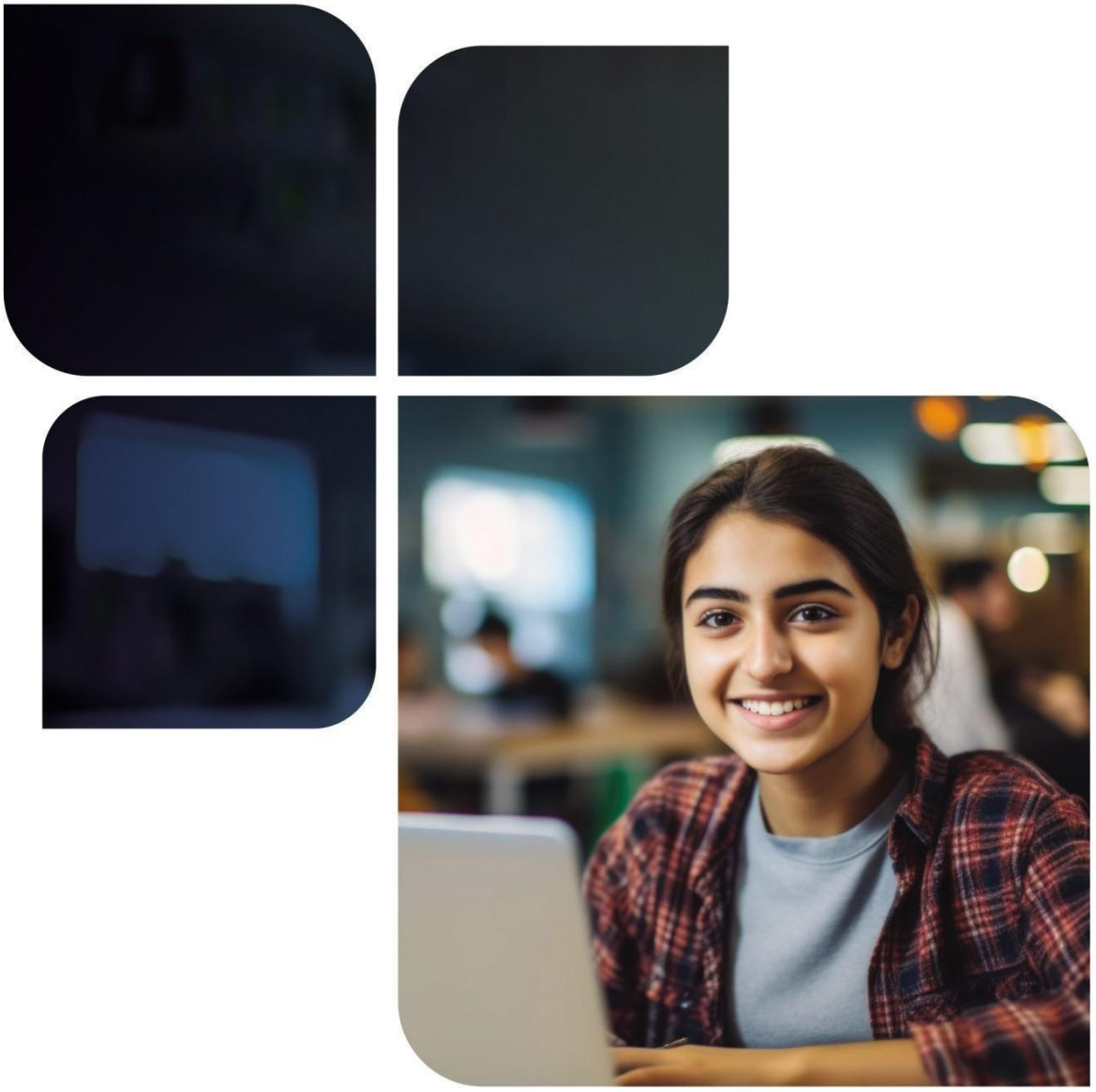
GLOBAL QUEST
TECHNOLOGIES

An ISO 9001 & ISO 21001
Certified Organizaton



The Quest
for your Dream Job
Ends Here!!

www.gqtech.in

SMART TO-DO LIST WITH DEADLINES

Name: Venna Chandana

Student ID: GQT-S0 1404

Mentor: Mr. BHEEMESH RAGHUPATHY

Institute: GLOBAL QUEST TECHNOLOGIES

Duration: June 2025

SMART TO-DO LIST WITH DEADLINES

ABSTRACT

The "**Smart TO-DO List with Deadlines**" is a Java-based console application aimed at helping users manage their daily tasks effectively. It allows users to add, track, and complete tasks by assigning deadlines and priority levels. This project emphasizes real-world task management with a smart console interface, highlighting Java concepts such as object-oriented programming, data validation, file handling, and input management.

The program enables users to add tasks with a title, description, deadline, and priority (High/Medium/Low). Tasks are automatically sorted by deadline and include status indicators such as Pending, Completed, or Overdue. Input validation ensures users do not enter invalid dates or characters. Completed tasks can be marked and all task data can be saved to a text file.

This project provides a practical example of using core Java to build productivity tools. It demonstrates structured programming, user interaction, exception handling, and persistent storage—all within a beginner-friendly console application.

ACKNOWLEDGEMENT

I sincerely thank **GLOBAL QUEST TECHNOLOGIES** for giving me the opportunity to work on this project titled "**Smart TO-DO List with Deadlines**". I express my deep gratitude to Mr. G. R. Narendra Reddy, Founder of Global Quest Technologies, for his constant support and encouragement.

My special thanks go to my mentor, **Mr. BHEEMESH RAGHUPATHY**, whose guidance and insights helped me implement key concepts of Java, from class structures to file handling. His mentorship was instrumental in enhancing my understanding of structured programming.

I also acknowledge my peers and the training environment for fostering collaboration, creativity, and focus throughout the project. This experience has improved my confidence in solving real-world problems using Java.

TABLE OF CONTENTS

CHAPTER: 1

INTRODUCTION

1. INTRODUCTION

"Smart TO-DO List with Deadlines" is a console-based application developed using Core Java. It is designed to help users manage and prioritize their daily tasks efficiently. The application allows users to create tasks by entering a title, description, deadline, and priority level (High, Medium, or Low). These tasks are then automatically sorted by deadline to help users focus on what matters most.

The application features a user-friendly, menu-driven interface through which tasks can be added, viewed, marked as completed, or saved to a file. Statuses such as Pending, Completed, or Overdue are displayed clearly, with ANSI color codes improving visibility and user experience.

At the core of the application lies object-oriented design. A custom Task class encapsulates all task details, while the SmartToDoList class manages user interaction and program logic. Tasks are stored in a dynamic ArrayList, and Java's LocalDate class ensures proper deadline formatting and validation.
To support data persistence, the program uses Java's FileWriter to save task summaries to a file (todo_list.txt). Robust input validation and error handling are included to ensure that the application is reliable and user-friendly.

Overall, this project offers a practical example of how foundational Java concepts—like classes, objects, conditionals, collections, file handling, and input validation—can be applied to solve real-world problems in a structured and meaningful way. serves as a practical demonstration of core Java concepts such as classes, objects, lists, conditionals, loops, user input handling, exception management, and file I/O. It offers a real-world scenario that reinforces programming fundamentals while providing a useful productivity tool for users to manage their daily activities effectively. a real-world implementation of foundational Java programming concepts such as input validation, collections, file I/O, and conditionals.

