Overview:

1. postfilecomponent: This component is responsible for allowing users to create new posts. It contains a form where users can input their post content and submit it. Once submitted, it interacts with the `PostService` to add the new post.

2. postlistcomponent: This component is responsible for displaying a list of posts. It subscribes to updates from the `PostService` to retrieve the list of posts and displays them in the UI.

3. postservice: This service acts as an intermediary between the components and the data source (either local storage or a server). It provides methods for adding new post and, retrieving a list of posts. It uses HTTP requests to communicate with a server and manages the state of the posts within the Angular application.

4. postinterface: This file defines an interface for representing a post object. It includes properties like `title` and `content`, that a post object should have. Using interfaces helps maintain consistency and type safety when working with post objects throughout the application.

Post.service.component.ts

```ts
// with node
import { Injectable } from '@angular/core';
import { BehaviorSubject, Observable } from 'rxjs';
import { Post } from './post.interface';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class PostService {
  private apiUrl = 'http://localhost:3000/posts';
  private posts: Post[] = [];
  private postsSubject: BehaviorSubject<Post[]> = new BehaviorSubject([]);

  constructor(private http: HttpClient) { }

  // 1. methods to handle local posts
  // add new posts locally
  addPost(post: Post): void {
    // assign incremental title for each post created
    post.title = `Post ${this.posts.length + 1}`
    this.posts.push(post);
    // update BehaviorSubject
    this.postsSubject.next([...this.posts]);
  }

  // get locally created posts
  getPosts(): Observable<Post[]> {
    return this.postsSubject.asObservable();
  }

  // 2. methods to handle Node.js server
  // fetch hardcoded data
  fetchPostsFromServer(): void {
    this.http.get<Post[]>(this.apiUrl).subscribe(posts => {
      this.posts = posts;
      this.postsSubject.next([...this.posts]);
    });
  }

  // add new post and send to Node.js server
  addPostToServer(post: Post): Observable<Post> {
    return this.http.post<Post>(this.apiUrl, post);
  }
}
```

Post.file.component.ts

src > app > post > post-file > TS post-file.component.ts > PostFileComponent > submitPost > newPost > content

```ts
import { Component } from '@angular/core';
import { PostService } from '../post.service';
import { Post } from '../post.interface';

@Component({
  selector: 'app-post-file',
  templateUrl: './post-file.component.html',
  styleUrls: ['./post-file.component.css']
})
export class PostFileComponent {
  postTitle: string
  postContent: string;

  constructor(private postService: PostService) {}

  submitPost() {
    if (!this.postContent || this.postContent.trim() === '') {
      return;
    }
    const newPost: Post = {
      title: this.postTitle,
      content: this.postContent
    };
    this.postService.addPost(newPost);
    this.postContent = '';
  }
}
```

