ONLINE TEST\_APPLICATION DDOCUMENTATION

Mean\_Phase2

@simplilearn

Submitted by

Registered email: [paramitapp10@gmail.com](mailto:paramitapp10@gmail.com)

lik to github repository:

Paramita

Link to github repository: [**https://github.com/pipilika10/test**](https://github.com/pipilika10/test)

Refistered emailed in sinplilearn:

[Paramita22pal@gmail.com](mailto:Paramita22pal@gmail.com)

Table of Contents

[Abstract 3](#_Toc129999403)

[1 Introduction 4](#_Toc129999404)

**2. The guiding steps……………………………………………………………………………………………………………………………..3**

Abstract

In the Phase2 of Full Stack development course with simplilearn, a test application is developed to bring the learners in full view of recent type of application development with **node js, type script, ANGULAR/CLI, bootstrap using VSCode.**

A step by step learning is conducted to understand the basic steps.

1. Introduction

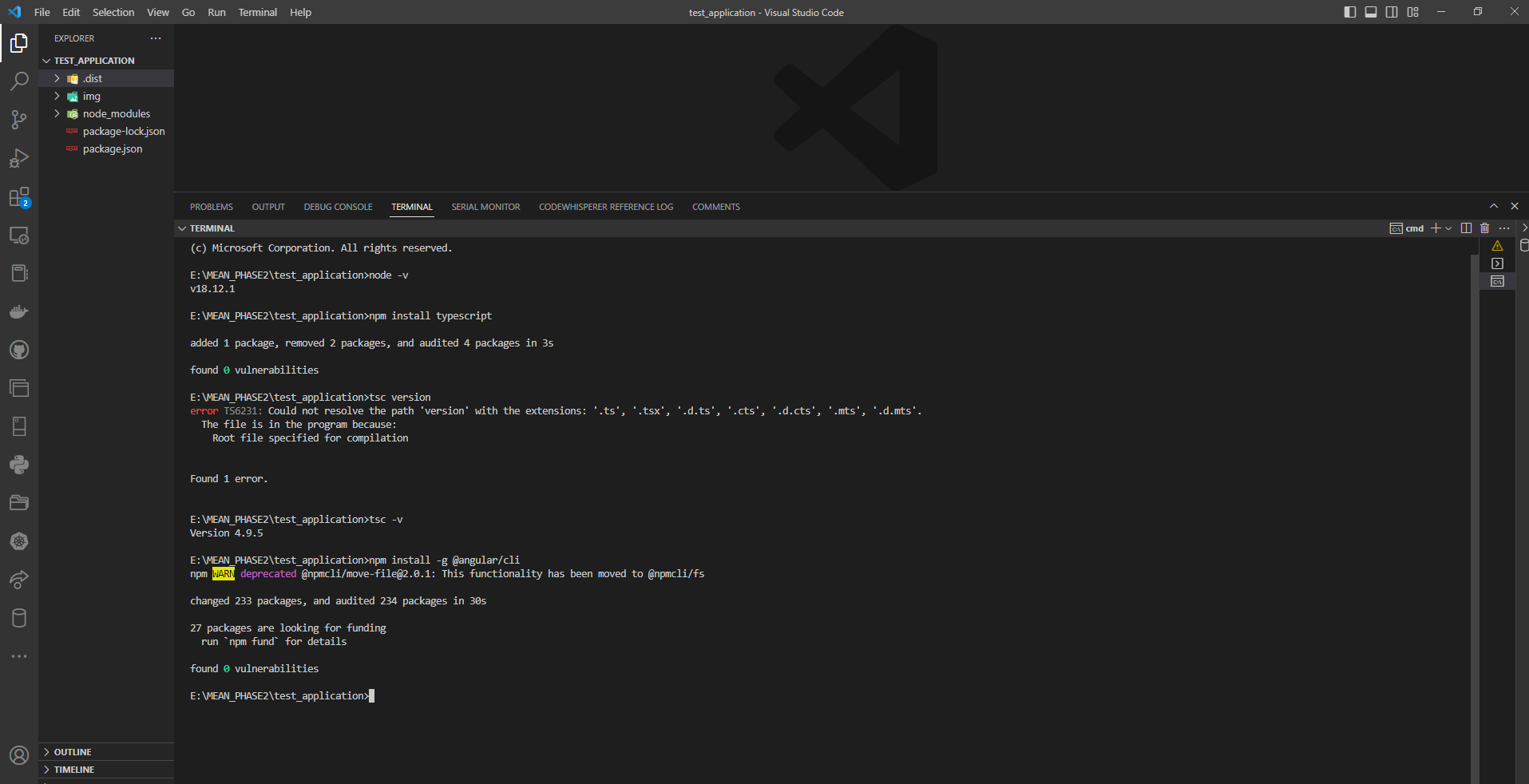
**Application development with nodejs, typescript, angular/cli, and bootstrap was brought up in the MEAn Phase2 of the development process. The following steps are documented to understand the process better.**

