Faculty of Engineering, Built Environment and Information Technology
Fakulteit Ingenieurswese, Bou-omgewing en Inligtingtegnologie / Lefapha la Boetšenere, Tikologo ya Kago le Theknolotši ya Tshedimošo

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

DEPARTMENT OF INFORMATICS

INF 354

SEMESTER TEST (1) **DATE: 2022-05-11**

Examiners	: Mr Ridewaan Hanslo : Mr Zola Mahlaza : Ms Demi Murray	**Time**	:	180 min
Moderator / External Examiner	: Prof Abejide Ade-Ibijola University of the Johannesburg	**Marks**	:	53

Student Number								Surname	Initials

Question Section	Module outcomes (as in Study Guide)							Marks allocated	Maximum mark
	MO1	MO2	MO3	MO4	MO5	MO6	MO7		
Section A	X	X							**18**
Section B	X								**10**
Section C		X							**25**
Total									**53**

Instructions

1. This paper consists of 3 sections with one question per section (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 file(s) in one of the three programs that you need to update or finalise.
4. Each sub-sets of instructions relate to activities and tasks in one of the files.
5. Answer all the questions – there are no optional questions.
6. Please read all questions, instructions and sub-sets of tasks very carefully.
7. After completing work on a relevant question, please upload the file(s) required to be uploaded to the correct upload area. In other words, each section will mention what files to upload and how to upload it.

The University of Pretoria commits itself to producing 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 – TYPESCRIPT (18)

For Section A, you need to complete one question; a TypeScript question. The question does not require access to an existing database file to be completed. In other words, repositories are generated through the application. All the source files are in the following zipped file:

- **Semester_Test_Section_A_Question.zip**

Once you have completed the 1 question in this section, **upload 2 files**, the **hero.ts** file and the **ihero.ts** file to the **Section A upload slot**.

Your task is to complete the codebase to get the application to function as described below. For the question, please do the following:
- Read the instructions carefully.
- Apply the necessary code changes to the specified file(s) in the required application.
- Only upload the modified file(s) associated with the question to ClickUP.

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

SECTION A - QUESTION 1 (TypeScript) (18 MARKS)

Only upload the **hero.ts** and **ihero.ts** files after completing this question. **Two files in total to upload**.

