EXPERIMENT NO. - 02

Roll No.	24
Name	Chaitanya Dinesh Dhayarkar
Class	D15B
Subject	Full Stack Development
Lab Outcome	Experiment based on React Hooks (useEffect, useContext, custom hooks)
Date of Performance/ Submission	
Signature & Grades	

AIM: Experiment based on React Hooks (useEffect, useContext, custom hooks)

THEORY:

1.Introduction

React Hooks were introduced in **React 16.8** to allow developers to use **state** and other React features without writing class components.
They make components more readable, reusable, and maintainable.

Among the commonly used hooks:

- **useEffect** → For handling side effects (API calls, timers, DOM updates).
- **useContext** → For accessing global data without prop drilling.
- **Custom Hooks** → For reusing stateful logic across multiple components.

This experiment demonstrates the usage of these hooks in a React application.

2.Key Concepts:

Theory

1. useEffect Hook

- The useEffect hook is used to perform **side effects** in functional components.
- Side effects include tasks like:
    - Fetching data from an API
    - Updating the DOM directly
    - Setting up subscriptions or timers

```
useEffect(() => {
   // code to run
   return () => {
      // cleanup (optional)
   };
}, [dependencies]);
```

The **dependency array** determines when the effect runs:

- [] → runs only once (on mount).
- [state] → runs when that state changes.

- No array → runs on every render.

2. useContext Hook

- The useContext hook allows you to consume values from **React Context API** directly without passing props manually at every level (prop drilling).
- Useful for **global states** such as themes, authentication, language preferences.

Steps:

1. Create a Context using React.createContext().
2. Wrap components with a **Context Provider**.
3. Use useContext(MyContext) inside child components to access data.

const ThemeContext = React.createContext("light");

const theme = useContext(ThemeContext);

3. Custom Hooks

- A **custom hook** is a reusable function whose name starts with "use" and may call other hooks.
- Helps in separating logic from UI.
- Example: Creating a custom hook useFetch(url) to fetch data from an API.

Benefits:

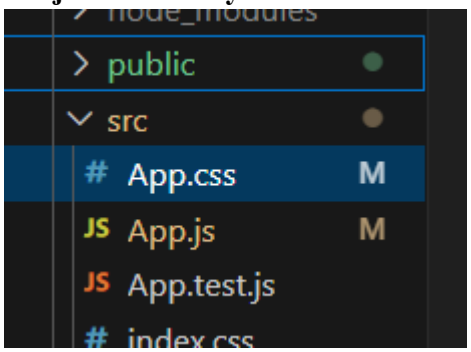
- Reusability of logic
- Cleaner and modular code
- Easier testing

2. Procedure

1. Create a React project (npx create-react-app quote-generator).
2. Store images in the public folder.
3. Create App.js and App.css files.
4. Define a Context (QuoteContext) for managing quotes and themes.
5. Build a Custom Hook (useRandomTheme) that randomly selects a quote + image.
6. Use useEffect inside the custom hook to automatically generate a quote when the app starts.
7. Provide the theme through QuoteProvider and consume it in QuoteCard using useContext.

8. Style the app using App.css

3. Project Directory Structure



CODE:

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />


    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />


    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>

  </body>
</html>
```

App.css

```
body, html, #root {
  margin: 0;
  padding: 0;
  height: 100%;
  font-family: 'Arial', sans-serif;
}

.app-container {
  height: 100vh;
  width: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 30px;
  transition: background-image 0.6s ease-in-out;

  /* Fix background image scaling */
  background-size: cover;
  background-position: center center;
  background-repeat: no-repeat;
}

.quote-card {
  background: rgba(255, 255, 255, 0.85);
  padding: 70px;
  border-radius: 20px;
  max-width: 700px;
  text-align: center;
  box-shadow: 0px 10px 30px rgba(0, 0, 0, 0.3);
```

