Services and Dependency Injection

1. Set Up the Project

First, create a new Angular project if you don't already have one:

```
ng new task-manager
cd task-manager
ng serve
```

---

2. Generate Components and a Service

Generate a component for the tasks:

```
ng generate component tasks
```

Generate a service for managing tasks:

```
ng generate service task
```

---

3. Create the Service

Open the `task.service.ts` file and add methods to fetch tasks.

src/app/task.service.ts

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root' // This makes the service available throughout
the app
})
export class TaskService {
  private tasks = [
```

```
    { id: 1, title: 'Learn Angular', completed: false },
    { id: 2, title: 'Build a project', completed: true },
    { id: 3, title: 'Master Dependency Injection', completed: false },
  ];

  constructor() {}

  // Get all tasks
  getTasks() {
    return this.tasks;
  }

  // Add a new task
  addTask(title: string) {
    const newTask = {
      id: this.tasks.length + 1,
      title,
      completed: false,
    };
    this.tasks.push(newTask);
  }

  // Toggle task completion
  toggleTaskCompletion(id: number) {
    const task = this.tasks.find(t => t.id === id);
    if (task) {
      task.completed = !task.completed;
    }
  }
}
```

---

4. Use the Service in a Component

Inject the service into the `TasksComponent` to use its methods.

`src/app/tasks/tasks.component.ts`

```
import { Component, OnInit } from '@angular/core';
import { TaskService } from '../task.service';

@Component({
  selector: 'app-tasks',
  templateUrl: './tasks.component.html',
  styleUrls: ['./tasks.component.css']
})
export class TasksComponent implements OnInit {
  tasks: any[] = []; // Holds the list of tasks

  constructor(private taskService: TaskService) {}

  ngOnInit(): void {
    // Fetch tasks when the component initializes
    this.tasks = this.taskService.getTasks();
  }

  addTask(title: string): void {
    if (title) {
      this.taskService.addTask(title);
      this.tasks = this.taskService.getTasks(); // Refresh the task
list
    }
  }

  toggleTask(id: number): void {
    this.taskService.toggleTaskCompletion(id);
    this.tasks = this.taskService.getTasks(); // Refresh the task list
  }
}
```

---

5. Create the Component Template

Update the `tasks.component.html` to display the tasks and allow adding/toggling tasks.

src/app/tasks/tasks.component.html

```
<div class="task-manager">
  <h1>Task Manager</h1>

  <!-- Add Task -->
  <input #taskInput type="text" placeholder="Enter new task" />
  <button (click)="addTask(taskInput.value); taskInput.value = ''">Add
Task</button>

  <!-- Task List -->
  <ul>
    <li *ngFor="let task of tasks">
      <span [class.completed]="task.completed">{{ task.title }}</span>
      <button (click)="toggleTask(task.id)">
        {{ task.completed ? 'Mark Incomplete' : 'Mark Complete' }}
      </button>
    </li>
  </ul>
</div>
```

---

6. Add Some Styling (Optional)

Update the `tasks.component.css` file for basic styling.

src/app/tasks/tasks.component.css

```
.task-manager {
  font-family: Arial, sans-serif;
  width: 300px;
  margin: 20px auto;
  text-align: center;
}

input {
  padding: 5px;
  width: 200px;
```

```
}

button {
  padding: 5px 10px;
  margin-left: 5px;
  cursor: pointer;
}

ul {
  list-style-type: none;
  padding: 0;
}

li {
  margin: 10px 0;
}

.completed {
  text-decoration: line-through;
  color: gray;
}
```

---

7. Update the Root Template

Replace the default app template (`app.component.html`) to include the `TasksComponent`.

`src/app/app.component.html`

```
<app-tasks></app-tasks>
```