```
import socket
import os
import json
from datetime import datetime
from termcolor import colored


USER_DATA_FILE = "user_data.json"
if not os.path.exists(USER_DATA_FILE):
    with open(USER_DATA_FILE, 'w') as file:
        json.dump({}, file)

def load_user_data():
    with open(USER_DATA_FILE, 'r') as file:
        return json.load(file)

def save_user_data(data):
    with open(USER_DATA_FILE, 'w') as file:
        json.dump(data, file)

def handle_client(client_socket):
    user_data = load_user_data()


    client_socket.send("Enter username: ".encode())
    username = client_socket.recv(1024).decode()

    client_socket.send("Enter password: ".encode())
    password = client_socket.recv(1024).decode()

    if username in user_data and user_data[username]["password"] == password:
        client_socket.send(f"Welcome {username}!\n".encode())
    else:
        client_socket.send("Invalid username or password. Exiting...\n".encode())
        client_socket.close()
        return

    while True:

        client_socket.send("Choose an action: [1] Add Task [2] View Tasks [3]
Remove Task [4] Export Tasks [5] Import Tasks [6] Quit\n".encode())
        action = client_socket.recv(1024).decode()

        if action == '1':
            client_socket.send("Enter task title: ".encode())
            title = client_socket.recv(1024).decode()
            client_socket.send("Enter task category: ".encode())
            category = client_socket.recv(1024).decode()
            client_socket.send("Enter task priority (high, medium, low):
".encode())
            priority = client_socket.recv(1024).decode()
            client_socket.send("Enter task deadline (YYYY-MM-DD): ".encode())
            deadline = client_socket.recv(1024).decode()

            task = {
                "title": title,
                "category": category,
                "priority": priority,
                "deadline": deadline,
```

```
                "completed": False
            }

            if username not in user_data:
                user_data[username] = {"tasks": []}
            user_data[username]["tasks"].append(task)
            save_user_data(user_data)
            client_socket.send(f"Task '{title}' added successfully!\n".encode())

        elif action == '2':
            tasks = user_data.get(username, {}).get("tasks", [])
            response = "Your tasks:\n"
            for task in tasks:
                response += f"{task['title']} | Category: {task['category']} |
Priority: {task['priority']} | Deadline: {task['deadline']}\n"
            client_socket.send(response.encode())

        elif action == '3':
            client_socket.send("Enter task title to remove: ".encode())
            title = client_socket.recv(1024).decode()
            tasks = user_data.get(username, {}).get("tasks", [])
            tasks = [task for task in tasks if task['title'] != title]
            user_data[username]["tasks"] = tasks
            save_user_data(user_data)
            client_socket.send(f"Task '{title}' removed successfully!\n".encode())

        elif action == '4':
            client_socket.send("Enter filename to export: ".encode())
            filename = client_socket.recv(1024).decode()
            with open(filename, 'w') as f:
                json.dump(user_data[username]["tasks"], f)
            client_socket.send(f"Tasks exported to '{filename}' successfully!\
n".encode())

        elif action == '5':
            client_socket.send("Enter filename to import: ".encode())
            filename = client_socket.recv(1024).decode()
            with open(filename, 'r') as f:
                imported_tasks = json.load(f)
                user_data[username]["tasks"].extend(imported_tasks)
                save_user_data(user_data)
            client_socket.send(f"Tasks imported from '{filename}' successfully!\
n".encode())

        elif action == '6':
            client_socket.send("Goodbye!\n".encode())
            break

    client_socket.close()

def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(("127.0.0.1", 9999))
    server.listen(5)
    print("Server started. Waiting for connections...")

    while True:
        client_socket, addr = server.accept()
        print(f"Connection established with {addr}")
```

```
        handle_client(client_socket)

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