Name: Kurt Pagal
NSID: zja641
Student ID: 11314773
Course: CMPT-353
Instructor: Ralph Deters

I made use of alert for testing the program, especially when testing initializing the database, adding the post to the database, and retrieving the posts from the database.

```
// Import required packages
import React from 'react';
import axios from 'axios';

export const Landing = () => {
    // A function to initialize and create a table for the database
    const initDatabase = () => {
        axios.get('http://localhost:81/init')
        .then(response => {
            console.log(response.data);
        }).then(alert("Created the database"))
        .catch(error => console.error(error));
    };

    initDatabase();     // insitial call to the initialize function

    return (
        <div>
            <h3>Welcome!</h3>
            <p>Create a new post or see all the previous posts you have created!</p>
        </div>
    );
}
```

```
// Import required packages
import "./AddPosts.css";
import React, { useState } from 'react';
import axios from 'axios';

export const AddPosts = ({ onAddPost }) => {
    // Set use state variables to an empty string
    const [ topic, setTopic ] = useState('');
    const [ data, setData ] = useState('');

    // Function to add the post to the database
    const addPost = () => {
        // Connect to the server
        axios.post('http://localhost:81/addPost', { topic, data })
        .then(response => {
            console.log(response.data);
            onAddPost();
            setTopic('');
            setData('');
        })
        .then(alert('Added post'))
        .catch( error => console.error(error) );
    };
```

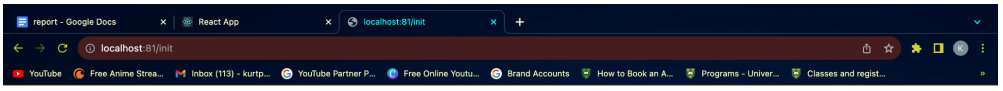
```
// Import required packages
import './GetPosts.css';
import React, { useState, useEffect } from 'react';
import axios from 'axios';

export const GetPosts = () => {
    // Set use state variables to an empty array
    const [ posts, setPosts ] = useState([]);

    // useEffect() function to get posts from the database
    useEffect(() => {
        axios.get('http://localhost:81/getPosts')
            .then(response => setPosts(response.data.posts))
            .then(alert("Grabing from the database"))
            .catch(error => console.error(error));
    }, []);

    return (
        <div className='container'>
            <h3>Posts</h3>
            <div className='info'>
                {/* Map the items from posts to a div style */}
                {posts.map(post => (
                    <div>
                        <h4 key={post.id}>{post.id}: {post.topic}</h4>
                        <li key={post.id}>{post.data}</li>
                    </div>
                ))}
            </div>
        </div>
    );
};
```

For the server side to test if the method was a success or a fail, I send to the server a status 500 for a failure and a simple success message when the method passed.

