**Exercise 4**

**OBJECTIVES**

1. **Explain the need and Benefits of component life cycle**

The component lifecycle helps manage the different phases a component goes through—from creation to removal. It allows developers to perform actions at specific stages like fetching data, updating the DOM, or cleaning up. Enables better memory management by freeing up unused resources. Improves application performance by controlling re-renders and updates efficiently. Enhances debugging and maintenance by knowing when and why components behave a certain way.

1. **Identify various life cycle hook methods**

**constructor()** – Initializes state and binds methods.

**componentDidMount()** – Invoked once after the component is mounted; used for API calls or DOM operations.

**shouldComponentUpdate()** – Determines if a component should re-render on state/prop changes.

**componentDidUpdate()** – Runs after component updates; used to react to prop/state changes.

**componentWillUnmount()** – Cleanup tasks before the component is destroyed (like removing event listeners).

1. **List the sequence of steps in rendering a component**

**Constructor**: Initializes state and props.

**Render**: Returns the JSX to describe the UI.

**React Updates the DOM** based on JSX.

**componentDidMount**: Runs after the component is added to the DOM.

**On updates**, React runs shouldComponentUpdate, then render, then componentDidUpdate.

**HANDS ON PRACTICE**

1. **Create a new react application using *create-react-app* tool with the name as “blogapp”**

npx create-react-app blogapp

1. **Create a new file named as Post.js in src folder with following properties**

**Post.js**

import React from 'react';

function Post({title, body})

{

  return(

    <div>

      <h2>{title}</h2>

      <p>{body}</p>

    </div>

  );

}

export default Post;

1. **Create a new class based component named as Posts inside Posts.js file**

**Posts.js**

import react, {Component} from 'react';

import Post from './Post';

class Posts extends Component{

  constructor(props)

  {

    super(props);

    this.state = {

      posts: [],

    };

    this.loadPosts = this.loadPosts.bind(this);

  }

  loadPosts = () => {

    fetch('https://jsonplaceholder.typicode.com/posts')

    .then(response => response.json())

    .then(data => this.setState({posts: data}))

    .catch(error => {

      throw new Error("Failed to fetch posts");

    })

  }

  componentDidMount(){

    this.loadPosts();

  }

  componentDidCatch(error, info)

  {

    alert("Error occured" + error.message);

  }

  render()

