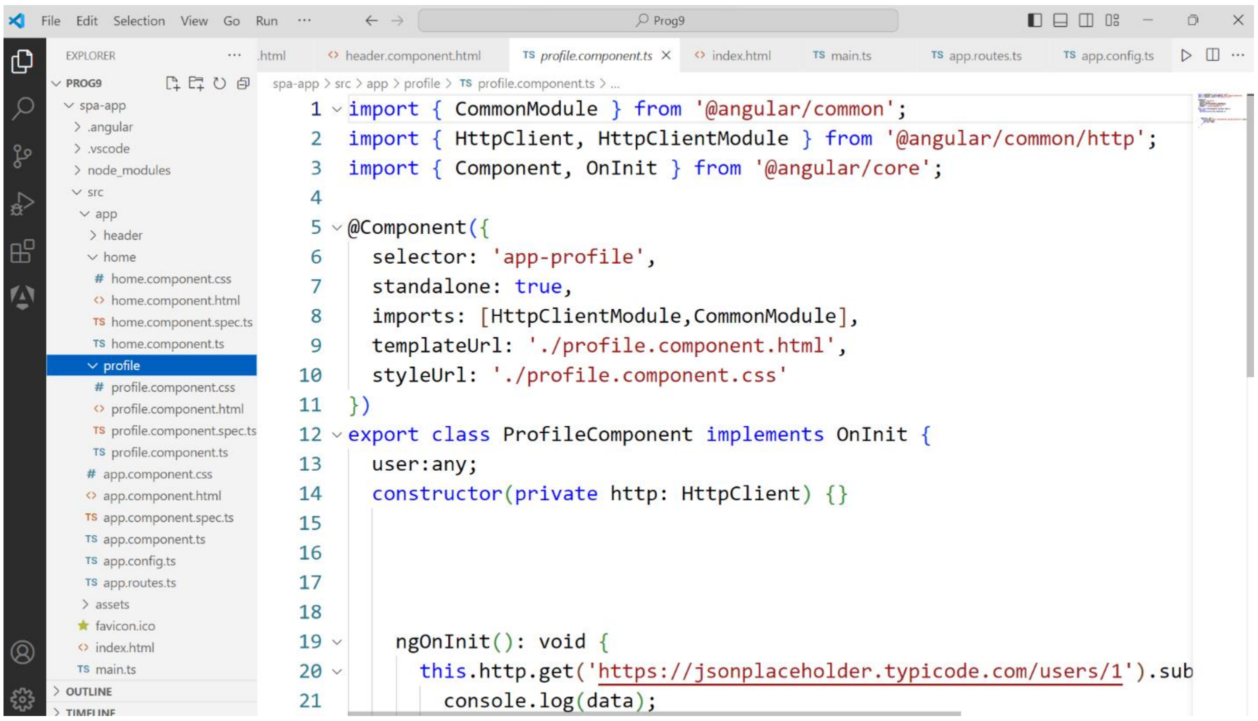
Week 10

Fetch user details from server using REST API and show in profile menu using Angular.



```
import { CommonModule } from '@angular/common';
import { HttpClient, HttpClientModule } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-profile',
  standalone: true,
  imports: [HttpClientModule,CommonModule],
  templateUrl: './profile.component.html',
  styleUrl: './profile.component.css'
})
export class ProfileComponent implements OnInit {
  user:any;
  constructor(private http: HttpClient) {}




    ngOnInit(): void {
      this.http.get('https://jsonplaceholder.typicode.com/users/1').sub
        console.log(data);
```

Note:-Install Node Js Software and Visual Studio.

Creating Angular Project Use below Commands

- npm install –g @angular/cli //creating cli
- ng version
- ng new prog9 //creating angular project SPA(Single page application) //app component is a default component.
- cd prog9
- ng g c header //creating header component
- ng g c home //creating home component
- ng g c profile //creating profile component
- ng serve //running angular project

Step 1:-
Index.html:-
Add bootstrap CDN links in the index.html

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"

crossorigin="anonymous"></script>

Step 2:-Add RouterModule to the **header.component.ts** imports

```
import { RouterModule } from '@angular/router';

imports: [RouterModule],
```

Step 3:-Add RouterModule, HeaderComponent to the **app.component.ts** imports

```
import { RouterModule, RouterOutlet } from '@angular/router';
import { HeaderComponent } from './header/header.component';

imports: [RouterOutlet,RouterModule,HeaderComponent],
```

Step 4:- Create navbar in **header.component.html**

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
    <div class="container-fluid">
      <a class="navbar-brand" [routerLink]="['/']">SPA-Demo</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page"  [routerLinkActiveOptions]="{exact:true}"
routerLinkActive="active" [routerLink]="['/']">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link"  [routerLinkActiveOptions]="{exact:true}" routerLinkActive="active"
[routerLink]="['/profile']">Profile</a>
          </li>