Post.interface

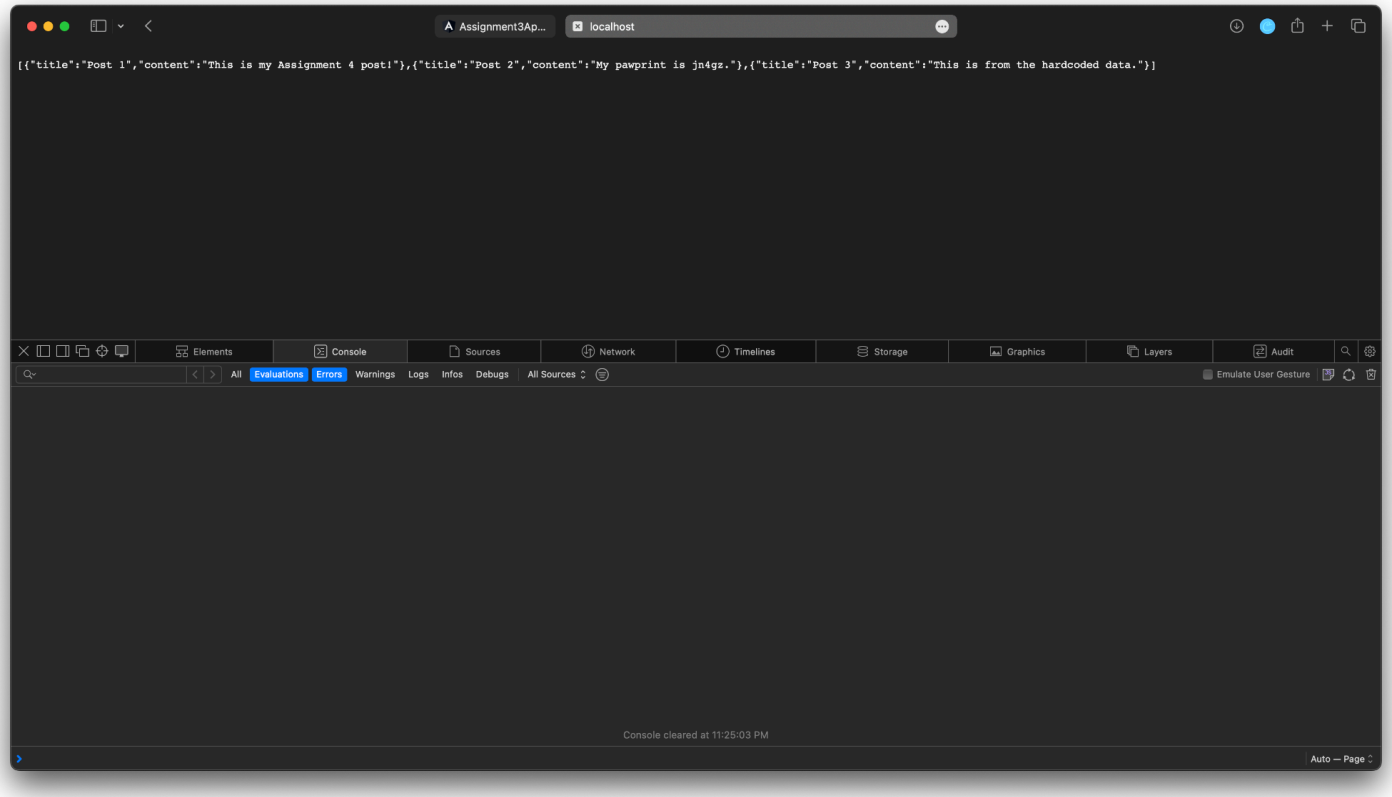
src > app > <> app.component.html > ...

```html
<app-header></app-header>
<app-post-file></app-post-file>
<app-post-list></app-post-list>
```

Post.list.component.ts

```
// node attempt 2
// seems subscription to server cause duplication
import { Component, OnInit, OnDestroy } from '@angular/core';
import { PostService } from '../post.service';
import { Subscription } from 'rxjs';
import { Post } from '../post.interface';

@Component({
  selector: 'app-post-list',
  templateUrl: './post-list.component.html',
  styleUrls: ['./post-list.component.css']
})
export class PostListComponent implements OnInit, OnDestroy {
  posts: Post[] = [];
  private postsSubscription: Subscription;
  private serverPostsSubscription: Subscription;

  constructor(private postService: PostService) {}

  ngOnInit() {
    // subscribe to local posts
    this.postsSubscription = this.postService.getPosts().subscribe(posts => {
      // update copy of the array
      this.posts = [...posts];
    });

    // fetch posts from Node.js server
    this.postService.fetchPostsFromServer();
  }

  ngOnDestroy() {
    // unsubscribe to avoid memory leaks
    this.postsSubscription.unsubscribe();
    if (this.serverPostsSubscription) {
      this.serverPostsSubscription.unsubscribe();
    }
  }
}
```

