LP 08 Ionic 1

Ionic

- Ionic 6 is an open-source UI toolkit that includes over 100 components.
- It works with Angular, React, Vue, and other frameworks.
- It's cross-platform and has an adaptable user interface.
- The stunning open-source platform for creating hybrid mobile apps

Key Features of Ionic Framework

- Cross-Platform Development
- Integration with Angular
- Component Library
- Capacitor
- Performance
- Development Efficiency
- Community and Support

Native vs Hybrid App

Native App

- Proficiency in each platform required
- Entirely separate code bases
- Timely & expensive development

Hybrid App

- HTML5 that acts like native
- Web wrapped in the native layer
- Direct access to native APIs

Native App	Hybrid App
Native apps are platform specific	Hybrid apps are platform independent
Respective development tools	It employs HTML5, CSS3, and JavaScript
Good performance	It has restricted performance
Time consuming apps	it can be produced more quickly than native apps..
Expensive app development	Using Capacitor, we can directly access native APIs

Ionic I - CLI

- ▪ To install the CLI: npm install -g @ionic/cli
- ▪ To create a project: ionic start projectname template
- ▪ To create an angular ionic app: ionic start projectname tabs --type=angular

- ▪ To test in a browser: ionic serve
- ▪ To see all templates available: ionic start --list

Ionic 1 source code

Account-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { AccountPage } from './account.page';

const routes: Routes = [
  {
    path: '',
    component: AccountPage
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class AccountPageRoutingModule {}
```

account.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

import { IonicModule } from '@ionic/angular';

import { AccountPageRoutingModule } from './account-routing.module';

import { AccountPage } from './account.page';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    AccountPageRoutingModule
  ],
  declarations: [AccountPage]
})
export class AccountPageModule {}
```

account.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>account</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">account</ion-title>
    </ion-toolbar>
  </ion-header>
</ion-content>
```

Account.page.ts

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

@Component({
  selector: 'app-account',
  templateUrl: './account.page.html',
  styleUrls: ['./account.page.scss'],
})
export class AccountPage implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Cart-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { CartPage } from './cart.page';

const routes: Routes = [
  {
    path: '',
    component: CartPage
  }
];
```

```
@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class CartPageRoutingModule {}
```

cart.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

import { IonicModule } from '@ionic/angular';

import { CartPageRoutingModule } from './cart-routing.module';

import { CartPage } from './cart.page';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    CartPageRoutingModule
  ],
  declarations: [CartPage]
})
export class CartPageModule {}
```

cart.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>cart</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">cart</ion-title>
    </ion-toolbar>
  </ion-header>
</ion-content>
```

Cart.page.ts

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

@Component({
  selector: 'app-cart',
  templateUrl: './cart.page.html',
  styleUrls: ['./cart.page.scss'],
})
export class CartPage implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Home-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { HomePage } from './home.page';

const routes: Routes = [
  {
    path: '',
    component: HomePage
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class HomePageRoutingModule {}
```

home.module.ts

```
import { CUSTOM_ELEMENTS_SCHEMA, NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

import { IonicModule } from '@ionic/angular';

import { HomePageRoutingModule } from './home-routing.module';
```

```
import { HomePage } from './home.page';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    HomePageRoutingModule
  ],
  schemas: [CUSTOM_ELEMENTS_SCHEMA],
  declarations: [HomePage]
})
export class HomePageModule {}
```

home.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title mode="md">
      <!-- md=material design -->
     <span>Home</span>
     <ion-icon name="chevron-down-outline"></ion-icon>
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div class="borderBottom">
  <swiper-container [modules]="swiperModules"
  [slidesPerView]="1.2"
  [CenteredSlides]="true"
  [autoplay]="true"
  [pagination]="{clickable:true, dynamicBullets: true}"
  [spaceBetween]="20">

    <swiper-slide>
      <img src="assets/jollof1.jpeg"></swiper-slide>
    <swiper-slide>
      <img src="assets/Shisayanma.jpg">
    </swiper-slide>
    <swiper-slide>
      <img src="assets/Briyani.jpeg">
    </swiper-slide>

  </swiper-container>
</div>
```

```
<ion-list>
  <ion-list-header class="ion-margin-bottom">
    <ion-label>
      <h4>Restaurants Nearby</h4>
      <p>Explore exclusive flavors available near you</p>
    </ion-label>
  </ion-list-header>
  <ion-item lines="none">
    <ion-thumbnail slot="start">
      <img src="assets/jollof1.jpeg" />
    </ion-thumbnail>
    <ion-label>
      <h4>Jollof of Africa</h4>
      <ion-text color="medium">
        <p class="pStyle">
         African Cuisine
        </p>
      </ion-text>
      <span>
        5
        <ion-icon name="star"></ion-icon>
        .
      </span>
      25 mins . R100 for two
      <ion-text color="tertiary">
        <p class="distance">
          2.59 kms away
        </p>
      </ion-text>
    </ion-label>
  </ion-item>

  <ion-item lines="none">
    <ion-thumbnail slot="start">
      <img src="assets/Shisayanma.jpg" />
    </ion-thumbnail>
    <ion-label>
      <h4>Ayoba Cafe Shisanyama</h4>
      <ion-text color="medium">
        <p class="pStyle">
         African Cuisine
        </p>
      </ion-text>
      <span>
        4.4
        <ion-icon name="star"></ion-icon>
        .
```

```
      </span>
      15 mins . R120
      <ion-text color="tertiary">
        <p class="distance">
          1.83 kms away
        </p>
      </ion-text>
    </ion-label>
  </ion-item>

  <ion-item lines="none">
    <ion-thumbnail slot="start">
      <img src="assets/Briyani.jpeg" />
    </ion-thumbnail>
    <ion-label>
      <h4>Spice-The Indian Kitchen</h4>
      <ion-text color="medium">
        <p class="pStyle">
         Asian Cuisine
        </p>
      </ion-text>
      <span>
        4.1
        <ion-icon name="star"></ion-icon>
        .
      </span>
      5 mins . R80
      <ion-text color="tertiary">
        <p class="distance">
          0.9 km away
        </p>
      </ion-text>
    </ion-label>
  </ion-item>
</ion-list>
</ion-content>
```

Home.page.css

```
ion-header {
    ion-icon{
        font-size:1rem;
    }
}
div {
    height: 30vh;
    swiper-container{
```

```
        swiper-slide{
            height: 25vh;
            img {
                width: 100%;
                height: 20vh;
                border-radius: 5px;
            }
        }
    }
}
.borderBottom {
    border-bottom: 1.5vh solid var(--ion-color-light);
}

ion-label {
    font-size: 0.9rem;
    font-weight: bold;
    h4 {
        font-weight: bold !important;
        font-size: 1.2rem;
    }
    p {
        font-size: 0.7rem;
    }
}

ion-item {
    font-size: 0.8rem;
    ion-thumbnail {
        height: 16vh;
        width: 25vw;
        img {
            border-radius: 5px;
            height: 100%;
            width: 100%;
        }
    }
    h4 {
        font-size: 1rem;
    }
}
.distance {
    font-size: 0.8rem;
    padding-top: 0.2vh;
}

.pStyle {
    padding-bottom: 0.5vh;
```

```
    font-size: 0.82rem !important;
}
```

Home.page.ts

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

@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
  styleUrls: ['./home.page.scss'],
})
export class HomePage implements OnInit {

  swiperModules = [IonicSlides];
  constructor() { }

  ngOnInit() {
  }

}
```

Search-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { SearchPage } from './search.page';

const routes: Routes = [
  {
    path: '',
    component: SearchPage
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class SearchPageRoutingModule {}
```

search.module,ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
```

```
import { FormsModule } from '@angular/forms';

import { IonicModule } from '@ionic/angular';

import { SearchPageRoutingModule } from './search-routing.module';

import { SearchPage } from './search.page';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    SearchPageRoutingModule
  ],
  declarations: [SearchPage]
})
export class SearchPageModule {}
```

search.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>search</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">search</ion-title>
    </ion-toolbar>
  </ion-header>
</ion-content>
```

Search.page.ts

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

@Component({
  selector: 'app-search',
  templateUrl: './search.page.html',
  styleUrls: ['./search.page.scss'],
})
export class SearchPage implements OnInit {
```

```
  constructor() { }

  ngOnInit() {
  }

}
```

Tabs-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { TabsPage } from './tabs.page';

const routes: Routes = [
  {
    path: '',
    component: TabsPage,
    children: [
    {
      path: 'home',
      loadChildren: () => import('./home/home.module').then( m =>
m.HomePageModule)
    },
    {
      path: 'cart',
      loadChildren: () => import('./cart/cart.module').then( m =>
m.CartPageModule)
    },
    {
      path: 'account',
      loadChildren: () => import('./account/account.module').then( m =>
m.AccountPageModule)
    },
    {
      path: 'search',
      loadChildren: () => import('./search/search.module').then( m =>
m.SearchPageModule)
    }
    ]
  },
  {
    path: '',
    redirectTo: '/tabs/home',
    pathMatch: 'full'
  },

];
```

```
@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class TabsPageRoutingModule {}
```

tabs.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

import { IonicModule } from '@ionic/angular';

import { TabsPageRoutingModule } from './tabs-routing.module';

import { TabsPage } from './tabs.page';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    TabsPageRoutingModule
  ],
  declarations: [TabsPage]
})
export class TabsPageModule {}
```

tabs.page.html

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="home">
    <ion-icon name="home-outline"></ion-icon>
      Home
    </ion-tab-button>
    <ion-tab-button tab="cart">
      <!-- <ion-badge>6</ion-badge> -->
      <ion-icon name="cart-outline"></ion-icon>
      Cart
    </ion-tab-button>
    <ion-tab-button tab="account">
      <ion-icon name="person-outline"></ion-icon>
      Account
    </ion-tab-button>
    <ion-tab-button tab="search">
```

```
      <ion-icon name="search-outline"></ion-icon>
      Search
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

Tabs.page.ts

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

@Component({
  selector: 'app-tabs',
  templateUrl: './tabs.page.html',
  styleUrls: ['./tabs.page.scss'],
})
export class TabsPage implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

App-routing.module.ts

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [

  {
    path: '',
    redirectTo: 'tabs',
    pathMatch: 'full'
  },
  {
    path: 'tabs',
    loadChildren: () => import('./pages/tabs/tabs.module').then( m =>
m.TabsPageModule)
  },
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
```

```
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

LP 09: Ionic II

Install & Configure

- Node.js (website)
- npm install -g @ionic/cli
- ionic start yourprojectname tabs --type=angular
- ionic serve -o

Ionic Framework Components Implementation

- ion-avatar
- ion-card
- ion-button | ion-fab-button
- ion-list
- ion-skeleton-text
- ion-refresher
- Modal
- Popover

Ionic II source code

API

Herocontroller.cs

```
using Ionic_II.Models;
using Microsoft.AspNetCore.Components.Forms;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Reflection.Metadata.Ecma335;

namespace Ionic_II.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class HeroController : ControllerBase
    {
        private readonly IHeroRepository _heroRepository;

        public HeroController(IHeroRepository HeroRepository)
        {
            _heroRepository = HeroRepository;
        }

        [HttpGet]
        [Route("GetAllHeroes")]
        public async Task<IActionResult> GetAllHeroes()
        {
            try
```

```
            {
                var results = await _heroRepository.GetAllHeroesAsync();
                return Ok(results);
            }
            catch (Exception)
            {
                return StatusCode(500,"Internal Server Error. Please contact
support.");
            }
        }

        [HttpGet]
        [Route("GetHero/{heroId}")]
        public async Task<IActionResult> GetHero(int heroId)
        {
            try
            {
                var result = await _heroRepository.GetHeroAsync(heroId);

                if (result == null) return NotFound("Hero does not exist. You
need to create it first");

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(500, "Internal Server Error. Please contact
support");
            }
        }
    }
}
```

AppDBContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace Ionic_II.Models
{
    public class AppDbContext:DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options): base
(options)
        {
        }

        public DbSet<Hero> Heroes { get; set; }
```

```
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            // Hero

            modelBuilder.Entity<Hero>()
                .HasData(
                new
                {
                    HeroId = 1,
                    Name = "Samuel Thomas 'Sam' Wilson (Falcon/Captain
America)",
                    Age = 40,
                    Birthday = "September 23",
                    Height = "178cm",
                    isAlive = true,
                    FileName = "Anthony-Mackie-Captain-America-4.webp",
                }
            );

            modelBuilder.Entity<Hero>()
                .HasData(
                new
                {
                    HeroId = 2,
                    Name = "Scott Lang (Ant-Man)",
                    Age = 34,
                    Birthday = "",
                    Height = "178cm",
                    isAlive = true,
                    FileName = "antman-and-the-wasp-marvel-4.webp",
                }
             );
            modelBuilder.Entity<Hero>()
                .HasData(
                new
                {
                    HeroId = 3,
                    Name = "Natasha Romanoff (Black Widow)",
                    Age = 39,
                    Birthday = "December 3",
                    Height = "164cm",
                    isAlive = true,
                    FileName = "black-widow-1.webp",
                }
            );
            modelBuilder.Entity<Hero>()
```

```
            .HasData(
            new
            {
                HeroId = 4,
                Name = "Colonel James 'Rhodey' Rhodes (War Machine)",
                Age = 55,
                Birthday = "October 6",
                Height = "173cm",
                isAlive = true,
                FileName = "Don-cheadle-as-rhodey-Cropped.webp",
            }
        );
        modelBuilder.Entity<Hero>()
            .HasData(
            new
            {
                HeroId = 5,
                Name = "Clint Barton (Hawkeye)",
                Age = 53,
                Birthday = "June 18",
                Height = "173cm",
                isAlive = true,
                FileName = "hawkeye.webp",
            }
        );
        modelBuilder.Entity<Hero>()
            .HasData(
            new
            {
                HeroId = 6,
                Name = "Peter Parker (Spider-Man)",
                Age = 19,
                Birthday = "August 10",
                Height = "170cm",
                isAlive = true,
                FileName = "peter-parker-Cropped.webp",
            }
        );
        modelBuilder.Entity<Hero>()
            .HasData(
            new
            {
                HeroId = 7,
                Name = "Steve Rogers (Captain America)",
                Age = 34,
                Birthday = "July 4",
                Height = "185cm",
                isAlive = false,
```

```
            FileName = "steve-rogers.webp",
        }
    );
    modelBuilder.Entity<Hero>()
        .HasData(
        new
        {
            HeroId = 8,
            Name = "Bruce Banner (The Hulk)",
            Age = 54,
            Birthday = "December 18",
            Height  =  "250cm",
            isAlive = true,
            FileName = "The-Incredible-Hulk.webp",
        }
    );
    modelBuilder.Entity<Hero>()
        .HasData(
        new
        {
            HeroId = 9,
            Name = "Thor",
            Age = 1059,
            Birthday = "",
            Height = "192cm",
            isAlive = true,
            FileName = "thor-lightning.webp",
        }
    );
    modelBuilder.Entity<Hero>()
        .HasData(
        new
        {
            HeroId = 10,
            Name = "Tony Stark (Iron Man)",
            Age = 53,
            Birthday = "May 29",
            Height = "185cm",
            isAlive = false,
            FileName = "tony-stark-iron-man.webp",
        }
    );
    modelBuilder.Entity<Hero>()
        .HasData(
        new
        {
            HeroId = 11,
            Name = "Vision",
```

```
                    Age = 3,
                    Birthday = "May 29",
                    Height = "178cm",
                    isAlive = true,
                    FileName = "Vision-Civil-War-Cropped.webp",
                }

            );
            modelBuilder.Entity<Hero>()
                .HasData(
                new
                {
                    HeroId = 12,
                    Name = "Wanda Maximoff (Scarlet Witch)",
                    Age = 30,
                    Birthday = "February 10",
                    Height  =  "168cm",
                    isAlive = true,
                    FileName = "Wanda-Scarlet-Witch-Cropped.webp",
                    Imagbase64
                }
            );
        }
    }
}
```

Hero.cs

```
using System.ComponentModel.DataAnnotations;

namespace Ionic_II.Models
{
    public class Hero
    {
        [Key]
        public int HeroId { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }
        public string Birthday { get; set; }
        public string Height { get; set; }
        public bool isAlive { get; set; }
        public string FileName { get; set; }
        public string ImageBase64 { get; set; }

    }
}
```

Herorepository.cs

```
using Microsoft.EntityFrameworkCore;

namespace Ionic_II.Models
{
    public class HeroRepository : IHeroRepository
    {
        private readonly AppDbContext _appDbContext;

        public HeroRepository(AppDbContext appDbContext)
        {
                _appDbContext = appDbContext;
        }
        public async Task<Hero[]> GetAllHeroesAsync()
        {
            IQueryable<Hero> query = _appDbContext.Heroes;
            return await query.ToArrayAsync();
        }

        public async Task<Hero> GetHeroAsync(int heroId)
        {
            IQueryable<Hero> query = _appDbContext.Heroes.Where(c => c.HeroId
== heroId);
            return await query.FirstOrDefaultAsync();
        }
    }
}
```

IHerorepository.cs

```
namespace Ionic_II.Models
{
    public interface IHeroRepository
    {
        // Hero
        Task<Hero[]> GetAllHeroesAsync();

        Task<Hero> GetHeroAsync(int heroId);

    }
}
```

FRONTEND

Hero-detail.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button></ion-back-button>
    </ion-buttons>
    <ion-title></ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-img src={{heroDetail?.imageBase64}}></ion-img>
  <ion-card no-margin>
    <ion-card-header>
    <ion-card-title>
      {{heroDetail?.name}}
    </ion-card-title>
  </ion-card-header>
  </ion-card>

  <ion-card>
    <ion-list lines="none">

      <ion-item>
        <ion-label>Age</ion-label>
        <ion-chip color="primary">{{heroDetail?.age}}</ion-chip>
      </ion-item>

      <ion-item>
        <ion-label>Height</ion-label>
        <ion-chip color="secondary">{{heroDetail?.height}}</ion-chip>
      </ion-item>

      <ion-item>
        <ion-label>Birthday</ion-label>
        <ion-chip>{{heroDetail?.birthday}}</ion-chip>
      </ion-item>

    </ion-list>
  </ion-card>

  <ion-fab vertical="top" horizontal="end" slot="fixed">
    <ion-fab-button (click)="openModal(heroDetail?.isAlive)">
      <ion-icon name="eye"></ion-icon>
    </ion-fab-button>
  </ion-fab>
```

```
</ion-content>
```

Hero-detail.page.ts

```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { IonicModule, ModalController, ToastController } from
'@ionic/angular';
import { HeroService } from '../services/hero.service';
import { ActivatedRoute } from '@angular/router';
import { HerostatusPage } from '../herostatus/herostatus.page';

@Component({
  selector: 'app-hero-detail',
  templateUrl: './hero-detail.page.html',
  styleUrls: ['./hero-detail.page.scss'],
  standalone: true,
  imports: [IonicModule, CommonModule, FormsModule]
})
export class HeroDetailPage implements OnInit {
  heroDetail:any
  constructor(private _toastController: ToastController, private _heroService:
HeroService, private _modal:ModalController, private route:ActivatedRoute) {

    this._heroService.getHero(+this.route.snapshot.params['heroId']).subscribe
(result => {

      this.heroDetail = result
      const toast = this._toastController.create({
        message: "Hero " + this.heroDetail.name + " is viewable",
        duration: 3000,
        position:"bottom"
      })

      toast.then((toastMessage) => {
        toastMessage.present();
      })
    })
  }

  ngOnInit():void {  }

  async openModal(status:boolean)
  {
    const statusModal = await this._modal.create({
      component: HerostatusPage,
      componentProps:{
```

```
        value:status
      }
    })

    return await statusModal.present()
  }
}
```

Hero-status.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>Hero Status</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-fab vertical="center" horizontal="center">
    <ion-fab-button *ngIf="value">
      <ion-icon name="happy-outline" *ngIf="value"> </ion-icon>
      Alive
    </ion-fab-button>
    <ion-fab-button *ngIf="!value">
      <ion-icon name="sad-outline" *ngIf="!value"></ion-icon>
      Dead
    </ion-fab-button>
  </ion-fab>

 <ion-button (click)="closeModal()" class="container" color="danger">
  <ion-icon name="close-circle-outline" size="large"></ion-icon>
 </ion-button>
</ion-content>
```

Herostatus.page.css

```
.container {
    position: absolute;
    bottom: 0;
    left: 0;
    right: 0;
    display: flex;
    justify-content: center;
    align-items: flex-end;
    height: 5vh;
}
```

Herostatus.page.ts

```
import { Component, Input, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { IonicModule, ModalController } from '@ionic/angular';

@Component({
  selector: 'app-herostatus',
  templateUrl: './herostatus.page.html',
  styleUrls: ['./herostatus.page.scss'],
  standalone: true,
  imports: [IonicModule, CommonModule, FormsModule]
})
export class HerostatusPage implements OnInit {
  @Input() value: any;
  constructor(private _modal: ModalController) { }

  ngOnInit() {

  }

  closeModal(){
    this._modal.dismiss()
  }

}
```

Popover.component.html

```
<ion-content>
  <ion-list>
    <ion-list-header>
      Test Menu List
    </ion-list-header>
    <ion-item>
      <ion-label>Menu-Item 1</ion-label>
    </ion-item>
    <ion-item>
      <ion-label>Menu-Item 2</ion-label>
    </ion-item>
    <ion-item>
      <ion-button (click)="displayMessage()">
        Popover Call
        <ion-icon name="happy-outline"> </ion-icon>
      </ion-button>
    </ion-item>
  </ion-list>
</ion-content>
```

Popover.component.ts

```
import { Component, OnInit} from '@angular/core';
import { Router } from '@angular/router';
import { IonicModule, PopoverController } from '@ionic/angular';

@Component({
  selector: 'app-popover',
  templateUrl: './popover.component.html',
  styleUrls: ['./popover.component.scss'],
  standalone: true,
  imports:[IonicModule]
})
export class PopoverComponent  implements OnInit {

  constructor(private route:Router, public popoverController:
PopoverController) { }

  ngOnInit() {}

  displayMessage()
  {
    alert("Popover call successful")
    this.popoverController.dismiss()
    this.route.navigate([""])
  }

}
```

Hero.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { map, Observable, Subject } from 'rxjs';
import { Hero } from '../shared/hero';

@Injectable({
  providedIn: 'root'
})
export class HeroService {

  apiUrl = 'http://localhost:5116/api/'

  httpOptions ={
    headers: new HttpHeaders({
      ContentType: 'application/json'
    })
  }
```

```
  constructor(private _httpClient: HttpClient) { }

  getHeroes(): Observable<Hero[]>{
    return this._httpClient.get<Hero[]>(`${this.apiUrl}Hero/GetAllHeroes`)
    .pipe(map(result => result))
  }

  getHero(heroId: number) {
    return this._httpClient.get(`${this.apiUrl}Hero/GetHero` + "/" + heroId)
    .pipe(map(result => result))
  }
}
```

Hero.ts

```
export interface Hero {
    heroId: Number;
    name:String;
    age:number;
    birthday:String;
    height:string;
    isAlive:boolean;
    filename:String;
    imageBase64:String;
}
```

Tab1.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Home
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="!Heroes">
  <ion-card>
    <ion-skeleton-text style="height:200px;" animated></ion-skeleton-text>
    <ion-card-header></ion-card-header>
  </ion-card>

  <ion-card>
    <ion-skeleton-text style="height:200px;" animated></ion-skeleton-text>
    <ion-card-header></ion-card-header>
```

```
  </ion-card>

  <ion-card>
    <ion-skeleton-text style="height:200px;" animated></ion-skeleton-text>
    <ion-card-header></ion-card-header>
  </ion-card>

  <ion-card>
    <ion-skeleton-text style="height:200px;" animated></ion-skeleton-text>
    <ion-card-header>

    </ion-card-header>
  </ion-card>
</div>

<ion-refresher slot="fixed" (ionRefresh)="refreshHeroes($event)">
<ion-refresher-content refreshingText="Loading Heroes..."></ion-refresher-
content>
</ion-refresher>

  <ion-card button *ngFor="let hero of (Heroes | async)" [routerLink]="['hero-
detail', hero.heroId]">
    <ion-img [src]="hero.imageBase64"></ion-img>
  </ion-card>
</ion-content>
```

Tab1.page.ts

```
import { Component } from '@angular/core';
import { IonicModule, ToastController } from '@ionic/angular';
import { ExploreContainerComponent } from '../explore-container/explore-
container.component';
import { Hero } from '../shared/hero';
import { HeroService } from '../services/hero.service';
import { AppModule } from '../app.module';
import { CommonModule } from '@angular/common';
import { Observable } from 'rxjs';
import { RouterLink } from '@angular/router';

@Component({
  selector: 'app-tab1',
  templateUrl: 'tab1.page.html',
  styleUrls: ['tab1.page.scss'],
  standalone: true,
  imports: [IonicModule, ExploreContainerComponent, AppModule, CommonModule,
RouterLink],
})
```

```
export class Tab1Page {
  Heroes: Observable<Hero[]>;
  constructor(private _toastController: ToastController, private _heroService:
HeroService) {
    this.Heroes = this._heroService.getHeroes();
  }

  ngOnInit() {
  }

  refreshHeroes(event:any){
    this.Heroes = this._heroService.getHeroes();

    event.target.complete();


      const toast = this._toastController.create({
        message: "Heroes  are  refreshed",
        duration: 3000,
        position:"bottom"
      })

      toast.then((toastMessage) => {
        toastMessage.present();
      })
  }
}
```

