**REACT.JS**

Table of Contents:

    Reacts Fragments..................................................................................1

    React Portals......................................................................................... 1

    React Refs (References).........................................................................2

    What is an Effect?..................................................................................3

    React use Reducer () .............................................................................3

    React use State () ..................................................................................4

Project ………………………………………………………………………………………………5

Outputs……………………………………………………………………………………………...18

1.React Fragments:

* We can use <> and </> instead of <div> and </div>. Because, it is an empty wrapper component. It doesn’t render any real HTML element to the DOM (Document Object Model). But it fulfills React’s/JSX requirement.
* The Code Execution is faster with use of React Fragments as Compared to the div tag.
* Less utilization of memory with the use of react Fragments instead of div tag.
* You can declare Fragments in two types: -
* <> and </>
* <Fragment> and </Fragment>

2. React Portals:

* React Portal is a first-class way to render child components into a DOM node outside of the parent DOM hierarchy defined by the component hierarchy.

3. React Refs (References):

* Refs are a function provided by react to access the DOM element and react element that you might have created on your own.
* They are used in cases where we want to change the value of the child component, without using props and all.
* They also provide us with good functionality as we can use callbacks with them.
* Depending on the type of the node, the value of the ref can defer.
* The ref attribute doesn’t have instances, and can’t be used on Functional Components.

Example: Adding Refs to DOM elements.

**import** React, { Component } from 'react';     
import { render } from 'react-dom';     
class App **extends** React.Component {     
constructor(props) {     
super(props);     
this.callRef = React.createRef();     
this.addingRefInput = **this**.addingRefInput.bind(**this**);     
}     
addingRefInput() {     
this.callRef.current.focus();     
}     
render() {     
return (     
<div>     
<h1>Hello World</h1>     
<input     
type="text"     
ref={**this**.callRef} />     
<input     
type="button"     
value="Enter text"     
onClick={**this**.addingRefInput}     
/>     
</div>    
);     
}     
}     
export **default** App;

* **Output:**   
  

4. React use Effect ():

* The use Effect () hook allows you to performs side effects in your components.
* Some examples of side components are: fetching data, directly updating the DOM and timers.

5. React use Reducer ():

* The use Reducer () can be used as a replacement for use State (), if you need “more powerful state management”.
* It allows custom state logic.
* The use Reducer () hook accepts two arguments.

* How is use Reducer () is used?

Const [state, dispatch FN] = use Reducer (reducer FN, initial State, init FN);

* State= The state snapshot used in the component render
* Dispatch FN= A function that is used to dispatch a new action
* Reducer FN= A function that is triggered automatically once an action is dispatched.
* Intial State= Initial state
* Init FN= A function to set the initial state programmatically.

6. React use State ():

* The main state management tool
* Great for independent use of state.
* Great if state updates are easy and limited to a few kinds of updates.
* At the top of your component, import the use State () Hook:

import {use State} from "react";

7. Creating a new project using Handling side effects, using reducers, and using context API, Use State method, named as Cart Icon Button……

To Start with my second React.js project, I first downloaded node.js from my internet browser.

* Then I installed it in my PC and went into Command Prompt or Terminal.
* I then entered the following syntax to install npm directories:
* **npx create-react-app my-app**

* Instead of my-app, I used sample-app for the new project.
* Then I opened Visual Studio Code and Opened the My-First-App folder and then opened terminal in Visual studio code.
* Then I entered the following syntax to get work of npm:
* **Npm start**
* This above command starts the app in development mode.

