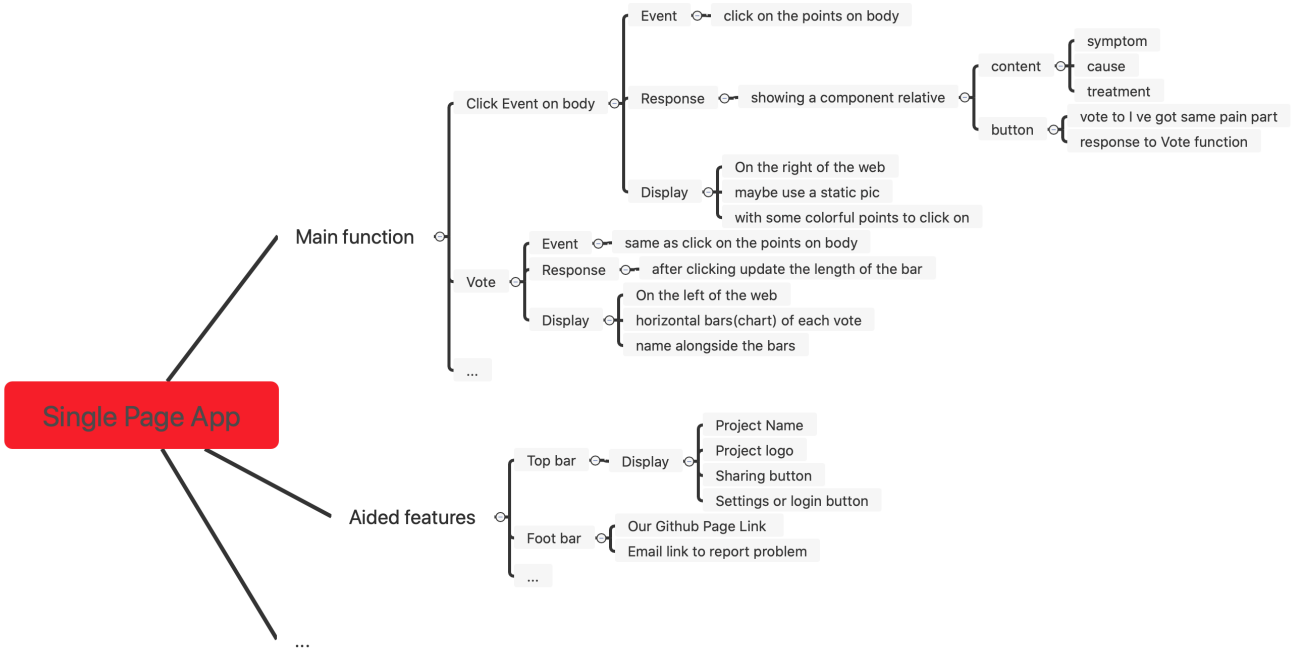
Title:

Front end Part of the whole porject.

Outline:

```
1. Main framework: Angular.
    1. Angular Introduction
    2. Front end Introduction
    3. Balance Between those two
2. Functionalities of our project
    1. body click
    2. vote
    3. comments
3. Layout of the single page web application.
  1. bootstrap
  2. div
  3. table
4. Features  implementation
    1. positions
    2. components
    3. codes
```



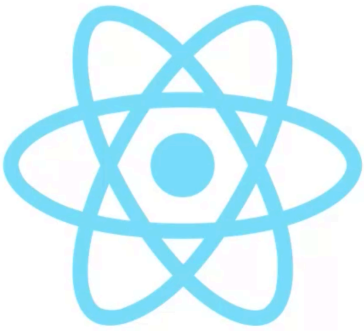
Content:

In this project, we will learn and use MEAN stack which is a development toolkit for creating single page applications. The whole project is aiming at practice of full stack development and MEAN is the abbrevation of four development tools, Mongo Database for data storage, Express web application framework, Angular Javascript web frontend, Node.js application server. The third part of that, 'A', Angular is going to be used as the framework of the frontend for the project. There's going to be a model from front to back, where frontend is in charge of display all the content we want to show to the users, not only the pure data and characters, but also the decoration and style pattern, apart from this, it also provides the features of interactions between users and the web page, as well as data transport from database behind backend to the components on the top of the frontend. In this part of documentation, we will be focusing on the frontend disign and implementation, including appearence and data, and some of the connection to the back end.

1 . Main framework: Angular.

Angular is an application design framework and development platform for creating efficient and sophisticated sigle-page apps.

There are some frameworks at the same level as Angular, like React, Angular uses the browser's DOM, while React uses the virtual DOM. Virtual DOM is a simplified version of DOM. By using the virtual DOM, you can change any element very quickly without having to render the entire DOM. It changes performance from excellent to excellent.



Feature	Angular package	React library
Data binding, dependency injection (DI)	@angular/core	MobX
Computed properties	rxjs	MobX
Component-based routing	@angular/router	React Router v4
Material design components	@angular/material	React Toolbox
CSS scoped to components	@angular/core	CSS modules
Form validations	@angular/forms	FormState
Project generator	@angular/cli	React Scripts TS

Angular the a pretty integrated and mature framework, wihch means it is more powerful but tough to get started for freshmen. We are going to use some basic features from Angurlar 11, which is stable and useful, supported and developed by Google. With the help of that, many processes could be simplified a lot no matter which stage we are. We make using angular easier by making use of Angular CLI tool, representing Command Line Interfaces, in some basic example we know that it could help create folders and files, put the template content into them and setup the configuration file and environment well automatically, which is helpful and friendly to new users but not good for new developers.

1.1 Angular Introduction

We are going to be using Angular for our client side development. Angular is used as a cross-platform tool, constructing applications with it makes codes can be used on different platform such as web, mobile web, mobile app, native app and desktop app. And Angular has the highest speed and performance on web platform because it renders bu web worker and service, as well as high scalability based on RxJS Immutables.js and other modules which suitable for huge amount data. Developers and fans of Angular make this community active and vibrant, i.g. components, templates, toolkits, packages, pulgins in IDE, etc. As a result, a virtuous circle has been formed, in which more and more useful tools can appear, and developers can easily use them.

There are some prerequisites. To use the Angular framework, you should be familiar with the following: HTML, CSS, Javascript. Knowledge of TypeScript is helpful, but not required.

To install Angular on your local system, we need the following: Node.js and npm package manager.

To install the Angular CLI, which will help us create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

To install the Angular CLI, open a terminal window and run the following command:

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

In this chapter, I only mention it to here. In the following chapters, I will explain in detail how to build an angular application from scratch.