**Software components used in developing the test\_application:**

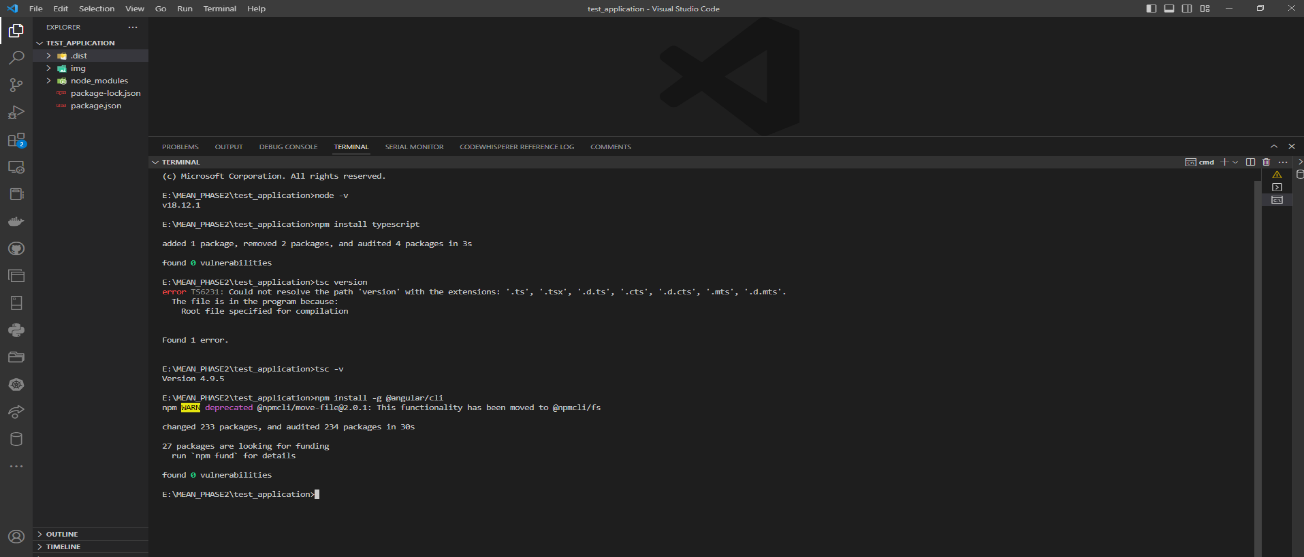
* **Visual studio**
* **Github**
* **node**
* **angular**
* **typescript**
* **bootstrap**