Tab2.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Popover Example
    </ion-title>
    <ion-buttons slot="end">
      <ion-button (click)="onPopoverButtonClicked($event)"
menu="notifications">
        <ion-icon name="ellipsis-horizontal"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-header collapse="condense">
    <ion-toolbar>
```

```
      <ion-title size="large">Tab 2</ion-title>
    </ion-toolbar>
  </ion-header>
</ion-content>
```

Tab2.page.ts

```
import { Component } from '@angular/core';
import { IonicModule, PopoverController } from '@ionic/angular';
import { ExploreContainerComponent } from '../explore-container/explore-
container.component';
import { PopoverComponent } from '../popover/popover.component';

@Component({
  selector: 'app-tab2',
  templateUrl: 'tab2.page.html',
  styleUrls: ['tab2.page.scss'],
  standalone: true,
  imports: [IonicModule, ExploreContainerComponent]
})
export class Tab2Page {

  constructor(private _popoverController: PopoverController) {}

  async onPopoverButtonClicked(clickEvent:any)
  {
      const _popover = await this._popoverController.create({
        event: clickEvent,
        component: PopoverComponent
      })

      await _popover.present()

    }
}
```

Tab3.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Profile
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
```

```
  <ion-avatar class="image-align">
    <ion-img src="assets/images/profile/profile.jpg"></ion-img>
  </ion-avatar>
  <ion-card>
    <ion-card-header>
      <ion-card-title>
        Tony Stark
      </ion-card-title>
    </ion-card-header>
    <ion-card-content>
      I am Iron Man.
    </ion-card-content>
  </ion-card>
</ion-content>
```

Tab3.page.ts

```
import { Component } from '@angular/core';
import { IonicModule } from '@ionic/angular';
import { ExploreContainerComponent } from '../explore-container/explore-
container.component';

@Component({
  selector: 'app-tab3',
  templateUrl: 'tab3.page.html',
  styleUrls: ['tab3.page.scss'],
  standalone: true,
  imports: [IonicModule, ExploreContainerComponent],
})
export class Tab3Page {
  constructor() {}
}
```

Tabs.page.html

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon aria-hidden="true" name="people-sharp"></ion-icon>
      <ion-label>Heroes</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab2">
      <ion-icon aria-hidden="true" name="information-circle-sharp"></ion-icon>
      <ion-label>Popover</ion-label>
    </ion-tab-button>
```

```
    <ion-tab-button tab="tab3">
      <ion-icon aria-hidden="true" name="person-sharp"></ion-icon>
      <ion-label>Profile</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

Tabs.page.ts

```
import { Component, EnvironmentInjector, inject } from '@angular/core';
import { IonicModule } from '@ionic/angular';

@Component({
  selector: 'app-tabs',
  templateUrl: 'tabs.page.html',
  styleUrls: ['tabs.page.scss'],
  standalone: true,
  imports: [IonicModule],
})
export class TabsPage {
  public environmentInjector = inject(EnvironmentInjector);

  constructor() {}
}
```

Tabs.route.ts

```
import { Routes } from '@angular/router';
import { TabsPage } from './tabs.page';

export const routes: Routes = [
  {
    path: 'tabs',
    component: TabsPage,
    children: [
      {
        path: 'tab1',
        loadChildren: () => [
          {
          path: '',
          loadComponent: () =>
          import('../tab1/tab1.page').then((m) => m.Tab1Page),
          },
          {
            path: 'hero-detail/:heroId',
```

```
            loadComponent: () => import('../hero-detail/hero-
detail.page').then( m => m.HeroDetailPage)
          },
        ]
      },
      {
        path: 'tab2',
        loadComponent: () =>
          import('../tab2/tab2.page').then((m) => m.Tab2Page),
      },
      {
        path: 'tab3',
        loadComponent: () =>
          import('../tab3/tab3.page').then((m) => m.Tab3Page),
      },
      {
        path: '',
        redirectTo: '/tabs/tab1',
        pathMatch: 'full',
      },
    ],
  },
  {
    path: '',
    redirectTo: '/tabs/tab1',
    pathMatch: 'full',
  },
];
```

App.routes.ts

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

export const routes: Routes = [
  {
    path: '',
    loadChildren: () => import('./tabs/tabs.routes').then((m) => m.routes),
  },
  // {
  //   path: 'herostatus',
  //   loadComponent: () => import('./herostatus/herostatus.page').then( m =>
m.HerostatusPage)
  // },
  // {
  //   path: 'hero-detail',
  //   loadComponent: () => import('./hero-detail/hero-detail.page').then( m
=> m.HeroDetailPage)
  // },
];
```

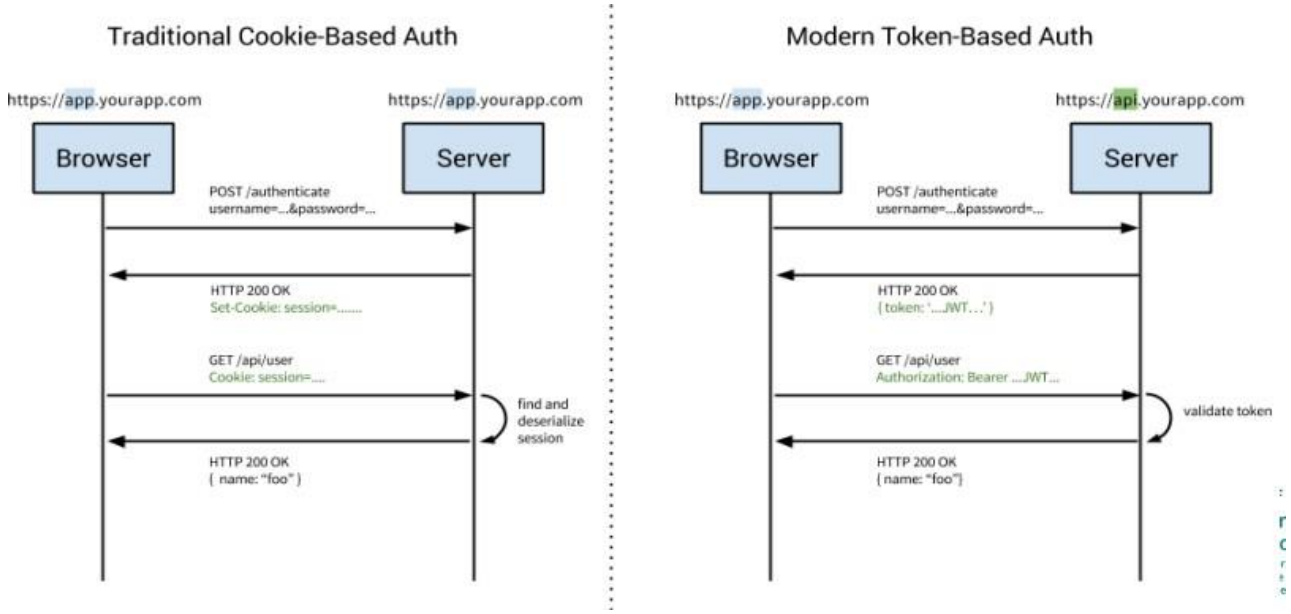
LP 10: Application Security

Theory

- Authenticate (check to see if the person exist)
- Authorize (check what authorization (e.g. role) the user has)
- Audit (log what the user did) user access to the things he/she was limited to by, e.g. Administrator.
- Without any sort of authentication and authorization, anyone with the link to the API can access the data and get private company information like customer details and can cause a lot of problems.
- This is not only to protect your own resources, but the confidentiality of your end users as well.

The most common response codes:

- 200 OK (Means everything is successful and the resources can be fetched)
- 404 Not Found (The server can not find the requested resources)
- 403 Forbidden (This means you don't have the access right to the content)
- 401 Unauthorized (Means you are not authenticated to request the resource)
- 400 Bad Request (Client Issue – server cannot/will not process the request)
- 500 Internal Server Error (Something has gone wrong on the server)



Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6
IkpXVCJ9.eyJzdWIiOiIxMjM0NTY
3ODkwIiwibmFtZSI6IkpvaG4gRG9
lIiwiYWRtaW4iOnRydWV9.TJVA95
OrM7E2cBab30RMHrHDcEfxjoYZge
FONFh7HgQ
```

Decoded

HEADER:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) ☐secret base64 encoded
```

LP 10: Application Security Source Code

Backend API

Coursecontroller.cs

```
using ApplicationSecurity_Backend.Models;
using ApplicationSecurity_Backend.ViewModels;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace ApplicationSecurity_Backend.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CourseController : ControllerBase
    {
        private readonly UserManager<AppUser> _userManager;
        private readonly RoleManager<IdentityRole> _roleManager;
        private readonly IRepository _repository;
        private readonly IUserClaimsPrincipalFactory<AppUser>
_claimsPrincipalFactory;
        private readonly IConfiguration _configuration;

        public CourseController(UserManager<AppUser> userManager,
RoleManager<IdentityRole> roleManager, IUserClaimsPrincipalFactory<AppUser>
claimsPrincipalFactory, IConfiguration configuration, IRepository repository)
        {
            _userManager = userManager;
            _roleManager = roleManager;
            _claimsPrincipalFactory = claimsPrincipalFactory;
            _configuration = configuration;
            _repository = repository;
        }

        [HttpGet]
        [Route("GetAllCourses")]
        [Authorize(AuthenticationSchemes =
JwtBearerDefaults.AuthenticationScheme)]
        public async Task<IActionResult> GetAllCoursesAsync()
        {
            try
```

```
            {
                var results = await _repository.GetAllCoursesAsync();
                return Ok(results);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError,
"Internal Server Error, please contact support");
            }
        }

        [HttpPost]
        [Route("Register")]
        public async Task<IActionResult> Register(UserViewModel uvm)
        {
            var user = await _userManager.FindByIdAsync(uvm.emailaddress);

            if (user == null)
            {
                user = new AppUser
                {
                    Id = Guid.NewGuid().ToString(),
                    UserName = uvm.emailaddress,
                    Email = uvm.emailaddress
                };

                var result = await _userManager.CreateAsync(user,
uvm.password);

                if (result.Errors.Count() > 0) return
StatusCode(StatusCodes.Status500InternalServerError, "Internal Server Error.
Please contact support.");
            }
            else
            {
                return Forbid("Account already exists.");
            }

            return Ok();
        }

        [HttpPost]
        [Route("Login")]
        public async Task<ActionResult> Login(UserViewModel uvm)
        {
            var user = await _userManager.FindByNameAsync(uvm.emailaddress);
```

```
            if (user != null && await _userManager.CheckPasswordAsync(user,
uvm.password))
            {
                try
                {
                    return await GenerateJWTToken(user);
                }
                catch (Exception)
                {
                    return
StatusCode(StatusCodes.Status500InternalServerError, "Internal Server Error.
Please contact support.");
                }
            }
            else
            {
                return NotFound("Does not exist");
            }
        }

        [HttpGet]
        private async Task<ActionResult> GenerateJWTToken(AppUser user)
        {
            var role = await _userManager.GetRolesAsync(user);
            IdentityOptions _identityOptions = new IdentityOptions();
            // Create JWT Token
            var claims = new List<Claim>
            {
                new Claim(JwtRegisteredClaimNames.Sub, user.Email),
                new Claim(JwtRegisteredClaimNames.Jti,
Guid.NewGuid().ToString()),
                new Claim(JwtRegisteredClaimNames.UniqueName, user.UserName),

            };

            if (role.Count() > 0)
            {
                claims.Add(new
Claim(_identityOptions.ClaimsIdentity.RoleClaimType, role.FirstOrDefault()));
            }

            var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Tokens:Key"]));
            var credentials = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                _configuration["Tokens:Issuer"],
```

```
                _configuration["Tokens:Audience"],
                claims,
                signingCredentials: credentials,
                expires: DateTime.UtcNow.AddHours(3)
            );

            return Created("", new
            {
                token = new JwtSecurityTokenHandler().WriteToken(token),
                user = user.UserName
            });
        }

        [HttpPost]
        [Route("CreateRole")]
        public async Task<IActionResult> CreateRole(string roleName)
        {
            var role = await _roleManager.FindByNameAsync(roleName);
            if (role == null)
            {
                role = new IdentityRole
                {
                    Id = Guid.NewGuid().ToString(),
                    Name = roleName
                };

                var result = await _roleManager.CreateAsync(role);

                if (result.Errors.Count() > 0) return
BadRequest(result.Errors);
            }
            else
            {
                return Forbid("Role already exists.");
            }

            return Ok();
        }

        [HttpPost]
        [Route("AssignRole")]
        public async Task<IActionResult> AssignRole(string emailAddress,
string roleName)
        {
            var user = await _userManager.FindByEmailAsync(emailAddress);
            if (user == null) return NotFound();

            var result = await _userManager.AddToRoleAsync(user, roleName);
```

```
            if (result.Succeeded) return Ok();

            return BadRequest(result.Errors);
        }


        [HttpGet]
        [Authorize(AuthenticationSchemes = "Bearer")]
        [Authorize(Roles = "Admin, Manager")]
        [Route("RoleTest")]
        public IActionResult RoleTest()
        {
            return Ok("You are an admin or manager!!!");
        }
    }
}
```

AppUserClaimsPrincipalFactory.cs

```
using ApplicationSecurity_Backend.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.Options;

namespace ApplicationSecurity_Backend.Factory
{
    public class AppUserClaimsPrincipalFactory:
UserClaimsPrincipalFactory<AppUser, IdentityRole>
    {
        public AppUserClaimsPrincipalFactory(UserManager<AppUser> userManager,
        RoleManager<IdentityRole> roleManager,
        IOptions<IdentityOptions> optionsAccessor)
        : base(userManager, roleManager, optionsAccessor)
        {
        }

    }
}
```

AppDBContext.cs

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace ApplicationSecurity_Backend.Models
{
    public class AppDbContext:IdentityDbContext<AppUser>
    {

        public AppDbContext(DbContextOptions<AppDbContext> options) :
base(options)
        {
        }

        public DbSet<Course> Courses { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 1,
                    Name = "AIM101",
                    Duration = "Semester",
                    Description = "Year 1, Semester 1. Academic Information
Management"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 2,
                    Name = "ALL121",
                    Duration = "Semester",
                    Description = "Year 1, Semester 2. Academic Literacy for
IT"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 3,
                    Name = "INF171",
```

```
                Duration = "Year",
                Description = "Year 1. Systems Analysis and Design"
            }
        );
        modelBuilder.Entity<Course>()
            .HasData(
            new
            {
                CourseId = 4,
                Name = "INF271",
                Duration = "Year",
                Description = "Year 2. Systems Analysis and Design"
            }
        );
        modelBuilder.Entity<Course>()
            .HasData(
            new
            {
                CourseId = 5,
                Name = "INF272",
                Duration = "Year",
                Description = "Year 2. Programming"
            }
        );
        modelBuilder.Entity<Course>()
            .HasData(
            new
            {
                CourseId = 6,
                Name = "INF214",
                Duration = "Semester",
                Description = "Year 2, Semester 1. Databases"
            }
        );
        modelBuilder.Entity<Course>()
            .HasData(
            new
            {
                CourseId = 7,
                Name = "INF315",
                Duration = "Semester",
                Description = "Year 3, Semester 1. Programming Management"
            }
        );
        modelBuilder.Entity<Course>()
            .HasData(
            new
            {
```

```
                    CourseId = 8,
                    Name = "INF324",
                    Duration = "Semester",
                    Description = "Year 3, Semester 2. IT Trends"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 9,
                    Name = "INF354",
                    Duration = "Semester",
                    Description = "Year 3, Semester 1. Programming"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 10,
                    Name = "INF370",
                    Duration = "Year",
                    Description = "Year 3. Project"
                }
            );
        }
    }
}
```

AppUser.cs

```
using Microsoft.AspNetCore.Identity;

namespace ApplicationSecurity_Backend.Models
{
    public class AppUser: IdentityUser
    {
    }
}
```

Course.cs

```
namespace ApplicationSecurity_Backend.Models
{
    public class Course
    {
        public int CourseId { get; set; }
        public string Name { get; set; }
        public string Duration { get; set; }
        public string Description { get; set; }
    }
}
```

IRepository.cs

```
namespace ApplicationSecurity_Backend.Models
{
    public interface IRepository
    {
        Task<Course[]> GetAllCoursesAsync();
    }
}
```

Repository.cs

```
using Microsoft.EntityFrameworkCore;

namespace ApplicationSecurity_Backend.Models
{
    public class Repository:IRepository
    {
        private readonly AppDbContext _appDbContext;


        public Repository(AppDbContext appDbContext)
        {
            _appDbContext = appDbContext;
        }

        public async Task<Course[]> GetAllCoursesAsync()
        {
            IQueryable<Course> query = _appDbContext.Courses;
            return await query.ToArrayAsync();
        }
    }
}
```

UserViewModel.cs

```
namespace ApplicationSecurity_Backend.ViewModels
{
    public class UserViewModel
    {
        public string emailaddress { get; set; }
        public string password { get; set; }

    }
}
```

Frontend Angular

Login.component.html

```
<div class="login-wrapper" fxLayout="row" fxLayoutAlign="center center">
  <button mat-stroked-button color="primary" class="btn-block"
(click)="Login()">Log in</button>


  <button mat-stroked-button color="primary" class="btn-block"
(click)="GetCourses()">Get Courses</button>

  <div>{{courses | json}}</div>
  </div>
```

Login.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { DataService } from '../services/data.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent {

  courses:any[] = []

  constructor(private router: Router, private dataService: DataService) { }


  Login(){
    this.dataService.Login().subscribe((result: any) =>
      localStorage.setItem('Token', JSON.stringify(result))
    )
```

```
  }

  GetCourses(){
    this.dataService.Courses().subscribe((result: any[]) =>
      {this.courses = result}
    )
  }

}
```

Auth-interceptors.ts

```
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from
"@angular/common/http";
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";

@Injectable()
export class AuthInterceptor implements HttpInterceptor {

    intercept(req: HttpRequest<any>,
              next: HttpHandler): Observable<HttpEvent<any>> {

        if (localStorage.getItem('Token')) {
            const jwt = JSON.parse(localStorage.getItem('Token')!)
            const token = jwt.token

            const cloned = req.clone({
                headers: req.headers.set("Authorization",
                    "Bearer " + token)
            });

             return next.handle(cloned);
        }
        else {
            return next.handle(req);
        }
    }
}
```

Data.service.ts

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

@Injectable({
  providedIn: 'root'
})
export class DataService {

  apiUrl = 'http://localhost:5240/api/'

  constructor(private httpClient: HttpClient) {
  }

  Login(){
    let user = new UserCredentials
    return this.httpClient.post(`${this.apiUrl}Course/Login`, user)
  }

  Courses(){
    return this.httpClient.get<any>(`${this.apiUrl}Course/GetAllCourses`)
  }

}

/*********************************************
 * *****************
 * This is where you simulate the login by entering your own credentials
created via the API Swagger UI
 * ******************
 *********************************************/
class UserCredentials {
  EmailAddress:string = 'test@gmail.com';
  Password:string = 'password123'
}
```

Material.module.ts

```
import {NgModule} from '@angular/core'

import { MatAutocompleteModule } from '@angular/material/autocomplete';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatNativeDateModule } from '@angular/material/core'
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
```

```
import { MatRadioModule } from '@angular/material/radio';
import { MatSelectModule } from '@angular/material/select';
import { MatSliderModule } from '@angular/material/slider';
import { MatSlideToggleModule } from '@angular/material/slide-toggle';
import { MatMenuModule } from '@angular/material/menu';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatCardModule } from '@angular/material/card';
import { MatDividerModule } from '@angular/material/divider';
import { MatExpansionModule } from '@angular/material/expansion';
import { MatGridListModule } from '@angular/material/grid-list';
import { MatListModule } from '@angular/material/list';
import { MatStepperModule } from '@angular/material/stepper';
import { MatTabsModule } from '@angular/material/tabs';
import { MatTreeModule } from '@angular/material/tree';
import { MatButtonModule } from '@angular/material/button';
import { MatButtonToggleModule } from '@angular/material/button-toggle';
import { MatBadgeModule } from '@angular/material/badge';
import { MatChipsModule } from '@angular/material/chips';
import { MatIconModule } from '@angular/material/icon';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatProgressBarModule } from '@angular/material/progress-bar';
import { MatRippleModule } from '@angular/material/core';
import { MatBottomSheetModule } from '@angular/material/bottom-sheet';
import { MatDialogModule } from '@angular/material/dialog';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { MatTooltipModule } from '@angular/material/tooltip';
import { MatPaginatorModule } from '@angular/material/paginator';
import { MatSortModule } from '@angular/material/sort';
import { MatTableModule } from '@angular/material/table';

@NgModule({
  declarations: [],
  exports: [
    MatAutocompleteModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatNativeDateModule,
    MatFormFieldModule,
    MatInputModule,
    MatRadioModule,
    MatSelectModule,
    MatSliderModule,
    MatSlideToggleModule,
    MatMenuModule,
    MatSidenavModule,
    MatToolbarModule,
    MatCardModule,
```

```
    MatDividerModule,
    MatExpansionModule,
    MatGridListModule,
    MatListModule,
    MatStepperModule,
    MatTabsModule,
    MatTreeModule,
    MatButtonModule,
    MatButtonToggleModule,
    MatBadgeModule,
    MatChipsModule,
    MatIconModule,
    MatProgressSpinnerModule,
    MatProgressBarModule,
    MatRippleModule,
    MatBottomSheetModule,
    MatDialogModule,
    MatSnackBarModule,
    MatTooltipModule,
    MatPaginatorModule,
    MatSortModule,
    MatTableModule
  ]
})
export class MaterialModule { }
```

LP 11: Reporting

Introduction to reporting

Reporting that may show detailed and aggregated data, facilitating analysis and decision-making, is a crucial type of output that any information system should provide.

What a report IS NOT --

- Data dump from a table
- List displayed from a table
- Any type of unprocessed data

What a report IS --

- A report is a document that presents information in an organized format for a specific audience and purpose.

Why do reporting

- Reports help organisations make better decisions
- The availability of automated reports avoids the need for manual effort to produce reports.
- Reporting can improve management effectiveness.
- Reporting can improve an organisation's responsiveness to issues.
- Reporting can optimise resource allocation and usage across organisational operations.

Report types

Summary reports (overview of details over time – aggregate data)

- provide concise accounts of business activities
- they can display data year-to-year – or another time frame
- can be tailored to include tables and graphs.

Exceptions reports

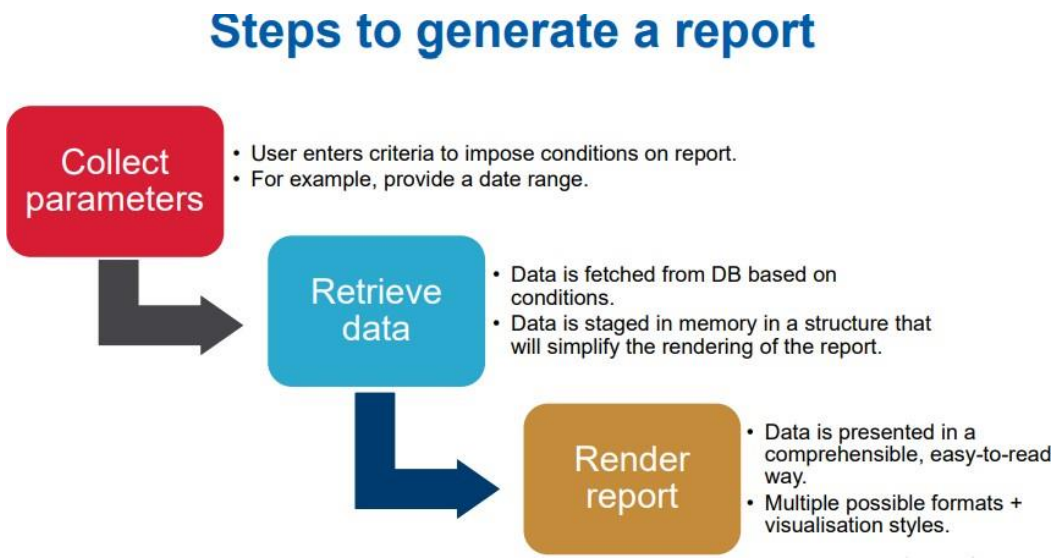
- any statistics that fall outside of a normal range such as cost overruns or production downtime

Trends report (categories over time)

- Categories of data pivoted against each other with a timeframe

Detail reports (specific instances of record joins at specific times)

- specific to narrowly-defined activities
- For example you might wish to review the sales of one representative, orders from one supplier or business from one store location



Important report design considerations

1. Format

- On-Screen
- Downloadable (PDF, Word, or image formats)

2. Visualisation Styles

- Tabular – data displayed in a table with rows/columns for subtotals and totals.
- Graphic – data displayed graphically, for example, a chart
- Hybrid – charts, and tables can be combined to create reports with richer information.
- Dashboard - a visual display of the most important information

3. Calculated information

- Aggregation - groups of records or data points are replaced with summarised values.