1.2 Problem Statement

In today's productivity-focused environment, users depend on task management tools to stay organized. However, many available solutions are either overly complex, cloud-dependent, or unsuitable for beginners aiming to build their programming skills. There is a distinct need for a lightweight, console-based application that provides essential task management features without requiring GUI libraries or internet access.

Beginner programmers also face challenges when trying to implement real-world logic using core Java concepts. Many basic applications fail to incorporate practical elements like deadlines, task prioritization, and data persistence. Additionally, these systems often lack intuitive user feedback, proper input validation, and structured status tracking for tasks.

This project addresses those limitations by offering a structured Java console application where users can interact with a menu to create, update, and monitor tasks. Each task includes a title, description, deadline, and priority level. Tasks are automatically sorted by deadline and categorized by status—Pending, Completed, or Overdue. The application validates user inputs and prevents incorrect formats, such as invalid dates or task numbers.

Furthermore, this system includes functionality for marking tasks as completed and saving all task information to a file for future reference. It provides a complete cycle of task management, from creation to completion and archival.

The "Smart TO-DO List with Deadlines" thus fills a critical gap for both end users and learners. It demonstrates how core programming techniques can solve real-world problems in a simple, effective, and educational way, making it a perfect project for academic purposes or personal use. with a smart, console-based TO-DO list that provides essential features like deadlines, priorities, status tracking, and file-saving.

CHAPTER: 2

OBJECTIVE

2. OBJECTIVE

The primary objective of the **"Smart TO-DO List with Deadlines"** project is to develop a functional, user-friendly console application using Core Java that helps users manage and prioritize their daily activities. The application is designed to serve as an educational tool that bridges theoretical Java programming concepts with a real-world use case.

The key objective is to allow users to create structured task entries by inputting a title, description, deadline, and priority level. Tasks are stored dynamically using an ArrayList, ensuring flexibility and scalability. Deadlines are validated using Java's built-in LocalDate class, and tasks are automatically sorted based on their due dates, ensuring the most urgent items appear first.

The system also enables users to track the progress of each task by marking them as Completed. It assigns a status label—Pending, Completed, or Overdue—based on the current date and user interaction. This enhances task visibility and encourages timely completion.

In addition, the application emphasizes data persistence by writing task information to a text file using FileWriter, providing a simple yet effective way to maintain a log of user entries.

Another important goal is to enforce clean coding practices such as modular class design, input validation, exception handling, and clear console output using ANSI color codes.

Ultimately, this project aims to demonstrate how core Java concepts can be applied in building a meaningful, structured, and interactive application that reflects real-world software behavior.

+

2.1 Scope of the Project

The "Smart TO-DO List with Deadlines" project has been carefully scoped to focus on the implementation of an effective and practical task management system using core Java. The scope includes all essential features that support task creation, tracking, and persistence, while intentionally excluding more advanced or GUI-based capabilities to maintain simplicity and educational value.

In Scope:

- The application operates entirely through a console-based interface, emphasizing command-line interaction.
- It enables users to add tasks by specifying a title, description, deadline, and priority level (High, Medium, or Low).
- Input validation is performed to ensure correct formatting of dates and valid data entry.
- The program tracks each task's status as Pending, Completed, or Overdue based on the deadline and user interaction.
- Tasks are stored dynamically using ArrayList and are sorted by deadline.
- Users can mark tasks as completed, and the updated status is reflected in the system.
- A task summary is saved to a text file using FileWriter, demonstrating data persistence.
- ANSI color codes enhance the user interface by visually distinguishing task statuses.

Out of Scope:

- No graphical user interface (GUI) elements like buttons or windows are used.
- Multi-user or profile-based task management is not implemented.
- Features such as real-time reminders, notifications, or scheduling integrations are excluded.
- The application does not use a database; it relies on simple text file storage.

This scope ensures a focused and manageable development process that reinforces Java programming concepts while offering a usable, structured task-tracking tool for users.

CHAPTER: 3

TOOLS AND TECHNOLOGIES USED

3. TOOLS & TECHNOLOGIES USED:

The "Smart TO-DO List with Deadlines" project utilizes essential tools and technologies from the Java programming ecosystem. Designed to demonstrate the power and flexibility of Core Java, the application emphasizes simplicity, reliability, and educational value through its toolset.

Programming Language: Java (Core Java) Java was selected for its object-oriented features, platform independence, and extensive support libraries. It enables structured development and reinforces best practices in programming.

IDE (Integrated Development Environment): IntelliJ IDEA / Eclipse / NetBeans Any of these Java IDEs can be used to develop the project. They provide features like auto-completion, debugging tools, and error detection, making the development process smoother and more productive.

Java Development Kit (JDK): JDK 8 or higher the project is compatible with Java SE 8 and later versions. It avoids deprecated features and utilizes modern APIs such as java.time for date handling.

Libraries and Classes Used:

- java.util.Scanner: For capturing user input
- java.util.ArrayList: For dynamic task storage
- java.time.LocalDate: For date validation and comparison
- java.io.FileWriter: For saving tasks to a text file

This combination of tools and technologies provides a strong foundation for understanding structured and object-oriented programming in Java.

3.1 Core Java Technologies

The **"Smart TO-DO List with Deadlines"** project integrates several essential Core Java technologies to provide a strong foundation for console-based task management. These technologies were selected for their relevance in teaching fundamental Java concepts while also allowing the application to function efficiently without third-party dependencies.

1. java.util Package:
- **Scanner:** Used for reading user input from the console in a simple and flexible way.
- **ArrayList:** Provides dynamic memory allocation to store and manage an expandable list of tasks.
- **Comparator:** Enables sorting of tasks based on custom logic (e.g., sorting by deadline).

2. java.time Package:
- **LocalDate:** Offers built-in support for date management, allowing the application to validate task deadlines, check for overdue statuses, and format dates effectively.
- **DateTimeFormatter:** Supports parsing and formatting dates in the yyyy-MM-dd pattern.

3. java.io Package:
- **FileWriter:** Allows the application to store task data persistently by writing formatted content to a local text file.

4. Object-Oriented Features:
- The project uses Java classes, objects, constructors, and methods to model real-world entities like tasks.
- Concepts such as encapsulation, abstraction, and modularity are applied using custom classes like Task and SmartToDoList.

5. Exception Handling:
- Try-catch blocks and validation logic ensure smooth user experience and prevent runtime crashes due to invalid inputs.

By utilizing these Core Java technologies, the project not only delivers a functional solution but also serves as a strong educational tool for learners to understand and apply Java programming in practical scenarios.

CHAPTER: 4

SYSTEM DESIGN

4. SYSTEM DESIGN:

The system design of the **"Smart TO-DO List with Deadlines"** project is structured around modular programming principles and object-oriented design. The application consists of two primary classes: the Task class and the SmartToDoList class. These classes work together to represent task data and manage the flow of the program, respectively.

The **Task class** encapsulates all relevant details of a task, including the title, description, deadline, priority level, and completion status. It includes methods to return a formatted task summary, including color-coded output to visually indicate task status (Pending, Completed, or Overdue).

The **SmartToDoList class** handles all user interactions through a menu-driven interface. This includes adding tasks, viewing the task list, marking tasks as completed, and saving tasks to a file. User input is validated using structured logic and exception handling to avoid crashes and ensure data correctness.

