DEPARTMENT OF INFORMATICS

INF 354

EXAMINATION **DATE: 2023-06-23**

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

Student Number								Surname	Initials
-	-	-	-	-	-	-	-	MEMO	--

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

Instructions

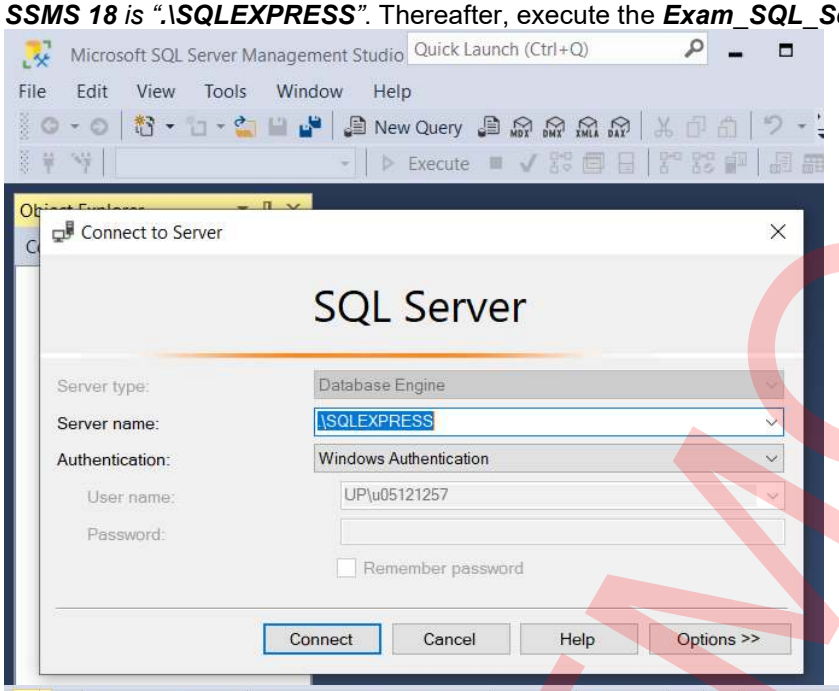
1. This paper consists of 4 sections with one question per section (sub-sets of instructions) each.
2. Each section relates to small semi-complete programs that need to be updated or finalised.
3. Each question relates to file(s) in the semi-complete programs that you need to update or finalise.
4. Each 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.

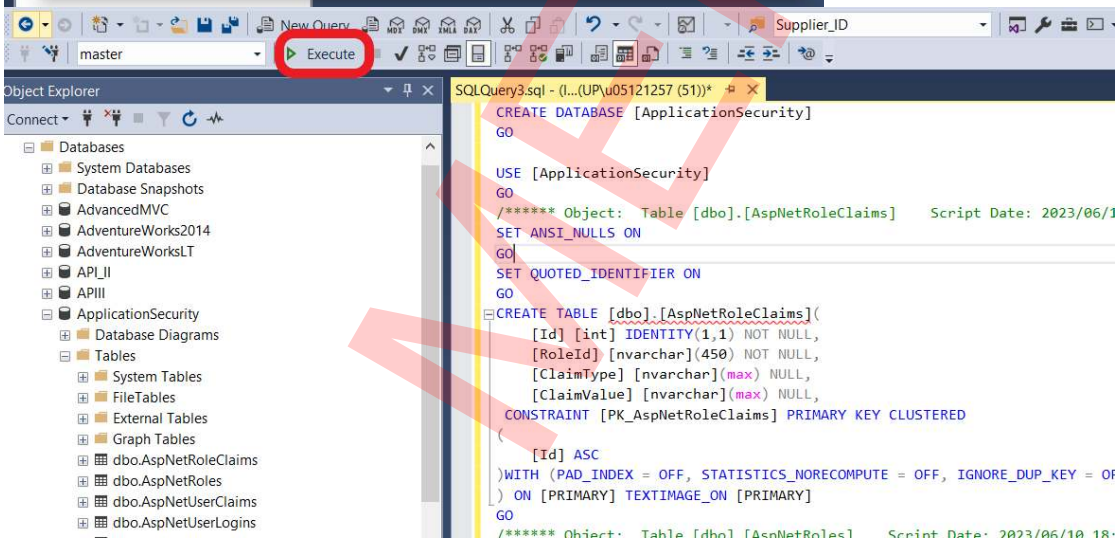
The University of Pretoria commits itself to producing academic work of integrity. I affirm that I am aware of and have read the Rules and Policies of the University, more specifically the Disciplinary Procedure and the Tests and Examinations Rules, which prohibit any unethical, dishonest or improper 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:

- **Examination_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

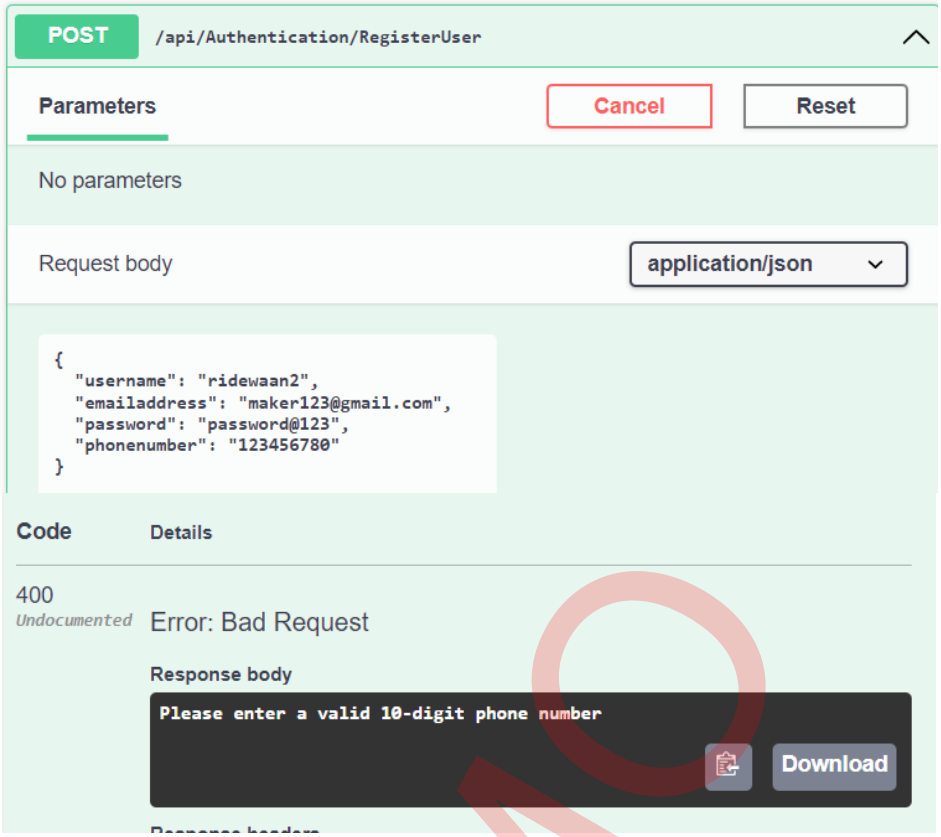


Figure 1: Phone number Bad request

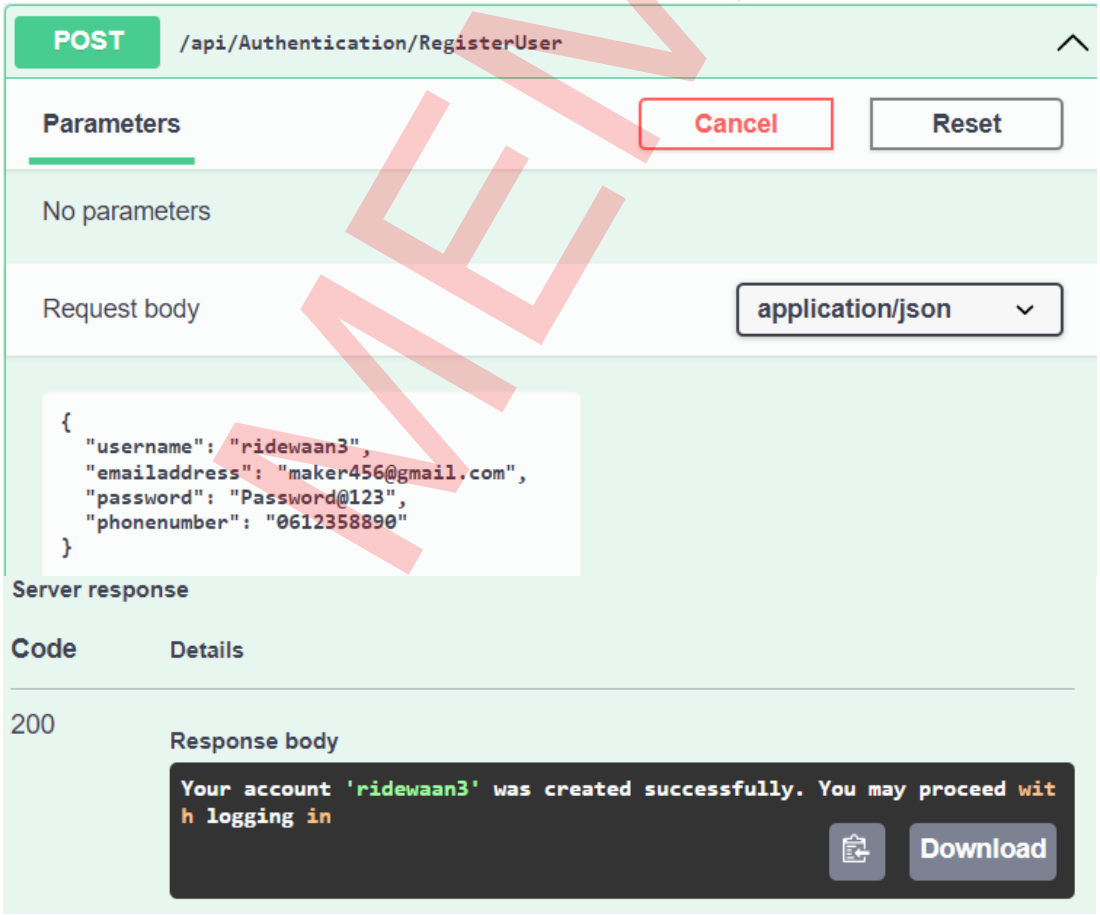


Figure 2: Successful account creation

SOLUTION SECTION A

File: Program.cs

```
// 5 marks in total for the password and user configs
builder.Services.AddIdentity<AppUser, IdentityRole>(options =>
{
    // 1 mark for each option (5 marks)
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequiredLength = 12;
    options.Password.RequireDigit = true;
    options.Password.RequireUppercase = true;
    options.User.RequireUniqueEmail = true;
})
.AddEntityFrameworkStores<AppDbContext>()
.AddDefaultTokenProviders();
```

File: AuthenticationController.cs

```
// 25 marks in total for the endpoint
[HttpPost]    // 1 mark
[Route("RegisterUser")]  // 1 mark
0 references
public async Task<IActionResult> RegisterUser(UserViewModel uvm)  // 3 marks
{
    var user = await _userManager.FindByNameAsync(uvm.username);  // 1 mark

    if (user == null)  // 1 mark
    {
        try // 2 marks
        {
            string phoneNumberPattern = @"^0\d{9}$";  // 1 mark

            bool isValidPhoneNumber = Regex.IsMatch(uvm.phonenumber, phoneNumberPattern);  // 1 mark

            if (!isValidPhoneNumber) return BadRequest("Please enter a valid 10-digit phone number"); // 1 mark

            user = new AppUser  // 5 mark
            {
                Id = Guid.NewGuid().ToString(),
                UserName = uvm.username,
                Email = uvm.emailaddress,
                PhoneNumber = uvm.phonenumber
            };

            var result = await _userManager.CreateAsync(user, uvm.password);  // 1 mark

            if (result.Errors.Count() > 0) // 5 marks
            {
                StringBuilder errorList = new StringBuilder("The following account registration errors need to be resolved. ");

                foreach (var error in result.Errors)
                {
                    errorList.Append($"{error.Code}: {error.Description}");
                }

                return BadRequest($"{errorList}");
            }
        }
        catch(Exception ex)
        {
            return StatusCode(StatusCodes.Status500InternalServerError, $"The following error occured {ex.Message}"); // part of try catch 2 marks
        }
    }
    else
    {
        return Conflict($"The username '{uvm.username}' already exists. Please use a different username");  // 1 mark
    }

    return Ok($"Your account '{uvm.username}' was created successfully. You may proceed with logging in"); // 1 mark
}
```