```
}

.title {
  font-size: 3.2rem;
  margin-bottom: 20px;
  color: #333;
}

.quote {
  font-size: 2.6rem;
  margin-bottom: 30px;
  color: #444;
  font-style: italic;
}

.btn {
  padding: 14px 25px;
  font-size: 2.1rem;
  border: none;
  border-radius: 12px;
  background-color: #007bff;
  color: white;
  cursor: pointer;
  transition: 0.3s;
}

.btn:hover {
  background-color: #0056b3;
}
```

App.js

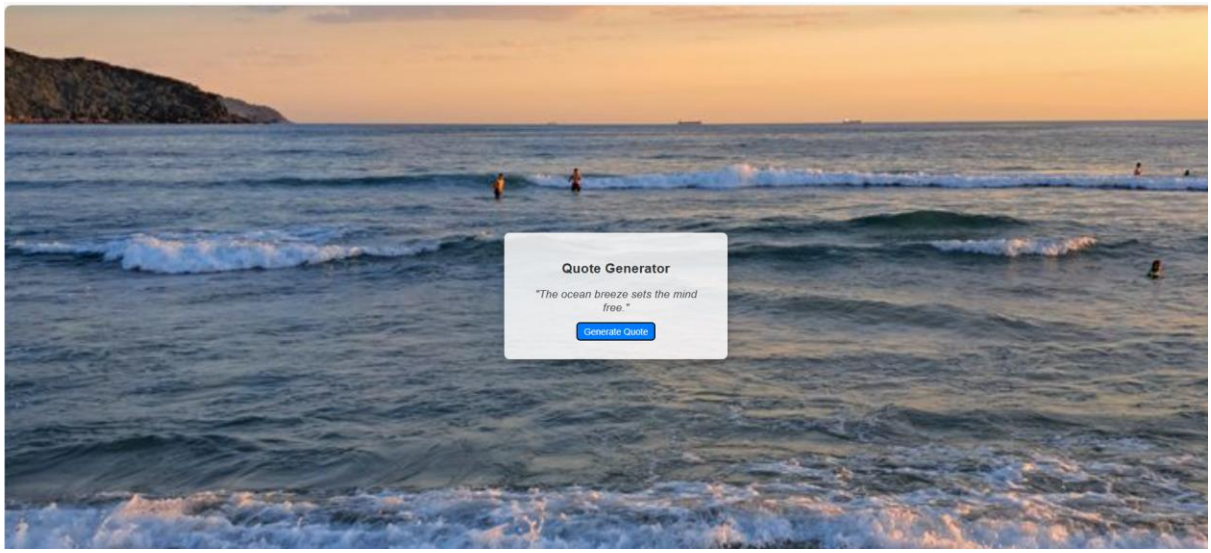
```
import React, { useState, useContext, createContext } from "react";
import "./App.css";
// Create context
const QuoteContext = createContext();
const themes = [
  {
    image: "book.png", // Books
    quote: "Between the pages of a book is a lovely place to be."
  },
  {
    image: "stars.jpg", // Galaxy
    quote: "In the middle of the night sky, dreams shine the brightest."
  },
  {
    image: "friends.jpg", // Cute cat
    quote: "Happiness is a warm little friend purring beside you."
  },
  {
    image: "ocean.jpg", // Beach
    quote: "The ocean breeze sets the mind free."
  },
  {
    image: "tree.jpg", // Forest
    quote: "Among trees, we find peace in every leaf."
  },
  {
    image: "mount.jpg", // Mountains
    quote: "Climb mountains, not so the world can see you, but so you can see the world."
  },
```

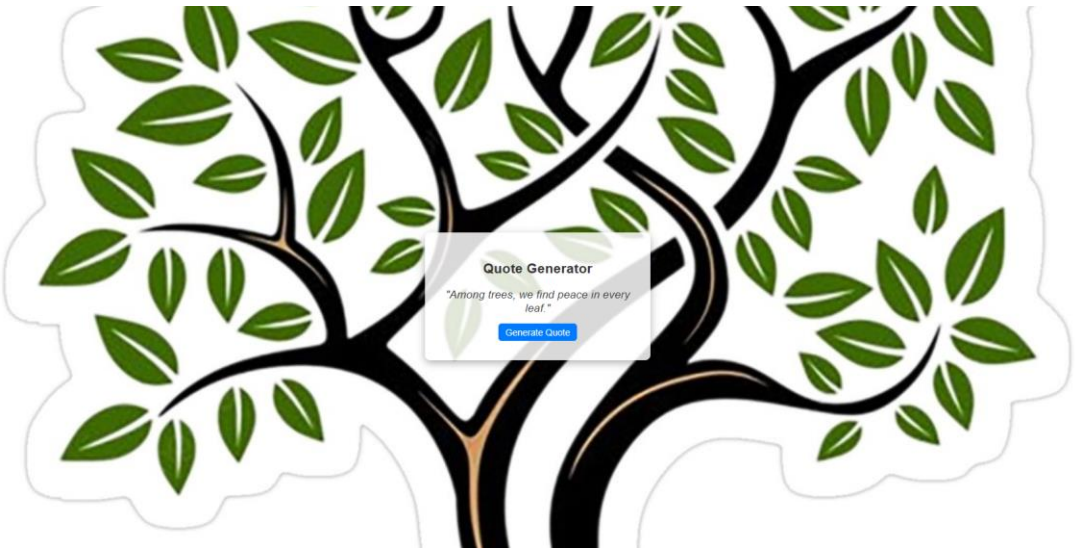
```
  {
    image: "bloom.jpg", // Flowers
    quote: "Let your soul bloom with kindness like flowers in spring."
  },
  {
    image: "sky.jpg", // Stars
    quote: "Stars can’t shine without darkness."
  },
  {
    image: "city.jpg", // City night
    quote: "Every city light tells a story of dreams alive."
  },
  {
    image: "coding.jpg", // Coding laptop
    quote: "Code your future, one line at a time."
  }
];

