Final Project Documentation
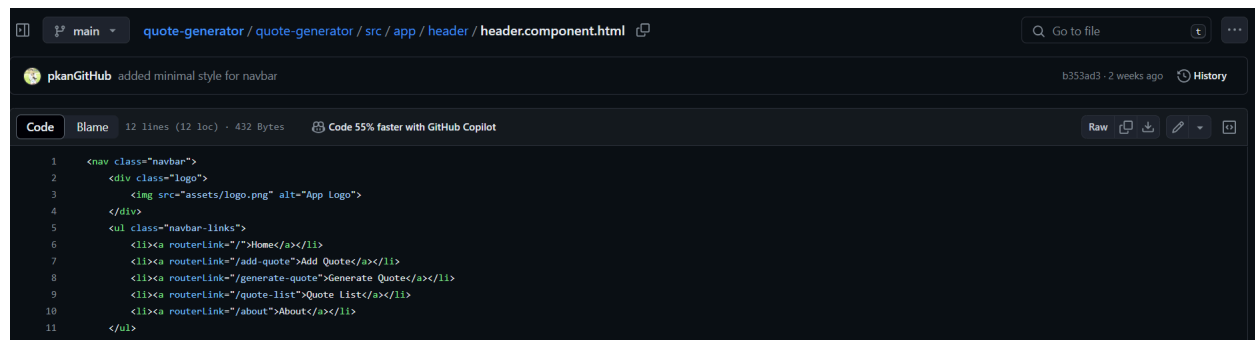
Maya Bowman, Stella Craig, Phyllis Kan, Michael Sturm, Upasana Roy
Github: https://github.com/pkanGitHub/quote-generator
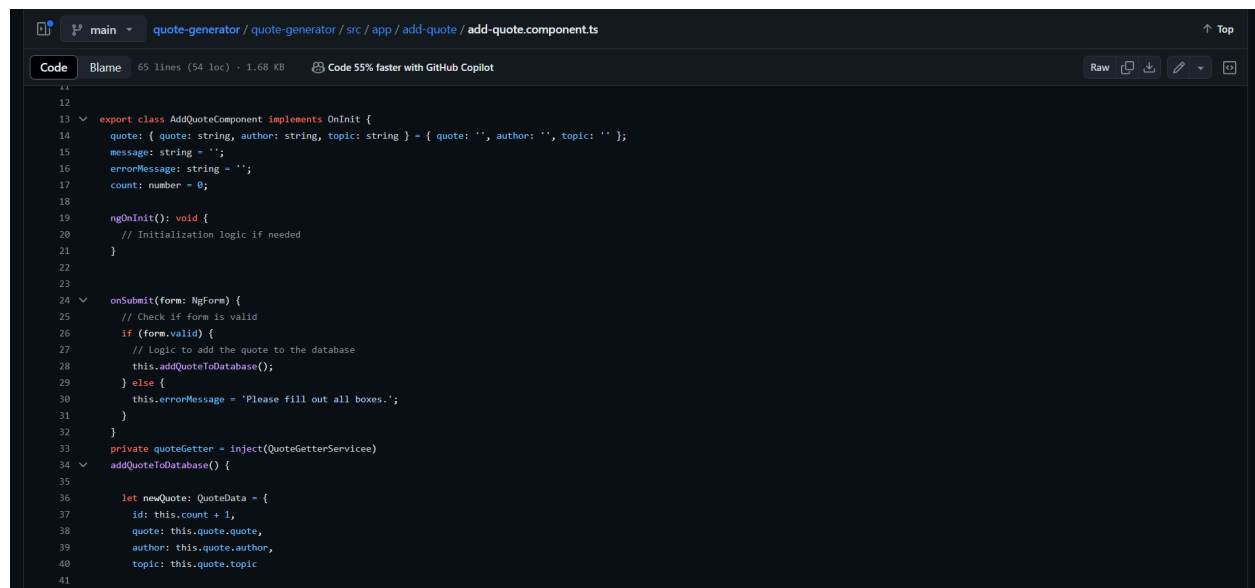
To run the project use "npm run dev".

