

## 11. Build a Task Manager Application using MongoDB, Express.js, Angular, Node.js (MEAN)

Server:

Exp11/server/app.js

```
const express = require('express');
const mongoose = require('mongoose');
const Task = require('./taskModel'); // Import the Task model
const cors = require('cors');

const app = express();
// --- Middleware ---

// 1. CORS Middleware: Allows your Angular app (running on a different port)
// to access the API
app.use(cors());

// Middleware to parse JSON request bodies
app.use(express.json());

// --- MongoDB Connection Setup ---
const MONGODB_URI = 'mongodb://127.0.0.1:27017/taskmanagerdb'; // Replace with
your actual MongoDB URI

mongoose.connect(MONGODB_URI)
  .then(() => console.log('Connected to MongoDB!'))
  .catch(err => console.error('Could not connect to MongoDB:', err));

// --- API Routes ---
app.get('/tasks', async (req, res) => {
  try {

    const tasks = await Task.find({});
    res.status(200).send(tasks);
  } catch (error) {
    // Handle server errors
    res.status(500).send(error);
  }
});

app.post('/tasks', async (req, res) => {
  try {
    // Create a new Task instance using the request body data
    const task = new Task(req.body);

    // Save the new task to the database
    await task.save();
  }
});
```

Start

free  
we

```

        // Respond with the created task and a 201 Created status
        res.status(201).send(task);
    } catch (error) {
        // Handle validation or other errors
        res.status(400).send(error);
    }
});

app.delete('/tasks/:id', async (req, res) => {
    const _id = req.params.id;

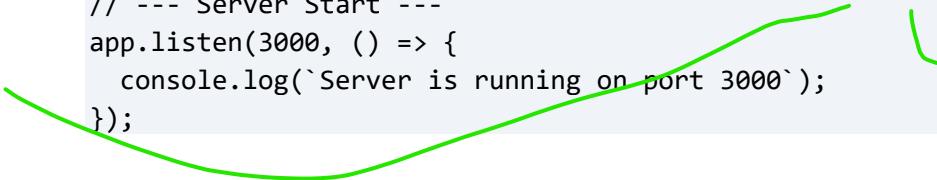
    try {
        // Find a task by its ID and delete it
        const task = await Task.findByIdAndDelete(_id);

        // Check if the task was found and deleted
        if (!task) {
            return res.status(404).send({ message: 'Task not found' });
        }

        // Respond with a success message or the deleted task
        res.send({ message: 'Task deleted successfully', deletedTask: task });
    } catch (error) {
        // Handle invalid ID format or other server errors
        res.status(500).send(error);
    }
});

// --- Server Start ---
app.listen(3000, () => {
    console.log(`Server is running on port 3000`);
});

```



## Exp11/server/taskModel.js

```

const mongoose = require('mongoose');

// Define the schema for a Task
const TaskSchema = new mongoose.Schema({
    title: {
        type: String,
        required: true,
        trim: true,
    },
    description: {
        type: String,
        trim: true,
    }
});

```

```

    },
    completed: {
      type: Boolean,
      default: false,
    },
  }, {
    timestamps: true // Adds createdAt and updatedAt fields automatically
  });

// Create and export the Task model
const Task = mongoose.model('Task', TaskSchema);
module.exports = Task;

```

**to run server: npm start // because we used nodemon package, inside package.json, give:** "start": "nodemon app.js"

```
{
  "name": "exp-11",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1",
    "start": "nodemon app.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^5.1.0",
    "mongodb": "^7.0.0",
    "mongoose": "^8.19.3",
    "nodemon": "^3.1.10"
  }
}
```

### Client:

#### Exp11/client/src/app/app.html

```

<div class="container">
  <h2 style="text-align: center;">Task Manager</h2>

  <div class="add-task-form">
    <input
      type="text"
      [(ngModel)]="newTaskTitle"