const QuoteProvider = ({ children }) => {
  const [theme, setTheme] = useState({
    image: "mount.jpg", // default
    quote: "Every sunrise is an invitation to brighten someone’s day."
  });

  const generateTheme = () => {
    const randomTheme = themes[Math.floor(Math.random() * themes.length)];
    setTheme(randomTheme);
  };

  return (
```

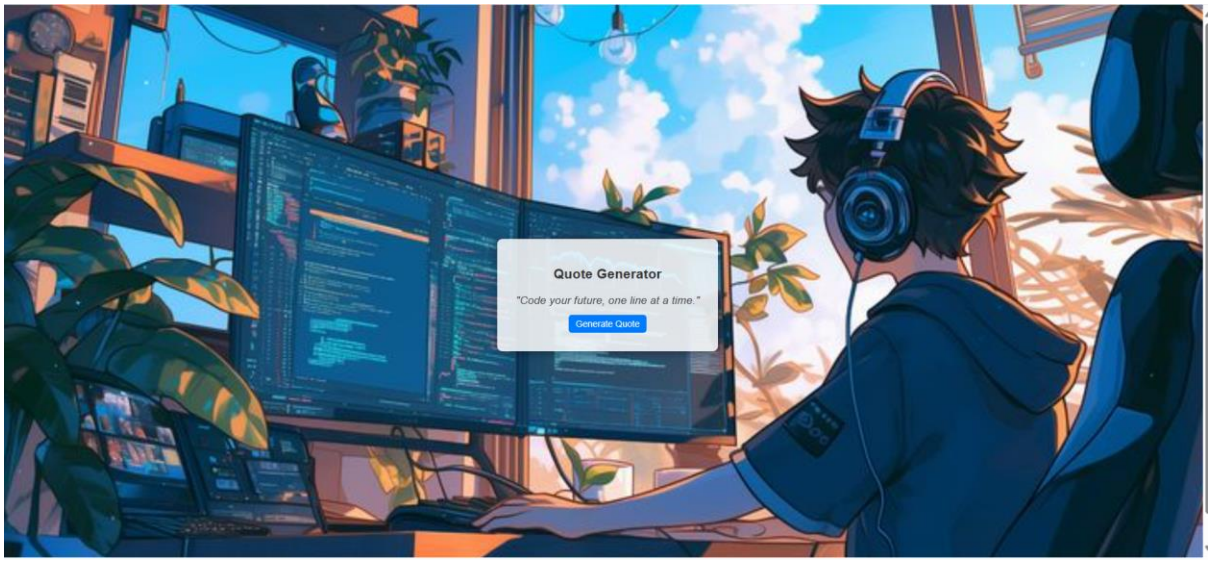
```
    <QuoteContext.Provider value={{ theme, generateTheme }}>
      {children}
    </QuoteContext.Provider>
  );
};
const QuoteCard = () => {
  const { theme, generateTheme } = useContext(QuoteContext);
  return (
    <div
      className="app-container"
      style={{
        backgroundImage: `url("${theme.image}")`
      }}
    >
      <div className="quote-card">
        <h1 className="title">Quote Generator</h1>
        <p className="quote">"{theme.quote}"</p>
        <button className="btn" onClick={generateTheme}>
          Generate Quote
        </button>
      </div>
    </div>
  );
};