  {

    return(

      <div>

        <h1>Posts</h1>

        {this.state.posts.map(post => (

          <Post key = {post.id} title = {post.title} body = {post.body}/>

        ))}

      </div>

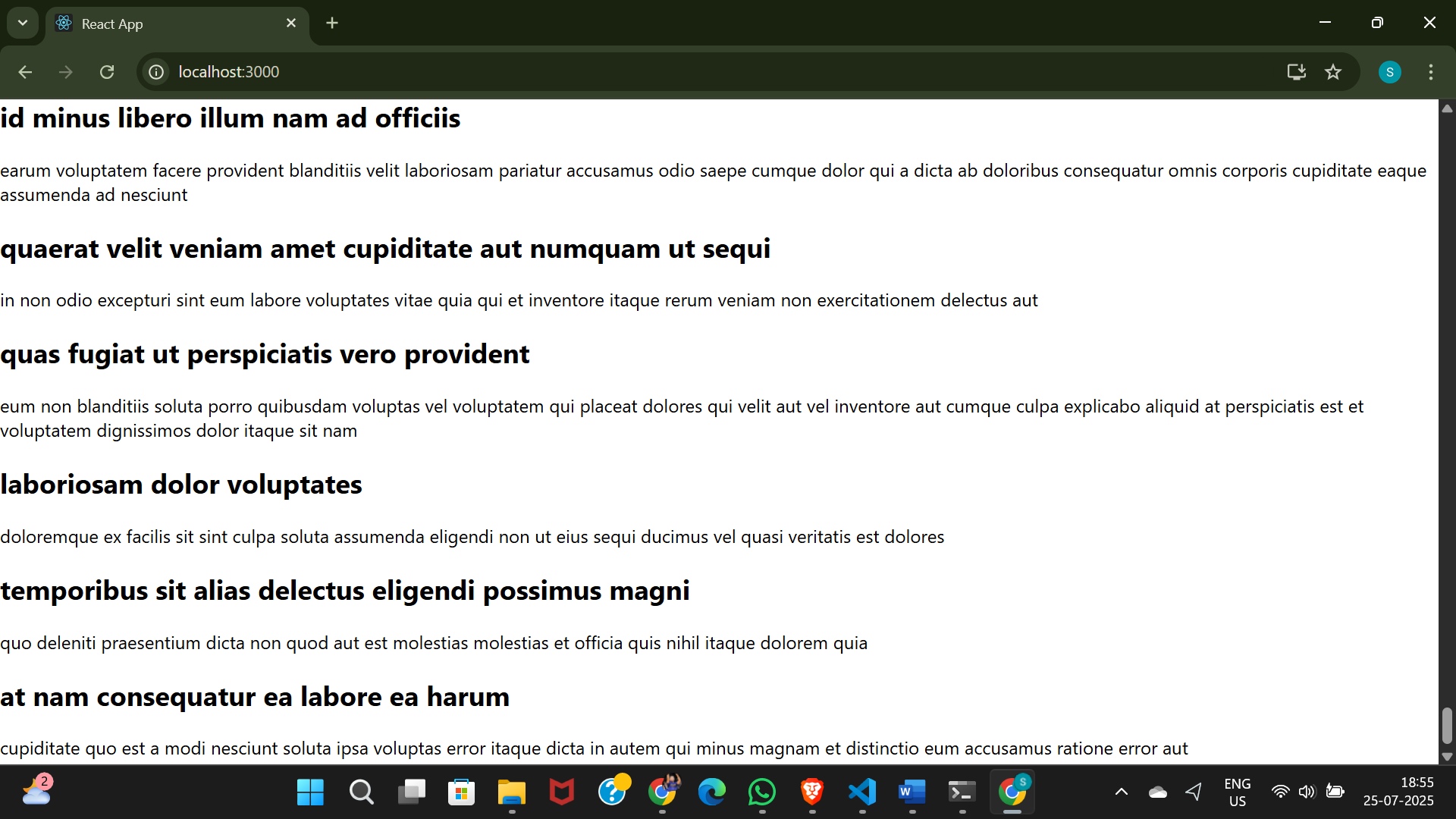
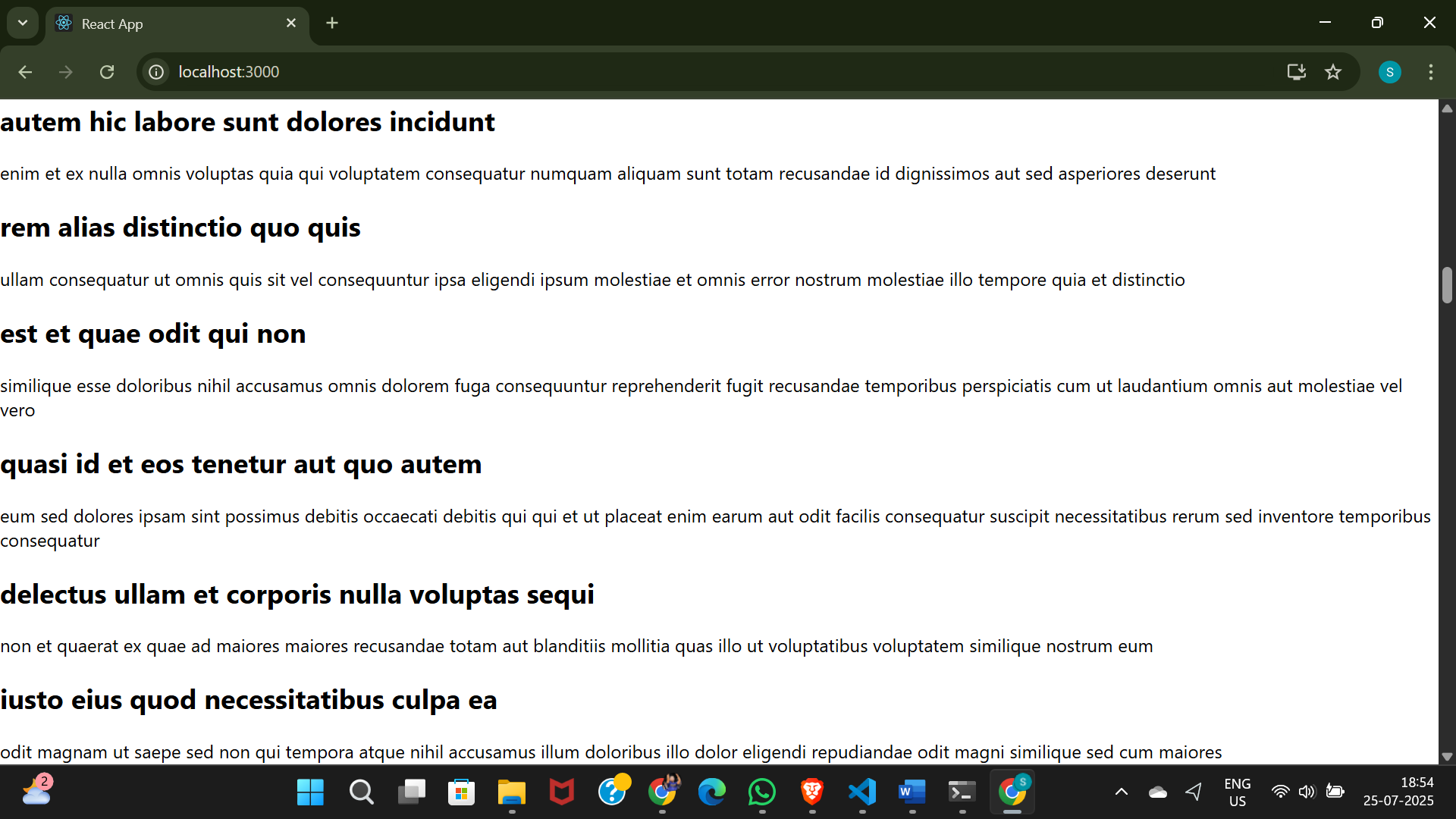
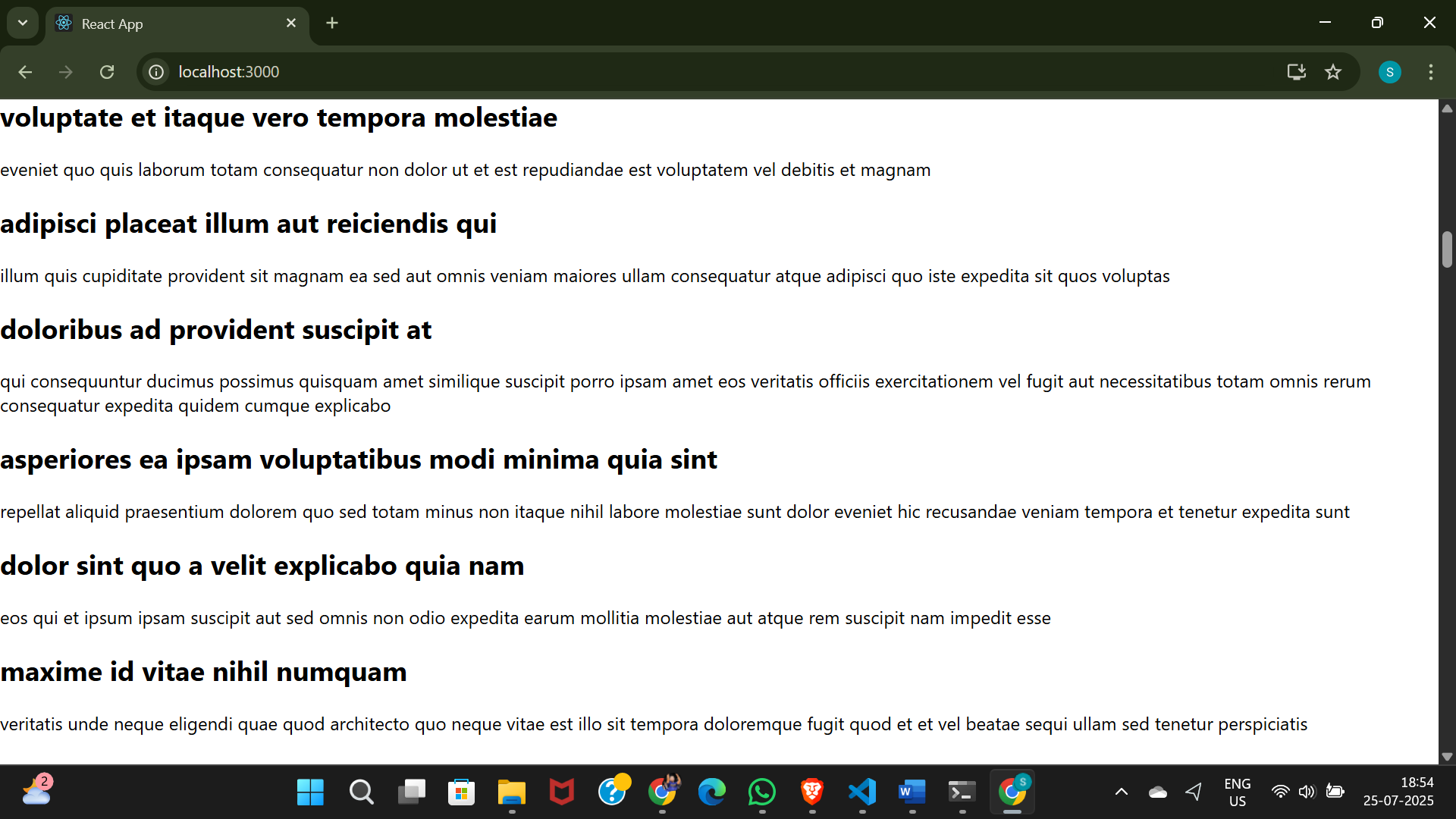
