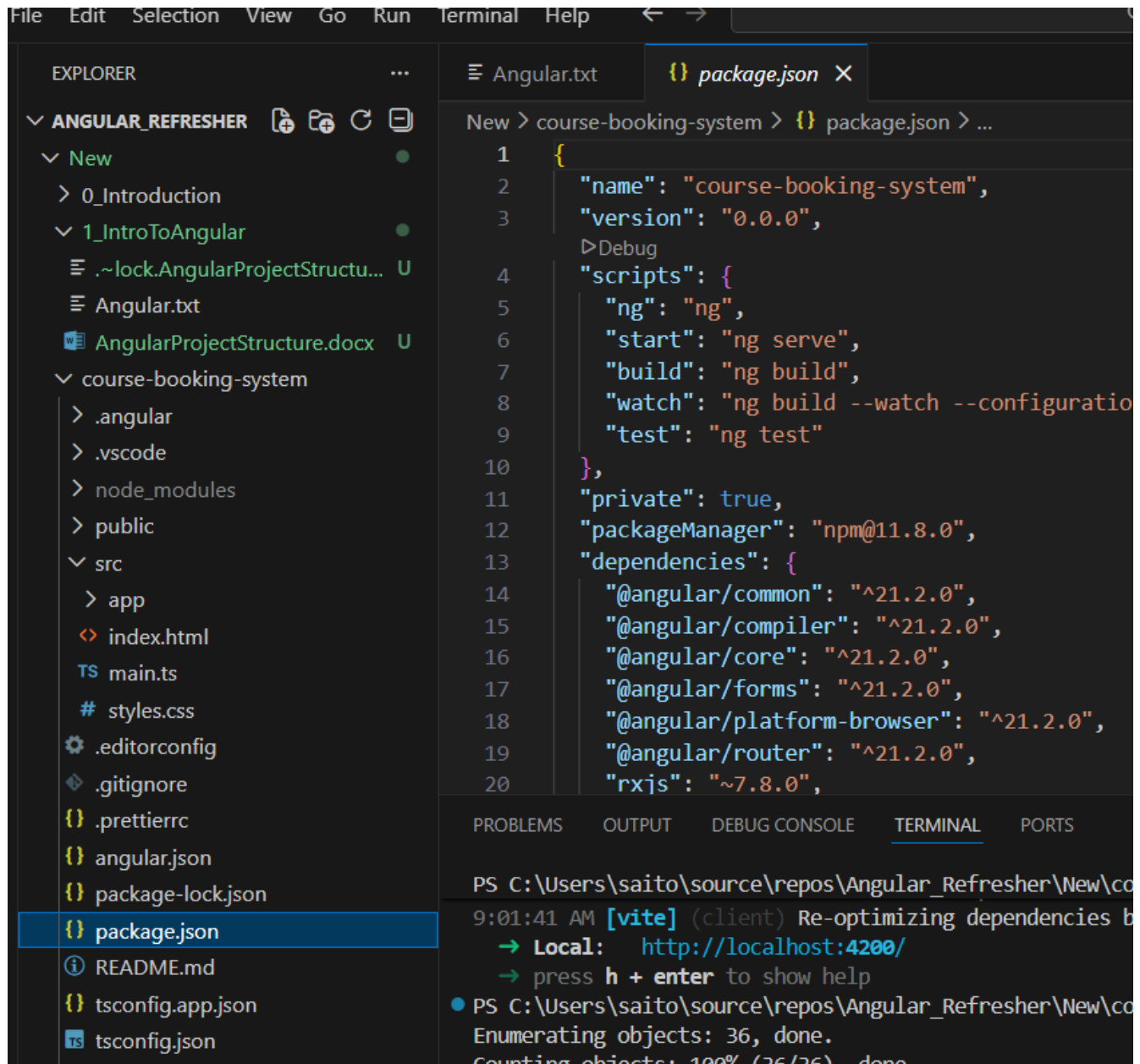


## Angular Project Structure

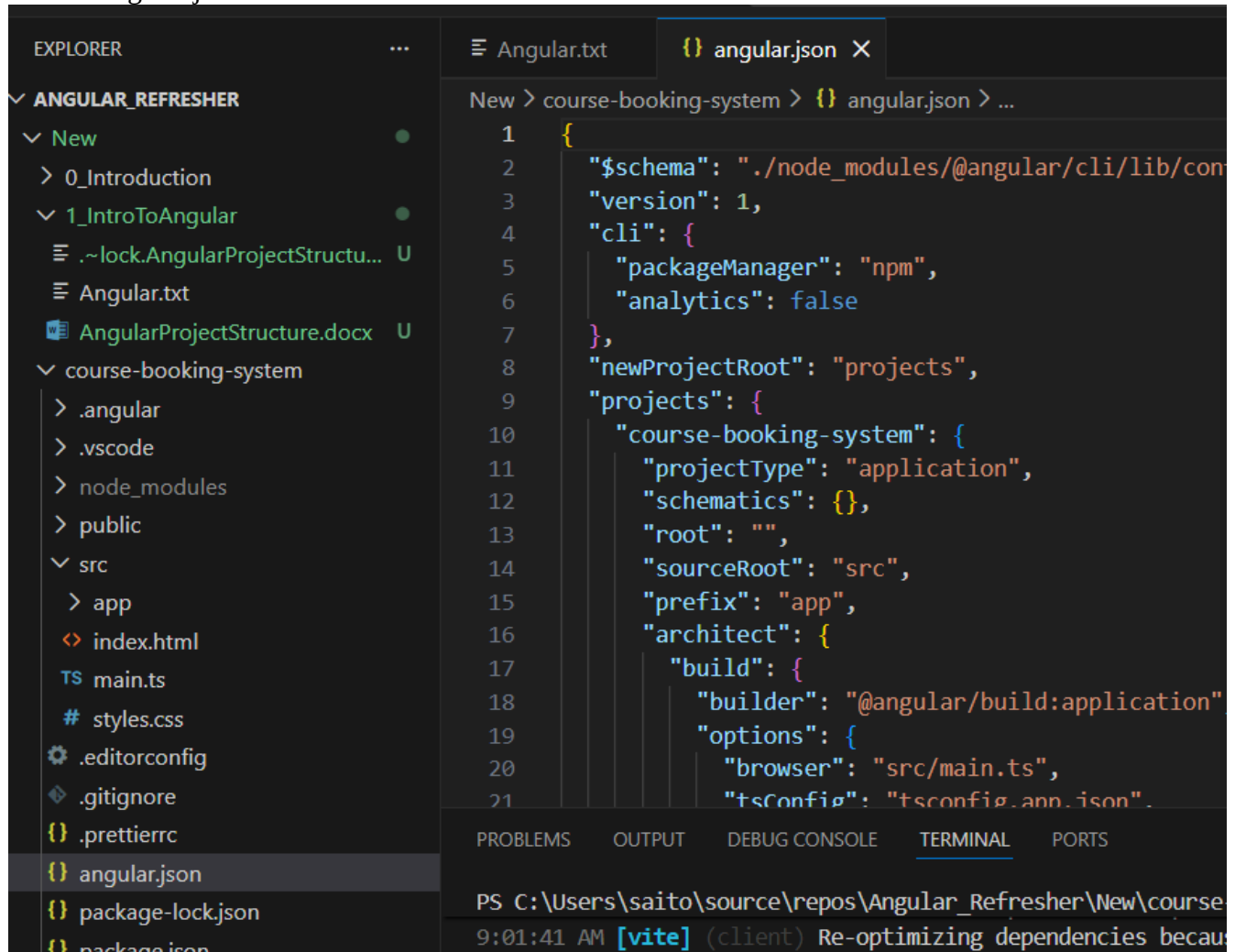


Lets start with top-level first: package.json. Heart of project dependencies

Lists all packages the app needs like Angular, TypeScript, Various libraries. They are downloaded into the node\_modules folder

node\_modules folder has a long list of files lot more than the package.json file. Thats because of the dependencies of the dependencies

Next is angular.json file



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure for 'ANGULAR\_REFRESHER' and 'course-booking-system'. The 'angular.json' file is selected in the Explorer. The main editor area shows the content of 'angular.json' with the following JSON structure:

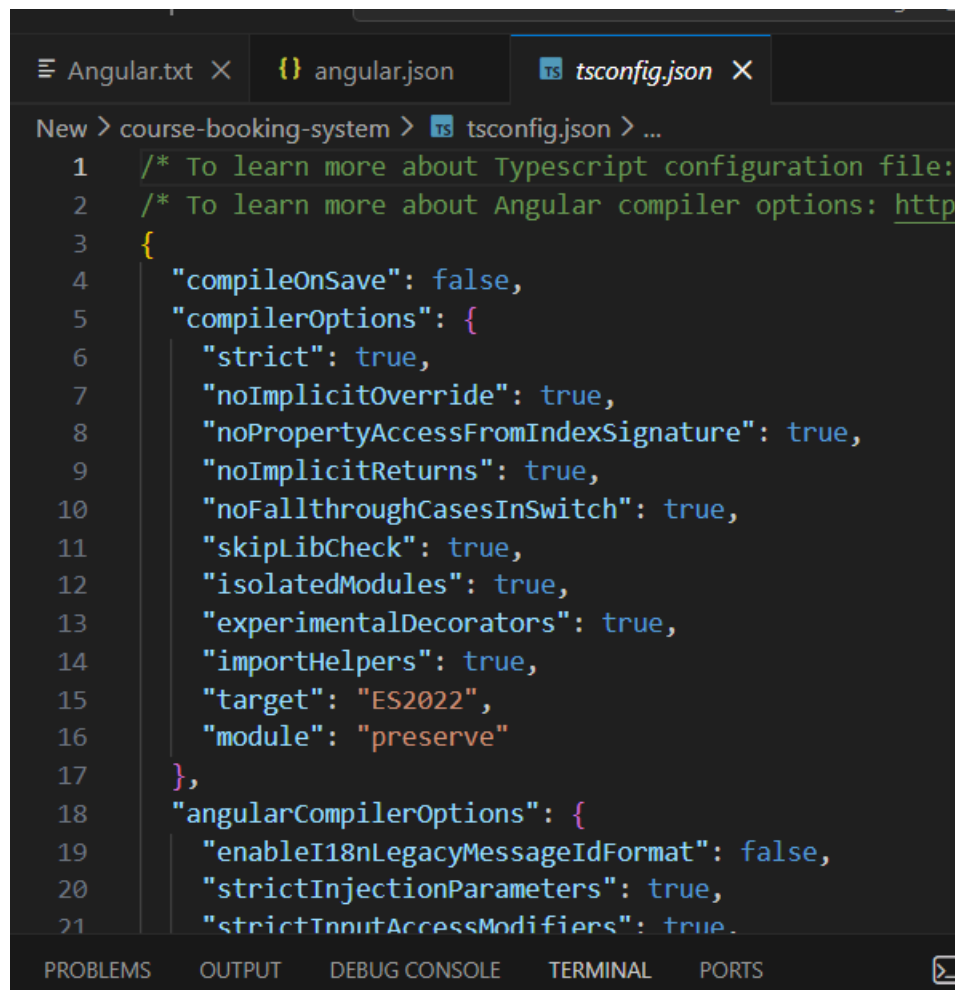
```
1 {
2   "$schema": "./node_modules/@angular/cli/lib/commands/ng.json",
3   "version": 1,
4   "cli": {
5     "packageManager": "npm",
6     "analytics": false
7   },
8   "newProjectRoot": "projects",
9   "projects": {
10     "course-booking-system": {
11       "projectType": "application",
12       "schematics": {},
13       "root": "",
14       "sourceRoot": "src",
15       "prefix": "app",
16       "architect": {
17         "build": {
18           "builder": "@angular/build:application",
19           "options": {
20             "browser": "src/main.ts",
21             "tsConfig": "tsconfig.app.json",

```

At the bottom, the Terminal panel shows the command prompt (PS) and the output of a Vite command: "9:01:41 AM [vite] (client) Re-optimizing dependencies because...".

angular.json => it has the CLI configuration, how our project is built, tested and includes settings for different environments and build options

