Project Title
Task Management System

Programming In Java (CSE2006)

Name : J BalaVarun
Reg n.o : 22BAC10044

1. Introduction:

Managing daily tasks efficiently plays an important role in personal productivity. However, many people struggle to keep track of deadlines and pending work. This leads to missed submissions, poor time management, and incomplete tasks.This project aims to develop a console-based To-Do Task Manager in Java that enables users to create tasks, assign deadlines, track status (PENDING/DONE), and delete tasks when necessary. The system is simple, lightweight, and uses object-oriented principles, making it suitable for learning Java in a structured way.

It uses ASCII-only output to ensure full compatibility with Windows Command Prompt and PowerShell, avoiding issues with emojis or Unicode symbols.

2. Problem Statement:

People often forget important tasks and deadlines because they do not have a simple method to track their activities. Traditional notes or memory-based reminders are inefficient and unreliable.To create a simple and efficient task management system that allows users to record tasks, set deadlines, track pending work, and update progress using a user-friendly Java console application.

3. Functional Requirements

1. Add Task:

Allows users to enter a task title, description, and deadline. The task is assigned a unique ID and stored in the system.

2. View All Tasks:

Displays all tasks in a clean ASCII format, sorted by deadline for better task tracking. Helps users get a complete overview of upcoming work.

3. View Pending Tasks:

Shows only tasks that are not yet completed. This helps users focus on unfinished work without distractions.

4. Mark Task as Done:

Updates the selected task's status from PENDING to DONE. Keeps the task list organized by completion.

5. Delete Task:

Removes a task permanently using its ID. Useful for removing outdated or incorrect entries.

6. Save Tasks to File:

Stores all tasks inside a file (tasks.txt) when the program exits. Ensures the task list is preserved for the next session.

4. Non-functional Requirements

1. Usability

The system provides a simple text-based menu for easy navigation. Users of any technical level can operate it effortlessly.

2. Performance

Responds instantly to user commands and handles tasks efficiently. Performs well even as the number of tasks grows.

3. Portability

Runs on any machine with Java installed, regardless of operating system. No additional setup or dependencies are required.

4. Reliability

Ensures tasks are saved and loaded correctly every time. Prevents data loss through consistent file storage.

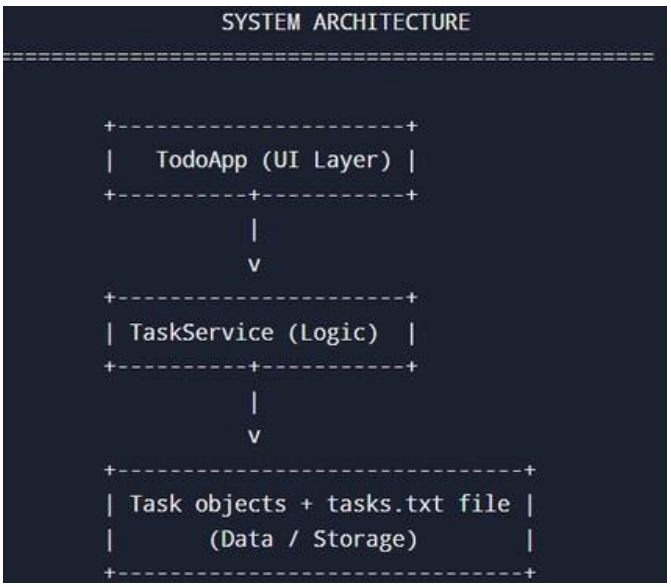
5. Maintainability

Uses a clean multi-file, multi-package architecture for easy updates. Clear separation of logic, UI, and data improves future modifications.