Reporting in Angular

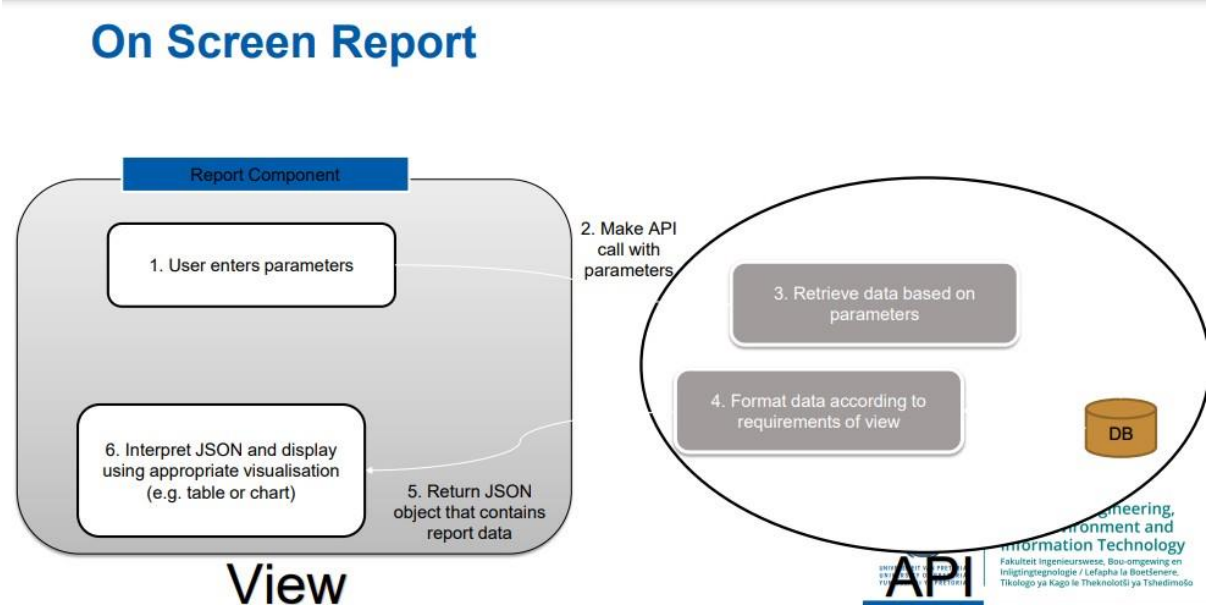


Chart.js

- A simple and flexible charting JavaScript library.
- Chart.js is an open source, community-maintained project (completely free!)
- Renders charts across all modern browsers because it outputs charts as custom HTML 5 Canvas Elements
- Redraws charts on window resize for responsiveness.
- To install in angular simply run the command:
    - npm install -g chart.js

LP 11: Reporting Source code

Backend API

Controller

RegionContoller.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ZAHike.API.Data;
using ZAHike.API.Models.Domain;

namespace ZAHike.API.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class RegionsController : ControllerBase
    {
        private readonly ZAHikeDbContext zAHikeDbContext;
        public RegionsController(ZAHikeDbContext zAHikeDbContext)
        {
            this.zAHikeDbContext = zAHikeDbContext;
        }

        [HttpGet]
       public async Task<IActionResult> GetAllRegions()
       {
            var listOfRegions = await zAHikeDbContext.Regions
                                    .Include(t => t.Trails)
                                    .ToListAsync();

            return Ok(listOfRegions);
        }
    }
}
```

ZAHikeDBContext.cs

```
using Microsoft.EntityFrameworkCore;
using ZAHike.API.Models.Domain;

namespace ZAHike.API.Data
{
    public class ZAHikeDbContext:DbContext
    {
        public ZAHikeDbContext(DbContextOptions<ZAHikeDbContext> options):
base(options)
```

```
        {

        }
        public DbSet<Region> Regions { get; set; }
        public DbSet<HikeTrail> Trails { get; set; }
        public DbSet<TrailDifficulty> TrailDifficulty { get; set; }
    }
}
```

Models

HikeTrail.cs

```
using System.ComponentModel.DataAnnotations;

namespace ZAHike.API.Models.Domain
{
    public class HikeTrail
    {
        [Key]
        public Guid Id { get; set; }
        public string Name { get; set; }
        public double Length { get; set; }
        public Guid RegionId { get; set; }
        public Guid TrailDifficultyId { get; set; }

        //Navigation Property
        public Region Region { get; set; }
        public TrailDifficulty TrailDifficulty { get; set; }
    }
}
```

Region.cs

```
using System.ComponentModel.DataAnnotations;

namespace ZAHike.API.Models.Domain
{
    public class Region
    {
        [Key]
        public Guid Id { get; set; }
        public string Code { get; set; }
        public string Name { get; set; }
        public double Area { get; set; }
        public double Lat { get; set; }
        public double Long { get; set; }
        public long Population { get; set; }
```

```
        //Navigation Property
        public IEnumerable<HikeTrail> Trails { get; set; }
    }
}
```

TrailDifficulty.cs

```
namespace ZAHike.API.Models.Domain
{
    public class TrailDifficulty
    {
        public Guid Id { get; set; }
        public string Code { get; set; }
    }
}
```

Frontend Angular

Charts.component.html

```
<div class="row">
    <div class="col-lg-6">
        <h2>Line Chart</h2>
        <canvas id="linechart"></canvas>
    </div>
    <div class="col-lg-6">
        <h2>Bar Chart</h2>
        <canvas id="barchart"></canvas>

    </div>
    <div class="col-lg-6">
        <h2>Pie Chart</h2>
        <canvas id="piechart"></canvas>

    </div>
    <div class="col-lg-6">
        <h2>Doughnut Chart</h2>
        <canvas id="dochart"></canvas>
    </div>
    <div class="col-lg-6">
        <h2>polarArea Chart</h2>
        <canvas id="pochart"></canvas>
    </div>

    <div class="col-lg-6">
        <h2>Radar Chart</h2>
        <canvas id="rochart"></canvas>
    </div>
```

```
</div>
```

Charts.component.ts

```
import { Component, ElementRef, OnInit, ViewChild } from '@angular/core';
import { Chart, registerables } from 'chart.js';
import { RegionModel } from '../Models/regionModel';
import { RegionService } from '../service/region.service';


Chart.register(...registerables);

@Component({
  selector: 'app-charts',
  standalone: true,
  imports: [],
  templateUrl: './charts.component.html',
  styleUrl: './charts.component.scss'
})
export class ChartsComponent implements OnInit{
  data: any;
  @ViewChild('myTemp')
  myTempRef!: ElementRef;


  constructor(private regionService : RegionService) {}

  ngOnInit(): void {
    this.regionService.getRegions().subscribe(response => {
      let regionList = response;

      this.data = response.$values;

      this.populateChartData(this.data);
      console.log('data',regionList)
      return regionList
    });
  }

  populateChartData(data: RegionModel[]) {

    let labelsData: string [] = [];
    let labelsPopulation: number [] = [];

    data.forEach((element: any) => {
      labelsData.push(element.code);
      labelsPopulation.push(element.population)
    });
```

```
new Chart("barchart", {
  type: 'bar',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1
    }]
  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      },
    }

  }
});


new Chart("piechart", {
  type: 'pie',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1
    }]
  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
  }
});

new Chart("dochart", {
  type: 'doughnut',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
```

```
        data: labelsPopulation,
        borderWidth: 1
      }]
    },
    options: {
      scales: {
        y: {
          beginAtZero: true
        }
      }
    }
  });

  new Chart("pochart", {
    type: 'polarArea',
    data: {
      labels: labelsData,
      datasets: [{
        label: '# of Population',
        data: labelsPopulation,
        borderWidth: 1
      }]
    },
    options: {
      scales: {
        y: {
          beginAtZero: true
        }
      }
    }
  });

  new Chart("rochart", {
    type: 'radar',
    data: {
      labels: labelsData,
      datasets: [{
        label: '# of Population',
        data: labelsPopulation,
        borderWidth: 1
      }]
    },
    options: {
      scales: {
        y: {
          beginAtZero: true
        }
      }
    }
```

```
      }
    });

    new Chart("linechart", {
      type: 'line',
      data: {
        labels: labelsData,
        datasets: [{
          label: '# of Population',
          data: labelsPopulation,
          borderWidth: 1

        }]

      },
      options: {
        scales: {
          y: {
            beginAtZero: true
          }
        }
      }
    });

    new Chart("bubchart", {
      type: 'bubble',
      data: {
        labels: labelsData,
        datasets: [{
          label: '# of Population',
          data: labelsPopulation,
          borderWidth: 1

        }]

      },
      options: {
        scales: {
          y: {
            beginAtZero: true
          }
        }
      }
    });

  }

}
```

Region.model.ts

```
export class RegionModel
{
    id: string = '';
    code: string = '';
    name: string = '';
    area: number = 0;
    lat: number = 0;
    long: number = 0;
    population: number = 0;
}
```

Region.service.ts

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

@Injectable({
  providedIn: 'root'
})
export class RegionService {

  constructor(private httpclient : HttpClient) { }
  public getRegions(): Observable<any> {
    let appheaders = this.getHeaderConfigurations();
    return this.httpclient.get<any[]>('https://localhost:7250/Regions', {
headers: appheaders});
  }

  private getHeaderConfigurations()
    {
        return new HttpHeaders({
            'Content-Type': 'application/json; charset=utf-8',
               'Access-Control-Allow-Origin': '*'
        });
    }
}
```

App.component.html

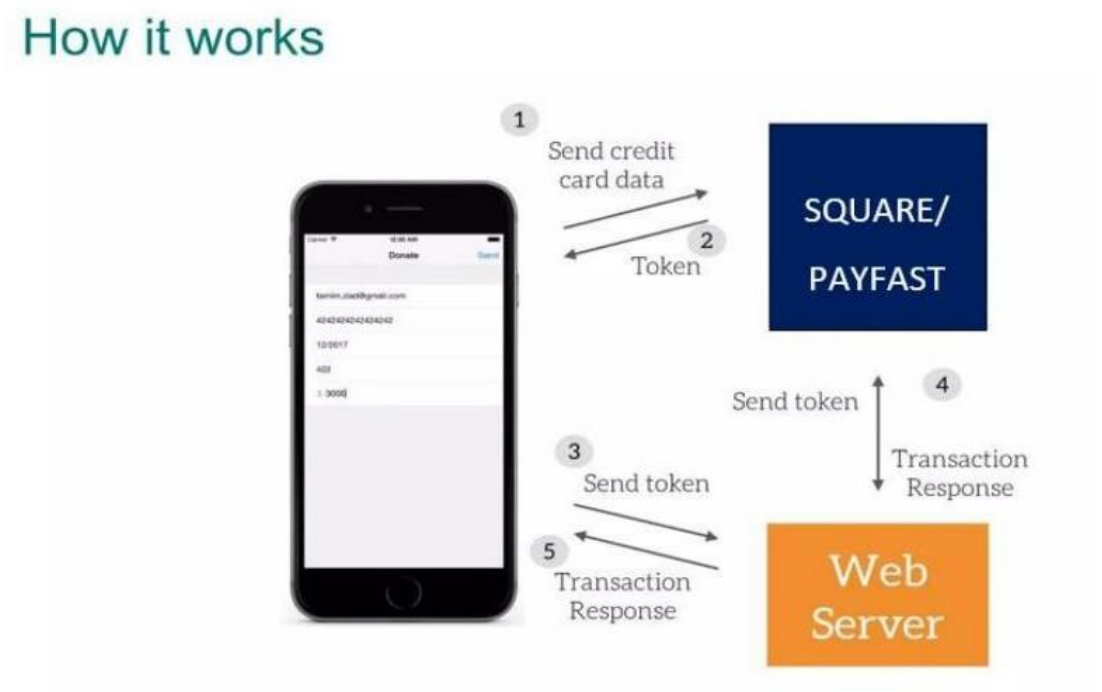
```
<!-- Navbar -->
<nav class="navbar navbar-expand-sm bg-success navbar-dark">
  <a class="navbar-brand" href="#"> FGNR Chart </a>
  <!-- Faerie Glen Nature Reserve -->
  <button
    class="navbar-toggler"
    type="button"
```

```
    data-toggle="collapse"
    data-target="#mynav"
  >
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="mynav">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
      <li class="nav-item">
        <a class="nav-link" href="">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="chart">Chart</a>
      </li>

    </ul>
    <form class="d-flex input-group w-auto">
      <input
          type="search"
          class="form-control"
          placeholder="Search"
          aria-label="Search"
        />

      <button class="btn btn-secondary my-sm-0" type="button">Search</button>
    </form>

  </div>

</nav>
<!-- Navbar -->

<router-outlet> </router-outlet>
```

LP 12: Advanced Concept 1

Introduction

- The technology merchants employ to receive customer debit or credit card payments is known as a payment gateway.
- The word covers both the online shopping cart payment processing portals and the actual card-reading hardware in real retail establishments.



How to integrate payments?

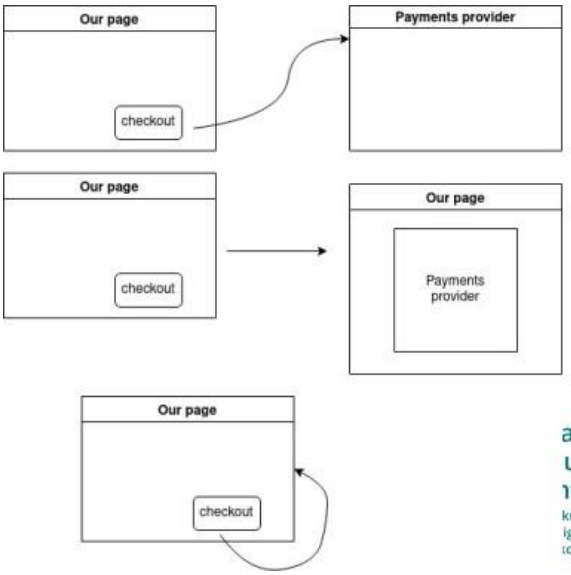
1. General approaches? How would you do it?

    1. Post a web form
    2. Java script injection
    3. Direct API access

2.

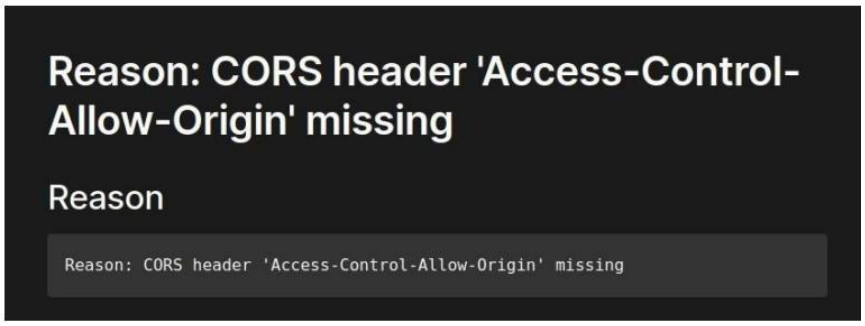
3. Why would one ever consider the first 2?

4. What about option 3?



CORS I

1. “Protected” apis



On the topic of CORS II

1. Consider the following:

    1. Browser brings “together” multiple applications in one
    1. environment
    2. MVC application running on tab 1
    3. Bank app open on tab

2. Browser’s solution: a strict same-origin policy

3. Implications of frontend/backend split? Scraping data from other websites? Etc.

4. CORS introduced ~2004 to safely implement a cross-origin policy

5. Q: What is an origin?

    1. https://mywebsite.com/api
    2. http://mywebsite.com/api/v2
    3. http://mywebsite.com/api/v2:8081
    4. https://myotherwebsite.com:4000

6. Idea behind CORS: give the server some control

7. The server decides the following

    1. Accepts all cross-origin requests
    2. Accepts only some cross-origin requests
    3. Decides on acceptable Http actions

8. 1. Benefits of CORS

    1. Wide audience for your API. They are not automatically blocked.
    2. Greater flexibility and control for the server
    3. Standardisation: no need for custom cross-origin solutions across different applications

2. Do not blindly accept all cross-origin requests

Advanced concepts 1 source code

Cancel.component.html

```
h1>Cancelled</h1>
<p>Not sure you want to buy?;<br /> we'll preserve your cart until you're
ready!</p>
```

Cancel.component.ts

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

@Component({
  selector: 'app-cancel',
  templateUrl: './cancel.component.html',
  styleUrls: ['./cancel.component.css']
})
export class CancelComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Cart.component.html

```
<h2 class="my-5">Items in cart</h2>

<div class="container products-container">
    <table class="table">
        <thead class="table-dark">
            <tr>
                <th scope="col">Subscription</th>
                <th scope="col">Price</th>
                <th scope="col">Quantity</th>
                <th scope="col">Total Cost</th>
            </tr>
        </thead>


        <tbody class="table-body">
            <tr *ngFor="let cartSub of subscriptionsInCart">
                <td>{{cartSub.subscription.name}}</td>
                <td>{{cartSub.subscription.price}}</td>
                <td>
                    <span class="increase" style="color:#89cff0">
```

```
                        <i class="fa-solid fa-circle-left fa-lg"
(click)="reduceProdCount(cartSub.subscription)"></i>
                    </span>
                    {{cartSub.quantity}}
                    <span class="decrease" style="color:#89cff0">
                        <i class="fa-solid fa-circle-right fa-lg"
(click)="increaseProdCount(cartSub.subscription)"></i>
                    </span></td>
                <td>{{cartSub.totalCost}}</td>
            </tr>
        </tbody>

        <tfoot class="table-footer">
            <tr>
                <td></td>
                <td></td>
                <td><b>Total:</b></td>
                <td>{{totalCostOfSubcriptionsInCart}}</td>
            </tr>
        </tfoot>

    </table>

    <br>

    <!-- <app-payfastcheckout></app-payfastcheckout> -->
    <app-squarecheckout></app-squarecheckout>
</div>
```

Cart.component.ts

```
import { Component, OnInit } from '@angular/core';
import { SubscriptionCartOrganiserService } from
'../services/SubscriptionCartOrganiser.service';
import { CartSubScription } from '../models/CartSubscriptionVM.model';
import { Subscription } from '../models/Subscription.model';

@Component({
  selector: 'app-cart',
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.css']
})
export class CartComponent implements OnInit {
  subscriptionsInCart : CartSubScription [] = [];
  totalCostOfSubcriptionsInCart :number = 0;

  constructor(private cartManager : SubscriptionCartOrganiserService) {
    this.loadSubscriptions();
```

```
    cartManager.cartProductsNumberDS.subscribe(num => {
        this.loadSubscriptions();
    });
  }

  ngOnInit(): void {
  }

  loadSubscriptions() {
    this.subscriptionsInCart = this.cartManager.getSubscriptionsInCart();
    this.totalCostOfSubcriptionsInCart =
this.cartManager.getTotalCostOfSubcriptionsInCart();
  }

  increaseProdCount (sub : Subscription) {
    for (var idx = 0; idx < this.subscriptionsInCart.length; idx++) {
      if (this.subscriptionsInCart[idx].subscription.id == sub.id) {
        this.cartManager.addProdFromCart(this.subscriptionsInCart[idx].subscri
ption);
      }
    }
  }

  reduceProdCount (sub : Subscription) {
    for (var idx = 0; idx < this.subscriptionsInCart.length; idx++) {
      if (this.subscriptionsInCart[idx].subscription.id == sub.id) {
         this.cartManager.removeProdFromCart(this.subscriptionsInCart[idx].sub
scription);
      }
    }
  }

}
```

Models

CartSubscriptionVM.model.ts

```
import { Subscription } from "./Subscription.model";

export class CartSubScription {
    subscription : Subscription;
    quantity : number = 1;
    totalCost : number = 0;

    constructor(subscr : Subscription, quant: number) {
        this.subscription = subscr;
        this.quantity = quant;
```

```
        this.totalCost = quant * subscr.price;
    }

    increment() {
        this.quantity +=1
    }
}
```

Subscription.model.ts

```
export class Subscription {
    id : number = 0;
    name : string = "";
    description : string = "";
    price : number = 0;
}
```

Navigation-bar.component.html

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" routerLink="">Home <span class="sr-
only">(current)</span></a>
      </li>
    </ul>
    <a class="navbar-brand" routerLink="cart">
        <span class="itemCount">{{numCartItems}}</span>
        <i class="fa-solid fa-cart-shopping"></i>
      </a>
  </div>
</nav>
```

Navigation-bar.component.ts

```
import { Component, OnInit } from '@angular/core';
import { SubscriptionCartOrganiserService } from
'../services/SubscriptionCartOrganiser.service';

@Component({
```

```
  selector: 'app-navigation-bar',
  templateUrl: './navigation-bar.component.html',
  styleUrls: ['./navigation-bar.component.css']
})
export class NavigationBarComponent implements OnInit {

  numCartItems : number = 0;
  constructor(private cartManager : SubscriptionCartOrganiserService) {
    this.numCartItems = cartManager.getNumberOfItemsInCart();

    cartManager.cartProductsNumberDS.subscribe(num => {
      this.numCartItems = num;
    });
   }

  ngOnInit(): void {
  }

}
```

Payfastcheckout.component.html

```
<button type="button" class="btn btn-primary m-3"
(click)="doOnSitePayment()">Checkout</button>
```

Payfastcheckout.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { SubscriptionCartOrganiserService } from
'../services/SubscriptionCartOrganiser.service';
import { Md5 } from 'ts-md5';
import { FormBuilder } from '@angular/forms'
import { environment } from 'src/environments/environment';
declare function payfast_do_onsite_payment(param1 : any, callback: any): any;

@Component({
  selector: 'app-payfastcheckout',
  templateUrl: './payfastcheckout.component.html',
  styleUrls: ['./payfastcheckout.component.css']
})
export class PayfastcheckoutComponent implements OnInit {

  constructor(private httpComms : HttpClient, private pageRouter : Router,
private cartManager : SubscriptionCartOrganiserService, private formBuilder:
FormBuilder) {
```

```
  }

  ngOnInit(): void {

  }

  getSignature(data : Map<string, string>) : string {
    let tmp = new URLSearchParams();
    data.forEach((v, k)=> {
      tmp.append(k, v)
    });
    let queryString = tmp.toString();
    let sig = Md5.hashStr(queryString);
    return sig;
  }

  async doOnSitePayment() {
    let onSiteUserData = new Map<string, string>();
    onSiteUserData.set("merchant_id", "10026206")
    onSiteUserData.set("merchant_key", "wy3z2mq4jknd2")

    onSiteUserData.set('return_url', window.location.origin + '/success')
    onSiteUserData.set('cancel_url', window.location.origin + '/cancel')

    onSiteUserData.set("email_address", 'test@user.com');

    onSiteUserData.set("amount",
this.cartManager.getTotalCostOfSubcriptionsInCart().toString());
    onSiteUserData.set("item_name", this.cartManager.getCartOrderName());

    onSiteUserData.set('passphrase', 'HelloWorldHello');

    let signature = this.getSignature(onSiteUserData);
    onSiteUserData.set('signature', signature);


    let formData = new FormData();
    onSiteUserData.forEach((val, key) => {
      formData.append(key, val);
    });

    let response = await fetch(environment.payfastOnsiteEndpoint, {
      method: 'POST',
      body: formData,
      redirect: 'follow'
    });

    let respJson = await response.json();
    let uuid = respJson['uuid'];
```

```
    payfast_do_onsite_payment({'uuid': uuid}, (res: any) => {
      if (res == true) {
        this.pageRouter.navigate(['/success'])
      }
      else {
        this.pageRouter.navigate(['/cancel'])
      }
    });
  }

  doFormPayment() {
    let onSiteUserData = new Map<string, string>();
    onSiteUserData.set("merchant_id", "10029580")
    onSiteUserData.set("merchant_key", "n85royxznbmq9")

    onSiteUserData.set('return_url', window.location.origin + '/success')
    onSiteUserData.set('cancel_url', window.location.origin + '/cancel')

    onSiteUserData.set("email_address", 'test@user.com');

    onSiteUserData.set("amount",
this.cartManager.getTotalCostOfSubcriptionsInCart().toString());
    onSiteUserData.set("item_name", this.cartManager.getCartOrderName());

    onSiteUserData.set('passphrase', 'HelloWorldHello');

    let signature = this.getSignature(onSiteUserData);
    onSiteUserData.set('signature', signature);

    let autoPaymentForm = this.formBuilder.group(onSiteUserData);

    this.httpComms.post('https://sandbox.payfast.co.za/eng/process',
onSiteUserData).subscribe(resp => {
      console.log(resp);
    });
  }

}
```

Productcatalog.component.html

```
<div class="row mt-5">
        <div class="card m-3" style="width: 20rem;" *ngFor="let prod of
products">
            <div class="card-body">
                <h5 class="card-title">{{prod.name}}</h5>
                <p class="card-text">
                    {{prod.description}}
                </p>
                <button type="button" class="btn btn-primary"
(click)="addSubscriptionToCart(prod)">
                    Add to cart
                </button>
            </div>
        </div>
</div>

<div aria-live="polite" aria-atomic="true" style="position: relative; min-
height: 200px;">
    <div class="toast" style="position: absolute; top: 0; right: 0;">
      <div class="toast-header">
        <img src="..." class="rounded mr-2" alt="...">
        <strong class="mr-auto">Cart items</strong>
        <small>Now</small>
        <button type="button" class="ml-2 mb-1 close" data-dismiss="toast"
aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="toast-body">
        Added item to cart.
      </div>
    </div>
  </div>
```

