## DEPARTMENT OF INFORMATICS

# INF 354

**JULY EXAM (1)** **DATE: 2021-07-16**

**Examiners** : Dr. JP van Deventer
: Mr. R Hanslo
: Mr. B Rampete

**Moderator / External Examiner** : Dr Johan Breytenbach
University of the Western Cape

**Time** : 180 min

**Marks** : 30

| 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 | X | X | X | X | X | 10 | 10 |
| Section B | X | X | X | X | X | X | X | 10 | 10 |
| Section C | X | X | X | X | X | X | X | 10 | 10 |
| Total | | | | | | | | 30 | 30 |

**Instructions**

1. This paper consists of 3 sections with several main questions (sub-sets of instructions) each.
2. Each section relates to a small semi-complete program that needs to be updated or finalised.
3. Each question relates to a file in one of the programs that you need to update or finalise.
4. Each sub-sets of instructions relates to activities and tasks in one of the files.
5. Each main question relates to a file that needs to be edited in one of the three programs you have been provided with.
6. Answer all the questions – there are no optional questions.
7. Please read all questions, instructions and sub-sets of tasks very carefully.
8. After completing work on a relevant question, please upload ONLY the edited file to the correct upload area.

The University of Pretoria commits itself to produce 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 – ANGULAR (10)

## BACKGROUND TO SECTION

Students are provided with a file named **INF354ExamSectionA_studentCopy.zip** which contains all the files necessary to complete the questions stipulated in this section. This section contains 2 main questions each relating to 1 file within the project provided.

In this section you will be completing the functionality to successfully log a user into an application. The Angular application uses a list of fake users that one would need to authenticate against. The navigation bar would act as the main indicator that the user has logged in. If and when you have completed all of the questions in this section, the state of the user (either logged in or not) should clearly be shown in the application.

```
const FAKE_USERS: User[ ] = [
 {
  email: 'boitumelo@gmail.com',
  password: 'mypassword',
  role: 'Admin'
 },
 {
  email: 'ridewaan@gmail.com',
  password: 'ridewaanpassword',
  role: 'User'
 },
 {
  email: 'phil@gmail.com',
  password: 'philpassword',
  role: 'Admin'
 },
];
```

**Fake users that you can use to log-into the application. This array can be found in the main.service.ts**

## PROJECT PREPARATION

- Download **INF354ExamSectionA_studentCopy.zip** file.
- Copy INF354ExamSectionA_studentCopy.zip to the root directory (for example, your C: or D: drive) of your computer and then unzip the file.
- Open the subsequent folder in Visual Studio Code.
- Make sure to complete the following installs to reconstitute the required node modules ➔ npm install

## GENERAL TASKS AND REQUIREMENTS

Your task is to complete the code base in order to get this application to function as described below. The steps required to achieve this have been broken down into separate questions. For each question, please do the following:

- Read the instructions carefully.
- Look for comments (hints) in the provided sample solution. The comments will guide you in terms of where the code modifications are required.
- Apply the necessary code changes to the specified file in the required application.
- Only upload the modified file associated with the question to ClickUP.

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

## QUESTIONS FOR SECTION A

Please complete the following questions. Please be careful to save regularly and to upload the correct files.

**SECTION A - QUESTION 1** **(7 MARKS)**

Only upload the **main.service.ts** file after completing this question.

This Angular application makes use of the BehaviorSubject paradigm to hold the logged in state of our application. If the value associated with the BehaviorSubject is true then the user is logged in, and if false, the user will not be logged in.

**1.1.** In this question, you would be required to complete a **get** property that will expose the private BehaviorSubject **bs** to the relevant component/s. This property should have a name getLoginState() and have a return type of Observable<boolean>. (3 marks).

**1.2.** You are required to complete a function with the name Login( ). This function, should accept a user object that will take in a parameter named user (4 marks).

- This parameter will be of type User (see the interfaces folder). In this function, you will be required to validate the incoming user object against the users in the FAKE_USERS array.
- If the user is valid / found, the value of the BehaviorSubject should be updated to true and an Observable of true should be returned by the function.
- If the user is invalid an Observable of false should be returned. This function should have a return type of Observable<boolean>.
- HINT: use the of () function that rxjs provides helps to convert any value to an observable.