This question requires you to complete the missing TypeScript code for the **hero.ts** and **ihero.ts** files in the shared folder of the angular application. *Note: the application will be broken if you try to run it. In other words, when you have successfully added the code in the **hero.ts** and **ihero.ts** files, the application will begin to display as seen in the **Expected Output***. Before you begin, you need to become familiar with the following **JSON** template (also found in the heroes.component.ts file):

```
[{
  "id": 1,
  "name": "Tony Stark (Iron Man)",
  "age": 53,
  "birthday": "May 29",
  "height": "185cm",
  "image": "assets/images/tony-stark-iron-man.webp",
  "alive": false,
},
{
  "id": 2,
  "name": "Steve Rogers (Captain America)",
  "age": 34,
  "birthday": "July 4",
  "height": "185cm",
  "image": "assets/images/steve-rogers.webp",
  "alive": false,
},
{
  "id": 3,
  "name": "Bruce Banner (The Hulk)",
  "age": 54,
  "birthday": "December 18",
  "height": "250cm",
  "image": "assets/images/The-Incredible-Hulk.webp",
  "alive": true,
},
{
  "id": 4,
  "name": "Thor",
  "age": 1059,
  "birthday": null,
  "height": "192cm",
  "image": "assets/images/thor-lightning.webp",
  "alive": true,
},
{
  "id": 5,
  "name": "Natasha Romanoff (Black Widow)",
  "age": 39,
  "birthday": "December 3",
  "height": "164cm",
  "image": "assets/images/black-widow-1.webp",
  "alive": true,
},
{
  "id": 6,
  "name": "Peter Parker (Spider-Man)",
  "age": 19,
  "birthday": "August 10",
  "height": "170cm",
  "image": "assets/images/peter-parker-Cropped.webp",
  "alive": true,
},
{
  "id": 7,
  "name": "Clint Barton (Hawkeye)",
  "age": 53,
  "birthday": "June 18",
  "height": "173cm",
  "image": "assets/images/hawkeye.webp",
  "alive": true,
},
{
  "id": 8,
  "name": "Colonel James 'Rhodey' Rhodes (War Machine)",
  "age": 55,
  "birthday": "October 6",
  "height": "173cm",
  "image": "assets/images/Don-cheadle-as-rhodey-Cropped.webp",
  "alive": true,
},
{
  "id": 9,
  "name": "Samuel Thomas 'Sam' Wilson (Falcon/Captain America)",
  "age": 40,
  "birthday": "September 23",
  "height": "178cm",
  "image": "assets/images/Anthony-Mackie-Captain-America-4.webp",
  "alive": true,
},
{
  "id": 10,
  "name": "Wanda Maximoff (Scarlet Witch)",
  "age": 30,
  "birthday": "February 10",
  "height": "168cm",
  "image": "assets/images/Wanda-Scarlet-Witch-Cropped.webp",
  "alive": true,
},
{
  "id": 11,
  "name": "Vision",
  "age": 3,
  "birthday": "May 29",
  "height": "May",
  "image": "assets/images/Vision-Civil-War-Cropped.webp",
  "alive": true,
},
{
  "id": 12,
  "name": "Scott Lang (Ant-Man)",
  "age": null,
  "birthday": null,
  "height": "178cm",
  "image": "assets/images/antman-and-the-wasp-marvel-4.webp",
  "alive": true,
}
]
```

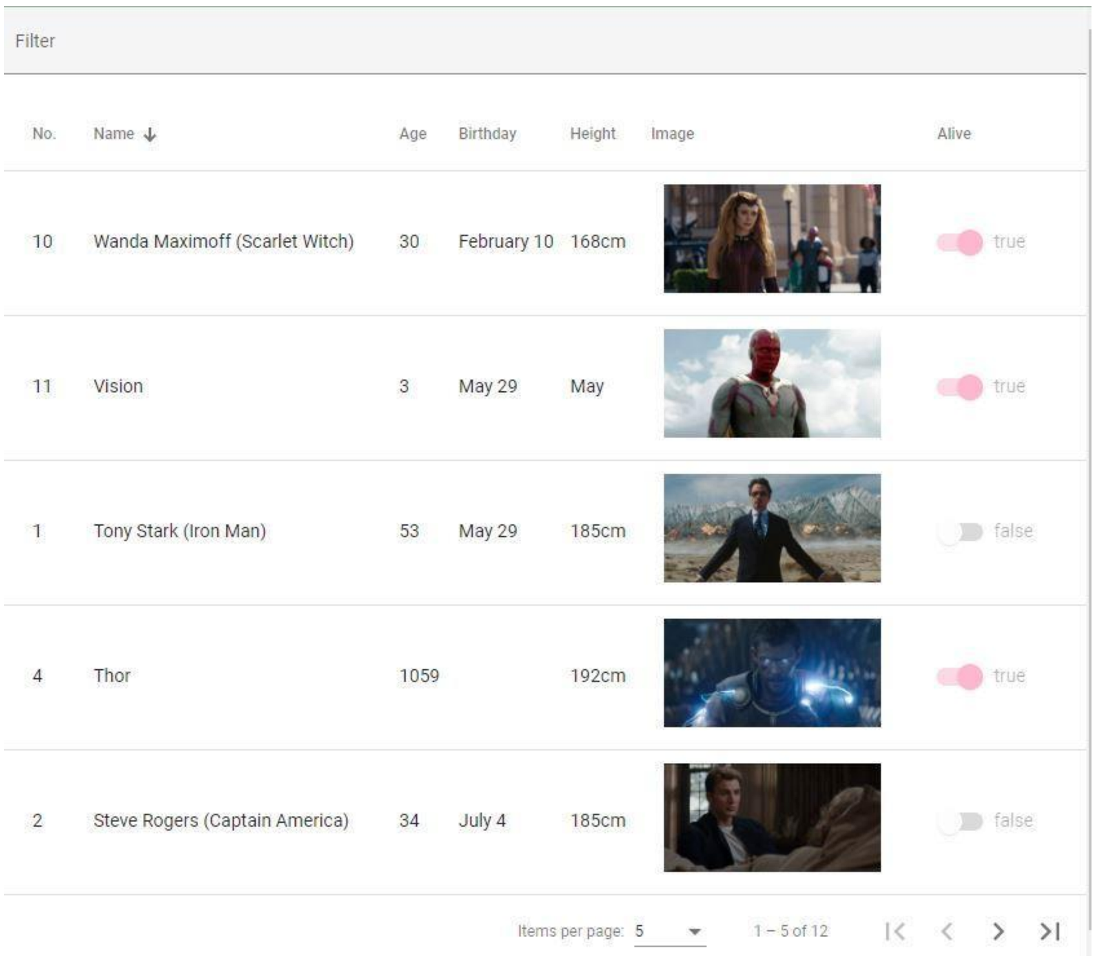
- For this question, in the “**ihero.ts**” you need to do the following **[5 Marks]**
  - Create an **Ihero interface** that can be *used/implemented* by the “**hero.ts**” file (2 marks).
  - The “**Ihero interface**” must only have the “**id**”, “**name**”, and “**image**” members in it (3 marks).
  - *Note: See the JSON template (above) for the correct data types to use.*