Angular

In order to meet the Angular requirement, we created a header component, footer component, and various other body components like about, add-quote, gen-quote, home, quote-list, and profile card. We used the header to route between all of the different pages as shown here:

We used the body components to send information to each other shown here in the onSubmit function:

In our footer we put the names of each of the group members:
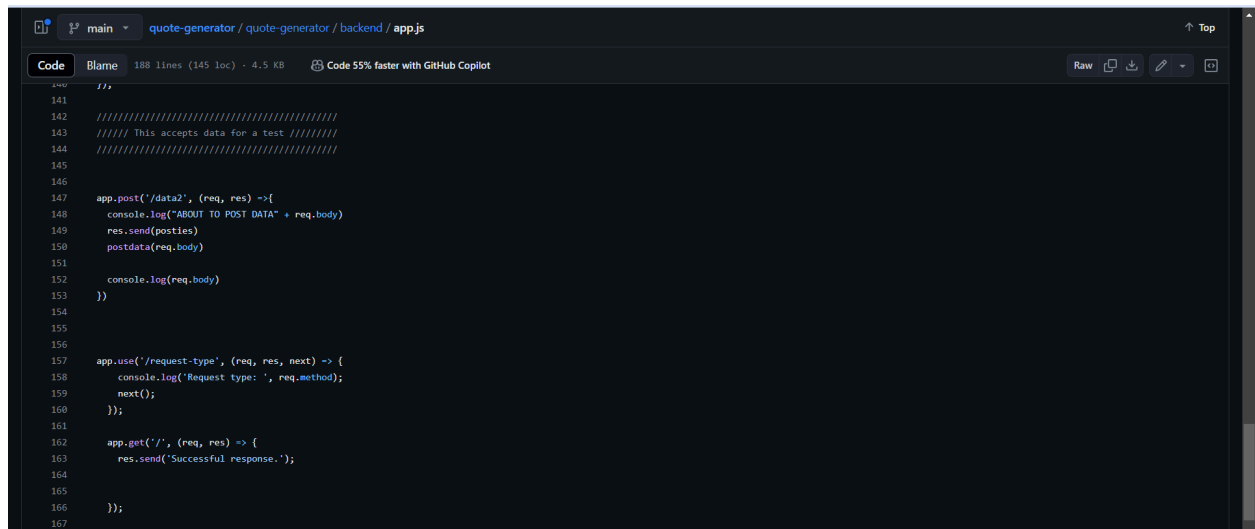
```
1   <div class="footer">
2       <div class="project-by">
3           Project by Maya B., Stella C., Phyllis K., Michael S., and Upasana R.
4       </div>
5       <div class="by-year">April 2024</div>
6   </div>
```
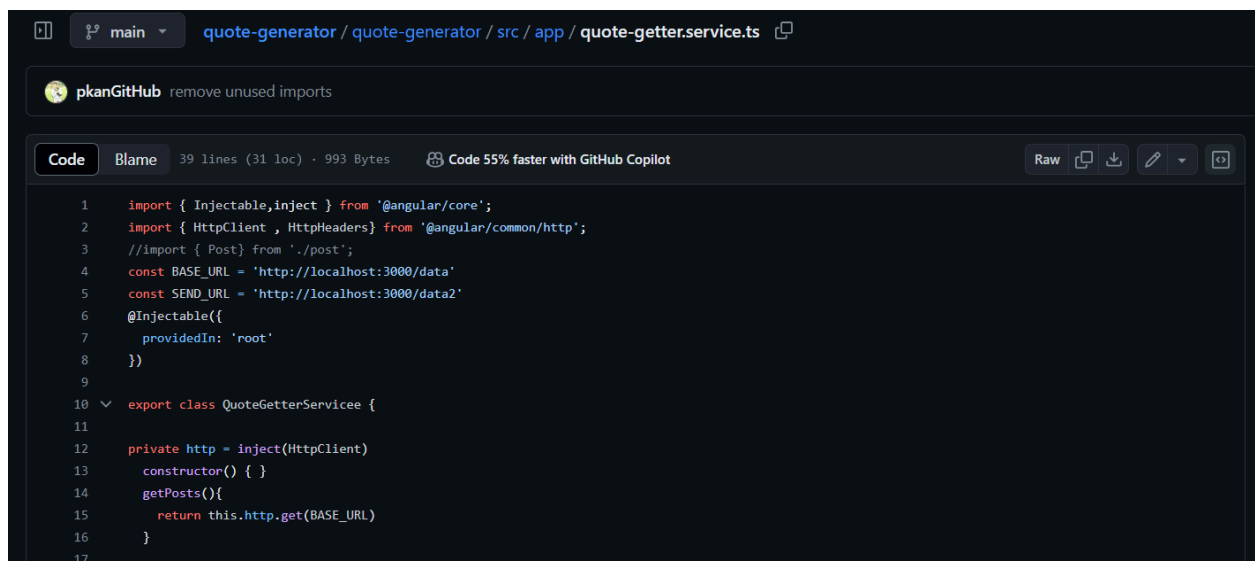
NodeJS