### SOLUTION SECTION A - QUESTION 1.1.

```
private bs = new BehaviorSubject<boolean>(false);
constructor() { }
/**
 * @returns Observable<boolean>
 * 1.1 Complete the getLoginState() get property that will exponse the BehaviorSubject here:
 *
 */
get getLoginState(): Observable<boolean> {
  return this.bs.asObservable();
}
```

### SOLUTION SECTION A - QUESTION 1.2.

```
Login(user: User): Observable<boolean> {
  const foundUser = FAKE_USERS.find(x => x.email === user.email && x.password === user.password);
  if (foundUser) {
    this.bs.next(true)
    return of(true);
  }
  return of(false);
}
```

## SECTION A - QUESTION 2 (3 MARKS)

Only upload the **top-nav.component.html** file after completing this question.

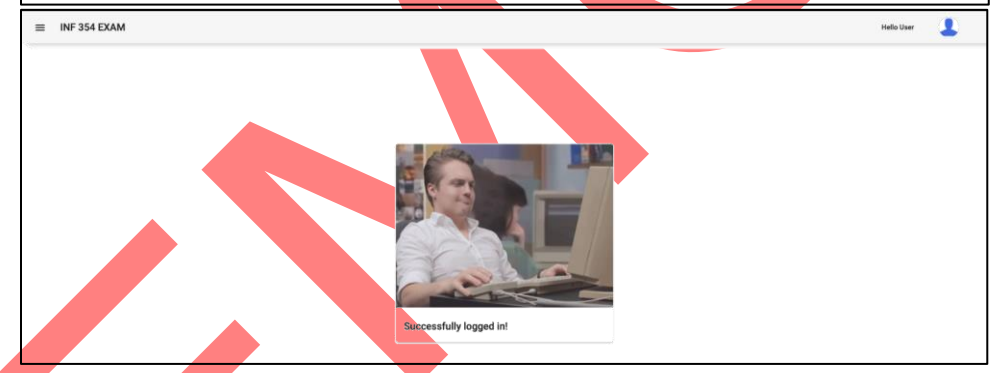
In this question, you will be required to complete the HTML page by implementing the correct structural directive to **conditionally** show a piece of HTML content in the navigation bar if the user has either logged in or not.

- The directive will be based on the loggedIn property in the top-nav component.
- If the user has been logged in successfully, the logged in content should be displayed.
- If the user is not logged in then the 'not logged in' content should be displayed

**Expected Output (SECTION A)**

**Figure 1: Not logged in**


**Figure 2: Logged in**






### SOLUTION SECTION A - QUESTION 2

```
<!-- Complete the structural directive to show the logged in content if the user has logged in
        and to show 'You're not logged in' if the user is not logged in -->
    <ng-container *ngIf="loggedIn; else elseTemplate">
      <!-- logged in content -->
      <div class="nav-controls">
          <button mat-button> Hello User</button>
          <div class="image-container">
            <img src="assets/user.png" alt="UserProfile">
          </div>
        </div>
    </ng-container>
    <ng-template #elseTemplate>
        <!-- Not logged in content -->
        <button mat-button>You're not logged In</button>
    </ng-template>
    <!-- ////////////////////////////////////// End ////////////////////////////////////// -->
```

**SECTION A TOTAL** **10**

# SECTION B – ADVANCED REPORTING (10)

## BACKGROUND TO SECTION

In this section a student will be required to complete an advanced report that includes grid based charts as well as data tables where tables are presented in other tables. The charts and tables obtain data from JSON data stored in *.ts files within the student exam project.

Students are provided with a file named **INF354ExamSectionB_studentCopy.zip** which contains all the files necessary to complete the questions stipulated in this section. This section contains 4 main questions each relating to 1 file (refer to table 1 below) within the project provided.

After completing each question students will be required to upload only the associated file to the specific question's upload area.