- Furthermore, in the “**hero.ts**” file you need to do the following **[13 Marks]**
  - Create a **Hero class** that can *use/implement* the **Ihero interface**. The “**Hero class**” must include the interface members and the remaining members. In other words, those not *used/implemented* in the interface. The initialization of the class members can be either through the "definite assignment assertion", a “constructor”, or “property setting”. *Note: You will not get extra marks for using a constructor. Furthermore, you cannot change the code in another file to avoid doing the initialization of the members* (11 marks).
  - In addition, in the “**hero.ts**” file you must create a TypeScript “**type**” datatype naming it “**AllowNull**”. AllowNull must cater for nulls as well as any member value “**data type**” that has **nulls** as displayed in the JSON template above. In other words, if the member usually holds a number where a null is currently showing then you must include that number data type as part of the newly created “**type**” (*HINT: a newly created type that can store multiple types of values. Think union type*). Once the “AllowNull” “type” is created, add this new type to the members that do have nulls as values as displayed in the JSON template (2 marks).

- Once you are done and have successfully implemented the required code, you can run the application and see the **Expected Output** displayed below.

- **NB: Do not forget to run “npm install” to allow the application packages to be re-installed.**
- **NB: The application was created using Visual Studio Code and Angular. You have already been given a document for any environment pre-setup installations required and how to do it, shared on the Course Module ClickUP site. Therefore, for this question, no additional environment pre-setup installations are needed.**

