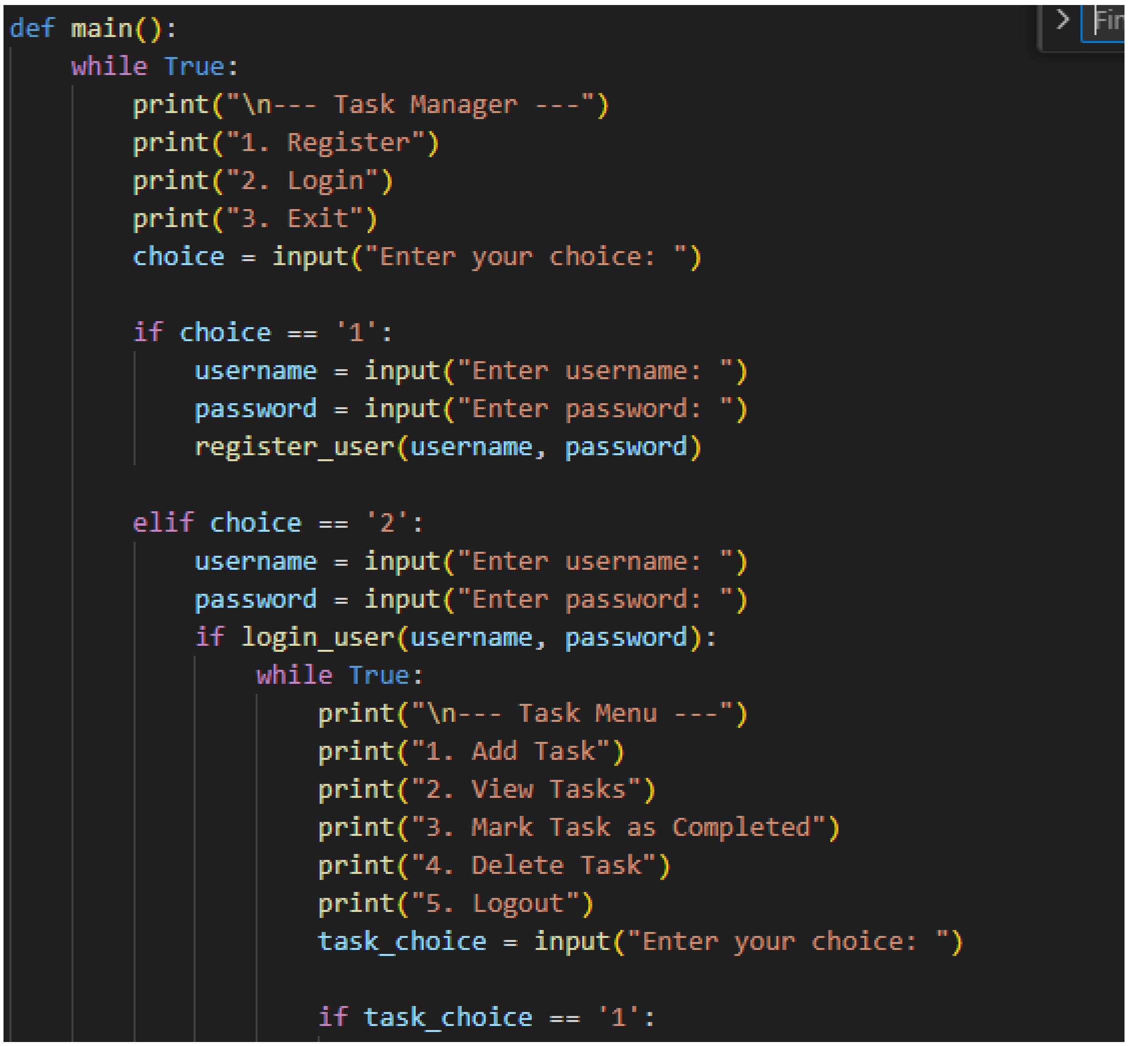
```python
def mark_task_completed(username, task_id):
    tasks_file = get_tasks_file(username)

    if not os.path.exists(tasks_file):
        print("No tasks found.")
        return

    with open(tasks_file, 'r') as file:
        tasks = json.load(file)

    for task in tasks:
        if task['id'] == task_id:
            task['completed'] = True
            with open(tasks_file, 'w') as file:
                json.dump(tasks, file, indent=4)
            print("Task marked as completed!")
            return

    print("Task ID not found.")
```