I have to mention a concept here, that is, the angular workspace and project file structure.
When we create a new project with angular cli, we will find that there will be a folder named after the project in the current directory. Open this folder, you will find that some basic files needed by the front end of the project already exist, and cli helped us create them. We need to make some appropriate modifications, additions and deletions to it.
It contains some configuration type files, source code files, dependent files, test files, etc.
Such as angular.json package.json src node_modules tsconfig.json and so on.

IMAGE: workspace configuration files

WORKSPACE CONFIG FILES	PURPOSE
`.editorconfig`	Configuration for code editors. See EditorConfig.
`.gitignore`	Specifies intentionally untracked files that Git should ignore.
`README.md`	Introductory documentation for the root app.
`angular.json`	CLI configuration defaults for all projects in the workspace, including configuration options for build, serve, and test tools that the CLI uses, such as TSLint, Karma, and Protractor. For details, see Angular Workspace Configuration.
`package.json`	Configures npm package dependencies that are available to all projects in the workspace. See npm documentation for the specific format and contents of this file.
`package-lock.json`	Provides version information for all packages installed into `node_modules` by the npm client. See npm documentation for details. If you use the yarn client, this file will be yarn.lock instead.
`src/`	Source files for the root-level application project.
`node_modules/`	Provides npm packages to the entire workspace. Workspace-wide `node_modules` dependencies are visible to all projects.
`tsconfig.json`	The base TypeScript configuration for projects in the workspace. All other configuration files inherit from this base file. For more information, see the Configuration inheritance with extends section of the TypeScript documentation.
`tslint.json`	Default TSLint configuration for projects in the workspace.

IMAGE: Application source files

APP SUPPORT FILES	PURPOSE
`app/`	Contains the component files in which your application logic and data are defined. See details below.
`assets/`	Contains image and other asset files to be copied as-is when you build your application.
`environments/`	Contains build configuration options for particular target environments. By default there is an unnamed standard development environment and a production ("prod") environment. You can define additional target environment configurations.
`favicon.ico`	An icon to use for this application in the bookmark bar.
`index.html`	The main HTML page that is served when someone visits your site. The CLI automatically adds all JavaScript and CSS files when building your app, so you typically don't need to add any `<script>` or `<link>` tags here manually.
`main.ts`	The main entry point for your application. Compiles the application with the JIT compiler and bootstraps the application's root module (AppModule) to run in the browser. You can also use the AOT compiler without changing any code by appending the `--aot` flag to the CLI `build` and `serve` commands.
`polyfills.ts`	Provides polyfill scripts for browser support.
`styles.sass`	Lists CSS files that supply styles for a project. The extension reflects the style preprocessor you have configured for the project.
`test.ts`	The main entry point for your unit tests, with some Angular-specific configuration. You don't typically need to edit this file.

1.2 Front end Introduction

As mentioned above, as a preparatory work, you need some foundation to learn and use angular, which includes html css and JavaScript.
In fact, the above three main knowledge are enough to complete the design and development of a web page, but if you want to run it completely, including the entire process of data from the server back-end database to the front-end, as well as the front-end page interacting with users Relying on some packages and tools is very necessary, only when writing some time-consuming, relatively simple pages, and relatively balanced static and dynamic, only use the above three skills. But if we want to make our application not only static pages, animations, interactions, not just aesthetically simple, display boring single pages, there is little

interaction, there is no data flow page, you still need to be other Understand the use of tools. However, these three must be the foundation, that is to say, the three are fully proficient in order to master other tools faster. The tool only saves time and enhances the effect, but the basic principles are the same.
Here is a simple explanation. We are responsible for the front-end. Before using angular, we have completed the basic learning of the above three knowledge by ourselves. HTML builds the structure of the entire web page, CSS typesets the structure to make it beautiful, and Javascript is the key to truly allowing users to communicate with the page, enough to make the page move and data flow.

1.3 Balance Between those two

Therefore, as a beginner in web page development, it is very important to be able to find a balance between the two points. Why is it important? The important thing is how to make a decent application in a short period of time, and also have a certain learning and understanding of full-stack development. This is a challenge.
One of the first points is that, starting from the basics, understand the principles and syntax from the bottom. The second point is to start directly from the angular framework to simplify the development process and save development time. The advantages and disadvantages of the two are very obvious. The former is simple and easy to learn, but it is far away from the real development, and the results produced are also very simple. If you don't need a lot of time to research, it is difficult to go deep and gain results. Create a powerful and beautiful interface. As for the latter, if you are eager to achieve success and have not a solid foundation, you will be blindfolded during the development process. Many things do not understand why you want to do this, and you don't know how to solve the problem. This is what we have summed up in actual learning and development.
Therefore, combined with the lecture provided by the teacher in the web chapter of the software tools classroom, we are committed to finding a kind of front-end development, and finding a balance between the old technology and the mature framework. Use the corresponding technology in the right and appropriate place to improve the high flexibility of the project and shorten the learning time, and smoothly complete the transition from the ignorant to the familiar.

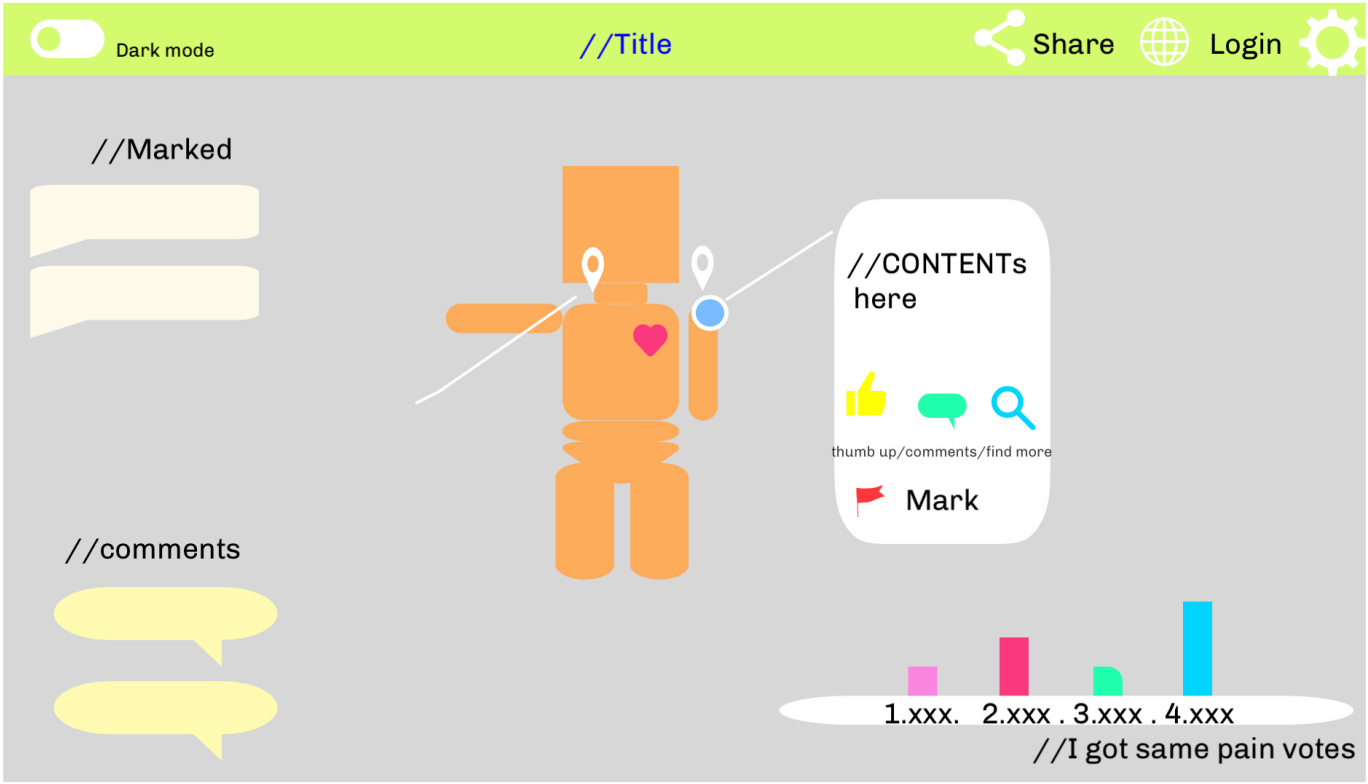
Examples:

Here is a brief list of two practical cases, but also to the following paragraph of the drama. First, why don't we use bootstrap, but directly use div tags and css statements to divide the page structure. Because our demand for bootstrap is not high, our need is simply to divide the page area, without taking into account the display on devices that are not used. Because of the precise operation of the mouse in our functional design, we didn't expect the application to run on a small-screen device during the development phase, but one of the features of bootstrap is the adaptively changing layout. We can do it in other ways, and it's just in line with the pace of learning, and of course, we don't deny that if we do it later, we'll choose bootstrap, just say that the result of the current balance is deprecation. In the second example, we added a script tag to the index .html file, and there are two reasons why the scripting language is not completely in the .ts file. One is that large-length scripts placed in ts files are not conducive to debugging at development time and can be falsely reported due to syntax limitations. Second, the function is actually different, it seems to be a mix of script tags and ts files, not standardized, but in fact, the response to special events pulled out, is organized. Of course, there are improvements, you can say that the statements in the script tag in the external file, replaced by the external introduction of script types, more regular.

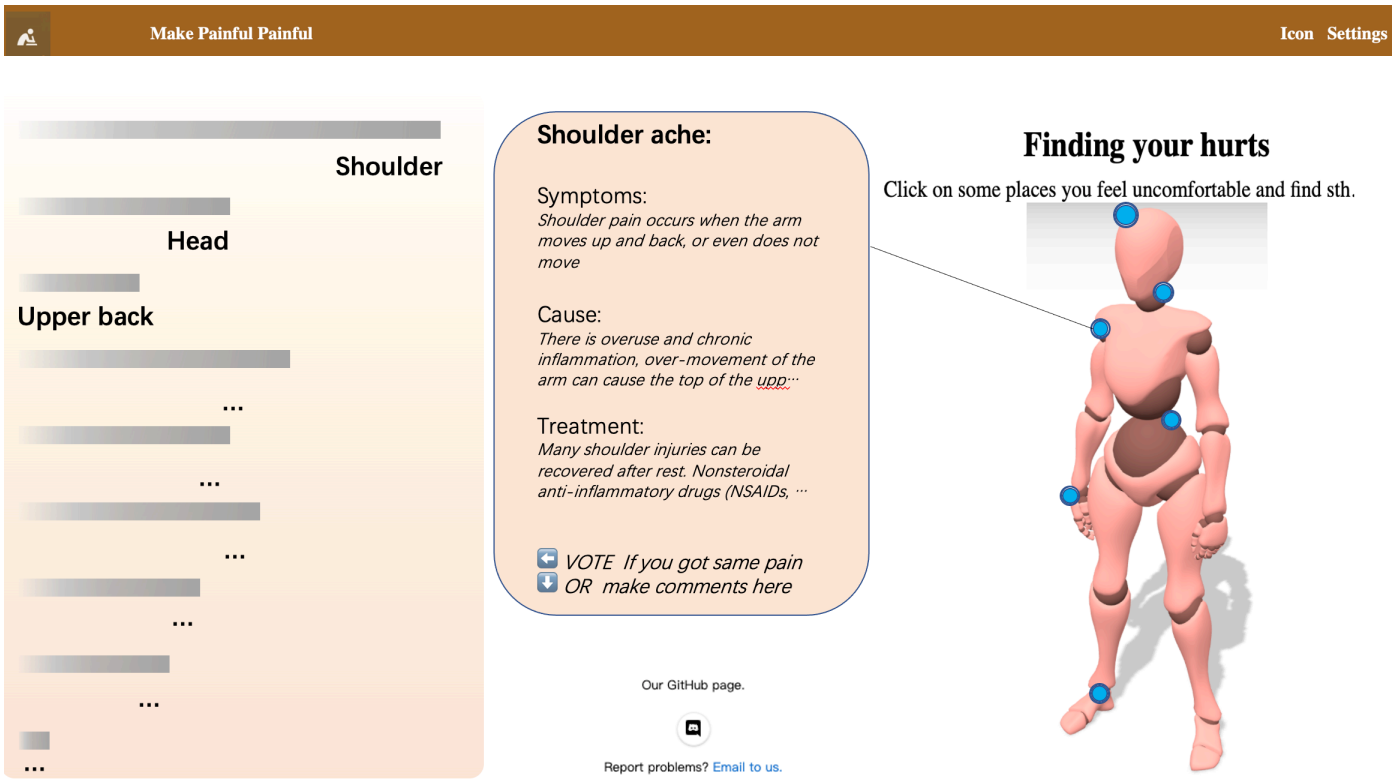
2 . Functionalities of our project

In this section, I'm going to cover the main features that our application will implement, because the determination of functionality doesn't fall into the front-end category, but it's relevant, and I'll briefly cover it here.
After all, we wouldn't develop features that the front end can't implement, or that are very difficult to implement. So we're still very confident in our understanding of Serious Play, just to say that we've made concessions on the feasibility of it. So from the first time we met to discuss what we wanted to do, we had a lot of whimsy about making our pages useful and playful, but after a period of study and discussion, we scaled back our engineering. Animations and effects that were initially expected but phased out are placed in future iterations and are currently only available in versions that are higher than the minimum feasible version.

prototype full of functions



lean mature prototype



sperate targets:

Doing so here has the following advantages: First, clear functionality and engineering scope, can be considered to a certain degree of feasibility analysis. Second, because we are a team with a lot of members, after splitting different functions, different modules can be left to different people in the development process, both developed and integrated. Shorten the development cycle and do your job.

difficulty of integration:

Of course, everything is a double-edged sword, which undoubtedly adds a step, that is, integration. This is also a technical difficulty in this step. In the actual operation process, we arranged two people complex front-end development, in the integration of the time is indeed different from the desired effect, after realignment, or solve the problem, relatively speaking, overall is improved efficiency.