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 pie 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 line 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 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: Do not run "npm install". 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

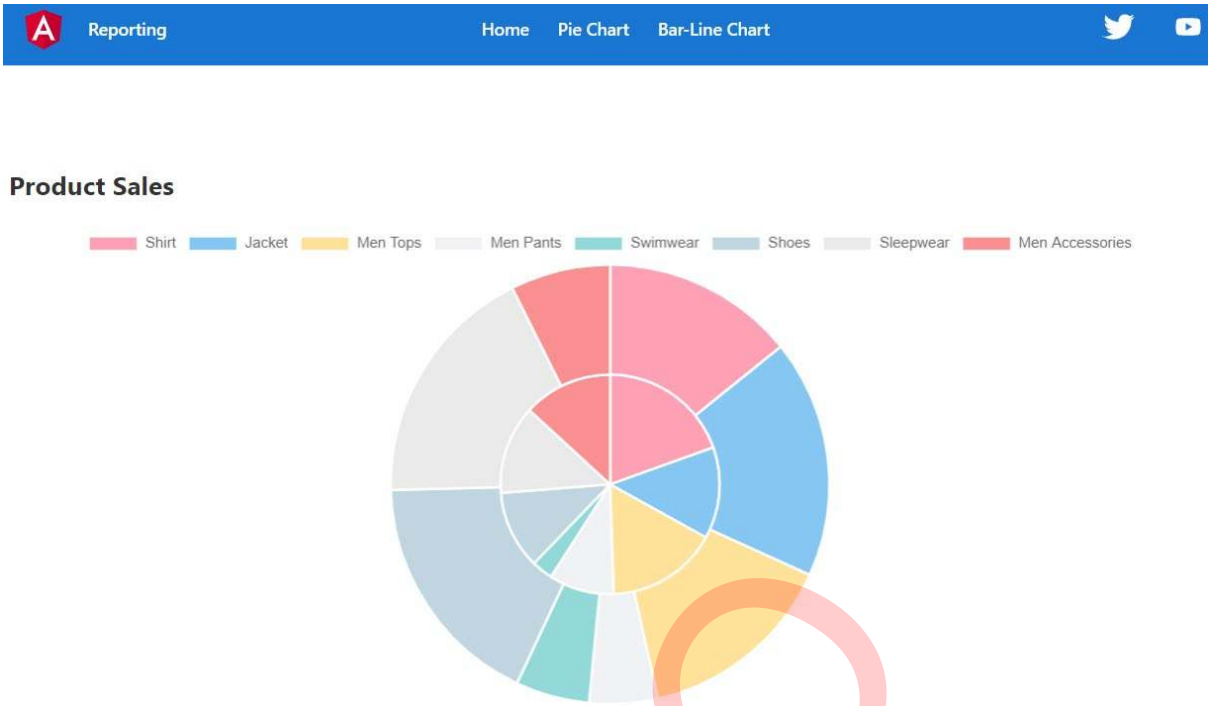


Figure 2: Pie Chart

Figure 3: Bar-Line Chart

SOLUTION SECTION B
File: app.component.html

```
<nav class: navbar navbar-expand-sm bg-success navbar-dark">
  <a class="navbar-brand" href="#"> Reporting </a>
  <button
  class="navbar-toggler"
  data-target="#mynav"|>
  <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="mynav">
  <ul class="navbar-nav mr-auto">
  <li class="nav-item">
  <a class="nav-1link" routerLink="home">Home</a> </li>
  <li class="nav-item">
  <a class="nav-1link" routerLink="pie-chart">Pie Chart</a></li>
  <li class="nav-item">
  <a class="nav-1link" routerLink="bar-line">Bar Line Chart</a> </li>
  </ul>
  </div>
</nav>
<!-- 4 Marks in total Note: routing must be working to get full mark-->
<!-- 1 mark for each of the tabs --> <!-- 1 mark for the reporting title -->
```

