INF 354

Assignment 3

Out of 50 Marks

DUE: 06 June 2022

IMPORTANT NOTES:

- This is an individual assignment.
- Homework assignments are based on assessment objectives. If an objective has been achieved a mark will be allocated.
- **All assignments are submitted via ClickUP, see the Assignments section**.
- **Please do not upload the assignment3Data.sql file with your assignment. In other words, you only upload your Angular app (.zip), API (.zip) and video demo (.mp4). We will use our copy of the assignment3Data.sql, when required. See the Database and API installation and configuration section.**
- **Please execute the assignment3Data.sql file before building your angular application. See the Database and API installation and configuration section.**
- **If you are caught plagiarising, we will give you zero percent (0%) and you will be reported for plagiarism immediately.** We will audit historical assignments throughout the semester. **We trust that you understand the importance of this point.**

VIDEO INSTRUCTIONS:

- Make sure that everything is running when you start recording the video. The video should not be longer than 15 minutes showing the items in the **Standard Requirements** against the **Rubric**.
- When showing something from the **Standard Requirements**, show us as much detail as required. ***See the Rubric for the assessment criteria***. For example, when assessing the "***Program Functionality***" you must show the validation working per page, page redirects, the email being sent with OTP generated, the data is saved to the database, the password is hashed, and the pages are working as expected. Similarly, for the "***Program Output,***" the correct notification messages are being displayed per page, the email displays the OTP message, the product's dashboard displays the correct data in the correct format, and all the pages are demonstrated. Further for the "***Code readability***" we expect you to show us your code and display the organization of the code, and descriptive names (*i.e. all the code used to create the program, **not the configuration files like package.json**, etc.*). **The same applies to the rest of the Rubric, see below.**
- If something did not work in your code, in the video explain to us what you wanted to do and what you wanted to achieve with your approach. **This is to assess you correctly according to the Rubric**.
- **See the "Video Recording and Compression, and Assignment Upload Guide" in the Assignments section on ClickUP for video recording, compression and upload assistance.**

SUBMISSION INSTRUCTIONS:

- In this assignment, you will be given the requirements that you need to implement.
- **Source Code:** Zip your source code files together and for the API name it **uXXXXXXXX_HW03_API.zip**, where the XXXXXXXX is your student number, e.g. u12345678_HW03_API.zip. Further, for the Angular app name it **uXXXXXXXX_HW03_Angular.zip**, where the XXXXXXXX is your student number, e.g. u12345678_HW03_Angular.zip.
- **Video Demo:** Do not zip your **video demo**. In other words, submit the actual "**.mp4**" file. Name the video demo **uXXXXXXXX_HW03.mp4**, where the XXXXXXXX is your student number, e.g. u12345678_HW03.mp4.
- If files are uploaded to the wrong upload area, we will not go and look for the upload. Uploads should be submitted correctly. Incorrect uploads will lead to a deduction for the missing upload. In other words, if either one or both of the code (.zip files) is not uploaded you lose 50%. Further, if no files are uploaded (neither the .zip and .mp4) you lose 100%.
- Please Note: **If you omit either the code (.zip) or the video (.mp4) submission you will automatically lose 50% of your assignment mark. Please take this seriously and plan accordingly to submit it on time.**
- ***Note: you upload the code (.zip files) and the video demo (.mp4 file) together in the same location in the Assignment 03 Submission section. See the ClickUP information in the Assignments section.***

- Please **do not** upload the "***node_modules"*** and "***.angular"*** folders for the Angular app. In other words, once you have completed your program and created your video, delete the "***node_modules"*** and "***.angular"*** folders. We as the **Lecturing Team** will reinstall the ***node_modules*** folder dependencies using the "***npm install***" terminal command, *where necessary*. **This is so that you do not take long to upload your code with the video demo.**
- The API does not need any files to be removed, before zipping it. I.e. just zip the API application.

SUBMISSION DEADLINE: 06 June 2022

- There shall be no extensions to the aforementioned deadline.
- If homework submissions are uploaded too late then upload errors **will** happen.
- Do not wait until the last minute to complete the assignment.
- Start working on the assignment as soon as possible.
- E-mail submissions **will not** be accepted.
- Late submissions **will not** be accepted.
- **No exceptions will be made for anyone.**

USE CASE:

- A marketing and sales company, **Top Sales**, requested your software company to build the first iteration of the reporting dashboard.
- You are requested to develop the back-end using a **.Net core 5 API** and the front-end using **Angular**.
- For the application, you need to build the capability for new users to register, log in to the dashboard (incorporate email and a **one-time pin** (OTP)), and view the product's dashboard.
- When the application is launched, the landing page must be the **login page**, and navigation to all other pages must be done via angular routing, subject to the restrictions that will be detailed under "***Standard Requirements***".