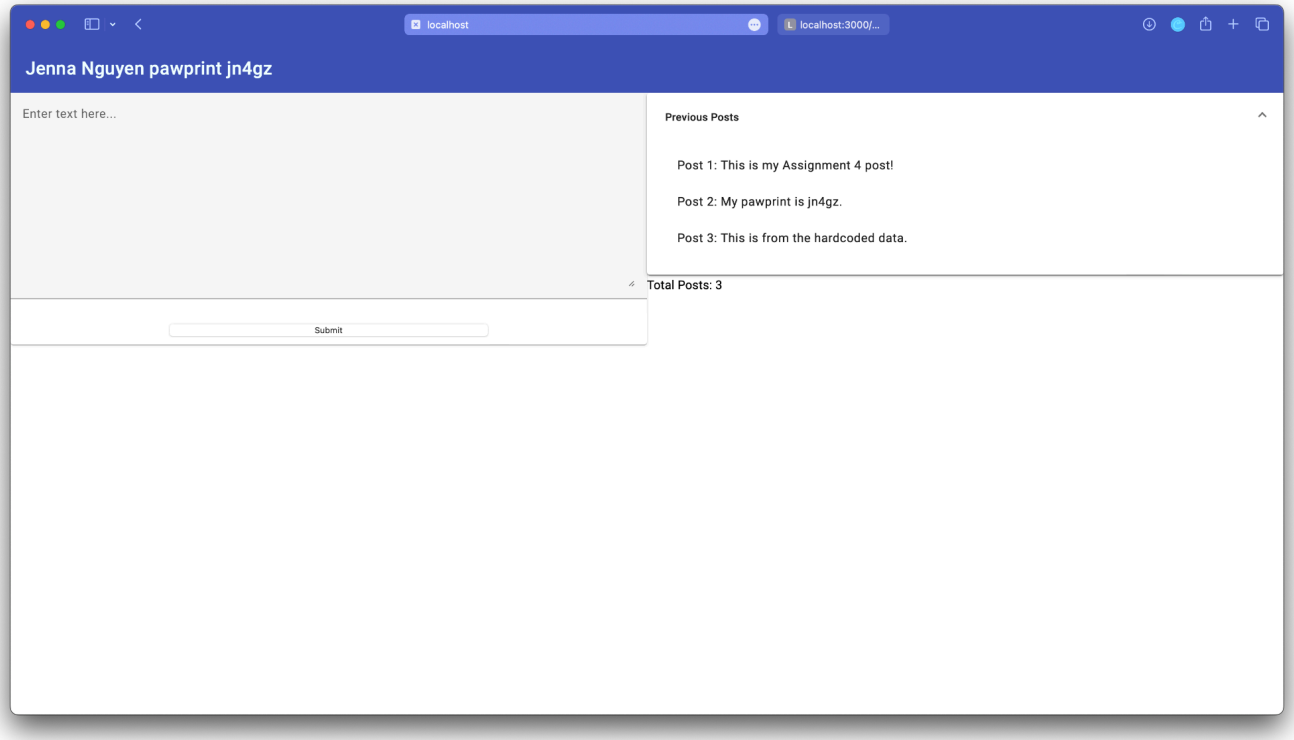
Json hardcoded data from Node.js and Node server.js