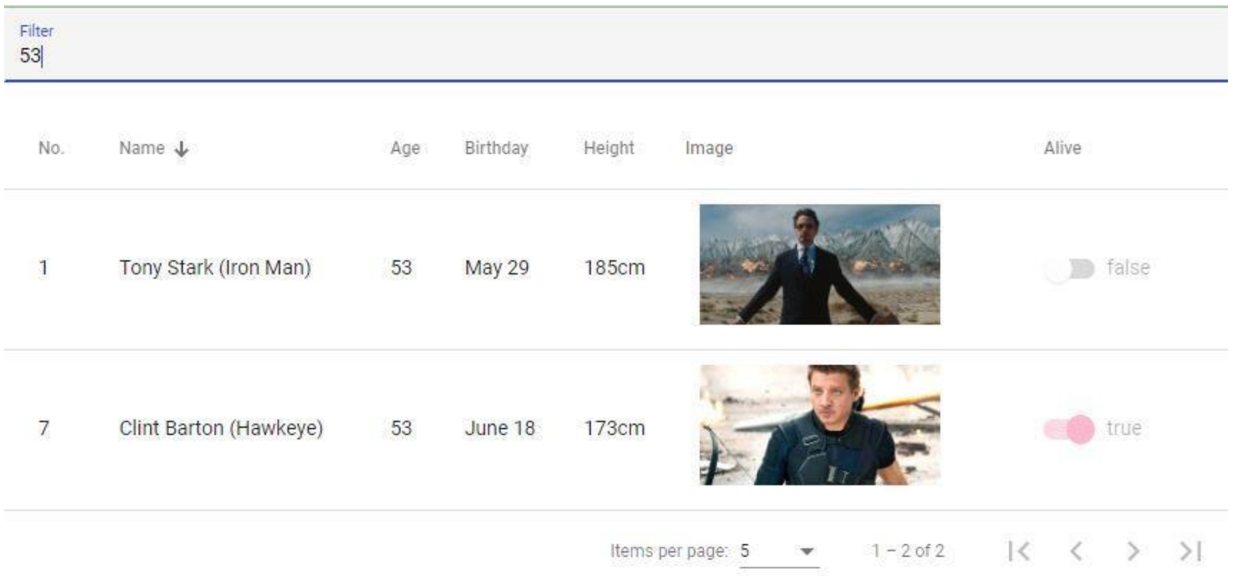
Q1 Expected Output



Filter

No.	Name ↓	Age	Birthday	Height	Image	Alive
10	Wanda Maximoff (Scarlet Witch)	30	February 10	168cm		true
11	Vision	3	May 29	May		true
1	Tony Stark (Iron Man)	53	May 29	185cm		false
4	Thor	1059		192cm		true
2	Steve Rogers (Captain America)	34	July 4	185cm		false

Items per page: 5 1 – 5 of 12



Filter
53

No.	Name ↓	Age	Birthday	Height	Image	Alive
1	Tony Stark (Iron Man)	53	May 29	185cm		false
7	Clint Barton (Hawkeye)	53	June 18	173cm		true

Items per page: 5 1 – 2 of 2

SECTION A TOTAL **18**

SECTION B – ANGULAR (10)

For Section B, you also need to complete one question. The question does not require access to an existing database file to be completed. It also does not require you to integrate any external libraries. All the source files are in the following zipped file:

Semester_Test_Section_B_Question.zip

Once you have completed the 1 question in this section, **upload** your entire project to the **Section B upload slot**. In other words, **Delete the 'node_modules' and '.angular' folders** then **zip** the project as the following **uXXXXXXXX_SecB.zip**, where the **XXXXXXXX** is your student number, e.g. **u12345678_SecB.zip**.

Your task is to complete the codebase to get the application to function as described below. For the question, please do the following:

- Read the instructions carefully.
- After downloading the zip file with the source, run it and see what it looks like prior to adding new functionality.
- Apply the necessary code changes to the specified file(s) in the required application.
- Upload your project as described above.

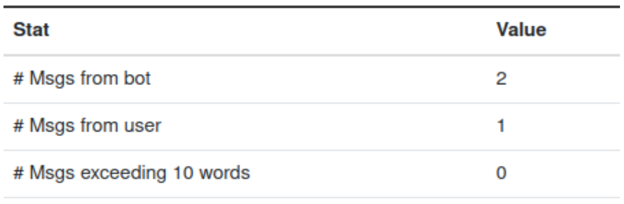
SECTION B - QUESTION 1 (10 MARKS)

A startup company in the chatbot space has created a new version of the ELIZA program (https://en.wikipedia.org/wiki/ELIZA) and they claim that their system is intelligent. In an effort to recruit customers, they have tasked you to build a web app that lets people interact with the program on a limited basis. Specifically, they allow people to submit a few messages to their ELIZA. Your job is to build an application that monitors the number of messages sent by a user, the number of messages sent by ELIZA, and the number of messages that exceed 10 words.

