Prerequisite
All basic angular assignments.

Setup
Install NVM to manage node and npm versions. Node v16+.
Angular version 13
Visual Studio Code Editor v 1.65 Feb 2022

General Instruction

- Code must be properly indented and as per the indentation format shared with you.
- Code should be well formatted to provide readability to the code.
- Code must use meaningful variable names, function/method names, file names which defines its purpose.
- Code must compile without any errors/warnings.
- Code must produce output strictly in the expected format wherever such format is provided.
- Add appropriate code for exception handling and print a meaningful exception message in case of an exception.
- Optimal code comment should be used to define code purpose & good practice for code maintainability.
- Unwanted, unused and commented source codes should not be part of your code.
- AOT compiled code and error free prod build for all the assignments.
- Use Angular devTool from Basic Angular assignment wherever required.
- Code for all exercises must be pushed under the branch name allocated to you on a Git repository. Branch name must be Participant's Merce username (e.g. "indulekhas" for Indulekha Singh)

Advance assignments

1. **[Change detection & Change detection Strategy, 12 hours]**
   Create an Angular app with a new component named "ChangeDetect". Create a class "Bulletin" and with properties "news_title" and "news_date" in the bulletin.ts file with a constructor to with these properties, and use this class in the "ChangeDetect" component, and bind the two properties in the view template of "ChangeDetect". Using the 2 properties demonstrate the change detection with "OnPush" and "Default" strategy. Create function "changeDetectionDefault()" for "default" strategy and "changeDetectionOnPush()" for "OnPush" strategy with necessary logic in "src/app/app.component.ts" and use the selector declared in "ChangeDetect" inside the "src/app/app.component.ts" template.
   ***Note** : Defined date in RFC 3339 format e.g. 1985-04-12T23:20:50.52Z for a global UTC timezone timestamp or 2015-12-31T23:59:50+05:30 for a timezone-included timestamp.*

2. **[Dynamic Component, @ViewChild, 8 hours]**
   Extend the above app and create a new component named "LiveNews". Use this

component to load the Bulletin news dynamically in “app/src/app.component.ts”. Create a “BulletinService” service. Use this service to get the list of bulletin news to display through the “LiveNews” component.“LiveNews” will cycle through the array of bulletins every 5 seconds and render the bulletin news in the view template.

3. **[Custom Pipes, 6 hours]**
   Extend the above app to implement custom pipes.
   a. Create a pipe to wrap the “news_title”. If the “news_title” length is greater than 20 words then the text should be wrapped via pipe. Create a pipe with the name as “wraptext”. This pipe will accept 2 parameters as text value and number of words to be displayed.
   b. Create another pipe to format the “news_date”. Name the pipe as “formatDate”, this pipe should accept 2 parameters as text value and another, the type in which datetime needs to be formatted. Accepted values for second parameters would be ‘short-date’ and ‘short-time’. The ‘short-date’ would transform the “news_date” as e.g “20 Jun 2019” and ‘short-time’ would transform the “news_date” as e.g. “14:20”.
      Use these created pipes in the template to transform the property values in the view template.

4. **[Custom attribute directives, 4 hours]**
   Extend the above assignment to create a custom attribute. Use the attribute naming convention as per best practices used in earlier assignments. Highlight the “news_title” with yellow color when the mouse cursor hovers on the “news_title” rendered in the view template.

5. **[Template based form,localStorage & Validation, 10 hours]**
   Extent the app created at ***Bootstrap, Component Template from basic angular assignment***. Create a new “AddCustomerByTplFrm” component. Design a form using template based form with below fields and validation.
   a. First name: Required & Character validation
   b. Last name: Options & Character validation
   c. Email: Required & accept valid email address
   d. DOB: Calendar component and validation will be such that the customer should be above 18 years old.
   e. Mobile: Validation for 10 digit number.
   f. Address: Optional input value.
   g. Pincode: Required & validation for 6 digit number.
   h. Submit button: Save the details to localStorage and display a success message in toast form. Similarly appropriate error messages should be displayed when validation fails and note that the details should not be stored in localStorage in case of failure.

6. **[Reactive form & validation, 12 hours]**

Convert the above app with Reactive form with validation. Create a new component named “AddCustomerByRctFrm”.