**2. The Guiding steps:**

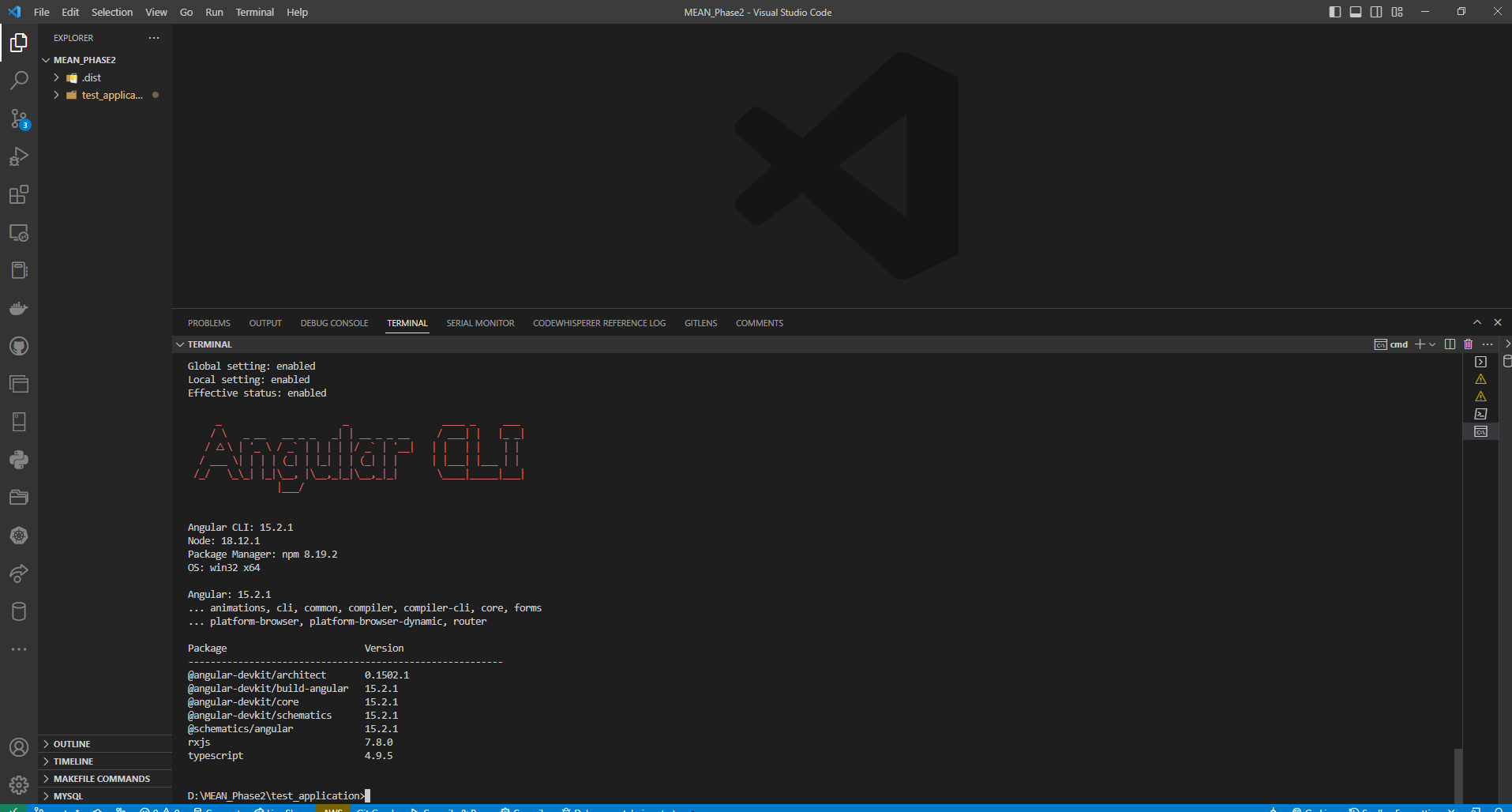
**Node.js and type script is installed in VSCode from terminal**