File: home.component.ts

```
export class HomeComponent {
  ngOnInit(): void {
  this.createChart(); // 1 mark
 }
 public chart: any;
 createChart(){
  this.chart = new Chart("barChart", {
   type: 'bar', // 1 mark
   data: {
    labels: ['Shirt', 'Jacket', 'Men Tops','Men Pants',
        'Swimwear', 'Shoes', 'Sleepwear','Men Accessories'], // 1 mark
     datasets: [
      {
       label: "2022",
       data: ['446','551', '462', '158', '171',
          '553', '566', '231'], // 1 mark
       backgroundColor: 'blue' // 0.5 mark
      },
      {
       label: "2023",
       data: ['623', '431', '525', '306', '100',
          '369', '417', '420'], // 1 mark
       backgroundColor: 'red' // 0.5 mark
      }
     ]
    }
   });
  }
}
```

File: home.component.html

```
<div class="chart-container">
  <h2>Product Sales</h2>
  <canvas id="barChart" >{{ chart }}</canvas>
</div> //2 marks
```

File: pie-chart.component.html

```
<div class="chart-container">
  <h2>Product Sales</h2>
  <canvas id="pie-chart" >{{ chart }}</canvas>
</div> //2 marks
```

File: pie-chart.component.ts

```
ngOnInit(): void {
  this.createChart();
}
public chart: any;
createChart(){
  this.chart = new Chart("pie-chart", {
    type: 'pie', //1 mark
    data: {
      labels: ['Shirt', 'Jacket', 'Men Tops','Men Pants',
      'Swimwear', 'Shoes', 'Sleepwear','Men Accessories'], //1 mark
       datasets: [
        {
          label: "2022",
          data: ['446','551', '462', '158', '171',
              '553', '566', '231'], //1 mark
        },
        {
          label: "2023",
          data: ['623', '431', '525', '306', '100',
               '369', '417', '420'], //1 mark
        }
      ]
    },
    });
  }

}
```

File: bar-line.component.html

```
<div class="chart-container">
  <h2>Product Sales</h2>
  <canvas id="bar-line" >{{ chart }}</canvas>
</div> //2 marks
```

File: bar-line.component.ts

```
ngOnInit(): void {
  this.createChart();
}
public chart: any;
createChart(){
  this.chart = new Chart("pie-chart", {
    type: 'line', //1 mark
    data: {
      labels: ['Shirt', 'Jacket', 'Men Tops','Men Pants',
      'Swimwear', 'Shoes', 'Sleepwear','Men Accessories'], //1 mark
       datasets: [
        {
          label: "2022",
          data: ['446','551', '462', '158', '171',
              '553', '566', '231'], //1 mark
        },
        {
                  type: 'bar', //1 mark
          label: "2023",
          data: ['623', '431', '525', '306', '100',
               '369', '417', '420'], //1 mark
        }
      ]
    },
    });
  }

}
```


