Step 1: Set Up the Backend

We'll start by setting up a Node.js server with Express and connect it to a MySQL database.

Backend Dependencies:

- **Express** for the server
- **mysql** for database connection
- **body-parser** for parsing incoming request bodies
- **cors** to enable CORS

Create a MySQL Database:

1. **Database Name**: `task_manager`
2. **Table Name**: `tasks`
3. **Table Schema**:

```
CREATE TABLE tasks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    completed BOOLEAN DEFAULT FALSE
);
```

Step 2: Set Up the Angular Frontend

We'll create an Angular project with components for listing tasks, a form for adding tasks, and buttons for deleting them.

Frontend Features:

- **TaskListComponent**: Displays tasks
- **AddTaskComponent**: Allows users to add tasks
- **Angular Material**: For styling and UI components

Step 3: Connect Frontend with Backend

We'll use Angular services to make HTTP requests to our Node.js backend.

Step-by-Step Guide

Backend Setup:

1. **Initialize a Node.js Project:**

    - Run `npm init -y` in your project directory to create a `package.json` file.
2. **Install Dependencies:**
    - Run `npm install express mysql body-parser cors`.
3. **Create Server:**
    - Set up an Express server in a file called `server.js`.

Server.js example:

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const mysql = require('mysql');

const app = express();
app.use(cors());
app.use(bodyParser.json());

const db = mysql.createConnection({
        host: 'localhost',
        user: 'yourUsername',
        password: 'yourPassword',
        database: 'task_manager'
});

db.connect(err => {
        if (err) throw err;
        console.log('MySQL connected...');
});

// API routes
app.get('/tasks', (req, res) => {
        db.query('SELECT * FROM tasks', (err, results) => {
        if (err) throw err;
        res.send(results);
        });
});

app.post('/tasks', (req, res) => {
        let task = { title: req.body.title, completed: false };
        db.query('INSERT INTO tasks SET ?', task, (err, result) => {
        if (err) throw err;
        res.send('Task added...');
        });
```

```
});

app.delete('/tasks/:id', (req, res) => {
        db.query('DELETE FROM tasks WHERE id = ?', [req.params.id], (err, result) => {
        if (err) throw err;
        res.send('Task deleted...');
        });
});

const port = 3000;
app.listen(port, () => {
        console.log(`Server running on port ${port}`);
});
```

Frontend Setup:

1. **Create Angular Project:**
    - Run `ng new task-manager-app --routing=false --style=scss`.
2. **Add Angular Material:**
    - Run `ng add @angular/material`.
3. **Generate Components:**
    - Run `ng generate component tasks` and `ng generate component add-task`.
4. **Create Service:**
    - Run `ng generate service task`.

Task Service example (task.service.ts):

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class TaskService {
  private apiUrl = 'http://localhost:3000/tasks';

  constructor(private http: HttpClient) {}

  getTasks(): Observable<any[]> {
        return this.http.get<any[]>(this.apiUrl);
```

```
  }

  addTask(title: string): Observable<any> {
        return this.http.post(this.apiUrl, { title });
  }

  deleteTask(id: number): Observable<any> {
        return this.http.delete(`${this.apiUrl}/${id}`);
  }
}
```