UI elements

```
src/app/app.component.html
<div style="text-align:center">
  <h1>
  LocalCast Weather
  </h1>
  <div>Your city, your forecast, right now!</div>
  <h2>Current Weather</h2>
  <div>current weather</div>
</div>
```

At this point, you should run npm start and navigate to http://localhost:5000 on your browser so that you can observe the changes you're making in real time.

Adding an Angular component

- need to display the current weather information, where `<div>current weather</div>` is locatedDisplay forecast information for current location

- In the terminal, execute `npx ng generate component current-weather`

- Observe the new files created in your `app` folder:

```
src/app
├── app.component.css
├── app.component.html
├── app.component.spec.ts
├── app.component.ts
├── app.module.ts
├── current-weather
    ├── current-weather.component.css
    ├── current-weather.component.html
    ├── current-weather.component.spec.ts
    └── current-weather.component.ts
```

Creating issues for Local Weather app

- A generated component has four parts:

  - current-weather.component.css contains any CSS that is specific to the component and is an optional file

  - current-weather.component.html contains the HTML template that defines the look of the component and rendering of the bindings, and can be considered the View, in combination with any CSS styles used

  - current-weather.component.spec.ts contains Jasmine-based unit tests that you can extend to test your component functionality

  - current-weather.component.ts contains the @Component decorator above the class definition and is the glue that ties together the CSS, HTML, and JavaScript code together. The class itself can be considered the ViewModel, pulling data from services and performing any necessary transformations to expose sensible bindings for the View, as shown as follows:

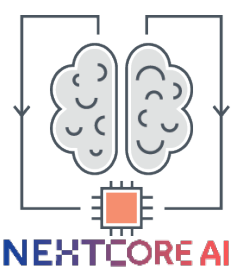# Creating issues for Local Weather app

```
src/app/current-weather/current-weather.component.ts
import { Component, OnInit } from '@angular/core'
@Component({
  selector: 'app-current-weather',
  templateUrl: './current-weather.component.html',
  styleUrls: ['./current-weather.component.css'],
})
export class CurrentWeatherComponent implements OnInit {
  constructor() {}

  ngOnInit() {}
}
```



```
src/app/current-weather/current-weather.component.ts
import { Component, OnInit } from '@angular/core'
@Component({
  selector: 'app-current-weather',
  templateUrl: './current-weather.component.html',
  styleUrls: ['./current-weather.component.css'],
})
export class CurrentWeatherComponent implements OnInit {
  constructor() {}

  ngOnInit() {}
}
```

- Update CurrentWeatherComponent with an inline template and styles:

CurrentWeatherComponent

src/app/current-weather/current-weather.component.ts

import { Component, OnInit } from '@angular/core'

@Component({
  selector: 'app-current-weather',
  template: `
  <p>
    current-weather works!
  </p>
  `,
  styles: [' ']
})
export class CurrentWeatherComponent implements OnInit {
constructor() {}

ngOnInit() {}
}

```
src/app/current-weather/current-weather.component.ts
import { Component, OnInit } from '@angular/core'

@Component({
  selector: 'app-current-weather',
  template: `
  <p>
    current-weather works!
  </p>
  `,
  styles: ['']
})
export class CurrentWeatherComponent implements OnInit {
constructor() {}

ngOnInit() {}
}
```

```
src/app/app.module.ts
...
import { CurrentWeatherComponent } from './current-weather/current-weather.component'
...
@NgModule({
declarations: [AppComponent, CurrentWeatherComponent],
...
```
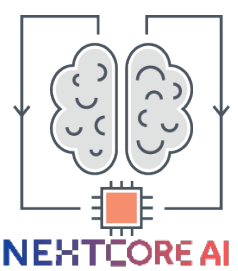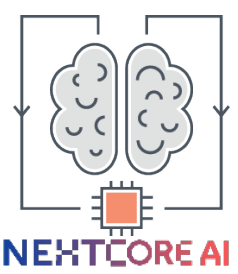
AppComponent

- Add the CurrentWeatherComponent to AppComponent by replacing `<div>current weather</div>` with `<app-current-weather></app-current-weather>`:

src/app/app.component.html
```
<divstyle="text-align:center">
<h1>
 LocalCast Weather
 </h1>
 <div>Your city, your forecast, right now!</div>
 <h2>Current Weather</h2>
 <app-current-weather></app-current-weather>
</div>
```

Initial render of app