Thought Note For Database Mongo DB and MySql:-

# TH for Todo.js:-

**Overview:-**

The React Todo List component is designed to facilitate task management within a web application. It provides users with a simple interface to add new tasks and delete existing ones. The component utilizes React, a popular JavaScript library for building user interfaces, and employs React Hooks for managing state within functional components. Additionally, it communicates asynchronously with a backend server to store and delete tasks using the Fetch API.

**Features:-**

1. Add New Tasks: Users can input a task in the designated text field and add it to the list by clicking the "Add task" button.
2. Delete Tasks: Users can remove tasks from the list by clicking the delete button associated with each task.
3. Asynchronous Communication: The component interacts with a backend server to store and delete tasks asynchronously. This ensures a smooth user experience and prevents blocking the main thread during network requests.

**Description**: This function handles the submission of a new task when the user submits the form.

1. Parameters:
   1. event: The submit event triggered when the user submits the form.
2. Behavior:
   1. Prevents the default form submission behavior to avoid page reload.
   2. Adds the new task entered by the user to the list of tasks stored in the component's state.
   3. Generates a unique task ID using the react-uuid library to ensure each task has a distinct identifier.
   4. Logs the task ID and name to the console for debugging purposes.
   5. Sends a POST request to the backend server to store the new task asynchronously.
   6. Clears the input field after successfully adding the task to the list.
3. deleteTask(t)
   1. Description: This function deletes a specified task from the list when the user clicks the delete button associated with that task.
4. Parameters:
   1. t: The task to be deleted from the list.
5. Behavior:
   1. Removes the specified task from the list of tasks stored in the component's state.
   2. Sends a DELETE request to the backend server to delete the task from the server asynchronously.

# TH for backend.js:-

1. **Overview:-**
   1. The Node.js backend code provides a RESTful API for managing tasks in a Todo list application. It utilizes the Express.js framework for handling HTTP requests and responses, along with the mysql2 package for interacting with a MySQL database. Additionally, it incorporates CORS middleware to enable cross-origin resource sharing.
2. **Features:-**
   1. GET Endpoint: Retrieves all tasks from the database and returns them as JSON.
   2. POST Endpoint: Adds a new task to the database.
   3. DELETE Endpoint: Deletes a specified task from the database.
3. **Endpoints:-**
   1. GET /task
      1. Description: Retrieves all tasks from the todo table in the database.
      2. Method: GET
      3. Response: JSON array containing task objects.
   2. POST /task
      1. Description: Adds a new task to the todo table in the database.
      2. Method: POST
      3. Request Body: JSON object containing task\_id and task\_name fields.
      4. Response: Success message indicating the task was added successfully.
   3. DELETE /cut
      1. Description: Deletes a specified task from the todo table in the database.
      2. Method: DELETE
      3. Request Body: JSON object containing the t field, which specifies the task to delete.
      4. Response: Success message indicating the task was deleted successfully.
4. **Database Schema:-**
   1. The todo table schema should include at least two columns:
   2. task\_id: Unique identifier for each task (e.g., UUID).
   3. task\_name: Name or description of the task.

# Some Screenshot From The Project:-