In order to meet the requirements for NodeJS, we built our backend in NodeJS and used HTTP for the connection to the data between our frontend and backend.


```
141
142   /////////////////////////////////////////
143   ////// This accepts data for a test /////////
144   /////////////////////////////////////////
145
146
147   app.post('/data2', (req, res) =>{
148       console.log("ABOUT TO POST DATA" + req.body)
149       res.send(posties)
150       postdata(req.body)
151
152       console.log(req.body)
153   })
154
155
156
157   app.use('/request-type', (req, res, next) => {
158       console.log('Request type: ', req.method);
159       next();
160   });
161
162   app.get('/', (req, res) => {
163       res.send('Successful response.');
164
165
166   });
167
```
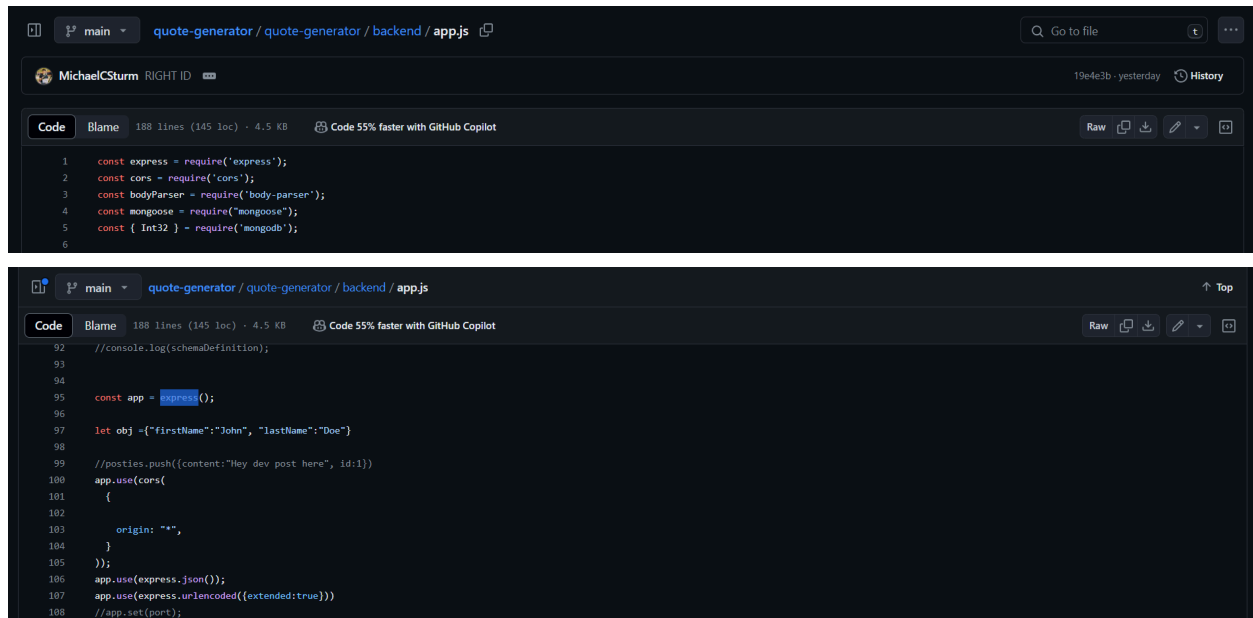


```
1   import { Injectable,inject } from '@angular/core';
2   import { HttpClient , HttpHeaders} from '@angular/common/http';
3   //import { Post } from './post';
4   const BASE_URL = 'http://localhost:3000/data'
5   const SEND_URL = 'http://localhost:3000/data2'
6   @Injectable({
7       providedIn: 'root'
8   })
9
10  export class QuoteGetterServicee {
11
12  private http = inject(HttpClient)
13      constructor() { }
14      getPosts(){
15          return this.http.get(BASE_URL)
16      }
17
```

Express.js

For the Express.js requirement, we used express to create a restful API between our backend and service.ts:

All of our communication and data transfer was done in our http from node as shown in the requirements for NodeJS above.

MongoDB

      Using mongo we allowed all IP's to be able to get and post to the db. We set it up so the service automatically calls the database to put into an array. We use this array to avoid the OnInit async waiting for db error. We also use this array to find the number of quotes in quotes to assign an id that counts up. We also have a quotes Schema to format the data to a pushable state.