STANDARD REQUIREMENTS:

- Login page:
  - The login page requires a Username (**a valid email address**) and a Password to proceed with logging in. If the username or password is not provided the logging in must be prevented (Fig. 1).
  - If the username or password is invalid (**does not match a user in the database**) the following notification message must be displayed "**Invalid user credentials.**" (Fig. 2).
  - When the user clicks on the link to register by "***Don't have an account? Register here***" they must be redirected to the Register page (*see the Register page section*).
  - When the user entered valid user credentials, an email with a 4-digit OTP must be sent to the user's email address (Fig. 3) and they must be redirected to the OTP page with the following notification message ("**The OTP has been sent to your email address.**") (Fig. 4)

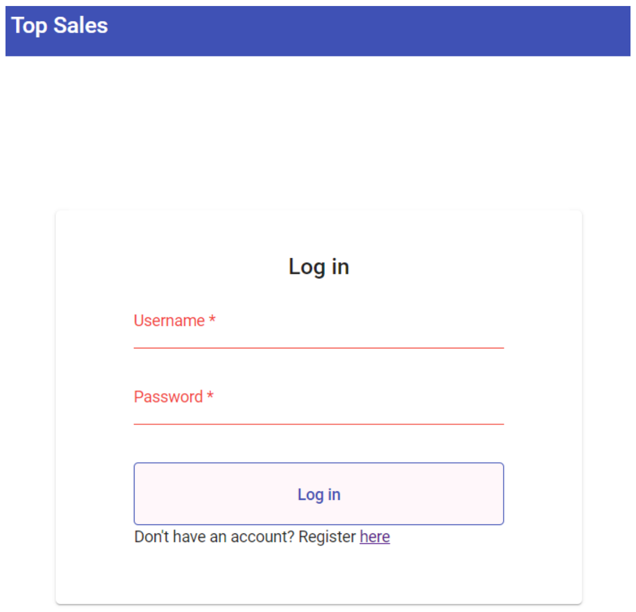
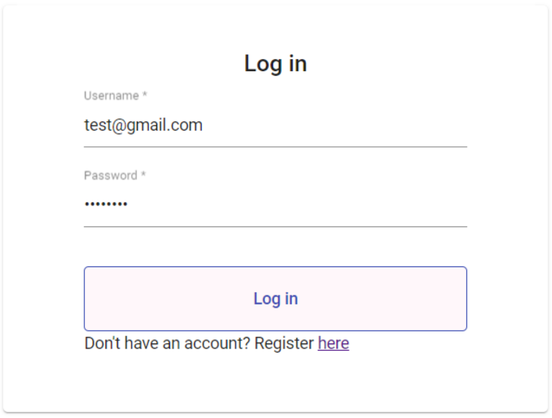


Fig. 1



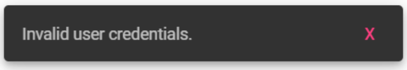
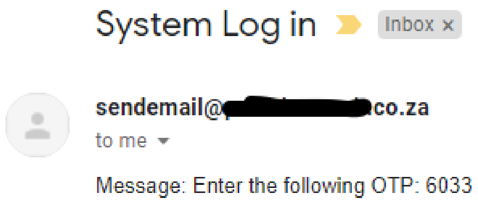

