INF 354 Memo Section B & C

Section A

Question 1: Web.config

```
<add name="RENTALEntities" connectionString="metadata=res://*/Models.Rental.csdl|res://*/Models.Rental.ssdl|res://*/Models.Rental.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=.;initial catalog=RENTALS;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />
```

```
;data source=.;initial catalog=RENTALS;integrated security=True;M
```

Question 2: Rentals Controller

a. Annotations

```
[HttpPost]                  // 1 mark
[Route("RentStuff")]        // 1 mark
```

b. Lambda Query

```
var dblines = db.RentalLines.Include(item => item.Rentable)
    .Where(l => demandRentableIDs.Contains(l.RentableID)
    && (l.EndDate >= demandMinDate && l.StartDate <= demandMaxDate)).ToList();
// 1 mark for db.RentalLines and Where
// 1 mark for including rentable
// 1 mark for .contains
// 1 mark for date range condition
// 1 mark for ToList
```

c. Creating Rental and associated RentalLine database records

```
//↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
Rental rental = new Rental();                   // 1 mark
rental.CustomerID = request.customerid;         // 1 mark
rental.Date = DateTime.Now.Date;                // 1 mark
rental.RentalLines = request.lines;             // 1 mark

db.Rentals.Add(rental);                         // 1 mark
db.SaveChanges();                               // 1 mark
//↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
```

Question 3: Rentals Controller

a. Annotations

```
//↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
[HttpGet]                                  // 1 - Mark
[Route("GetLastCustomerRental")]           // 1 - Mark
//↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
```

b. Query

```
/*Provide Lambda query to get latest rental transaction from customer.
 * As part of the rental object, the associated Customer and RentalLine objects should be included.
 * As part of each RentalLine object returned, the associated Rentable object should be included:  */
//↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
Rental rental = db.Rentals.Include(i=>i.Customer).Include(i => i.RentalLines.Select(rl=>rl.Rentable))
    .Where(r => r.CustomerID == customerid).OrderByDescending(r=>r.Date).FirstOrDefault();
// 1 mark for declaration of Rental variable >> Rental rental = ...
// 1 mark for db.Rentals with Where
// 1 mark for inclusion of child Customer
// 1 mark for inclusion of grandchild Rentable
// 1 mark for OrderByDesc Date
// 1 mark for FirstOrDefault
//↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
```

c. Dynamic object creation

```
/*Provide code to populate the dynamic object created above. It should contain:
 * Name, Email, and Cell of Customer
 * RentalID and Date of Rental
 * Array of RetailLines, each of which includes:
 *      - Description and DailyPrice of Rentable
 *      - StartDate and EndDate RentalLine
 *      - Days calculate by working out the number of days that the Rentable will be rented based on Start and End Date. StartDate and EndDate should be included.
 *      - LinePrice calculated by multiplying the daily price with the number of days the item will be rented.
 *
 * Total rental amount calculated by adding together the price of each line item
 */
//↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
response.Name = rental.Customer.Name;
response.Email = rental.Customer.Email;
response.Cell = rental.Customer.Cell;
response.RentalID = rental.RentalID;
response.Date = rental.Date.ToShortDateString();
response.Total = (rental.RentalLines.Sum(rl => rl.Rentable.PricePerDay * ((rl.EndDate - rl.StartDate).Days + 1))).ToString("C", CultureInfo.CreateSpecificCulture("en-ZA"));
response.RentalLines = rental.RentalLines.Select(rl => new {
    Rentable = rl.Rentable.Description,
    DailyPrice = rl.Rentable.PricePerDay.ToString("C", CultureInfo.CreateSpecificCulture("en-ZA")),
    StartDate = rl.StartDate.ToShortDateString(), EndDate = rl.EndDate.ToShortDateString(),
    Days = ((rl.EndDate - rl.StartDate).Days + 1),
    LinePrice = (rl.Rentable.PricePerDay * ((rl.EndDate - rl.StartDate).Days + 1)).ToString("C", CultureInfo.CreateSpecificCulture("en-ZA"))
}).ToList();
// You don't have to mark answer incorrect or deduct any marks if they don't have the CultureInfo stuff
//  1 marks for adding Name, Email, Cell, RentalID and Date to dynamic object
//  3 marks for adding Total - use of Sum(); PricePerDay multiplied by Number of days; Calculation of number of days
//  3 marks for adding list of RentalLines - Select() function used; inclusion of all relevant fields in selected object; calculation of line price
//↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
```

