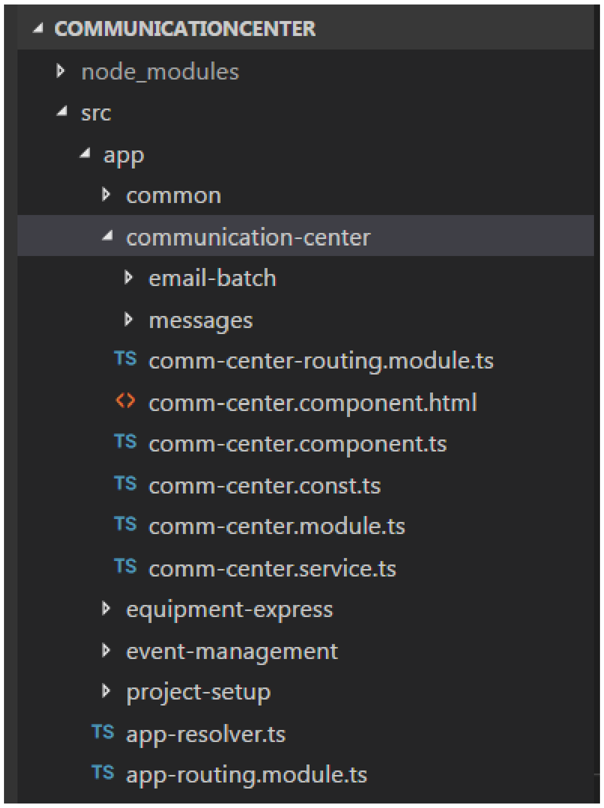
Angular Coding Standards

Typescript

- For folder/file name all characters should be small letter and it should be word separated by dashed-case "-" and all "component", "module" etc. should be separated by "." File name should be meaningful.
  Eg:
  

  ```
  ◢ COMMUNICATIONCENTER
    ▷ node_modules
    ◢ src
      ◢ app
        ▷ common
        ◢ communication-center
          ▷ email-batch
          ▷ messages
          comm-center-routing.module.ts
          comm-center.component.html
          comm-center.component.ts
          comm-center.const.ts
          comm-center.module.ts
          comm-center.service.ts
        ▷ equipment-express
        ▷ event-management
        ▷ project-setup
        app-resolver.ts
        app-routing.module.ts
  ```
- Variable should be in camel case
  Eg:
  ```
  private gridLength: number;
  ```
- Constant variable must be placed under *.const.ts files
- Constant variables should be in Upper case separated by _
  Eg:
  ```
  ccConstants.MESSAGES.PAGE_SIZE
  ```
- Default values should be stored in a const and be used in component.
  Eg:
  ```
  private pageSize: number = ccConstants.MESSAGES.PAGE_SIZE;
  ```
- Class Name should be in Pascal case
  Eg:
  ```
  export class MessageComponent implements OnInit {
  ```
- All API calls should be called only from services and NOT from Component. Component should only hold the reference to the service API method and call it.

- Every method should have a proper description Tag

```
    /**
    * @name getInputPaymentInformationData
    * @kind function
    *
    * @description
    * Get data for Payment Information
    */
    public getInputPackageInformationData(inputJsonPath): any {
        return this.http.get(inputJsonPath).map(result =>
result.json());
    }
```

- All methods should have a proper return type
  Eg:
  ```
  private pageChange(event: PageChangeEvent): void {
  ```

- All properties should be a strongly typed
  Eg:
  ```
  private gridLength: number;
  ```
- Angular lifecycle events should be placed at the end of the class.
  Eg: **ngOnit(), ngOnChanges()** etc.
- String should be enclosed with single quotes for *.ts and double quotes for *.html
- All conditional validation should have a === and not ==.
- Tab space should be 4 spaces
- Use dashed-case for naming the element selectors of components.
  Eg: selector: **'delivery-info'**
- Name events without the prefix on and name event handler methods with the prefix on followed by the event name.
  <!-- avoid -->
  @Output() onSavedTheDay = new EventEmitter<boolean>();
  <delivery-info (onSavedTheDay)="onSavedTheDay($event)"></ delivery-info >
  <!-- do-->
  @Output() savedTheDay = new EventEmitter<boolean>();
  < delivery-info (savedTheDay)="onSavedTheDay($event)"></ delivery-info >

P.S:

- ✓ Remove unwanted code
- ✓ Remove unused import statements
- ✓ Remove the commented code
- ✓ Remove debugger
- ✓ Install TsLint extension to identify all warnings. Remove all warnings from the application

HTML

- All elements which are displayed in UI should have an id mapped to it (Helpful for Automation scripts in identifying through xpath).
- Class attribute should be the last attribute of the element.
- String should be given using double quotes “” for *.html
- Id property should be using camel case
- Class name should be using dashed-case

  Eg:

  ```
  <div id="commonMainHeader" class="common-grid-header">
  ```
- All static elements should be retrieved from *.json of translate file and should be decorated with Translate pipe

  Eg:

  ```
  {{ "MESSAGE.ARCHIVE" | translate }}
  ```

<u>Installation Setup for Angular APP</u>

Get the latest version of ngApp from VSTS

npm install -g @angular/cli

npm install to install all packages (Note: Have latest npm version)

Use the angular cli dev web server:

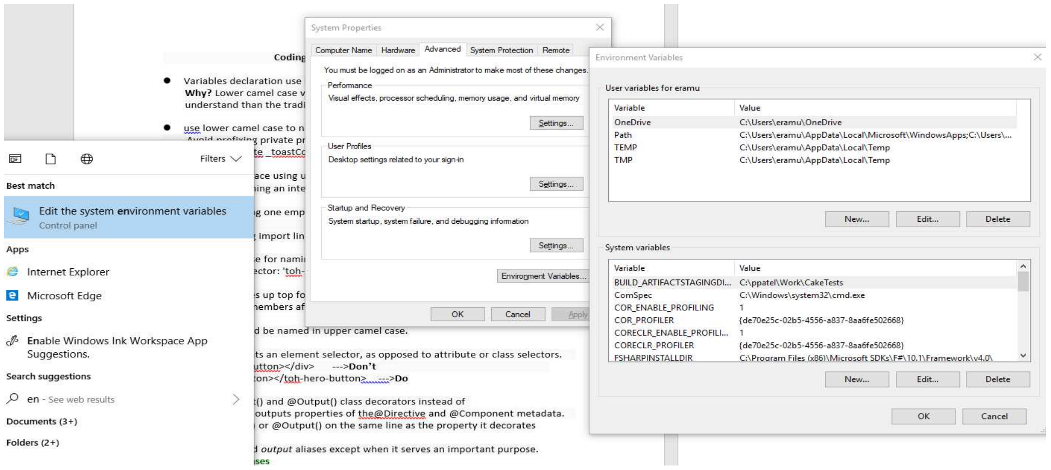
ng serve –open to run angular application

Note if we receive error as “ng is not recognized as the name of a cmdlet”, try this below comment

Powershell:

$env:PATH = "C:\Users\wsmith\AppData\Roaming\npm;C:\Program Files\nodejs\;" + $env:PATH

Or SET PATH in environment variable



You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

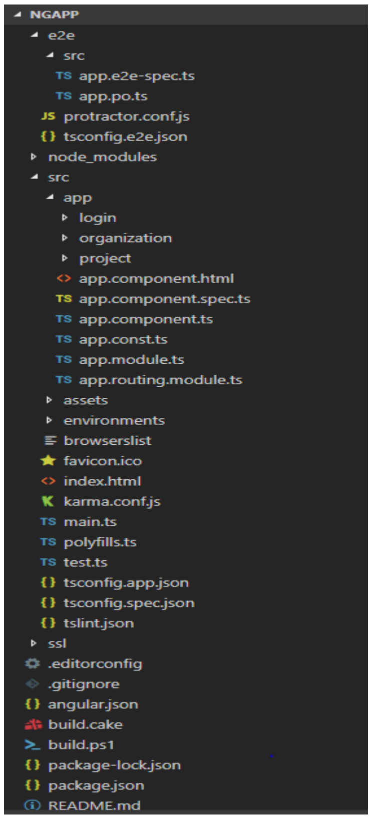
User variables for eramu

Variable	Value
OneDrive	C:\Users\eramu\OneDrive
Path	C:\Users\eramu\AppData\Local\Microsoft\WindowsApps;C:\Users\...
TEMP	C:\Users\eramu\AppData\Local\Temp
TMP	C:\Users\eramu\AppData\Local\Temp

