# IT2154 Practical Assignment 2: TypeScript Concurrent Programming and Ionic Application (30 Marks 30% ICA)

## Instructions

- This is a group assignment. You will work on this assignment in a team of 2 members. Your tutor has the right to grant a team of 3 members due to the class size is an odd number.
- You are working within your team on this assignment. You should not copy or practice plagiarism across teams. Student who is caught cheating and practice plagiarism will face the following penalty according to NYP policy.
  - First time in any assessment will fail the entire module.
  - Second time in any assessment will fail the all modules in that semester.
  - Third time in any assessment will be removed from the Polytechnic
- Download the `student_resource.zip` file from Blackboard. In the zip file, you are given two projects.
  - `Dazala` is a C# dotnet MVC core web application which will be used as the backend API service. You don't need to make any change to this project.
  - `mobiledzala` is a pre- & partially-developed Ionic React app. You are supposed to complete the implementatoin in this app according to the instructions.
- You are required to submit the completed `mobiledazala` ionic app. Any extra submitted materials will not be assessed.
- Include your names and admin numbers in the begining of the source code files that you have modifies
- Zip up the `mobiledzala` folder and submit via Blackboard, remove the `node_modules` folder inside might help to reduce the size of the zip file.
- Deadline: 16th August 2020 23:59

| Part | Marks |
|---|---|
| Welcome Tab | 4 Marks |
| Product Tab | 13 Marks |
| Shopping Cart Tab | 8 Marks |
| Challenge | 5 Marks |

## Setup

1. In a seperate command prompt / terminal, change the working folder to `Dazala` project's folder.

2. Run the following to start the project

```
$ dotnet run
```

3. Make sure the API web service is running by opening `https://localhost:5001/api/product/all` in the browser or Postman
4. Install `npm` and `node.js` if you have not done so. They can be downloaded from `https://nodejs.org/en/download/`
5. Install `@ionic/cli` annd `capacitor` if you have not done so.

```
$ npm install @ionic/cli capacitor
```

6. Use Visual Studio Code to open the `mobiledazala` folder. Start a terminal in VSC, assuming you are in the root folder of `mobiledazala`, run

```
$ npm install
```

7. You are given a pre-created ionic react project with a 3-tab template. The plan of development is as follows,
   1) Tab 1 will be a welcome page.
   2) Tab 2 will be a product listing page.
   3) Tab 3 will be a shopping cart page.

# Part 1: Welcome Tab (4 Marks)

In `Tab1.tsx`,

1. Import `IonLabel` from the `'@ionic/react'` library.
2. In the `IonPage`, below the `IonHeader`, you find a

```
<IonContent class="center">
   {/* TODO: fixme, but you might not need the curly brackets */}
</IonContent>
```

add an `IonLabel` with the message "Welcome to Dazala!". You may delete the part with `{/* TODO: ... */}`.

3. Change the content in the two `IonTitle` into "Welcome to Dazala".
4. A sample screenshot can be found in Figure 1.

# Part 2: Product Tab (13 Marks)

In `Tab2.tsx`, we are going to build a product listing page. The listings come from the backend API end point `https://localhost:50001/api/product/all`. This end point returns the list of product in json format. Hence in this tab, we need to define an equivalent data structure to house the results coming from the API, then render it into the `IonPage`.