The design is divided into logical layers:

- **Input Layer:** Uses the Scanner class to capture user input.
- **Logic Layer:** Validates deadlines with LocalDate, sorts tasks using Comparator, and updates status dynamically.
- **Output Layer:** Displays tasks with color formatting and saves them using FileWriter.
-

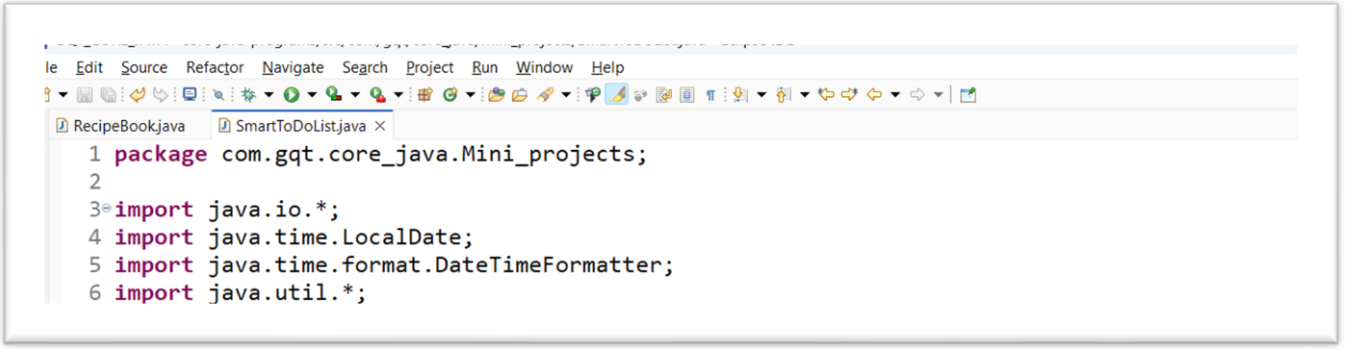
Error handling ensures a smooth user experience by catching invalid input and preventing unexpected behavior. The use of Java's collection framework (ArrayList) enables dynamic management of task data. The modular design and code reusability make the system easy to understand, maintain, and extend in the future.

CHAPTER: 5

CODE EXPLANATION

5. CODE EXPLANATION:

1. Package Declaration and Import Statements:

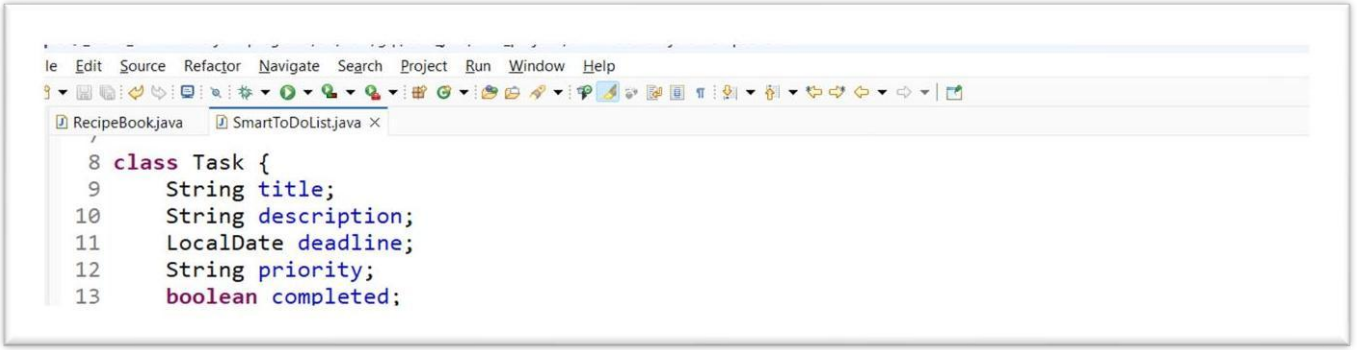


This defines the package in which your Java class resides. It helps organize and manage files in larger projects.

These imports bring in Java utility classes:

- Scanner, ArrayList, Comparator: for input and collections
- LocalDate, DateTimeFormatter: for date handling
- FileWriter: for file operations.

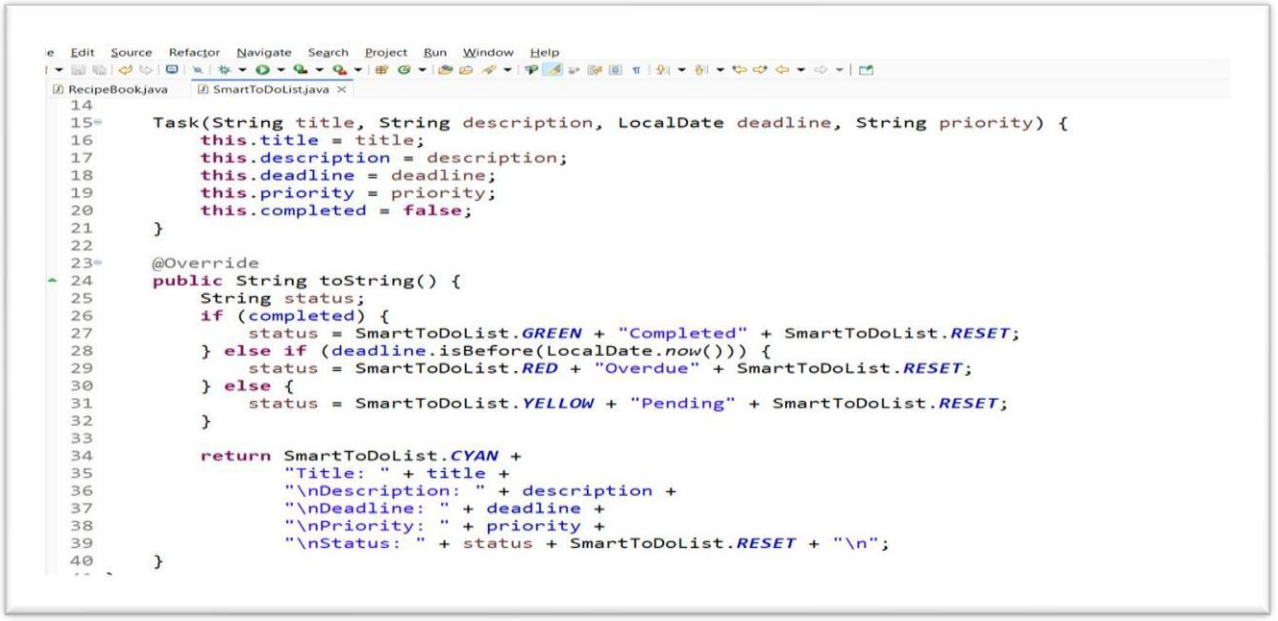
2. Task Class



-- This is a blueprint to represent a single task.

-- Each task stores a **title**, **description**, **deadline**, **priority**, and **completion status**.

2.1Constructor:



```
    Task(String title, String description, LocalDate deadline, String priority) {
        this.title = title;
        this.description = description;
        this.deadline = deadline;
        this.priority = priority;
        this.completed = false;
    }

    @Override
    public String toString() {
        String status;
        if (completed) {
            status = SmartToDoList.GREEN + "Completed" + SmartToDoList.RESET;
        } else if (deadline.isBefore(LocalDate.now())) {
            status = SmartToDoList.RED + "Overdue" + SmartToDoList.RESET;
        } else {
            status = SmartToDoList.YELLOW + "Pending" + SmartToDoList.RESET;
        }

        return SmartToDoList.CYAN +
                "Title: " + title +
                "\nDescription: " + description +
                "\nDeadline: " + deadline +
                "\nPriority: " + priority +
                "\nStatus: " + status + SmartToDoList.RESET + "\n";
    }
```

-Initializes the task with values entered by the user. By default, completed = false.