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

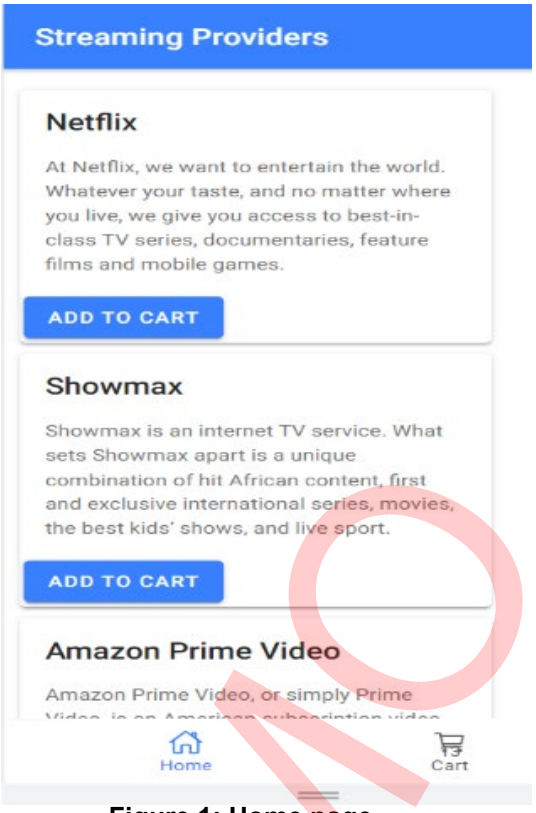


Figure 1: Home page

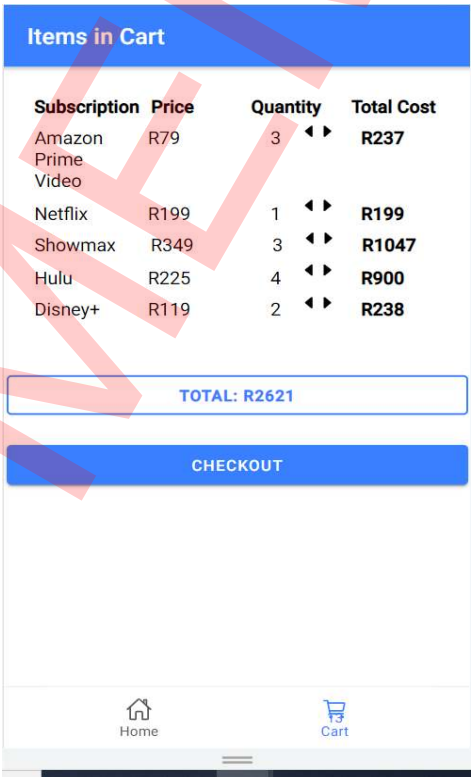


Figure 2: Cart page

SOLUTION SECTION C

File: tabs.page.html

```
<ion-tabs>
  <ion-tab-bar slot="bottom"> <!-- 1 Mark  -->
    <ion-tab-button tab="home">
      <ion-icon name="home-outline"></ion-icon>
      <ion-label>Home</ion-label>
      <!-- 1 Mark  -->
    </ion-tab-button>
    <ion-tab-button tab="cart">
      <ion-fab>{{numCartItems}}</ion-fab> <!-- 1 Mark  -->
        <ion-icon name="cart-outline"></ion-icon
      <ion-label>Cart</ion-label><!-- 1 Mark  →
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
<!-- 4 Marks in total -->
```

File: tabs-routing.module.ts

```
    path: 'tabs',
    component: TabsPage,
    children: [ //1 Mark

  {
    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'
}
];
```

File: home.page.html

```
<ion-header>
  <ion-toolbar color="primary">
    <ion-title>Streaming  Providers</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content>
   <ion-list>     <!-- 1 mark for the header and title -->
    <div class="card m-3" style="width: 20rem;" *ngFor="let prod of products">
<!-- 1 mark -->
          <ion-card>
            <ion-card-header>
              <ion-card-title>{{prod.name}}</ion-card-title>
            </ion-card-header>            <!-- 2 marks -->
    <ion-card-content>{{prod.description}}</ion-card-content> <!-- 1 mark --> <ion-
button (click)="addSubscriptionToCart(prod)">Add to cart</ion-button>
<!-- 2 marks -->
          </ion-card>
        </div>
      </ion-list>
    </ion-content>
```

File: cart.page.html

```
<ion-header>
    <ion-toolbar color="primary">
      <ion-title>Items in Cart</ion-title>
      <ion-buttons slot="end">
        <ion-button fill="clear">
     <ion-icon name="power-outline" color="white" slot="icon-only"></ion-icon>
        </ion-button>
      </ion-buttons>
    </ion-toolbar>
    </ion-header>     <!-- 2 marks for the header and title-->
    <ion-content class="ion-margin">
        <ion-grid class="ion-margin">
          <ion-row>
            <ion-col><b>Subscription</b></ion-col>
            <ion-col><b>Price</b></ion-col>
            <ion-col><b>Quantity</b></ion-col>
            <ion-col><b>Total Cost</b></ion-col>
        </ion-row>              <!-- 2 marks -->
        <ion-row *ngFor="let cartSub of subscriptionsInCart">
            <ion-col>{{cartSub.subscription.name}}</ion-col>
            <ion-col>R{{cartSub.subscription.price}}</ion-col>
            <ioncol>{{cartSub.quantity}}</ion-col>
            <!-- 2 marks -->
            <ion-icon name="caret-back-outline"
(click)="reduceProdCount(cartSub.subscription)"></ion-icon>
            <ion-icon name="caret-forward-outline"
(click)="increaseProdCount(cartSub.subscription)"></ion-icon>
 <ion-col><b>R{{cartSub.totalCost}}</b></ion-col>
          <!-- 3 marks -->
         </ion-row>
        </ion-grid>
       <ion-col size="6">
           <ion-button fill="outline" expand="block" color="primary"><b>Total:
R{{totalCostOfSubcriptionsInCart}}</b></ion-button>
         </ion-col>
    <!-- 1 mark -->
      <ion-col size="6">
         <ion-button  expand="block" (click)="setOpen(true)">Checkout</ion-button>
         <ion-modal [isOpen]="isModalOpen">
           <ng-template>
             <ion-header>
               <ion-toolbar>
                 <ion-title></ion-title>
                 <ion-buttons slot="end">
                   <ion-button (click)="setOpen(false)">Close</ion-button>
                 </ion-buttons>
               </ion-toolbar>
             </ion-header>
             <ion-content class="ion-padding">
               <p>
                 Payment Successful
               </p>
             </ion-content>
           </ng-template>
         </ion-modal>
       </ion-col>
       <!-- 3 marks in total -->
     <!-- 2 marks for implementing modals and 1 mark for payment successful message -->
```