Section B

Question 1: Removed

Question 2 (question2.component.scss):

question2.component.scss

C: > Users > Ziel > Desktop > Moderation Files > question2.component.scss > ...

```scss
// Complete this section - DO NOT CHANGE ANY OTHER FILES & ONLY UPLOAD THIS FILE
#headers {
    margin: auto;
    text-align: center;   // 1 Mark for centering the header, does not need to look exactly like this.
}

$mainColor : green; // 2 marks for having variable and colour to green.. If colour is not green I only gave 1 mark

#paragraph {
    text-align: center;
    border: 1px solid black;
    margin: auto;
    font-family: "Bitstream Vera Serif Bold";
    color: $mainColor;
}

#button {
    background-color: $mainColor;  // 2 marks if Paragraph, button and th have variable colours. if colour is just changed to green only 1 mark was given
    // Enter your code here
}

th {
    background-color: $mainColor;
    // Enter your code here
}

#buttonClicked {                    // Make sure it is for buttonClickedID - I subtracted 1 mark if it was wrong ID used
    background-color: green;        // 1 Mark for background-color Green, if variable is used it is still fine.
    color: black;                   // 1 Mark for color = Black. or any way to change text colour to black.
}

tr:nth-child(even) {background: #CCC}  // 0.5 marks for having some sort of even property on rows on the table and 0.5 marks setting the colour to #CCC or #CCCCCC

//Explain here
// It is not robust to use local variables for anything that is referenced more than once, because when you decide          // 2 Marks for explanation
// to change that local variable you only have one point to change and not many
// anything along those lines
//
//
//
//

```

Question 3 (Question3.component.ts):

```typescript
//Only modify this file

import { Component, OnInit } from '@angular/core';
//Import the necessary modules to allow access to the service and to the router module
import { Router } from '@angular/router';    // 0.5 mark
import { HttpService } from '../http.service'; // 0.5 Mark



@Component({
  selector: 'app-question3',
  templateUrl: './question3.component.html',
  styleUrls: ['./question3.component.scss']
})
export class Question3Component implements OnInit {

  constructor(private router: Router, private apiService : HttpService) { } // 2 marks   - If constructor and imports are correct it is 3 marks.
  //Add the necessary members to the constructor class to allow routing and access to the service

  Beer : any;
  Button2Clicked : boolean = false;
  MultipleNumber : number

  ngOnInit(): void {
    // In the http Service File is a function cal getBeerData that is linked to a public API that fetches Beer Data
    // Write the code to assign that data to a variable called Beer
    this.apiService.getBeerData().subscribe( res => {                          // 2 mark for getBeerData().Subscribe()
      this.Beer = res;                                                         // 1 Mark
    })
  }

  Button1() {
    //Navigation. when this button is clicked reroute the page to go to the Question1 page
    this.router.navigate(['/question1']);          // 2 marks can be different does not have to use naviagte property. NavigateByUrl also works and 'question1' also works.
  }

  //This button receives 2 number parameters. Return the multiplication of this number and bind it to variable MultipleNumber
  // Complete the function so that it also only receives numbers as input and returns nothing and is a void function

  Button2(num1 : number, num2 : number) : void  {                    // 2 Marks - 1 Mark for having : number next to the nums and 1 Mark for having : void
    // Complete code here
    this.Button2Clicked = true;
    this.MultipleNumber =  num1 * num2;
  }

}

```

Question 4 (http.service.ts):

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { ZipOperator } from 'rxjs/internal/observable/zip';

// ---------- CODE HERE FOR QUESTION 4 ------------------
// const httpOptions = {
//   headers: new HttpHeaders({                              // 1 mark for attempting - 2 marks for having  headers: and new HttpHeaders.
//     "Access-Control-Allow-Origin": '*'                    // 2 marks - 1 for "Access-Control-Allow-Origin" and one for the localhost in the question or *
//   })
// }
// Something along those lines
//(Note if other method that was used for express or something I deducted one mark as it is for a different technology)
// -----------------------------------------------------


@Injectable({
  providedIn: 'root'
})
export class HttpService {
  BreweryAPI = 'https://api.openbrewerydb.org/breweries/';
  jsonObject : Object = {}
  constructor(private http : HttpClient) { }


  getBeerData() {
    return this.http.get(this.BreweryAPI)
  }

  //Create your functions HERE
  CreateDummyData() {
    // enter code here
    return this.http.post('http://dummy.restapiexample.com/api/v1/create', this.jsonObject);
  }
```

Section C:

Question 1 (reporting.component.ts):

```typescript
//MAKE THE APPROPIATE CHANGES TO THIS FILE
// USE npm install chart.js --save
// and npm install @types/chart.js .. to install chart.js

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

//Import code here
import * as Chart from 'chart.js';

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

  constructor() { }

  chart : any = [];

  ngOnInit(): void {
  }

  // Fill in the objects with data in document
  headers = ["Jacques","Dwayne","Ziel","Phil","Chane","Bob"]      // 1 mark for creating header. Can be like mine or ["Name", "Age"]
  data = [27,24,23,45,34,54]                                     // 1 mark for data. If above is like mine data needs to like mine as well. Otherwise Data needs to be [{"name: Jacques", "age": 27},{"name: Dwayne", "age": 24}] and so on....