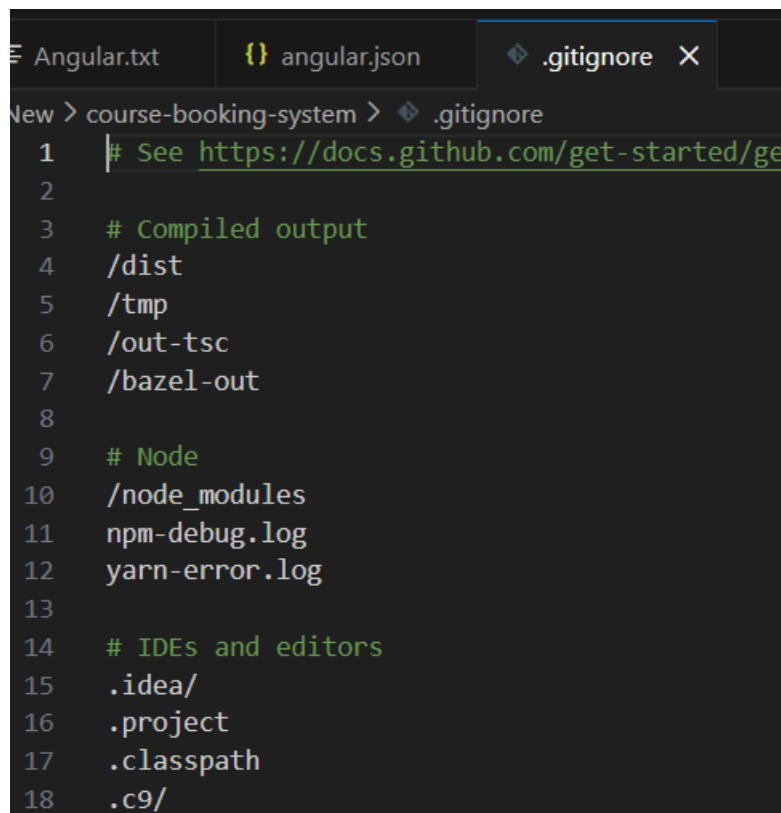
tsconfig.json



The screenshot shows the Visual Studio Code editor with three tabs: 'Angular.txt', 'angular.json', and 'tsconfig.json'. The 'tsconfig.json' tab is active, displaying the following TypeScript configuration file:

```
1  /* To learn more about Typescript configuration file:
2  /* To learn more about Angular compiler options: http
3  {
4      "compileOnSave": false,
5      "compilerOptions": {
6          "strict": true,
7          "noImplicitOverride": true,
8          "noPropertyAccessFromIndexSignature": true,
9          "noImplicitReturns": true,
10         "noFallthroughCasesInSwitch": true,
11         "skipLibCheck": true,
12         "isolatedModules": true,
13         "experimentalDecorators": true,
14         "importHelpers": true,
15         "target": "ES2022",
16         "module": "preserve"
17     },
18     "angularCompilerOptions": {
19         "enableI18nLegacyMessageIdFormat": false,
20         "strictInjectionParameters": true,
21         "strictInputAccessModifiers": true.
```

tsconfig.json => TypeScript configuration file, it sets rules for TypeScript compilation, supports code consistency. Helps with consistency and catching errors early on.

