**Hands on Assessment**

**Developing an application using Angular 7**

**Duration: 3 hours Max Marks: 50**

**Instructions to use the project file provided:**

* You need to code the assessment in the folder containing partially coded project
* Read the problem statement, examples and the other details provided carefully and implement the solution
* Download the project **BuffetBooking** in to your system and unzip it
* **BuffetBookingWS** folder contains the **server side code**
* **BuffetBookingUI** folder must contain the **client side code**
* **DO NOT** alter the function name or the argument list of the function that is provided to you
* **DO NOT** alter the **meta data** provided in the components
* **DO NOT** add any new functions apart from the one given in the file where you write the solution
* Use the **bootstrap classes** wherever necessary

**Note:**

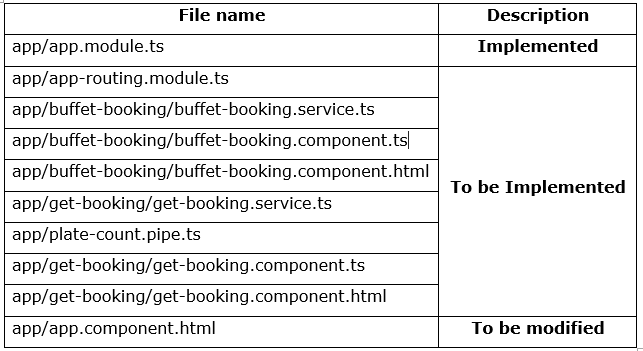
* The express application runs in port **3080**
* Execute **npm install** command in both the **BuffetBookingWS** and **BuffetBookingUI** folders
* Start the web service using appropriatecommand
* Set up the database
* Submit the employee id folder containing only the *src* of the project
* Any compilation errors will lead to zero marks in Hands on exam

**Problem Description:**

Infy Restaurant wants to automate the process of booking a buffet to their customers. The following functionalities are required to be implemented.

* Book a buffet
* View bookings

**In BuffetBookingUI folder:**

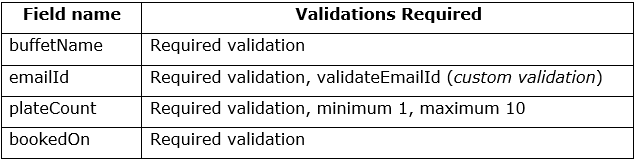


**app/buffet-booking/buffet-booking.service.ts: (To be implemented)**

* The method **bookBuffet(data)** consumes the web service exposed at the URI.
  + It sends a PUT request to URI “**http://localhost:3080/bookBuffet/:emailId**” by passing form data
  + After sending the request, the response must be converted to an observable of type **BuffetBooking**
  + Return the response back to the **buffet-booking.component.ts**

**app/buffet-booking/buffet-booking.component.ts: (To be implemented)**

* **BuffetBooking** class contains the following attributes:
  + errorMessage : string
  + successMessage : string
  + buffetBookingForm: FormGroup
  + price: Number
* Inject the instances of **FormBuilder** and **BuffetBookingService** into the constructor
* Create a FormGroup object named **buffetBookingForm** using FormBuilder instance, having **buffetName, emailId, plateCount** & **bookedOn** as form controls
* Add the validators for each form controls as specified below:



**validateEmailId(formControl) : (to be implemented)**

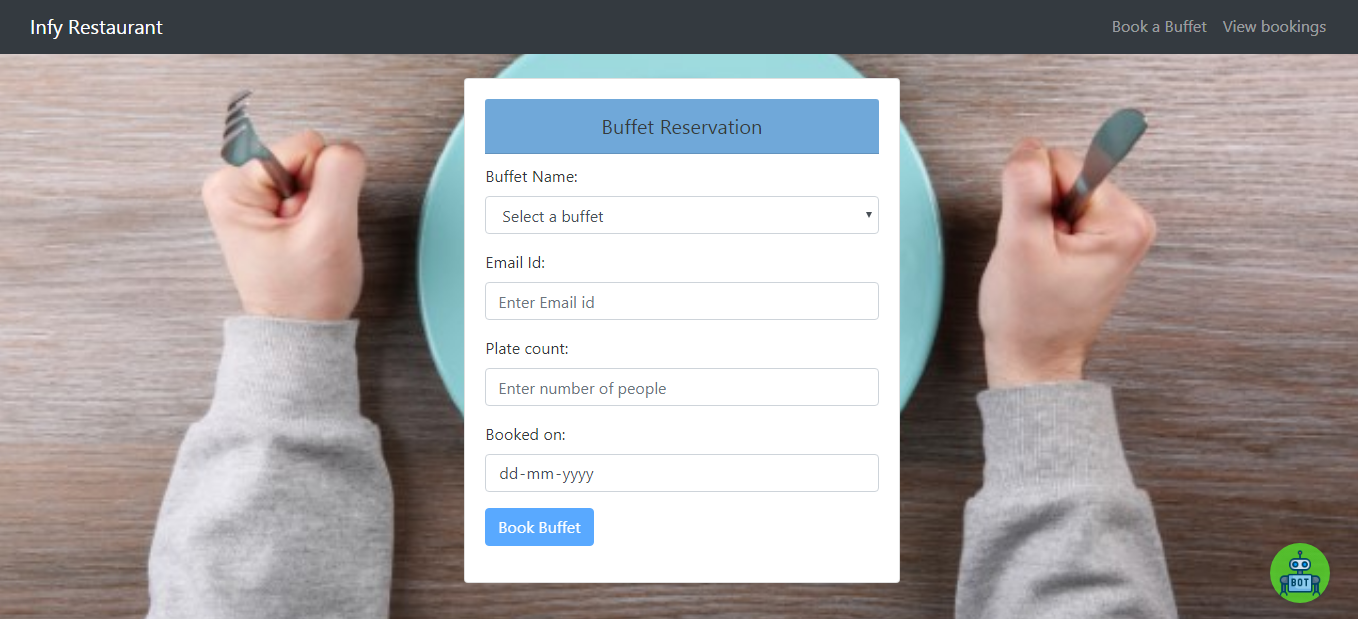
* This function should validate the emailId entered by the user against the following pattern

**<<part1>>@<<part2>>.<<part3>>**

* + **part1**:- It should begin with an alphabet followed by alphabets, numbers, dot[.] or underscore(\_)
  + **part2:-** Only alphabets are allowed
  + **part3:-** Minimum two and maximum four alphabets are allowed
* If the validation fails, the error message **“Enter a valid Email id”** should be stored as a JSON object for the **emailError** property of the custom validator
* Else, it should return **null**
* **bookBuffet()** : **(To be implemented)**
  + It should reset the values of errorMessage and successMessage to **null**
  + It should invoke **bookBuffet()** of **BuffetBookingService** by passing the value of **buffetBookingForm** object as a parameter, which in turn returns an observable
  + The Observable object returned should be subscribed by passing two callback functions i.e – success, error
  + The success callback should populate the **successMessage** & **price** with the message & price property from the response
  + The error callback should populate the **errorMessage** with the message in response

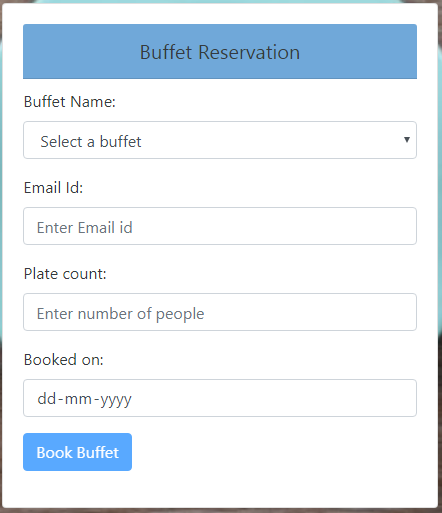
**app/app.component.html (To be modified)**

* Add the tag required to load the view of routed component
* When **“http://localhost:4200”** is entered in the browser, the **BuffetBookingComponent** should be displayed as shown below:

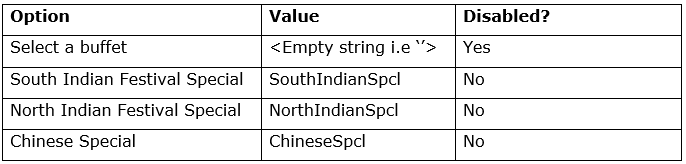


**app/buffet-booking/buffet-booking.component.html : (To be implemented)**

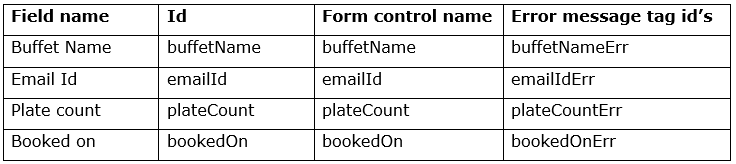
When **Book a buffet** is clicked from **app-component.html**, then initially **buffet-booking.component.html** view should be displayed as shown below:



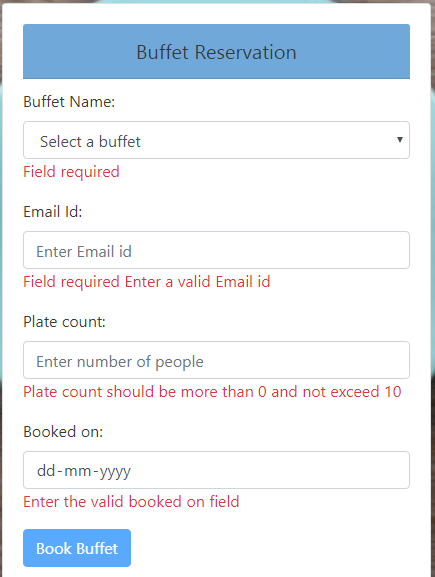
* Populate the Buffet Name field with the following static values



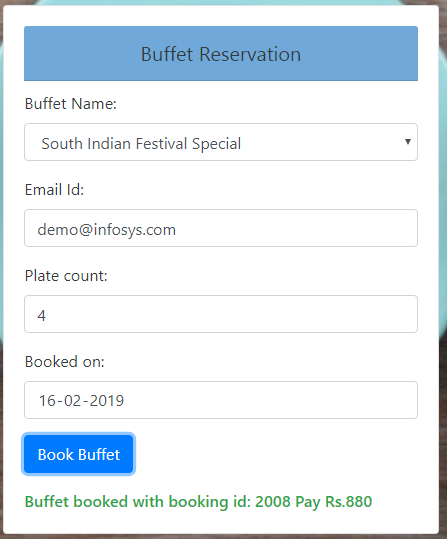
* Form should be bound to **buffetBookingForm** object
* The form fields should be bound to the form builder metadata values given in **BuffetBookingComponent** class**.**
* The following id and form control name has to be added for each fields:



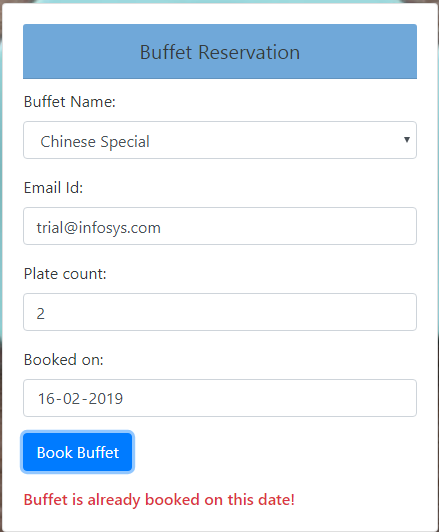
* Validation messages should be rendered only when the field becomes **dirty** (Email Id & Plate count) **or touched** (Buffet name & Booked on) or validation fails for a form control as shown below:



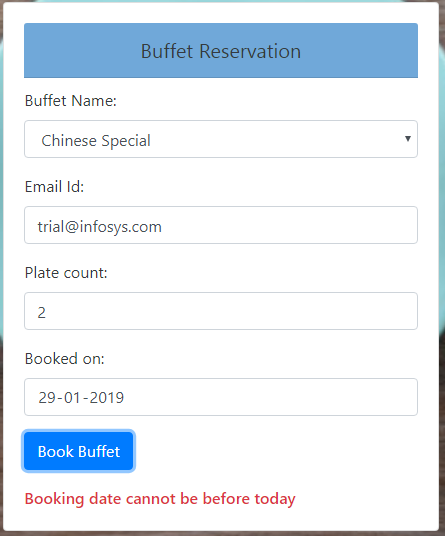
* Button should be disabled; until the form becomes valid
* When the form is submitted, **bookBuffet()** of **BuffetBookingComponent** should be invoked
* On successful execution, the success message should be displayed as shown below:



* In case if you are trying to book the buffet on a date which is already booked, the error message should be displayed as shown below:



* In case you enter a past date, the error message should be displayed as shown below:



**get-booking/get-booking.service.ts (To be implemented)**

* Method **getBooking(emailId)** will send **GET** request to the URI: **http://localhost:3080/fetchBooking/:emailId** which in turn returns an Observable of type **BuffetBooking array.**
* The data is returned back to **GetBookingComponent**

**get-booking/get-booking.component.ts (To be implemented)**

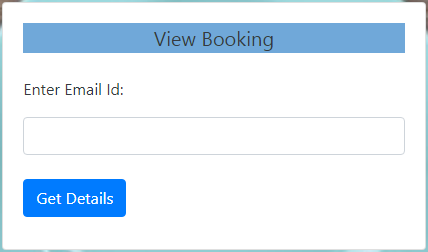
* **GetBookingComponent** class consists of the following properties and methods:
  + - errorMessage: string;
    - emailId: string;
    - selectedBooking: BuffetBooking[]
    - getBooking()
* Instance of **GetBookingService** should be injected into the constructor

**getBooking(): (to be implemented)**

* + Invoke the **getBooking(emailId)** method of **getBookingService** which in turn returns an observable. The Observable object returned should be subscribed by passing two callback functions
    - The success callback should populate the **selectedBooking** with the data in response
    - The failure callback should populate the **errorMessage** with the message in response

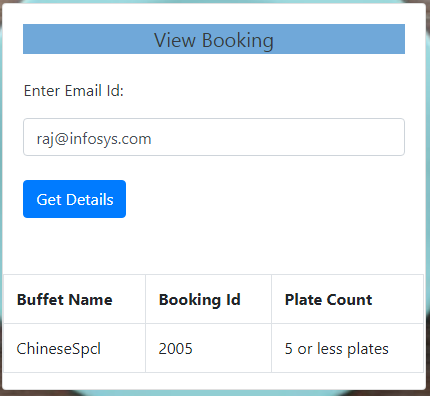
**get-booking/get-booking.component.html (To be implemented)**

When **Bookings** is clicked from **app-component.html**, then initially **get -booking.component.html** view should be displayed as shown below:



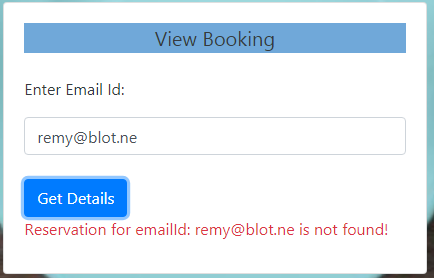
**Note: DO NOT USE a form**

* When user enters email id & clicks on **Get Details** button, details of that particular email Id should be shown as below:



**Note:** Plate count is modified using custom pipe

* If entered email Id doesn’t exist, it should display error as per given screenshot



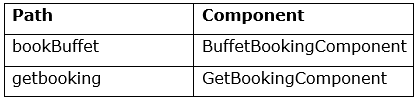
* If the correct email id is mentioned and **Get Details** button is clicked, the error message ( if any ) must disappear and the appropriate details must be displayed

**app/plate-count.pipe.ts (To be implemented)**

* Transform the **Plate count** using this pipe
* If the **Plate count** is greater than 5, it should be transformed as **more than 5 plates**
* Else it should be transformed as **5 or less plates**

**app/app-routing.module.ts (To be implemented)**

* Configure the routes for the below information:



* In case an **Invalid URL** is provided, the user must be redirected to the **Book Buffet page**.

~~~~~~~~~~~~~~ All the Best ~~~~~~~~~~~~~~