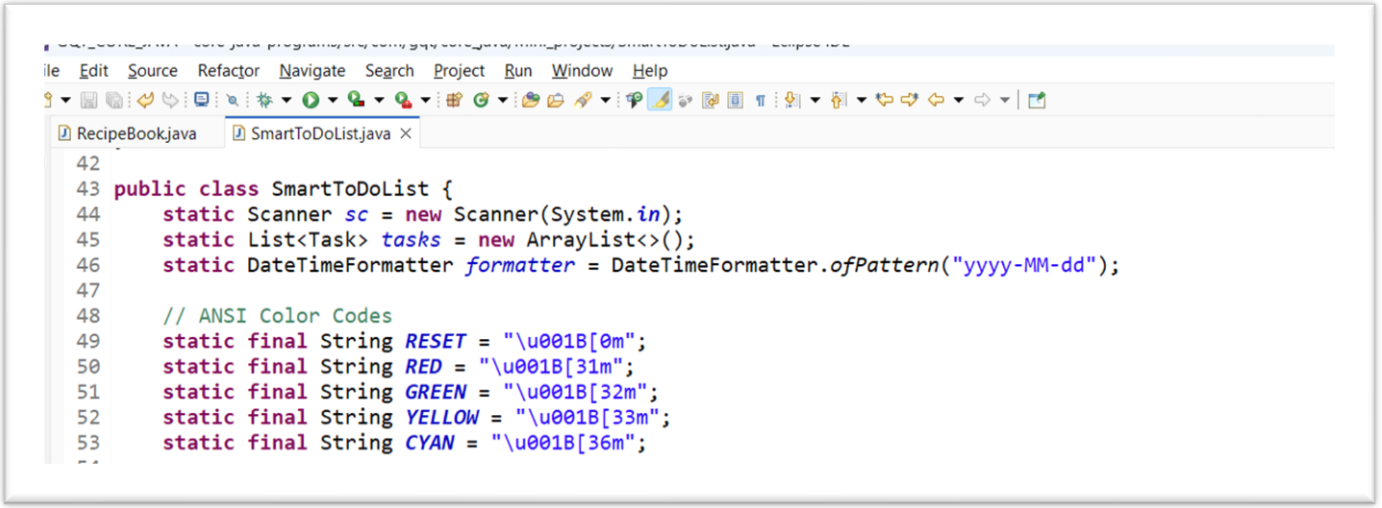
toString() Method

-Overrides the default string output to show a formatted task summary with colored status (Pending, Completed, or Overdue).

3. SmartToDoList Class (Main Logic):

This class contains the main program structure and user interaction.

4. Global Variables and Setup:



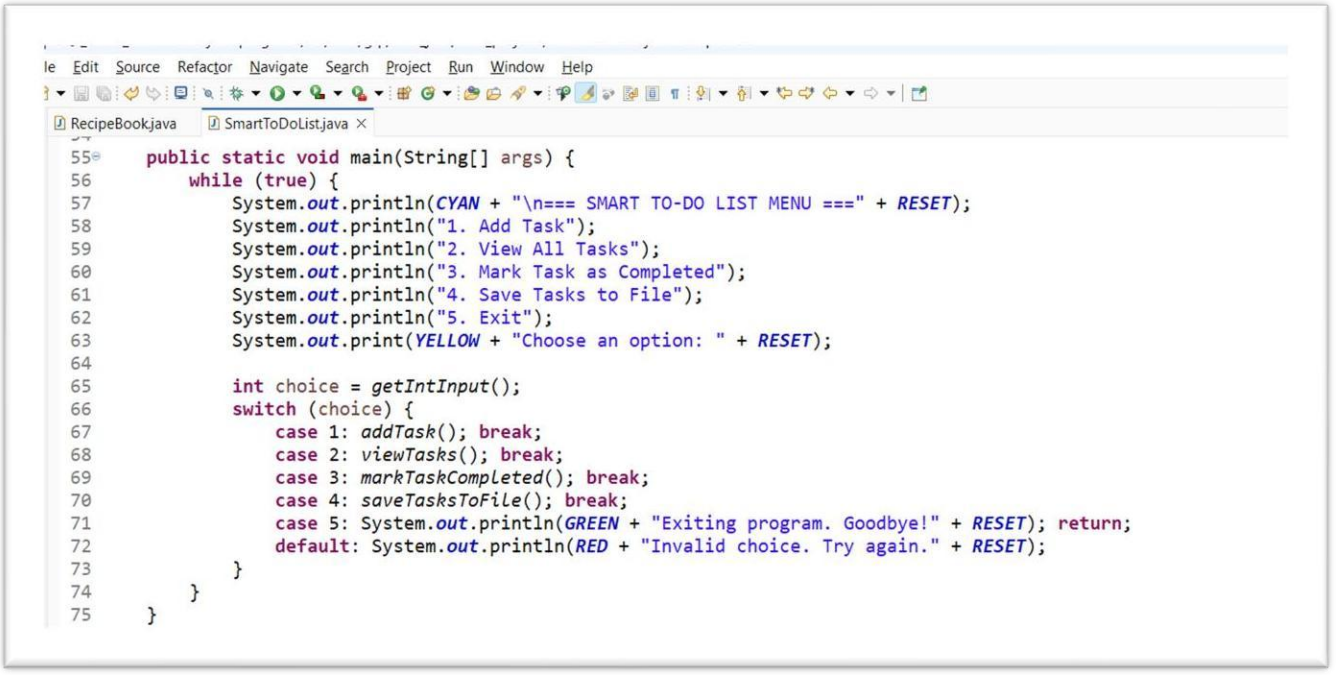
```
public class SmartToDoList {
    static Scanner sc = new Scanner(System.in);
    static List<Task> tasks = new ArrayList<>();
    static DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

    // ANSI Color Codes
    static final String RESET = "\u001B[0m";
    static final String RED = "\u001B[31m";
    static final String GREEN = "\u001B[32m";
    static final String YELLOW = "\u001B[33m";
    static final String CYAN = "\u001B[36m";
```

- tasks: Stores all task objects
- formatter: Parses string input into a date

ANSI Color Codes

Used to add color to console output (e.g., RED for errors, GREEN for success).