Productcatalog.component.ts

```
import { Component, Input, OnChanges, OnInit, SimpleChanges } from
'@angular/core';
import { Subscription } from '../models/Subscription.model';
import { FakeSubscriptionDataService } from
'../services/FakeSubscriptionData.service';
import { SubscriptionCartOrganiserService } from
'../services/SubscriptionCartOrganiser.service';

@Component({
  selector: 'app-productcatalog',
  templateUrl: './productcatalog.component.html',
```

```
  styleUrls: ['./productcatalog.component.css']
})
export class ProductcatalogComponent implements OnInit {

  products : Subscription [] = [];

  constructor(private fakeDataProvider : FakeSubscriptionDataService, private
cartSubscriptionService : SubscriptionCartOrganiserService) {
    this.products = fakeDataProvider.getOfferedSubscriptions();
  }

  ngOnInit(): void {
  }

  addSubscriptionToCart(product : Subscription) {
    this.cartSubscriptionService.addProdFromCart(product);
  }

}
```

Services

FakeSubscriptiondata.service.ts

```
import { Injectable } from "@angular/core";
import { Subscription } from "../models/Subscription.model";

@Injectable({
    providedIn: 'root'
})
export class FakeSubscriptionDataService {
    subscriptions : Subscription[];

    constructor () {
        this.subscriptions = [
            {
              id: 1,
              name: "Netflix",
              description:  "At Netflix, we want to entertain the world.
Whatever your taste, and no matter where you live, we give you access to best-
in-class TV series, documentaries, feature films and mobile games.",
              price : 100
            },
            {
              id: 2,
              name: "Showmax",
              description: "Showmax is an internet TV service. What sets
Showmax apart is a unique combination of hit African content, first and
```

```
exclusive international series, movies, the best kids’ shows, and live
sport.",
              price : 500
            },
            {
              id: 3,
              name: "Tencent Video",
              description: "Tencent Video is China's second-largest video-
streaming platform. It includes a variety of categories of online videos. The
most popular categories on the platform include Chinese TV shows and China-
made animation shows.",
              price : 800
            },
            {
              id: 4,
              name: "BBC iPlayer",
              description: "BBC iPlayer is a video on demand service from the
BBC. The service is available on a wide range of devices, including mobile
phones and tablets, personal computers and smart televisions.",
              price : 900
            },
          ];
    }

    getOfferedSubscriptions () {
        return this.subscriptions;
    }
}
```

SubscriptionCartOrganiser.service.ts

```
import { Injectable } from "@angular/core";
import { Subject } from "rxjs";
import { CartSubScription } from "../models/CartSubscriptionVM.model";
import { Subscription } from "../models/Subscription.model";

@Injectable({
    providedIn: 'root'
})
export class SubscriptionCartOrganiserService {
    static tmpSubscriptionsCartName : string = "ls-cart-subscriptions";
    cartProductsNumberDS = new Subject<number>();
    cartItemsOrderName : string = "Subs Order @ ";

    notifyOnNewItemInCart() {
        this.cartProductsNumberDS.next(this.getNumberOfItemsInCart());
    }
```

```
    getLocalStorageSubscriptions(): Subscription[] {
        let storedSubString =
localStorage.getItem(SubscriptionCartOrganiserService.tmpSubscriptionsCartName
)
        let cartSubscriptions = [];
        if (storedSubString) {
          cartSubscriptions = JSON.parse(storedSubString)
        }
        return cartSubscriptions;
    }
    getNumberOfItemsInCart() : number {
        return this.getLocalStorageSubscriptions().length
    }

    getSubscriptionsInCart() : CartSubScription[] {
        let localStorageSubs = this.getLocalStorageSubscriptions();
        let cartSubscriptions : CartSubScription[] = [];

        let subCounts = new Map<Number, Number>(); //temporary storage
        localStorageSubs.forEach(sub => {
            if (!subCounts.has(sub.id)) {
                let count = localStorageSubs.filter(currSub => currSub.id ==
sub.id).length;
                subCounts.set(sub.id, count)
                let cartSub = new CartSubScription(sub, count);
                cartSubscriptions.push(cartSub);
            }
        });
        return cartSubscriptions;
    }

    getTotalCostOfSubcriptionsInCart() : number {
        let totalCost = 0;

        let cartSubs = this.getSubscriptionsInCart();
        cartSubs.forEach(cartSub => {
            totalCost += (cartSub.subscription.price * cartSub.quantity);
        });

        return totalCost;
    }

    getCartOrderName() {
        return this.cartItemsOrderName + Date.now();
    }

    addSubscriptionToCart(product : Subscription) {
```

```
        let storedSubString =
localStorage.getItem(SubscriptionCartOrganiserService.tmpSubscriptionsCartName
)

        let cartSubscriptions = [];
        if (storedSubString) {
          cartSubscriptions = JSON.parse(storedSubString)
        }
        cartSubscriptions.push(product);
        localStorage.setItem(SubscriptionCartOrganiserService.tmpSubscriptions
CartName, JSON.stringify(cartSubscriptions))

        this.notifyOnNewItemInCart();
      }


    removeProdFromCart(subscr : Subscription) {
        let storedSubString =
localStorage.getItem(SubscriptionCartOrganiserService.tmpSubscriptionsCartName
)

        let cartSubscriptions = [];
        if (storedSubString) {
          cartSubscriptions = JSON.parse(storedSubString)
        }
        for (var idx = 0; idx < cartSubscriptions.length; idx++) {
            if (cartSubscriptions[idx].id == subscr.id) {
                cartSubscriptions.splice(idx, 1);
                break;
            }
        }

        localStorage.setItem(SubscriptionCartOrganiserService.tmpSubscriptions
CartName, JSON.stringify(cartSubscriptions))

        this.notifyOnNewItemInCart();
    }

    addProdFromCart(subscr : Subscription) {
        this.addSubscriptionToCart(subscr);
        this.notifyOnNewItemInCart();
    }

    clearCart () {
        localStorage.removeItem(SubscriptionCartOrganiserService.tmpSubscripti
onsCartName);
        this.notifyOnNewItemInCart();
    }
```

Squarecheckout.component.html

```
<h2 class="my-5">Card details</h2>
<div id="card-container"></div>
<br>
<button type="button" class="btn btn-primary m-3"
(click)="onCheckout()">Checkout</button>

<div id="payment-status-container"></div>
```

Squarecheckout.component.ts

```
import { Component, OnInit } from '@angular/core';
import { environment } from 'src/environments/environment';
import { SubscriptionCartOrganiserService } from
'../services/SubscriptionCartOrganiser.service';
import { Router } from '@angular/router';
declare let Square : any;

@Component({
  selector: 'app-squarecheckout',
  templateUrl: './squarecheckout.component.html',
  styleUrls: ['./squarecheckout.component.css']
})
export class SquarecheckoutComponent implements OnInit {

  appId : string = environment.squareApplicationId;
  locationId : string = environment.squareLocationId;
  baseEndpoint : string = environment.squareEndpoint;
  card : any;

  constructor(private pageRouter : Router, private cartManager :
SubscriptionCartOrganiserService) {

  }

  async ngOnInit(): Promise<void> {

    const payments = Square.payments(this.appId, this.locationId);
    this.card = await payments.card();
    this.card.attach('#card-container');
  }

  async onCheckout() {
    try {
      let tokResp = await this.card.tokenize()
      this.doOnSitePayment(tokResp);
    }
    catch (e) {
```

```
      console.error(e);
    }

  }

  //https://developer.squareup.com/explorer/square/payments-api/create-payment?env=sandbox&appId=sq0idp-q5eD7Vs5yNKv_i6nJcbTUA&prefill=create-payment
  async doOnSitePayment(cardData : any) {

    let idemK = cardData.idempotencyKey || 'adsadsadasdadsdssd';

    let paymentData = {
      idempotency_key: idemK,
      locationId: this.locationId,
      source_id: cardData.token,
      amount_money : {
        amount : this.cartManager.getTotalCostOfSubcriptionsInCart(),
        currency : 'USD',
        label : this.cartManager.getCartOrderName()
      },
      verificationToken: cardData.verificationToken
    };

    const paymentResponse = await fetch(this.baseEndpoint+'/payments', {
      method: 'POST',
      headers: {
        'Authorization': 'Bearer EAAAELK9muY7VNHiyGPITTPkuGhorj8AFxL5qu4ReDRDWnJVFDxOKktzNBjqb9Xo',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(paymentData)
    });

    if (paymentResponse.ok) {
      this.pageRouter.navigate(['/success'])
    } else {
      console.log(`Cannot pay with Square. Error is ${JSON.stringify(paymentData)}`);
    }
  }

}
```

Success.component.html

```
<h1>Success</h1>
<p>We received your purchase;<br /> Your items will be sent shortly!</p>
```

Success.component.html

```
import { Component, OnInit } from '@angular/core';
import { SubscriptionCartOrganiserService } from
'../services/SubscriptionCartOrganiser.service';

@Component({
  selector: 'app-success',
  templateUrl: './success.component.html',
  styleUrls: ['./success.component.css']
})
export class SuccessComponent implements OnInit {

  constructor(private cartManager : SubscriptionCartOrganiserService) {

  }

  ngOnInit(): void {
    this.cartManager.clearCart();
  }

}
```

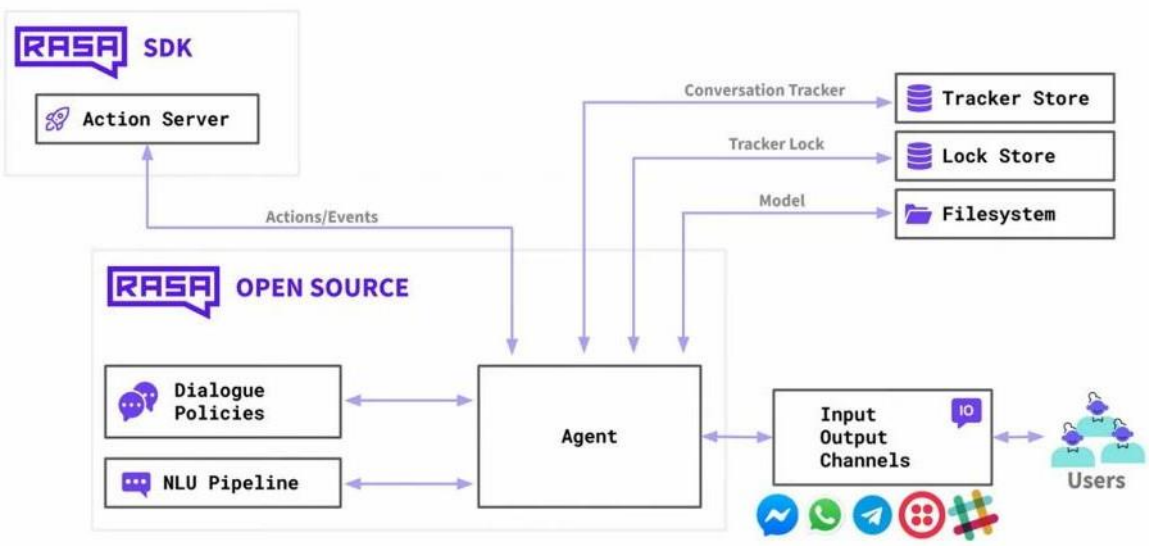
App-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CancelComponent } from './cancel/cancel.component';
import { CartComponent } from './cart/cart.component';
import { ProductcatalogComponent } from
'./productcatalog/productcatalog.component';
import { SuccessComponent } from './success/success.component';

const routes: Routes = [
  {path : '', component: ProductcatalogComponent},
  {path : 'cart', component: CartComponent},
  {path : 'success', component: SuccessComponent},
  {path: 'cancel', component: CancelComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

LP 13: Advanced Concepts II

Theory

- NLP: Natural Language Processing
- NLU: Natural Language Understanding
- NLG: Natural Language Generation



Terminology:

- Narrow assistant – defined skill-set
- Intents - are the goals or meaning the user is trying to convey
- Entities – are important keywords, that you want to capture, that the models extract from users message
- Custom Action - getting the assistant to do something for you. E.g., connect to a database, posting data to an API, send data to a spreadsheet
- Forms – uses slots to store values

Key Files:

- NLU.yml – create intents
- Domain.yml – Register intents and Write response templates
- Stories.yml – Write stories using new intents and responses

Install requirements:

- Visual studio for C++ dependencies (via VS)
- Python 3.10
- Rasa

Key commands:

- pip install rasa (install rasa)
- rasa init (start new project)
- rasa shell (command line)
- rasa interactive (interactive learning session)
- rasa run --enable-api (enable api)
- rasa run -m model --enable-api --cors "*" -p 5005 (run
- on specifc port with cors)

Advanced Concepts II Source Code

Frontend Angular

Chat-support.component.html

```
<div id="assistant">
  <button id="assistant-popup-button" (click)="openSupportPopup()">
    Chat Support?
  </button>
  <div id="assistant-popup" [style.display]="isOpen ? 'block' : 'none'">
    <div id="assistant-popup-header">
      Your friendly Assistant
      <button id="assistant-popup-close-button" (click)="openSupportPopup()">
       X
      </button>
    </div>
    <div id="assistant-popup-body">
      <div class="messages" #scrollMe>
        <div *ngFor="let message of messages" class="message">
          <div [class]="message.type">
            {{ message.message }}
          </div>
        </div>
        <div
          *ngIf="loading"
          class="message"
          style="width: 100%; display: block"
        >
          <div [class]="'client'">...</div>
        </div>
      </div>
    </div>
    <form id="assistant-popup-footer" [formGroup]="chatForm">
      <input
        formControlName="message"
        type="text"
        id="assistant-popup-input"
        placeholder="Type your message here..."
      />
```

```
      <button
        id="assistant-popup-submit-button"
        [disabled]="!chatForm.valid"
        (click)="sendMessage()"
      >
        Submit
      </button>
    </form>
  </div>
</div>
```

Chat-support.component.css

```
@import
url("https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap
");
#assistant {
  font-family: "Roboto", sans-serif;
  #assistant-popup-button {
    position: fixed;
    bottom: 20px;
    right: 20px;
    padding: 10px 20px;
    background-color: #333;
    color: #ffffff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 14px;
    z-index: 1000;
  }

  #assistant-popup {
    position: fixed;
    bottom: 40px;
    right: 20px;
    width: 450px;
    height: 50vh;
    min-height: 450px;
    background-color: white;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
    border-radius: 5px;
    z-index: 1000;
    display: none;
    #assistant-popup-header {
      background-color: #333;
      color: white;
```

```
    font-size: 18px;
    padding: 10px;
    border-top-left-radius: 5px;
    border-top-right-radius: 5px;
    #assistant-popup-close-button {
      float: right;
      border: none;
      background-color: transparent;
      color: #fff;
      font-size: 14px;
      cursor: pointer;
    }
  }

  #assistant-popup-body {
    height: calc(100% - 133px);
    padding: 10px;
  }

  #assistant-popup-footer {
    background-color: #333;
    color: white;
    font-size: 14px;
    padding: 10px;
    border-bottom-left-radius: 5px;
    border-bottom-right-radius: 5px;
    #assistant-popup-input {
      width: 100%;
      padding: 10px;
      border: 1px solid #fff;
      border-radius: 5px 5px 0 0;
      box-sizing: border-box;
      font-size: 14px;
    }

    #assistant-popup-submit-button {
      width: 100%;
      padding: 10px;
      background-color: #2ca1da;
      color: #fff;
      border: none;
      border-radius: 0 0 5px 5px;
      cursor: pointer;
      font-size: 14px;
    }
  }

  .messages {
```

```
      height: 100%;
      overflow: auto;
      .message {
        display: flow-root;
        width: 100%;
        .client {
          background-color: #d7d7d7;
          color: #333;
          padding: 10px;
          border-radius: 5px;
          margin-bottom: 10px;
          display: inline-block;
          max-width: 80%;
        }

        .user {
          border: 0.5px solid #333;
          background-color: #85ff7a;
          color: #333;
          padding: 10px;
          border-radius: 5px;
          margin-bottom: 10px;
          display: inline-block;
          max-width: 80%;
          text-align: right;
          float: right;
        }
      }
    }
  }
}
```

Chat-support.component.ts

```
import { Component, ViewChild } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { MessageService } from '../service/api.service';

export interface Message {
  type: string;
  message: string;
}

@Component({
  selector: 'app-chat-support',
  templateUrl: './chat-support.component.html',
  styleUrls: ['./chat-support.component.scss'],
```

```
})
export class ChatSupportComponent {
  isOpen = false;
  loading = false;
  messages: Message[] = [];
  chatForm = new FormGroup({
    message: new FormControl('', [Validators.required]),
  });
  @ViewChild('scrollMe') private myScrollContainer: any;

  constructor(private messageService: MessageService) {
  }

  openSupportPopup() {
    this.isOpen = !this.isOpen;
  }

  sendMessage() {
    const sentMessage = this.chatForm.value.message!;
    this.loading = true;
    this.messages.push({
      type: 'user',
      message: sentMessage,
    });
    this.chatForm.reset();
    this.scrollToBottom();
    this.messageService.sendMessage(sentMessage).subscribe((response: any) =>
{
      for (const obj of response) {
        let value
        if (obj.hasOwnProperty('text') ) {
          value = obj['text']
          this.pushMessage(value)

        }
        if (obj.hasOwnProperty('image') ) {
          value = obj['image']
          this.pushMessage(value)
        }
      }
    });
  }

  pushMessage(message:string){
     this.messages.push({
        type: 'client',
        message: message,
      });
```

```
      this.scrollToBottom();
  }

  scrollToBottom() {
    setTimeout(() => {
      try {
        this.myScrollContainer.nativeElement.scrollTop =
          this.myScrollContainer.nativeElement.scrollHeight + 500;
      } catch (err) {}
    }, 150);
  }
}
```

Services

Api.service.ts

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

@Injectable({
  providedIn: 'root',
})
export class MessageService {
  constructor(private http: HttpClient) {}

  sendMessage(message: string) {
    return this.http.post('http://localhost:5005/webhooks/rest/webhook', {
message: message });
  }
}
```

Assignment 2

Frontend

Accounts.page.html

```
<!-- <ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>account</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">account</ion-title>
    </ion-toolbar>
  </ion-header>
</ion-content> -->

<ion-header [translucent]="true" class="custom-header">
  <ion-toolbar class="custom-toolbar">
    <ion-title class="custom-title">account</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true" class="custom-content">
  <ion-card class="custom-card">
    <ion-card-content>
      <ion-item>
        <ion-label class="custom-label">Full Name</ion-label>
        <ion-input [(ngModel)]="accountName" [disabled]="!isUpdating"></ion-
input>
      </ion-item>
      <ion-item>
        <ion-label class="custom-label">Cell Phone Number</ion-label>
        <ion-input [(ngModel)]="cellNumber" [disabled]="!isUpdating"></ion-
input>
      </ion-item>
      <ion-item>
        <ion-label class="custom-label">Email Address</ion-label>
        <ion-input [(ngModel)]="emailAddress" [disabled]="!isUpdating"></ion-
input>
      </ion-item>
      <ion-button (click)="updateCust()" expand="block" class="custom-
button">{{ isUpdating ? 'Cancel' : 'Edit' }}</ion-button>
      <ion-button *ngIf="isUpdating" (click)="saveCust()" expand="block"
class="custom-button">Save</ion-button>

      <ion-item lines="none" class="custom-item">
```

```
        <ion-icon name="home-outline" slot="start" class="custom-icon"></ion-
icon>
        <ion-label class="custom-label">Edit Address</ion-label>
        <ion-button slot="end" fill="clear" class="custom-button">Click to
edit
          <ion-icon name="arrow-forward-circle"></ion-icon>
        </ion-button>
      </ion-item>
    </ion-card-content>
  </ion-card>

  <ion-card class="custom-card">
    <ion-card-content>
      <ion-header [translucent]="true" class="custom-header">
        <ion-toolbar class="custom-toolbar">
          <ion-title class="custom-title">Previous Orders</ion-title>
        </ion-toolbar>
      </ion-header>
      <ion-row *ngIf="previousOrders.length > 0" class="ion-align-items-center
ion-justify-content-center">
        <ion-card class="custom-card">
          <ion-card-content>
            <ion-card-title class="custom-title">{{ previousOrders[0].name
}}</ion-card-title>
            <ion-label class="custom-label">Previous order {{
previousOrders[0].total }} </ion-label>
            <ion-button expand="block" (click)="reOrder(previousOrders[0])"
class="custom-button">Reorder</ion-button>
          </ion-card-content>
        </ion-card>
      </ion-row>
    </ion-card-content>
  </ion-card>

  <ion-button size="medium" (click)="ButtonHelp()" class="custom-
button">Request Help</ion-button>
</ion-content>
```

Account.page.ts

```
import { Component, OnInit } from '@angular/core';
import { NavController } from '@ionic/angular';
import { ModalController } from '@ionic/angular';
import { CartPage } from '../cart/cart.page';
import { Router, NavigationExtras } from '@angular/router';
import { AlertController } from '@ionic/angular';
```

```
@Component({
  selector: 'app-account',
  templateUrl: './account.page.html',
  styleUrls: ['./account.page.scss'],
})
export class AccountPage implements OnInit {
  // Properties for user account information and orders
  accountName: string = ''; cellNumber: string = ''; emailAddress: string =
'';isUpdating: boolean = false;previousOrders: { name: string, total: number,
delivered: boolean }[] = [];

  // Constructor to inject dependencies
  constructor(private router: Router, private nav: NavController, private
alertController: AlertController) { }

  // Initialization logic
  ngOnInit() {
    // Load user account information and previous orders
    this.loadCust(); this.loadOrders();
  }

  // Toggle update mode for user account information
  updateCust() {
    this.isUpdating = !this.isUpdating;
  }

  // Save updated user account information to localStorage
  saveCust() {
    localStorage.setItem('username',
this.accountName);localStorage.setItem('phonenumber',
this.cellNumber);localStorage.setItem('email', this.emailAddress);

    this.isUpdating = false; // Exit update mode
  }

  // Load user account information from localStorage
  private loadCust() {
    this.accountName = localStorage.getItem('username') || '';this.cellNumber
= localStorage.getItem('phonenumber') || '';this.emailAddress =
localStorage.getItem('email') || '';
  }

  // Reorder items from a previous restaurant order
  reOrder(restaurant: any) {
    // Pass restaurant details to cart page using Angular Router navigation
    const navigationExtras: NavigationExtras = {
      state: {
        restaurantDetails: restaurant
```

```
      }
    };
    this.router.navigate(['/cart'], navigationExtras);
  }

  // Load previous orders from localStorage
  loadOrders() {
    const savedOrders = localStorage.getItem('Restaurants');
    if (savedOrders) {
      // Parse saved orders and map them to the previousOrders array
      this.previousOrders = JSON.parse(savedOrders).map((order: any) => ({
        name: order.restaurantName,total: order.restaurantDishPrice,delivered:
order.delivered || true,
      }));
    }
  }

  // Show help message in an alert dialog
  async ButtonHelp() {
    const toast = await this.alertController.create({
      header: 'Get Help', message: 'Get your own help',
    });
    toast.present(); // Display the alert
  }
}
```

Cart.page.html

```
<ion-content [fullscreen]="true" class="custom-content">
  <ion-header collapse="condense">
  </ion-header>
  <ion-content class="custom-content-inner">
    <ion-grid class="custom-grid">
      <ion-row class="ion-align-items-center ion-justify-content-center">
        <ion-col size="5" class="ion-text-center">
          <img src="{{ order.Restaurant.image }}" class="custom-image" />
        </ion-col>
      </ion-row>
    </ion-grid>
    <ion-grid class="custom-grid">
      <ion-row class="ion-align-items-center ion-justify-content-center">
        <ion-col size="2" class="ion-text-center">
          <ion-icon aria-hidden="true" name="list-outline" color="medium"
size="large" class="custom-icon"></ion-icon>
        </ion-col>
        <ion-col size="10">
          <ion-input placeholder="Instructions" class="custom-input"></ion-
input>
```

```
        </ion-col>
      </ion-row>
    </ion-grid>
    <ion-list class="custom-list">
      <ion-list-header class="custom-list-header">
        <ion-label class="custom-label">Your Order</ion-label>
      </ion-list-header>
      <ion-item class="custom-item">
        <ion-label class="custom-label">Order Total</ion-label>
        <ion-label slot="end" class="custom-label">R {{
order.Restaurant.price*order.Restaurant.orderAmount }} </ion-label>
      </ion-item>
      <ion-item class="custom-item">
        <ion-label class="custom-label">Delivery Fee</ion-label>
        <ion-label slot="end" class="custom-label">R {{
order.Restaurant.deliverFee }}</ion-label>
      </ion-item>
      <ion-item class="custom-item">
        <ion-label class="custom-label">Total to pay</ion-label>
        <ion-label slot="end" class="custom-label">R {{
(order.Restaurant.price*order.Restaurant.orderAmount) +
order.Restaurant.deliverFee }} </ion-label>
      </ion-item>
    </ion-list>
    <ion-item class="custom-item">
      <ion-label class="custom-label">Home</ion-label>
      <ion-label slot="end" class="custom-label">R {{
(order.Restaurant.price*order.Restaurant.orderAmount) +
order.Restaurant.deliverFee }} </ion-label>
      <ion-button slot="end" class="custom-button">Change Current
Location</ion-button>
    </ion-item>
    <ion-button expand="block" (click)="PayOrder()" class="custom-button">Pay
Here</ion-button>
  </ion-content>
</ion-content>
```

Cart.page.ts