Invalid user credentials. X

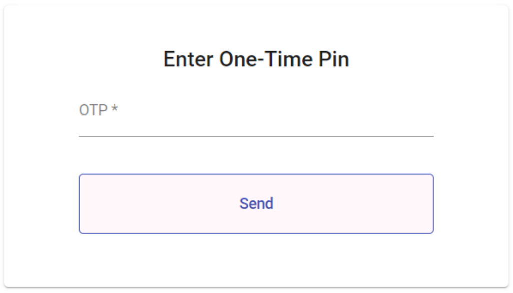
Fig. 2


System Log in
Inbox ×

sendemail@[redacted]co.za
to me

Message: Enter the following OTP: 6033

Fig. 3



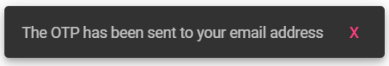

The OTP has been sent to your email address X

Fig. 4

- Register page:
    - The register page requires a **valid email address** and a **valid password** (**between 6 to 16 characters is allowed**) to proceed with registering. If the email address or password is not provided the registration must be prevented (Fig. 5).
    - If the user account already exists (**matches a user in the database**), the following notification message must be displayed "**User account already exists.**" (Fig. 6)
    - When the user is successfully registered, they must be redirected to the Login page with the following notification message "**Registered successfully.**". (Fig. 7)
    - Note: The password stored in the database must be hashed.

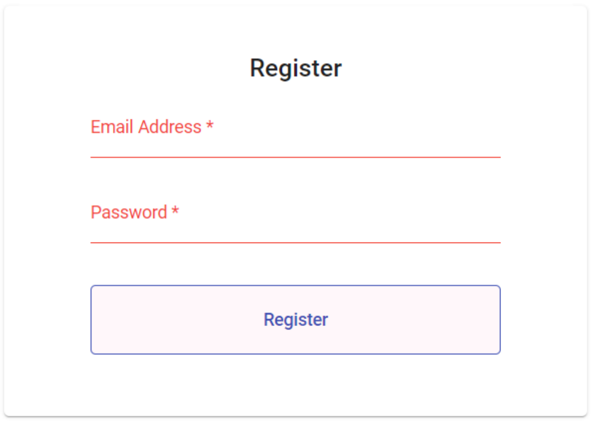
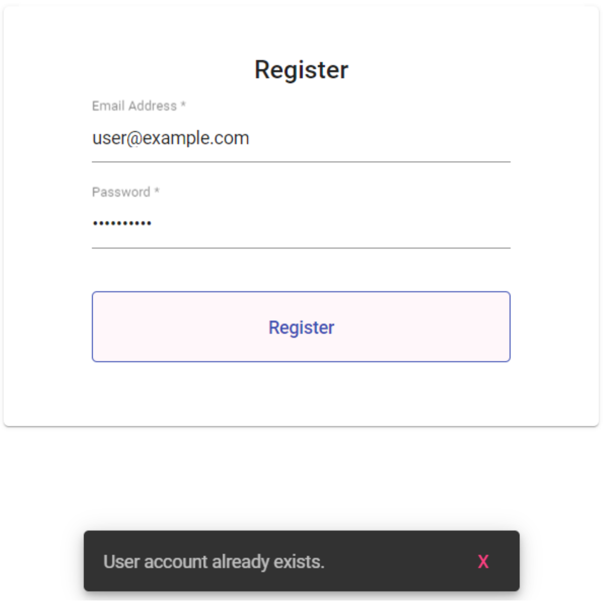


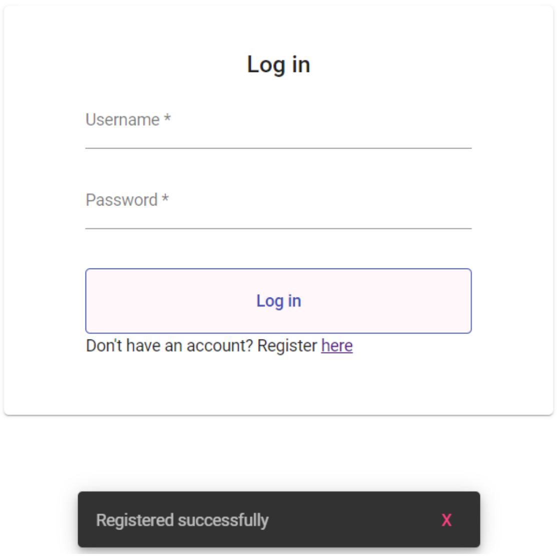
Fig. 5



user@example.com

User account already exists.