**Angular/cli is installed from terminal**



**To check for versions of installed APIs**



**The way angular CLI works:**

**Angular** is used for

* front end development

SPA

by **dynamically rewritting the current web page with new**

**data from web server ,instead of default method of a web browser loading**

**entire new pages.**

Major benefit of SPA

**1. Faster**

**2. Better user experience as it is faster**

* **Angular JS**

**supported only javascript**

**Angular complete rewrite of Angular JS**

* **Component based**

**creating reusable components,services,directives,pipes**

* **Installing angular->**

**npm install -g @angular/cli**

* **to check version**

**ng version**

**or**

**ng --version**

* **npm i -g @angular/cli@{version-number}**

**The way a new Project is created with angular is:**

* **ng new <project name>**
* **To run angular project->**
* **ng serve**

**or**

* **ng s**

**Steps followed to create an angular app with bootstrap:**

* **A folder named test\_application and is browsed in it in terminal**

**using cd test\_application**

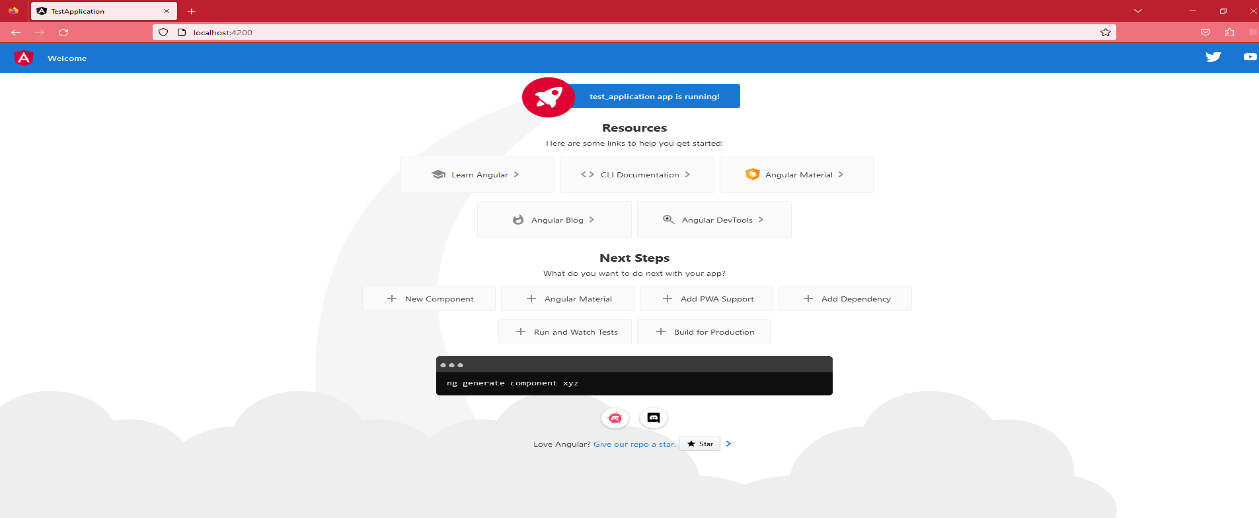
* **ng new test\_app\_ang**

**choose y and css**

**browse in it in terminal using cd test\_app\_ang**

* **npm install bootstrap**
* **ng serve**

**app.component.html**



* **In angular.json file, below line is added in styles array ->**

**"node\_modules/bootstrap/dist/css/bootstrap.min.css"**

**So, after adding above line, the style array will look like this->**

**"styles": [**

**"node\_modules/bootstrap/dist/css/bootstrap.min.css",**

**"src/styles.css"**

**]**

* **In app.component.html, below line is added**

**<app-header></app-header>**

* **ng serve**
* **main.ts**

**act as entry point of the angular app**

**helps in creating the browser environment for the application to run.**

* **platformBrowserDynamic**