**Folder Structure:**

Source: -

App.js

Index.js

Index.css

Components: -

Comments: -

CommentItem.js

CommentItem.module.css

Comments.js

Comments.module.css

CommentsList.js

CommentsList.module.css

NewCommentForm.js

NewCommentForm.module.css

Layout: -

Layout.js

Layout.module.css

MainNavigation.js

MainNavigation.module.css

Quotes: -

HighlightedQuote.js

HighLightedQuote.module.css

NoQuotesFound.js

NoQuotesFound.module.css

QuotesItem.js

QuotesItem.module.css

QuotesList.js

QuotesList.module.css

UI: -

Card.js

Card.module.css

LoadingSpinner.js

LoadingSpinner.module.css

Pages: -

AllQuotes.js

NewQuote.js

NotFound.js

QuoteDetail.js

import { Route, Switch, Redirect} from 'react-router-dom';

import AllQuotes from './pages/AllQuotes';

import NewQuote from './pages/NewQuote';

import QuoteDetail from './pages/QuoteDetail';

import Layout from './components/layout/Layout';

import NotFound from './pages/NotFound';

function App() {

  return (

    <Layout>

      <Switch>

        <Route path='/' exact>

          <Redirect to='/quotes' />

        </Route>

        <Route path='/quotes' exact>

          <AllQuotes />

        </Route>

        <Route path='/quotes/:quoteId'>

          <QuoteDetail />

        </Route>

        <Route path='/new-quote'>

          <NewQuote />

        </Route>

        <Route path='\*'>

          <NotFound />

        </Route>

      </Switch>

    </Layout>

  );

}

export default App;

import classes from '../comments/CommentItem.module.css';

const CommentItem = (props) => {

  return (

    <li className={classes.item}>

      <p>{props.text}</p>

    </li>

  );

};

export default CommentItem;

import { useState, useEffect, useCallback } from 'react';

import { useParams } from 'react-router-dom';

import classes from './Comments.module.css'

import NewCommentForm from './NewCommentForm';

import useHttp from '../../hooks/use-http';

import { getAllComments } from '../../lib/api';

import LoadingSpinner from '../ui/LoadingSpinner';

import CommentsList from './CommentsList';

const Comments = () => {

  const [isAddingComment, setIsAddingComment] = useState(false);

  const params = useParams();

  const { quoteId } = params;

  const { sendRequest, status, data: loadedComments } = useHttp(getAllComments);

  useEffect(() => {

    sendRequest(quoteId);

  }, [quoteId, sendRequest]);

  const startAddCommentHandler = () => {

    setIsAddingComment(true);

  };

  const addedCommentHandler = useCallback(() => {

    sendRequest(quoteId);

  }, [sendRequest, quoteId]);

  let comments;

  if (status === 'pending') {

    comments = (

      <div className='centered'>

        <LoadingSpinner />

      </div>

    );

  }

  if (status === 'completed' && loadedComments && loadedComments.length > 0) {

    comments = <CommentsList comments={loadedComments} />;

  }

  if (

    status === 'completed' &&

    (!loadedComments || loadedComments.length === 0)

  ) {

    comments = <p className='centered'>No comments were added yet!</p>;

  }

  return (

    <section className={classes.comments}>

      <h2>User Comments</h2>

      {!isAddingComment && (

        <button className='btn' onClick={startAddCommentHandler}>

          Add a Comment

        </button>

      )}

      {isAddingComment && (

        <NewCommentForm

          quoteId={quoteId}

          onAddedComment={addedCommentHandler}

        />

      )}

      {comments}

    </section>

  );

};

export default Comments;

import CommentItem from './CommentItem';

import classes from './CommentsList.module.css';

const CommentsList = (props) => {

  return (

    <ul className={classes.comments}>

      {props.comments.map((comment) => (

        <CommentItem key={comment.id} text={comment.text} />

      ))}

    </ul>

  );

};

export default CommentsList;

import { useRef, useEffect } from 'react';

import useHttp from '../../hooks/use-http';

import { addComment } from '../../lib/api';

import LoadingSpinner from '../ui/LoadingSpinner';

import classes from './NewCommentForm.module.css';

const NewCommentForm = (props) => {

  const commentTextRef = useRef();

  const { sendRequest, status, error } = useHttp(addComment);

  const { onAddedComment } = props;

  useEffect(() => {

    if (status === 'completed' && !error) {

      onAddedComment();