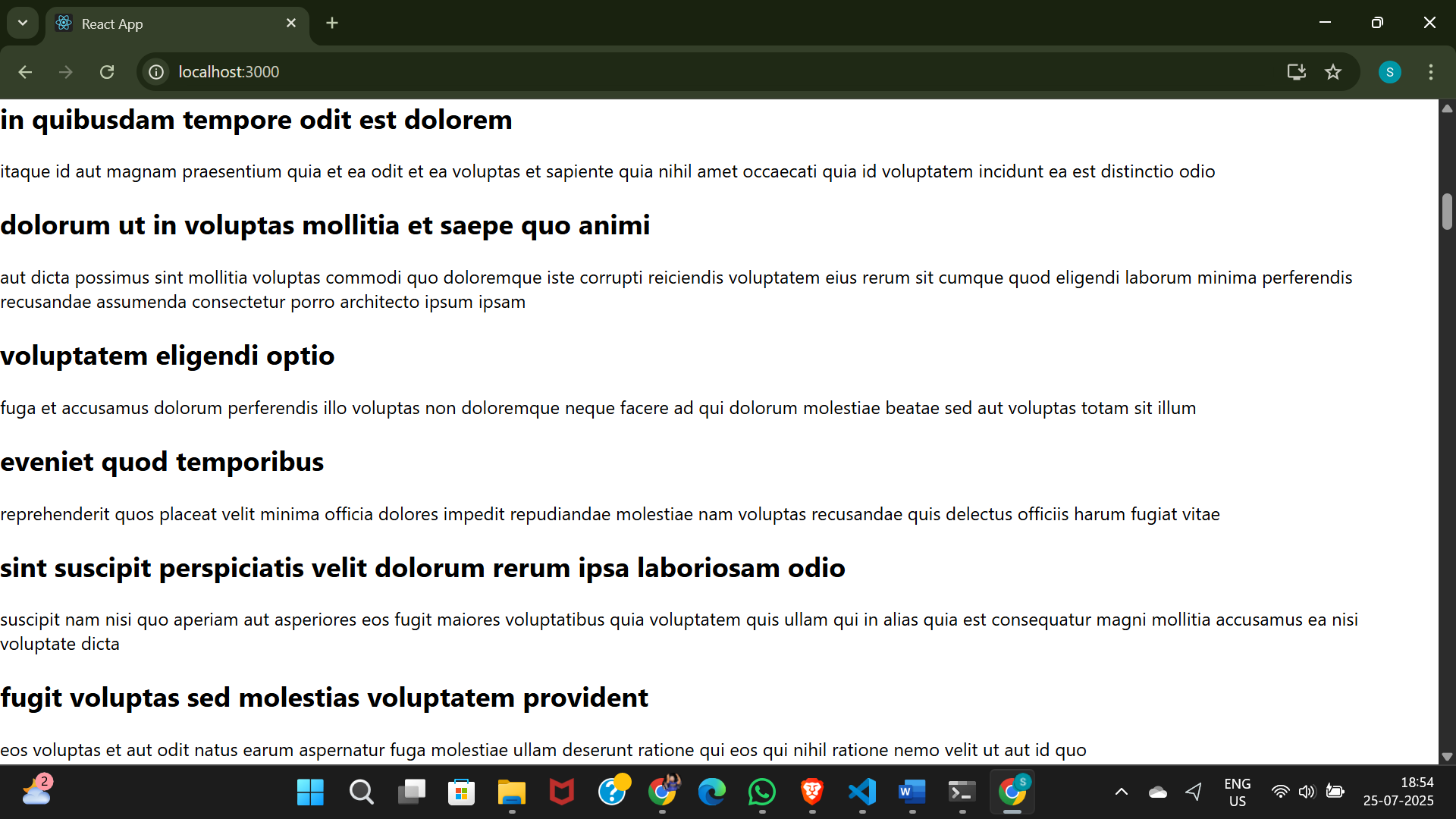
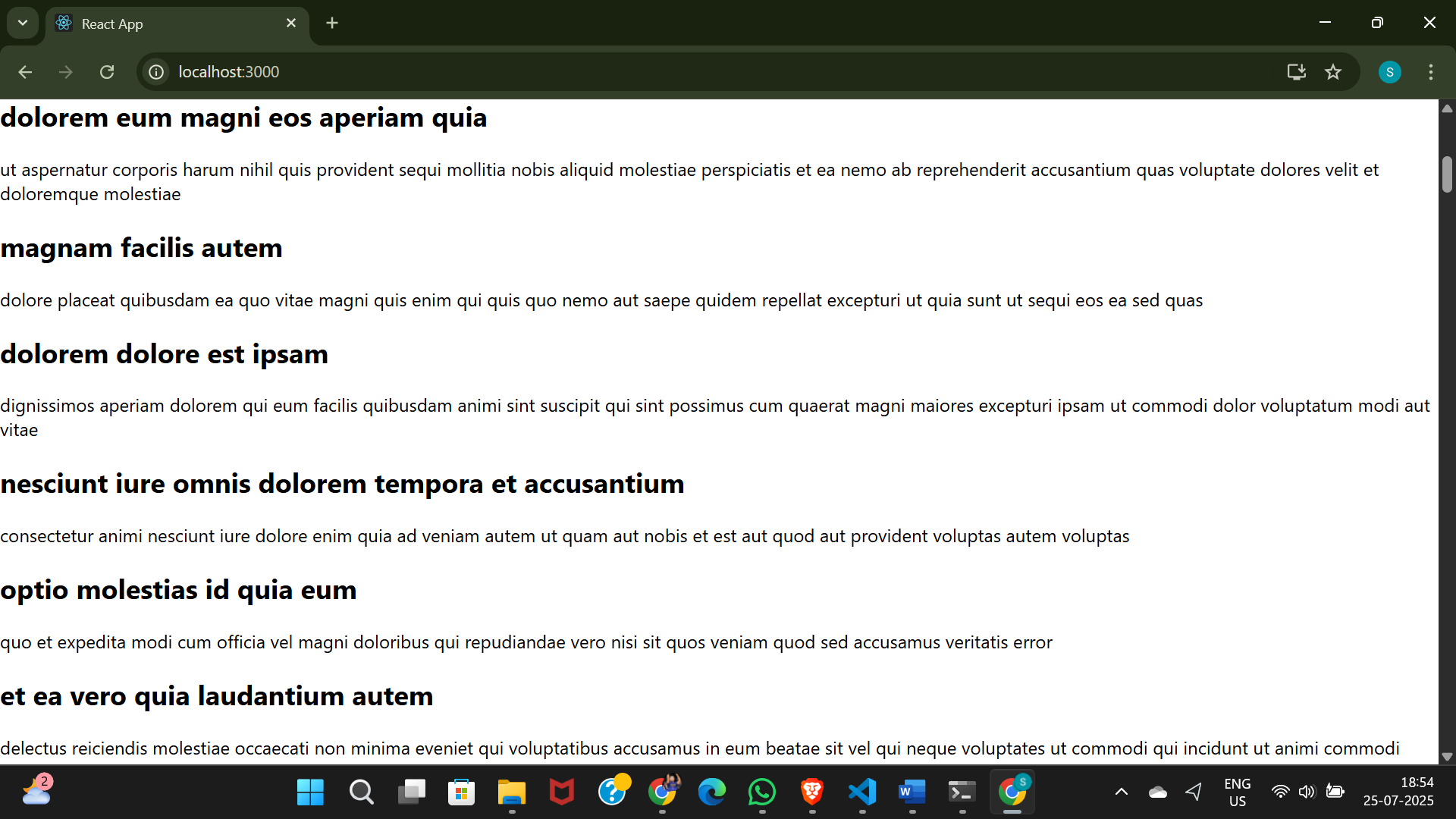