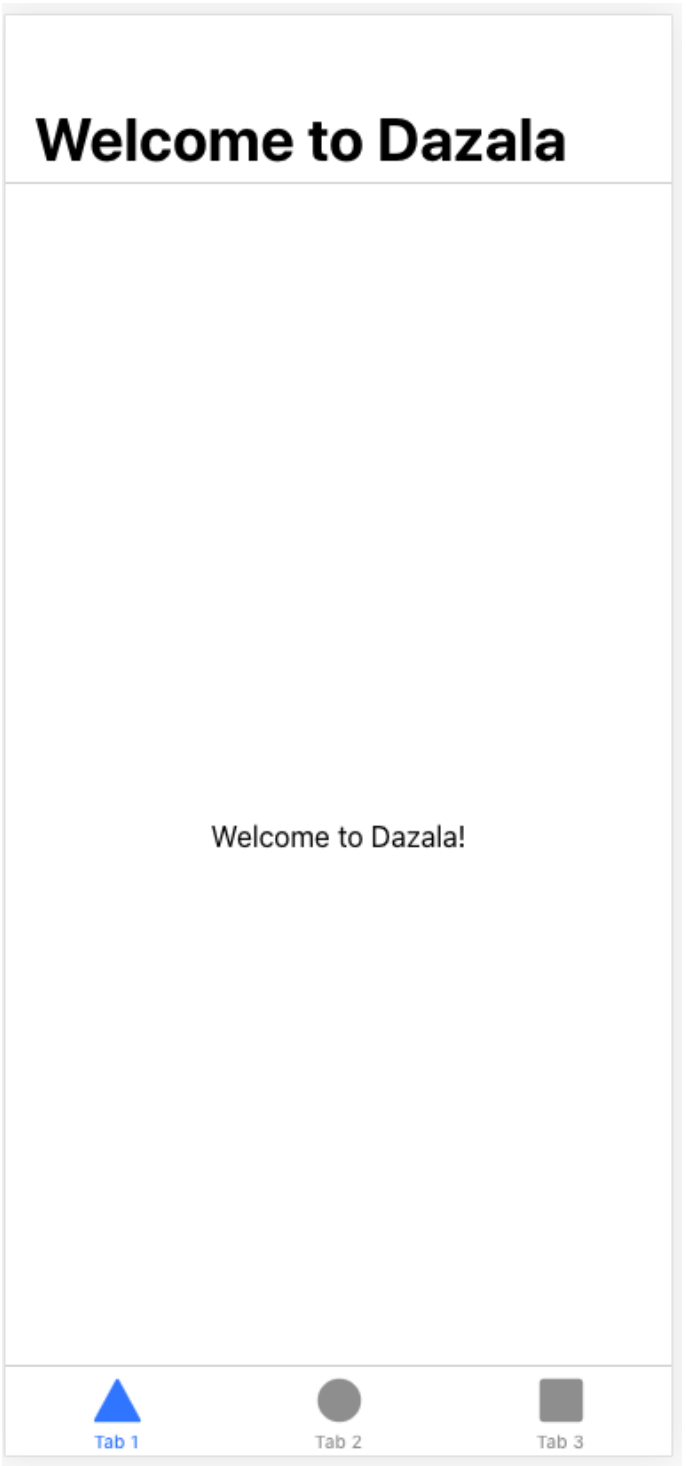


Figure 1: Welcome Page

As the interaction between the user interface and the backend API must be made simulatanously with some synchronization, Promise and call-back will the first choice for implementation.

1. Define an `interface` named `Product` which should has the following fields
    1. `id` of type `string`
    2. `name` of type `string`
    3. `description` of type `string`
    4. `price` of type `number`
2. Define a state object of type `Product[]` which has two accessors, `products` and `setProducts`. The state object should be initialized to an empty array.
3. Define a Promise named `fetchProducts` which peforms the following in sequence
    1. makes an HTTP GET API call to `"https://localhost:5001/api/product/all"` to retrieve all the products from the backend.
    2. When the result is ready, converts it into an array of `Product`
    3. Save the array into the `products` state object.
    4. Finally, set the `showLoading` state object to `false`.
4. Define a `makeLink` function which takes a `product` and return a link of `string` type. The link is created by concatenating `"/tab3/"` with the product id.
5. Complete the `IonList` in the `IonPage` to render the list of `products`. For each product
    1. an `IonItem` should be created whose routerLink is set to `/tab3/{id}` where {id} is the id of the product
    2. an `IonLabel` should be used in displaying the item's name, description and price.
6. A sample screenshot is attached in Figure 2,

# Part 3: Shopping Cart Tab (8 Marks)

In `Tab3.tsx`, we are about to implement a shoppping cart that keeps track of the products that the user would like to purchase. In `Tab2.tsx` when a product is clicked, the product ID is passed over to `Tab3.tsx`. This is achieved by adding a `:id` as a parameter into the routing rules in `App.tsx` and incorporating a `CartPageProps` into the functional component `React.FC` in `Tab3.tsx`. The product id will be captured as `match.params.id` in `Tab3.tsx`. These modification have been applied to the template code given to you.