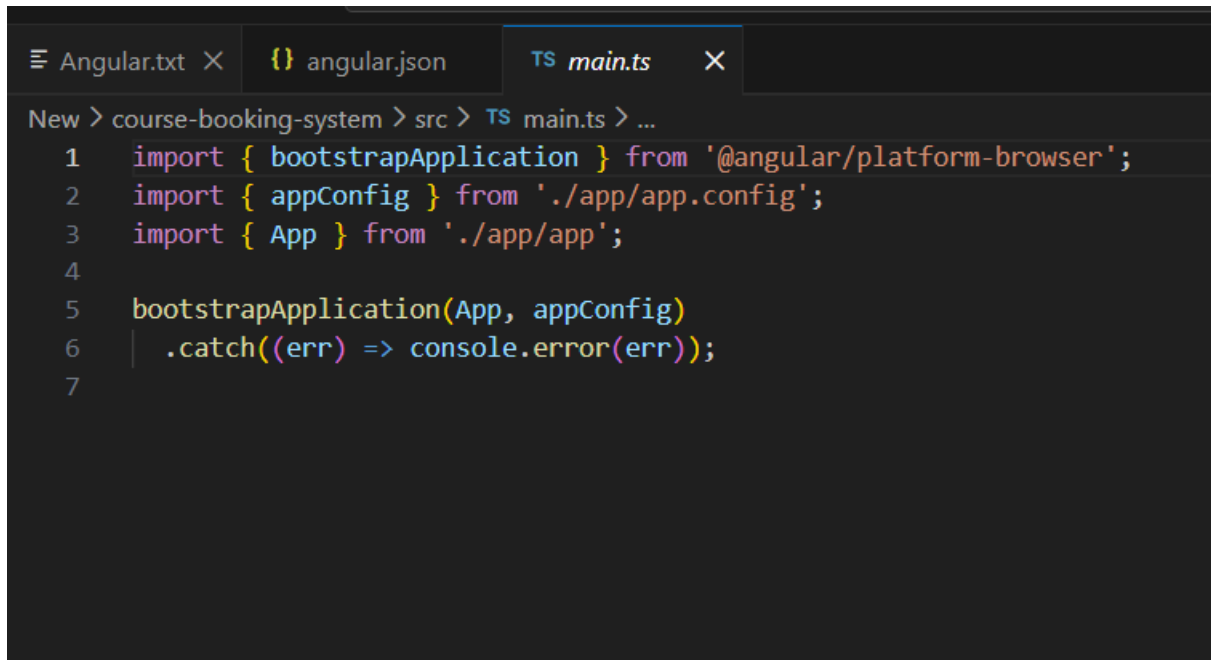


The screenshot shows the Visual Studio Code editor with three tabs: 'Angular.txt', 'angular.json', and '.gitignore'. The '.gitignore' tab is active, displaying the following content:

```
1  # See https://docs.github.com/get-started/ge
2
3  # Compiled output
4  /dist
5  /tmp
6  /out-tsc
7  /bazel-out
8
9  # Node
10 /node_modules
11 npm-debug.log
12 yarn-error.log
13
14 # IDEs and editors
15 .idea/
16 .project
17 .classpath
18 .c9/
```

.gitignore file tells Git which folders and files to ignore. It prevents unnecessary files like huge node\_modules folder or sensitive files containing environment secrets from being tracked in a version control.

/src/ folder where the magic happens. The application code lives here.  
main.ts

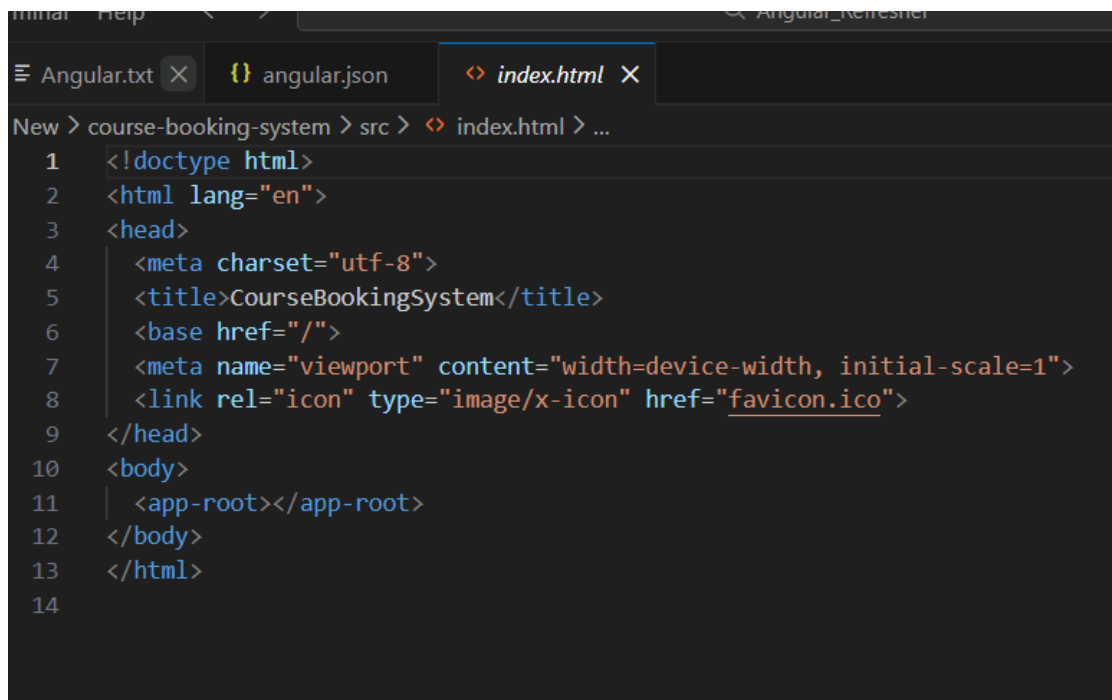


The screenshot shows a code editor with three tabs: 'Angular.txt', 'angular.json', and 'TS main.ts'. The 'main.ts' tab is active, displaying the following TypeScript code:

```
New > course-booking-system > src > TS main.ts > ...
1  import { bootstrapApplication } from '@angular/platform-browser';
2  import { appConfig } from './app/app.config';
3  import { App } from './app/app';
4
5  bootstrapApplication(App, appConfig)
6    .catch((err) => console.error(err));
7
```