- **Delete Task:**

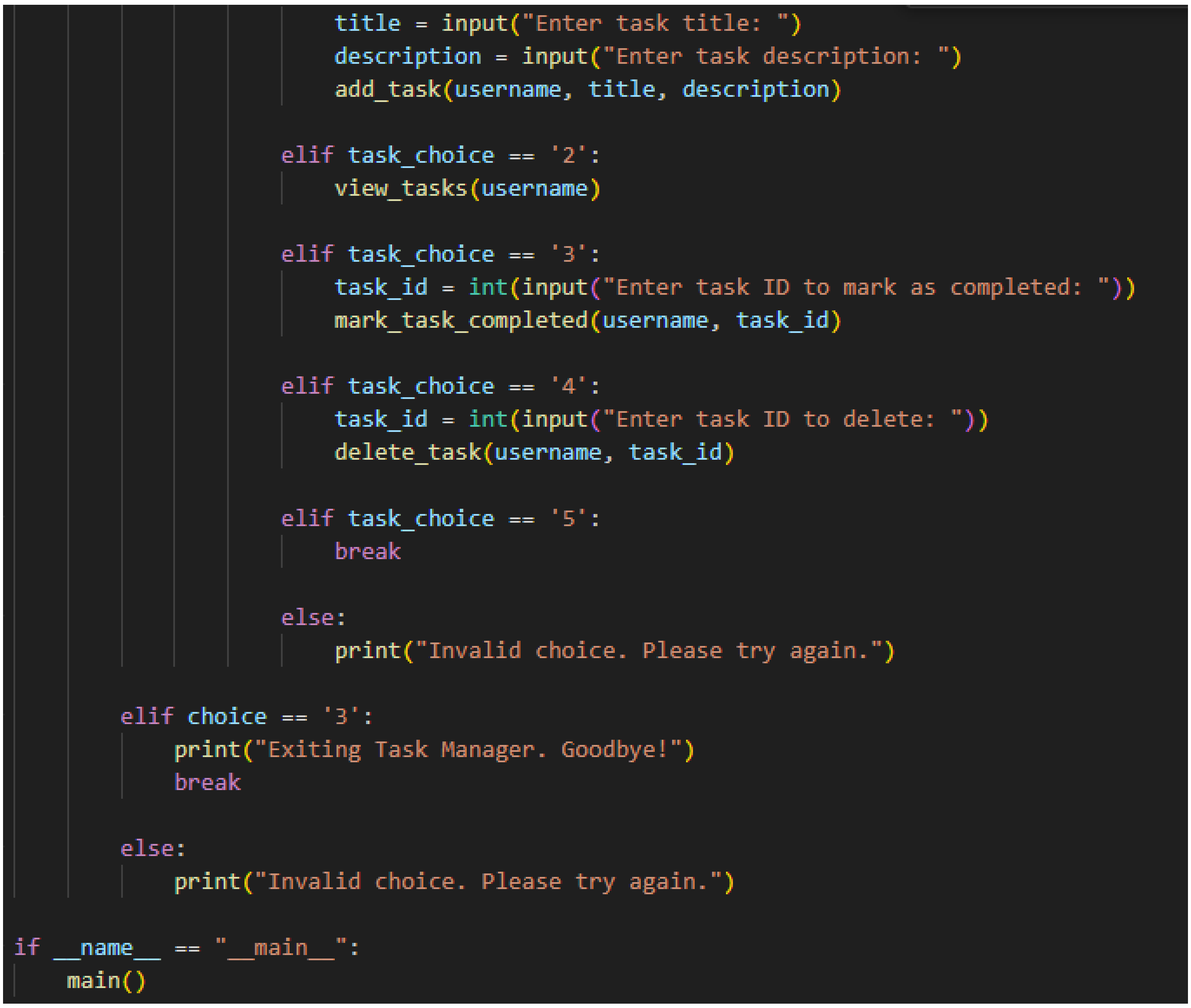


```
def delete_task(username, task_id):
    tasks_file = get_tasks_file(username)

    if not os.path.exists(tasks_file):
        print("No tasks found.")
        return

    with open(tasks_file, 'r') as file:
        tasks = json.load(file)

    tasks = [task for task in tasks if task['id'] != task_id]

    with open(tasks_file, 'w') as file:
        json.dump(tasks, file, indent=4)

    print("Task deleted successfully!")
```

- **Main Task:**



```
def main():
    while True:
        print("\n--- Task Manager ---")
        print("1. Register")
        print("2. Login")
        print("3. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            username = input("Enter username: ")
            password = input("Enter password: ")
            register_user(username, password)

        elif choice == '2':
            username = input("Enter username: ")
            password = input("Enter password: ")
            if login_user(username, password):
                while True:
                    print("\n--- Task Menu ---")
                    print("1. Add Task")
                    print("2. View Tasks")
                    print("3. Mark Task as Completed")
                    print("4. Delete Task")
                    print("5. Logout")
                    task_choice = input("Enter your choice: ")

                    if task_choice == '1':
```



```python
                        title = input("Enter task title: ")
                        description = input("Enter task description: ")
                        add_task(username, title, description)

                    elif task_choice == '2':
                        view_tasks(username)

                    elif task_choice == '3':
                        task_id = int(input("Enter task ID to mark as completed: "))
                        mark_task_completed(username, task_id)

                    elif task_choice == '4':
                        task_id = int(input("Enter task ID to delete: "))
                        delete_task(username, task_id)

                    elif task_choice == '5':
                        break

                    else:
                        print("Invalid choice. Please try again.")

        elif choice == '3':
            print("Exiting Task Manager. Goodbye!")
            break

        else:
            print("Invalid choice. Please try again.")

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

Conclusion:

The **Task Manager with User Authentication** is a practical solution for organizing tasks while ensuring user privacy and data security. This project demonstrates how to integrate fundamental programming concepts like file handling, authentication, and user-specific data management into a cohesive and functional application.