IMY 220
Assignment 7

TypeScript

General Instructions

- This assignment must be completed and submitted by the due date which is available on ClickUP.
- This assignment is a take-home assignment and may take one or two days to complete.
- This assignment should be completed on your own, but you may come to the tutorial session or email the assistant lecturer if you need further assistance.
- No late / email submissions will be accepted.

This assignment focuses on **Typescript**. You will be required to create a simple to-do list application using Typescript and Node. **TypeScript :any is not allowed and will not award you any marks**.

Note: *Where appropriate, you should* ***always use ES6 syntax****. You may* ***not*** *write any loops (e.g.,* `for` *loops /* `while` *loops) or use the* `.forEach()` *function in this assignment. All of the functionality must be implemented with the appropriate JS Array functions as discussed in the slides and resources linked from the slides. Marks are awarded for following these instructions.*

Part 1 - Simple Node Project

Initialise a simple Node project with TypeScript as shown in the lecture videos / slides. You do **not** need to create an entire React application for this assignment, just a simple Node project with a *package.json, server.js,* and *index.ts* file.

Download the provided files off of ClickUP. These files contain *index.html* which can be placed in the root directory of your project (next to *index.ts*). *index.html* has already been set up with a simple “add task” form and a list to display tasks.

server.js should initialise a simple Express server and serve *index.html*, which already links to *index.js* (that should be generated from your *index.ts*).

Part 2 - Task Interface

Create a Typescript Interface called **Task** inside of *index.ts*. This interface should be created by carefully looking at the "add task" form in *index.html* and selecting the best types that correspond to the values returned from each input. Pay attention to which fields are required.
Take note:

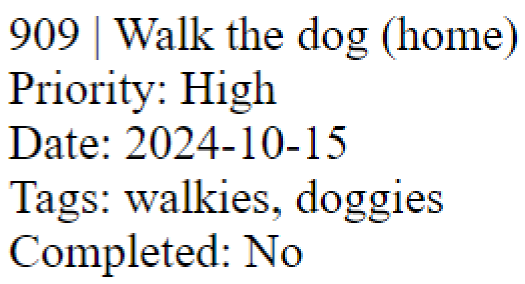
- Add an *id* to the interface. Ids are integers.
- The *date* can be a string (since that is the type returned from a date input)
- The *category* can **only** be one of the given values in the input.
- The *priority* is a range that returns either one, two or three that corresponds to Low, Medium and High respectively. To make this semantically easier to understand, create an **enumeration** to use in your Interface. You can define this enumeration in *index.ts* as well.
- The *tags* should be **optional** but if present, contain multiple values.

Part 3 - TaskManager Class

Create a **TaskManager** class in *index.ts*. The constructor should take an **optional** array of *Task* objects to assign to its member variable (a *Task* array).

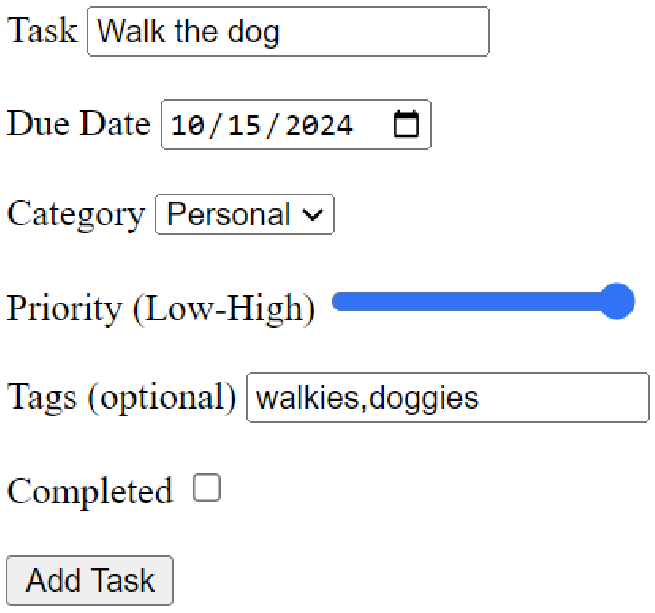
This class should have the following member functions:

- **getTasks()** which returns the member variable
- **addTask(task)** which adds the passed in task to the member variable.
- **listTasks()** which returns an array of strings where each string represents a task from the member variable, and looks as follows:

  ```
  909 | Walk the dog (home)
  Priority: High
  Date: 2024-10-15
  Tags: walkies, doggies
  Completed: No
  ```

  Where 909 is the ID, and the input form looks like this:


Task: Walk the dog

Due Date: 10/15/2024

Category: Personal

Priority (Low-High)

Tags (optional): walkies,doggies

Completed ☐

You will need to account for any missing values, and displaying the priority correctly.

- **sortTasksbyPriority()** which sorts the member variable by priority in descending order (from highest to lowest).
- **findTask(input)** which finds a given task in the member variable using the input parameter. This parameter can either be an integer (corresponding to the task id) or a string (corresponding to the task name). The function returns the task (or undefined if no task is found).

Remember that this assignment uses TypeScript, meaning that you need to specify the parameter and return types of all functions and the types of variables. Your code will not run otherwise and marks will be awarded for correct typing.

You may test your class works by using the code already defined in *index.html*.

Submission Instructions

Place these in a folder named `A5_u12345678` where `12345678` is your student number.

- index.ts

Zip the **folder** and upload this to ClickUP in the relevant submission slot before the deadline.