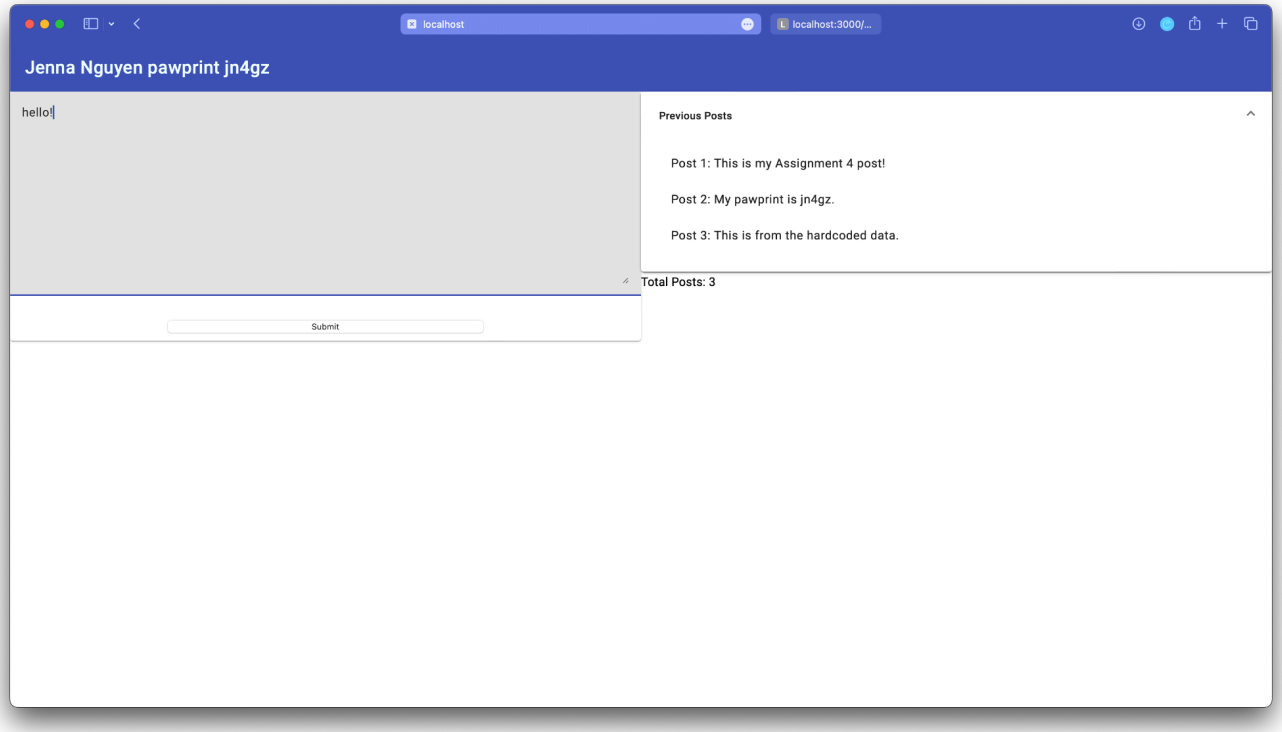


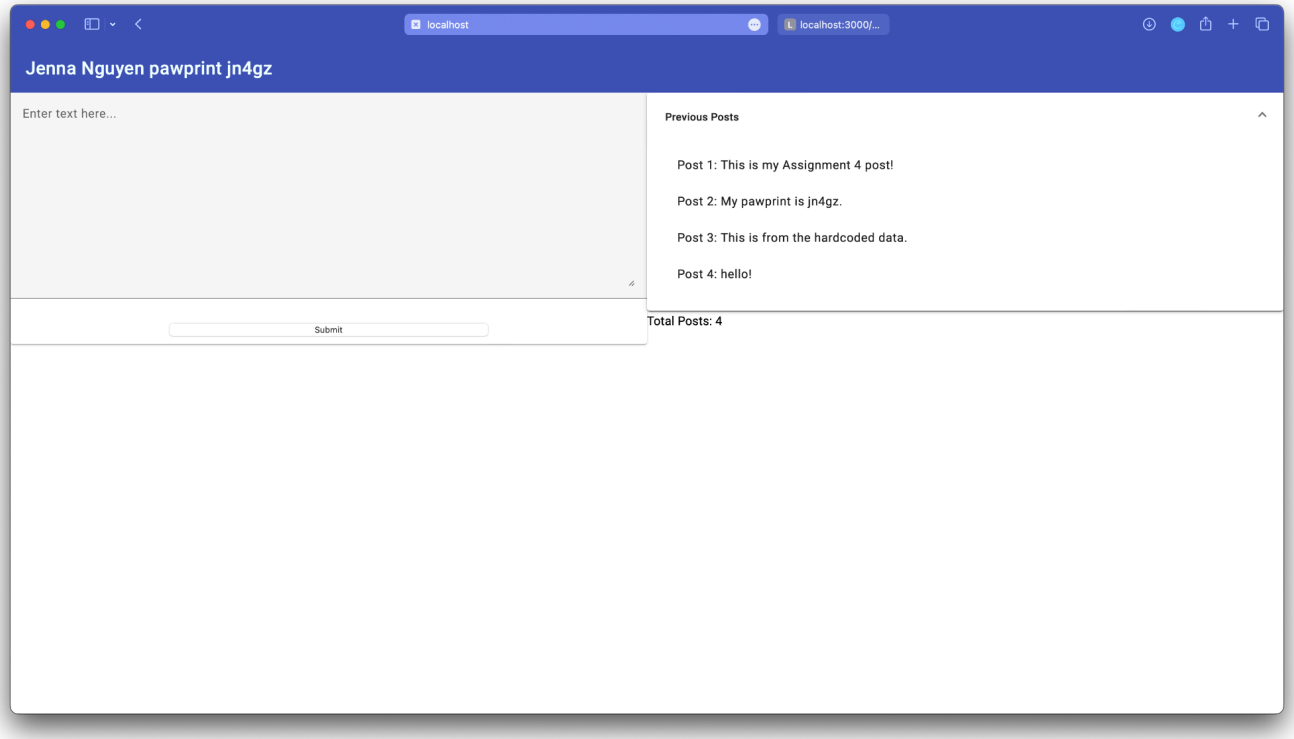
node-server > JS server.js > ...

