Project Report: Task Manager Pro

1. Introduction

Task Manager Pro is a simple, console-based Java application designed to help users manage their daily tasks. The goal of the project is to demonstrate how core Java programming concepts can be used to solve a real-world problem while keeping the implementation simple and easy to understand. The project fulfills all functional and non-functional requirements specified in the course, along with proper modular architecture, documentation and testing.

The system allows multiple users to register, log in, and manage their tasks. Users can create, update, and delete tasks while also generating summary reports of their work. The system stores all user and task data in plain text files using a lightweight storage mechanism.

---

2. Problem Statement

Many students and working professionals face difficulties in tracking their daily activities, leading to missed deadlines or poor time management. To address this issue, the Task Manager Pro application offers a simple way to create, manage and monitor tasks in a multi-user environment without the need for complex tools.

---

3. Objectives

- Develop a simple task management system in Java.
- Implement core programming concepts including classes, objects, file handling, conditions, loops and modular programming.
- Provide a clean console interface for interacting with the application.
- Ensure reliable data persistence using text files.
- Produce meaningful reports for each user.
- Meet the course requirements for documentation, architecture and testing.

---

4. Functional Requirements

Mandatory Modules

User Management Module

- Register new users

- Login existing users
- List existing users

Task Management Module

- Create tasks
- List tasks for the logged-in user
- Update task status (todo, doing, done)
- Delete tasks

Reporting Module

- Display task summary (counts of todo, doing, done)
- Export a detailed text report

Input and Output Structure

Inputs: Username, full name, task details (title, priority), task ID, status updates.

Outputs: Menu-driven results printed in the console, including confirmation messages, summary reports and error messages.

---

5. Non-Functional Requirements

1. **Usability:** Simple, menu-based navigation suitable for beginners.
2. **Reliability:** Basic validation ensures that invalid inputs do not crash the program.
3. **Maintainability:** Code is divided into packages such as `model`, `service`, `storage`, `utils` and `ui`.
4. **Resource Efficiency:** Uses only plain text files; no heavy libraries or databases.
5. **Error Handling Strategy:** Avoids exceptions where possible; invalid inputs simply return feedback to the users.

---

6. System Architecture

Architectural Pattern

The project follows a **three-layer architecture**:

- **UI Layer (ConsoleUI):** Handles interactions with the user.
- **Service Layer (UserService, TaskService):** Contains business logic.
- **Storage Layer (Storage):** Manages file read/write operations.

Architecture Diagram (Description)

```
+----------------------+
|      ConsoleUI       |
+----------+-----------+
           |
           v
+----------------------+        +--------------------+
|     UserService      |<----->|     Storage        |
+----------------------+        +--------------------+
|     TaskService      |<----->| Reads/Writes Files |
+----------------------+        +--------------------+
```

---

7. Process Flow / Workflow

High-level workflow:

1. User opens application.
2. User registers or logs in.
3. After login, user accesses task menu.
4. User performs operations:
   a) Create
   b) View
   c) Update
   d) Delete
5. Generate report
6. User logs out.
7. System returns to main menu.
8. User exits application.

---

8. UML Diagrams

8.1 Use Case Diagram (Text Description)

Actor: User
Use Cases: Register, Login, Create Task, List Tasks, Update Task, Delete Task, Generate Report, Export Report

8.2 Class Diagram (Text Description)

Classes involved:

- `User`
- `Task`
- `UserService`
- `TaskService`
- `Storage`

- `ConsoleUI`
- `ReportGenerator`
- `Validator`

8.3 Sequence Diagram (Login + Create Task)

```
User -> ConsoleUI : enter username
ConsoleUI -> UserService : login(username)
UserService -> Storage : loadUsers()
Storage -> UserService : return users
UserService -> ConsoleUI : return user object
ConsoleUI -> TaskService : createTask(owner,title,priority)
TaskService -> Storage : saveTasks()
```

---

9. Storage Design

Users File (users.txt)

Format:

```
username,fullname
```

Tasks File (tasks.txt)

Format:

```
id,owner,title,status,priority
```

This ensures lightweight persistence without databases.

---

10. Implementation Summary

The system is implemented fully in Java using separate packages for clarity. Key concepts used:

- Classes and objects
- File handling (`FileReader`, `FileWriter`, `PrintWriter`)
- Collections (`ArrayList`)
- Exception handling
- Modular design with packages

All code is included in the project directory under `src/com/taskmanager/`.

---

11. Testing

A simple manual test (`TestTaskService.java`) verifies core functionality:

- Task creation
- Listing tasks
- Checking task count

Users are encouraged to test edge cases manually:

- Invalid priority input
- Attempting to delete another user’s task
- Exporting report to different filenames

---

12. Conclusion

Task Manager Pro is a minimal yet complete Java project created to cover real-world problem solving while satisfying the academic requirements of modularity, documentation, diagrams, testing and structured development. The system is simple, easy to run and understand, and showcases core Java programming concepts effectively.

It serves as a practical demonstration of how a beginner-level Java project can be scaled into a structured, well-documented application.