Our app has three functions, one is to be able to click on the human body, generate corresponding, give users feedback some valuable information, one is the voting session, the last is, the comment area. We'll cover it in more detail below.

2.1 body click

This feature is arguably one of the most important features we want to do, and we want users to be attracted to the images of people on the page when they open the page and eager to interact. We specifically mark the vulnerability of the characters, and the cues of these places make it feel like they can click and interact with them.

Implementation: Chognyan qi does this part, which he writes in more detail

One of the difficulties: Here is the point of implementing this function is positioning and mouse response, of course, this is also called the same difficulty. We don't need to capture the position of the mouse, we just need to set the events that can be triggered in the specified locale. It's hard to know how this triggerable area can be relatively fixed when the page size or scroll changes, and the positioning doesn't shift when

inserted into the overall page as a component, but according to what we expect, we do encounter this problem, and finally switch between several positioning attempts and get the right solution for relative coordinates.

2.2 vote

intro

This part we think is an area that can be used as an interaction between the user and the user. The previous section was primarily about user-to-page interaction, and if voting data can be retained, user-user communication is another experience, not a similar feature, which is the main reason we want to implement this feature. To show everyone who comes to this page how many people are suffering from the same condition as him, or to see which of them is more prevalent, but they don't have it, and can be used as a reference for prevention, which makes this part very valuable, both to provide interactive entertainment and to provide value through reliable data.

implement

The implementation method we are going to use the chart .js or ng2-chart mentioned by the teacher in tourial, as a solution. But this is Plan B, and of course it's called Version 2.0 rather than Plan B. Because in version 1.0, front-end development had planned to set up display styles and data streams through html css and javascript homemade tables, and wanted to change the length of the bar chart with vote data. This may seem feasible, but in fact, it's not small, and the bar charts implemented in this way are a lot less than the more mature chartjs toolkits. So it's transformed into plan B, sitting on the chart with chartjs.

key

The main point here is to familiarize yourself with the properties and usage methods of chartjs in angular, change the style of the template to the style we expect, such as color, display of hidden axes, display of hidden table lines, whether the quantum starts from scratch, change vertical tables to horizontal, and so on.

hard

The difficulty here is not expected, but let us encounter, the chart will not be updated with the button pressed, data updates, we found through testing, when the button is pressed, the data has changed, but the length of the bar chart has not changed, here is a difficult point, in writing here has not been resolved, but still looking for elegant solutions.

2.3 comments

brief intro

Comments would not be so difficult if voting could be achieved. Very similar features, but can give users a different experience, because voting is a choice question, comments are a question and answer. This gives the user more freedom to express, rather than choose from just a few options, which makes sense.

hard

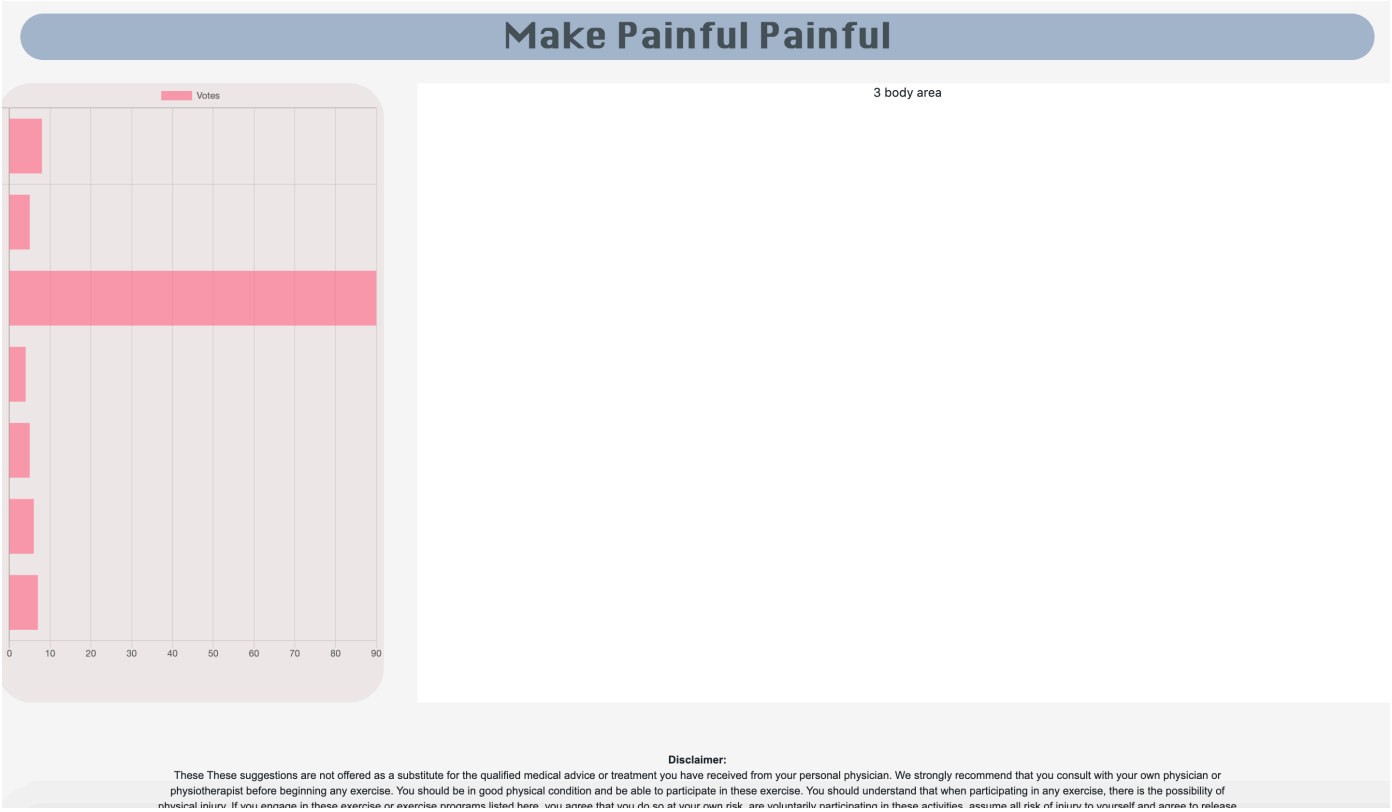
The difficulty here is the transmission of data, unlike voting, which requires only an array to get, which is not the same. A mature comment area may need to display a lot of information, such as the speaker's name, time, and content. The display area is limited, and how to select the data in the huge data to display in the limited space. When the user has entered the content in the input area and submitted it, it is critical and difficult to have the latest message appear in the display area to give the user immediate feedback.

