



Assignment 02

Out of 50 Marks

DUE: 25 April 2025

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 App (.zip) and video demo (.mp4).**
- **Refer to the L&P: 07 & 08 Ionic I and Ionic II Source Code shared with you previously for guidance on the concepts. See the code on ClickUP under Course Content.**
- **If you are caught plagiarising, we will give you zero per cent (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 respective components functionality is working. Similarly, for the “**Program Output**.” 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*). **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 to implement.
- **Source Code:** Zip your source code files together, and for the Ionic app name it **uXXXXXXXX_HW02_IONIC.zip**, where the XXXXXXXX is your student number, e.g., u12345678_HW02_IONIC.zip.
- **Video Demo:** **Do not** zip your **video demo**. In other words, submit the actual “**.mp4**” file. Name the video demo **uXXXXXXXX_HW02.mp4**, where the XXXXXXXX is your student number, e.g., u12345678_HW02.mp4.
- If files are uploaded to the wrong upload area, we will not look for the upload. Uploads should be submitted correctly.
- **Please Note:** If you omit the code (.zip) or the video (.mp4) submission, you will lose **50%** of your assignment mark. If no files are uploaded (neither the .zip nor .mp4), you lose **100%**. 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 02 Submission section. See the ClickUP information in the Assignments section (when readily available).**
- Please **do not** upload the “**node_modules**” or the **Ionic App**. In other words, once you have completed your program and created your video, delete the “**node_modules**”. As the Lecturing Team, we will reinstall the **node_modules** folder dependencies using the “**npm install**” terminal command, *where necessary*. **This is so you do not take long to upload your code with the video demo.**

SUBMISSION DEADLINE: 25 April 2025

- Submission is due at **11:59 AM**
- There shall be no extensions to the deadline above.
- 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:

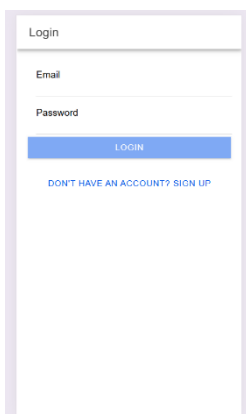
GetFit is a small health and wellness focused business that provides personalised fitness plans for its users. GetFit (client) has requested you to develop a mobile application using Ionic, Angular, and TypeScript that allows users to:

- Sign up and log in to their accounts.
- Browse a list of workout programs (e.g., Weight Loss, Muscle Gain, Cardio).
- View work-out details, including exercises, duration, and required equipment.
- Mark workouts as completed and track their progress.

The app should store user progress locally using Ionic Storage (optional), and users should be able to reset their progress if needed.

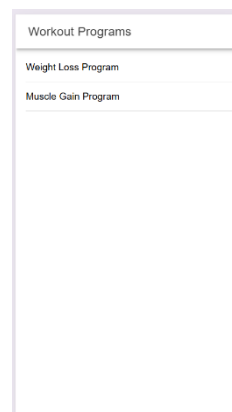
STANDARD REQUIREMENTS:

- ✚ **Project Setup:**
 - Create an **Ionic Angular** project.
 - Configure necessary dependencies such as **Ionic Storage**(optional) or **hardcode**.
- ✚ **User Authentication:**
 - Implement a simple login/signup form (authentication can be simulated using local storage) – user input (<https://ionicframework.com/docs/api/input>)
- ✚ **Workout Program Listing:**
 - Create a **Workout Service** in Angular to manage fitness programs.
 - Display a list of available workout programs using an **Ionic List component** (<https://ionicframework.com/docs/api/list>)
- ✚ **Workout Details Page:**
 - When a user selects a workout, navigate to a details page (suitable navigation)
 - Show a description, list of exercises, and an option to mark the workout as completed.
- ✚ **Tracking Progress:**
 - Store completed workouts using **Ionic Storage** (optional).
 - Display progress on a dedicated page.
 - Allow users to reset their progress.
- ✚ **UI & Navigation:**
 - Use Ionic UI components for a user-friendly interface.
 - Implement a **suitable navigation method** for switching between Home, Workouts, and Progress pages.



A mobile app screen mockup for user authentication. It features a title bar labeled 'Login'. Below the title bar are two input fields: 'Email' and 'Password'. A blue button labeled 'LOGIN' is positioned below the password field. At the bottom of the screen, there is a link that reads 'DON'T HAVE AN ACCOUNT? SIGN UP'.

FIG. 1: User Authentication



A mobile app screen mockup for listing workout programs. It has a title bar labeled 'Workout Programs'. Below the title bar, there is a list of two items: 'Weight Loss Program' and 'Muscle Gain Program'. The list is contained within a scrollable area, indicated by a vertical line on the right side.