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: Do not run "npm install". 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

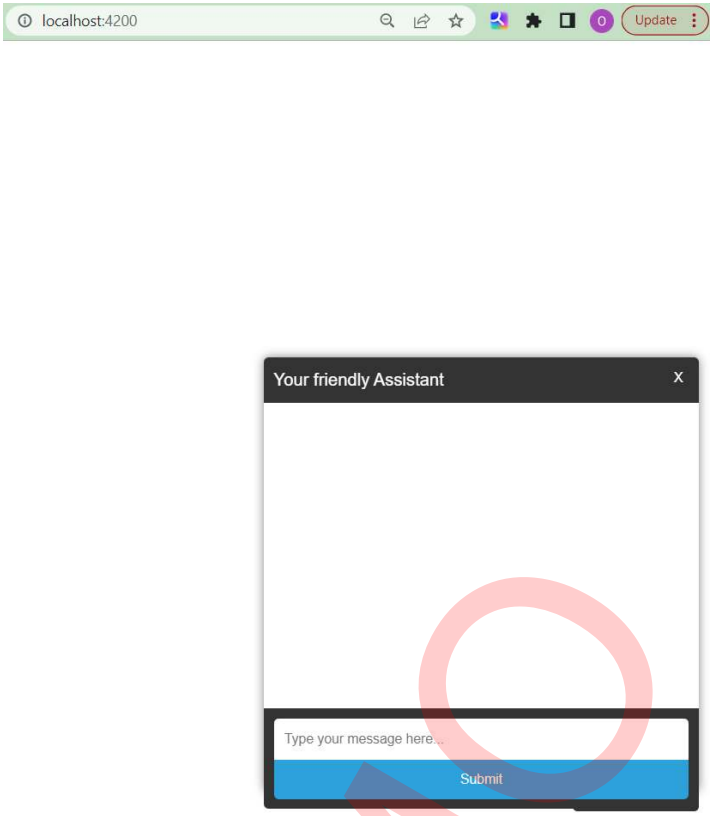


Figure 1: Before conversation

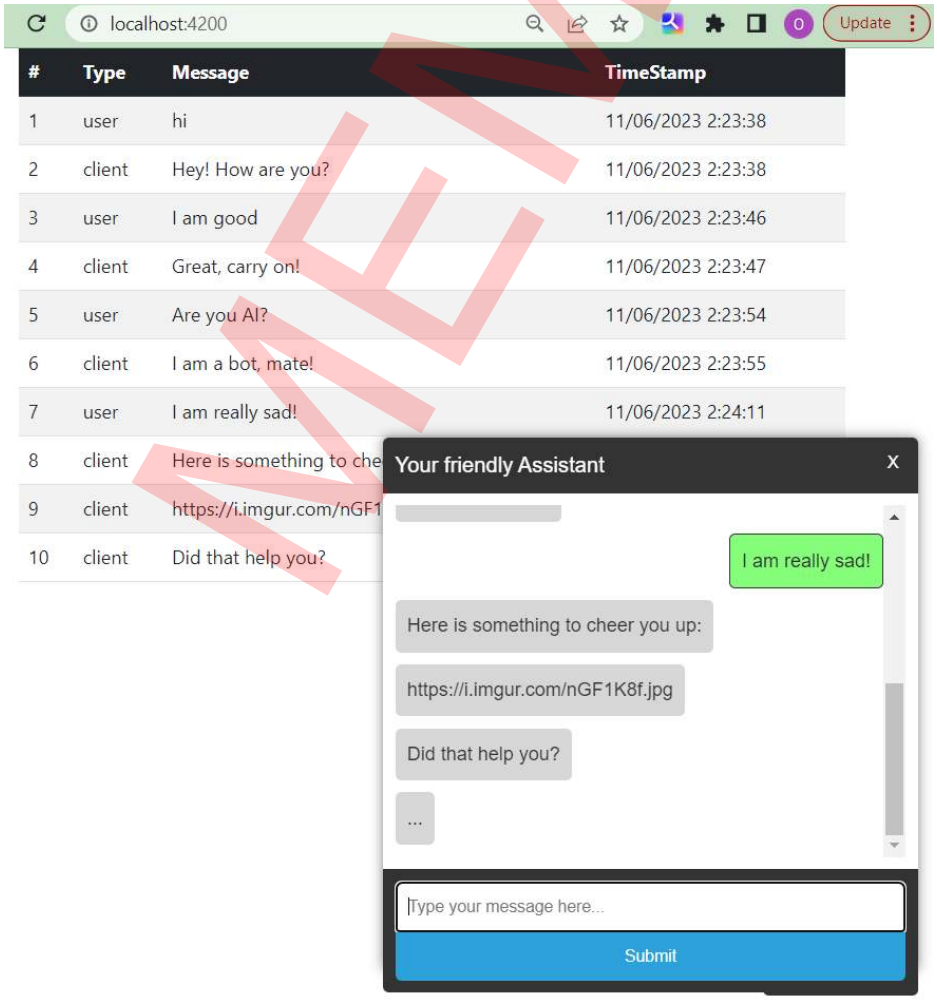


Figure 2: During conversation

SOLUTION SECTION D

File: chat-support.component.ts

```
// 5 marks in total for this section
export interface Message {
  type: string;
  message: string;
  dateTimeStamp: string;  // 1 mark
}
```

```
export class ChatSupportComponent {
  isOpen = false;
  loading = false;
  currentDateTime:any   // 1 mark
  messages: Message[] = [];
  chatForm = new FormGroup({
    message: new FormControl('', [Validators.required]),
  });
```

```
sendMessage() {
  this.currentDateTime=this.getDateTimeStamp(); // 1 mark for both the getDateTimeStamp calls
  const sentMessage = this.chatForm.value.message!;
  this.loading = true;
  this.messages.push({
    type: 'user',
    message: sentMessage,
    dateTimeStamp: this.currentDateTime // 1 mark for both the currentDateTime assignments
  });
  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.currentDateTime= this.getDateTimeStamp(); // 1 mark for both the getDateTimeStamp calls
     this.messages.push({
        type: 'client',