## PROJECT PREPARATION

- Download **INF354ExamSectionB_studentCopy.zip** file.
- Copy INF354ExamSectionB_studentCopy.zip to the root directory (for example, your C: or D: drive) of your computer and then unzip the file.
- Open the subsequent folder in Visual Studio Code
- Make sure to complete the following installs to reconstitute the required node modules:
    - npm install
    - npm install chart.js --save
    - npm install @swimlane/ngx-charts --save
    - npm install @angular/cdk --save
    - npm install bootstrap
- **Please note** that after you have installed "npm install @swimlane/ngx-charts --save" Visual Studio Code will indicate certain errors. Installing "npm install @angular/cdk --save" will resolve these errors.
- After installing "npm install @angular/cdk --save" several warnings will be listed. The warnings listed are just warnings. It should not have an impact on running or executing the program when running "ng serve –o"

## QUESTION FILES AND MARK ALLOCATION

**Table 1: Question Files**

| QUESTION | FILE RELATED TO QUESTION | TOTAL MARKS |
|---|---|---|
| Question 1 | data-tables.component.html | **3** |
| Question 2 | app-routing.module.ts | **2** |
| Question 3 | app.component.html | **1** |
| Question 4 | tree-map-chart.component.ts | **4** |
| **TOTAL** | | **10** |



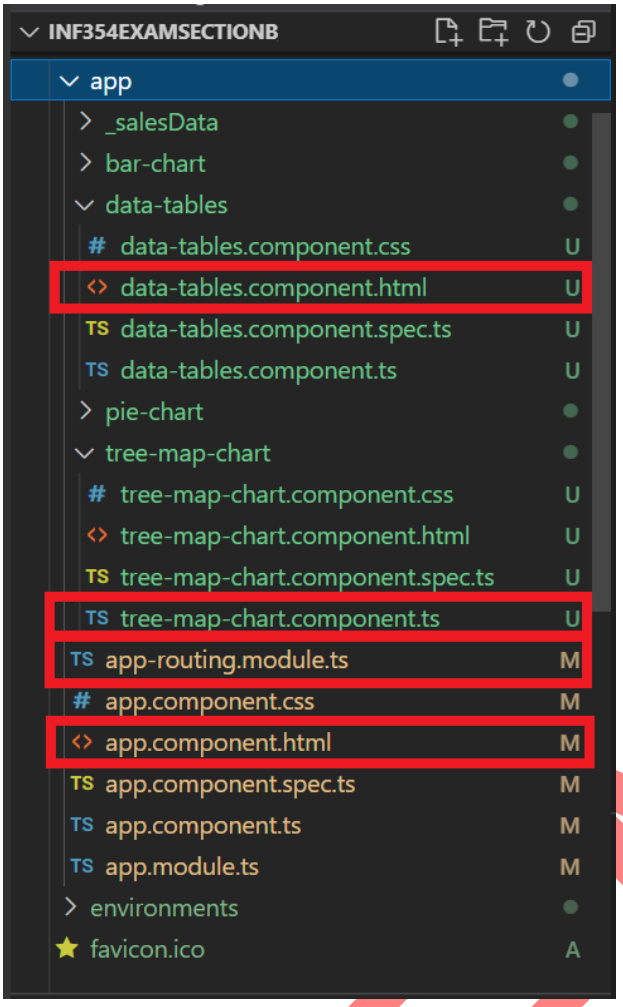
## GENERAL TASKS AND REQUIREMENTS

Your task is to complete the code base found in the reconstituted **INF354ExamSectionB_studentCopy folder** in order to get this application to function as described below. The steps required to achieve this have been broken down into separate questions. For each question, please do the following:

- Read the instructions carefully.
- Look for comments (hints) in the provided sample solution. The comments will guide you in terms of where the code modifications are required.
- Apply the necessary code changes to the specified file in the required application.
- Only upload the modified file associated with the question to ClickUP.

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

## PROJECT FILES STRUCTURE




Please only edit the files indicated in this image Do not edit other files as this would be unnecessary.

Do not edit or modify content of any files not listed below:

**Question 1 (Q1) = data-tables.component.html**
**Question 2 (Q2) = app-routing.module.ts**
**Question 3 (Q3) = app.component.html**
**Question 4 (Q4) = tree-map-chart.component.ts**

**Note on Question 4**:

- The ( tree-map-chart.component.html ) file content has been disabled to avoid loading errors.
- The content on this page needs to be enabled after completing the required edits in the ( tree-map-chart.component.ts ) file to view the page.