        </ul>
      </div>
    </div>
  </nav>
```

Step 5:-Configure route links in **app.routes.ts**

```
import { Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { ProfileComponent } from './profile/profile.component';

export const routes: Routes = [
    {
        path:'',
        component:HomeComponent,
        title:'Home'
    },
    {
        path:'profile',
        component:ProfileComponent,
        title:'Profile'
    }
];
```

Step 6:-Use header selector in the **app.component.html** along with <router-outlet>

```
<app-header></app-header>
<router-outlet></router-outlet>
```

Profile.component.html

```
<div class="card shadow-sm w-50 mt-5 p-2 mx-auto " *ngIf="user">
<img class="avatar d-block mx-auto" src= "https://cdn-icons-
png.freepik.com/512/5733/5733290.png"alt="Icon" />
    <h2 class="text-center">Hi, {{ user.name }}</h2>
    <table class="table table-bordered mt-3">
        <tr>
            <th>Username</th>
            <td>{{user.username}}</td>
        </tr>
        <tr>
            <th>Email</th>
            <td>{{user.email}}</td>
        </tr>
        <tr>
            <th>Phone</th>
            <td>{{ user.phone }}</td>
        </tr>
        <tr>
            <th>City</th>
            <td>{{ user.address?.city }}</td>
        </tr>
        <tr>
            <th>ZIP Code</th>
            <td>{{ user.address?.zipcode }}</td>
        </tr>
        <tr>
            <th>Website</th>
            <td>{{ user.website }}</td>
        </tr>
    </table>

  </div>
```

Profile.component.ts

```
import { CommonModule } from '@angular/common';
import { HttpClient, HttpClientModule } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
```

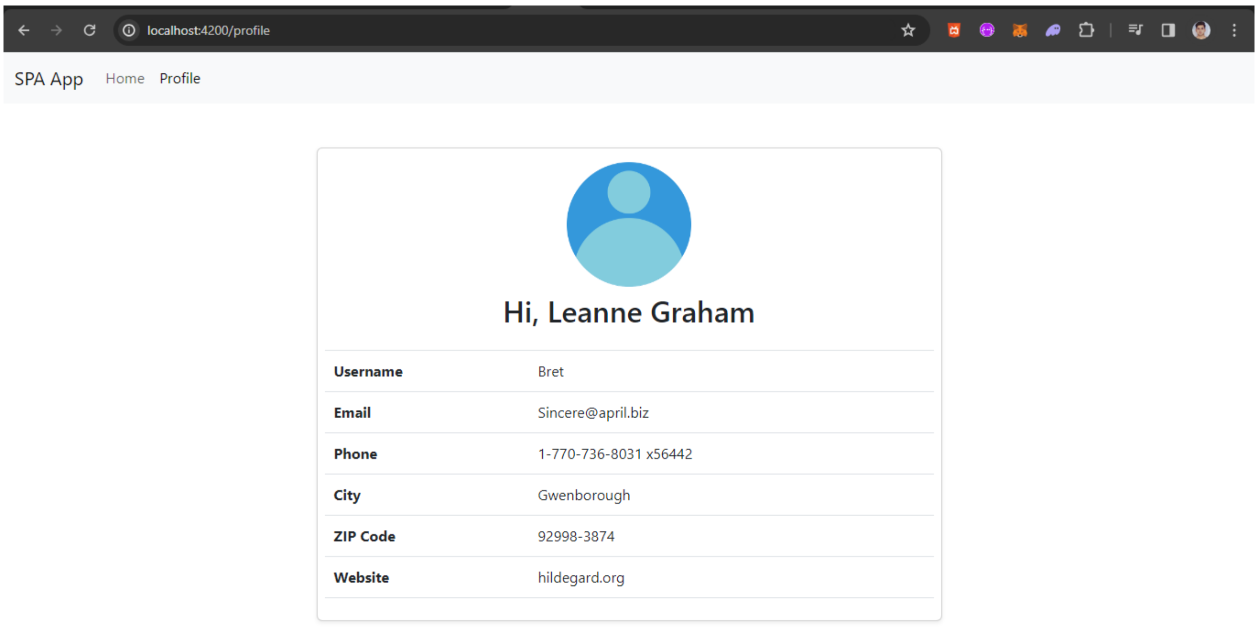
```
@Component({
  selector: 'app-profile',
  standalone: true,
  imports: [HttpClientModule,CommonModule],
  templateUrl: './profile.component.html',
  styleUrl: './profile.component.css'
})
export class ProfileComponent implements OnInit {
  user:any;
  constructor(private http: HttpClient) {}



    ngOnInit(): void {
      this.http.get('https://jsonplaceholder.typicode.com/users/1').subscribe((data: any) => {
        console.log(data);
        this.user = data;
        console.log("API Data::::::::::::",data);
      });
    }