```

```

placeholder="Enter new task title"
(keyup.enter)="addTask()">
<input
type="text"
[(ngModel)]="newTaskDescription"
placeholder="Optional description"
(keyup.enter)="addTask()">
<button (click)="addTask()">Add Task</button>
</div>

<hr>

<div *ngIf="tasks.length === 0">
  <p>No tasks yet! Add one above.</p>
</div>

<ul class="task-list">
  <li *ngFor="let task of tasks" class="task-item">
    <span>{{ task.title }}</span>
    <!-- DISPLAY DESCRIPTION HERE -->
    <p style="font-size: 0.9em; color: #555;">{{ task.description }}</p>
    <button class="delete-btn"
(click)="deleteTask(task._id)">Delete</button>
  </li>
</ul>
</div>

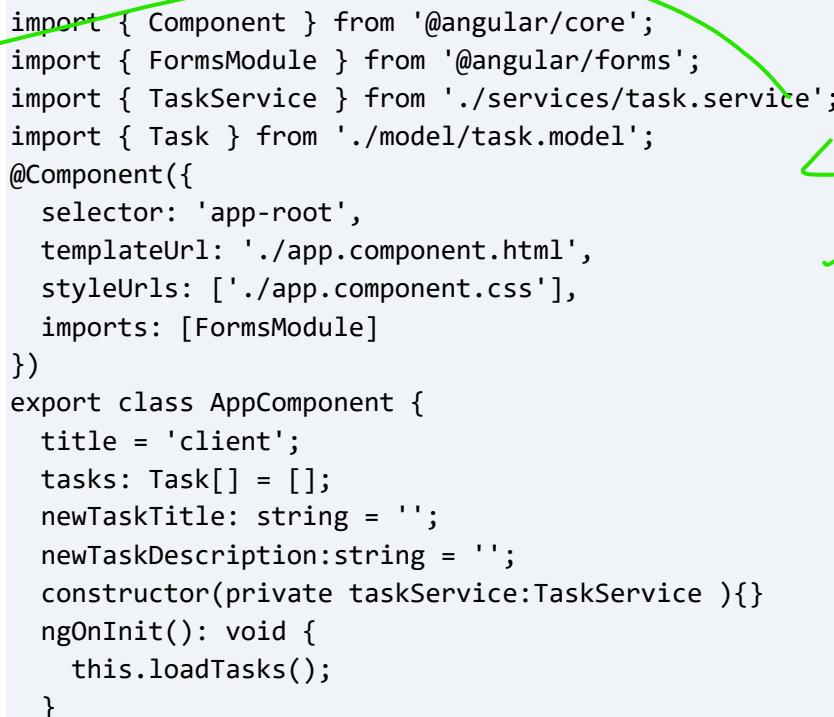
```

### Exp11/client/src/app/app.ts

```

import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { TaskService } from './services/task.service';
import { Task } from './model/task.model';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  imports: [FormsModule]
})
export class AppComponent {
  title = 'client';
  tasks: Task[] = [];
  newTaskTitle: string = '';
  newTaskDescription:string = '';
  constructor(private taskService:TaskService){}
  ngOnInit(): void {
    this.loadTasks();
  }
}

```



```

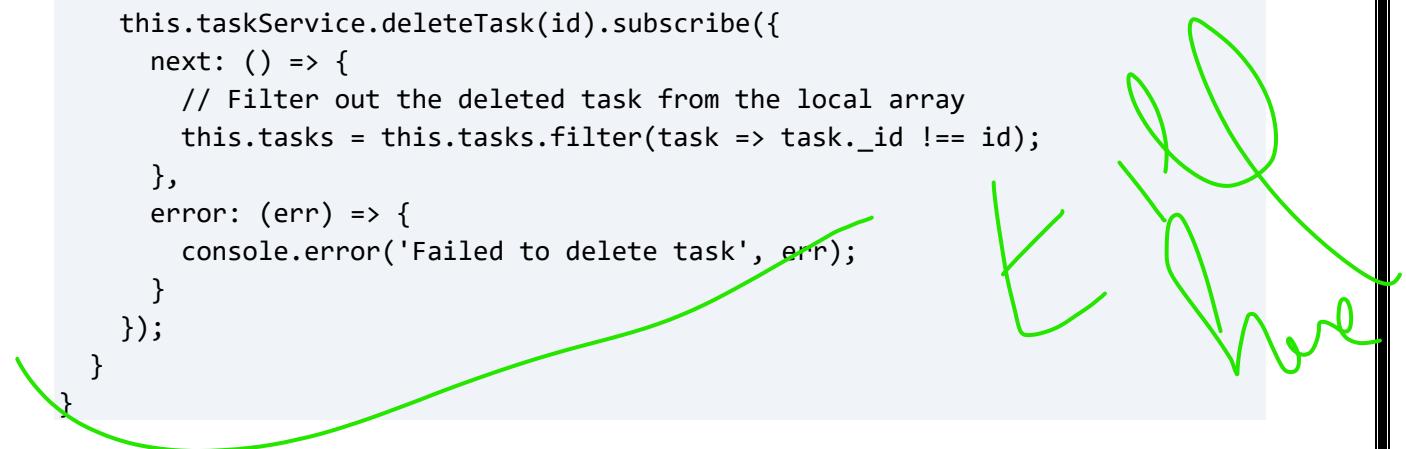
// Loads all tasks from the backend
loadTasks(): void {
    this.taskService.getTasks().subscribe({
        next: (data) => {
            this.tasks = data;
        },
        error: (err) => {
            console.error('Failed to load tasks', err);
            // Basic user feedback needed here in a real app
        }
    });
}

// Adds a new task
addTask(): void {
    if (!this.newTaskTitle.trim()) {
        return; // Prevent adding empty tasks
    }

    this.taskService.addTask(this.newTaskTitle, this.newTaskDescription).subscribe({
        next: (task) => {
            this.tasks.push(task); // Add the new task to the local array
            this.newTaskTitle = ''; // Clear the input
            this.newTaskDescription = '';
        },
        error: (err) => {
            console.error('Failed to add task', err);
        }
    });
}

// Deletes a task
deleteTask(id: string): void {
    this.taskService.deleteTask(id).subscribe({
        next: () => {
            // Filter out the deleted task from the local array
            this.tasks = this.tasks.filter(task => task._id !== id);
        },
        error: (err) => {
            console.error('Failed to delete task', err);
        }
    });
}
}