5. Main Method (Program Starts Here):



-- Shows a menu in a loop
-- Asks user to select options 1–5
-- Calls appropriate methods based on selection

6. addTask() Method:

Handles task creation:

- Gets user input: title, description, deadline, priority
- Validates date format using LocalDate.parse()
- Rejects tasks with past deadlines
- Creates a Task object and adds it to the list

```
77    static void addTask() {
78        System.out.print(YELLOW + "Enter task title: " + RESET);
79        String title = sc.nextLine();
80
81        System.out.print(YELLOW + "Enter task description: " + RESET);
82        String description = sc.nextLine();
83
84        System.out.print(YELLOW + "Enter deadline (yyyy-MM-dd): " + RESET);
85        String dateStr = sc.nextLine();
86        LocalDate deadline;
87        try {
88            deadline = LocalDate.parse(dateStr, formatter);
89        } catch (Exception e) {
90            System.out.println(RED + "Invalid date format. Task not added." + RESET);
91            return;
92        }
93
94        if (deadline.isBefore(LocalDate.now())) {
95            System.out.println(RED + "Deadline is in the past. Task not added." + RESET);
96            return;
97        }
98
99        System.out.print(YELLOW + "Enter priority (High/Medium/Low): " + RESET);
100       String priority = sc.nextLine();
101
102       Task task = new Task(title, description, deadline, priority);
103       tasks.add(task);
104       System.out.println(GREEN + "Task added successfully." + RESET);
105   }
```

7. viewTasks() Method

- Checks if the task list is empty
- Sorts tasks by deadline using Comparator
- Displays each task with status and color.

```
107    static void viewTasks() {
108        if (tasks.isEmpty()) {
109            System.out.println(RED + "No tasks available." + RESET);
110            return;
111        }
112
113        tasks.sort(Comparator.comparing(task -> task.deadline));
114        System.out.println(CYAN + "\n=== All Tasks Sorted by Deadline ===" + RESET);
115        for (int i = 0; i < tasks.size(); i++) {
116            System.out.println(CYAN + "Task #" + (i + 1) + RESET);
117            System.out.println(tasks.get(i));
118        }
119    }
```

8. markTaskCompleted() Method

- Displays all tasks
- Asks the user to enter task number
- Marks the selected task's completed field as true

```java
static void markTaskCompleted() {
    viewTasks();
    if (tasks.isEmpty()) return;

    System.out.print(YELLOW + "Enter the task number to mark as completed: " + RESET);
    int index = getIntInput();
    if (index < 1 || index > tasks.size()) {
        System.out.println(RED + "Invalid task number." + RESET);
        return;
    }

    tasks.get(index - 1).completed = true;
    System.out.println(GREEN + "Task marked as completed." + RESET);
}
```

9. saveTasksToFile() Method

-- Uses FileWriter to write all task details to todo_list.txt

-- ANSI codes are removed before writing to file.

```java
static void saveTasksToFile() {
    try (FileWriter fw = new FileWriter("todo_list.txt")) {
        for (Task task : tasks) {
            fw.write(task.toString().replaceAll("\u001B\\[[;\\d]*m", "")); // remove ANSI for file
            fw.write("----------\n");
        }
        System.out.println(GREEN + "Tasks saved to todo_list.txt successfully." + RESET);
    } catch (IOException e) {
        System.out.println(RED + "Error writing to file." + RESET);
    }
```

10. getIntInput() Method

- A helper method for safe integer input
- Re-prompts until the user enters a valid number.

```java
static int getIntInput() {
    while (!sc.hasNextInt()) {
        System.out.print(RED + "Please enter a valid number: " + RESET);
        sc.nextLine(); // clear invalid input
    }
    int num = sc.nextInt();
    sc.nextLine(); // clear newline
    return num;
}
}
```

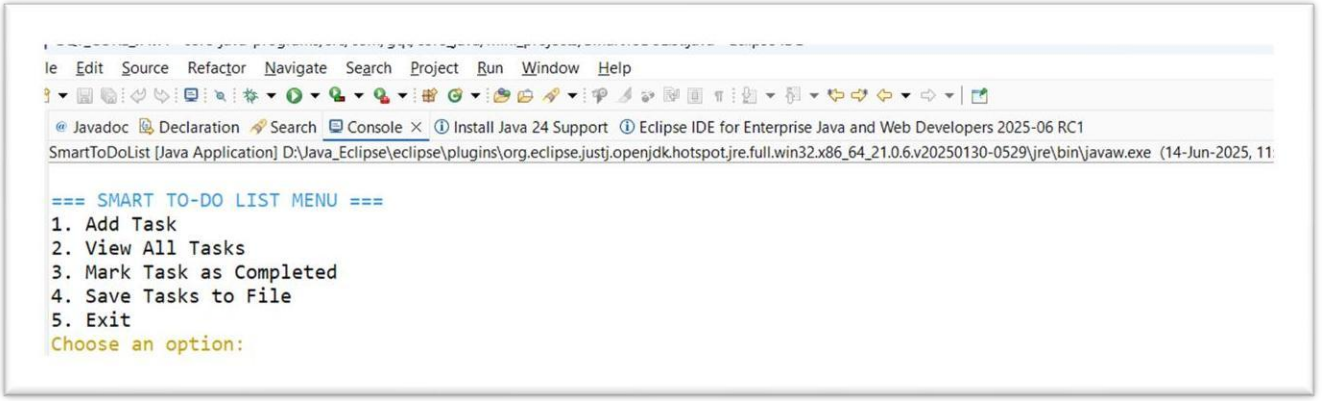
CHAPTER: 6

SAMPLE CONSOLE OUTPUT

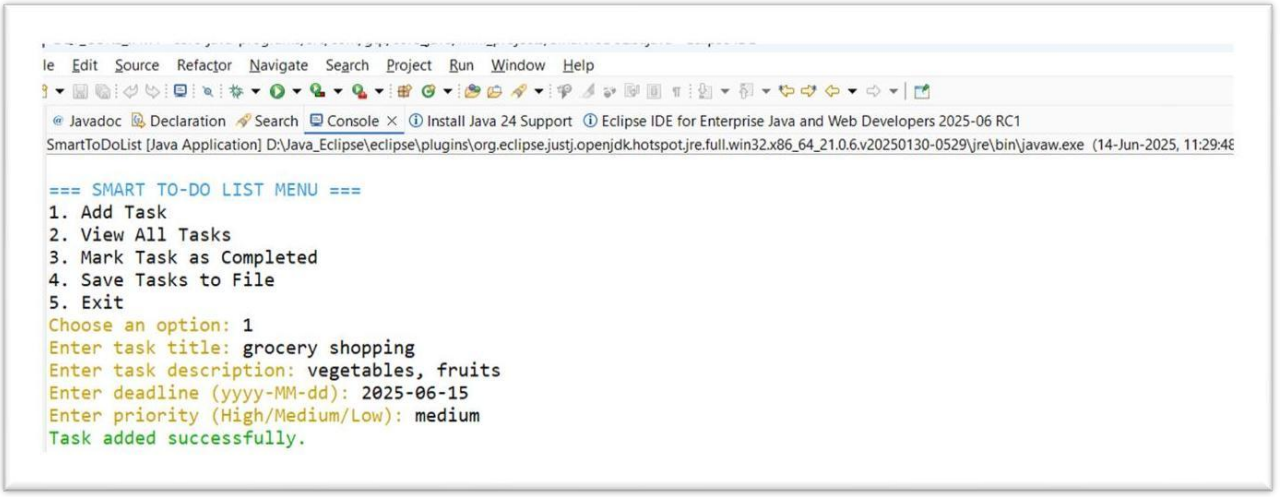
6. SAMPLE CONSOLE OUTPUT

The output of the **Smart TO-DO List with Deadlines** application reflects a clean, user-friendly console experience that enables users to manage tasks interactively. Through a menu-driven system, users can add new tasks, view all scheduled tasks, mark tasks as completed, and save their task list to a file. The application ensures a smooth experience with proper input validation and informative color-coded console feedback.
Below is a representation of the sample output generated during program execution:

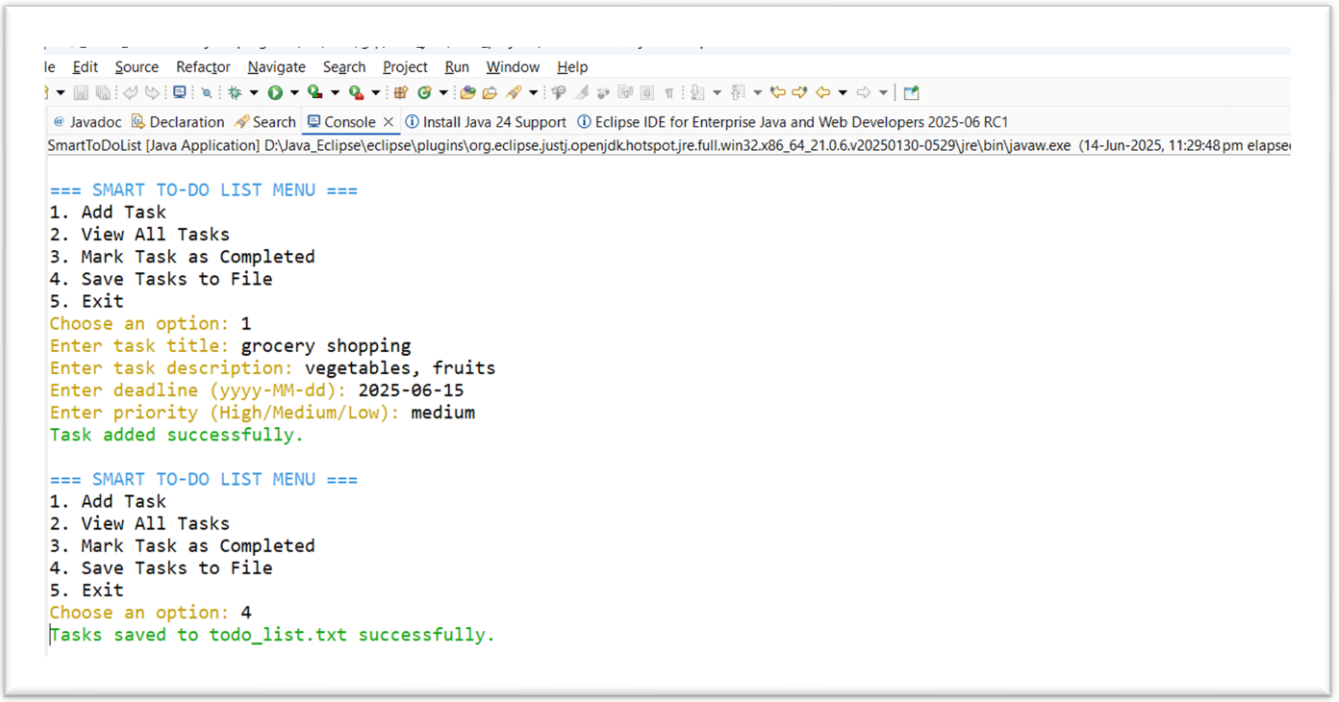
➡ When you run the program then the console ask the enter the choice:



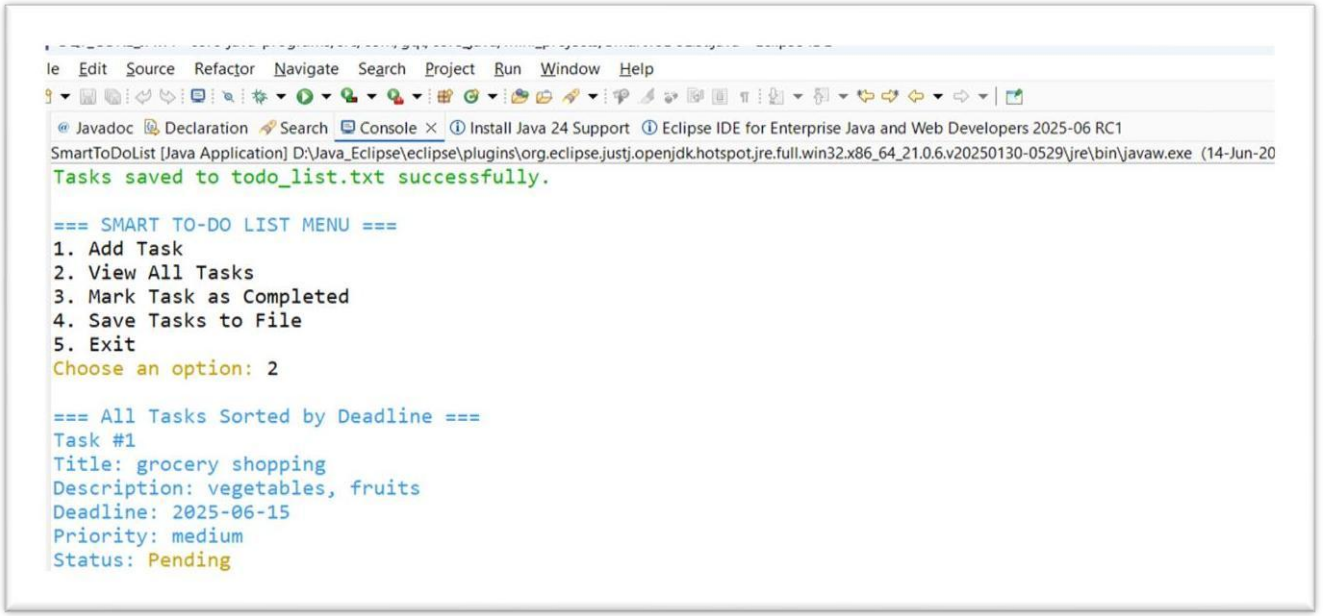
➡ When you enter the choice number 1 then it shows to enter your work title with description, date priority, etc..,:



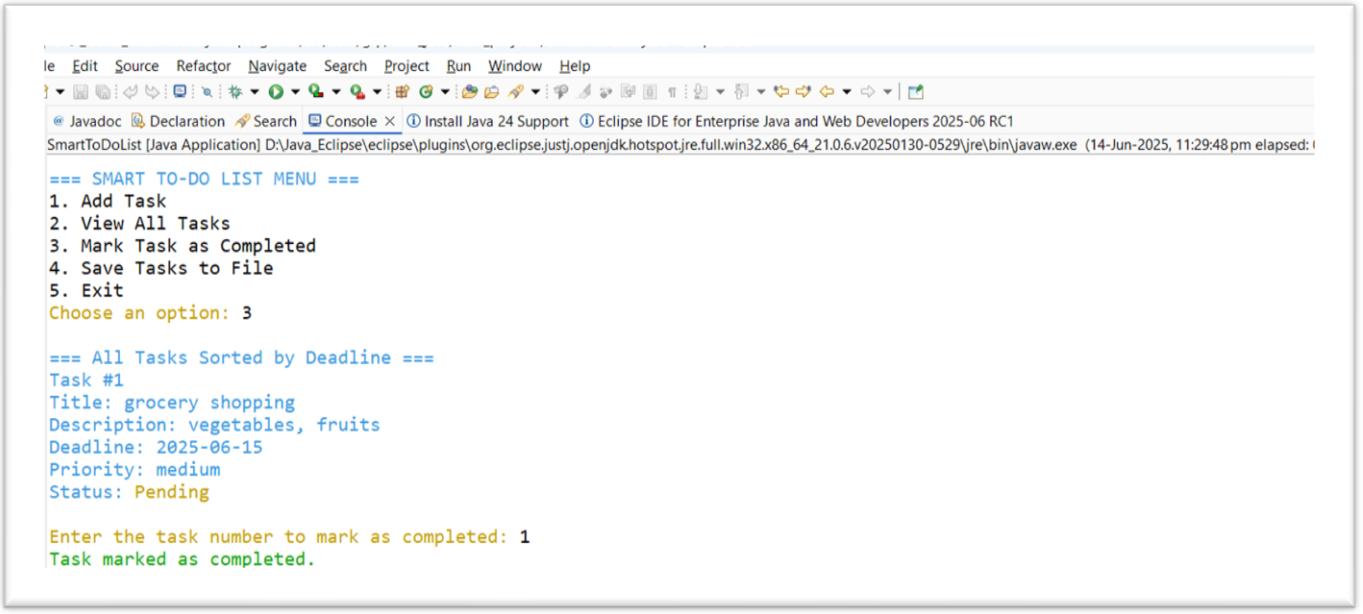
➡ Enter user enter the task which to be completed then clicks the choice number 4 to save it in the todo_list.txt file:



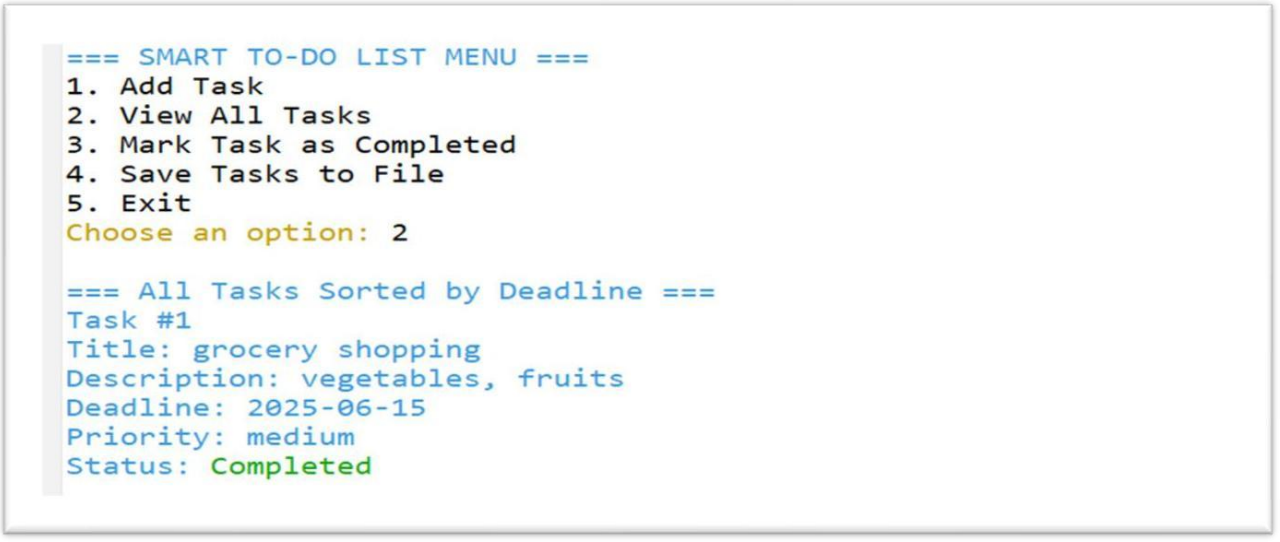
➡ If user want to see tasks then enter the choice number:2



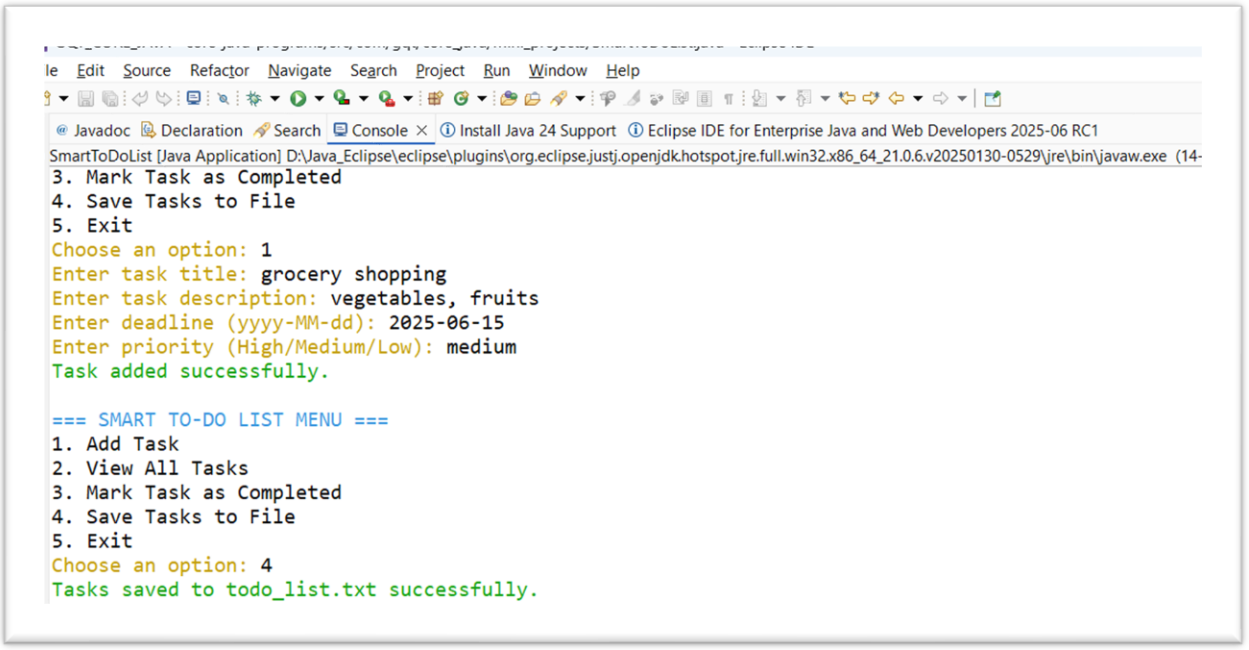
➡ If user want to make it as completed then enter the choice number 3 and then enter which Task number should be completed:



➡ If user want see that the task is marked as completed or not click the choice number 2:



➡ File output (todo_list.txt):



The output demonstrates real-time updates to task status and confirms successful operations, ensuring users can effectively manage their time-sensitive responsibilities.

CHAPTER: 7
CONCLUSION

7. Conclusion:

The **"Smart TO-DO List with Deadlines"** project represents a comprehensive approach to task management through a structured and interactive console-based Java application. It effectively combines practical utility with academic learning, offering users a reliable platform to manage daily activities and priorities. The project demonstrates how fundamental programming concepts can be transformed into real-world solutions using only core Java components.

Through the implementation of this project, the essential features of a task manager have been successfully realized: users can add, view, and mark tasks as completed, while the system intelligently sorts tasks by their deadlines and assigns meaningful statuses such as Pending, Completed, or Overdue. The application ensures that users receive accurate feedback through intuitive color-coded output, enhancing the overall user experience.

In addition to functional capabilities, the project excels in applying object-oriented programming principles. The separation of concerns through well-defined classes (Task and SmartToDoList) promotes code readability, maintainability, and scalability. Java's built-in features, such as the ArrayList, Scanner, LocalDate, and FileWriter, were utilized efficiently to handle data management, input, validation, and persistence.