3 . Layout of the single page web application.

We mentioned the functional division above, since the function needs to be divided and integrated, then the page actually needs to be divided and integrated. It makes sense to add some well-known tags such as header and footer to html5.
Our front-end design development divides our pages into three areas, which are more traditional types. Here's a quick look at each section:
The first, the title bar area, ready to place the logo project name and some potential features waiting for iterations, is reflected in our early prototypes, such as sharing settings, and so on, which we won't implement, but provides an area that can be added. Our part uses gray-blue background color, more low-key, but also more in line with our theme, the color of the font relative to the background color is relatively dark, but belongs to the same color system. At first the typography was placed in the top-only position, but also right-angle rectangle, and later felt not soft enough and beautiful to change to a sunken suspended style, and changed to fillets, and the length with the width of the page stretched.

The second area, where the main features and content are displayed, includes voting, messages, and the human body. The original plan was to divide the three sections into three sub-regions, where the tables were relatively independent, and the other two functions were located in the same component, popping up comments when body part events were triggered. But in the subsequent implementation process, changes, because of the complexity of the division of the body part independent. The bar chart and the commons are integrated because, in the bar chart component, there are clear buttons when voting, which is very much in line with the syntax of the angular. In general, the changes are intended for better development and more in line with the design of interactive logic.

old middle area



new middle area

The third area, which is a footer area, was originally planned to house disclaimer, GitHub page link, issue report email, and is currently the first to be implemented.

overall

So what method to divide these three areas, from traditional to modern methods have about the following three, first, the use of cable tags, which is the earliest web design when developers like the most commonly used typography, but over time and the emergence of a variety of practical tools, this approach gradually eliminated. Similar but more modern than this approach there is a library of front-end components called bootstrap, which has a feature feature, a grid system, to divide the layout of pages. But in fact, bootstrap in addition to this feature has a lot of useful features, but here we only need one, layout planning. So we used more direct div blocks plus css positioning to divide the area of our web pages. The advantage of doing this over bootstrap is that div blocks can be displayed in very highly customized blocks after they have been sorted by css. For example, we want header's top bar section to always be top of everything, no matter how the page size changes and how the page scrolls, it stays in place and at the top. In another example, our body part is expected to float above the table, rather than being squeezed to the next row after the browser window width has been manually adjusted to be smaller. How to implement it I'll list the code and explain it in the next section.

3.1 bootstrap

Bootstrap is a popular front-end framework, based on html css jss. It is simple and flexible relatively fast for web development.

The advantages are as follows:
Mobile device compatibility, all mainstream browsers support bootstrap
Easy to use, as long as you have basic knowledge of html and css, you can start learning bootstrap, responsive design, and can adapt to desktops, tablets and mobile phones.
It provides a simple and unified solution for developers to create interfaces
It contains powerful built-in components, easy to customize
It also provides opportunities for web customization
It is open source

It's really powerful, but it's also because it's so powerful that our demo application at this stage may not be able to use it, at least in terms of zoning, we have more flexibility and more appropriate options. I believe that in a real-world enterprise-level project, bootstrap has an absolute advantage, but in our project, fun itself is a part of our project, our page does not have a commercial use plan, so nothing so rigorous and have some casual style is still acceptable. Of course, we won't use bootstrap for zoning at this stage, but we'll take advantage of its other features.

3.2 div + CSS

This approach is best suited to layout division, but it has several advantages: first, the tool does not need to be very powerful, just good enough to achieve the requirements, and that is it. Second, the degree of customization is higher, more flexible. In my example above, it is not difficult to see its advantages.

Similarly, in the next section, I'll use our actual code and some appropriate and necessary explanations to explain how we can use this approach to solve the web page division requirement.

3.3 table tag

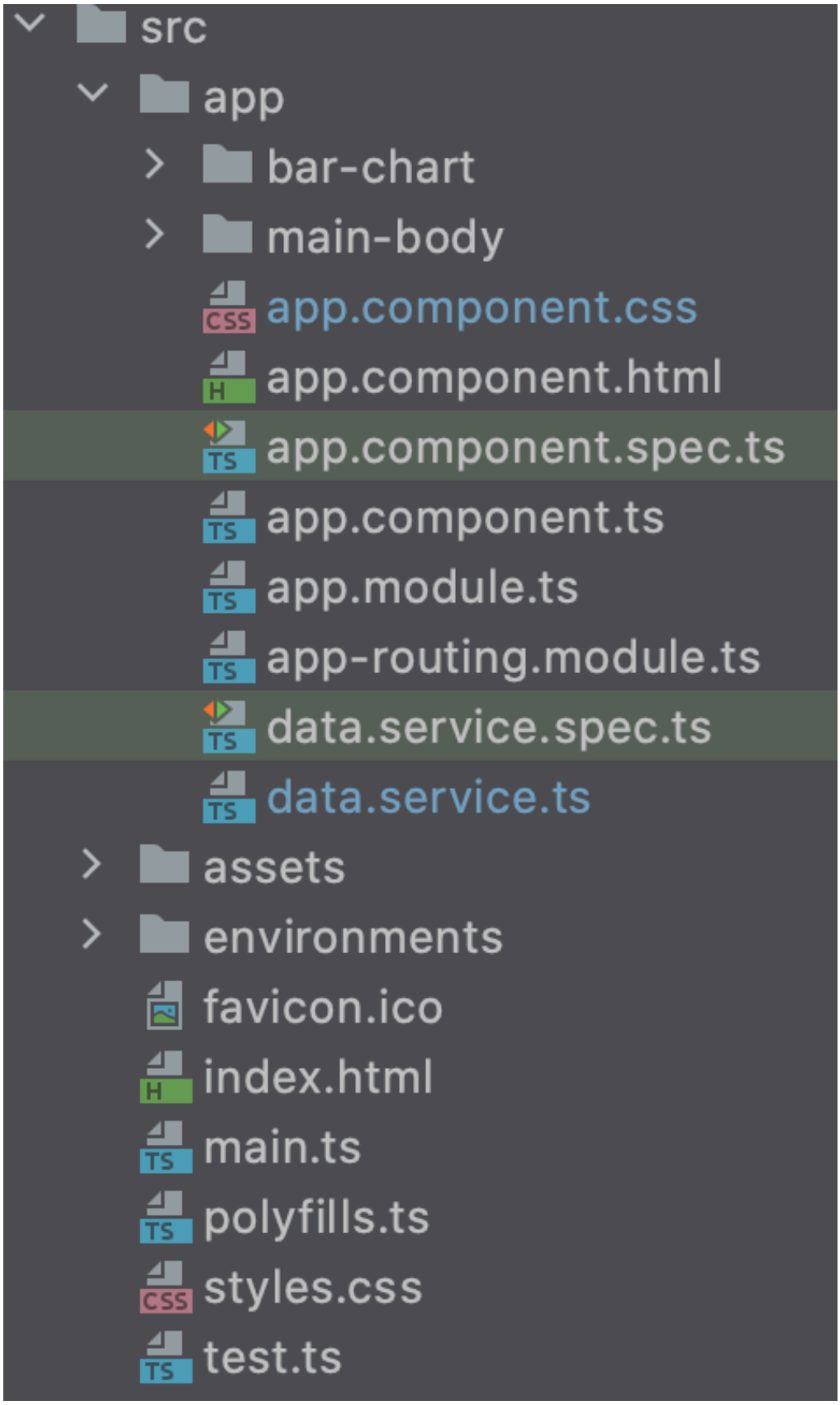
This is a traditional and outdated approach, and functional limitations exist, so this approach is not chosen to implement it.