```
import { Component, OnInit } from '@angular/core';
import { orders } from '../order';
import { AlertController } from '@ionic/angular';
import { ToastController } from '@ionic/angular';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-cart',
  templateUrl: './cart.page.html',
```

```
  styleUrls: ['./cart.page.scss'],
})
export class CartPage implements OnInit {
  // Properties for order and restaurant details
  order!: orders;
  restaurantDetails: any;

  // Constructor to inject dependencies
  constructor(private toastController: ToastController, private route:
ActivatedRoute) {}

  // Initialization logic
  ngOnInit() {
    // Load order details and restaurant details from localStorage and route
snapshot
    this.order = JSON.parse(localStorage.getItem('Order')!);
    this.restaurantDetails =
this.route.snapshot.paramMap.get('restaurantDetails');
  }

  // Method to handle payment and display a toast message
  async PayOrder() {
    const toast = await this.toastController.create({
      message: 'Payment has cleared',
      duration: 3000, // Duration of the toast message
    });
    toast.present(); // Display the toast message
  }
}
```

Home.page.html

```
<ion-header [translucent]="true" class="custom-header">
  <ion-toolbar class="custom-toolbar">
    <ion-title mode="md" class="custom-title">
      <span>Home</span>
      <ion-icon name="chevron-down-outline" class="custom-icon"></ion-icon>
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true" class="custom-content">
  <ion-header collapse="condense">
  </ion-header>
  <ion-content>
    <ion-card class="custom-card">
      <ion-card-header class="custom-card-header">
```

```
        <ion-card-title class="custom-card-title">Available Restaurants</ion-
card-title>
        <ion-card-subtitle class="custom-card-subtitle">Pick a
restaurant</ion-card-subtitle>
      </ion-card-header>
      <ion-card-content class="custom-card-content">
        <ion-list>
          <ion-item *ngFor="let item of Restaurants"
(click)="additionToCart(item)" class="custom-item">
            <ion-thumbnail slot="end">
              <!-- {{ item.image }} -->
              <!-- <img src= {{ item.image }} class="custom-thumbnail" /> -->
              <img src= ..\..\..\..\assets\chips.jpeg class="custom-thumbnail"
/>
            </ion-thumbnail>
            <ion-label class="custom-label">
              <h5>{{ item.name }}</h5>
              <p>{{ item.meal }}</p>
              <ion-text>
                <ion-icon name="fast-food-outline" size="small"></ion-icon>
                {{ item.ratings }}
              </ion-text>
            </ion-label>
            <ion-text slot="start" class="custom-text">
              <h6>R {{ item.price }}</h6>
              <p>Distance: {{ item.distance }}km</p>
            </ion-text>
          </ion-item>
        </ion-list>
      </ion-card-content>
    </ion-card>
  </ion-content>
</ion-content>
```

Home.page.ts

```
import { Component, OnInit } from '@angular/core';
import { IonicSlides } from '@ionic/angular';
import { IonicModule } from '@ionic/angular';
import { CommonModule } from '@angular/common';
import { orders } from '../order';
import { Restaurant } from '../restaurant';
import { AlertController } from '@ionic/angular';

@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
```

```
  styleUrls: ['./home.page.scss'],
})
export class HomePage implements OnInit {
  // Array to hold restaurant objects
  Restaurants: Restaurant[] = [];
  // Object to hold order details
  Order!: orders;

  // Constructor to inject dependencies
  constructor(private alertController: AlertController) {}

  // Initialization logic
  ngOnInit() {
    // Load restaurants and order details from localStorage
    this.Restaurants = JSON.parse(localStorage.getItem('Restaurants') ||
'[]');
    this.Order = JSON.parse(localStorage.getItem('Order') || '{}');

    // If no restaurants are saved, populate the database
    if (this.Restaurants == null) {
      this.restDb();
    }

    // If no order exists, initialize it
    if (!this.Order) {
      this.Order = new orders();
    }
  }

  // Show success message in an alert dialog
  async successMessage() {
    const alert = await this.alertController.create({
      message: 'New item has been added to cart',
      buttons: ['OK']
    });
    await alert.present();
  }

  // Method to populate restaurant database
  restDb() {
    // Clear existing restaurants array
    this.Restaurants = [];

    // Create and add restaurant objects
    let restauranta = new Restaurant();
    restauranta.id = 1;
    restauranta.name = "CasaBella"; restauranta.meal = "Pasta";
restauranta.image = "..\\..\\..\\..\\assets\\image\\pasta.png";
```

```
restauranta.ratings = 10; restauranta.distance = 9; restauranta.arrivalTime =
15; restauranta.price = 115;
    this.Restaurants.push(restauranta);

    let restaurantb = new Restaurant();
    restaurantb.id = 2;
    restaurantb.name = "Nandos"; restaurantb.meal =
"Chicken";  restaurantb.image =
"..\\..\\..\\..\\assets\\image\\chicken.png"; restaurantb.ratings =
7;  restaurantb.distance = 5;   restaurantb.arrivalTime =
25; restaurantb.price = 60;
    this.Restaurants.push(restaurantb);

    let restaurantc = new Restaurant();
    restaurantc.id = 3;
    restaurantc.name = "Adega"; restaurantc.meal = "Prawns"; restaurantc.image
= "..\\..\\..\\..\\assets\\image\\prawns.png"; restaurantc.ratings = 10;
restaurantc.distance = 20; restaurantc.arrivalTime = 45; restaurantc.price =
180;
    this.Restaurants.push(restaurantc);

    let restaurantd = new Restaurant();
    restaurantd.id = 4;
    restaurantd.name = "Fireroom"; restaurantd.meal =
"Burger";  restaurantd.image =
"..\\..\\..\\..\\assets\\image\\burger.png"; restaurantd.ratings =
9;  restaurantd.distance =13;  restaurantd.arrivalTime =
30; restaurantd.price = 160;
    this.Restaurants.push(restaurantd);

    // Save restaurants to localStorage
    localStorage.setItem('Restaurants', JSON.stringify(this.Restaurants));
  }

  // Save order details to localStorage
  saveCart() {
    localStorage.setItem('Order', JSON.stringify(this.Order));
  }

  // Method to add item to cart and display success message
  additionToCart(res: Restaurant) {
    if (this.Order.Restaurant != null) {
      if (this.Order.Restaurant.id == res.id) {
        this.Order.Restaurant.orderAmount += 1;
      } else {
        res.orderAmount = 1;
        this.Order.Restaurant = res;
      }
```

```
    } else {
      res.orderAmount = 1;
      this.Order.Restaurant = res;
    }
    // Save order details to localStorage and display success message
    this.saveCart();
    this.successMessage();
  }
}
```

Search.page.html

```
<ion-header [translucent]="true" class="custom-header">
  <ion-toolbar class="custom-toolbar">
    <!-- Add any header content if needed -->
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true" class="custom-content">
  <ion-header collapse="condense">
    <ion-toolbar class="custom-toolbar">
      <ion-title size="large" class="custom-title">Search for a
restaurant</ion-title>
    </ion-toolbar>
  </ion-header>
  <ion-content>
    <ion-searchbar placeholder="Search Restaurants" (ionInput)="searchRest()"
[(ngModel)]="searchString" class="custom-searchbar"></ion-searchbar>
    <br>
    <ion-list class="custom-list">
      <ion-item *ngFor="let item of searchedRestaurants"
(click)="additionToCart(item)" class="custom-item">
        <ion-thumbnail slot="end">
          <img src="{{ item.image }}" class="custom-thumbnail" />
        </ion-thumbnail>
        <ion-label class="custom-label">
          <h5>{{ item.name }}</h5>
          <p>{{ item.meal }}</p>
          <ion-text color="primary" style="font-size: 15px;">
            <ion-icon name="fast-food-outline" size="small"></ion-icon>
            {{ item.ratings }}
          </ion-text>
        </ion-label>
        <ion-text slot="start" class="custom-text">
          <h6>{{ item.arrivalTime }} R {{ item.price }}</h6>
          <p>Distance: {{ item.distance }}km</p>
```

```
        </ion-text>
      </ion-item>
    </ion-list>
  </ion-content>
</ion-content>
```

Search.page.ts

```
import { Component, OnInit } from '@angular/core';
import { IonicModule } from '@ionic/angular';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { Restaurant } from '../restaurant';
import { orders } from '../order';
import { AlertController } from '@ionic/angular';

@Component({
  selector: 'app-search',
  templateUrl: './search.page.html',
  styleUrls: ['./search.page.scss'],
})
export class SearchPage implements OnInit {
  // Arrays to hold restaurant objects
  Restaurants: Restaurant[] = [];
  searchedRestaurants: Restaurant[] = [];
  // Object to hold order details
  Order!: orders;
  // Variable to hold search string
  searchString: string = "";

  // Constructor to inject dependencies
  constructor(private alertController: AlertController) { }

  // Initialization logic
  ngOnInit() {
    // Load restaurants and order details from localStorage
    this.Restaurants = JSON.parse(localStorage.getItem('Restaurants')!);
    this.Order = JSON.parse(localStorage.getItem('Order')!);
    // Set searchedRestaurants to all restaurants initially
    if (this.searchString == "") {
      this.searchedRestaurants = this.Restaurants;
    }
  }

  // Method to add item to cart and display success message
  additionToCart(res: Restaurant) {
    if (this.Order.Restaurant != null) {
```

```
      if (this.Order.Restaurant.id == res.id) {
        this.Order.Restaurant.orderAmount += 1;
      } else {
        res.orderAmount = 1;
        this.Order.Restaurant = res;
      }
    } else {
      res.orderAmount = 1;
      this.Order.Restaurant = res;
    }
    // Save order details to localStorage and display success message
    this.saveCart();
    this.successMessage();
  }

  // Method to filter restaurants based on search criteria
  searchRest() {
    // Clear previous search results
    this.searchedRestaurants = [];
    // Filter restaurants based on search criteria
    this.searchedRestaurants = this.Restaurants.filter(
      (foodItem) =>
        foodItem.name.toLowerCase().includes(this.searchString.toLowerCase())
||
        foodItem.meal.toLowerCase().includes(this.searchString.toLowerCase())
||
        foodItem.price == Number(this.searchString.toLowerCase()) ||
        foodItem.distance == Number(this.searchString.toLowerCase())
    );
  }

  // Method to save order details to localStorage
  saveCart() {
    localStorage.removeItem('Order');
    localStorage.setItem('Order', JSON.stringify(this.Order));
  }

  // Method to display success message in an alert dialog
  async successMessage() {
    const alert = await this.alertController.create({
      message: 'New item has been added to cart',
      buttons: ['OK']
    });
    await alert.present();
  }

}
```

Order.ts

```
import { Restaurant } from "./restaurant"

export class orders {
        // Property to hold restaurant details in an order

    Restaurant!: Restaurant;
}
```

Restaurant.ts

```
export class Restaurant {
    id : number = 0; name!: string ; meal: string = ""; price: number = 0;
arrivalTime: number =0; deliverFee: number = 50; orderAmount:number = 0;
ratings: number = 0; distance: number = 0; image: string = "";

}
```

Tabs-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { TabsPage } from './tabs.page';

const routes: Routes = [
  {
    path: '',
    component: TabsPage,
    children: [
    {
      path: 'home',
      loadChildren: () => import('./home/home.module').then( m =>
m.HomePageModule)
    },
    {
      path: 'cart',
      loadChildren: () => import('./cart/cart.module').then( m =>
m.CartPageModule)
    },
    {
      path: 'account',
      loadChildren: () => import('./account/account.module').then( m =>
m.AccountPageModule)
    },
    {
      path: 'search',
```

```
      loadChildren: () => import('./search/search.module').then( m =>
m.SearchPageModule)
    }
    ]
  },
  {
    path: '',
    redirectTo: '/tabs/home',
    pathMatch: 'full'
  },

];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class TabsPageRoutingModule {}
```

tabs.page.html

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="home">
    <ion-icon name="home-outline"></ion-icon>
      Home
    </ion-tab-button>
    <ion-tab-button tab="cart">
      <!-- <ion-badge>6</ion-badge> -->
      <ion-icon name="cart-outline"></ion-icon>
      Cart
    </ion-tab-button>
    <ion-tab-button tab="account">
      <ion-icon name="person-outline"></ion-icon>
      Account
    </ion-tab-button>
    <ion-tab-button tab="search">
      <ion-icon name="search-outline"></ion-icon>
      Search
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

App-routing.module.ts

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';
```

```
const routes: Routes = [

  {
    path: '',
    redirectTo: 'tabs',
    pathMatch: 'full'
  },
  {
    path: 'tabs',
    loadChildren: () => import('./pages/tabs/tabs.module').then( m =>
m.TabsPageModule)
  },
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Assignment 3

Backend

Controller

AuthenticationController.cs

```
using Assignment3_Backend.Models;
using Assignment3_Backend.ViewModels;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace Assignment3_Backend.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthenticationController : ControllerBase
    {
        private readonly UserManager<AppUser> _userManager;
        private readonly IUserClaimsPrincipalFactory<AppUser>
_claimsPrincipalFactory;
        private readonly IRepository _repository;
        private readonly IConfiguration _configuration;

        public AuthenticationController(UserManager<AppUser> userManager,
IUserClaimsPrincipalFactory<AppUser> claimsPrincipalFactory, IRepository
repository, IConfiguration configuration)
        {
            _repository = repository;
            _userManager = userManager;
            _claimsPrincipalFactory = claimsPrincipalFactory;
            _configuration = configuration;
        }

        [HttpPost]
        [Route("Register")]
        public async Task<IActionResult> Register(UserViewModel uvm)
        {
            var user = await _userManager.FindByIdAsync(uvm.emailaddress);

            if (user == null)
            {
                user = new AppUser
```

```
                {
                    Id = Guid.NewGuid().ToString(),
                    UserName = uvm.emailaddress,
                    Email = uvm.emailaddress
                };

                var result = await _userManager.CreateAsync(user,
uvm.password);

                if (result.Errors.Count() > 0) return
StatusCode(StatusCodes.Status500InternalServerError, "Internal Server Error.
Please contact support.");
            }
            else
            {
                return Forbid("Account already exists.");
            }

            return Ok();
        }

        [HttpPost]
        [Route("Login")]
        public async Task<ActionResult> Login(UserViewModel uvm)
        {
            var user = await _userManager.FindByNameAsync(uvm.emailaddress);

            if (user != null && await _userManager.CheckPasswordAsync(user,
uvm.password))
            {
                try
                {
                    var principal = await
_claimsPrincipalFactory.CreateAsync(user);
                    return GenerateJWTToken(user);
                }
                catch (Exception)
                {
                    return
StatusCode(StatusCodes.Status500InternalServerError, "Internal Server Error.
Please contact support.");
                }
            }
            else
            {
                return NotFound("Does not exist");
            }
        }
```

```
        [HttpGet]
        private ActionResult GenerateJWTToken(AppUser user)
        {
            // Create JWT Token
            var claims = new[]
            {
                new Claim(JwtRegisteredClaimNames.Sub, user.Email),
                new Claim(JwtRegisteredClaimNames.Jti,
Guid.NewGuid().ToString()),
                new Claim(JwtRegisteredClaimNames.UniqueName, user.UserName)
            };

            var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Tokens:Key"]));
            var credentials = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                _configuration["Tokens:Issuer"],
                _configuration["Tokens:Audience"],
                claims,
                signingCredentials: credentials,
                expires: DateTime.UtcNow.AddHours(3)
            );

            return Created("", new
            {
                token = new JwtSecurityTokenHandler().WriteToken(token),
                user = user.UserName
            });
        }

    }
}
```

ReportController.cs

```
using Assignment3_Backend.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Dynamic;

namespace Assignment3_Backend.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
```

```
    public class ReportController : ControllerBase
    {

        private readonly IRepository _repository;
        public ReportController(IRepository repository)
        {
            _repository = repository;
        }

        [HttpGet]
        [Route("ProductsReport")]
        public async Task<ActionResult<dynamic>> ProductsReport()
        {
            try
            {
                List<dynamic> productsreport = new List<dynamic>();

                var results = await _repository.GetProductsReportAsync();

                dynamic brands = results
                            .GroupBy(p => p.Brand.Name)
                            .Select(b => new
                            {
                                Key = b.Key,
                                ProductCount = b.Count()
                            ,
                                ProductTotalCost = Math.Round((double)b.Sum(p
=> p.Price), 2)
                            ,
                                ProductAverageCost =
Math.Round((double)b.Average(p => p.Price), 2)
                            });

                dynamic productTypes = results
                            .GroupBy(p => p.ProductType.Name)
                            .Select(pt => new
                            {
                                Key = pt.Key,
                                ProductCount = pt.Count()
                            ,
                                ProductTotalCost =
Math.Round((double)pt.Sum(p => p.Price), 2)
                            ,
                                ProductAverageCost =
Math.Round((double)pt.Average(p => p.Price), 2)
                            });

                dynamic productList = results
```

```
                    .GroupBy(p => new { BrandName = p.Brand.Name,
ProductTypeName = p.ProductType.Name, ProductName = p.Name })
                    .Select(p => new
                    {
                        p.Key.BrandName,
                        p.Key.ProductTypeName,
                        p.Key.ProductName,
                        ProductPrice = Math.Round((double)p.Sum(x => x.Price),
2)
                    });

                productsreport.Add(brands);
                productsreport.Add(productTypes);
                productsreport.Add(productList);


                return productsreport;
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError,
"Internal Server Error. Please contact support.");
            }
        }
    }
}
```

StoreController.cs

```
using Assignment3_Backend.Models;
using Assignment3_Backend.ViewModels;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;
using System;
using System.Globalization;
using System.IdentityModel.Tokens.Jwt;
using System.IO;
using System.Linq;
using System.Net.Http.Headers;
using System.Runtime.InteropServices;
using System.Security.Claims;
```

```
using System.Text;
using System.Threading.Tasks;

namespace Assignment3_Backend.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class StoreController : ControllerBase
    {
        private readonly IRepository _repository;
        public StoreController(IRepository repository)
        {
            _repository = repository;
        }

        [HttpGet]
        [Route("ProductListing")]
        public async Task<ActionResult> ProductListing()
        {
            try
            {
                var results = await _repository.GetProductsAsync();

                dynamic products = results.Select(p => new
                {
                    p.ProductId,
                    p.Price,
                    ProductTypeName = p.ProductType.Name,
                    BrandName = p.Brand.Name,
                    p.Name,
                    p.Description,
                    p.DateCreated,
                    p.DateModified,
                    p.IsActive,
                    p.IsDeleted,
                    p.Image
                });

                return Ok(products);
            }
            catch (Exception)
            {

                return StatusCode(StatusCodes.Status500InternalServerError,
"Internal Server Error. Please contact support.");
            }
        }
```

```
        [HttpPost, DisableRequestSizeLimit]
        [Route("AddProduct")]
        public async Task<IActionResult> AddProduct([FromForm] IFormCollection
formData)
        {
            try
            {
                var formCollection = await Request.ReadFormAsync();

                var file = formCollection.Files.First();

                if (file.Length > 0)
                {

                    using (var ms = new MemoryStream())
                    {
                        file.CopyTo(ms);
                        var fileBytes = ms.ToArray();
                        string base64 = Convert.ToBase64String(fileBytes);

                        string price = formData["price"];
                        decimal num = decimal.Parse(price.Replace(".", ","));

                        var product = new Product
                        {
                            Price = num
                            ,
                            Name = formData["name"]
                            ,
                            Description = formData["description"]
                            ,
                            BrandId = Convert.ToInt32(formData["brand"])
                            ,
                            ProductTypeId =
Convert.ToInt32(formData["producttype"])
                            ,
                            Image = base64
                            ,
                            DateCreated = DateTime.Now
                        };


                        _repository.Add(product);
                        await  _repository.SaveChangesAsync();

                    }

                    return Ok();
                }
```

```
                else
                {
                    return BadRequest();
                }
            }
            catch (Exception ex)
            {
                return StatusCode(500, $"Internal server error: {ex}");
            }
        }


        [HttpGet]
        [Route("Brands")]
        public async Task<ActionResult> Brands()
        {
            try
            {
                var results = await _repository.GetBrandsAsync();

                return Ok(results);
            }
            catch (Exception)
            {

                return StatusCode(StatusCodes.Status500InternalServerError,
"Internal Server Error. Please contact support.");
            }
        }


        [HttpGet]
        [Route("ProductTypes")]
        public async Task<ActionResult> ProductTypes()
        {
            try
            {
                var results = await _repository.GetProductTypesAsync();

                return Ok(results);
            }
            catch (Exception)
            {

                return StatusCode(StatusCodes.Status500InternalServerError,
"Internal Server Error. Please contact support.");
            }
        }
```

```
    }
}
```

Factory

AppUserClaimsPrincipalFactory.cs

```
using Assignment3_Backend.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.Options;

namespace Assignment3_Backend.Factory
{
    public class AppUserClaimsPrincipalFactory:
UserClaimsPrincipalFactory<AppUser, IdentityRole>
    {
        public AppUserClaimsPrincipalFactory(UserManager<AppUser> userManager,
        RoleManager<IdentityRole> roleManager,
        IOptions<IdentityOptions> optionsAccessor)
        : base(userManager, roleManager, optionsAccessor)
        {
        }

    }
}
```

Models

AppDBContext.cs

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace Assignment3_Backend.Models
{
    public class AppDbContext:IdentityDbContext<AppUser>
    {

        public AppDbContext(DbContextOptions<AppDbContext> options) :
base(options)
        {
        }

        public DbSet<Brand> Brands { get; set; }
        public DbSet<Product> Products { get; set; }
        public DbSet<ProductType> ProductTypes { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
```

```
        {
            base.OnModelCreating(modelBuilder);
        }
    }
}
```

AppUser.cs

```
using Microsoft.AspNetCore.Identity;

namespace Assignment3_Backend.Models
{
    public class AppUser: IdentityUser
    {
    }
}
```

BaseEntity.cs

```
namespace Assignment3_Backend.Models
{
    public abstract class BaseEntity
    {
        public string Name { get; set; }
        public string Description { get; set; }
        public DateTime DateCreated { get; set; }
        public DateTime? DateModified { get; set; }
        public bool IsActive { get; set; } = true;
        public bool IsDeleted { get; set; } = false;
    }
}
```

Brand.cs

```
namespace Assignment3_Backend.Models
{
    public class Brand : BaseEntity
    {
        public int BrandId { get; set; }
        public virtual ICollection<Product> Products { get; set; }
    }
}
```

IRepository.cs

```
namespace Assignment3_Backend.Models
{
    public interface IRepository
    {
        Task<bool> SaveChangesAsync();
        Task<Product[]> GetProductsAsync();
        Task<ProductType[]> GetProductTypesAsync();
        Task<Brand[]> GetBrandsAsync();

        Task<Product[]> GetProductsReportAsync();

        void Add<T>(T entity) where T : class;
    }
}
```

Product.cs

```
using Microsoft.EntityFrameworkCore.Metadata.Internal;
using System.ComponentModel.DataAnnotations.Schema;

namespace Assignment3_Backend.Models
{
    public class Product : BaseEntity
    {
        public int ProductId { get; set; }
        public decimal Price { get; set; }

        public string? Image { get; set; }
        public int BrandId { get; set; }
        public int ProductTypeId { get; set; }

        public ProductType ProductType { get; set; }
        public Brand Brand { get; set; }
    }
}
```

ProductType.cs

```
namespace Assignment3_Backend.Models
{
    public class ProductType : BaseEntity
    {
        public int ProductTypeId { get; set; }

        public virtual ICollection<Product> Products { get; set; }
    }
}
```

Repository.cs

```
using Microsoft.EntityFrameworkCore;

namespace Assignment3_Backend.Models
{
    public class Repository:IRepository
    {
        private readonly AppDbContext _appDbContext;

        public Repository(AppDbContext appDbContext)
        {
            _appDbContext = appDbContext;
        }

        public void Add<T>(T entity) where T : class
        {
            _appDbContext.Add(entity);
        }
        public async Task<bool> SaveChangesAsync()
        {
            return await _appDbContext.SaveChangesAsync() > 0;
        }

        public async Task<Product[]> GetProductsAsync()
        {
            IQueryable<Product> query = _appDbContext.Products.Include(p =>
p.Brand).Include(p => p.ProductType);

            return await query.ToArrayAsync();
        }

        public async Task<Brand[]> GetBrandsAsync()
        {
            IQueryable<Brand> query = _appDbContext.Brands.OrderBy(p =>
p.Name); ;

            return await query.ToArrayAsync();
        }

        public async Task<ProductType[]> GetProductTypesAsync()
        {
            IQueryable<ProductType> query = _appDbContext.ProductTypes;

            return await query.ToArrayAsync();
        }

        public async Task<Product[]> GetProductsReportAsync()
        {
```

```
            IQueryable<Product> query = _appDbContext.Products.Include(p =>
p.Brand).Include(p => p.ProductType).Where(p=> p.IsActive == true).OrderBy(p
=> p.Brand.Name);

            return await query.ToArrayAsync();
        }
    }
}
```

View Model

ProductViewModel.cs