```



## Exp11/client/src/app/app.css

```
.container {
    max-width: 600px;
    margin: 50px auto;
```

```
font-family: Arial, sans-serif;

/* --- STYLES FOR THE MAIN BOX --- */
padding: 25px;
border: 1px solid #ccc; /* Light gray border around everything */
border-radius: 12px; /* Rounded corners */
box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1); /* Subtle shadow */
background-color: #ffffff; /* White background */
}

.add-task-form {
  display: flex;
  gap: 10px;
  margin-bottom: 20px; /* Added margin for separation */
}

input[type="text"] {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

input[type="text"]:first-child {
  flex-grow: 1;
}

button {
  padding: 10px 15px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}

.task-list {
  list-style-type: none;
  padding: 0;
}

/* Styles for individual task items */
.task-item {
  display: flex;
  justify-content: space-between;
```

```

    align-items: center;
    padding: 10px;
    border: 1px solid #eee; /* Added border to individual tasks */
    border-radius: 4px;
    margin-bottom: 8px;
    background-color: #f9f9f9;
}

.task-item:last-child {
    border-bottom: 1px solid #eee; /* Maintain consistency */
}

/* NEW: Styles for the checkbox and text content */
.task-content {
    display: flex;
    align-items: center;
    flex-grow: 1;
    gap: 10px;
}

.task-content input[type="checkbox"] {
    transform: scale(1.5); /* Make checkbox easier to click */
    cursor: pointer;
}

.delete-btn {
    background-color: #dc3545;
    margin-left: 15px;
    padding: 8px 12px;
}

.delete-btn:hover {
    background-color: #bd2130;
}

```

### **Exp11/client/src/app/model/task.model.ts**

```

export interface Task {
    _id: string; // MongoDB automatically adds this
    title: string;
    description?: string;
    completed: boolean;
    createdAt: Date;
}

```

### **Exp11/client/src/app/services/task.service.ts**

```

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Task } from './model/task.model';
@Injectable({
  providedIn: 'root'
})
export class TaskService {
  private apiUrl = 'http://localhost:3000/tasks'; // my Node.js API endpoint
  constructor(private http:HttpClient) { }
  // READ: Get all tasks
  // getTasks(){
  //   return this.http.get(this.apiUrl);
  // }
  // READ: Get all tasks
  getTasks(): Observable<Task[]> {
    return this.http.get<Task[]>(this.apiUrl);
  }

  // CREATE: Add a new task
  addTask(title: string,description:string): Observable<Task> {
    const newTask = { title, description }; // Only sending the title for simplicity
    return this.http.post<Task>(this.apiUrl, newTask);
  }

  // DELETE: Delete a task
  deleteTask(id: string): Observable<any> {
    return this.http.delete(` ${this.apiUrl}/${id}`);
  }
}

```

**To run client: ng serve**

**Start mongodb using the command: “mongod” in cmd**

**OUTPUT:**

The image shows a screenshot of a "Task Manager" application. At the top center, it says "Task Manager". Below that is a form with three fields: "Enter new task title", "Optional description", and a blue "Add Task" button. Underneath the form, there are two task entries, each in its own row. The first task is "homework" with the description "exp 11 complete" and a red "Delete" button. The second task is "gardening" with the description "buy plant" and a red "Delete" button.

Task Title	Description	Action
homework	exp 11 complete	Delete
gardening	buy plant	Delete