The project also highlights the importance of user input validation and exception handling in building robust applications. Invalid formats, such as incorrect dates or numbers, are gracefully managed without crashing the program. This not only ensures application stability but also improves the user’s interaction with the system.

By integrating these core concepts, the project proves to be both educational and practical. It serves as a foundational learning experience for Java beginners and a functional tool for users who need a simple offline task organizer. The minimal yet effective use of external libraries demonstrates the power of native Java in developing useful applications.

In conclusion, the **"Smart TO-DO List with Deadlines"** project showcases the potential of Core Java in solving real-world problems. It successfully bridges the gap between learning and application by turning theory into practice, making it a valuable asset for both academic assessment and everyday use.

CHAPTER: 8

FUTURE ENHANCEMENTS

8. Future Enhancements:

The current version of the "Smart TO-DO List with Deadlines" is a fully functional and beginner-friendly console-based application. However, there are several opportunities for enhancement to expand its capabilities, improve user experience, and make it more scalable and feature-rich.

One of the most impactful enhancements would be integrating a **Graphical User Interface (GUI)** using JavaFX or Swing. This would make the application more visually appealing and accessible to non-technical users, replacing the command-line interface with buttons, lists, popups, and calendar pickers.

Another valuable improvement would be adding **real-time reminders and notifications**. By scheduling alerts for upcoming or overdue tasks, users could stay more effectively on top of their deadlines. This could be implemented using background threads or system tray notifications.

Introducing a **time field** alongside the current date-based deadline would increase the granularity of task scheduling, making it possible to set tasks for specific hours and minutes.

For better integration with existing productivity tools, the application could be enhanced to **sync tasks with Google Calendar or Outlook**, allowing users to view their to-do list across platforms.

Storing tasks in a **database like MySQL or SQLite** would allow the system to manage larger volumes of data, support multiple users, and add login-based functionality. This would also enable the use of SQL queries for searching, sorting, and filtering tasks.

Other enhancements include:

- **Search and Filter options:** Users could locate specific tasks by keyword, priority, or status.
- **Task categories or tags:** To organize tasks into work, personal, study, etc.
- **Exporting tasks** to formats like CSV or PDF for sharing or printing.

Each of these enhancements would add to the practicality and appeal of the application, making it more robust for real-world use while offering further learning opportunities in Java programming, database interaction, and user interface design.

CHAPTER: 9

REFERENCES

9. References:

In the development of the "Smart TO-DO List with Deadlines" project, various technical resources and educational platforms were instrumental in providing both conceptual knowledge and practical solutions. These references were essential for understanding Core Java concepts, syntax usage, and effective application design. Below is a detailed list of references used during the development of this application:

1. **Oracle-Java-Documentation**
   The official Java documentation from Oracle was extensively referred to for understanding class libraries, especially java.util, java.io, and java.time. It provided insights into how to effectively implement-collections,filehandling,anddateoperations.
   Link: https://docs.oracle.com/javase

2. **W3Schools-Java-Tutorial**
   This platform was useful for reinforcing basic Java syntax, understanding control statements, and grasping object-oriented concepts in a simple and clear manner.
   Link: https://www.w3schools.com/java

3. **GeeksforGeeks-Java-Programming-Guide**
   Served as a primary reference for implementing Java collections, exception handling, and date/time formatting. Also helpful in understanding best practices for writing efficient Java code.
   Link: https://www.geeksforgeeks.org/java

4. **TutorialsPoint-Java-Resources**
   Used for reviewing topics related to file I/O operations, class structures, and basic Java programmingworkflows.
   Link: https://www.tutorialspoint.com/java

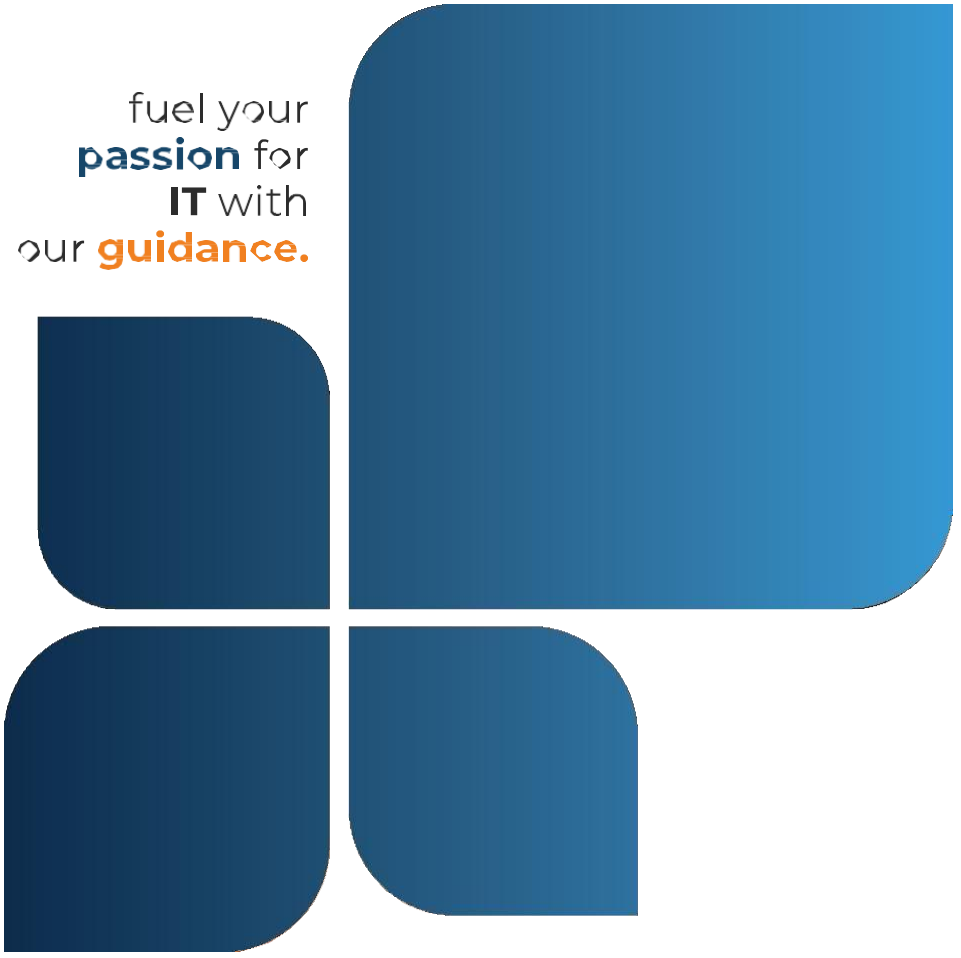
5. **Stack-Overflow**
   Community-driven discussions and answers were referenced to resolve specific coding errors, especially around Scanner input issues, file writing, and ANSI escape codes in the console.
   Link: https://stackoverflow.com

6. **Global-Quest-Technologies**
   Provided mentorship, training environment, and academic guidance throughout the course of the project. It was a valuable resource for applying Java skills in practical scenarios.

These references played a crucial role in transforming conceptual understanding into a fully functional Java-based project that is educationally rich and technically sound.

GLOBAL QUEST
TECHNOLOGIES

fuel your
passion for
IT with
our **guidance.**



Global Quest Technologies

#324,2nd Floor, 3 A Cross, Near
Seshadripuram First Grade College,
Above City Union Bank,
Yelahanka New Town,
Bengaluru-560064

+91 9448 403 469 | 080-49720009
info@gqtech.in | www.gqtech.in