**supports execution of Angular Apps on different supported browsers**

**bootstrapModule(AppModule)**

**it will boot or start AppModule**

* **selector**

**html file / template**

* **Components**

**Syntax to create component**

**ng g c <component name>**

* **Angular.json**

**primary configuration file for an angular project**

**contains settings of angular project**

**to look for all the paths and configurations and to check which is the main file**

* **Directives**

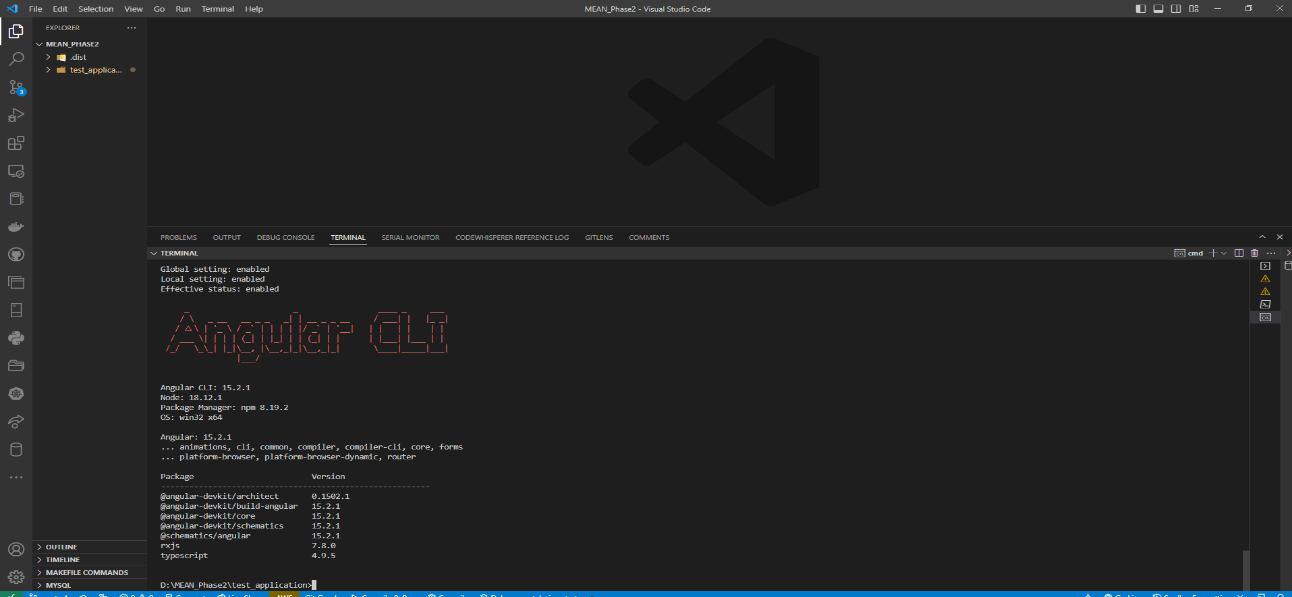
**ng**

* **reusable entity with predefined functionality**

**Type of directives->**

* **1Structural directive**
* **responible for HTML layout**
* **shape or reshape the html view by simply adding or removing HTML elements from the DOM**
* **is prefixed before the structural directive**
* **a)\*ngFor**
* **iterate over a collection of data**
* **Syntax**
* **\*ngFor = "let <value> of <collection>"**
* **<value> Variable name**
* **<collection> property on ur component which is collection of data**
* **b) \*ngIf**
* **evaluates a condition and based on the result further action will be taken**

**A test\_application folder is created:**



**The way data binding occurs in an application:**

**Binding data (in component's file=.ts file) and UI(.html file/template)**

**2 types**

**1. One way data binding**

**a) Interpolation**

**data flows from .ts to .html**

**{{propertyName}}**

**must result in a string**

**b) Property Binding**

**data flows from .ts to .html file**

**syntax**

**<html element> [DOM property]= "<component property>"**

**<element [property] ='typescript\_property'>**

**<span[innerHTML]='FirstName'></span>**

**<img [src]='imagePath'/>**

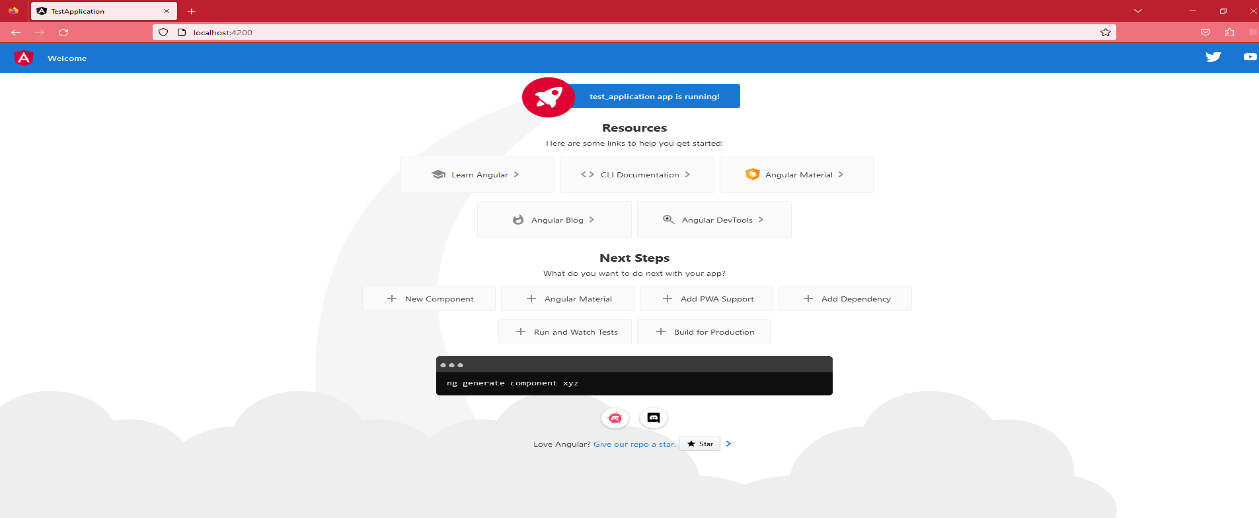
**<button [disabled]='Isdisabled'>Click me</button>**