The overall flow in the `Tab3.tsx` is as follows * A cart will be implemented as a dictionary stored as a LocalStorage object. * When given a product id, we check whether the product id is already existing in the cart. * If the cart contains

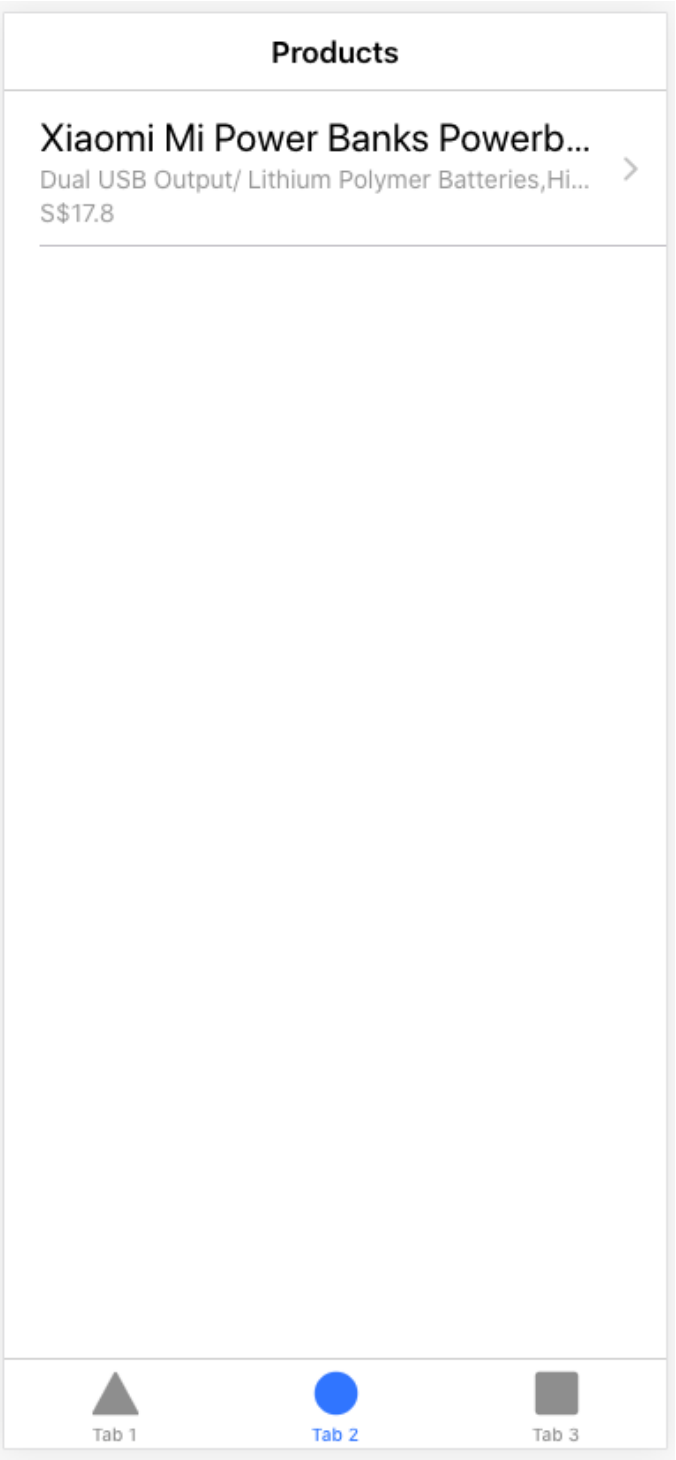


Figure 2: Product Listing

the product id, increment its quantity by one, otherwise add an entry to the shopping cart with the product id and 1 as the quatity. * Display the list of items in the shopping cart in the `IonPage`.