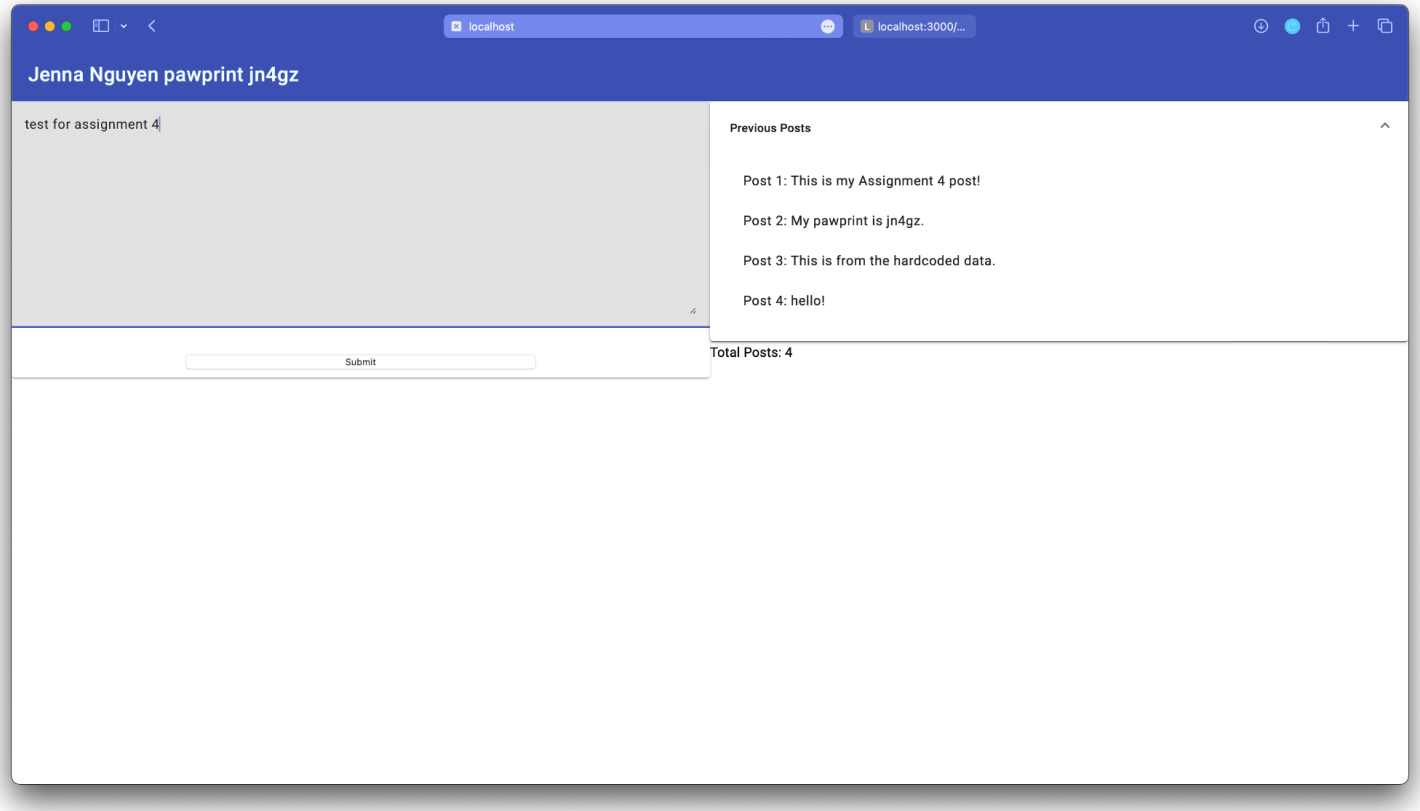
```
1   const express = require('express');
2   const cors = require('cors');
3
4   const app = express();
5   const port = 3000;
6
7   app.use(cors());
8
9   // hardcoded data
10  const posts = [
11      {title: 'Post 1', content: 'This is my Assignment 4 post!'},
12      {title: 'Post 2', content: 'My pawprint is jn4gz.'},
13      {title: 'Post 3', content: "This is from the hardcoded data."}
14  ]
15
16  // route to send post data to the frontend
17  app.get('/posts', (req, res) => {
18      res.json(posts);
19  });
20
21  // start server
22  app.listen(port, () => {
23      console.log(`Server is running on http://localhost:${port}`);
24  });
```