    }

  }, [status, error, onAddedComment]);

  const submitFormHandler = (event) => {

    event.preventDefault();

    const enteredText = commentTextRef.current.value;

    // optional: Could validate here

    sendRequest({ commentData: { text: enteredText }, quoteId: props.quoteId });

  };

  return (

    <form className={classes.form} onSubmit={submitFormHandler}>

      {status === 'pending' && (

        <div className='centered'>

          <LoadingSpinner />

        </div>

      )}

      <div className={classes.control} onSubmit={submitFormHandler}>

        <label htmlFor='comment'>Your Comment</label>

        <textarea id='comment' rows='5' ref={commentTextRef}></textarea>

      </div>

      <div className={classes.actions}>

        <button className='btn'>Add Comment</button>

      </div>

    </form>

  );

};

export default NewCommentForm;

import React from 'react';

import classes from 'react';

import MainNavigation from './MainNavigation';

const Layout = (props) => {

    return (

      <React.Fragment>

        <MainNavigation />

        <main className={classes.main}>{props.children}</main>

      </React.Fragment>

    );

  };

  export default Layout;

import { NavLink } from 'react-router-dom';

import classes from './MainNavigation.module.css';

const MainNavigation = () => {

    return (

      <header className={classes.header}>

        <div className={classes.logo}>Great Quotes</div>

        <nav className={classes.nav}>

          <ul>

            <li>

              <NavLink to='/quotes' activeClassName={classes.active}>

                All Quotes

              </NavLink>

            </li>

            <li>

              <NavLink to='/new-quote' activeClassName={classes.active}>

                Add a Quote

              </NavLink>

            </li>

          </ul>

        </nav>

      </header>

    );

  };

  export default MainNavigation;

import classes from './HighlightedQuote.module.css';

const HighlightedQuote = (props) => {

  return (

    <figure className={classes.quote}>

      <p>{props.text}</p>

      <figcaption>{props.author}</figcaption>

    </figure>

  );

};

export default HighlightedQuote;

import { Link } from 'react-router-dom';

import classes from './NoQuotesFound.module.css';

const NoQuotesFound = () => {

  return (

    <div className={classes.noquotes}>

      <p>No quotes found!</p>

      <Link className='btn' to='/new-quote'>

        Add a Quote

      </Link>

    </div>

  );

};

export default NoQuotesFound;

import { Fragment, useRef, useState } from 'react';

import { Prompt } from 'react-router-dom';

import Card from '../ui/Card';

import LoadingSpinner from '../ui/LoadingSpinner';

import classes from './QuoteForm.module.css';

const QuoteForm = (props) => {

  const [isEntering, setIsEntering] = useState(false);

  const authorInputRef = useRef();

  const textInputRef = useRef();

  function submitFormHandler(event) {

    event.preventDefault();

    const enteredAuthor = authorInputRef.current.value;

    const enteredText = textInputRef.current.value;

    // optional: Could validate here

    props.onAddQuote({ author: enteredAuthor, text: enteredText });

  }

  const finishEnteringHandler = () => {

    setIsEntering(false);

  };

  const formFocusedHandler = () => {

    setIsEntering(true);

  };

  return (

    <Fragment>

      <Prompt

        when={isEntering}

        message={(location) =>

          'Are you sure you want to leave? All your entered data will be lost!'