**Event binding**

**even is any action performed by user**

**angular supports all events from javascript**

**a function to be executed when a certain event occurs**

**onClick ---->(click)** 

**Syntax**

**<element (event) = function()>**

**$event - contains information about an event**

**"any"**

**The first page [app.component.html**

**Components are created [ng g c <component-name>**

**within the application and compo nent.html of each are modified according to usage:**

**Creating responsive forms:**

**Creating browser bundles to allow the execution on the browser:**

**Components in Angular application encapsulates the template, data and behaviour of view.**

**Components are also known as View Components.**

**Every app has one component which is the root component.**

**Here in this case it is the app-component.**

{}

{}

{}

**A real world application encapsulates more than one component:**

**Navbar**

{}

{}

**In Angular a deeper nesting of components occur:**

Template

Parent Component

Nested Component

Event inout

Template

Input

Class

Class

**Deeper Nesting:**

Main app

B1

A1

B2

B3

A2

Angular Component Lifecycle  
A component has a lifecycle managed by Angular.  
• Angular creates the component and renders it, creates and renders its children,  
checks when its data-bound properties change, and destroys it before removing it  
from the DOM.  
• Angular offers lifecycle hooks that provide visibility to the key lifecycle moments of  
the component. It also provides the ability to act when these moments occur.  
• A directive has the same set of lifecycle hooks, excluding the hooks that are  
specific to component content and views.

**Pushing project to github:**

**The pushing in github repository had created refspec error after the first time. However it was overcome with effort.**

**Installing the service worker could not be effected in the current system.**

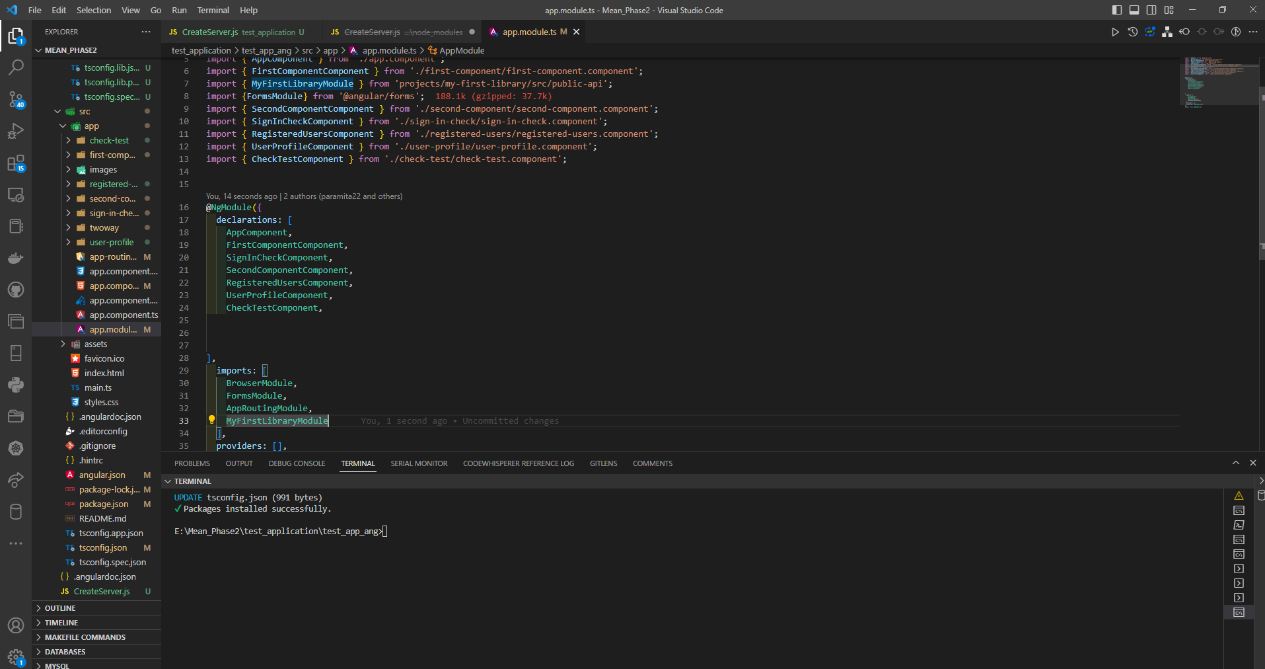
**Installing the globally enabled http server:**

