Homework 2 – Ionic with Angular and TypeScript

Out of 50 Marks

DUE: 15 May 2024

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**.
- **You only upload your Ionic (.zip), and video demo (.mp4).**
- **If you are caught for plagiarism, 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 all the validation, tab navigation, basket creation, and order generation functionality working. Similarly, for the "***Program Output***" the validation error message(s), the control update(s), and the product, basket item, and order listings are expected to be 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 name it **uXXXXXXXX_HW02.zip**, where the XXXXXXXX is your student number, e.g. u12345678_HW01.zip.
- **Video Demo:** **Do not** zip your **video demo**. In other words, submit the actual "**.mp4**" file. Name the video demo **uXXXXXXXX_HW01.mp4**, where the XXXXXXXX is your student number, e.g. u12345678_HW01.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 the code (.zip file) is not uploaded you lose **50%.** Further, if no files are uploaded (neither the .zip nor .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 file) and the video demo (.mp4 file) together in the same location in the Assignment 02: Ionic Submission section. See the ClickUP information in the Assignments section.***
- Please **do not** upload the "***node_modules"*** and "***.angular"*** folders. 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.**

SUBMISSION DEADLINE: 15 May 2024

- There shall be no extensions to the aforementioned deadline.
- If homework submissions are uploaded too late then upload errors **will** happen.

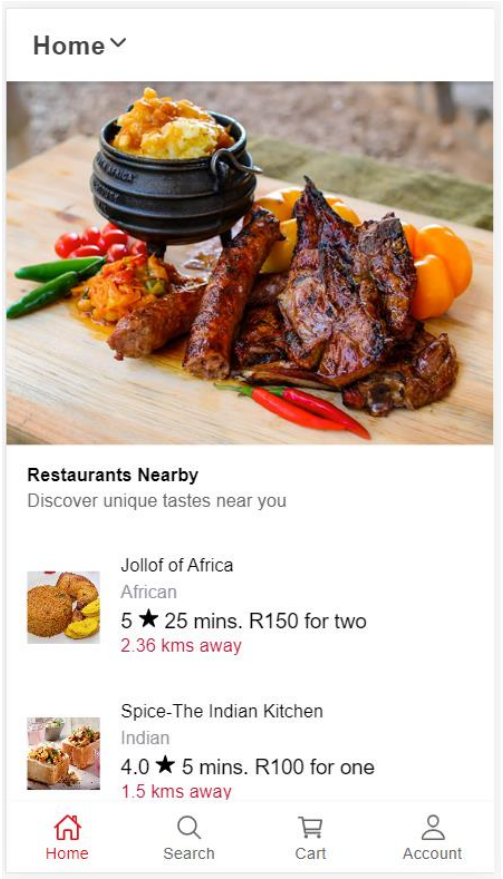
- Do not wait until the last minute to complete the assignment.
- Start working on the assignment as soon as possible.
- E-mail submissions **will not** be accepted.
- Late submissions **will not** be accepted.
- **No exceptions will be made for anyone.**

USE CASE:

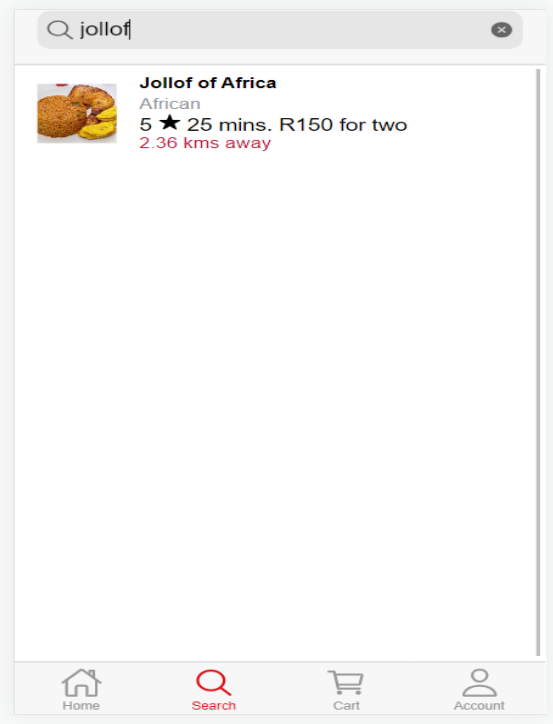
- An online delivery App called 354 Delivery has tasked you with redesigning their mobile application for use by their customers
- You will need to create a front end with four pages: A **Home page**, a **Search page**, a **Cart page,** and an **Account page**.
- You can use any four restaurants of your choice - You **only** need Ionic and Angular knowledge, and **no** back-end API coding is required.
- When the application is launched, the landing page must be the **Home page**, and navigation to the second, third, and fourth pages must be done via **tabs**, subject to the restrictions that will be detailed under "***Standard Requirements***".