function App() {
  return (
    <QuoteProvider>
      <QuoteCard />
    </QuoteProvider>
```

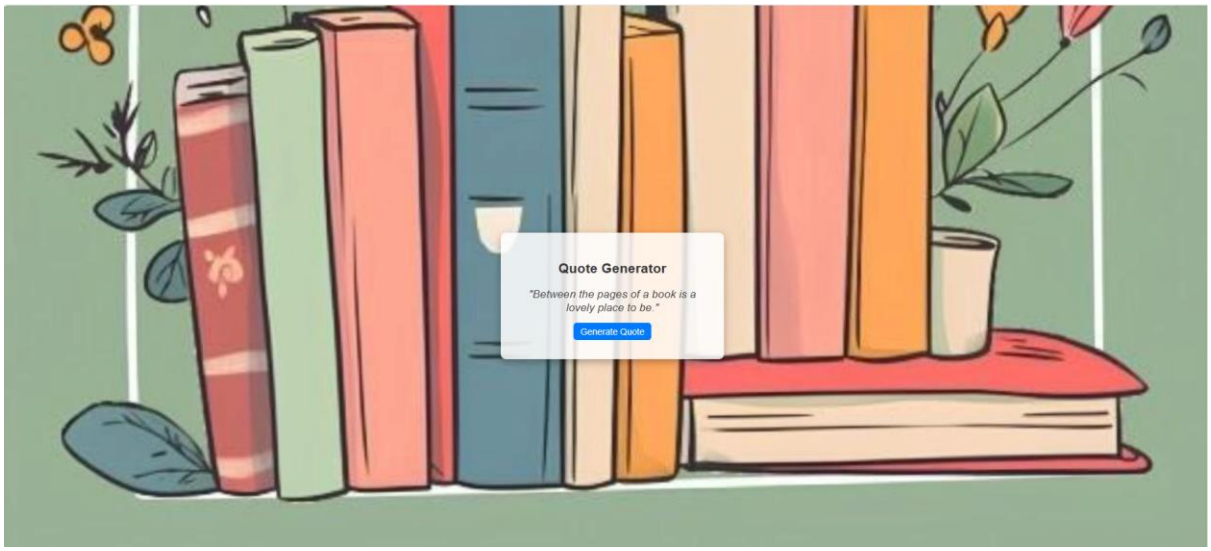
```
  );
}
export default App;
```

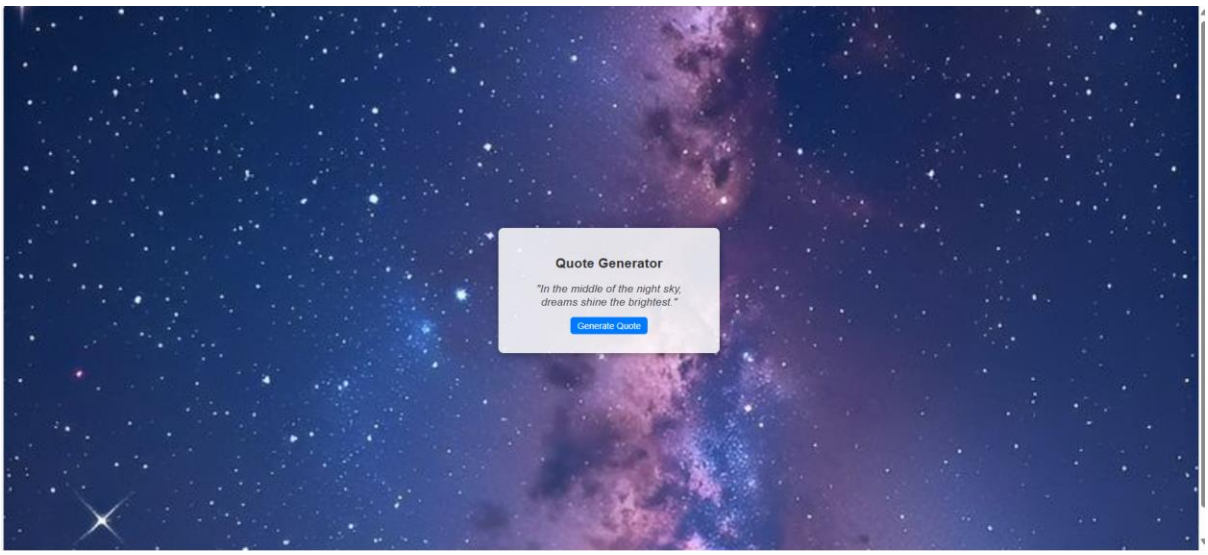
Output:

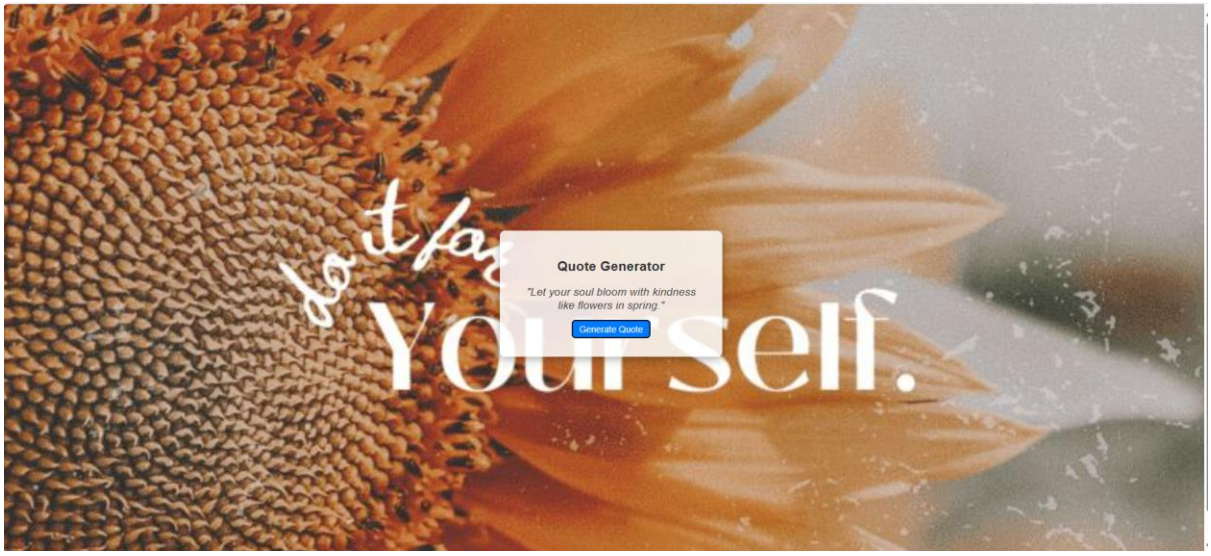


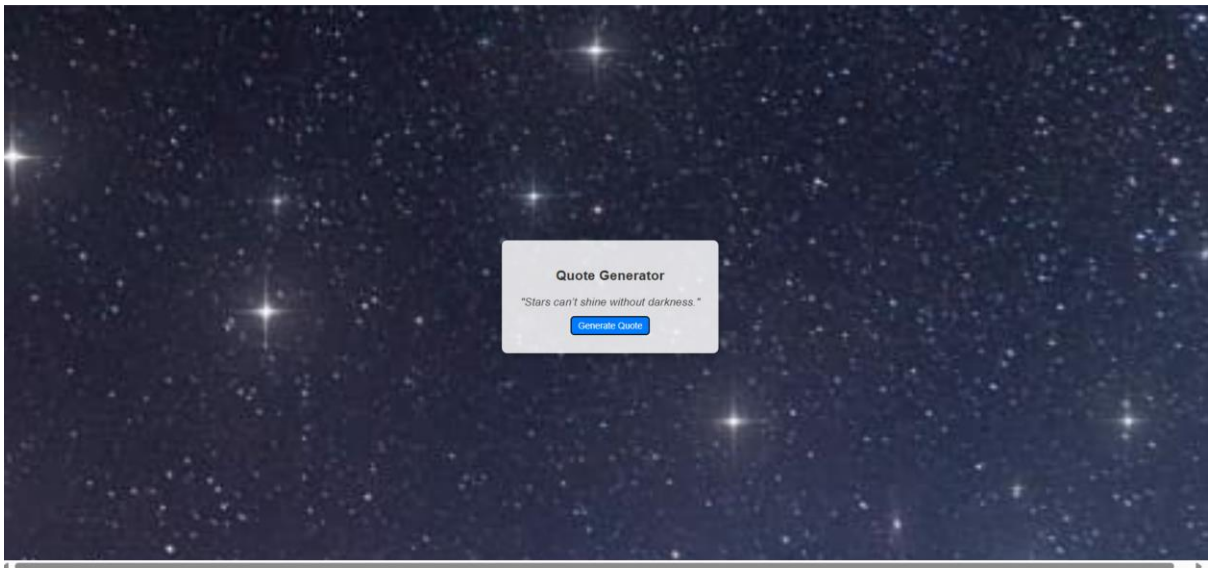



Quote Generator
"Code your future, one line at a time."
Generate Quote


Quote Generator
"Between the pages of a book is a lovely place to be."
Generate Quote


Quote Generator
"In the middle of the night sky, dreams shine the brightest."
Generate Quote





Steps to Host on Netlify:

Step 1: Prepare Your Files

1. Put your two files in a single folder on your computer:
    - index.html (rename quote_chrome_preview.html to index.html)
    - 2164670f-d591-494c-93aa-083512aa4261.png (keep the same name)
2. Open the index.html file in a code editor and update the image path:
3. <img src="2164670f-d591-494c-93aa-083512aa4261.png" alt="Quote generator screenshot">

(No /mnt/data/ — just the file name, so it works online.)

Step 2: Create a Netlify Account

1. Go to https://www.netlify.com/.
2. Sign up (you can use GitHub, Google, or email).

Step 3: Drag & Drop Deployment

1. After logging in, go to https://app.netlify.com/drop.
2. Drag your **entire folder** (with index.html + image) into the page.
3. Wait a few seconds — Netlify will deploy your site.

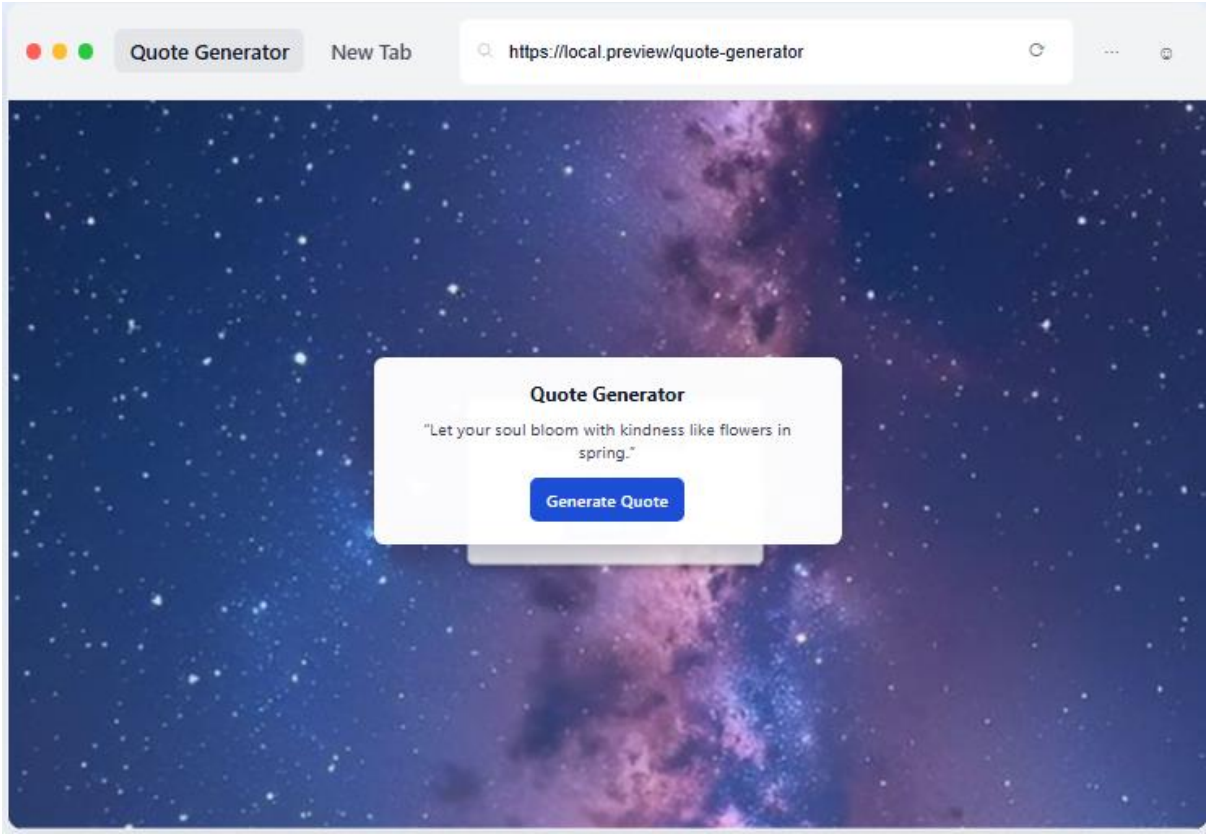
Step 4: Get Your Live Link

- You'll get a random live URL like:
- https://funny-name-12345.netlify.app/
- Click it, and your page will be live on the internet!

Optional: Customize Your URL

- In the Netlify dashboard, click **Site settings → Change site name** to set a custom subdomain, e.g.:
- https://quote-preview.netlify.app

Host On Netlify:



Conclusion:

In this experiment, we successfully built a responsive and interactive real-time weather dashboard using Tailwind CSS and the OpenWeatherMap API. Tailwind CSS's utility-first approach allowed us to rapidly design a clean, modern, and mobile-friendly user interface without writing custom CSS. The integration of OpenWeatherMap enabled live weather data retrieval, which was dynamically displayed in visually appealing gradient cards. This project demonstrated how combining Tailwind CSS with an external API can produce a professional, data-driven web application that is both functional and aesthetically engaging.