4 . Fitures implementation

This chapter, is the last section of front-end development, the previous sections describe the comparative theory, stay in the design ideas and implementation ideas. In this section we'll actually paste some code in, do some explanations, and the entire development process. There may be fewer detours to show the latest code directly, but words will still be used to describe some of the key turning points and the different scenarios explored.
Below, I will cut into the different sections, detailing how to build the current page, but also mention that there are better implementation methods, room for improvement, problems that have been solved and unresolved difficulties.

4.1 files

The following diagram, part of the entire project Chinese and folders, shows two custom components that contain key code to implement our main functionality.



4.2 Steps

this Part is to start describing the process of building the entire project. And take the final code as an example to report.

We are going to start a new project called dashboard using Angular CLI. Create a new workspace and an initial application. A workspace contains the files for one or more projects. A project is the set of files that comprise an app, a library, or end-to-end (e2e) tests. Here is to create a new workspace.

```
ng new spa --routing

cd spa
npm install ng2-charts
npm install chart.js
npm install bootstrap
```

The Angular CLI installs the necessary Angular `npm` packages and other dependencies. This can take a few minutes. It also creates the following workspace and starter project files:

A new workspace, with a root folder named `spa`.

An initial skeleton app project, also called `spa` (in the `src` subfolder).

An end-to-end test project (in the e2e subfolder).

Related configuration files.

Go to the workspace directory and launch the application. The `ng serve` command builds the app, starts the development server, watches the source files, and rebuilds the app as you make changes to those files. The `--open` flag opens a browser to `http://localhost:4200/`.

```
cd spa
ng serve --open
```

Make changes to the application, Change the application title

Open the project in IDE and navigate to the `src/app` folder to make some changes to the starter app.

Open the component class file (`app.component.ts`) and change the value of the `title` property to 'Make Painful Painful'.

Add scripts

To make use of the charting framework we need to add the following script to the 'build' in the `angular.json` file. This will ensure scripts are added where required to the HTML body. We are going to add the Chart.js framework globally. so it will be available to every component in your application. We can see in `build: { }` object; you just need to add this script:

```
"scripts": ["node_modules/chart.js/dist/Chart.js"]
```

Delete the template codes in the `app.component.html`, which means delete all the things in that file.

Using Chart.js to create a Bar Chart

Now we have setup the required dependencies we can actually create some charts.

```
ng g c bar-chart
```

This command calls a utility function which helpfully adds:

1. src/app/bar-chart/bar-chart.component.html
2. src/app/bar-chart/bar-chart.component.ts
3. src/app/bar-chart/bar-chart.component.css
4. src/app/bar-chart/bar-chart.component.spec.ts

Go and open bar-chart.component.html and replace the boilerplate

bar-chart works!

content with something to load our newly created bar chart.

```
<div style="display: inline-block; width: 500px; " >
  <canvas baseChart width="2" height="3"
          [datasets]="barChartData"
          [labels]="barChartLabels"
          [options]="barChartOptions"
          [plugins]="barChartPlugins"
          [legend]="barChartLegend"
          [chartType]="barChartType"
          [colors]="barChartColors">
  </canvas>
</div>
```

This part is only for the chart, we also need button and comments place. So, we amend code like these

followed.

```
<div style="display: inline-block;height: 800px; width: 150px; position: absolute;left:
550px">
  <button *ngFor="let barChartLabel of barChartLabels"
  style="display: block; width: 120px; margin: 38px 0 37px 0; height: 40px; border-
radius: 5px"
          (click)="clkbtn(barChartLabel)"
  >
    {{barChartLabel}}
  </button>
</div>
```

```
<div style="display: inline-block; height: 700px; width: 300px; margin: 50px;
position: absolute;left: 680px;
            border-radius: 30px; padding: 20px; background-color: #e9e2e5">
  <strong>Comments:</strong>

<!--  <div id="tst" style="border: #46505a solid 1px; width: 100px; height: 100px">-->
<!--    {{tstString}}-->
<!--  </div>-->

</div>
```