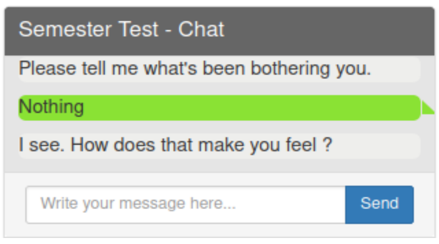
You are given a project that has an implementation of ELIZA, hence you do not have to implement the bot yourself. Similarly, the chat window that can be used by users to enter messages and see ELIZA's responses has already been implemented, hence you must not recreate it. Your task is only to edit the existing **ChatstatsComponent** (both **chatstats.component.ts** and **chatstats.component.html**) to monitor the current statistics and edit the "**onAddNewMessage**" of the **ChatPopupComponent** (in **chatpopup.component.ts**) to transfer each new message to your new component.

After running your code, the application must look like the following screenshot:

Q1 Expected Output



Stat	Value
# Msgs from bot	2
# Msgs from user	1
# Msgs exceeding 10 words	0



Semester Test - Chat

Please tell me what's been bothering you.

Nothing

I see. How does that make you feel ?

- For the chat statistics task **[total: 5 Marks]**
  - In the "**chatstats.component.html**" file, create a regular **bootstrap table** whose columns are labeled '**Stat**' and '**Value**'. *Note: Bootstrap has already been linked to the project via the **index.html** page, hence there is no need for you to add relink it manually* (2 marks)
  - In the above table, create a table row with the labeled "# Msgs from bot" and the calculated number of messages from a bot. For every **Message** (in **message.model.ts**), bot authored messages can be detected via the variable **isBotAuthored** (1 mark)
  - In the above table, create a table row with the labeled "# Msgs from user" and the calculated number of messages sent by a human. For every **Message** (in **message.model.ts**), messages that are not bot authored can be detected via the **Message**'s variable **isBotAuthored.** HINT: *If it is not bot authored* (1 mark)
  - In the above table, create a table row with the labeled "# Msgs exceeding 10 words" and the calculated number of messages with more than 10 words. You must consider all elements in a sentence that are separated by a single space as a word (e.g., the sentence "Is this an example ?" contains 5 words) (1 mark)

- Message transfer between 'chatpopup' and 'chatstats' task **[total: 5 Marks]**
  - In the method "**onAddNewMessage**" of **ChatPopupComponent** (in **chatpopup.component.ts**), transfer each message to your **ChatstatsComponent** (in **chatstats.component.ts**). *HINT: Inject the ChatMessageManagerService in the constructor and call its AddMessage method, parsing the message* (4 marks).
  - For every new message received in **ChatstatsComponent** (in **chatstats.component.ts**), if the incoming **Message** (in **message.model.ts**) is **isBotAuthored** then add an increment to the variable you use to monitor the number of bot authored messages, if it is not **isBotAuthored** then add an increment to the variable you used to monitor the number of messages authored by a user, and if the **Message**'s **content** has more than ten words (i.e., items separated by a single space) then add an increment to the variable you use to monitor the number of messages exceeding ten words (1 mark)

- **NB: Do not forget to run "npm install" to allow the application packages to be re-installed.**
- **NB: The application was created using Visual Studio Code and Angular. You have already been given a document for any environment pre-setup installations required and how to do it, shared on the Course Module ClickUP site. Therefore, for this question, no additional environment pre-setup installations are needed.**

SECTION B TOTAL **10**

SECTION C – IONIC (25)

For Section C, you also need to complete one question. The question does not require access to an existing database file to be completed. It also does not require you to integrate any external libraries. All the source files are in the following zipped file:

Semester_Test_Section_C_Question.zip

Once you have completed the 1 question in this section, **upload** your entire project to the **Section C upload slot**. In other words, **Delete the 'node_modules' and '.angular' folders** then **zip** the project as the following **uXXXXXXXX_SecC.zip**, where the **XXXXXXXX** is your student number, e.g. **u12345678_SecC.zip**.