## RUNNING THE PROJECT FOR THE FIRST TIME AFTER FINAL PREPARATION (NG SERVE –O)

When running the project for the first time, you will be presented with the following screen. As you can see there are 4 sections that needs to be completed in the aforementioned file list to ensure that this report displays all the content found in the data folder correctly.


Please complete all the questions stipulated below to complete and populate the aforementioned screen with the appropriate details for the successful display and completion of this report.

## QUESTIONS FOR SECTION B

Please complete the following questions. Please be careful to save regularly and to upload the correct files.

**SECTION B - QUESTION 1** **( 3 MARKS )**

Only upload the **data-tables.component.html** file after completing this question.

**1.1.** Make use of a loop to access and display the main data in the multipart dataset in the JSON data so as to display ALL the relevant data as a table (1 mark). Please make sure to assign appropriate values in the headers to the data being displayed and assign data values by assigning the appropriate names found in the multipart dataset (1/2 mark).

**1.2.** Make use of a sub-loop to access the data subset (nested data) in the multipart dataset so as to display the data subset within each relevant associated data row linked to the data series linked to the main dataset (1 mark). Please make sure to assign appropriate values to the data being displayed in headers and data values (1/2 mark).

### Multipart data being accessed

```
export const productMultiDataOptions = [
  {
    "name": "Hamburgers",
    "series": [
      {
        "name": "January",
        "value": 567
      }, {
        "name": "February",
        "value": 554
      }, {
        "name": "March",
        "value": 789
      }
    ]
  }, {
```

### Hint and simplified description

Data need to be accessd in the multipart data file located in the project. In both instances of question 1, you need to make use of loops to access the main data ("name") and the data subset ("series") to display both sets of data as tables in tables.

Ensure the usage of the correct variable and data assignments in the loops as well as the associated data assignments.

### Expected output (Section B – Question 1)

If loops were added appropriately the provided sections then bootstrap will be implemented as seen below.

| Product | Monthly Sales | |
|---|---|---|
| Hamburgers | **Month** | **Units (Volume)** |
| | January | 567 |
| | February | 554 |
| | March | 789 |
| Fried Foodstuffs | **Month** | **Units (Volume)** |
| | January | 789 |
| | February | 456 |
| | March | 221 |
| Sandwhiches | **Month** | **Units (Volume)** |
| | January | 345 |
| | February | 123 |
| | March | 234 |

## SOLUTION SECTION B - QUESTION 1.1.

```
<tbody>
   <tr *ngFor = "let productMultiData of productMultiDataOptions">
     <td>{{ productMultiData.name }}</td>
     <td>
       <table class="table table-bordered table-hover table-sm">
         <thead class="table-primary">
           <tr>
             <th class="col-sm-6">Month</th>
             <th class="col-sm-6">Units (Volume)</th>
            </tr>
         </thead>
         <tbody>
```

## SOLUTION SECTION B - QUESTION 1.2.

```
<tbody>
  <tr *ngFor = "let nestedMultiData of productMultiData?.series">
   <td>{{ nestedMultiData.name }}</td>
   <td>{{ nestedMultiData.value }}</td>
  </tr>
</tbody>
```

## SECTION B - QUESTION 2 ( 2 MARKS )

Only upload the **app-routing.module.ts** file after completing this question.

Add appropriate app-routing and component routing paths to the app-routing.module.ts file ensure that all the relevant charts and the data table is routed and displayed on the app.component.html page when apps are assigned. Make sure to include the appropriate path as well as the appropriate component in the routing path.

## SOLUTION SECTION B - QUESTION 2

```
{path: 'data-tables', component: DataTablesComponent},
{path: 'bar-chart', component: BarChartComponent},
{path: 'pie-chart', component: PieChartComponent},
{path: 'tree-map-chart', component: TreeMapChartComponent}
```