Fig. 6

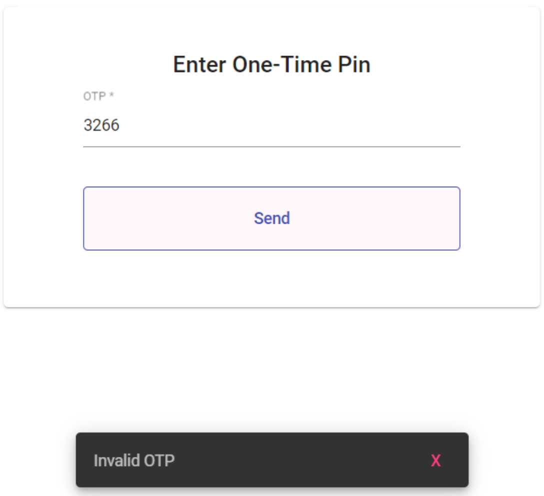


Don't have an account? Register here

Registered successfully

Fig. 7

- OTP page:
  - Once the user entered their valid credentials on the Login page, enter the 4-digit OTP sent to the email address (*see the Login page section*).
  - The OTP is required to be able to continue.
  - If the user enters an invalid OTP (**which does not match the OTP in the email and stored on the server**), the following notification message must be displayed "**Invalid OTP.**" (Fig. 8).
  - When the user enters a **valid OTP**, they must be redirected to the **ProductDashboard page** (*see the ProductDashboard page section*).

