Project Report
Smart Task Scheduler

Abstract:

This report outlines the development of the” Smart Task Scheduler,” a desktop application built with Java and JavaFX. The primary objective of this project is to create an intuitive tool for managing daily tasks, prioritized by urgency and deadline. The application leverages a Priority Queue data structure to automatically sort tasks, ensuring that the most critical items are always at the forefront. Task data is persisted between sessions using JSON serialization via the Gson library. The resulting application provides a clean user interface for adding, deleting, and filtering tasks, demonstrating a practical application of core Java principles, data structures, and GUI development.

1 Introduction:

In a fast-paced environment, effective task management is crucial for productivity. Many individuals struggle to organize a fluctuating list of responsibilities, often losing track of high-priority items. The” Smart Task Scheduler” project was undertaken to address this problem by creating a simple, yet powerful, desktop application. Unlike basic to-do lists, this scheduler intelligently organizes tasks based on user-defined priority (High, Medium, Low) and a specified deadline. This report details the project’s objectives, the technologies used, the development process, and the final outcome.

2 Tools Used:

The project was built using a combination of core Java technologies and modern libraries:

- Java (JDK 17): The core programming language used for all application logic.
- JavaFX: The modern GUI framework used to build the user interface, including windows, dialogs, lists, and buttons.
- Apache Maven: The build automation and dependency management tool. It was used to manage project libraries (JavaFX, Gson) and package the final executable .jar file.
- Gson Library: A Google-developed library for serializing Java objects into JSON format and deserializing them back. This was used for saving and loading tasks to a file.
- IntelliJ IDEA / Eclipse: The Integrated Development Environment (IDE) used for writing, debugging, and managing the code.

3 Steps Involved in Building the Project:

The development was approached in a modular, step-by-step manner:

1. Project Setup: A new Maven project was initialized. The pom.xml file was configured to include dependencies for JavaFX (controls, fxml) and Gson.

2. Data Model:

- A Task.java class was created to act as the data model. It includes fields for title, priority, and deadline.
- The Comparable interface was implemented, with the compare To method programmed to sort tasks first by priority (High > Medium > Low) and then by the earliest deadline.
- A TaskManager.java class was created to encapsulate the core logic, holding tasks in a Priority Queue.

3. Storage Layer:

- A Storage.java utility class was built.
- It contains two static methods: saveTasks(List) and loadTasks().
- These methods use Gson to write the task list to a tasks.json file and read from it on startup, handling potential IOExceptions.

4. User Interface (UI) Development:

- MainApp.java was created as the entry point, extending Application. It sets up the main stage, a ListView to display tasks, and toolbar buttons.
- A TaskDialog.java class was built to create a custom pop-up dialog for adding new tasks, featuring text fields and a ComboBox for priority.
- A TaskCell.java class was implemented to customize how each task appears in the ListView, adding colors (Red for High, Orange for Medium) based on priority.

5. Integration and Logic: The UI elements were connected to the TaskManager and Storage classes. The ”Add Task” button opens the dialog, and on success, adds the new task to the TaskManager, refreshes the UI's ObservableList, and calls Storage.saveTasks() to ensure persistence.

6. Packaging and Deployment:

- The maven-shade-plugin was added to the pom.xml to package the application and all its dependencies into a single executable .jar file.
- A Launcher.java class was created as a workaround for a common JavaFX deployment issue, ensuring the application could be launched directly from the JAR file.

4 Conclusion:

The” Smart Task Scheduler” project successfully meets all its objectives. It provides a clean, functional, and persistent task management solution. The use of a Priority Queue as the underlying data structure proved highly effective for sorting tasks automatically. This project was a valuable exercise in combining core Java concepts (data structures, file I/O) with a modern GUI framework (JavaFX) and build system (Maven). The final application is stable, easy to use, and serves as a strong foundation for future enhancements, such as editable tasks, calendar views, or cloud synchronization.