Java Coding Assignment: Task Management System

Overview

Design and implement a simple **Task Management System**, similar to a basic version of a todo-list app, but focusing on modular design, clean code, and testability. The system should allow users to create, update, delete, and list tasks. Each task has an ID, title, description, due date, priority, and status.

Requirements

Core features

1. **Create a Task**
    - Tasks should have the following fields:
        - ID (unique identifier)
        - Title (non-empty string)
        - Description (optional)
        - Due date (optional)
        - Priority (enum: LOW, MEDIUM, HIGH)
        - Status (enum: PENDING, IN_PROGRESS, COMPLETED)
2. **Update a Task**
    - Modify any of the modifiable fields (title, description, due date, priority, status).
    - Validate fields on update.
3. **Delete a Task**
    - Remove a task by ID.
4. **List Tasks**
    - List all tasks with filtering options on:
        - Status
        - Priority
        - Due date range

Additional Requirements

- Store tasks **in-memory**. Persistence is not required.

- Design the system to accommodate future extensibility (e.g., adding users or project grouping).
- Exception handling: handle invalid input gracefully.
- Write comprehensive **unit tests** covering core functionalities.
- Use **Java 8+ features** where appropriate (Streams, Optionals, Lambdas).
- Follow clean code principles (naming, separation of concerns, etc.).

Deliverables

1. **Source code** organized into modules/packages with clear separation (e.g., model, service, repository, api).
2. A set of **unit tests** for the main functionalities.
3. A short **README** or comments describing how to run the program and the design decisions made.

Evaluation Criteria

- Correctness: Does the system fulfill the requirements?
- Code Quality: Readability, modularity, adherence to Java best practices.
- Unit Test Coverage and Quality: Are tests meaningful and comprehensive?
- Error Handling: Robustness to invalid input and edge cases.
- Design: Separation of concerns, extensibility.
- Use of Java features: Effective use of Streams, Optionals, Enums, etc.

Optional Extensions (for bonus consideration)

- Add sorting option to list tasks by due date or priority.
- Implement a simple CLI or REST API interface (can be mocked or lightweight).
- Implement concurrency control if accessed from multiple threads.