```
namespace Assignment3_Backend.ViewModels
{
    public class ProductViewModel
    {
        public decimal price { get; set; }
        public int producttype { get; set; }
        public int brand { get; set; }
        public string description { get; set; }
        public string name { get; set; }
    }
}
```

UserViewModel.cs

```
namespace Assignment3_Backend.ViewModels
{
    public class UserViewModel
    {
        public string emailaddress { get; set; }
        public string password { get; set; }

    }
}
```

Frontend Angular

Authentication

Login.component.html

```
<div class="login-wrapper" fxLayout="row" fxLayoutAlign="center center">
    <mat-card class="box" *ngIf="!isLoading">
      <mat-card-header>
        <mat-card-title>Log in</mat-card-title>
      </mat-card-header>
      <form class="form" [formGroup]="loginFormGroup">
        <mat-card-content>
          <mat-form-field class="full-width">
            <mat-label>Username</mat-label>
            <input matInput placeholder="Enter the User's Email address"
formControlName="emailaddress">
          </mat-form-field>
          <mat-form-field class="full-width">
            <mat-label>Password</mat-label>
            <input matInput type="password" placeholder="Enter the User's
Password" formControlName="password">
          </mat-form-field>
        </mat-card-content>
        <button mat-stroked-button color="primary" class="btn-block"
(click)="LoginUser()">Log in</button>
        <div>Don't have an account? Register <a
[routerLink]="['../register']">here</a></div>
      </form>
    </mat-card>
    <mat-progress-spinner mode="indeterminate" value="50" *ngIf="isLoading">
    </mat-progress-spinner>
  </div>
```

Login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { APIService } from '../services/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { MatSnackBar } from '@angular/material/snack-bar';
import { HttpErrorResponse } from '@angular/common/http';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {
```

```
  loginFormGroup: FormGroup = this.fb.group({
    emailaddress: ['', [Validators.required, Validators.email]],
    password: ['', Validators.required],
  })

  isLoading:boolean = false

  constructor(private router: Router, private apiService: APIService, private
fb: FormBuilder, private snackBar: MatSnackBar) { }

  ngOnInit(): void {
  }

  async LoginUser(){
    if(this.loginFormGroup.valid)
    {
      this.isLoading = true

      await
this.apiService.LoginUser(this.loginFormGroup.value).subscribe(result => {
        localStorage.setItem('User', JSON.stringify(result))
        this.loginFormGroup.reset();
        this.router.navigateByUrl('productListing');
      })
    }
  }

}
```

Register.component.html

```
<div class="login-wrapper" fxLayout="row" fxLayoutAlign="center center">
    <mat-card class="box">
      <mat-card-header>
        <mat-card-title>Register</mat-card-title>
      </mat-card-header>
      <form class="form" [formGroup]="registerFormGroup">
        <mat-card-content>
          <mat-form-field class="full-width">
            <mat-label>Email Address</mat-label>
            <input matInput placeholder="Enter a valid Email address"
formControlName="emailaddress">
          </mat-form-field>
          <mat-form-field class="full-width">
              <mat-label>Password</mat-label>
            <input type="password" matInput placeholder="Enter between 6 to 16
characters" formControlName="password">
```

```
          </mat-form-field>
        </mat-card-content>
        <button mat-stroked-button color="primary" class="btn-block"
(click)="RegisterUser()">Register</button>
      </form>
    </mat-card>
  </div>
```

Register.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { APIService } from '../services/api.service';
import { MatSnackBar } from '@angular/material/snack-bar';
import { HttpErrorResponse } from '@angular/common/http';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.scss']
})
export class RegisterComponent implements OnInit {

  registerFormGroup: FormGroup = this.fb.group({
    emailaddress: ['', [Validators.required, Validators.email]],
    password: ['', [Validators.required, Validators.minLength(6),
Validators.maxLength(16)]],
  })

  constructor(private router: Router, private apiService: APIService, private
fb: FormBuilder, private snackBar: MatSnackBar) {



  }

  ngOnInit(): void {
  }

  RegisterUser(){

    if(this.registerFormGroup.valid)
    {
        this.apiService.RegisterUser(this.registerFormGroup.value).subscribe((
) => {
        this.registerFormGroup.reset();
        this.router.navigate(['']).then((navigated: boolean) => {
```

```
          if(navigated) {
            this.snackBar.open(`Registered successfully`, 'X', {duration:
5000});
          }
       });
      })
    }
  }

}
```

Products

Add-products.component.html

```
<div class="login-wrapper" fxLayout="row" fxLayoutAlign="center center">
    <mat-card class="box" >
      <mat-card-header>
        <mat-card-title>Add Product</mat-card-title>
      </mat-card-header>
      <form class="form" [formGroup]="productForm" (submit)="onSubmit()"
enctype="multipart/form-data">
        <mat-card-content>
          <mat-form-field class="full-width">
            <mat-label>Name</mat-label>
            <input matInput placeholder="Enter the Product Name"
formControlName="name">
          </mat-form-field>
          <mat-form-field class="full-width">
            <mat-label>Price</mat-label>
            <input type="number" step="0.01" matInput placeholder="Enter the
Product Price" formControlName="price">
          </mat-form-field>
          <mat-form-field>
            <mat-label>Brand</mat-label>
            <mat-select formControlName="brand">
                <mat-option *ngFor="let item of
brandsData"  [value]="item.brandId">{{item.name}}</mat-option>
            </mat-select>
        </mat-form-field>
        <mat-form-field>
            <mat-label>Product Type</mat-label>
            <mat-select formControlName="producttype" >
                <mat-option *ngFor="let item of
productTypesData"  [value]="item.productTypeId">{{item.name}}</mat-option>
            </mat-select>
        </mat-form-field>
        <mat-form-field class="full-width">
            <mat-label>Product Description</mat-label>
```

```
            <textarea matInput formControlName="description"> </textarea>
          </mat-form-field>
          <div>
            <input formControlName="file" type="file" id="file" #file
placeholder="Choose file" (change)="uploadFile(file.files)"
style="display:none;">
            <button mat-stroked-button color="primary"
(click)="file.click()">Upload File</button>
            <span *ngIf="fileNameUploaded.length > 0">
{{fileNameUploaded}}</span>
          </div>
        </mat-card-content>
        <button mat-stroked-button color="primary" class="btn-block"
>Submit</button>

      </form>
    </mat-card>
  </div>
```

Add-product.component.ts

```
import { Component, OnInit } from '@angular/core';
import { APIService } from '../services/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Brands } from '../shared/brands';
import { ProductTypes } from '../shared/product-types';


@Component({
  selector: 'app-add-products',
  templateUrl: './add-products.component.html',
  styleUrls: ['./add-products.component.scss']
})
export class AddProductsComponent implements OnInit {
   formData = new FormData();
   brandsData:Brands[]=[]
   productTypesData:ProductTypes[]=[]
   fileNameUploaded = ''

  productForm: FormGroup = this.fb.group({
    name: ['', Validators.required],
    file: ['', Validators.required],
    price: ['', Validators.required],
    brand: [null, Validators.required],
    producttype: [null, Validators.required],
    description: ['', Validators.required]
```

```
  })

  constructor(private apiService: APIService, private fb: FormBuilder, private
router: Router, private snackBar: MatSnackBar) { }

  ngOnInit(): void {
    this.GetBrands()
    this.GetProductTypes()
  }

GetBrands(){
  this.apiService.getBrands().subscribe(result => {
    let brandList:any[] = result
    brandList.forEach((element) => {
      this.brandsData.push(element)
    });
  });
}

GetProductTypes(){
   this.apiService.getProductTypes().subscribe(result => {
   let productTypeList:any[] = result
   productTypeList.forEach((element) => {
      this.productTypesData.push(element)
    });
  });
}

  uploadFile = (files: any) => {
    let fileToUpload = <File>files[0];
    this.formData.append('file', fileToUpload, fileToUpload.name);
    this.fileNameUploaded = fileToUpload.name
  }

  onSubmit() {
    if(this.productForm.valid)
    {
      this.formData.append('name', this.productForm.get('name')!.value);
      this.formData.append('price', this.productForm.get('price')!.value);
      this.formData.append('description',
this.productForm.get('description')!.value);
      this.formData.append('brand', this.productForm.get('brand')!.value);
      this.formData.append('producttype',
this.productForm.get('producttype')!.value);

      this.apiService.addProduct(this.formData).subscribe(() => {
        this.clearData()
```

```
        this.router.navigateByUrl('productListing').then((navigated: boolean)
=> {
          if(navigated) {
            this.snackBar.open(this.productForm.get('name')!.value + ` created
successfully`, 'X', {duration: 5000});
          }
       });
      });
    }
  }

  clearData(){
    this.formData.delete("file");
    this.formData.delete("name");
    this.formData.delete("price");
    this.formData.delete("description");
    this.formData.delete("brand");
    this.formData.delete("producttype");
  }

}
```

Product-listing.component.html

```
<div class="mat-elevation-z8">
    <mat-form-field appearance="fill">
        <mat-label>Filter</mat-label>
        <input matInput (keyup)="applyFilter($event)" placeholder="start
typing..." #input>
    </mat-form-field>

    <table mat-table [dataSource]="dataSource" matSort>

        <ng-container matColumnDef="image">
            <th mat-header-cell *matHeaderCellDef></th>
            <td mat-cell *matCellDef="let element"> <img
src='{{element.image}}'>  </td>
          </ng-container>

      <ng-container matColumnDef="name">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Name </th>
        <td mat-cell *matCellDef="let element"> {{element.name}} </td>
      </ng-container>

      <ng-container matColumnDef="price">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Price </th>
```

```
        <td mat-cell *matCellDef="let element"> {{element.price |
currency:'$':'symbol':'1.2-2'}} </td>
      </ng-container>

      <ng-container matColumnDef="brand">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Brand </th>
        <td mat-cell *matCellDef="let element"> {{element.brandName}} </td>
      </ng-container>

      <ng-container matColumnDef="productTypeName">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Product Type
</th>
        <td mat-cell *matCellDef="let element"> {{element.productTypeName}}
</td>
      </ng-container>

      <ng-container matColumnDef="description">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Description
</th>
        <td mat-cell *matCellDef="let element"> {{element.description}} </td>
      </ng-container>

      <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
      <tr mat-row class="mat-row" *matRowDef="let row; columns:
displayedColumns;"></tr>

      <tr class="mat-row" *matNoDataRow>
        <td class="mat-cell" colspan="4">No data matching the filter
"{{input.value}}"</td>
      </tr>
    </table>
    <mat-paginator [pageSize]="10" [pageSizeOptions]="[3, 5, 10]"
showFirstLastButtons> </mat-paginator>
  </div>
```

Product-listing.component.ts

```
import { AfterViewInit, Component, OnInit, ViewChild } from '@angular/core';
import { MatTableDataSource } from '@angular/material/table';
import { ProductListing } from '../shared/product-listing';
import { APIService } from '../services/api.service';
import { MatPaginator } from '@angular/material/paginator';
import { MatSort } from '@angular/material/sort';

@Component({
  selector: 'app-product-listing',
  templateUrl: './product-listing.component.html',
  styleUrls: ['./product-listing.component.scss']
```

```
})
export class ProductListingComponent implements AfterViewInit, OnInit {
  displayedColumns: string[] = ['image', 'name', 'price','brand',
'productTypeName', 'description'];
  dataSource = new MatTableDataSource<ProductListing>();
  constructor(private apiService: APIService) { }

  @ViewChild(MatPaginator) paginator!: MatPaginator;
  @ViewChild(MatSort) sort!: MatSort;

  ngOnInit(): void {
    this.apiService.getProducts().subscribe((products:any) =>
{this.dataSource.data = products});
  }

  ngAfterViewInit() {
    this.dataSource.paginator = this.paginator;
    this.dataSource.sort = this.sort;
  }

  applyFilter(event: Event) {
    const filterValue = (event.target as HTMLInputElement).value;
    this.dataSource.filter = filterValue.trim().toLowerCase();
  }

}
```

Reporting

Product.component.html

```
<div *ngIf="chartsLoaded">
    <div class="chart-container">
        <div class="chart-item">
            <canvas baseChart
                [data]="brandData"
                [type]="'bar'"
                [options]="brandChartOptions"
            >
            </canvas>
        </div>
        <div class="chart-item">
            <canvas baseChart
                [data]="productTypeData"
                [type]="'bar'"
                [options]="productTypeChartOptions"
            >
            </canvas>
        </div>
```

```
    </div>

        <h1>Active Products Report</h1>
    <mat-accordion multi="true">
        <mat-expansion-panel *ngFor="let brand of groupedProducts | keyvalue">
          <mat-expansion-panel-header>
            {{ brand.key }}
          </mat-expansion-panel-header>

          <mat-list>
            <mat-list-item  *ngFor="let productType of brand.value |
keyvalue">
              <h4 mat-line>{{ productType.key }}</h4>
              <div *ngFor="let product of productType.value">
                <p mat-line>{{ product['productName'] }} - {{
product['productPrice'] | currency }}
                </p>
              </div>
            </mat-list-item>
          </mat-list>

        </mat-expansion-panel>
    </mat-accordion>
</div>
```

Products.component.ts

```
import { Component, OnInit } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { ChartData, ChartOptions } from 'chart.js';
import { APIService } from '../services/api.service';
import { HttpErrorResponse } from '@angular/common/http';

@Component({
  selector: 'app-products',
  templateUrl: './products.component.html',
  styleUrls: ['./products.component.scss']
})
export class ProductsComponent implements OnInit {

  chartsLoaded:boolean = false;
  brands:any[] = []
  productTypes:any[] = []
  products:any[] = []
  groupedProducts: { [brandName: string]: { [productType: string]: any[] } } =
{};
  constructor(private apiService: APIService, private snackBar: MatSnackBar) {
}
```

```
ngOnInit(): void {
  this.GenerateProductReport()
  console.log(this.brands)
  console.log(this.productTypes)
  console.log(this.products)
  console.log(this.groupedProducts)

}

// Brand Chart
brandData: ChartData<'bar'> = {
  labels: [],
  datasets: [
    { data: [], label:'Brands', backgroundColor: '#90E0EF' },
  ],
};

brandChartOptions: ChartOptions = {
  responsive: true,
  scales: {
    x: {
      position: 'bottom',
      grid: {
        display: false
      }
    },
    y: {
      ticks: {
        stepSize: 1
      },
      grid:  {
        display: false
      }
    }
  },
  plugins: {
    title: {
      display: true,
      text: 'Product Count by Brands',
    },
  },
};

// Product Type Chart
productTypeData: ChartData<'bar'> = {
  labels: [],
  datasets: [
```

```
    { data: [], label: 'Product Types', backgroundColor: '#00B4D8' },
  ],
};

productTypeChartOptions: ChartOptions = {
  responsive: true,
  scales: {
    x: {
      grid:  {
        display: false
      }
    },
    y: {
      ticks: {
        stepSize: 1
      },
      grid:  {
        display: false
      }
    }
  },
  plugins: {
    title: {
      display: true,
      text: 'Product Count by Product Type',
    },
  },
};


GenerateProductReport()
{
  this.apiService.GenerateProductReport().subscribe(result => {
    let brandData:any[] = result[0]
    let productTypeData:any[] = result[1]
    let productList:any[] = result[2]
    let brandProductTypesData:any[]= result[3]

    brandData.forEach((element) => {
      this.brandData.labels?.push(element.key)
      this.brandData.datasets[0].data.push(element.productCount)
      this.brands.push(element)
    });

    productTypeData.forEach((element) => {
      this.productTypeData.labels?.push(element.key)
      this.productTypeData.datasets[0].data.push(element.productCount)
      this.productTypes.push(element)
    });
```

```
      productList.forEach((element) => {
        this.products.push(element)
      });

      this.chartsLoaded = true;
      this.groupProducts()
    }, (response: HttpErrorResponse) => {
        if (response.status === 500){
          this.snackBar.open(response.error, 'X', {duration: 5000});
        }
    })
  }

  groupProducts() {
    for (const product of this.products) {

      if (!this.groupedProducts[product.brandName]) {
        this.groupedProducts[product.brandName] = {};
      }
      if (!this.groupedProducts[product.brandName][product.productTypeName]) {
        this.groupedProducts[product.brandName][product.productTypeName] = [];
      }
      this.groupedProducts[product.brandName][product.productTypeName].push(pr
oduct);
    }
  }

}
```

Services

Api.service.ts

```
import { Injectable } from '@angular/core';
import { Observable, map, of} from 'rxjs';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { RegisterUser } from '../shared/register-user';
import { LoginUser } from '../shared/login-user';
import { User } from '../shared/user';
import { Product } from '../shared/product';

@Injectable({
  providedIn: 'root'
})

export class APIService {

apiUrl = 'http://localhost:5240/api/'
```

```
httpOptions ={
  headers: new HttpHeaders({
    ContentType: 'application/json'
  })
}
  constructor(private httpClient: HttpClient) {
  }

  RegisterUser(registerUser: RegisterUser){
    return this.httpClient.post(`${this.apiUrl}Authentication/Register`,
registerUser, this.httpOptions)
  }

  getProducts() {
    return this.httpClient.get(`${this.apiUrl}Store/ProductListing`)
    .pipe(map(result => result))
  }

  LoginUser(loginUser: LoginUser){
    return this.httpClient.post<User>(`${this.apiUrl}Authentication/Login`,
loginUser, this.httpOptions)
  }

  addProduct(file:FormData){

    return this.httpClient.post(`${this.apiUrl}Store/AddProduct`, file)
  }

  getBrands(): Observable<any>
  {
    return this.httpClient.get(`${this.apiUrl}Store/Brands`)
    .pipe(map(result => result))
  }

  getProductTypes(): Observable<any>
  {
    return this.httpClient.get(`${this.apiUrl}Store/ProductTypes`)
    .pipe(map(result => result))
  }

  GenerateProductReport(): Observable<any>{
    return this.httpClient.get(`${this.apiUrl}Report/ProductsReport`)
    .pipe(map(result => result))
  }
}
```

Shared

Brand.ts

```
export interface Brands {
    brandId: number;
    name: string;
}
```

Login-user.ts

```
export interface LoginUser {
    userName: string;
    password: string;
}
```

Material.module.ts

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

import { MatAutocompleteModule } from '@angular/material/autocomplete';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatNativeDateModule } from '@angular/material/core'
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatRadioModule } from '@angular/material/radio';
import { MatSelectModule } from '@angular/material/select';
import { MatSliderModule } from '@angular/material/slider';
import { MatSlideToggleModule } from '@angular/material/slide-toggle';
import { MatMenuModule } from '@angular/material/menu';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatCardModule } from '@angular/material/card';
import { MatDividerModule } from '@angular/material/divider';
import { MatExpansionModule } from '@angular/material/expansion';
import { MatGridListModule } from '@angular/material/grid-list';
import { MatListModule } from '@angular/material/list';
import { MatStepperModule } from '@angular/material/stepper';
import { MatTabsModule } from '@angular/material/tabs';
import { MatTreeModule } from '@angular/material/tree';
import { MatButtonModule } from '@angular/material/button';
import { MatButtonToggleModule } from '@angular/material/button-toggle';
import { MatBadgeModule } from '@angular/material/badge';
import { MatChipsModule } from '@angular/material/chips';
import { MatIconModule } from '@angular/material/icon';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatProgressBarModule } from '@angular/material/progress-bar';
```

```
import { MatRippleModule } from '@angular/material/core';
import { MatBottomSheetModule } from '@angular/material/bottom-sheet';
import { MatDialogModule } from '@angular/material/dialog';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { MatTooltipModule } from '@angular/material/tooltip';
import { MatPaginatorModule } from '@angular/material/paginator';
import { MatSortModule } from '@angular/material/sort';
import { MatTableModule } from '@angular/material/table';



@NgModule({
  declarations: [],
  exports: [
    MatAutocompleteModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatNativeDateModule,
    MatFormFieldModule,
    MatInputModule,
    MatRadioModule,
    MatSelectModule,
    MatSliderModule,
    MatSlideToggleModule,
    MatMenuModule,
    MatSidenavModule,
    MatToolbarModule,
    MatCardModule,
    MatDividerModule,
    MatExpansionModule,
    MatGridListModule,
    MatListModule,
    MatStepperModule,
    MatTabsModule,
    MatTreeModule,
    MatButtonModule,
    MatButtonToggleModule,
    MatBadgeModule,
    MatChipsModule,
    MatIconModule,
    MatProgressSpinnerModule,
    MatProgressBarModule,
    MatRippleModule,
    MatBottomSheetModule,
    MatDialogModule,
    MatSnackBarModule,
    MatTooltipModule,
    MatPaginatorModule,
    MatSortModule,
```

```
    MatTableModule
  ]
})
export class MaterialModule { }
```

product-listing.ts

```
export interface ProductListing {
    image: string;
    name: string;
    price: number;
    description: string;
    brand: string;
    productTypeName: string;
}
```

Product-type.ts

```
export interface ProductTypes {
    productTypeId: number;
    name: string;
}
```

Product.ts

```
export interface Product {
    name: string;
    price: number;
    description: string;
    brand: number;
    producttype: number;
}
```

Register-user.ts

```
export interface RegisterUser {
    emailaddress: string;
    password: string;
}
```

User.ts

```
export interface User {
    token?: string
    user?: string
}
```

App-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './authentication/login.component';
import { RegisterComponent } from './authentication/register.component';
import { ProductListingComponent } from './products/product-
listing.component';
import { AddProductsComponent } from './products/add-products.component';
import { ProductsComponent } from './reporting/products.component';

const routes: Routes = [
  {path: 'login', component:LoginComponent},
  {path: 'register', component:RegisterComponent},
  {path:'productListing', component:ProductListingComponent},
  {path:'addProduct', component:AddProductsComponent},
  {path:'productReporting', component:ProductsComponent},
  {path: '', redirectTo: 'login', pathMatch:'full'}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

App.component.html

```
<mat-toolbar color="primary">
  <mat-toolbar-row>
    <!-- <button mat-icon-button *ngIf="isLoggedIn">
      <mat-icon (click)="sidenav.toggle()">menu</mat-icon>
    </button> -->
    <h1>INF354 Assignment 3</h1>

  </mat-toolbar-row>

</mat-toolbar>
<mat-sidenav-container>
  <mat-sidenav #sidenav mode="side" opened="true" class="side-container"
*ngIf="isLoggedIn">
    <mat-nav-list>

      <a mat-list-item [routerLink]="'/productListing'"> Product Listing </a>
      <a mat-list-item [routerLink]="'/addProduct'"> Add Product </a>
      <a mat-list-item [routerLink]="'/productReporting'"> Product Reporting
</a>
      <a mat-list-item (click)="logout()"> Logout </a>
```

```
    </mat-nav-list>
  </mat-sidenav>
  <mat-sidenav-content class="content-container">
    <div style="height: 88vh;">
      <router-outlet></router-outlet>
    </div>
  </mat-sidenav-content>
</mat-sidenav-container>
```

App.component.scss

```
.side-container {
    min-width: 200px;

  }

  .content-container {
    padding: 10px;

  }
```

App.component.ts

```
import { Component,AfterContentChecked, ViewChild } from '@angular/core';
import { MatSidenav } from '@angular/material/sidenav';
import { Router } from '@angular/router';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent implements AfterContentChecked {
  @ViewChild('sidenav', {static:true}) sidenav!: MatSidenav;

  isLoggedIn = false;

  constructor(private router: Router) {}

  toggleSidenav(){
    this.sidenav.toggle();
  }

  ngAfterContentChecked(){
    if(localStorage.getItem('User'))
    {
      this.isLoggedIn = true;
```

```
    }
    else{
      this.isLoggedIn = false;
    }
  }

  logout(){
    if(localStorage.getItem('User'))
    {
      localStorage.removeItem('User')
      this.router.navigateByUrl('login');
    }
  }

}
```

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Faculty of Engineering, Built Environment and Information Technology

Fakulteit Ingenieurswese, Bou-omgewing en Inligtingtegnologie / Lefapha la Boetšenere, Tikologo ya Kago le Theknolotši ya Tshedimošo

DEPARTMENT OF INFORMATICS

INF 354

EXAMINATION

DATE: 2023-06-23

Examiners	:	Mr Ridewaan Hanslo	**Time**	:	180 min
	:	Dr Timothy Adeliyi			
Moderator / External Examiner	:	Dr Chinedu Okonkwo University of the Johannesburg	**Marks**	:	95

Student Number								Surname	Initials

Question Section	Module outcomes (as in Study Guide)							Marks allocated	Maximum mark
	MO1	MO2	MO3	MO4	MO5	MO6	MO7		
Section A			X		X				**30**
Section B	X					X			**25**
Section C		X							**25**
Section D	X			X			X		**15**
Total									**95**

Instructions

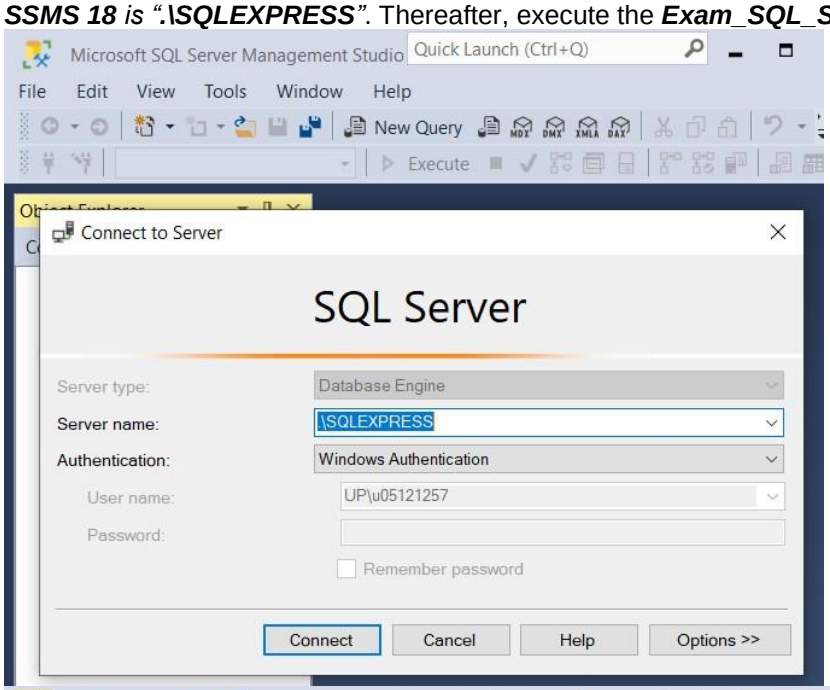
1. This paper consists of 4 sections with one question per section (sub-sets of instructions) each.
2. Each section relates to small semi-complete programs that need to be updated or finalised.
3. Each question relates to file(s) in the semi-complete programs that you need to update or finalise.
4. Each sub-sets of instructions relate to activities and tasks in one of the files.
5. Answer all the questions – there are no optional questions.
6. Please read all questions, instructions and sub-sets of tasks very carefully.
7. After completing work on a relevant question, please upload the file(s) required to be uploaded to the correct upload area. In other words, each section will mention what files to upload and how to upload them.