Your task is to complete the codebase to get the application to function as described below. For the question, please do the following:

- Read the instructions carefully.
- After downloading the zip file with the source, run it and see what it looks like prior to adding new functionality.
- Apply the necessary code changes to the specified file(s) in the required application.
- Upload your project as described above.

SECTION C - QUESTION 1 (25 MARKS)

A new company would like to automate the creation of new emails for all its users. They have tasked you to build a mobile app that allows each user to enter their details and create a new company email for them.

Your job is to build an application that receives the user's first and last name and then generate a new company email based on the details that they have provided.

You are given a project that has the completed **HTML for all components as well as an empty service** (**MainService**). You must edit the **home.page.ts** file, to check if the register form submitted is valid (*based on the provided validators*) and display a **toast alert** to indicate whether the **registration was successful or unsuccessful** and then **call a function within the service and navigate to the details component**. *Note: You will need to include all necessary injections into the constructor.*

If the register form is valid, the toast alert message should be '*Registration successful*' and if the register form is invalid, the toast alert message should be '*Registration unsuccessful*'. *Note: the toast should be displayed at the top of the page and should display for a duration of 2 seconds.*

If the form is valid and the toast alert has been displayed, you will then need to make a call to the **MainService** (**createEmail** function) and *send the entire register form as a parameter*.

After which you will be required to navigate to the **details component**. *Note: you should **not** navigate to the details component if the form is invalid.*

The application will begin to display as seen in the **Expected Output (***Figure 1: Register image***)**.

In this question, you will be required to edit the **home.page.ts** and the **main.service.ts**.

- **Home.page.ts**: **[total: 15 Marks]**
    - You will need to create an asynchronous function called **submitRegister**, within this function you will be require to make use of a toast controller to create an alert that is positioned at the top with a duration of 2 seconds. (4 marks).
    - After you have created the alert, you will be required to check if the **register form** is valid (1 mark).
    - If the register form is **valid**, you will need to set the toast controller's message to '***Registration successful***', after which you will need to call the **createEmail** function that is in the **MainService** and send the entire register form as a parameter. Lastly, you will need to present the toast controller and navigate to the details component. (7 marks)
    - If the **register form** is *invalid*, you will need to set the **toast controller's** message to '*Registration unsuccessful*' and present the toast controller. The user should stay on the home page until the registration has been successful. (4 marks)