1. `addItem` function is implemented as a normal function, which takes care of adding a product id into a shopping cart. Modify this function so that it returns a `Promise<Cart>` instead of a `Cart`.
2. The function `allItem` should takes a cart and return the list of `CartItem`s stored inside. The function's body is incomplete. Complete the function's body according to the requirement.
3. Complete the call of `useEffect` so that it would chain up the following actions in sequence.
    1. call `addItem` to add `match.params.id` into the cart.
    2. given the new cart, update the LocalStorage object.
    3. finally, set `showLoading` to `false`.
4. In `IonList` element in the `IonPage`, display the list of shopping cart items.
5. A sample screenshot is attached in Figure 3,

# Part 4: Challenge (5 Marks)

The shopping cart page is not very user friendly, we should display the name of the product instead of the product id. Please update `Tab3.tsx` accordingly to accomodate this change.

A sample screenshot is attached in Figure 4,

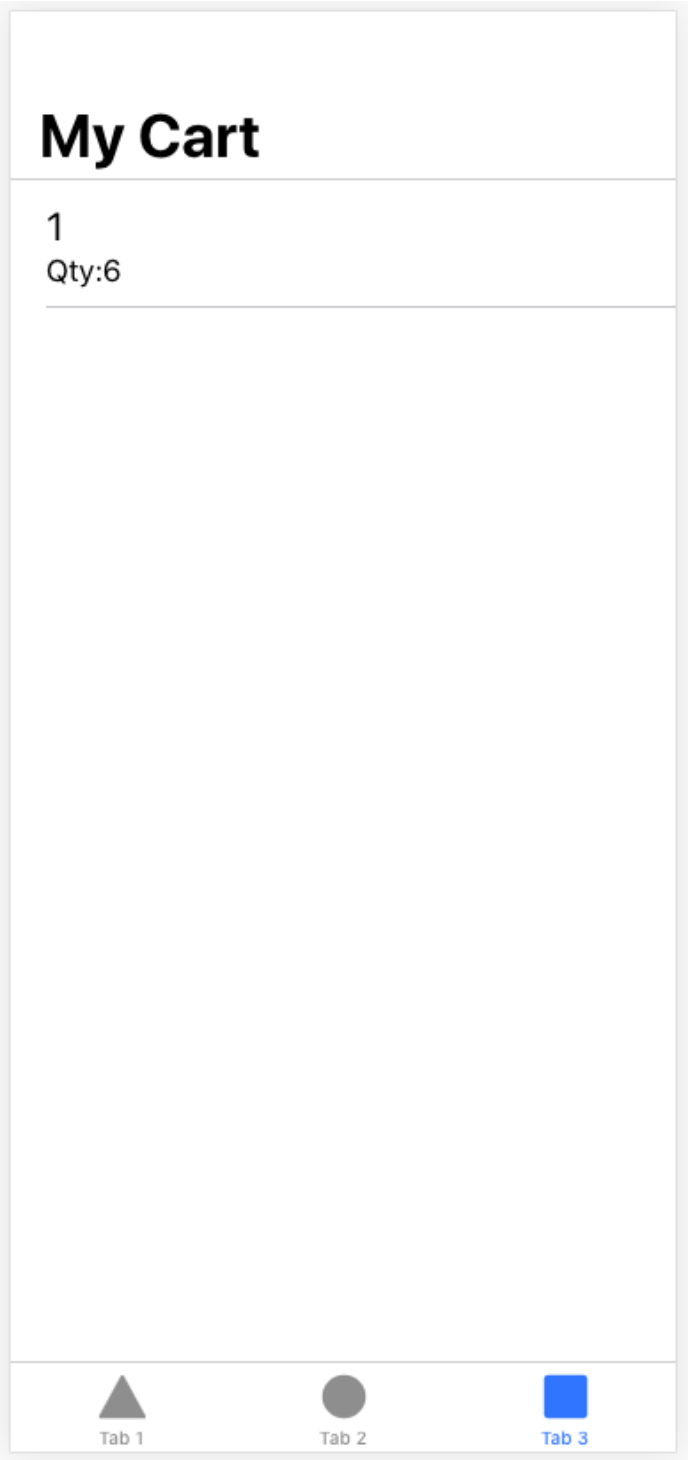


Figure 3: Shopping Cart

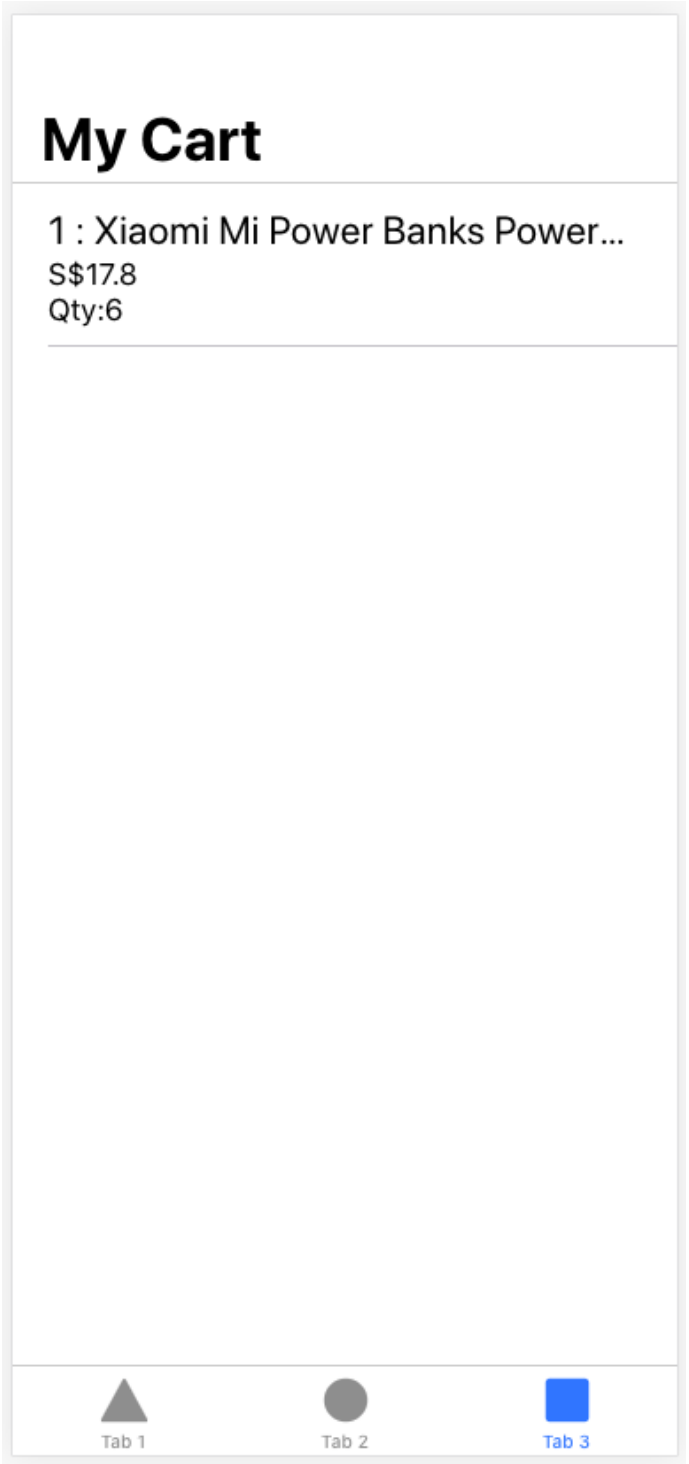


Figure 4: Shopping Cart with product name