**Job Simulation: Task Manager with Live Status Dashboard**

**Project Title:**

Implementation of a Task Manager

**Role:**

Front-End Development Intern

**Technology Stack:**

HTML, CSS, JavaScript

**Objective**

To design and implement a live task manager that:

* Loads tasks asynchronously
* Displays real-time task progress
* Allows users to control task execution and delayed notifications

This project introduces key JavaScript timing functions (setTimeout, setInterval, clearInterval) and asynchronous handling using async/await.

**Task Overview**

As part of your hands-on learning experience, you are required to build an interactive dashboard that simulates live task progress updates. The system should:

* Load task data asynchronously
* Update each task’s progress in real-time
* Enable users to start and stop progress
* Trigger delayed notifications using setTimeout

Each task must be presented with:

* A task name
* A progress percentage
* A current status

The dashboard UI should be visually styled using CSS to enhance usability and responsiveness.

**Task Requirements**

**1. Functionality**

* Use async/await to simulate loading task data with a delay (Promise + setTimeout)
* Display a **"Loading…"** message while tasks are being fetched
* Use setInterval() to increase each task’s progress by **10% per second**
* Use clearInterval() to stop progress updates on button click
* Use setTimeout() to display a **delayed notification**
* Ensure that:
  + Progress stops at **100%**
  + Status updates to **"Completed"** upon completion

**2. User Interface (UI)**

* Build a responsive and visually appealing layout using **HTML and CSS**
* Include a **header** titled **"Async Task Manager"**
* Provide control buttons:
  + **Load Tasks**
  + **Start Progress**
  + **Stop Progress**
  + **Delay Notification**
* Show each task in a **card or row format** displaying:
  + Task name
  + Status (Pending, In Progress, Completed)
  + A visual progress bar
* Use background color indicators:
  + Green for **Completed**
  + Yellow for **In Progress**
* Display notification messages in a **separate panel**

**3. Code Structure**

* Use document.getElementById() or querySelector() for DOM interaction
* Store setInterval() IDs in variables for precise control via clearInterval()
* Write **modular functions** to handle:
  + Task loading
  + Progress updates
  + Stopping updates
  + Displaying delayed messages
* Ensure task progress is capped at 100% to prevent overflows or errors

**Bonus (Optional)**

* Disable the **Start Progress** button while progress is active
* Assign each task a **different delay duration**
* Add **audio or animations** on task completion
* Implement a **Light/Dark Mode Toggle** using a CSS class switch

**Deliverables**

Submit a project folder containing the following files:

* index.html – HTML layout and structure of the task dashboard
* style.css – Styling rules for design, status indicators, and responsiveness
* script.js – JavaScript logic for:
  + Asynchronous loading
  + Real-time progress control
  + Timing-based interactions

**Note:** Include a 2–3 line comment in script.js explaining how async loading and real-time updates are handled using async/await, setInterval(), and setTimeout().

**Learning Outcomes**

By completing this simulation, you will gain hands-on experience with:

* Handling asynchronous data fetching using async/await
* Implementing real-time logic using setInterval() and clearInterval()
* Managing delayed events via setTimeout()
* Building and styling dynamic user interfaces with HTML/CSS/JS
* Structuring scalable and interactive dashboards with clean JavaScript