STANDARD REQUIREMENTS:

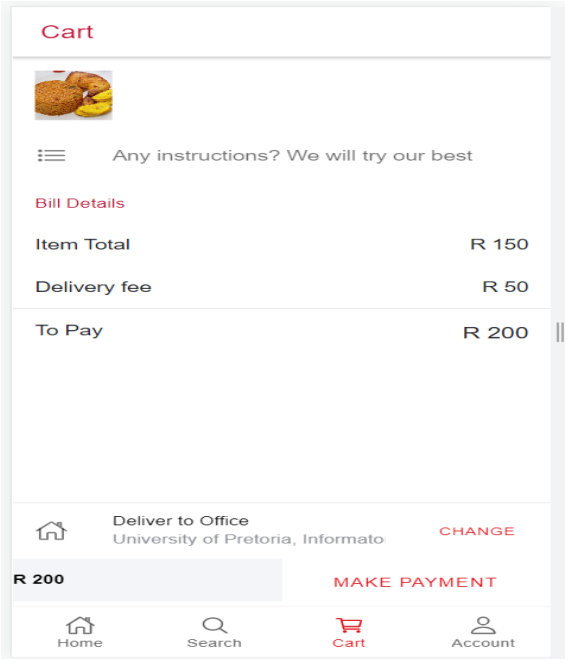
- **Home page**:
    - The list of all four restaurants (*including their name, type of dish, ratings, distance to home/school address, and price*) should be displayed on the page, along with the top dish of the restaurant. Note: Students must choose any four restaurants of their choice and one dish per restaurant
    - A user must be able to access all four tabs.
    - When you click on the restaurant on the home page it should automatically create "one" order for you in the Cart tab
    - Data manipulation should be done via local storage (see Angular II lecture source code for local storage implementation example).
    - Images can be stored in the app's assets folder (see the Angular II lecture source code example).



- **<u>Search page</u>**:
  - This page must allow a user to search for any of the restaurants from the home page along with the *name, type of dish, ratings, distance to home/school address, and price* provided for each restaurant.
  - When you click on the restaurant in the search tab it should automatically create “one” order for you in the Cart tab.

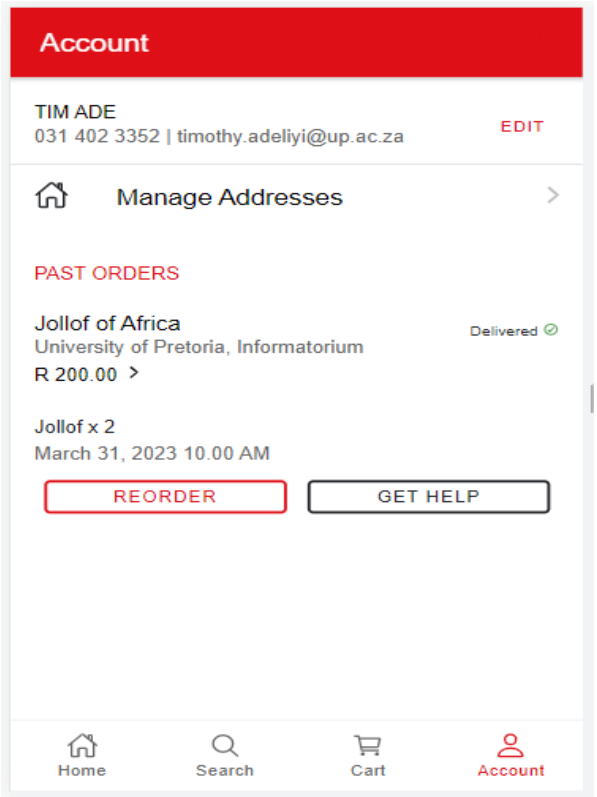


- **<u>Cart page</u>:**
  - This page must allow a user to view the order placed. The delivery price is fixed while the product price is automatically picked up from the homepage.
  - It is important to note that the total should be calculated based on any modifications to the order and should include delivery charges.
  - Users should be able to type instructions for delivery
  - When a user clicks on make payment, a modal pop-up should state that payment was successful
  - You are not expected to implement the “Deliver to Office” section. It is just for display purposes. In other words, you still need to add it to the front end but do not need to code its functionality.



- **Account page**:

  - The account page should display the customer's details and past-delivered orders. Note that users should be able to use add/edit their account details
  - The reorder button should take users back to the cart page with the previous order selected
  - When users click on “Get Help” there should be a modal pop-up with some text displayed
  - You are not expected to implement the “Manage Addresses” section. It is just for display purposes. In other words, you still need to add it to the front end but do not need to code its functionality.



SUGGESTIONS:

- 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 Ionic app any way you want, so long as it can perform the functionality required as specified in the Standard Requirements.

RUBRIC:

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

Program (50 pts)	(Exceptional)	(Very 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 few runtime errors and/or the program crashes at one screen/section.* (6)	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 couple of minor inconsistencies. *E.g. Two of the functional requirements are incorrect or one is missing.* (6)	Program functionality has major inconsistencies. *E.g. A significant number of the functional requirements are 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 the 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 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)	The 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)	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 descriptive names. I.e. *There is nothing to fault on.* (5)	The program code is organized and makes use of white space. Variables have descriptive names. *E.g. There are one or two variable naming convention issues or white space issues.* (4)	Program code is mostly organized and makes use of white space. Most variables have descriptive names. *E.g. There are a few variable naming convention issues or program code organization that could be improved.* (3)	The program code is somewhat organized, and not easy to read and understand. *E.g. There are plenty of variable naming convention issues or the code is challenging to follow.* (2)	Program code is difficult to read. *E.g. Variable naming conventions are missing or the code is hard to follow.* (0)

Program (50 pts)	**(Exceptional)**	**(Very good)**	**(Satisfactory)**	**(Poor)**	**(Very poor)**
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 adequate. *E.g. The student demonstrated and displayed most of the required functionality, output, interfaces, and code. However, two of the functionality, output, interfaces, or code descriptions or illustrations were lacking or one was missing.* (6)	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, or code descriptions or illustrations were lacking or two were 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 most were missing.* (0)