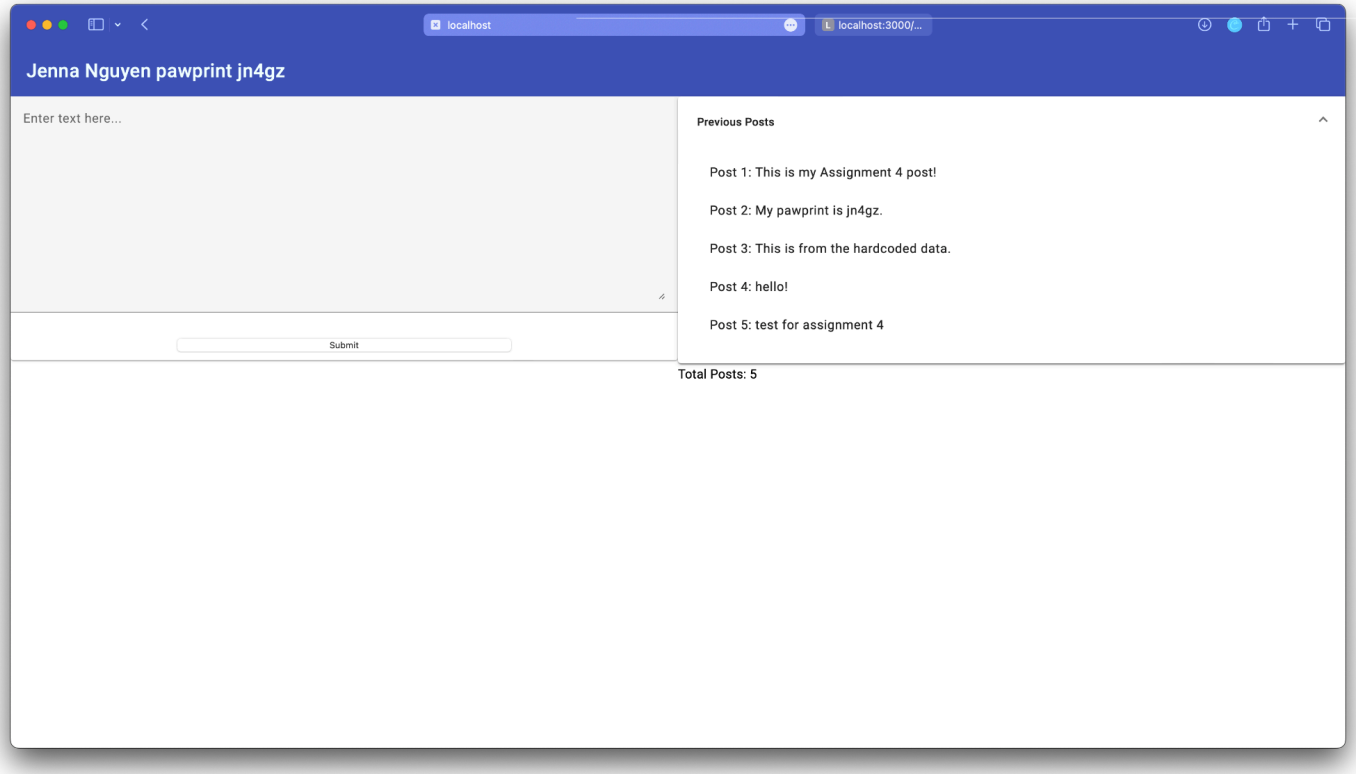
Demo shows hardcoded data and allows user to create new posts that are added to list






localhost
localhost:3000/...
Jenna Nguyen pawprint jn4gz
Enter text here...
Submit
Previous Posts
Post 1: This is my Assignment 4 post!
Post 2: My pawprint is jn4gz.
Post 3: This is from the hardcoded data.
Post 4: hello!
Total Posts: 4


localhost
localhost:3000/...
Jenna Nguyen pawprint jn4gz
test for assignment 4
Submit
Previous Posts
Post 1: This is my Assignment 4 post!
Post 2: My pawprint is jn4gz.
Post 3: This is from the hardcoded data.
Post 4: hello!
Total Posts: 4


localhost
localhost:3000/...
Jenna Nguyen pawprint jn4gz
Enter text here...
Submit
Previous Posts
Post 1: This is my Assignment 4 post!
Post 2: My pawprint is jn4gz.
Post 3: This is from the hardcoded data.
Post 4: hello!
Post 5: test for assignment 4
Total Posts: 5