Routing.ts

   To meet the Routing requirement, we created the navbar using RouterModule, a built-in Angular module for handling routing. The RouterModule is imported into the main application module. Each route in the RouterModule consists of a path and a component field for all components we created, such as home, gen_quote, add_quote, quote_list, and about.

```ts
quote-generator > src > app > TS app-routing.module.ts > ...
 1    import { NgModule } from '@angular/core';
 2    import { RouterModule, Routes } from '@angular/router';
 3    import { HomeComponent } from './home/home.component';
 4    import { AddQuoteComponent } from './add-quote/add-quote.component';
 5    import { GenQuoteComponent } from './gen-quote/gen-quote.component';
 6    import { QuoteListComponent } from './quote-list/quote-list.component';
 7    import { AboutComponent } from './about/about.component';
 8
 9    const routes: Routes = [
10        {path: '', component: HomeComponent, pathMatch: 'full'},
11        {path: 'add-quote', component: AddQuoteComponent},
12        {path: 'generate-quote', component: GenQuoteComponent},
13        {path: 'quote-list', component: QuoteListComponent},
14        {path: 'about', component: AboutComponent},
15    ];
16
17    @NgModule({
18        imports: [RouterModule.forRoot(routes)],
19        exports: [RouterModule]
20    })
21    export class AppRoutingModule { }
```

To link up the routes in the header component, we used HTML list elements such as <nav>, <ul>, <li> for each route link. And for each <li> element, it would contain an Angular router link directive, pointing to the corresponding path defined in the RouterModule.

```html
v <nav class="navbar">
v     <div class="logo">
          <img src="assets/logo.png" alt="App Logo">
      </div>
v     <ul class="navbar-links">
          <li><a routerLink="/">Home</a></li>
          <li><a routerLink="/add-quote">Add Quote</a></li>
          <li><a routerLink="/generate-quote">Generate Quote</a></li>
          <li><a routerLink="/quote-list">Quote List</a></li>
          <li><a routerLink="/about">About</a></li>
      </ul>
  </nav>
```

Service.js

It's an injectable that has loadPosts which posts the quote to the DB using mongo and getPosts which gets the data. This injectable is injected into addQuoteComponent, QuoteListComponent and GenQuoteComponent. This helps the project be very scalable.

Component.css

Our application contains a proper layout using css padding and margins. Our application is visually symmetric using css with the components.

App.component.css:

quote-list.component.css:


```
quote-generator > src > app > # app.component.css > ...
       Phyllis Kan, 2 weeks ago | 1 author (Phyllis Kan)
 1    html, body {
 2        height: 100%;
 3        margin: 0;
 4    }
 5
 6    main {
 7        display: flex;
 8        min-height: 100vh;
 9        flex-direction: column;
10    }
11
12    .content {
13        flex: 1;
14    }
```

```
quote-generator > src > app > quote-list > # quote-list.component.css > ...
       You, yesterday | 1 author (You)
 1    /* Styles for the quote list */
 2    .quote-list {
 3        width: 80%;          /* Set the width to 80% of the viewport width */
 4        text-align: center; /* Center align the text */
 5        margin: 0 auto;     /* Center the list horizontally */
 6        list-style: none;   /* Remove default list styling */
 7        padding: 0;         /* Remove default padding */
 8    }
 9
10    h1 {
11        font-weight: bold;  /* Make the text bold */
12        text-decoration: underline;  /* Underline the text */
13    }
14
15    .quote-item {
16        text-align: center; /* Center align the text */
17        margin-bottom: 20px; /* Space between quotes */
18    }
19
20    .quote-text {
21        color: □black; /* Set the text color to black */
22    }
23
24    .author-text {
25        color: ▪grey; /* Set the author's name to grey */
26    }
27
28    .topic-text {
29        color: ▪lightgray;
30    }
```

And the following examples:

Subjects

We used subjects and subscriptions to handle the data that will be displayed from our database.

```
18  v      loadPosts(CurrPost:JSON) {
19

20

21

22

23          console.log(JSON.stringify(CurrPost) + "AYO THIS IS IN QUOTE GETTER SERVICE")
24          const headers = new HttpHeaders()
25                .set('Authorization', 'my-auth-token')
26                .set('Content-Type', 'application/json');
27          //console.log(body);
28          this.http.post(SEND_URL, CurrPost, { headers:headers })
29          .subscribe(data => {
30            console.log(data)
31          });
32        }
33      }
```