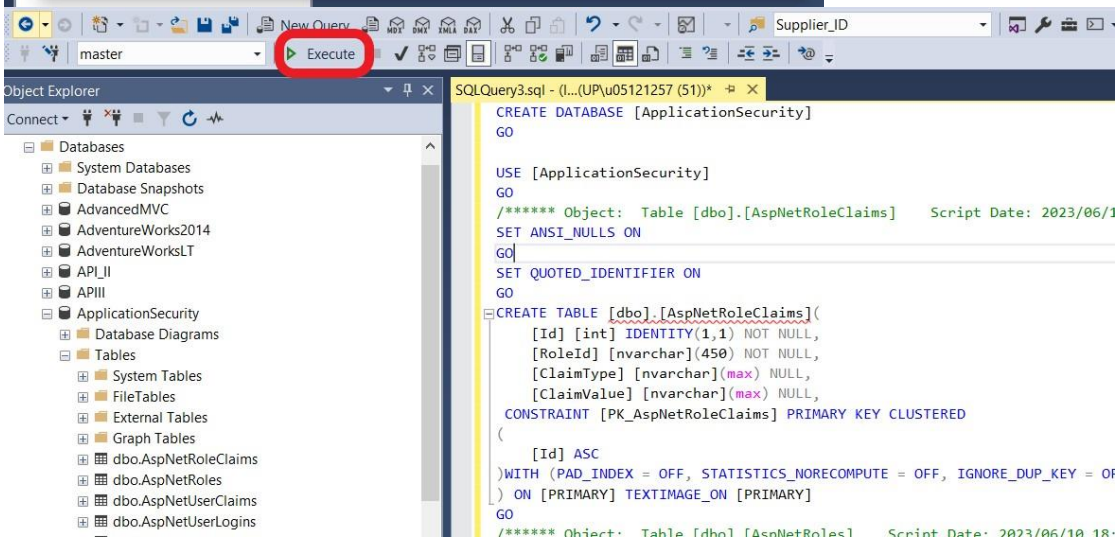
The University of Pretoria commits itself to producing academic work of integrity. I affirm that I am aware of and have read the Rules and Policies of the University, more specifically the Disciplinary Procedure and the Tests and Examinations Rules, which prohibit any unethical, dishonest or improper conduct during tests, assignments, examinations and/or any other forms of assessment. I am aware that no student or any other person may assist or attempt to assist another student, or obtain help, or attempt to obtain help from another student or any other person during tests, assessments, assignments, examinations and/or any other forms of assessment.

SECTION A – APPLICATION SECURITY (30)

For Section A, you need to complete **one question**; an **Application Security question**. *The question requires you to run a SQL script file (**Exam_SQL_Script.sql**) in the **lab's SQL Server Management Studio** (**SSMS 18**) before proceeding with this. **NB: you do not need to do the migration from the application. In other words, doing the "add-migration" and "update-database" is unnecessary for this question**.*

Before proceeding make sure you logged in to the lab PC using "**.\user**" as your **Username** and your **Password must be left empty** (*I.e., do not enter a password, leave it blank*). To create the database and test your completed program **for the question in Section A** you need to have MS SQL Server running. The **Server name** (login) for ***SSMS 18** is "**.\SQLEXPRESS**"*. Thereafter, execute the ***Exam_SQL_Script.sql*** in **SSMS 18**.





Thereafter, you proceed to complete the question for Section A.

All the source files are in the following zipped file:

- **Exam_SectionA_Question.zip**

Once you have completed the question in this section, upload the ***Program.cs*** file and the ***AuthenticationController.cs*** file to the **Section A upload slot**.

Your task is to complete the codebase to get the application to function as described below. For the question, please do the following:

- Read the instructions carefully.

- Add the necessary code to the specified file(s) in the required application.
- Only upload the modified file(s) associated with the question to ClickUP.

(**IMPORTANT: DO NOT UPLOAD ZIP, TAR, RAR or SLN files)**

SECTION A - QUESTION (APPLICATION SECURITY) (30 MARKS)

Only upload the ***Program.cs*** and the ***AuthenticationController.cs*** files after completing this question. **Two files in total to upload**.

Company X is a startup company that is trying to build their first iteration of .Net Identity Security within an API. As the new Software Engineer you need to help them achieve this.

- **1.1 In the "*Program.cs*" file you need to do the following [5 Marks]**
  a) Add configurations to make sure the application does not allow for duplicate email accounts. Further all passwords should be 12 characters or more, have a digit in it, have an uppercase character and contain a non-alphanumeric character.

- **1.2 In the "*AuthenticationController.cs*" file you need to do the following [25 Marks]**
  a) Create a "***RegisterUser***" endpoint/function to allow a User to register on the application. *Note: The endpoint/function* ***route*** *name should be the same as the endpoint/function name.* Furthermore, the endpoint/function is an asynchronous endpoint/method taking a view model as a parameter and returning an "*IActionResult*". ***Note, all the code for 1.2 must be completed within this endpoint/function (no other location)*** (5 marks).
  b) The username should be compared to the existing account names to check if it already exists (1 mark)
  c) If the account name does not exist, the account creation code should be done inside a try/catch block. If any **Exceptions** happen it should be *caught* in the *catch block* and a **500-status code** with the specific details of the Exception should be returned*. (3 marks)*
  d) If the account already exists, a **409-status code** with the name of the existing account should be returned. (1 mark)
  e) The account should be created in the database passing through the **User name**, **Email**, **Phone number**, and **User id**. (6 marks)
  f) Phone number validation should only allow for ten digits and the first digit must be a zero. If the phone number is not valid, a **400-status code** with the message "*Please enter a valid 10-digit phone number*" must be returned. (3 marks)
  g) All ***Identity*** errors that are returned during the account creation attempt, should be logged with a **400-status code** with the details of any returned errors. (5 marks)
  h) If the account is created successfully a **200-status code** with the message "*Your account 'add your user name' was created successfully. You may proceed with logging in*" (1 mark)

- Once you are done and have successfully implemented the required code, you can run and test the application and see the **Section A Expected Output** examples displayed below (*Figure 1: Phone number Bad request, Figure 2: Successful account creation*).

- **NB: The application was created using Visual Studio 2022. You have already been given a document for any environment pre-setup installations required and how to do it, shared on the Course Module ClickUP site. Therefore, for this question, no additional environment pre-setup installations are needed.**

Section A Expected Output

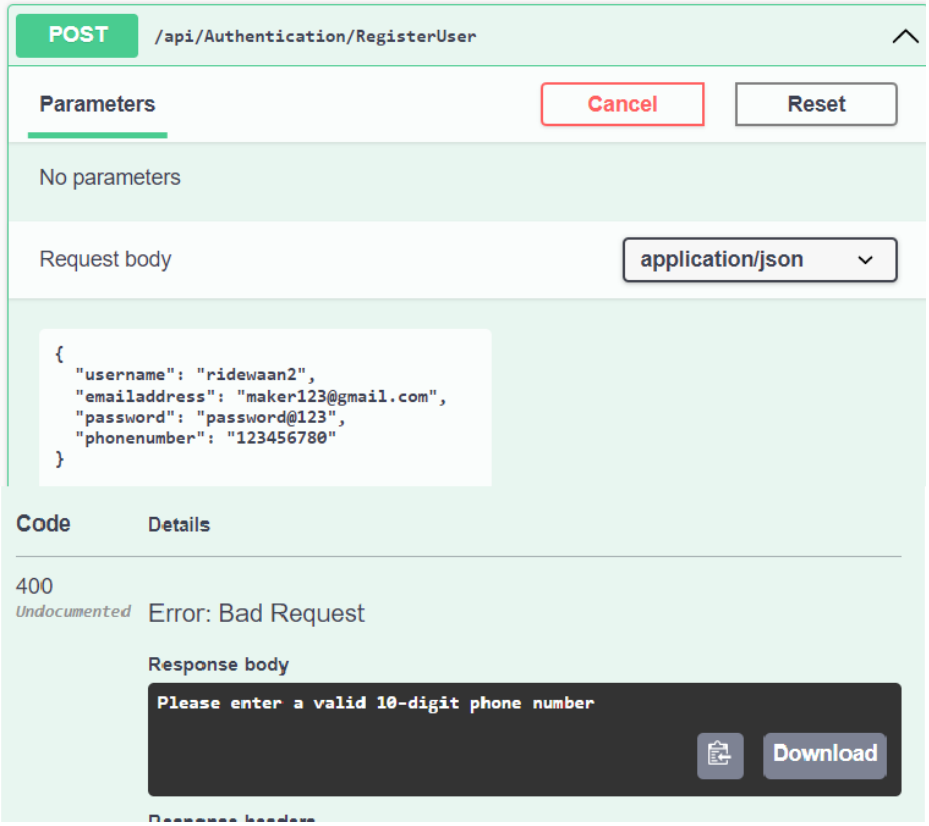


Figure 1: Phone number Bad request

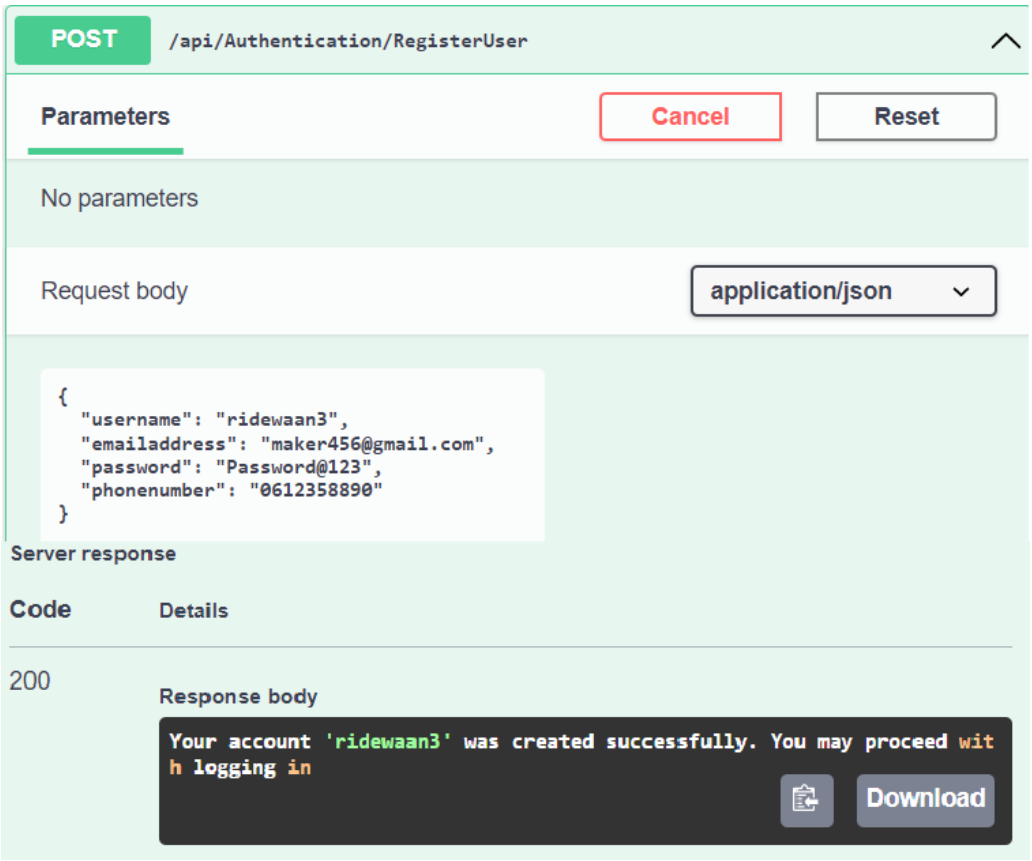


Figure 2: Successful account creation

SECTION A TOTAL **30**

SECTION B – REPORTING (25)

For Section B, you need to complete **one question**; an Angular Reporting **question**. *The question does not require access to an existing database file to be completed.*

All the source files are in the following zipped files:

- **Exam_SectionB_Question.zip**

Once you have completed the question in this section, **upload 7 files**, the **home.component.ts**, **home.component.html**, **pie-chart.component.ts**, **pie-chart.component.html**, **bar-line.component.ts**, **bar-line.component.html**, and **app.component.html** files to the **Section B upload slot**.

Your task is to complete the codebase to get the application to function as described below. For the question, please do the following:

- Read the instructions carefully.
- Add the necessary code to the specified file(s) in the required application. If the file does not exist you need to create it.
- Only upload the files associated with the question to ClickUP.

(**IMPORTANT: DO NOT UPLOAD ZIP, TAR, RAR or SLN files)**

SECTION B - QUESTION (REPORTING) (25 MARKS)

Only upload the **home.component.ts**, **home.component.html**, **pie-chart.component.ts**, **pie-chart.component.html**, **bar-line.component.ts**, **bar-line.component.html**, and **app.component.html** files after completing the question. **Seven files in total to upload**.

Our mission at TimTop Clothing is to provide our customers with a unique and stylish selection of clothing and accessories that allow them to embrace their personal style. TimTop Clothing was founded by a passionate fashion enthusiast out of a deep love for fashion and a desire to create a shopping experience that goes above and beyond the ordinary. We understand that fashion is about expressing oneself, feeling confident, and finding pieces that make you look and feel your best.

You are given an unfinished project and must complete the code in the files **home.component.ts**, **home.component.html**, **pie-chart.component.ts**, **pie-chart.component.html**, **bar-line.component.ts**, **bar-line.component.html**, and **app.component.html** under the provided function declarations. In quarter 1, you must present three charts (*bar chart*, *pie chart*, and *bar line chart*) to the CEO that show sales data for men's clothing items between 2022 and 2023. ***You will then be required to create the chart data using the data from Table 1***.

Table 1: TimTop Clothing Sales Data for Quarter 1 for 2022 and 2023

	2022	2023
Shirts	**446**	**623**
Jacket	**551**	**431**
Men Tops	**462**	**525**
Men Pants	**158**	**306**
Swimwear	**171**	**100**
Shoes	**553**	**369**
Sleepwear	**566**	**417**
Men Accessories	**231**	**420**

1.1 In the "app.component.html" file you need to do the following [4 Marks]

a. A functional navigation bar with the *home page*, *pie chart page*, and *bar line page*.

1.2 In the "home.component.ts" file you need to do the following [6 Marks]

a. Create a **Chart function** for bar chart which must run as soon as the page loads the home page. (1 Mark)
b. Create a **function** for the bar chart. This should include the *data* and *labels* for the bar chart. (4 Marks)
c. Labels for 2022 should have a blue background color while labels for 2023 should have a red background color. (1 Mark)

1.3 In the "home.component.html" file you need to do the following [2 Marks]

a. The bar chart should have a header titled "*Product Sales*". (1 Mark)
b. Create a container and use angular string interpolation to render the bar chart variable. (1 Mark)

1.4 In the "pie-chart.component.ts" file you need to do the following [4 Marks]

a. Create a method for the pie chart. This should include the data and labels for the pie chart.

1.5 In the "pie-chart.component.html" file you need to do the following [2 Marks]

a. The pie chart should have a header titled "*Product Sales*". (1 Mark)
b. Create a container and use angular string interpolation to render the pie chart variable. (1 Mark)

1.6 In the "bar-line.component.ts" file you need to do the following [5 Marks]

a. Create a method for the bar line chart. This should include the data and labels for the bar line chart. Note that 2022 data should be represented using the line graph while the 2023 data should be represented using the bar graph.

1.7 In the "bar-line.component.html" file you need to do the following [2 Marks]

a. The bar line chart should have a header titled "*Product Sales*". (1 Mark)
b. Create a container and use angular string interpolation to render the bar line chart variable. (1 Mark)

- Once you are done and have successfully implemented the required code, you can run and test the application and see the **Section B Expected Output** examples displayed below (*Figure 1: Bar chart, Figure 2: Pie Chart, and Figure 3: Bar line chart*).

- **NB: <u>Do not run "npm install"</u>. The application was created using Visual Studio Code and Angular. You have already been given a document for any environment pre-setup installations required and how to do it, shared on the Course Module ClickUP site. Therefore, for this question, no additional environment pre-setup installations are needed.**

Section B Expected Output

Figure 1: Bar Chart

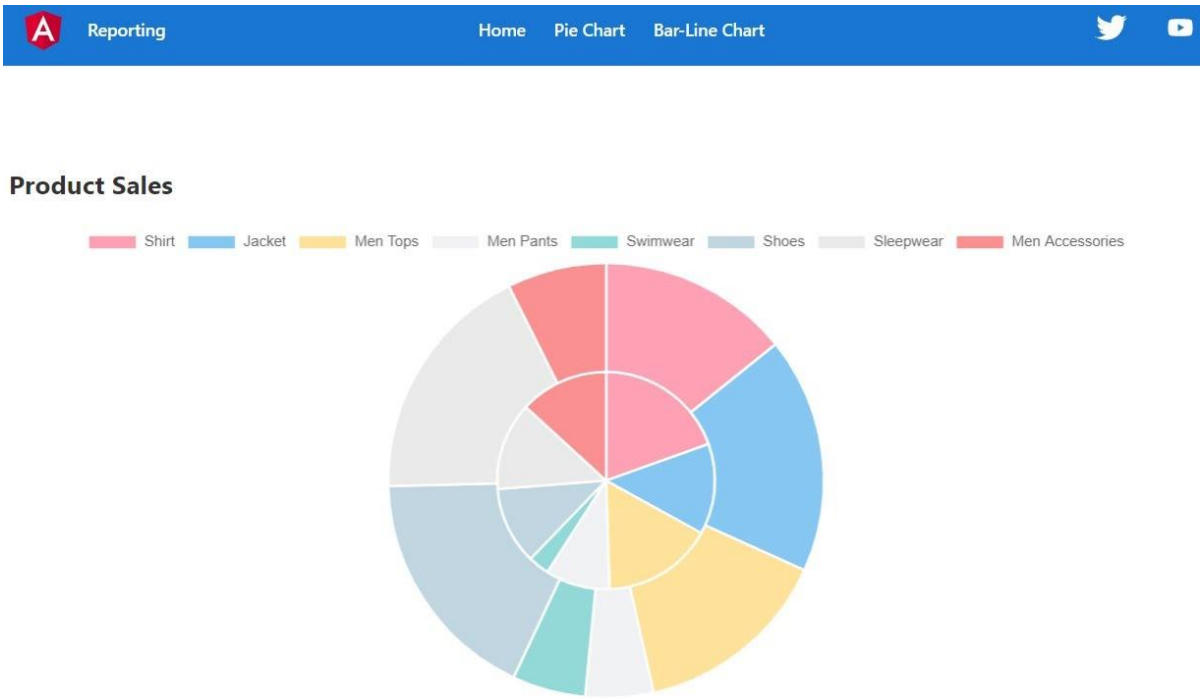


Figure 2: Pie Chart

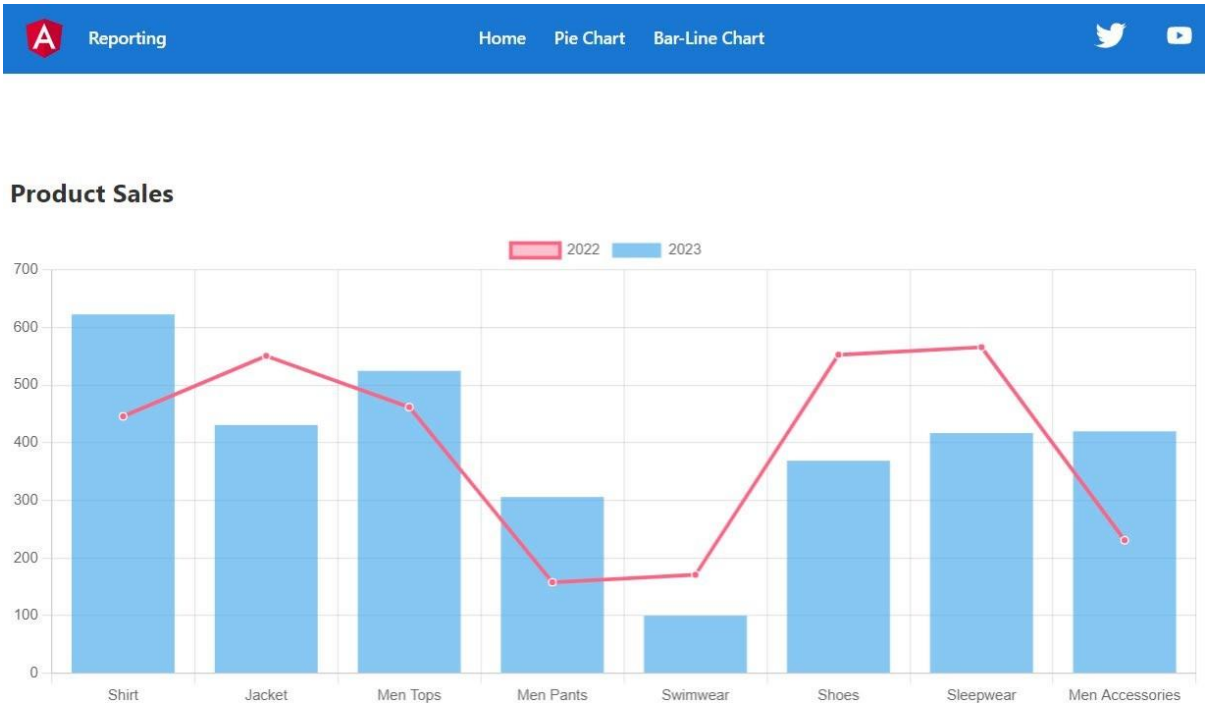


Figure 3: Bar-Line Chart

SECTION B TOTAL **25**

SECTION C – IONIC (25)

For Section C, you need to complete **one question**; an Ionic **question**. *The question does not require access to an existing database file to be completed.*

All the source files are in the following zipped files:

- **Exam_SectionC_Question.zip**

Once you have completed the question in this section, **upload 4 files**, the ***tabs.page.html, tabs-routing.module.ts, home.page.html,*** *and* ***cart.page.html*** files to the **Section C upload slot**.

Your task is to complete the codebase to get the application to function as described below. For the question, please do the following:

- Read the instructions carefully.
- Add the necessary code to the specified file(s) in the required application. If the file does not exist you need to create it.
- Only upload the files associated with the question to ClickUP.

(**IMPORTANT: DO NOT UPLOAD ZIP, TAR, RAR or SLN files)**

SECTION C - QUESTION (IONIC) (25 MARKS)

Only upload the ***tabs.page.html, tabs-routing.module.ts, home.page.html,*** *and* ***cart.page.html*** files after completing the question. **Four files in total to upload**.

Due to your adept knowledge of programming in Ionic, a new company that provides the top 5 streaming services in South Africa has approached you to create a mobile website. Your job is to create the **home page** and **cart page** using the provided data in the given code, and *you must also ensure that the routing from each page works*.