System variables

Variable	Value
BUILD_ARTIFACTSTAGINGDI...	C:\ppatel\Work\CakeTests
ComSpec	C:\Windows\system32\cmd.exe
COR_ENABLE_PROFILING	1
COR_PROFILER	{de70e25c-02b5-4556-a837-8aa6fe502668}
CORECLR_ENABLE_PROFILI...	1
CORECLR_PROFILER	{de70e25c-02b5-4556-a837-8aa6fe502668}
FSHARPINSTALLDIR	C:\Program Files (x86)\Microsoft SDKs\F#\10.1\Framework\v4.0\

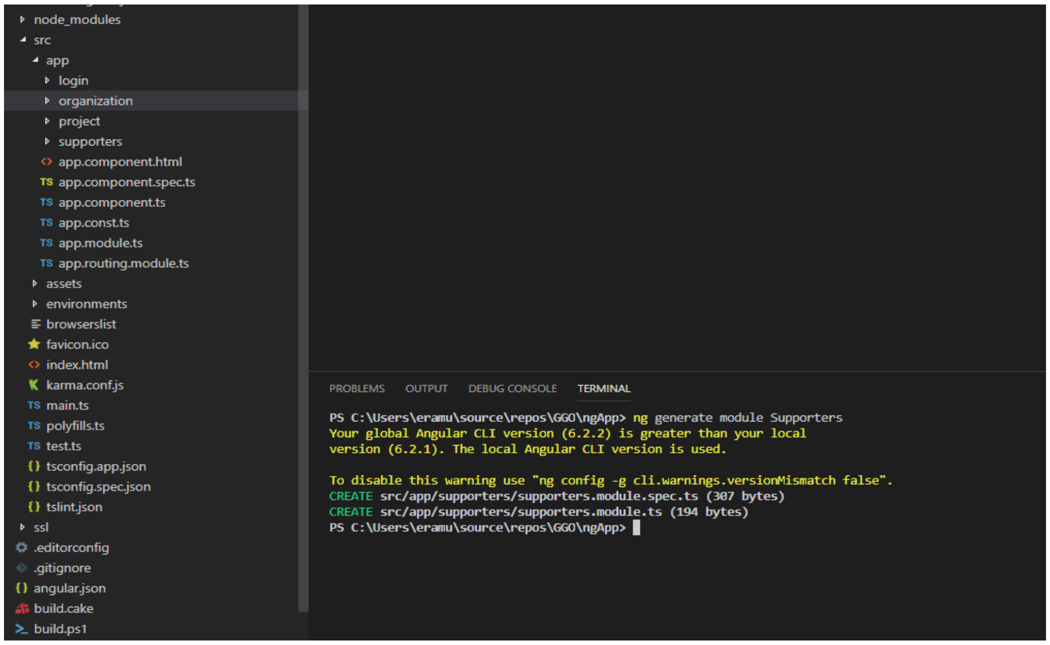
Open the NGAPP in VS code, as displayed below



```
NGAPP
  e2e
    src
      app.e2e-spec.ts
      app.po.ts
    protractor.conf.js
    tsconfig.e2e.json
  node_modules
  src
    app
      login
      organization
      project
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.const.ts
      app.module.ts
      app.routing.module.ts
    assets
    environments
    browserslist
    favicon.ico
    index.html
    karma.conf.js
    main.ts
    polyfills.ts
    test.ts
    tsconfig.app.json
    tsconfig.spec.json
    tslint.json
  ssl
  .editorconfig
  .gitignore
  angular.json
  build.cake
  build.ps1
  package-lock.json
  package.json
  README.md
```

