**EXP-8: GETTING THE DATA FROM THE SERVER**

**1. Introduction**

This project demonstrates fetching and displaying user data from a server API using Angular 20 standalone components. It showcases important Angular concepts such as **HttpClient**, **Services**, **Dependency Injection (DI)**, **standalone component architecture**, and \**template rendering with ngFor*.

**🎯 2. Objectives**

* To create an Angular application that fetches user data from a remote server.
* To implement and demonstrate:
  + Use of **HttpClient** for HTTP requests
  + Creating and using Angular **Services** (@Injectable)
  + **Standalone components** without NgModule
  + **Dependency Injection** using constructors
  + **ngOnInit** lifecycle hook for initializing data
  + Rendering data dynamically using **\*ngFor** in templates

**💻 3. Technologies Used**

| **Technology** | **Purpose** |
| --- | --- |
| Angular (v20) | Frontend framework |
| TypeScript | Application logic and structure |
| HTML/CSS | Layout and styling |
| Bootstrap (optional) | For responsive UI |
| VS Code | Code editor/IDE |
| Angular CLI | Project scaffolding and development |

**📂 4. Project Structure**

src/

├── app/

│ ├── fetchdata.ts <-- Angular Service

│ ├── app.ts <-- Root standalone component

│ ├── app.html <-- Root component template

│ ├── app.css <-- Styling

├── main.ts <-- Application bootstrap

├── main.server.ts <-- Server bootstrap

**🧩 5. Component and Service Descriptions**

**🔹 App (Root Component)**

* Standalone component displaying user data in a table.
* Calls Fetchdata service in ngOnInit() to fetch users.
* Stores the fetched data in userList and renders it using \*ngFor.

**🔹 Fetchdata (Service)**

* Injectable service with providedIn: 'root'.
* Uses HttpClient to make a GET request to https://jsonplaceholder.typicode.com/users.
* Provides a method getUser() returning an Observable of user data.

**🔧 6. Implementation Steps**

✅ **Step 1: Create a new Angular project**

ng new fetchnew --standalone

cd fetchnew

✅ **Step 2: Create the service**

ng generate service fetchdata

✅ **Step 3: Implement the service (fetchdata.ts)**

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

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({ providedIn: 'root' })

export class Fetchdata {

constructor(private http: HttpClient) {}

getUser(): Observable<any> {

return this.http.get('https://jsonplaceholder.typicode.com/users');

}

}

✅ **Step 4: Create the root component (app.ts)**

import { Component, OnInit } from '@angular/core';

import { CommonModule } from '@angular/common';

import { Fetchdata } from './fetchdata';

@Component({

selector: 'app-root',

standalone: true,

imports: [CommonModule],

templateUrl: './app.html',

styleUrls: ['./app.css']

})

export class App implements OnInit {

userList: any[] = [];

constructor(private fetchService: Fetchdata) {}

ngOnInit(): void {

this.fetchService.getUser().subscribe({

next: (users) => this.userList = users,

error: (err) => console.error('HTTP Error:', err)

});

}

}

✅ **Step 5: Create the template (app.html)**

<h2 style="text-align:center;">User Data from Server</h2>

<table style="margin: 0 auto; border: 1px solid black; border-collapse: collapse;">

<thead>

<tr>

<th style="border: 1px solid black; padding: 10px;">Name</th>

<th style="border: 1px solid black; padding: 10px;">Id</th>

<th style="border: 1px solid black; padding: 10px;">Email</th>

<th style="border: 1px solid black; padding: 10px;">Street</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let user of userList">

<td>{{ user.name }}</td>

<td>{{ user.id }}</td>

<td>{{ user.email }}</td>

<td>{{ user.address.street }}</td>

</tr>

</tbody>

</table>

✅ **Step 6: add the styles (app.css)**

h2 {

  text-align: center;

  color: #2c3e50;

  font-family: 'Arial', sans-serif;

  margin-bottom: 20px;

}

/\* Table styling \*/

table {

  margin: 0 auto;

  border-collapse: collapse;

  width: 80%;

  font-family: 'Verdana', sans-serif;

  box-shadow: 0 4px 8px rgba(0,0,0,0.2);

}

/\* Table header \*/

th {

  border: 1px solid #2980b9;

  padding: 12px;

  background-color: #3498db;

  color: white;

  text-align: center;

}

/\* Table rows \*/

td {

  border: 1px solid #2980b9;

  padding: 12px;

  text-align: center;

  transition: background-color 0.3s;

}

/\* Zebra striping \*/

tbody tr:nth-child(even) {

  background-color: #ecf0f1;

}

tbody tr:nth-child(odd) {

  background-color: #ffffff;

}

/\* Hover effect \*/

tbody tr:hover {

  background-color: #f1c40f;

  color: #2c3e50;

  font-weight: bold;

  cursor: pointer;

}

✅ **Step 7: Bootstrap the application (main.ts)**

import 'zone.js'; // In TERMINAL npm install zone.ts

import { bootstrapApplication } from '@angular/platform-browser';

import { App } from './app/app';

import { provideHttpClient } from '@angular/common/http';

bootstrapApplication(App, {

providers: [provideHttpClient()]

});

**OUTPUT:**