1.1 In the "tabs.page.html" and "tabs-routing.module.ts" files you need to do the following [5 Marks]

a. A functional tab bar with the ***home page***, and ***cart page*** at the bottom of the screen. (3 marks)
b. The tabs must be visible on both pages. (1 mark)
c. The cart tab must show the total number of items added to the cart. (1 mark)

1.2 In the "home.page.html" file you need to do the following [7 Marks]

a. The home page should have a title – "*Streaming Providers*" using the ion-header component. (1 mark)
b. The five streaming service *names* and *descriptions* should be called using interpolation coupled with the card component. (4 marks)
c. Make use of the add to cart buttons to add a streaming service to the cart. (2 marks)

1.3 In the "cart.page.html" file you need to do the following [13 Marks]

a. The cart page should have a title – "*Items in Cart*" using the ion-header component. (2 marks)
b. Once an item is added to the cart, the content of the cart page should have the following headers in a table format: **Subscription**, **Price**, **Quantity**, and **Total Cost**. *Each of these headers must be automatically populated*. (4 marks)
c. The quantity must be able to increase or decrease the number of items in the cart. (4 marks)
c. A checkout button must be used incorporating it with modals displaying a "*payment successful message"*. (3 marks)

- Once you are done and have successfully implemented the required code, you can run and test the application and see the **Section C Expected Output** examples displayed below (*Figure 1: Home page, Figure 2: Cart page*).

- **NB: <u>Do not run "npm install"</u>. The application was created using Visual Studio Code and Ionic. You have already been given a document for any environment pre-setup installations required and how to do it, shared on the Course Module ClickUP site. Therefore, for this question, no additional environment pre-setup installations are needed.**

Section C Expected Output

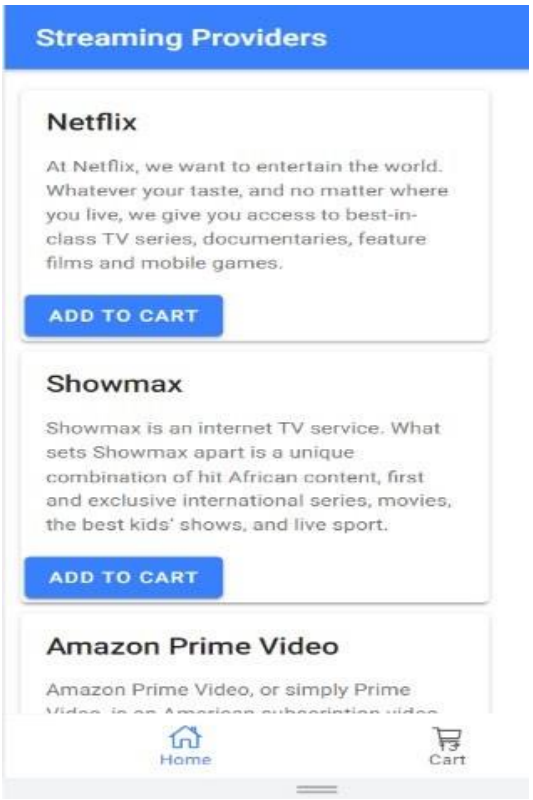


Figure 1: Home page

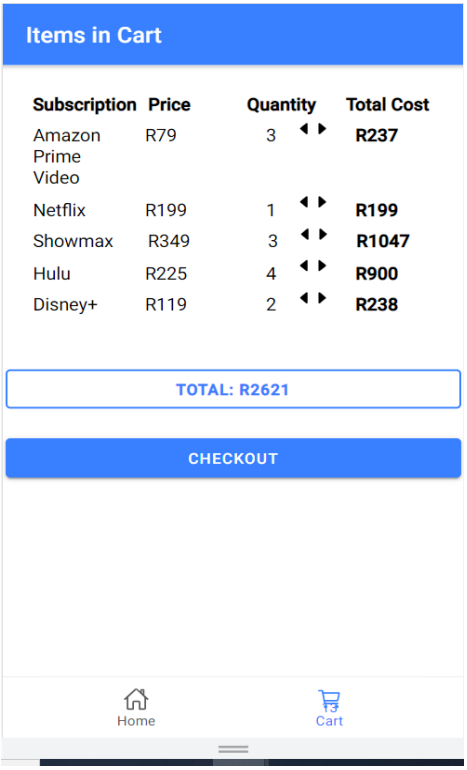


Figure 2: Cart page

SECTION C TOTAL **25**

SECTION D – ADVANCED CONCEPTS (15)

For Section D, you need to complete **one question**; an Angular chatbot **question**. *The question does not require access to an existing database file to be completed.*

All the source files are in the following zipped files:

- **Exam_SectionD_Question.zip**

Once you have completed the question in this section, **upload 2 files**, the **chat-support.component.ts**, and **chat-support.component.html** files to the **Section D upload slot**.

Your task is to complete the codebase to get the application to function as described below. For the question, please do the following:

- Read the instructions carefully.
- Add the necessary code to the specified file(s) in the required application.
- Only upload the files associated with the question to ClickUP.

(**IMPORTANT: DO NOT UPLOAD ZIP, TAR, RAR or SLN files)**

SECTION D - QUESTION (ADVANCED CONCEPTS) (15 MARKS)

Only upload the **chat-support.component.ts** and the **chat-support.component.html** files after completing this question. **Two files in total to upload**.

Your employer is an insurance company that is looking at implementing chatbots to lower the demand on the call-centre support staff. They require you to update their current chatbot front-end to, for now, be able to track the conversations between the end-users and the chatbot angular client (app). They want you to create a logging prototype on the html page using basic angular code.

- **1.1 In the "chat-support.component.html" file you need to do the following [10 Marks]**
  a) Create a table with adequate styling using bootstrap (*note, bootstrap is already included in the app*) to record the conversations that you are having with the chatbot in the chatbot window. The table should have headers for the incremental conversation number (#), the user **type** (e.g., end-*user* or the chatbot *client*), the **message** text, and a date **timestamp** *(format: dd/MM/yyyy h:mm:ss)* of when each message was captured. (4 marks).
  b) If the end-user has not begun a conversation with the chatbot, the table with the logs should be hidden. (1 mark)
  c) Each chat text should be logged populating the #, **type**, **message**, and the **timestamp** columns. (5 marks)

- **1.2 In the "chat-support.component.ts" file you need to do the following [5 Marks]**
  a) Implement the code to allow the date **timestamp** *(format: dd/MM/yyyy h:mm:ss)* functionality to be passed with the other message data as part of the conversation logs (*see 1.1*). (5 marks)

- Once you are done and have successfully implemented the required code, you can run and test the application and see the **Section D Expected Output** examples displayed below (*Figure 1: Before conversation, Figure 2: During conversation*).

- **NB: <u>Do not run "npm install"</u>. The application was created using Visual Studio Code and Angular. You have already been given a document for any environment pre-setup installations required and how to do it, shared on the Course Module ClickUP site. Therefore, for this question, no additional environment pre-setup installations are needed.**

Section D Expected Output

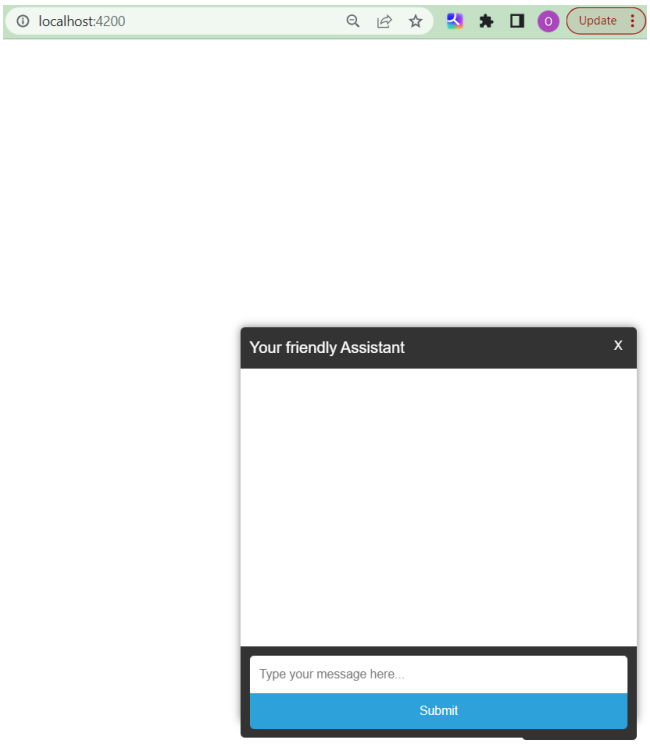


Figure 1: Before conversation

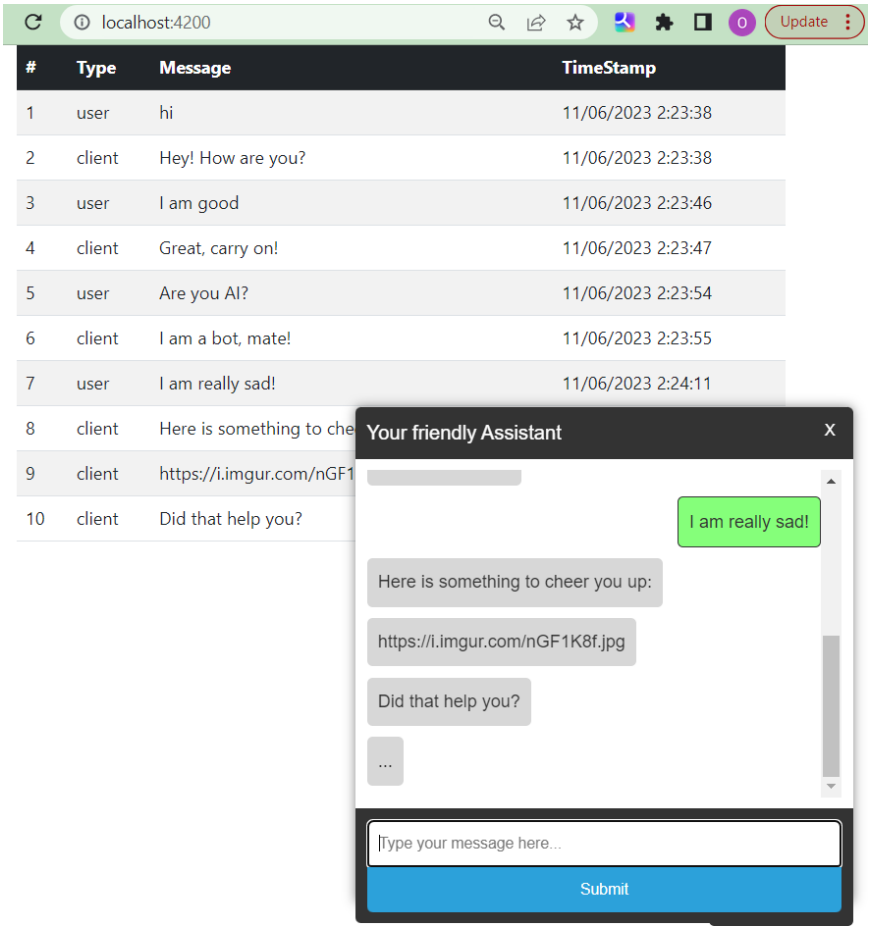


Figure 2: During conversation

SECTION D TOTAL **15**

SECTION A - QUESTION (APPLICATION SECURITY)

Program.cs

```
using ApplicationSecurity.Factory;
using ApplicationSecurity.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddCors(options => options.AddDefaultPolicy(
    include =>
    {
        include.AllowAnyHeader();
        include.AllowAnyMethod();
        include.AllowAnyOrigin();
    }));

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddIdentity<AppUser, IdentityRole>(options =>
{
    // Enforce password requirements
    options.Password.RequireDigit = true;
    options.Password.RequireUppercase = true;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequiredLength = 12;

    // Prevent duplicate emails
    options.User.RequireUniqueEmail = true;
})
.AddEntityFrameworkStores<AppDbContext>()
.AddDefaultTokenProviders();

builder.Services.AddAuthentication();

builder.Services.AddScoped<IUserClaimsPrincipalFactory<AppUser>,
AppUserClaimsPrincipalFactory>();

builder.Services.AddDbContext<AppDbContext>(options =>
            options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnectio
n")));

builder.Services.AddScoped<IRepository, Repository>();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
```

```
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCors();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

AuthenticationController.cs

```
using ApplicationSecurity.Models;
using ApplicationSecurity.ViewModels;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using System.Linq;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace ApplicationSecurity.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthenticationController : ControllerBase
    {
        private readonly UserManager<AppUser> _userManager;
        private readonly IRepository _repository;
        private readonly IUserClaimsPrincipalFactory<AppUser> _claimsPrincipalFactory;
        private readonly IConfiguration _configuration;
        private readonly ILogger<AuthenticationController> _logger;

        public AuthenticationController(UserManager<AppUser> userManager,
                                        IUserClaimsPrincipalFactory<AppUser>
claimsPrincipalFactory,
                                        IConfiguration configuration,
                                        IRepository repository,
                                        ILogger<AuthenticationController> logger)
        {
            _userManager = userManager;
            _claimsPrincipalFactory = claimsPrincipalFactory;
            _configuration = configuration;
            _repository = repository;
            _logger = logger;
        }

        [HttpPost("RegisterUser")]
        public async Task<IActionResult> RegisterUser([FromBody] RegisterViewModel model)
        {
            // Check if username already exists
            var existingUser = await _userManager.FindByNameAsync(model.Username);
            if (existingUser != null)
            {
```

```
                return Conflict($"Username '{model.Username}' already exists.");
            }

            // Validate phone number
            if (!Regex.IsMatch(model.PhoneNumber, @"^0\d{9}$"))
            {
                return BadRequest("Please enter a valid 10-digit phone number.");
            }

            var user = new AppUser
            {
                UserName = model.Username,
                Email = model.Email,
                PhoneNumber = model.PhoneNumber
            };

            try
            {
                var result = await _userManager.CreateAsync(user, model.Password);
                if (result.Succeeded)
                {
                    return Ok($"Your account '{model.Username}' was created successfully. You
may proceed with logging in.");
                }

                // Handle identity errors
                var errors = result.Errors.Select(e => e.Description).ToList();
                return BadRequest(errors);
            }
            catch (System.Exception ex)
            {
                _logger.LogError(ex, "An error occurred while creating the user.");
                return StatusCode(500, $"Internal server error: {ex.Message}");
            }
        }
    }

    public class RegisterViewModel
    {
        public string Username { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
        public string PhoneNumber { get; set; }
    }

    public class AppUser : IdentityUser
    {
        // Additional properties can go here if needed
    }
}
```

SECTION B – REPORTING

app.component.html

```
<nav>
  <ul>
    <li><a routerLink="/">Home</a></li>
    <li><a routerLink="/pie-chart">Pie Chart</a></li>
    <li><a routerLink="/bar-line">Bar Line Chart</a></li>
  </ul>
</nav>
<router-outlet></router-outlet>
```

home.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Chart } from 'chart.js/auto';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  public barChart: any;

  ngOnInit(): void {
    this.createBarChart();
  }

  createBarChart() {
    this.barChart = new Chart("barChart", {
      type: 'bar',
      data: {
        labels: ['Shirts', 'Jacket', 'Men Tops', 'Men Pants', 'Swimwear', 'Shoes',
'Sleepwear', 'Men Accessories'],
        datasets: [
          {
            label: '2022',
            data: [446, 551, 462, 158, 171, 553, 566, 231],
            backgroundColor: 'blue'
          },
          {
            label: '2023',
            data: [623, 431, 525, 306, 100, 369, 417, 420],
            backgroundColor: 'red'
          }
        ]
      },
      options: {
        responsive: true,
        scales: {
          y: {
            beginAtZero: true
          }
        }
```

```
      }
    });
  }
}
```

home.component.html

```
<h2>Product Sales</h2>
<div>
  <canvas id="barChart"></canvas>
</div>
```

pie-chart.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Chart } from 'chart.js/auto';

@Component({
  selector: 'app-pie-chart',
  templateUrl: './pie-chart.component.html',
  styleUrls: ['./pie-chart.component.css']
})
export class PieChartComponent implements OnInit {
  public pieChart: any;

  ngOnInit(): void {
    this.createPieChart();
  }

  createPieChart() {
    this.pieChart = new Chart("pieChart", {
      type: 'pie',
      data: {
        labels: ['Shirts', 'Jacket', 'Men Tops', 'Men Pants', 'Swimwear', 'Shoes',
'Sleepwear', 'Men Accessories'],
        datasets: [
          {
            label: 'Product Sales',
            data: [446, 551, 462, 158, 171, 553, 566, 231],
            backgroundColor: ['red', 'blue', 'green', 'yellow', 'purple', 'orange', 'cyan',
'pink']
          }
        ]
      },
      options: {
        responsive: true
      }
    });
  }
}
```

pie-chart.component.html

```
<h2>Product Sales</h2>
<div>
  <canvas id="pieChart"></canvas>
</div>
```

bar-line.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Chart } from 'chart.js/auto';

@Component({
  selector: 'app-bar-line',
  templateUrl: './bar-line.component.html',
  styleUrls: ['./bar-line.component.css']
})
export class BarLineComponent implements OnInit {
  public barLineChart: any;

  ngOnInit(): void {
    this.createBarLineChart();
  }

  createBarLineChart() {
    this.barLineChart = new Chart("barLineChart", {
      type: 'bar',
      data: {
        labels: ['Shirts', 'Jacket', 'Men Tops', 'Men Pants', 'Swimwear', 'Shoes',
'Sleepwear', 'Men Accessories'],
        datasets: [
          {
            type: 'line',
            label: '2022',
            data: [446, 551, 462, 158, 171, 553, 566, 231],
            borderColor: 'blue',
            fill: false
          },
          {
            type: 'bar',
            label: '2023',
            data: [623, 431, 525, 306, 100, 369, 417, 420],
            backgroundColor: 'red'
          }
        ]
      },
      options: {
        responsive: true,
        scales: {
          y: {
            beginAtZero: true
          }
        }
      }
    });
```

```
  }
}
```

bar-line.component.html

```
<h2>Product Sales</h2>
<div>
  <canvas id="barLineChart"></canvas>
</div>
```

SECTION C – IONIC

tabs.page.html

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="home">
      <ion-icon name="home"></ion-icon>
      <ion-label>Home</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="cart">
      <ion-icon name="cart"></ion-icon>
      <ion-label>Cart</ion-label>
      <ion-badge *ngIf="cartItemCount > 0">{{ cartItemCount }}</ion-badge>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

tabs-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { TabsPage } from './tabs.page';

const routes: Routes = [
  {
    path: '',
    component: TabsPage,
    children: [
      {
        path: 'home',
        loadChildren: () => import('../home/home.module').then(m => m.HomePageModule)
      },
      {
        path: 'cart',
        loadChildren: () => import('../cart/cart.module').then(m => m.CartPageModule)
      },
      {
        path: '',
        redirectTo: '/tabs/home',
        pathMatch: 'full'
      }
    ]
  },
```

```
  {
    path: '',
    redirectTo: '/tabs/home',
    pathMatch: 'full'
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class TabsPageRoutingModule {}
```

home.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>Streaming Providers</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-card *ngFor="let service of streamingServices">
    <ion-card-header>
      <ion-card-title>{{ service.name }}</ion-card-title>
    </ion-card-header>
    <ion-card-content>
      {{ service.description }}
      <ion-button (click)="addToCart(service)">Add to Cart</ion-button>
    </ion-card-content>
  </ion-card>
</ion-content>
```

cart.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>Items in Cart</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-grid>
    <ion-row>
      <ion-col>Subscription</ion-col>
      <ion-col>Price</ion-col>
      <ion-col>Quantity</ion-col>
      <ion-col>Total Cost</ion-col>
    </ion-row>
    <ion-row *ngFor="let item of cartItems">
      <ion-col>{{ item.name }}</ion-col>
      <ion-col>{{ item.price | currency }}</ion-col>
      <ion-col>
        <ion-button (click)="decreaseQuantity(item)">-</ion-button>
        {{ item.quantity }}
        <ion-button (click)="increaseQuantity(item)">+</ion-button>
```

```
      </ion-col>
      <ion-col>{{ (item.price * item.quantity) | currency }}</ion-col>
    </ion-row>
  </ion-grid>
  <ion-button (click)="checkout()">Checkout</ion-button>
</ion-content>

<ion-modal #paymentModal>
  <ion-header>
    <ion-toolbar>
      <ion-title>Payment</ion-title>
    </ion-toolbar>
  </ion-header>
  <ion-content>
    Payment successful!
    <ion-button (click)="dismissModal()">Close</ion-button>
  </ion-content>
</ion-modal>
```

Supporting TypeScript for Home and Cart Pages

home.page.ts

```
import { Component, OnInit } from '@angular/core';
import { NavController } from '@ionic/angular';
import { CartService } from '../services/cart.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
  styleUrls: ['./home.page.scss'],
})
export class HomePage implements OnInit {
  streamingServices = [
    { name: 'Netflix', description: 'Watch TV shows and movies', price: 150 },
    { name: 'Amazon Prime', description: 'Movies and TV shows', price: 100 },
    { name: 'Disney+', description: 'Exclusive Disney content', price: 120 },
    { name: 'Hulu', description: 'Stream live TV', price: 130 },
    { name: 'Showmax', description: 'Local and international content', price: 140 }
  ];

  constructor(private cartService: CartService, private navCtrl: NavController) {}

  ngOnInit() {}

  addToCart(service) {
    this.cartService.addToCart(service);
  }
}
```

cart.page.ts

```
import { Component, OnInit } from '@angular/core';
import { CartService } from '../services/cart.service';

@Component({
  selector: 'app-cart',
  templateUrl: './cart.page.html',
  styleUrls: ['./cart.page.scss'],
})
export class CartPage implements OnInit {
  cartItems = [];

  constructor(private cartService: CartService) {}

  ngOnInit() {
    this.cartItems = this.cartService.getCart();
  }

  increaseQuantity(item) {
    this.cartService.increaseQuantity(item);
  }

  decreaseQuantity(item) {
    this.cartService.decreaseQuantity(item);
  }

  checkout() {
    // Implement checkout functionality here
    this.cartService.clearCart();
    // Trigger modal for payment success
  }

  dismissModal() {
    // Implement modal dismissal
  }
}
```

cart.service.ts

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

@Injectable({
  providedIn: 'root'
})
export class CartService {
  private cart = [];

  constructor() {}

  addToCart(item) {
    const existingItem = this.cart.find(cartItem => cartItem.name === item.name);
    if (existingItem) {
      existingItem.quantity++;
    } else {
      this.cart.push({ ...item, quantity: 1 });
    }
  }

  getCart() {
    return this.cart;
  }

  increaseQuantity(item) {
    const cartItem = this.cart.find(cartItem => cartItem.name === item.name);
    if (cartItem) {
      cartItem.quantity++;
    }
  }

  decreaseQuantity(item) {
    const cartItem = this.cart.find(cartItem => cartItem.name === item.name);
    if (cartItem && cartItem.quantity > 1) {
      cartItem.quantity--;
    }
  }

  clearCart() {
    this.cart = [];
  }
}
```

SECTION D – ADVANCED CONCEPTS

chat-support.component.html

```
<div class="container mt-4">
  <h2>Chat Support</h2>
  <div class="chat-window">
    <div class="messages">
      <div *ngFor="let message of messages">
        <strong>{{ message.type }}:</strong> {{ message.text }}
      </div>
    </div>
    <div class="input-group mt-3">
      <input [(ngModel)]="userMessage" (keyup.enter)="sendMessage()" type="text" class="form-
control" placeholder="Type your message...">
      <div class="input-group-append">
        <button class="btn btn-primary" (click)="sendMessage()">Send</button>
      </div>
    </div>
  </div>

  <table *ngIf="messages.length > 0" class="table table-striped mt-4">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">Type</th>
        <th scope="col">Message</th>
        <th scope="col">Timestamp</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let log of conversationLogs; let i = index">
        <th scope="row">{{ i + 1 }}</th>
        <td>{{ log.type }}</td>
        <td>{{ log.text }}</td>
        <td>{{ log.timestamp }}</td>
      </tr>
    </tbody>
  </table>
</div>
```

chat-support.component.ts

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

@Component({
  selector: 'app-chat-support',
  templateUrl: './chat-support.component.html',
  styleUrls: ['./chat-support.component.css']
})
export class ChatSupportComponent {
  userMessage: string = '';
  messages: { type: string, text: string }[] = [];
  conversationLogs: { type: string, text: string, timestamp: string }[] = [];
```

```
  sendMessage() {
    if (this.userMessage.trim() === '') {
      return;
    }

    const timestamp = new Date().toLocaleString('en-GB', {
      day: '2-digit',
      month: '2-digit',
      year: 'numeric',
      hour: '2-digit',
      minute: '2-digit',
      second: '2-digit'
    });

    const userMessageObj = {
      type: 'End-user',
      text: this.userMessage,
      timestamp: timestamp
    };

    this.messages.push(userMessageObj);
    this.conversationLogs.push(userMessageObj);

    // Simulate chatbot response
    setTimeout(() => {
      const botMessageObj = {
        type: 'Chatbot',
        text: 'This is a response from the chatbot.',
        timestamp: new Date().toLocaleString('en-GB', {
          day: '2-digit',
          month: '2-digit',
          year: 'numeric',
          hour: '2-digit',
          minute: '2-digit',
          second: '2-digit'
        })
      };
      this.messages.push(botMessageObj);
      this.conversationLogs.push(botMessageObj);
    }, 1000);

    this.userMessage = '';
  }
}
```