We created the module login, Organization and projects

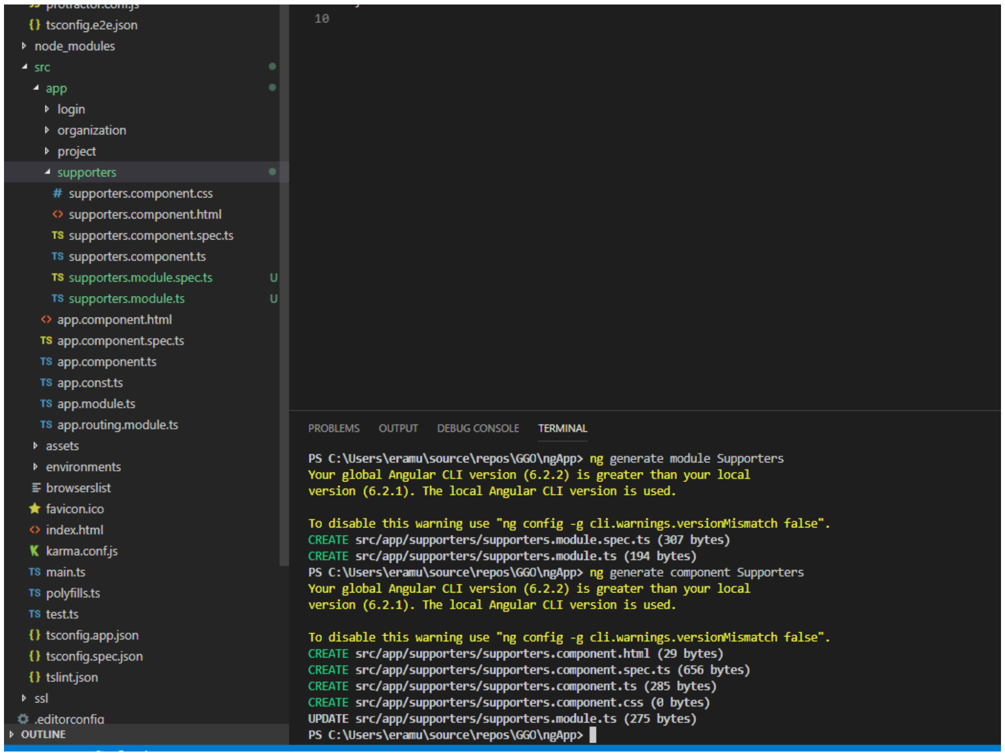
Using ng comment, we can create the module, components and test files



```
node_modules
src
  app
    login
    organization
    project
    supporters
    app.component.html
    app.component.spec.ts
    app.component.ts
    app.const.ts
    app.module.ts
    app.routing.module.ts
  assets
  environments
  browserslist
  favicon.ico
  index.html
  karma.conf.js
  main.ts
  polyfills.ts
  test.ts
  tsconfig.app.json
  tsconfig.spec.json
  tslint.json
ssl
.editorconfig
.gitignore
angular.json
build.cake
build.ps1
```

```
PS C:\Users\eramu\source\repos\GGO\ngApp> ng generate module Supporters
Your global Angular CLI version (6.2.2) is greater than your local
version (6.2.1). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".
CREATE src/app/supporters/supporters.module.spec.ts (307 bytes)
CREATE src/app/supporters/supporters.module.ts (194 bytes)
PS C:\Users\eramu\source\repos\GGO\ngApp>
```

ng generate component/module [file name]



```
PS C:\Users\eramu\source\repos\GGO\ngApp> ng generate module Supporters
Your global Angular CLI version (6.2.2) is greater than your local
version (6.2.1). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".
CREATE src/app/supporters/supporters.module.spec.ts (307 bytes)
CREATE src/app/supporters/supporters.module.ts (194 bytes)
PS C:\Users\eramu\source\repos\GGO\ngApp> ng generate component Supporters
Your global Angular CLI version (6.2.2) is greater than your local
version (6.2.1). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".
CREATE src/app/supporters/supporters.component.html (29 bytes)
CREATE src/app/supporters/supporters.component.spec.ts (656 bytes)
CREATE src/app/supporters/supporters.component.ts (285 bytes)
CREATE src/app/supporters/supporters.component.css (0 bytes)
UPDATE src/app/supporters/supporters.module.ts (275 bytes)
PS C:\Users\eramu\source\repos\GGO\ngApp>
```

Uses and implementation of Karma Jasmine for testing

(1) These libraries being specified in the command line that are required to work with karma.

Package.json

```
"karma": "^1.7.1",

"karma-chrome-launcher": "^2.2.0",

"karma-jasmine": "^1.1.1",

"karma-jasmine-html-reporter": "^0.2.2",

"karma-phantomjs-launcher": "^1.0.4",

"karma-sourcemap-loader": "^0.3.7",

"karma-typescript": "^3.0.13",

"karma-webpack": "^2.0.9",
```

(2) This file is required by karma.conf.js and loads recursively all the .spec and framework files.

(3) Within the configuration file, the configuration code is put together by setting module.exports to point to a function which accepts one argument: the configuration object.