6. Compatibility

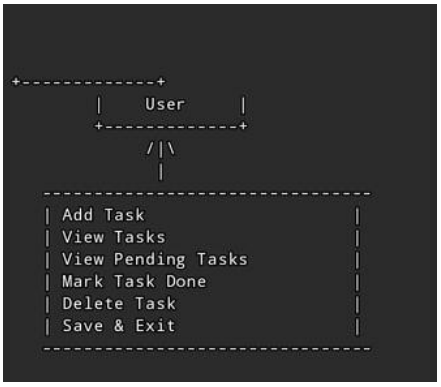
ASCII-only interface avoids Unicode issues in Windows terminals. Works smoothly on CMD, PowerShell, and basic editors.

5. System Architecture

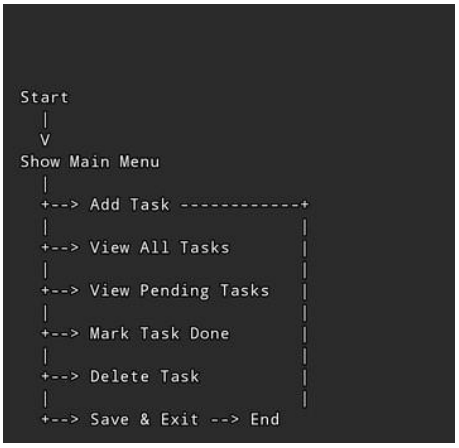


6. Design Diagram

a) Use Case Diagram:



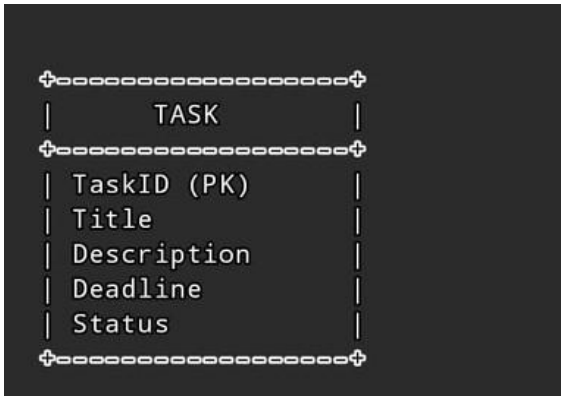
b) Workflow Diagram:



c) Sequence Diagram:

d) Class Diagram:

e) ER Diagram:



7. Design Decisions and Rationale

- **ASCII output instead of emojis**

To avoid Unicode issues on PowerShell/CMD.

- **ArrayList for task storage**

Because tasks are sequential and dynamic.

- **File storage instead of database**
-

Lightweight, simple, easier for beginners.

- **Three-package structure**
-

Better code organization:
objects → Data model
manager → Logic
app → UI

- **LocalDate for deadline**

Type-safe date handling.

8. Implementation Details

Language: Java (OOP)

Packages:

- objects.Task
- manager.TaskService
- app.TodoApp

Sorting by: deadline using Comparator

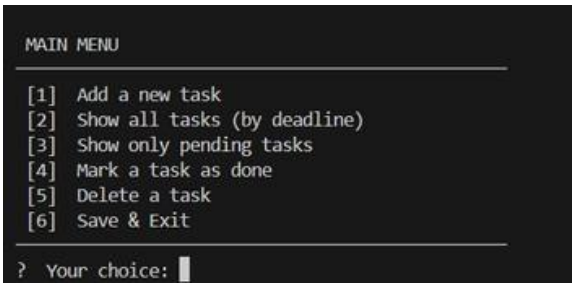
Storage: Plain text file tasks.txt

Error Handling:

- Invalid date format
- Invalid task ID
- Empty task list

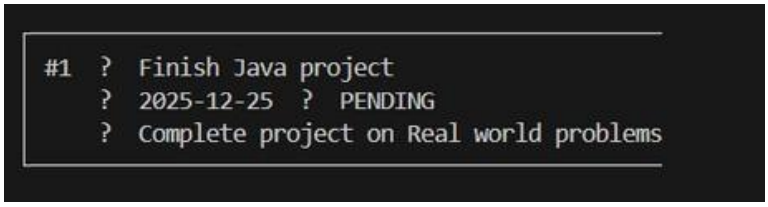
9. Screenshots

1. Main Menu:

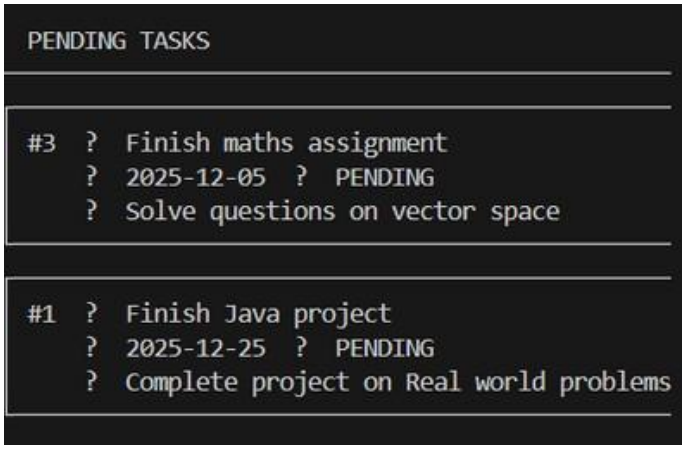


2. Add Task:

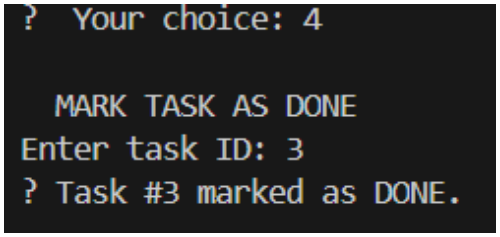
3. View All Tasks:



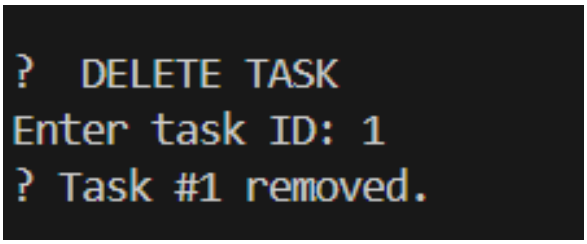
4. View Pending Tasks:



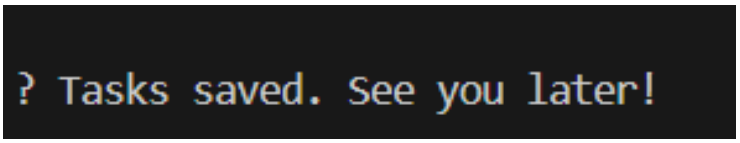
5. Mark Task as Done:



6. Delete Task:



7. Save Tasks to File:



10. Testing Approach

Unit Testing:

- Adding tasks
- Marking done
- Removing tasks

Input Validation Testing:

- Invalid date
- Empty title
- Invalid Task ID

File Handling Testing:

- Data loads correctly
- Data saves correctly

Sorting Tests:

- Tasks arranged by deadline

11. Challenges Faced

- **Unicode emojis not supported in PowerShell**

* Solved by switching to ASCII**.**

- **Maintaining persistence**

* Implemented file-based storage.

- **Handling date parsing errors.**

* Used try-catch with LocalDate.parse.

- **Ensuring multi-file package structure works**

* Used correct compile command with -d out.

12. Learnings & Key Takeaways

- Importance of OOP design
- Multi-file Java project structure
- File handling using Java I/O
- Exception handling
- Designing ASCII-friendly UI
- Sorting objects using Comparator
- Importance of modular architecture

13. Future Enhancements

- Add task priorities (High/Medium/Low)
- Add overdue detection (Deadline < Today)
- Search task by title
- Edit existing tasks
- Export to CSV format
- Add colored output using ANSI codes
- Turn into GUI app using JavaFX

14. References

- Oracle Java Documentation
- Java I/O Tutorials
- Java Collections Framework Documentation
- College Lecture Notes
- StackOverflow