**Creating projects with library:**

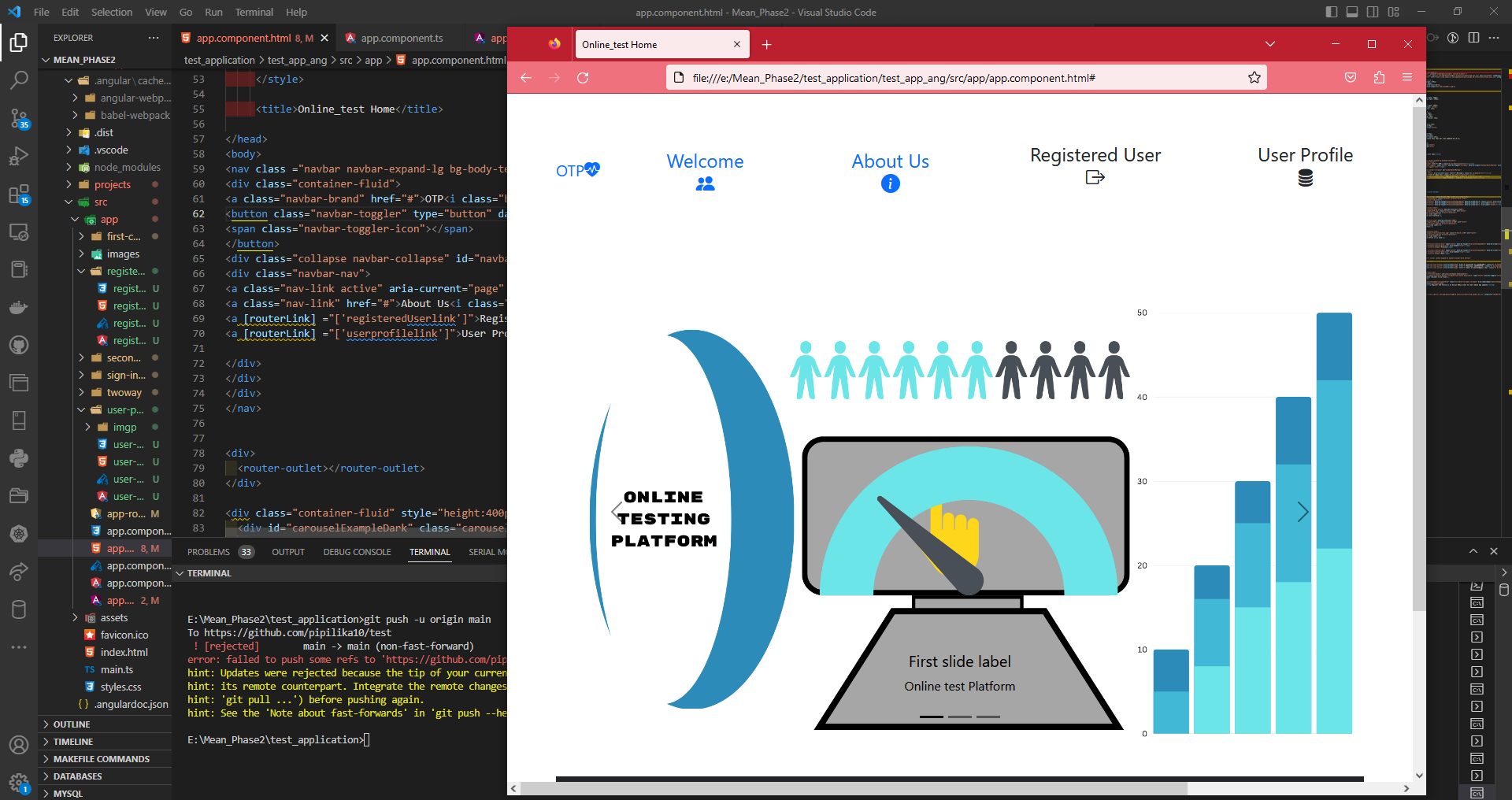
**app.module.ts**

**All Components are imported in the .ts file to ensure a smooth execution**

**All Components are put in the declaration array**



**The landing page of the test\_application:**



**pwa app and service component:**

**To make an app visible to users we deploy it in the web server that is accessible to https, the following are needed.**

