**G1f22ubscs048  
sadiqa   
 To Do List  
Summary**

We built a simple **To-Do List** backend application using **Node.js** and **Express.js**. The purpose of this app is to manage tasks, allowing the user to add, view, and delete tasks.

1. **Setup**:
   * We started by setting up a basic **Node.js** project and installed **Express.js**, which helps us create the server and handle API routes easily.
2. **Task Management**:
   * Instead of using a database, we used a simple **array** to store the tasks. Each task is an object with:
     + id: A unique identifier for each task.
     + taskName: The name/description of the task.
3. **API Routes**:
   * We created **3 API routes** to perform different actions on the tasks:
     + **POST /addTask**:
       - This route allows users to add a new task. The user sends the task name in the body of the request (using Postman), and the server adds this task to the array with a unique id.
     + **GET /tasks**:
       - This route retrieves and displays all tasks currently stored in the array. When the user sends a GET request, the server responds with the list of tasks.
     + **DELETE /task/:id**:
       - This route allows users to delete a task by its id. The user sends a DELETE request with the task ID in the URL (using Postman), and the server removes that specific task from the array.
4. **Testing with Postman**:
   * We used **Postman** to test the API:
     + To add a task, we used a POST request to /addTask.
     + To view all tasks, we used a GET request to /tasks.
     + To delete a task, we used a DELETE request to /task/:id.
5. **Array as Data Storage**:
   * Instead of using a complex database, we kept the tasks in an array. Every time the server restarts, the tasks will reset, but for this basic version, an array is enough for storage.

**Outcome:**

* The app allows users to:
  + **Add tasks** by sending a POST request.
  + **View tasks** by sending a GET request.
  + **Delete tasks** by sending a DELETE request with a specific task ID.