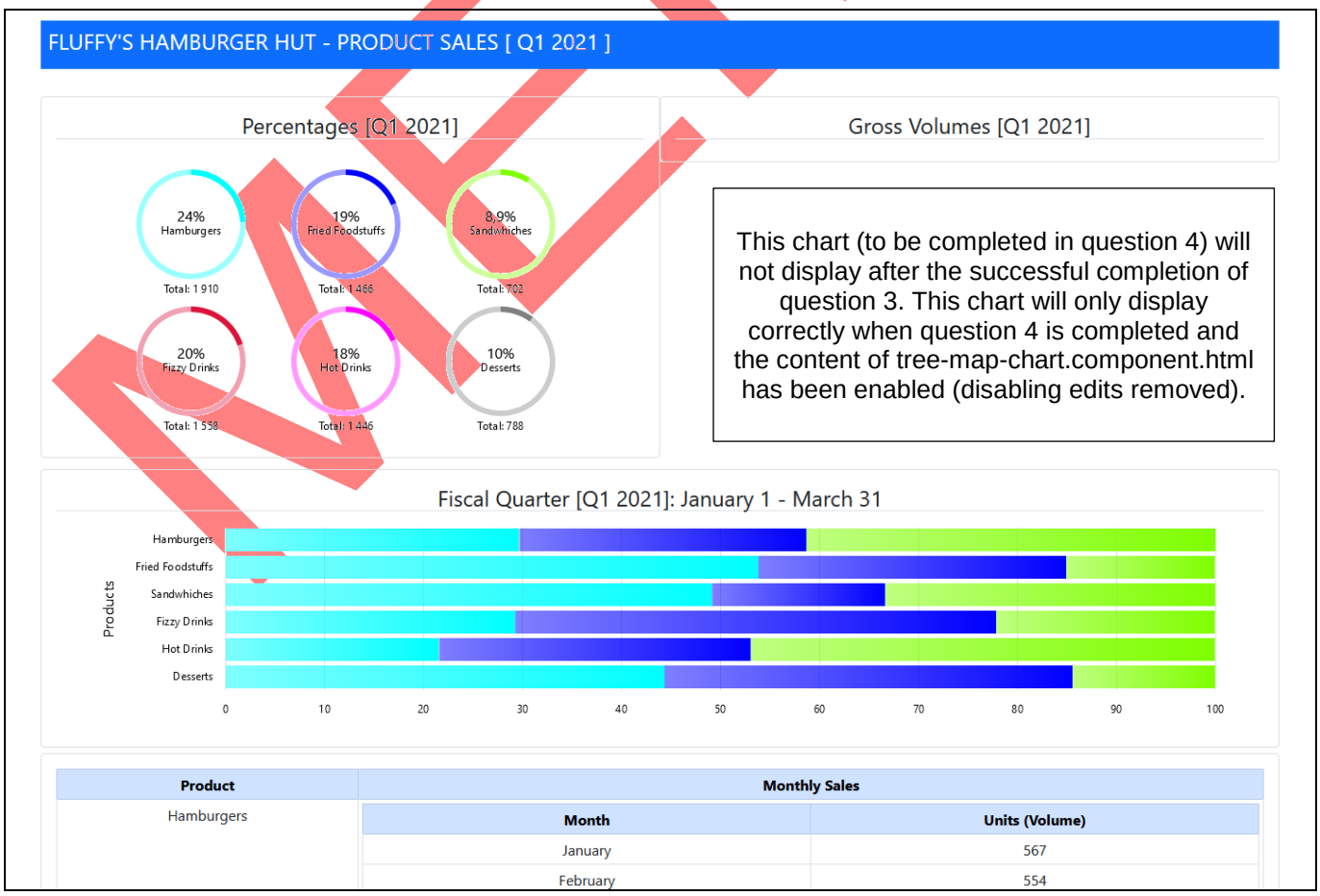
1/2 each for complete path. If path is not complete but only certain sections are completed a student can only get a maximum of 1/2 a mark. Routing is important and know how to do this properly is a concern.

## SECTION B - QUESTION 3 ( 1 MARK )

Only upload the **app.component.html** file after completing this question.

Add the relevant app modules to the app.component.html to display the following appropriately on the report. Ensure that you enable the Pie Chart, the Tree Map Chart, the Bar Chart and finally the Data Table.