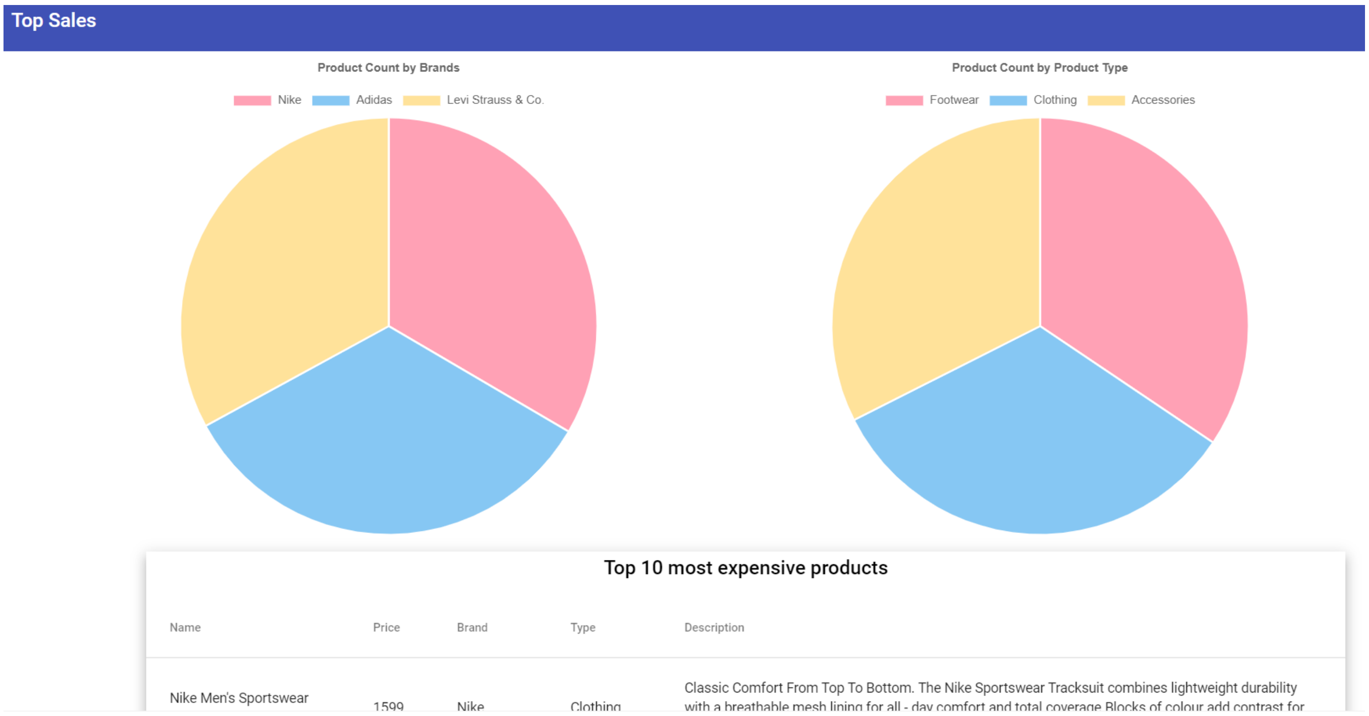

Enter One-Time Pin

OTP *

3266

Fig. 8

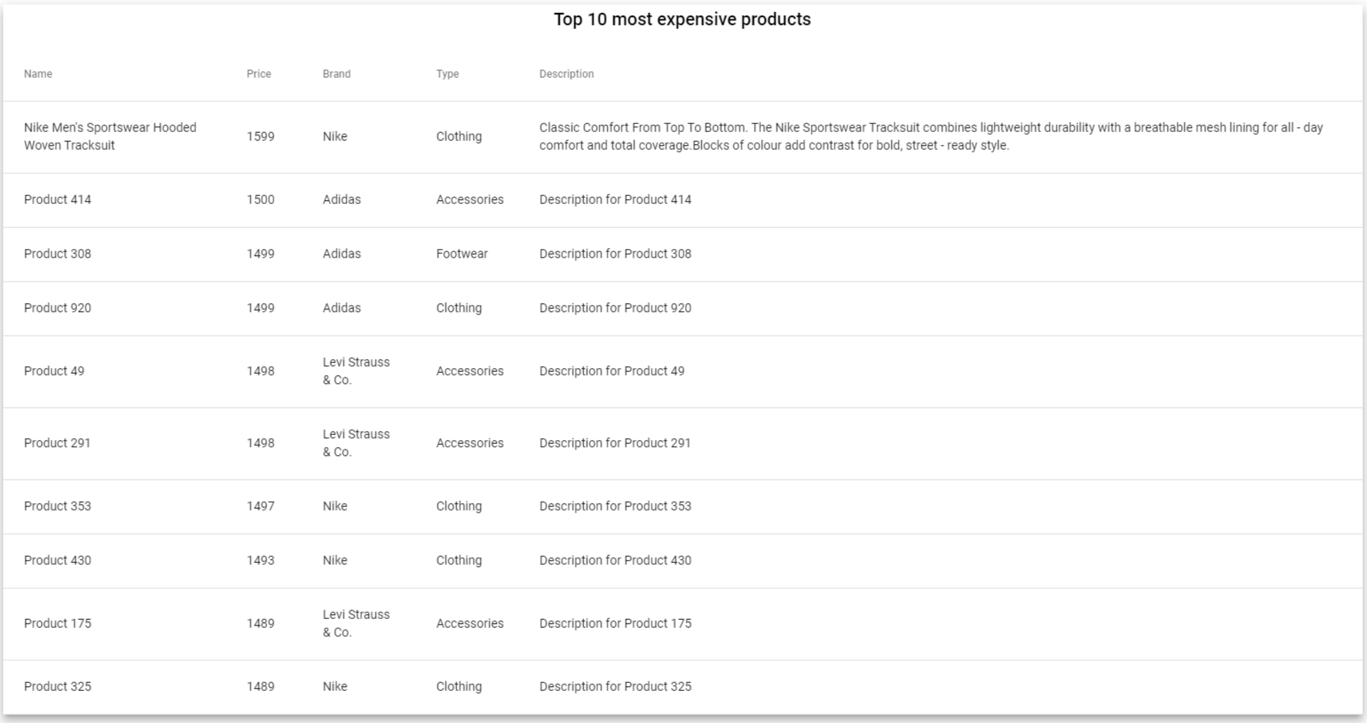
- ProductDashboard page:
  - The product dashboard must display 2 pie charts. One for the *Product count grouped by Brands*, and the other for the *Product count grouped by Product Type* (**Fig. 9**).
  - On the same page, you must display the "**Top 10 most expensive products**" based on the product price. The columns to display are the product name (**Name**), the product price (**Price**), product brand (**Brand**), product type (**Type**), and the product description (**Description**) (**Fig. 10**).
  - Note: Your top 10 products will not be the same as that displayed.



Top 10 most expensive products

Name	Price	Brand	Type	Description
Nike Men's Sportswear	1599	Nike	Clothing	Classic Comfort From Top To Bottom. The Nike Sportswear Tracksuit combines lightweight durability with a breathable mesh lining for all - day comfort and total coverage Blocks of colour add contrast for