FIG. 2: Workout Program Listing

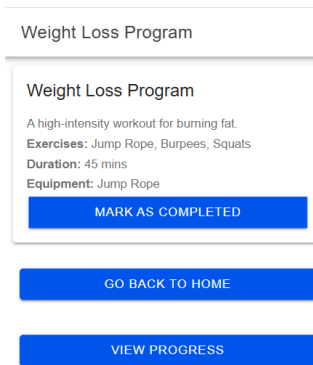


Fig. 3: Work-out Details

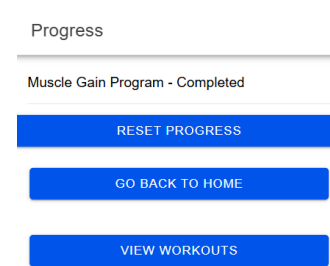


FIG. 4: Tracking Progress

[The images provided are a guideline related to the requirements – your application will differ]

IONIC INSTALLATION AND CONFIGURATION:

- IONIC
 - You can work off of the **Source Code** made available in the **IONIC I** and **IONIC II** lectures (L&P 07 and L&P 08) or go to <https://ionicframework.com/> for additional components and utilities
 - Refer to the L&P 07 for the installation guidelines and version checks.

SUGGESTIONS:

- You can **design your UI** *any way you want*, as long as it has **all the necessary components and output required as specified in the Standard Requirements** (include relevant SCSS with images <assets> for all pages).
- You can **develop your IONIC APP** *any way you want*, as 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)	(Good)	(Satisfactory)	(Poor)	(Very poor)
Program Execution	The program executes correctly with no syntax or runtime errors. <i>I.e. the program has no execution issues.</i> (10)	The program executes with one or two syntax or runtime errors. <i>E.g. the program loads with no crashing but displays minor bugs in the debugger.</i> (8)	The program executes with a few syntax or runtime errors. <i>E.g. A couple of runtime errors and/or the program crashes at one screen/section.</i> (6)	The program executes with many syntax or runtime errors. <i>E.g. A few runtime errors and/or the program crashes at two screens/sections.</i> (5)	The program executes with major errors. <i>E.g. The program can execute, however, it is plagued with the runtime or syntax errors, or the program keeps crashing during use.</i> (3)	The program does not execute. <i>I.e. The application fails to run.</i> (0)
Program Functionality	Program functionality is in line with the requirements. <i>I.e. the program has all the correct functionality implemented.</i> (10)	Program functionality has one minor inconsistency. <i>E.g. One of the functional requirements is incorrect.</i> (8)	Program functionality has a few minor inconsistencies. <i>E.g. Two of the functional requirements are incorrect or one is missing.</i> (6)	Program functionality has many inconsistencies. <i>E.g. Some of the functional requirements are incorrect or half is missing.</i> (5)	Program functionality has major inconsistencies. <i>E.g. Most of the functional requirements are incorrect or missing.</i> (3)	Program functionality is missing. <i>E.g. None of the functionality works or all the functionality is missing.</i> (0)
Program Output	The program displays the correct output in line with the requirements. <i>I.e. It produces the same output as required.</i> (10)	The program has one or two very minor output discrepancies. <i>I.e. It produces output with barely noticeable inconsistencies. E.g. one or two formatting issues.</i> (8)	The program has a few output discrepancies. <i>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.</i> (6)	The program has many output discrepancies. <i>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.</i> (5)	The program has major output discrepancies. <i>I.e. The output is plagued with inconsistencies. E.g. The program does not return the requested output or all the requested output is not as requested in the requirements.</i> (3)	Output is incorrect. <i>E.g. The program does not provide any of the requested output or all the requested output is not as requested in the requirements.</i> (0)
Program Interface (UI) – SCSS elements	The program interface is professionally done. <i>I.e. The interface is implemented correctly and looks very good with styling and images.</i> (5)	The program interface is done well. <i>I.e. The interface is implemented correctly and looks good. E.g. One or two styling/layout issues.</i> (4)	N/A	The program interface is good enough. <i>I.e. The interface is implemented correctly and looks okay. E.g. A few styling/layout issues.</i> (3)	The program interface is poorly done. <i>I.e. The interface is mostly incorrect or looks poorly done. E.g. The layout is mostly incorrect or has plenty of styling issues.</i> (2)	The program interface is very poor. <i>I.e. The interface is entirely incorrect or is very poorly done. E.g. The layout is completely incorrect or the styling is missing.</i> (0)
Code Readability	The program code is well organized and makes good use of white space.	The program code is organized and makes use of white space.	N/A	Program code is mostly organized and makes use of white space. Most variables	Program code is somewhat organized, and not easy to read and understand. <i>E.g. There are plenty of variable</i>	Program code is difficult to read. <i>E.g. Variable naming conventions are</i>

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