}
```

Output:



Hi, Leanne Graham

Username	Bret
Email	Sincere@april.biz
Phone	1-770-736-8031 x56442
City	Gwenborough
ZIP Code	92998-3874
Website	hildegard.org

VIVA QUESTIONS:

1.Elaborate about URL?

Ans: A URL (Uniform Resource Locator) is a reference or address used to access resources on the internet. It provides a way to retrieve information from across the web, acting as a means to locate and specify the path to a particular piece of data or resource, such as a webpage, an image, a video file, or any other content accessible through the web. URLs are a fundamental part of the World Wide Web (WWW) and are standardized by the Internet Engineering Task Force (IETF).

2.What is the concept of statelessness in REST?
Ans: The concept of statelessness in REST (Representational State Transfer) is a foundational principle that dictates that each request from a client to a server must contain all the information the server needs to understand and fulfill the request. This means that the server does not store any state (or context) about the client session on the server side. Instead, it is the client's responsibility to keep track of the application state and send it with each request when necessary.

Examples of Statelessness
In a RESTful API, authentication data, such as tokens, are sent with each request rather than relying on a session stored on the server. This way, the server can authenticate the user without knowing anything about the user's previous interactions.

A RESTful web service might require clients to include page number and size in request parameters to fetch a specific subset of resources, rather than keeping track of the client's pagination state between requests.

3.List out the HTTP methods?
Ans:HTTP (Hypertext Transfer Protocol) defines a set of request methods to indicate the desired action to be performed for a given resource. These methods are sometimes referred to as HTTP verbs. Each has its own specific purpose and semantics in terms of interaction with web resources. Here's a list of the primary HTTP methods and their
intended use:

1.GET
2.POST
3.PUT
4.DELETE
5.PATCH
6.HEAD
7.OPTIONS
8.CONNECT
9.TRACE

4.What is reactive programming and how does it relate to angular ?

Ans: Advantages of Using Reactive Programming in Angular: Improved Performance: Reactive programming can improve the performance of Angular applications, as it enables efficient handling of events, asynchronous data, and
state management.

Modularity and Maintainability: By treating data as streams and using functional programming techniques, applications become more modular and easier to maintain and test.

Scalability: The asynchronous nature of reactive programming fits well with scalable, real-time applications, allowing Angular applications to easily scale and adapt to complex user interfaces and data flows.

5.What are the differences between REST and AJAX ?

REST (Representational State Transfer) and AJAX (Asynchronous JavaScript and XML) are both concepts used in web development, but they serve different purposes and operate at different layers of application design. Understanding the differences between them is crucial for effectively designing and implementing web applications.

REST
Conceptual Framework: REST is an architectural style or set of guidelines for designing networked applications. It is not tied to any specific protocol, but it is most commonly used over HTTP.

Statelessness: A core principle of REST is that each request from a client to a server must contain all the information the server needs to fulfill the request (statelessness). The server does not retain session state, which helps in scaling the
application.

Resource-Based: In REST, everything is considered a resource, and each resource is accessed via a common interface (URLs) and manipulated using standard HTTP methods (GET, POST, PUT, DELETE, etc.).

Use of Standard HTTP Methods: RESTful services use standard HTTP methods explicitly for their intended purposes, e.g., retrieving data with GET, submitting data to be processed to a resource using POST, updating a resource using PUT, or removing a resource with DELETE.

Data Format Agnostic: While RESTful services can use AJAX under the hood to fetch data asynchronously, they are agnostic about the format of the data returned. JSON and XML are commonly used, but HTML, plain text, and other formats can also be RESTfully served.

AJAX
Technology: AJAX is a technique for creating fast and dynamic web pages. It allows web pages to update asynchronously by exchanging small amounts of data with the server behind the scenes. This means it's possible to update parts of a web page without reloading the whole page.

Uses JavaScript: AJAX is not a technology in itself, but a term that refers to the use of a group of technologies together, including HTML or XHTML, Cascading Style Sheets (CSS), JavaScript, the Document Object Model (DOM), XML, XSLT, and the XMLHttpRequest object. It heavily relies on JavaScript to make asynchronous calls to
the server.

Focused on Client-Side: AJAX is primarily a client-side concept focusing on improving the user experience by making asynchronous calls from the web page to the server to fetch data or submit data without reloading the web page.

Partial Page Updates: The key advantage of AJAX is its ability to make partial page updates. AJAX can send data to, and retrieve data from, a server asynchronously without interfering with the display and behavior of the existing page.

Not Data Format Agnostic: AJAX typically deals with XML or JSON as the format for transferring data between the client and server, with JSON being the more popular format in modern web applications due to its lightweight nature and ease of use with JavaScript.