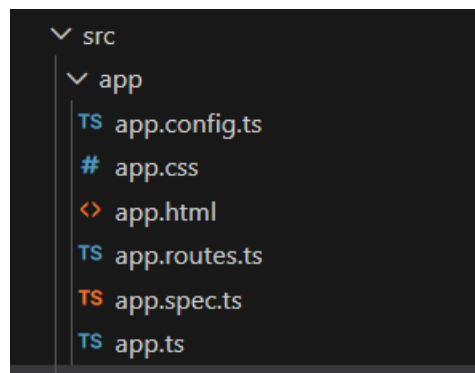
This is the entry point for our application. It bootstraps the Angular app, essentially kicking off the whole process.

index.html



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>CourseBookingSystem</title>
6   <base href="/">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="icon" type="image/x-icon" href="favicon.ico">
9 </head>
10 <body>
11   <app-root></app-root>
12 </body>
13 </html>
14
```

index.html is the main html file that Angular injects our App into. This determines what we will see on the page. Then styles.css, this file contains global styles for our app, affecting look and feel across all components. It's where we can setup color screens, typography, and overall design aesthetics. Asset components: they are in the 'app' folder. That's where we have the application components, services, and other core functionality resides

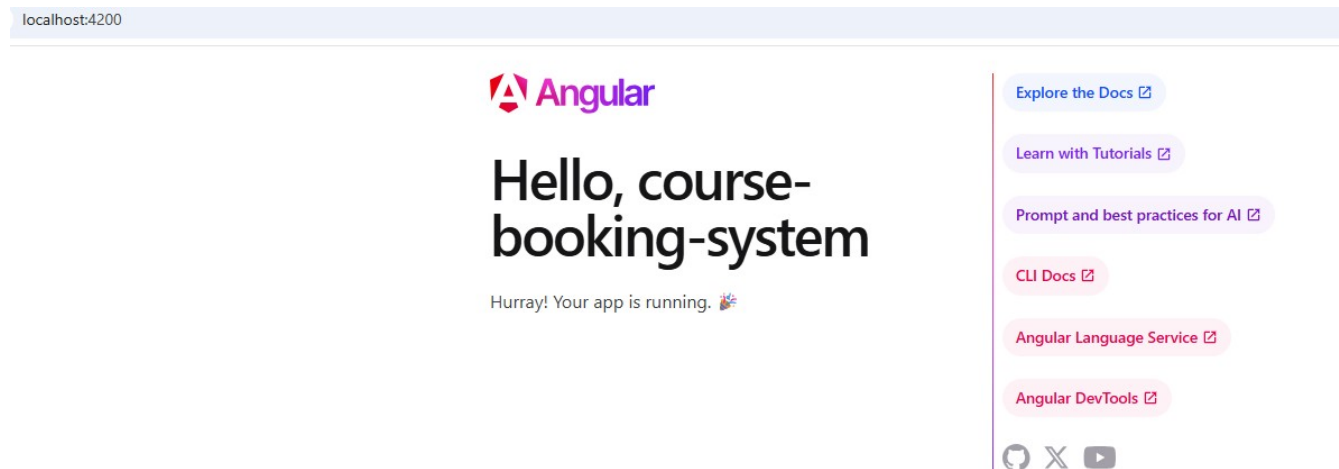


app component is in app.ts

```
Angular.txt X {} angular.json TS app.ts X
New > course-booking-system > src > app > TS app.ts > ...
1 import { Component, signal } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3
4 @Component({
5   selector: 'app-root',
6   imports: [RouterOutlet],
7   templateUrl: './app.html',
8   styleUrls: ['./app.css']
9 })
10 export class App {
11   protected readonly title = signal('course-booking-system');
12 }
13
```

it consists of a TypeScript file, HTML, a CSS file and a test file. This file defines the root component of our application. This is the first component that loads and serves as the parent for all other components.

We can see this component is injected into index.html `<app-root></app-root>`



app.routes.ts defines main navigation within our app. It defines how users move between different components and views

```
New > course-booking-system > src > app > TS app.routes.ts > ...  
1  import { Routes } from '@angular/router';  
2    
3  export const routes: Routes = [];  
4
```

'public' folder is the folder for aesthetics assets like images or other files that don't require processing by Angular. Good for storing assets