        message: message,
        dateTimeStamp: this.currentDateTime // 1 mark for both the currentDateTime assignments
      });
      this.scrollToBottom();
  }

  getDateTimeStamp() // 1 mark for getting the date
  {
    return this.datepipe.transform((new Date), 'dd/MM/yyyy h:mm:ss')
  }
```

File: chat-support.component.html

```
<!-- 10 marks for this section -->
<!-- 1 mark for adequate styling-->
<div class="container" *ngIf="messages.length > 0"> <!-- 1 mark for ngif -->
    <table class="table table-striped table-hover align-middle"> <!-- 1 mark -->
      <thead class="table-dark"> <!-- 1 mark for table head section with relevant headings-->
      <tr>
        <th>#</th>
        <th>Type</th>
        <th>Message</th>
        <th>TimeStamp</th>
      </tr>
    </thead>
    <tbody> <!-- 1 mark for table body section-->
      <tr *ngFor="let item of messages; let i = index"> <!-- 1 mark for the ngfor-->
        <td>
            {{i + 1}} <!-- 1 mark -->
        </td>
        <td>
            {{ item.type }} <!-- 1 mark -->
        </td>
        <td>
          {{ item.message }} <!-- 1 mark -->
        </td>
        <td>{{item.dateTimeStamp}}</td> <!-- 1 mark -->
      </tr>
    </tbody>
    </table>
</div>
```


SECTION D TOTAL **15**