### Expected Output (Section B – Question 2)



**SOLUTION SECTION B - QUESTION 3 (**1/4 mark per proper app assignment**)**

```
<div class="container">
  <div class="2col">
     <div class="c2">
       <div class="col-md-12">
         <div class="card text-left"><div align="center">
           <div class="card-body">
              <h4 class="card-title border-bottom">Percentages [Q1 2021]</h4>
<!-- ------------------------------------------------------------------ -->
<!-- Add 3.1 here -->
              <app-pie-chart></app-pie-chart>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="c3"><div class="card text-centre"><div align="center">
      <div class="card-body">
        <h4 class="card-title border-bottom">Gross Volumes [Q1 2021]</h4>
<!-- ------------------------------------------------------------------ -->
<!-- Add 3.2  here -->
            <app-tree-map-chart></app-tree-map-chart>
          </div>
        </div>
      </div>
    </div>
    <div class="c1"><div class="col-md-12">
      <div class="card text-left"><div align="center">
          <div class="card-body">
            <h4 class="card-title border-bottom">Fiscal Quarter [Q1 2021]: January 1 - March 31</h4>
<!-- ------------------------------------------------------------------ -->
<!-- Add 3.3  here -->
            <app-bar-chart></app-bar-chart>
          </div>
        </div>
      </div>
    </div>
      <div class="c1"><div class="col-md-12">
         <div class="card text-left">
          <div class="card-body">
<!-- ------------------------------------------------------------------ -->
<!-- Add 3.4  here -->
            <app-data-tables></app-data-tables>
        </div>
      </div>
    </div>
  </div>
</div>
```

## SECTION B - QUESTION 4 ( 4 MARKS )

Only upload the **tree-map-chart.component.ts** file after completing this question.

Add the required data assignment(s), object assignment(s), selection event assignments(s) as well as the required label formatting assignment(s) to complete the data being accessed to the tree-map options in the tree-map-chart.component.ts file. Complete the constructor, selection event, and label formatting to functionally complete the TreeMapComponent so that it displays correctly on tree-map-chart.component.html AND app.component.html.

Please note that the Tree Map Chart options (tree-map-chart.component.html) have been disabled (edited out) to avoid loading errors. To view and test the final results of the added assignments required in this question the Tree Map options (tree-map-chart.component.html) will have to be enabled.

### Final Output (Section B – Question 4)


| Product | Monthly Sales | |
|---|---|---|
| Hamburgers | **Month** | **Units (Volume)** |
| | January | 567 |
| | February | 554 |
| | March | 789 |
| Fried Foodstuffs | **Month** | **Units (Volume)** |
| | January | 789 |
| | February | 456 |
| | March | 221 |
| Sandwhiches | **Month** | **Units (Volume)** |
| | January | 345 |
| | February | 123 |
| | March | 234 |
| Fizzy Drinks | **Month** | **Units (Volume)** |
| | January | 456 |
| | February | 757 |
| | March | 345 |
| Hot Drinks | **Month** | **Units (Volume)** |
| | January | 312 |
| | February | 456 |
| | March | 678 |
| Desserts | **Month** | **Units (Volume)** |
| | January | 345 |
| | February | 321 |
| | March | 112 |

## SOLUTION SECTION B - QUESTION 4

```
// Add 4.1 here (1 mark)
  // ---------------------------------------------------------------

  productSingleDataOption: any[] = []; // can add undefined with "(!)" OR "| undefined" OR "| any"
                                       // productSingleDataOption: any[]! = [];
                                       // productSingleDataOption: any[] = [] | any;
                                       // productSingleDataOption: any[] = [] | undefined;
                                       // view: any;
                                       // This is version dependant.

  //   Additional option / variations that could have been added or considered (version dependant):
  //   view: any[] = [];
  //   public productSingleDataOption: [] | any = productSingleDataOption
  // ---------------------------------------------------------------
  // DO NOT EDIT BETWEEN THESE SECTION LINES -----------------------
  gradient: boolean = false;
  animations: boolean = true;
  colorScheme = {
    domain: ['aqua', 'blue', 'chartreuse', 'crimson', 'fuchsia', 'gray']
  };
  // ---------------------------------------------------------------
  // DO NOT EDIT BETWEEN THESE SECTION LINES -----------------------


  // Add 4.2 here (1 mark)
  // ---------------------------------------------------------------

  constructor() {
    Object.assign(this, { productSingleDataOption });
  }


  // Add 4.3 here (1 mark)
  // ---------------------------------------------------------------

  onSelect(event: any) {
    console.log(event);
  }


  // Add 4.4 here (1 mark)
  // ---------------------------------------------------------------

  labelFormatting(c: { label: any; }) {
    return `${(c.label)}`;
  }
```

| SECTION B TOTAL | 10 |
|---|---|

# SECTION C – PYTHON DJANGO (10)

**BACKGROUND TO SECTION**

In this section a student will be required to complete complete one question; a **Python Django** question.

