To-Do List App

Fundamental of Programming 24/25

A to-do list is one of the most popular introductory projects, widely used to demonstrate understanding of key programming concepts and build problem-solving skills. This project offers an opportunity to practice Java programming, object-oriented principles, and basic data storage solutions while working on a real-world application.

Project Introduction

Time management is crucial for any student, especially for freshmen who are adjusting to university life. The goal of this project is to develop a simple To-Do List Application that will help students track and manage their tasks effectively. By building this project, students will learn key programming concepts while also creating a useful tool.

Project Prerequisite

Before starting, ensure you have a strong understanding of the following topics, as they are essential for completing this assignment:

- Sorting Algorithms – Implement a sorting mechanism (e.g., Bubble Sort or any other algorithm).
- Searching Algorithms – Implement a searching mechanism (e.g., Linear Search or any other algorithm).
- Object-Oriented Programming (OOP) – Proper usage of classes, inheritance, and polymorphism.
- Basic Database Operations – Familiarize yourself with SQL commands for data manipulation.
- Input/Output Handling – Reading and writing data, either from a CSV file or database.

Outline

Basic Features (8 marks)

Task Creation (1 mark)

The application should allow users to create tasks with a

- **Title**
- **Description**
- **Due Date**
- **Completion Status (complete / incomplete).**
- **Category**
- **Priority Level (low / medium / high)**

```
=== Add a New Task ===
Enter task title: Finish Assignment
Enter task description: Complete the Java programming assignment
Enter due date (YYYY-MM-DD): 2024-10-15
Enter task category (Homework, Personal, Work): Homework
Priority level (Low, Medium, High): High

Task "Finish Assignment" added successfully!
```

Task Management (½ mark)

Users should be able to mark tasks as completed or incomplete and update task details such as the description or due date.

```
=== Mark Task as Complete ===
Enter the task number you want to mark as complete: 1

Task "Finish Assignment" marked as complete!
```

Task Deletion (½ mark)

Implement the functionality for users to delete tasks from the checklist.

```
=== Delete a Task ===
Enter the task number you want to delete: 2

Task "Grocery Shopping" deleted successfully!
```

Task Sorting (½ mark)

Users must be able to sort tasks by due date in ascending or descending order.

Implement a sorting algorithm such as Bubble Sort, Selection Sort, or another in Java. This functionality should be independent of the SQL query.

```
=== Sort Tasks ===
Sort by:
1. Due Date (Ascending)
2. Due Date (Descending)
3. Priority (High to Low)
4. Priority (Low to High)

> 1
Tasks sorted by Due Date (Ascending)!
```

Task Searching (½ mark)

Users are able to use the **full text search** feature in the system to search for the relevant tasks.

Implement a searching algorithm such as linear search, binary search, or another in Java. This functionality should be independent of the SQL query.

```
=== Search Tasks ===
Enter a keyword to search by title or description: Finish

=== Search Results ===
1. [Incomplete] Finish Assignment - Due: 2024-10-15 - Category: Homework -
Priority: High
```

Recurring Tasks (1 mark)

Add an option for users to create recurring tasks (e.g., daily, weekly, monthly).

The system should **automatically create new tasks** when one recurrence is **completed**.

```
=== Add a Recurring Task ===
Enter task title: Weekly Report
Enter task description: Submit the weekly progress report
Enter recurrence interval (daily, weekly, monthly): Weekly

Recurring Task "Weekly Report" created successfully!
```

Task Dependencies (2 mark)

Users should be able to link tasks, marking one as dependent on another. (**Multiple Dependencies and Nested Dependencies are possible!**)

```
=== Set Task Dependency ===
Enter task number that depends on another task: 3
Enter the task number it depends on: 1

Task "Project Submission" now depends on "Finish Assignment."
```

In some cases, certain tasks depend on the completion of other tasks. For example, if you have a project that consists of several smaller tasks, one task might need to be completed before another can begin. This is known as a **task dependency**.

How Task Dependencies Work:

- A "dependent task" is one that cannot be started until another "preceding task" has been marked as complete.
- For instance, if Task B depends on Task A, Task A must be completed before Task B can be started.

Rules for Task Dependencies:

1. When a user tries to mark a dependent task as "complete" while its preceding task is still "incomplete," the system should not allow it. Instead, a warning message will be displayed.
2. If a dependent task is attempted to be marked as complete without following the dependency order, the system will give an error message to remind the user of the dependency.