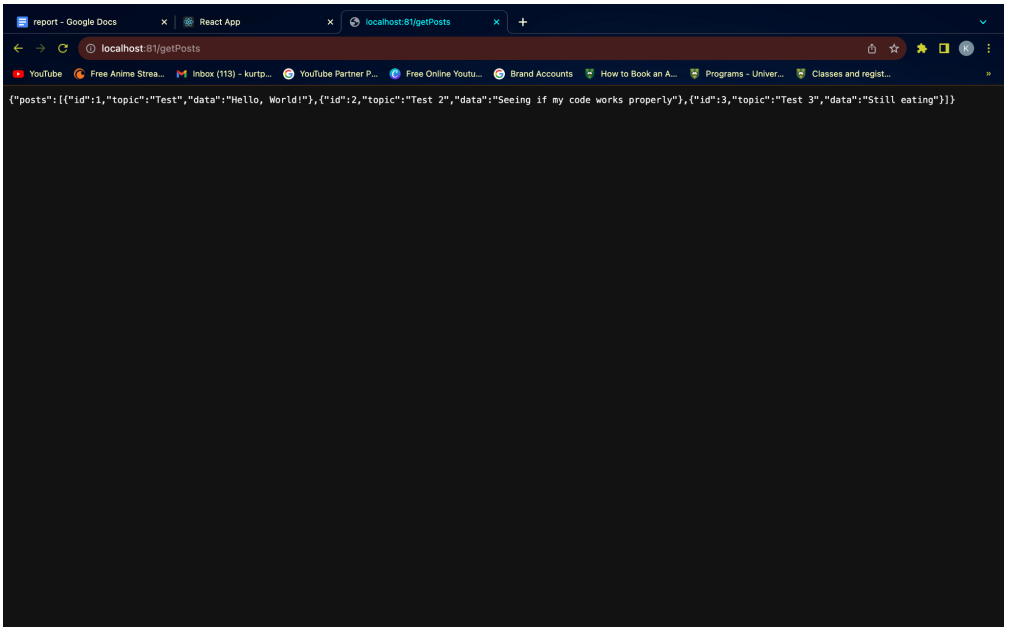
```
// Method to create database postdb
app.get("/init", (req, res) => {
    connection.query(`CREATE DATABASE IF NOT EXISTS postdb`, function(createError, createResults) {
        if (createError) {
            console.log(createError);
            res.status(500).send("Error creating the database");
        }
    });

    // Use the created database
    connection.query(`USE postdb`, function(useError, useResults) {
        if (useError) {
            console.log(useError);
            res.status(500).send("Error using the database");
        }

        // Create a query
        const createTableQuery = `CREATE TABLE IF NOT EXISTS posts
            (id int unsigned NOT NULL auto_increment,
            topic varchar(100) NOT NULL,
            data varchar(100) NOT NULL,
            PRIMARY KEY (id))`;

        // Query for creating the databse
        connection.query(createTableQuery, function (createError, createResults) {
            if (createError) {
                console.log(createError);
                res.status(500).send("Error creating the table");
            }

            // Delete all previous rows in the table if it already exists
            connection.query('DELETE FROM posts', function (delError, delResults) {
                if (delError) {
                    console.log(delError);
                    res.status(500).send("Error deleting existing entries");
                }

                // When deleting the elements from the table, reset the QUTO_INCREMENT
                connection.query('ALTER TABLE posts AUTO_INCREMENT = 1', function (resetError, resetResults) {
                    if (resetError) {
                        console.log(resetError);
                        res.status(500).send('Error resetting id');
                    }

                    // Notify the server that the database and table is created
                    res.send('Database and Table created');
                })
            });
        });
    });
});
```

```
// A method that adds a post to the database
app.post("/addPost", (req, res) => {
    // Get the value from the topic and data field from the page
    var topic = req.body.topic;
    var data = req.body.data;

    // Create a query
    var query = `INSERT INTO posts (topic, data) VALUES ("${topic}", "${data}")`;

    // Query to insert the topic and data into the databse
    connection.query(query, function(insertError, insertResults) {
        if (insertError) console.log(insertError);
        res.send("New post has been added");
    });
});

// A method that grabs the elements from the database
app.get("/getPosts", (req, res) => {
    // Create a query
    const databasePost = `SELECT * FROM posts`;

    // Query to grab elements from the database and send the data
    connection.query(databasePost, function(error, results) {
        if (error) console.log(error);
        res.send({ 'posts': results });
    });
});
```

To test if the server is working as intended, I went to the url in the browser and checked with the url handles to check with if the page is retrieving what the server is sending.



Database and Table created



To check if the database was created and that posts were actually added to it, I went into mysql used mysql commands to check for the database and the table posts

```
mysql> USE postdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_postdb   |
+--------------------+
| posts              |
+--------------------+
1 row in set (0.01 sec)
```

```
mysql> SELECT * FROM posts;
+----+--------+---------------------------------+
| id | topic  | data                            |
+----+--------+---------------------------------+
|  1 | Test   | Hello, World!                   |
|  2 | Test 2 | Seeing if my code works properly |
|  3 | Test 3 | Still eating                    |
+----+--------+---------------------------------+
3 rows in set (0.01 sec)
```