Students are provided with a file named **INF354ExamSectionC_studentCopy.zip** which contains all the files necessary to complete the questions stipulated in this section. This section contains 1 main question.

After completing each question students will be required to upload only the associated file to the specific question's upload area.

## PROJECT PREPARATION

- Download **INF354ExamSectionC_studentCopy.zip** file.
- Copy INF354ExamSectionC_studentCopy.zip to the root directory (for example, your C: or D: drive) of your computer and then unzip the file.
- Make sure you have Python 3.9 installed on your computer. Check the version of python with the **'python --version**' command in the Visual Studio Code terminal window.
- The installation is available from https://www.python.org/downloads/ as demonstrated in the Python 1 lecture.
- Next, make sure you have '**virtualenv**' installed on your computer. Type '**pip install virtualenv**' in your Visual Studio (VS) Code terminal window and press enter.
- Next, in VS Code click '**File**' followed by **'Open Folder**' and select the unzipped folder which contains all the python code. The folder opened within VS Code includes folders and the '**db.sqlite3**', '**manage.py**', and '**requirements.txt**' files.
- Next, within the VS Code terminal window, type the following command and press enter: '**python -m venv venv**'
- Next, within the VS Code terminal window, type the following command and press enter: '**venv/Scripts/activate**'. You should now be within the '**venv**' python virtual environment as demonstrated in the Python 1 lecture.
- Next, within the VS Code terminal window, type the following command and press enter: '**pip install -r requirements.txt**'
- Once the packages are installed you can continue completing this question.

Once you have completed the question in this section, upload the **views.py** file **only** to the Section C upload slot.

## GENERAL TASKS AND REQUIREMENTS

Your task is to complete the codebase to get this application to function as described below. The steps required to achieve this have been broken down as follows:

- Read the instructions carefully.
- Look for comments (hints) in the provided sample solution. The comments will guide you in terms of where the code modifications are required.
- Apply the necessary code changes to the specified file in the required application.
- Only upload the modified file associated with the question to ClickUP.

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

## QUESTIONS FOR SECTION C

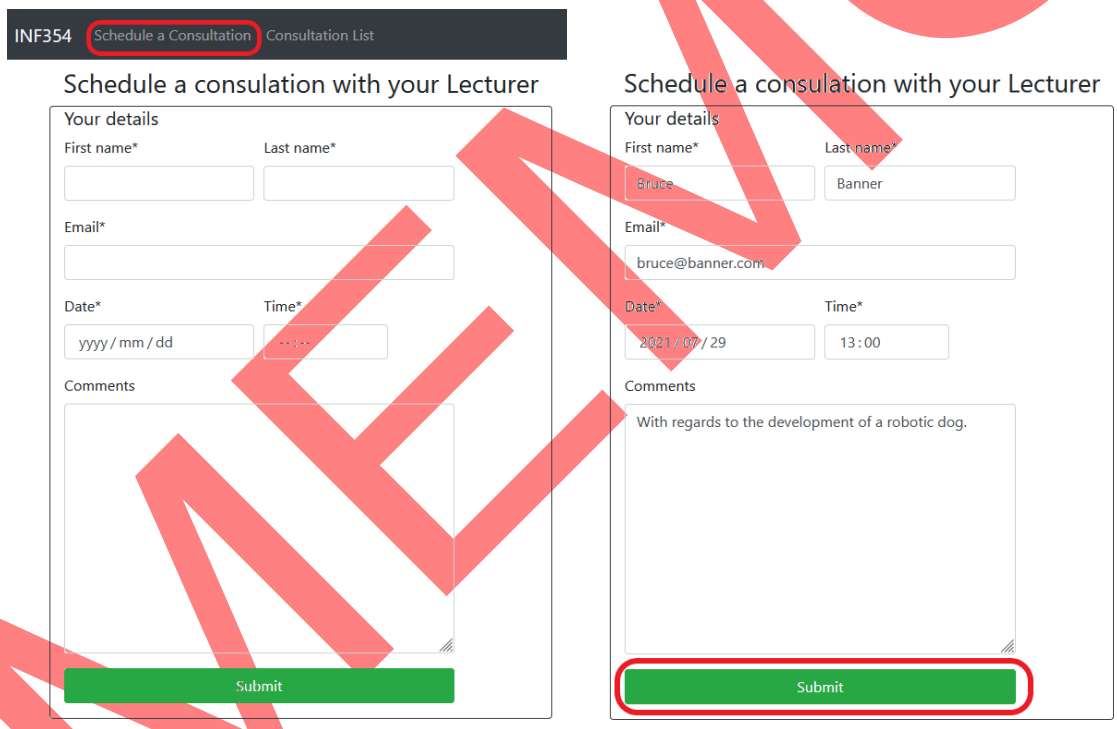
Please complete the following questions. Please be careful to save regularly and to upload the correct file.