7. **[Http Client POST API, Observable, localStorage 12 hours]**
   Modify the above assignment to create a simple Registration form with bootstrap. This form should call to the API endpoint at https://reqres.in/ which will accept “email” and “password”.The API call should be implemented through service file using “Observable (subscribe() method)” and not via “promises”. Used the naming convention as per best practices followed in the earlier assignments. After each successful registration save a response token value in localStorage named as “token”.

8. **[Http Client GET API & Services & Observable, 10 hours]**
   Create a new user listing Component in the assignment created at ***Bootstrap, Component Template from basic angular assignment*** similar to the design created for customer listing view. This component should call to the API endpoint from https://reqres.in/ to fetch a list of all the users and display in the HTML template in **listview** using ngFor directives.The API call should be implemented through a Angular Service. The last column of each row should have 3 action icons.
   a. View details (eye icon)
   b. Edit (pencil icon)
   c. Delete (bin icon)

9. **[Http Client API, DI Services & Observable, 12 hours]**
   Extend the above app to implement the above 3 action icons defined in the last column of the user listing view. Check the API from https://reqres.in/**.**
   a. When the user clicks on “View details icon” user detail API will be called and the response should be displayed in a new component named as “UserDetail”.
   b. When the user clicks on “Edit icon” user detail API will be called and the response should be displayed in a new component named as “EditDetail”. All the data should be mapped in a form with an “Update” button.
   c. When a user clicks the “Update” button, an update API for the user needs to be called and a success or fail message needs to be displayed based on the response received and the component should navigate to the user listing view.
   d. When the user clicks on “Delete icon” then a confirmation prompt would ask “Are you sure to delete the user?”. When the user clicks yes, then the delete API will be called and the record will be removed from the user list view.

      Each successful operation will be notified with a proper success message through alert box and error with proper alert box.

10. **[Interceptor, 4 hours]**
    Modify the above app to implement an interceptor. Using the concept of Interceptor Service, pass the stored localStorage value from “token” in the header as “token” in all

the above API except Registration API.

11. **[Miscellaneous angular components, 6 hours]**
    Create a new Component “MyDemoComp” in the above Angular application to perform and display various components based on users selections. The screen will have the following buttons with the actions associated with them. Usage of any well suited MIT plugin for the best for these components can be used for the demonstration.
    a. Alert : Clicking on the alert button will display a well formatted alert using Angular components.
    b. Toast : Clicking on toast button will display a toast with some valid message for 3 seconds and then the message should disappear.
    c. Loader: Clicking on loader will display the loader for 3 seconds and should disappear.
    d. Confirm: Click on confirm button should show a dialog box with the confirmation layout which will have 2 buttons as Okay and Cancel.
    e. Input Confirm: Click on the input confirm button should show a dialog box with the confirmation layout which will have 2 buttons as Okay and Cancel and an input field to accept any comment.

12. **[Error handling, 14 hours]**
    Create an Angular application to demonstrate a simple working calculator. The calculator should have basic 4 operations. Use try catch block to handle errors
    a. Addition
    b. Subtraction
    c. Multiplication
    d. Division
       The app should take care of all the exceptions while performing the operations like division by zero,operation of more than single digit, facility to perform decimal number operation. The UI should be designed considering any standard Calculator with decimal operation feature.

13. **[Angular Material basic component ,12 hours]**
    Create an Angular application with Angular material for demonstrating following components from https://material.angular.io/components/categories
    a. Stepper
    b. Datepicker
    c. Dialog
    d. Sorting Table
    e. Chips (Advance selector)
    f. Sidenav

14. **[Angular Material table component ,12 hours]**
    Create an Angular application with Angular material to demonstrate various table options for <table mat-table> from https://material.angular.io/components/table/overview

a. Basic table with Styling and footer row.
b. Dynamically changing table data with Add,Remove and shuffle actions.
c. Expandable table with filter

15. **[Angular Material autocomplete component ,6 hours]**
    Create an Angular application with Angular material to demonstrate various autocomplete from https://material.angular.io/components/autocomplete/overview
    a. Simple autocomplete.
    b. First highlighted option.
    c. Option group autocomplete with long list.
    d. List with images.

16. **[Create UAT environment support in angular.json, 5 hours]**
    For any assignment from the above, create a support for a new environment as UAT. The app should be compiled with this newly created environment.

17. **[Build and serve the app, 2 hours]**
    Build the above API related app using “ng-build –prod”. Use the “http-server” npm package to serve and test the app.