3. A cycle of dependencies is not allowed! ***Tips: Use slow and fast pointers to detect cycle***
    - Task A depends on Task B
    - Task B depends on Task C
    - Task C depends on Task A

Warning Messages:

- If the user attempts to mark a dependent task complete before its preceding task, show a warning like:
    - `Warning: Task "Task B" cannot be marked as complete because it depends on "Task A". Please complete "Task A" first.`

Example Scenario:

1. Task A: "Complete research for the project" (Preceding task)
2. Task B: "Write project report" (Dependent task)

If the user tries to complete Task B before completing Task A, the system should prevent the action and display the warning.

```
=== View All Tasks ===
1. [Incomplete] Task A: Complete research - Due: 2024-10-12
2. [Incomplete] Task B: Write project report - Due: 2024-10-15 (Depends on
Task A)
```

```
=== Mark Task as Complete ===
Enter the task number you want to mark as complete: 2

Warning: Task "Write project report" cannot be marked as complete because
it depends on "Complete research." Please complete "Complete research"
first.

=== Mark Task as Complete ===
Enter the task number you want to mark as complete: 1

Task "Complete research" marked as complete!

=== Mark Task as Complete ===
Enter the task number you want to mark as complete: 2

Task "Write project report" marked as complete!
```

Edit Task (1 mark)

Users are able to select which task to edit for the information. Task Dependencies (2 mark) function can be added here.

```
=== View All Tasks ===
1. [Incomplete] Task A: Complete research - Due: 2024-10-12
2. [Incomplete] Task B: Write project report - Due: 2024-10-15 (Depends on
Task A)
```

```
=== Edit Task ===
Enter the task number you want to edit: 1

What would you like to edit?
1. Title
2. Description
3. Due Date
4. Category
5. Priority
6. Set Task Dependency
7. Cancel

> 2
Enter the new description: Complete literature review

Task "Complete research" has been updated to "Complete literature review."
=== View All Tasks ===
1. [Incomplete] Task A: Complete literature review - Due: 2024-10-12
2. [Incomplete] Task B: Write project report - Due: 2024-10-15 (Depends on
Task A)
```

Storage System (½ mark)

Store tasks in a database (preferably SQL-based), or alternatively, in a CSV file.

Data Load State (½ mark)

The fetched data should be converted and loaded as an object (Refer to OOP for more information). ***Hints: A Class of Task should be created.***

Extra Features (Max 4 marks)

Graphical User Interface (2 marks)

Create a simple GUI using JavaFX, Java Spring, or HTML (**with no JavaScript**). This GUI should allow users to:

- Add new tasks.
- View tasks in a list.
- Sort tasks using buttons for ascending/descending order.
- Mark tasks as complete.
- Delete tasks.

The design doesn't need to be complex, but it should be functional and user-friendly.

Email Notification System (1 mark)

Implement an email notification system that alerts users when a task's due date is approaching. For example:

- Send an email reminder when a task is due within 24 hours.
- Use any email API or JavaMail for sending emails.

Provide configuration options for the user to input their email address.

```
=== Email Notification ===
Sending reminder email for task "Finish Assignment" due in 24 hours.
```

Data Analytics (1 mark)

Implement a simple analytics dashboard to show the users statistics such as:

- Number of tasks completed vs. pending.
- Task completion rate over time.
- Categorized task summary.

```
=== Analytics Dashboard ===
- Total Tasks: 10
- Completed: 5
- Pending: 5
- Completion Rate: 50%
- Task Categories: Homework: 3, Personal: 5, Work: 2
```

Vector Search (3 marks)

Vector Search feature, which allows users to search for tasks based on semantic similarity rather than exact keyword matches. For example, searching for "**assignment**" could return results like "**homework**," "**project**," or "**task**," if these words are semantically related in the context of task descriptions.

Steps to Implement Vector Search:

1. **Generate Embeddings:**
   You will need to convert task descriptions into vector embeddings. Embeddings are mathematical representations of words that capture their semantic meaning.

   To achieve this, you can use a pre-trained word embedding model such as Word2Vec or BERT to generate vectors from task descriptions.

   Java libraries like **deeplearning4j** can be used for this. Or you can find any **Hugging Face Inference API** which is suitable such as sentence-transformers/all-MiniLM-L6-v2 to create embedding for your tasks.

   Store the generated embeddings in your database alongside the task descriptions.

2. **Database Integration:**
   Modify your database schema to store embedding vectors as part of each task's data. Embedding vectors can be stored as arrays of floating-point numbers.

3. **Querying with Vector Search:**
   When a user inputs a search query, convert the query into a vector using the same embedding model.

   Compare this vector with the stored task embeddings to find the most similar tasks.

   Implement a **cosine similarity function** to calculate the similarity between the query vector and task vectors. **The closer the similarity score is to 1, the more semantically related the tasks are.**

4. **Return Search Results:**
   Sort the results based on similarity and return the most relevant tasks to the user.

```
=== Search Tasks ===
Enter a keyword or phrase to search tasks: assignment
```

```
=== Vector Search Results ===
1. [Incomplete] Submit Homework - Due: 2024-10-12 - Category: Homework
2. [Incomplete] Finish Project - Due: 2024-10-15 - Category: Work
```