Lab 2: Advanced JavaScript Concepts for MERN

Project: Task Reminder System

Objective: Build a task reminder system that schedules reminders and handles asynchronous delays.

Tasks

- Create an array of task objects, where each task has fields like title, dueTime (in minutes from now), and priority.
- Write functions to:
- Add a new task to the list.
- Sort tasks by their priority.
- Display tasks due within a certain timeframe (e.g., next 10 minutes).
- Simulate sending reminders using setTimeout() based on the task's dueTime.

Use try...catch to handle errors, such as invalid task data or missing fields.Export and import the task management functions using Jav aScript modules.

INPUT:-INDEX.HTML

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Task Reminder System</title>

    <link rel="stylesheet" href="styles.css">
```

```
</head>

<body>

    <div class="container">

        <h1>Task Reminder System</h1>

        <form id="task-form">

            <input type="text" id="task-title" placeholder="Task Title"
required>

            <input type="number" id="task-time" placeholder="Due Time
(minutes)" required>

            <input type="number" id="task-priority" placeholder="Priority"
required>

            <button type="submit">Add Task</button>

        </form>

        <div id="task-list">

            <h2>Upcoming Tasks</h2>

            <ul id="tasks"></ul>

        </div>

    </div>
```

```
    <script src="script.js"></script>

</body>

</html>
```

Styles.css

```
body {

    font-family: Arial, sans-serif;

    background-color: #f4f4f4;

    margin: 0;

    padding: 0;

}

.container {

    max-width: 600px;

    margin: 50px auto;

    padding: 20px;

    background: white;
```

```
    border-radius: 10px;

    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

h1, h2 {

    text-align: center;

    color: #00796b;

}

form {

    display: flex;

    flex-direction: column;

    gap: 10px;

}

input, button {

    padding: 12px;

    font-size: 16px;

    border: 1px solid #bbb;
```

```
    border-radius: 5px;

}

button {

    background: #00796b;

    color: white;

    cursor: pointer;

}

button:hover {

    background: #004d40;

}

ul {

    list-style: none;

    padding: 0;

}

li {

    padding: 12px;
```

```
    background: #ffcc80;

    border-radius: 5px;

    margin-bottom: 8px;

    display: flex;

    justify-content: space-between;

    align-items: center;

}
```

Script.js

```
const tasks = [];



// Function to add a task

function addTask(title, dueTime, priority) {

    try {

        if (!title || isNaN(dueTime) || isNaN(priority)) {

            throw new Error("All fields (title, dueTime, priority) are
required.");
```

```
        }

        const task = { title, dueTime, priority, addedAt: new Date() };

        tasks.push(task);

        displayTasks();

        scheduleReminder(task);

        alert(`✅ Task "${title}" added successfully!`);

    } catch (error) {

        console.error(`❌ Error adding task: ${error.message}`);

    }

}



// Function to sort tasks by priority

function sortTasks() {

    tasks.sort((a, b) => a.priority - b.priority);

}
```

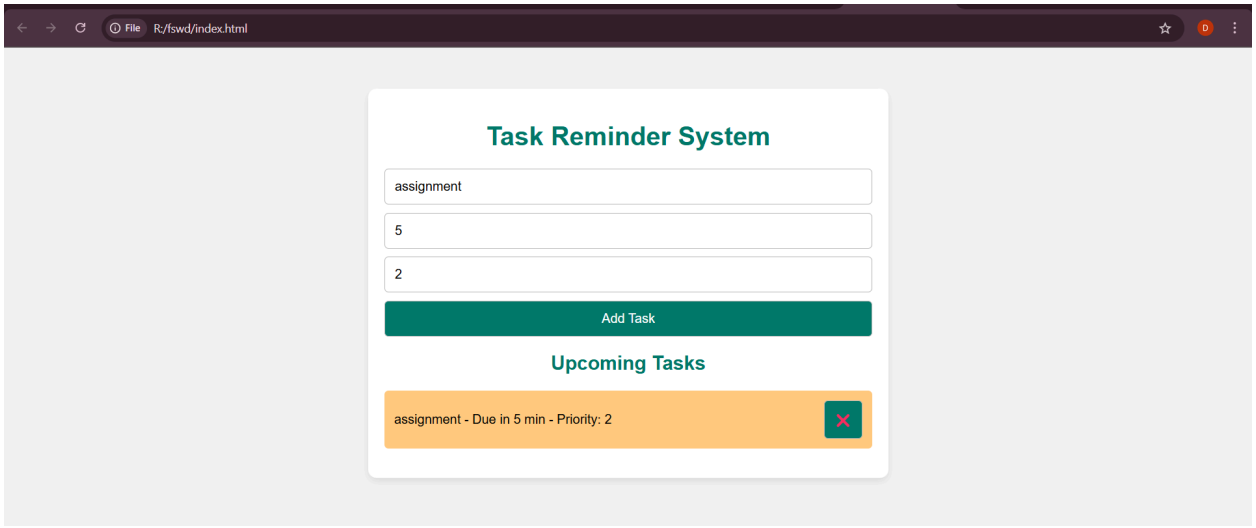
```
// Function to display tasks due within a certain timeframe (e.g., next 10 minutes)

function getUpcomingTasks(minutes = 10) {

    const now = new Date();

    return tasks.filter(task => {

        const taskDue = new Date(task.addedAt.getTime() + task.dueTime * 60000);

        return (taskDue - now) / 60000 <= minutes;

    });

}



// Function to delete a task

function deleteTask(index) {

    tasks.splice(index, 1);

    displayTasks();

}
```

```
// Function to get all tasks

function getTasks() {

    return tasks;

}




// Function to schedule reminders

function scheduleReminder(task) {

    setTimeout(() => {

        alert(`🔔 Reminder: Task '${task.title}' is due now!`);

    }, task.dueTime * 60000);

}




// Function to update UI

function displayTasks() {

    const taskList = document.getElementById("tasks");

    taskList.innerHTML = "";
```

```
    sortTasks();




    getTasks().forEach((task, index) => {

        const li = document.createElement("li");

        li.innerHTML = `${task.title} - Due in ${task.dueTime} min -
Priority: ${task.priority}

                        <button class="delete-btn"
data-index="${index}">❌</button>`;

        taskList.appendChild(li);

    });




    document.querySelectorAll(".delete-btn").forEach(button => {

        button.addEventListener("click", function () {

            const index = this.getAttribute("data-index");

            deleteTask(index);

            displayTasks();

        });
```

```
    });

}



// Form event listener

document.getElementById("task-form").addEventListener("submit", event => {

    event.preventDefault();

    const title = document.getElementById("task-title").value;

    const dueTime = parseInt(document.getElementById("task-time").value);

    const priority =
parseInt(document.getElementById("task-priority").value);



    addTask(title, dueTime, priority);

});
```

OUTPUT:-



Github repository:-https://github.com/RuchaParmar-D24IT156/Practicle-/tree/main