Here we're using the baseChart directive which is added to a canvas element. Furthermore, the attributes datasets, labels, options, legend and chartType are bound to class members which are added to the implementation of class BarChartComponent in bar-chart-component.ts:

```
import {Component, OnInit, ViewChild} from '@angular/core';
import { DataService } from '../data.service';
import { ChartDataSets, ChartOptions, ChartType } from 'chart.js';
import { Color, BaseChartDirective, Label } from 'ng2-charts';
// import * as pluginAnnotations from 'chartjs-plugin-annotation';

import { ChangeDetectorRef } from '@angular/core';

@Component({
  selector: 'app-bar-chart',
  templateUrl: './bar-chart.component.html',
  styleUrls: ['./bar-chart.component.css']
})

export class BarChartComponent implements OnInit {


  constructor(private dataService: DataService) { }

  public barChartOptions = {
    scaleShowVerticalLines: false,
    responsive: true,
    legend: {
      labels: { fontColor: 'black' }
    },
    scales: {
      xAxes: [{
        ticks: { fontColor: 'black', beginAtZero: true,
          backdropColor: { color: 'rgba(255,255,255,0.1)' } },
        gridLines: { color: 'rgba(255,255,255,0.1)' },
        color: { color: 'rgba(255,255,255,0.1)' },
        borderColor: { color: 'rgba(255,255,255,0.1)' },
        display: true
      }],
```

```
      yAxes: [{
        ticks: { fontColor: 'black' },
        gridLines: { color: 'rgba(255,255,255,0.1)' },
        display: false
      }]
    },
  };
  public barChartLabels = ['Head', 'Eyes', 'Neck Pain', 'Shoulder', 'Cervical',
'Lumbar', 'Hand', 'Limb Pain', 'Foot'];
  public barChartType = 'horizontalBar';
  public barChartLegend = true;
  public barChartData = [
    {data: [3, 2, 1, 4, 8, 6, 7, 5, 8], label: 'Vote if you got same pain by click
buttons on the right'},
    // {data: [28, 48, 40, 19, 86, 27, 90], label: 'Series B'}
  ];
  public barChartPlugins = [];
  public barChartColors: Color[] = [
    {
      borderColor: 'black',
      backgroundColor: 'rgba(90,85,85,0.3)',
    },
  ];

  stats: any = [];
  currentdata = null;
  currentIndex = -1;
  title = '';
  indexOfBar = 0;
  tstString = '';

  ngOnInit() {
    this.retrieveData();
    this.retrieveComments();
  }

  retrieveData() {
    this.dataService.getAll().subscribe(
      data => {
        this.stats = data;
        this.barChartLabels = this.stats.barChartLabels;
        this.barChartData = this.stats.barChartData;
      },
      error => {
        console.log(error);
      });
  }
```

```
  retrieveComments() {
    this.dataService.getComments().subscribe(
      data => {
        this.stats = data;
        this.tstString += this.stats[0].content;
      },
      error => {
        console.log(error);
      });
  }

  clkbtn(barChartLabel: string){
    this.indexOfBar = this.barChartLabels.indexOf(barChartLabel);
    this.barChartData[0].data[this.indexOfBar]++;
  }
}
```

Now we have setup the required dependencies we can actually create some charts.

```
ng g c main-body
```

This command calls a utility function which helpfully adds:

1. src/app/main-body/main-body.component.html
2. src/app/main-body/main-body.component.ts
3. src/app/main-body/main-body.component.css
4. src/app/main-body/main-body.component.spec.ts

html

```
<img id="img1" src="/assets/body.png" alt="body" useMap="#planetMap">
<map name="planetMap" id="planetMap">
  <area class="tmp" shape="circle" coords="270,140,15" alt="Shoulder" id="s1"
onMouseOver="show1()">
  <area class="tmp" shape="circle" coords="100,130,15" alt="Neck" id="s2"
onMouseOver="show2()">
  <area class="tmp" shape="circle" coords="280,300,15" alt="Back" id="s3"
onMouseOver="show3()">
  <area class="tmp" shape="circle" coords="290,360,15" alt="Lumbar" id="s4"
onMouseOver="show4()">
  <area class="tmp" shape="circle" coords="85,425,15" alt="Wrist" id="s5"
onMouseOver="show5()">
  <area class="tmp" shape="circle" coords="305,575,15" alt="Leg" id="s6"
onMouseOver="show6()">
  <area class="tmp" shape="circle" coords="170,65,15" alt="Eyes" id="s7"
onMouseOver="show7()">
  <area class="tmp" shape="circle" coords="120,30,15" alt="Head" id="s8"
onMouseOver="show8()">
```

```
  <area class="tmp" shape="circle" coords="320,675,15" alt="Foot" id="s9"
onMouseOver="show9()">
</map>

<div class="circle1" id = "circle1"></div>
<div class="circle2" id = "circle2"></div>
<div class="circle3" id = "circle3"></div>
<div class="circle4" id = "circle4"></div>
<div class="circle5" id = "circle5"></div>
<div class="circle6" id = "circle6"></div>
<div class="circle7" id = "circle7"></div>
<div class="circle8" id = "circle8"></div>
<div class="circle9" id = "circle9"></div>
```

CSS

```
#img1{
  z-index:1;
  position:absolute;
  left:0px;
  top:0px;
  /*height: 800px;*/
  /*width: 392px;*/
}

.circle1{
  z-index: 100;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 270px;
  top: 140px;
}

.circle2{
  z-index: 101;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 100px;
  top: 130px;
}
```

```css
.circle3{
  z-index: 102;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 280px;
  top: 300px;
}

.circle4{
  z-index: 103;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 290px;
  top: 360px;
}

.circle5{
  z-index: 104;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 85px;
  top: 425px;
}

.circle6{
  z-index: 105;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 305px;
  top: 575px;
}
```

```
.circle7{
  z-index: 106;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 170px;
  top: 65px;
}

.circle8{
  z-index: 106;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 120px;
  top: 30px;
}

.circle9{
  z-index: 106;
  width: 10px;
  height: 10px;
  border-radius: 10px;
  border: 2px solid red;
  box-sizing: border-box;
  position: absolute;
  left: 330px;
  top: 680px;
}
```

amend script in index.html

```
<script>
    window.onload = function(){
      adjust();
      // passInfo();
    }



    function adjustPosition(position) {
      var myImage = document.getElementById("img1");
```

```
      if (typeof myImage.naturalWidth == "undefined") {
        // IE 6/7/8
        var i = new Image();
        i.src = myImage.src;
        var rw = i.width;
        var rh = i.height;
      }
      else {
        // HTML5 browsers
        var rw = myImage.naturalWidth;
        var rh = myImage.naturalHeight;
      }
      // 获取宽高
      let pageWidth = rw;
      let pageHeight = rh;
      // 图片原始尺寸
      let imageWidth = 392;
      let imageHeight = 800;

      let each = position.split(",");

      for (let i = 0; i < 2; i++) {
        if(i%2==0){
          // 新的x轴坐标
          each[i] = Math.round(parseInt(each[i]) * pageWidth / imageWidth).toString();
        }else{
          // 新的y轴坐标
          each[i] = Math.round(parseInt(each[i]) * pageHeight /
imageHeight).toString();
        }
      }
      let newPosition = "";
      for (let j = 0; j < each.length; j++) {
        newPosition += each[j];
        if (j < each.length - 1) {
          newPosition += ",";
        }
      }
      return newPosition;
    }

    function adjust() {
      let map = document.getElementById("planetMap");
      let area = map.getElementsByTagName('area');

      for (let i = 0; i < area.length; i++) {
        let oldCoords = area[i].getAttribute("coords");
        let newCoords = adjustPosition(oldCoords);
        area[i].setAttribute("coords", newCoords);
```

```
      }
    }

    function show1(){
      alert("Class: Shoulder"+"\n\n"
        +"Common symptoms: Shoulder pain occurs when the arm moves up and back, or even does not move"+"\n\n"
        +"Cause: There is overuse and chronic inflammation, over-movement of the arm can cause the top of the upper limb bone to squeeze the shoulder sleeve muscles toward the upper end of the shoulder blade, which can cause muscle inflammation and swelling. If you continue to exercise despite inflammation, the tendon weakens and tears."+"\n\n"
        +"Treatment: Many shoulder injuries can be recovered after rest. Nonsteroidal anti-inflammatory drugs (NSAIDs, such as ibuprofen) can also be used for short-term pain relief (up to 72 hours). Exercise helps."
      );
    }

    function show2(){
      alert("Class: Neck pain"+"\n\n"
        +"Common symptoms: Cervical cervical vertebral disease"+"\n\n"
        +"

            ...
            ...

    function show9()
            ...
            hand in the direction of your back. Or choose a grip, handball, etc. At the same time, also avoid cold hands."
      );
    }
  </script>
```