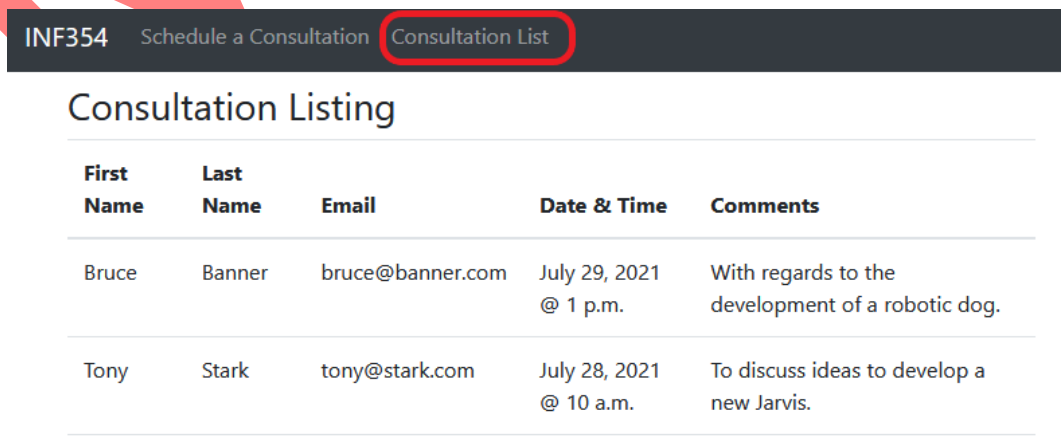
**SECTION C - QUESTION 1 (Python Django) (10 MARKS)**

Only upload the **views.py** file after completing this question.

- In the '**consultapp**' folder you need to add the code for the **'views.py**' file to schedule a consultation and display the consultation listing.
- In other words, you must create a method / function that '**GET**' and '**POST**' the '**ScheduleConsultForm**' through the '**consult_create.html**' page.
- Further, you must create a method / function to display the successfully saved consultations through the **'consult_list.html**' page.
- After the '**POST**' of a new consultation has been successfully saved to the database, the user must be redirected to the '**Consultation Listing**' page as demonstrated in the images below.
- **NB:** All the other features and functionality have already been created for you. For example, the settings, routing, web pages, database, form, and model have already been created. You just need to update the **'views.py**' file to create the consultation and display the results, as mentioned above.
- Additional '**Hints**' have been added as comments in the **'views.py**' file.

**Q1 Expected Output (SECTION C)**



## SOLUTION SECTION C

```
from django.shortcuts import render, redirect

from .forms import ScheduleConsultForm
from .models import ScheduleConsult

# Create your views here.


def index(request):
    if request.method == 'POST':
        form = ScheduleConsultForm(data=request.POST)

        if form.is_valid():
            form.save()
            return redirect('consult_list')

    else:
        form = ScheduleConsultForm()

    return render(request, 'consult_create.html', {
        'form': form
    })


def consult_list(request):
    consultation_list = ScheduleConsult.objects.all()
    return render(request, 'consult_list.html', {
        'consultation_list': consultation_list
    })
```

**SECTION C TOTAL** **10**