        }

      />

      <Card>

        <form

          onFocus={formFocusedHandler}

          className={classes.form}

          onSubmit={submitFormHandler}

        >

          {props.isLoading && (

            <div className={classes.loading}>

              <LoadingSpinner />

            </div>

          )}

          <div className={classes.control}>

            <label htmlFor='author'>Author</label>

            <input type='text' id='author' ref={authorInputRef} />

          </div>

          <div className={classes.control}>

            <label htmlFor='text'>Text</label>

            <textarea id='text' rows='5' ref={textInputRef}></textarea>

          </div>

          <div className={classes.actions}>

            <button onClick={finishEnteringHandler} className='btn'>Add Quote</button>

          </div>

        </form>

      </Card>

    </Fragment>

  );

};

export default QuoteForm;

import { Link } from 'react-router-dom';

import classes from './Quoteitem.module.css';

const QuoteItem = (props) => {

  return (

    <li className={classes.item}>

      <figure>

        <blockquote>

          <p>{props.text}</p>

        </blockquote>

        <figcaption>{props.author}</figcaption>

      </figure>

      <Link className='btn' to={`/quotes/${props.id}`}>

        View Fullscreen

      </Link>

    </li>

  );

};

export default QuoteItem;

import { Fragment } from 'react';

import { useHistory, useLocation } from 'react-router-dom';

import QuoteItem from './Quoteitem';

import classes from './QuoteList.module.css';

const sortQuotes = (quotes, ascending) => {

  return quotes.sort((quoteA, quoteB) => {

    if (ascending) {

      return quoteA.id > quoteB.id ? 1 : -1;

    } else {

      return quoteA.id < quoteB.id ? 1 : -1;

    }

  });

};

const QuoteList = (props) => {

  const history = useHistory();

  const location = useLocation();

  const queryParams = new URLSearchParams(location.search);

  const isSortingAscending = queryParams.get('sort') === 'asc';

  const sortedQuotes = sortQuotes(props.quotes, isSortingAscending);

  const changeSortingHandler = () => {

    history.push({

      pathname: location.pathname,

      search: `?sort=${(isSortingAscending ? 'desc' : 'asc')}`

    });

  };

  return (

    <Fragment>

      <div className={classes.sorting}>

        <button onClick={changeSortingHandler}>

          Sort {isSortingAscending ? 'Descending' : 'Ascending'}

        </button>

      </div>

      <ul className={classes.list}>

        {sortedQuotes.map((quote) => (

          <QuoteItem

            key={quote.id}

            id={quote.id}

            author={quote.author}

            text={quote.text}

          />

        ))}

      </ul>

    </Fragment>

  );

};

export default QuoteList;

import { useReducer, useCallback } from 'react';

function httpReducer(state, action) {

  if (action.type === 'SEND') {

    return {

      data: null,

      error: null,

      status: 'pending',

    };

  }

  if (action.type === 'SUCCESS') {

    return {

      data: action.responseData,

      error: null,

      status: 'completed',

    };

  }

  if (action.type === 'ERROR') {

    return {

      data: null,

      error: action.errorMessage,

      status: 'completed',

    };

  }

  return state;

}

function useHttp(requestFunction, startWithPending = false) {

  const [httpState, dispatch] = useReducer(httpReducer, {

    status: startWithPending ? 'pending' : null,

    data: null,

    error: null,

  });

  const sendRequest = useCallback(

    async function (requestData) {

      dispatch({ type: 'SEND' });

      try {

        const responseData = await requestFunction(requestData);

        dispatch({ type: 'SUCCESS', responseData });

      } catch (error) {

        dispatch({

          type: 'ERROR',

          errorMessage: error.message || 'Something went wrong!',

        });

      }

    },

    [requestFunction]

  );

  return {

    sendRequest,

    ...httpState,

  };

}

export default useHttp;

const FIREBASE\_DOMAIN = 'https://react-prep-67068-default-rtdb.firebaseio.com';

export async function getAllQuotes() {

  const response = await fetch(`${FIREBASE\_DOMAIN}/quotes.json`);

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not fetch quotes.');

  }

  const transformedQuotes = [];

  for (const key in data) {

    const quoteObj = {

      id: key,

      ...data[key],

    };

    transformedQuotes.push(quoteObj);

  }

  return transformedQuotes;

}

export async function getSingleQuote(quoteId) {

  const response = await fetch(`${FIREBASE\_DOMAIN}/quotes/${quoteId}.json`);

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not fetch quote.');