  GenerateChart() {
    // Create your chart here - use the chart variable initiliazed above
    this.chart = new Chart("canvasID", {            // 1 mark for this.chart = new Chart and "canvasID or canvas"
      type: 'bar', // 1 mark if type: is bar .. this needs to be included
      data: {
          labels: this.headers,  // or this.data.name or this.data['name'] depends on how they created their objects
          datasets: [{
              label: 'Age of Users', // 1 mark
              data: this.data, // or this.data['age'] or this.data.age
              barPercentage: 0.8, // 1 mark
              backgroundColor: 'cyan',  // 1 mark with border colour
              borderColor: '#000080',
              borderWidth: 1
          }]                                                     // Not everything needs to be where I put it, it can be in different places as long as they have the correct terms it is fine
      },
      options: {
          scales: {
              yAxes: [{
                  ticks: {
                      beginAtZero: true, // 1 mark for both BeginAtZero and max... BeginAtZero could also be min: 0
                      max : 100
                  }
              }]
          }
      }
    });
  }
}

```

Question 2 (security.component.ts):

a. Role

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

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

  constructor() { }
  Username : string;
  Password : string;
  UserLoggedIn : any;

  AdminRoleTrue: boolean;
  EmployeeButtonClicked: boolean;
  LoginClicked : boolean;

  //Add Role to this Objects
  Users = [
    {
      "Username": "Ziel",
      "Password": "Ziel123",
      "AccessToken": "3Z2I1LE",
      "Role": "Admin"    // 1 mark for adding This row.
    },
    {
      "Username": "Phil",
      "Password": "Phil123",
      "AccessToken": "P3H2I1L",
      "Role": "Employee"  // 1 Mark for adding this row both here and the object below
    },
    {
      "Username": "Jacques",
      "Password": "Jacques123",
      "AccessToken": "3JA21CSQEU",
      "Role": "Employee"
    }
  ]

  ngOnInit(): void {
  }

  // Do not Change
  Login() {
    this.Users.forEach(element => {
      if (element.Username == this.Username && element.Password == this.Password) {
        localStorage.setItem("accessToken", element.AccessToken);
        this.LoginClicked = true;
        setTimeout(() => {
        this.LoginClicked = false
        }, 3000);
      }
    });

  }
```

b. AdminRole function implementation

```
// Do not Change
Login() {
  this.Users.forEach(element => {
    if (element.Username == this.Username && element.Password == this.Password) {
      localStorage.setItem("accessToken", element.AccessToken);
      this.LoginClicked = true;
       setTimeout(() => {
       this.LoginClicked = false
       }, 3000);
    }
  });

}


//Do not Change
FakeLogout() {
  localStorage.removeItem("accessToken");
  this.AdminRoleTrue = false;
  this.EmployeeButtonClicked = false;
}

// Implment AdminRole Code here.
AdminRole() {
  this.Users.forEach(element => {                                    // 1 mark for some sort of loop
    if (localStorage.getItem("accessToken") == element.AccessToken) {   // 1 marks for if they used localStorage
      if (element.Role == "Admin") {                                 // 1 mark for and if statement to check if Admin
        this.AdminRoleTrue = true;                                   // 1 mark if the variable is same as mine or person made external one and called it true.
      }
    }
  });
}

// Implment EmployeeRole Code here.
EmployeeRole() {
  this.EmployeeButtonClicked = true;         // 1 Mark If there is an if to check for employee give 0, because admin should also be able to access this function
}

}

```

Section D:

a. Method to translate text to morse code

```
#Method to translate text to morse code
def translateText(self, text):
    output = ""
    words = text.split()                                    # ← 1 mark for splitting text into words list
    for i, word in enumerate(words):                        # ← 1 mark for 2-dimensional loop to loop through each word, each letter
        for j, c in enumerate(word):                        #       (or equivalent e.g. while or recursion)
            key = c.upper()
            if key in self.CharMorseMap:
                output+=self.CharMorseMap[key]              # ← 1 mark for accessing dictionary to do translation
            else:
                return "Text included invalid characters."  # ← 1 mark for handling potential invalid characters
            if j+1 < len(word):                             # ← 1 mark to add space after every letter
                output+= " "
        if i+1 < len(words):                                # ← 1 mark to add / after every word
            output+=" / "

    return output
```

b. Instantiation of Translator class

```
#Instantiation of Translator class
translator = Translator()                   # ← 1 mark for object creation
```

c. Managing user input

```
#Depending on user's choice, prompt user to enter text or morse code and pass the entered value into the relevant translation method
#Cater for invalid user inputs
if choice == "1":                                       # ← 1 mark handling of choice 1
    text = input("Enter words:")
    print(translator.translateText(text))
elif choice == "2":                                     # ← 1 mark handling of choice 2
    morse = input("Enter morse code:")
    print(translator.translateMorse(morse))
elif choice == "3":                                     # ← 1 mark handling of choice 3
   print("Bye Bye")
   exit()
else:
    print("\r\n----------------------------------------\r\nThat choice does not exist.")
    run()
```