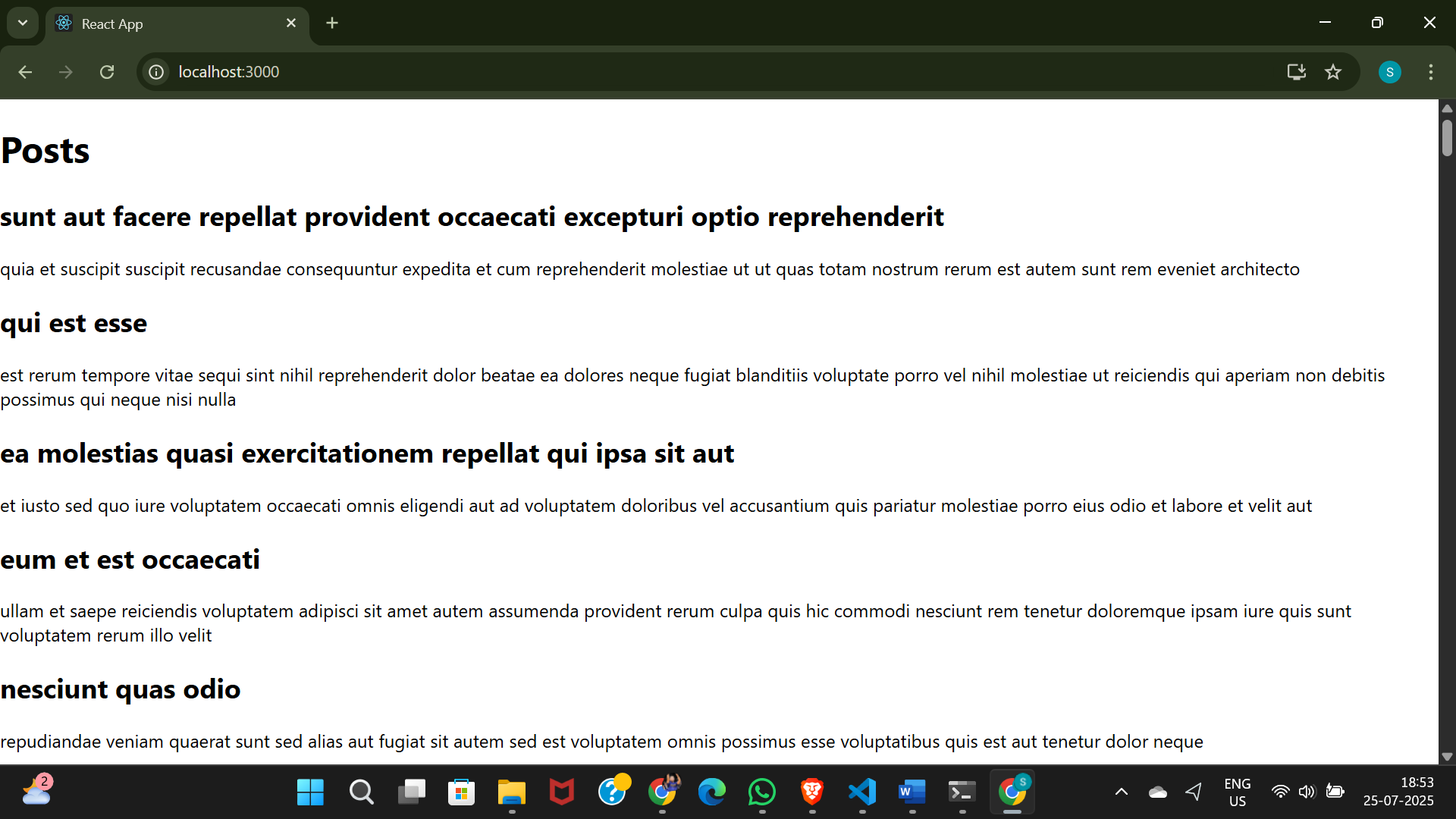
    );

  }

}

export default Posts;

1. **Output**

****