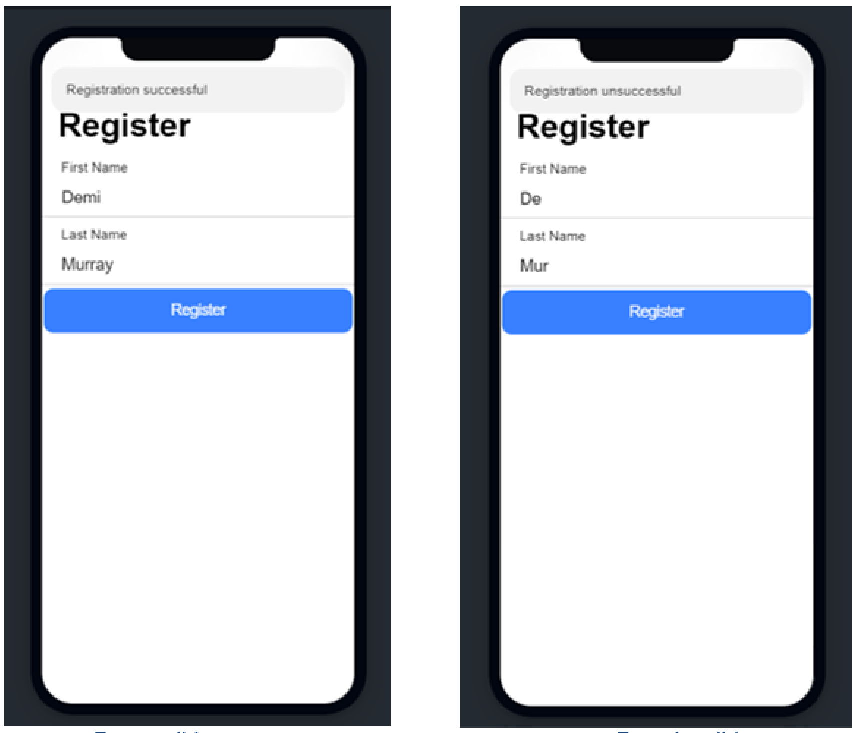
You will then need to create a **function** within the main service called **createEmail**, that will accept an object of type **any** as a parameter. This function needs to create a new email address for the user based on their first and last name and set this email address, as well as the first and last name as an item in **local storage** under the key name '*userDetails*'.

- **Main.service.ts**: **[total: 10 Marks]**
  - You will need to create a function called **createEmail**, that receives an **object** of type *any* in its parameter – the object received will be the **register form**. (2 marks)
  - In the above function, create a new object that will have properties for **firstName**, **lastName**, and **email**. (1 mark)
  - You will need to set the **firstName** and **lastName** properties as the *user's first name and last name* (*using the object received in the parameter*). You will then need to set the value of the email property to the **first letter of the first name**, the **first 3 letters of the last name** and include '**@company.com**' at the end. *For example, firstName: Demi, lastName: Murray, email: DMur@company.com. HINT: you will use the substring method on the string.*
  - You will then be required to set the entire object as an item in local storage, under the key name '**userDetails**'. *HINT: you must **stringify** the object when setting it as an item in local storage.* (3 marks)

- Once you are done and have successfully implemented the required code, your final details page will look like that displayed in the **Expected Output** below **(***Figure 2: User details image***)**. *Note: please do **not** edit anything in the details component):*

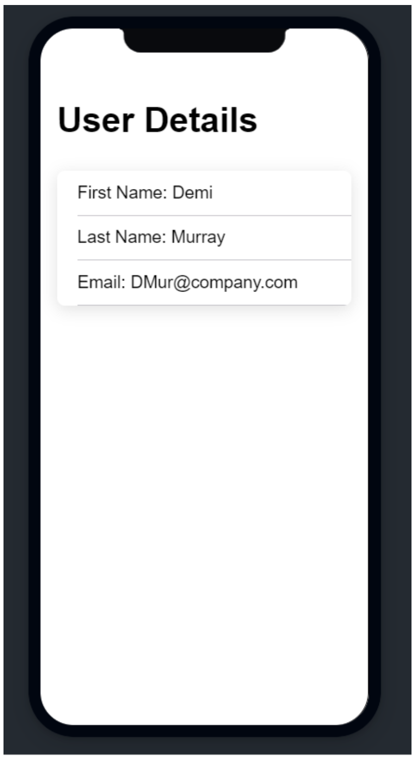
- **NB: Do not forget to run "npm install" to allow the application packages to be re-installed.**
- **NB: The application was created using Visual Studio Code and Ionic. You have already been given a document for any environment pre-setup installations required and how to do it, shared on the Course Module ClickUP site. Therefore, for this question, no additional environment pre-setup installations are needed.**

Q1 Expected Output



Form valid *Form invalid*

Figure 1: Register image


User Details

First Name: Demi

Last Name: Murray

Email: DMur@company.com

Figure 2: User details image

SECTION C TOTAL **25**