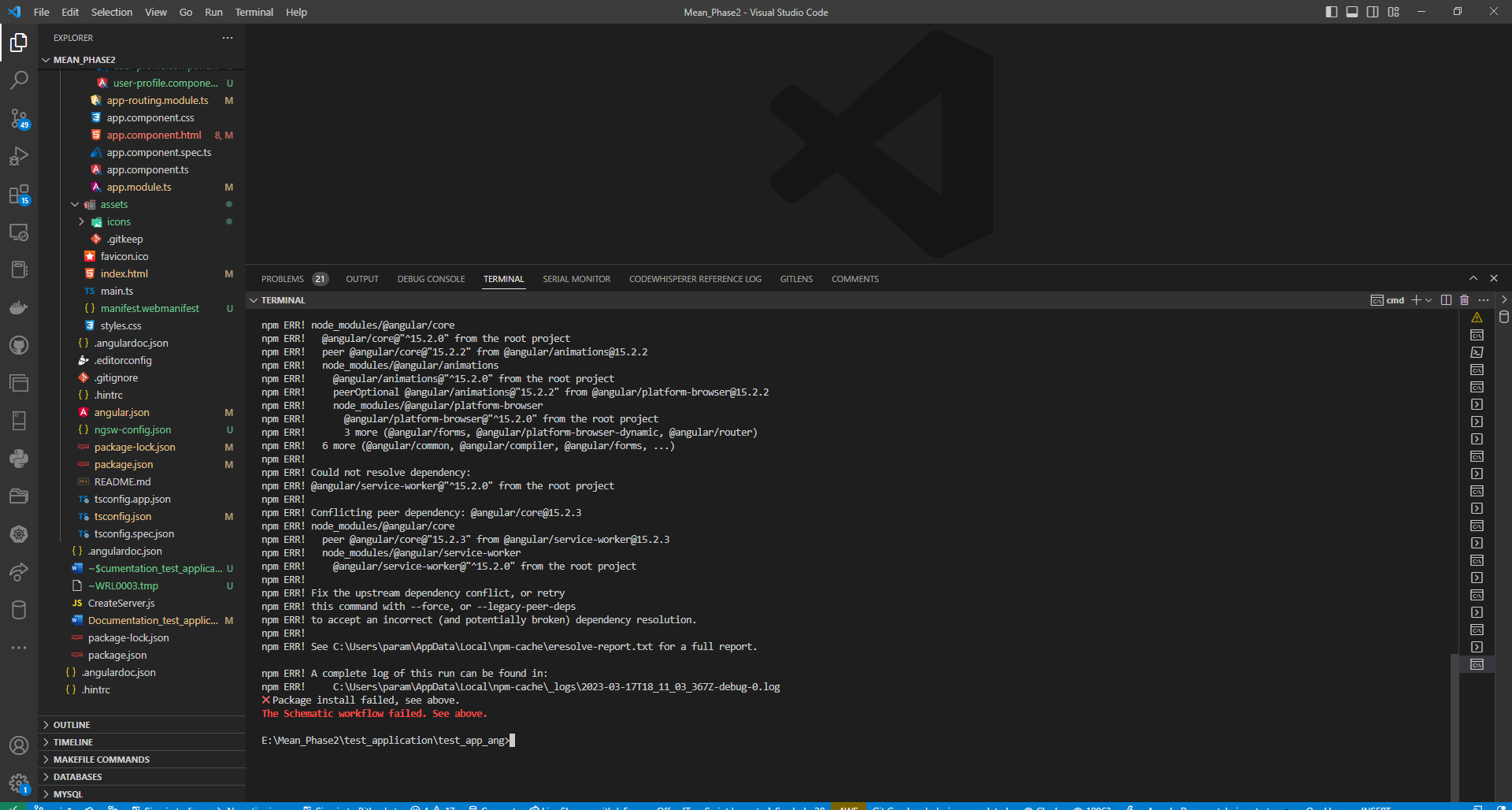
**Back-end: The app may use and access dynamically data stored in web server**

**Front-end: The resources needed for the app may be installed on the users device as html, and javascript.**

**Service workers are specialized that intercept network requests from your PWA and enable scenarios that were previously limited to native apps, including:**

* **Offline support.**
* **Advanced caching.**
* **Running background tasks such as receiving PUSH messages, adding badges to the app icon, or fetching data from a server.**

**@angular/pwa could not be installed . An installation failure occurred:**



The application could not implement the service worker for testing.