Fig. 9



Top 10 most expensive products

Name	Price	Brand	Type	Description
Nike Men's Sportswear Hooded Woven Tracksuit	1599	Nike	Clothing	Classic Comfort From Top To Bottom. The Nike Sportswear Tracksuit combines lightweight durability with a breathable mesh lining for all - day comfort and total coverage.Blocks of colour add contrast for bold, street - ready style.
Product 414	1500	Adidas	Accessories	Description for Product 414
Product 308	1499	Adidas	Footwear	Description for Product 308
Product 920	1499	Adidas	Clothing	Description for Product 920
Product 49	1498	Levi Strauss & Co.	Accessories	Description for Product 49
Product 291	1498	Levi Strauss & Co.	Accessories	Description for Product 291
Product 353	1497	Nike	Clothing	Description for Product 353
Product 430	1493	Nike	Clothing	Description for Product 430
Product 175	1489	Levi Strauss & Co.	Accessories	Description for Product 175
Product 325	1489	Nike	Clothing	Description for Product 325

Fig. 10

DATABASE AND API INSTALLATION AND CONFIGURATION:

- API:
  - An "**API Template**" has been created with the default configuration. In other words, the Cors, Database Connection, and the **Brand**, **Product Type**, and **Product** entities and .Net Framework installations to get you started (**Assignment3_API.zip**).
  - Open the **Assignment03_API .Net Core application** in Visual Studio 2019. For example, by clicking on **Assignment03_API.sln**.
  - Once the application loads, open the "**appsettings.json**" file in the Solution Explorer.
  - Change and save the **Server** location, pointing to your *SQL Server Server Name* (*line 11*). Alternatively, you can just replace the server name with a period (.), see the example below.
  - Example:

```
optionsBuilder.UseSqlServer("Server=.;Database=Assignment3;Trusted_Connection=True;MultipleActiveResultSets=True");
```

  - Next, open the Package Manager Console (**View** > **Other Windows** > **Package Manager Console**) and run each of the following 2 commands individually to create the database tables from the entities mentioned above.
    - add-migration initial
    - update-database
  - The **Brands**, **Products**, and **ProductTypes** tables will be created in the **Assignment3** *MS SQL Server database*.
  - Next, *see the Database section below*, on how to populate the database with the "**assignment3Data.sql**" script.
  - Now, run the API and have it running when you are trying to connect your Angular app to it. In other words, both the API and the Angular app must be running for the application to be working correctly.

- Database:
  - For example, open MS SQL Server Management Studio, click on **File** > **Open** > **File** and locate the **assignment3Data.sql** file.
  - Once it is opened click the **Execute button** (*or F5*) to populate the **Brands**, **Products** and **ProductTypes** tables.
  - This data will be used to generate the charts and the table on the product's dashboard.

SUGGESTIONS AND HOMEWORK:

- Suggestions:
  - For the API, you will likely create 1 or 2 controllers (*the API controllers with endpoints (functions) to talk to the database and Angular App*).
    - For example, a **ProductController** with 1 endpoint (function) to **GET** the *ProductDashboard* data from the **Brands**, **ProductTypes**, and **Products** tables.
    - An **AuthenticationController** with 3 endpoints (functions) to **Register (POST)** a user, **Login (POST)** a *user and submit an email*, and **Verify (POST)** an *OTP*.
  - You can use any method to create a new user account in the database, however, the **password must be Hashed**. I.e. whatever method you use for password hashing is up to you, so long it works, and you can show this.
  - You can use *any method* to **send the OTP email**, so long it works, and you can show this. The email header and body can be anything, so long it displays the OTP.
  - You can use *any method* for the **notification messages**, i.e. it can use snackbar or toast, so long it displays correctly.
  - You can **design your UI** *any way you want*, **so long it has all the controls and output required as specified in the Standard Requirements**.
  - You can **develop your API** *any way you want*, **so long as it can perform the functionality required as specified in the Standard Requirements**.

- Homework:
  - For sending email in .Net, google **System.Net.Mail** namespace.
  - For creating charts, see ng2charts-example and google **ng2charts**.

RUBRIC:

Your assignment submission will be marked according to the following rubric:

Program (50 pts)	**(Exceptional)**	**(Very good)**	**(Good)**	**(Satisfactory)**	**(Poor)**	**(Very poor)**
Program Execution	The program executes correctly with no syntax or runtime errors. *I.e. the program has no execution issues.* (10)	The program executes with one or two syntax or runtime errors. *E.g. the program loads with no crashing but displays minor bugs in the debugger.* (8)	The program executes with a few syntax or runtime errors. *E.g. A couple of runtime errors and/or the program crashes at one screen/section.* (6)	The program executes with many syntax or runtime errors. *E.g. A few runtime errors and/or the program crashes at two screens/sections.* (5)	The program executes with major errors. *E.g. The program can execute, however, it is plagued with runtime or syntax errors, or the program keeps crashing during use.* (3)	The program does not execute. *I.e. The application fails to run.* (0)
Program Functionality	Program functionality is in line with the requirements. *I.e. the program has all the correct functionality implemented.* (10)	Program functionality has one minor inconsistency. *E.g. One of the functional requirements is incorrect.* (8)	Program functionality has a few minor inconsistencies. *E.g. Two of the functional requirements are incorrect or one is missing.* (6)	Program functionality has many inconsistencies. *E.g. Some of the functional requirements are incorrect or half is missing.* (5)	Program functionality has major inconsistencies. *E.g. Most of the functional requirements is incorrect or missing.* (3)	Program functionality is missing. *E.g. None of the functionality works or all the functionality is missing.* (0)
Program Output	The program displays correct output in line with the requirements. *I.e. It produces the same output as required.* (10)	The program has one or two very minor output discrepancies. *I.e. It produces output with barely noticeable inconsistencies. E.g. one or two formatting issues.* (8)	The program has a few output discrepancies. *I.e. It produces output with easily noticeable inconsistencies. E.g. The program does not return some of the data or there are a few formatting issues.* (6)	The program has many output discrepancies. *I.e. It produces output with many noticeable inconsistencies. E.g. The program does not return half of the data or there are plenty of formatting issues.* (5)	The program has major output discrepancies. *I.e. The output is plagued with inconsistencies. E.g. The program does not return most of the data or there are substantial formatting issues.* (3)	Output is incorrect. *E.g. The program does not provide any of the requested output or all the formatting is not as requested in the requirements.* (0)
Program Interface (UI)	The program interface is professionally done. I.e. *The interface is implemented correctly and looks very good.* (5)	The program interface is done well. *I.e. The interface is implemented correctly and looks good. E.g. One or two styling/layout issues.* (4)	N/A	The program interface is good enough. *I.e. The interface is implemented correctly and looks okay. E.g. A few styling/layout issues.* (3)	The program interface is poorly done. *I.e. The interface is mostly incorrect or looks poorly done. E.g. The layout is mostly incorrect or has plenty of styling issues.* (2)	The program interface is very poor. *I.e. The interface is entirely incorrect or is very poorly done. E.g. The layout is completely incorrect or the styling is missing.* (0)
Code Readability	The program code is well organized and makes good use of white space. Variables have	Program code is organized and makes use of white space. Variables have descriptive names.	N/A	Program code is mostly organized and makes use of white space. Most variables have descriptive	Program code is somewhat organized, and not easy to read and understand. *E.g. There are plenty of variable naming convention issues*	Program code is difficult to read. *E.g. Variable naming conventions are*

Program (50 pts)	**(Exceptional)**	**(Very good)**	**(Good)**	**(Satisfactory)**	**(Poor)**	**(Very poor)**
	descriptive names. I.e. *There is nothing to fault on.* (5)	*E.g. There are one or two variable naming convention issues or white space issues.* (4)		names. *E.g. There are a few variable naming convention issues or program code organization that could be improved.* (3)	*or the code is challenging to follow.* (2)	*missing or the code is hard to follow.* (0)
Video Demonstration	The program is exceptionally well presented. I.e. *The student demonstrated and displayed all the required functionality, output, interfaces, and code.* (10)	The program is well presented. *E.g. The student demonstrated and displayed all the required functionality, output, interfaces, and code. However, one of the descriptions or illustrations was lacking.* (8)	The program presentation is good. *E.g. The student demonstrated and displayed most of the required functionality, output, interfaces, and code. However, two of the functionality, output, interfaces and code descriptions or illustrations were lacking or missing.* (6)	The program presentation is adequate. *E.g. The student demonstrated and displayed most of the required functionality, output, interfaces, and code. However, a few to half of the functionality, output, interfaces and code descriptions or illustrations were lacking/missing.* (5)	The program is presented poorly. *E.g. The student demonstrated and displayed a few of the required functionality, output, interfaces, and code. However, most functionality, output, interfaces, and code descriptions or illustrations were lacking/missing.* (3)	The program has barely been presented or has been presented very poorly. *E.g. The student failed to demonstrate and display the required functionality, output, interfaces, and code or it was missing.* (0)