Karma.conf.js

```
module.exports = function(config) {

config.set({

//...code

});

};
```

Base path that will be used to resolve all patterns (eg. files, exclude)

```
basePath: '../..',
```

List of test frameworks you want to use. Typically, you will set this to ['jasmine'], ['mocha'] or ['qunit'].

```
frameworks: [ 'jasmine' ],
```

List all of the files you want to include and exclude.

```
files: [
{ pattern: './src/app/**.spec.ts', watched: false }

],

exclude: [],
```

Preprocess matching files before serving them to the browser.

```
preprocessors: {

'./src/app/main.spec.ts': ['webpack', 'sourcemap']

},
```

Test results reporter to use. (eg. 'dots', 'progress')

```
reporters: ['progress'],
```

Will be used as the port when launching browsers.

```
port: 9876,
```

A list of browsers to launch and capture. When Karma starts up. (eg. Chrome, PhantomJS)

```
browsers: ['Chrome'],
```

How long will Karma wait for a message from a browser before disconnecting from it (in ms).

```
browserNoActivityTimeout: 20000000,
```

Redefine default mapping from file extensions to MIME-type.

```
mime: {

'text/x-typescript': ['ts','tsx']

},
```

If true, Karma captures browsers, runs the tests and exits.

```
singleRun: true,
```

How many browser should be started simultaneous

```
concurrency: Infinity,
```

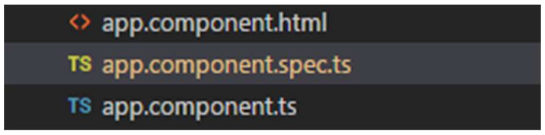
If this is set to true, the tests run in watch mode. If you change any test and save the file the tests are re-build and re-run.

```
autoWatch: true,
```

Set Webpack configuration, but set the entry to spec files

```
webpack: {

module: testWebpackConfig.module,

resolve: testWebpackConfig.resolve,

plugins: testWebpackConfig.plugins

}
```

(4) Every component having separate test file with extension .spec.ts


app.component.html
app.component.spec.ts
app.component.ts

(5) The describe function is used to give a description of the test. In our case, we are giving the description 'Sample test' to our test.

Sample test Case for app.component.spec.ts

```
describe('App Component unit test', function() {

 //...code

});
```

create the components in local variables.

```
let appComponent: AppComponent;

let fixture: ComponentFixture<AppComponent>;

let appService: AppService;

let translateService: TranslateService;
```

beforeEach function – This function is used to load our Angular JS module before the test run.

```
beforeEach(() => {

TestBed.configureTestingModule({

declarations: [ AppComponent, SlideoutMenuComponent ],

imports: [ RouterTestingModule, RestangularModule,
TranslateModule.forRoot()],
providers: [ AppService, SlideoutMenuService, Restangular ]

});


translateService = TestBed.get(TranslateService);

fixture = TestBed.createComponent(AppComponent);

appService = TestBed.get(AppService);

});
```

The it(string, function) function defines an individual Test Spec, this contains one or more Test Expectations.

```
it('Should return User name string', function() {

//--code

});
```

The spyOn is used for to set mock data for service methods.

```
spyOn(appService, 'getCurrentUser').and.returnValue(Observable.of({

'result': {

   'userName': 'User Name'

}

}));
```

appComponent.getCurrentUser is used to call getCurrentUser method in appComponent.

```
appComponent.getCurrentUser();
```

The expect(actual) expression is what we call an Expectation. In conjunction with a Matcher it describes an expected piece of behaviour in the application.

```
expect(appComponent.userName).toBe('User Name');
```

Jasmine comes with a few pre-built matchers like so:

expect(array).toContain(member);

expect(fn).toThrow(string);

expect(fn).toThrowError(string);

expect(instance).toBe(instance);

expect(mixed).toBeDefined();

expect(mixed).toBeFalsy();

expect(mixed).toBeNull();

```
expect(mixed).toBeTruthy();

expect(mixed).toBeUndefined();

expect(mixed).toEqual(mixed);

expect(mixed).toMatch(pattern);

expect(number).toBeCloseTo(number, decimalPlaces);

expect(number).toBeGreaterThan(number);

expect(number).toBeLessThan(number);

expect(number).toBeNaN();

expect(spy).toHaveBeenCalled();

expect(spy).toHaveBeenCalledTimes(number);

expect(spy).toHaveBeenCalledWith(...arguments);
```