Now we go back to the `app.component.html` file where we can just implement the separating the page.

for header, top bar

```
<!--This area is made for the header-->
<div class="areas" id="header">
  {{title}}
</div>
```

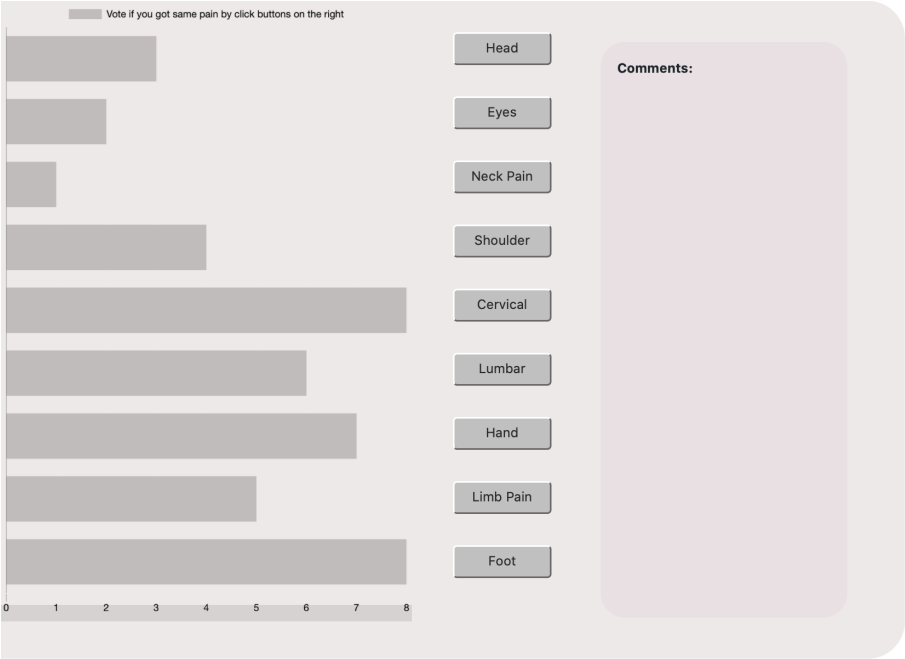
Make Painful Painful

for main content area

```
<!--This area is made for the main content, including 3 parts-->
<div style="min-width: 2000px">
  <div id="body_left" >
    <div id="votesBarBox" >
      <app-bar-chart  id="votesBar"></app-bar-chart>
    </div>
  </div>

  <div id="body_right">
<!--      <img src="../assets/body.png" alt="123" height="800px" width="620px">-->
<!--    <div style="height: 800px; width: 200px; display: inline-block">-->

<!--    </div>-->
    <app-main-body> </app-main-body>
  </div>
</div>
```



for footer

```
<div class="areas" id="footer">
  <strong>
    Disclaimer:
  </strong>
  <hr/>
  These Th... ligence.
</div>
```

Disclaimer:

These These suggestions are not offered as a substitute for the qualified medical advice or treatment you have received from your personal physician. We strongly recommend that you consult with your own physician or physiotherapist before beginning any exercise. You should be in good physical condition and be able to participate in these exercise. You should understand that when participating in any exercise, there is the possibility of physical injury. If you engage in these exercise or exercise programs listed here, you agree that you do so at your own risk, are voluntarily participating in these activities, assume all risk of injury to yourself and agree to release and discharge our website from any and all claims or causes of action. known or unknown, arising out of Make Painful painful negligence.

In addition to the changes to the above two files that are similar to those in the teacher's tutorial, we have also changed the style to make the page similar to the layout we expected.

So we modify the content of the file `bar-chart.component.css` like this:

```
#votesBar{
  background-color: #80808030;
  height: auto;
  width: auto;
  top: 0;
  bottom: 0;
  margin: 0;
  padding: 0;
}

.container{
  margin: 0;
  padding: 0;
  height:auto !important;

  min-height:344px;
}
.areas{
  text-align: center;
}

#header{
  height: 60px ;
  margin: 30px;
  padding: 0;
  border-radius: 30px;
  background-color: rgb(162,181,205);
  position: sticky;
  top : 30px;
  z-index: 999;
  text-align: center;
  font-family: Silom;
  font-size: 45px;
  line-height: 60px;
  color: #46505a;
}


#footer{
  margin: 10px 200px 0 200px;
  height: 60px;

  clear: right;
  font-size: 14px;
```

```
  font-family: Arial;

  position: relative;
  top: 10px;
}

#votesBarBox{
  min-width: 1620px;

}

#body_left{
  display: inline-block;
  /*position: relative;*/
  border-radius: 0 45px 45px 0;
  height: 800px;
  width: 1100px;
  min-width: 500px;
  background-color: rgb(238,233,233);
}

#body_right{
  display: inline-block;
  position: relative;
  border-radius: 45px;
  height: 800px;
  width: 30%;
  min-width: 400px;
  vertical-align: top;
  /*background-color: rgb(245,245,245);*/
  /*background-color: #FFFFFF;*/
  float: right ;
  right: 0;
  z-index: 500;
  position: fixed;

}

#card-right{
  height: 800px;
  width: 300px;
  min-width: 300px;
  display: inline-block;
  border: red solid 1px;
  vertical-align: top;
}
```

Update app.module

First we need to make sure the main imports are present to reflect our additions:

```
import {BrowserModule} from '@angular/platform-browser';
import {NgModule} from '@angular/core';

import {ChartsModule} from 'ng2-charts';

import {AppRoutingModule} from './app-routing.module';
import {AppComponent} from './app.component';
import {BarChartComponent} from './bar-chart/bar-chart.component';


@NgModule({
  declarations: [
    AppComponent,
    BarChartComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ChartsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {
}
```

Before running you need to make sure `angularCompilerOptions` are removed from your `tsconfig.json`. In various recent versions of Angular CLI strict injection checking sometimes gets enabled in the boilerplate code which would prevent the Chart.js canvas render from working. Edit this file and make sure your `tsconfig.json` looks like the below:

```
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
    "importHelpers": true,
```

```
    "target": "es2015",
    "module": "es2020",
    "lib": [
      "es2018",
      "dom"
    ]
  }
}
```