  }

  const loadedQuote = {

    id: quoteId,

    ...data,

  };

  return loadedQuote;

}

export async function addQuote(quoteData) {

  const response = await fetch(`${FIREBASE\_DOMAIN}/quotes.json`, {

    method: 'POST',

    body: JSON.stringify(quoteData),

    headers: {

      'Content-Type': 'application/json',

    },

  });

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not create quote.');

  }

  return null;

}

export async function addComment(requestData) {

  const response = await fetch(`${FIREBASE\_DOMAIN}/comments/${requestData.quoteId}.json`, {

    method: 'POST',

    body: JSON.stringify(requestData.commentData),

    headers: {

      'Content-Type': 'application/json',

    },

  });

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not add comment.');

  }

  return { commentId: data.name };

}

export async function getAllComments(quoteId) {

  const response = await fetch(`${FIREBASE\_DOMAIN}/comments/${quoteId}.json`);

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not get comments.');

  }

  const transformedComments = [];

  for (const key in data) {

    const commentObj = {

      id: key,

      ...data[key],

    };

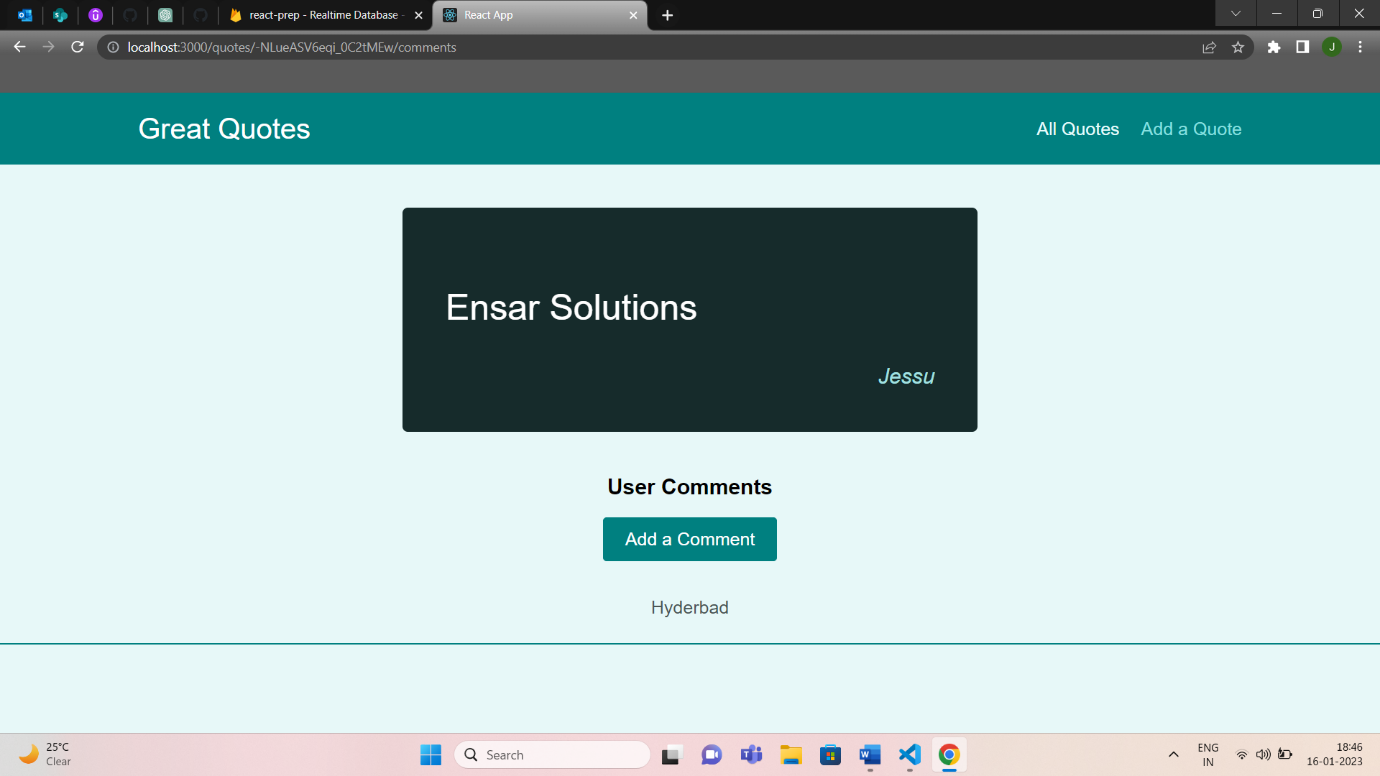
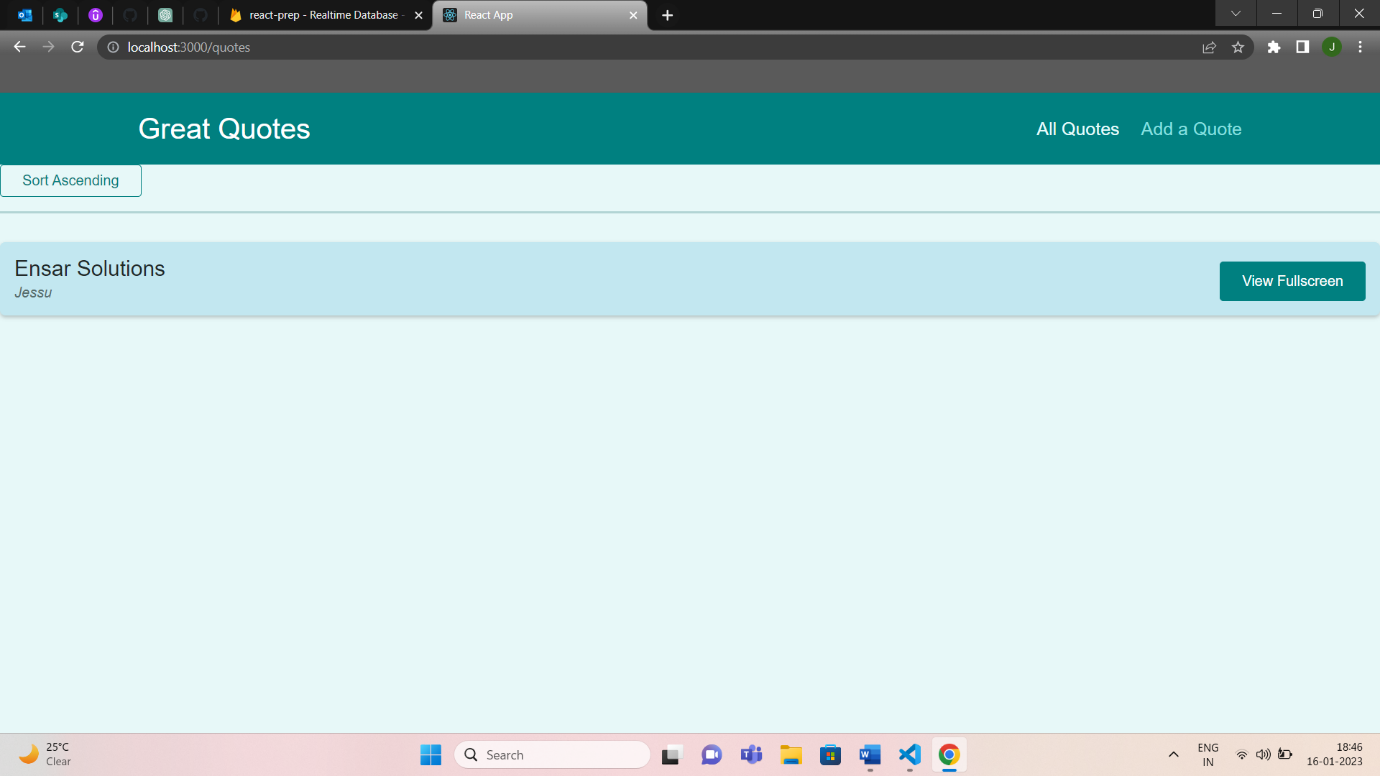
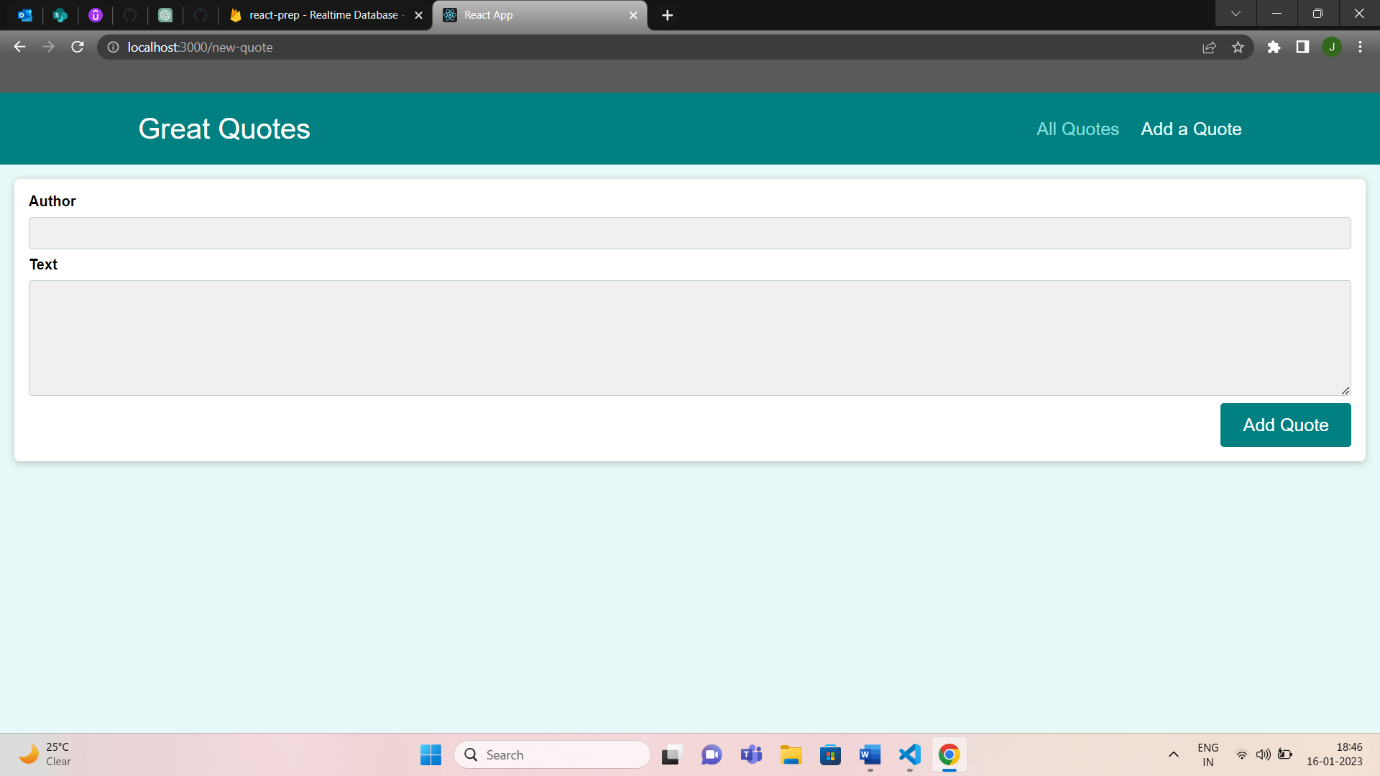
    transformedComments.push(commentObj);

  }

  return transformedComments;

}

Outputs: -



Thank You.