TRAINOSYS

REACTJS & TYPESCRIPT

Day 5

HYBRID
TECHNOLOGY
COURSES

Course Overview

- Introduction to ReactJS and TypeScript (Day 1)
- React Component Development (Day 2)
- React Routing and Data Fetching (Day 3)
- State Management with Redux and TypeScript (Day 4)
- Advance TypeScript and Project Development (Day 5)

Program

Day 04

- Introduction to state management with Redux
- Setting up Redux in a TypeScript-based React application
- Creating actions and reducers with TypeScript
- Managing global state with Redux in TypeScript
- Connecting React components to Redux store using TypeScript

Day 05

- Advanced TypeScript features for React development
- Error handling and debugging in TypeScript-based React applications
- Testing React components with TypeScript using Jest and React Testing Library
- Performance optimization techniques for React apps with TypeScript
- Final project development and deployment considerations
- Data Manipulation using TypeScript
- Report front end visualization

TRAINOSYS

Testing in ReactJS

HYBRID
TECHNOLOGY
COURSES

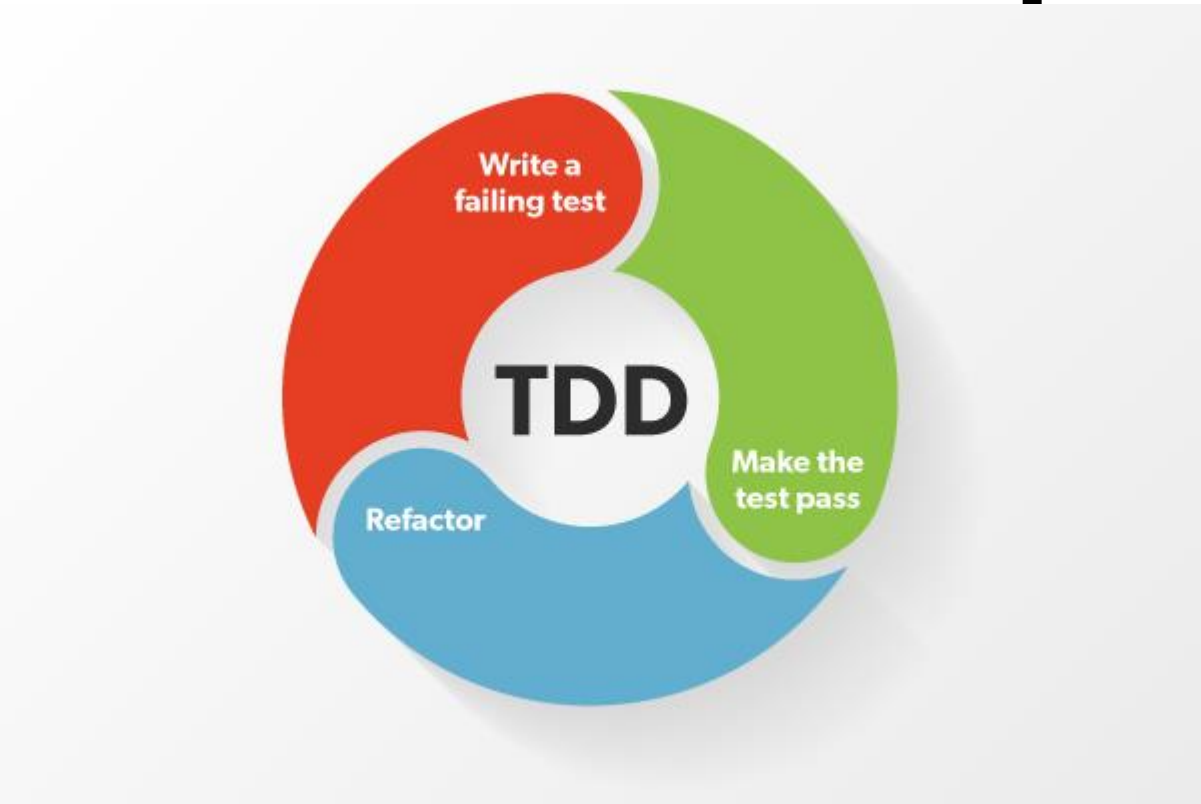
Why Testing is Important?

- Ensure reliable and bug-free code
- Catch issues before they reach production
- Maintainable Codebase - facilitate refactoring and code changes
- Serves as documentation
- Automated regression

Tools for testing in React

- Jest
- Mocha
- Chai
- React Testing Library
- Enzyme
- Testing Library (dom-testing-library, @testing-library/dom)
- Cypress Framework
- Testing Library (react-hooks-testing-library)
- Sinon

Test-Driven Development



Benefits of TDD

- Clear, Concise Code
- Minimize context switching
- Catches bugs early
- Better live documentation

Tools we're going to use

- ***React Testing Library*** – Testing library for React Components
- ***Vitest*** – unit testing framework built on top of Vite (Vue, React, Svelte withTypeScript / JSX support.)
- ***Happy Dom*** – in-browser DOM emulation library used for running tests in the browser-like environment without the need for a real browser. (alternative for JSDOM or Puppeteer)
- ***Jest*** (optional) – A powerful and widely used testing framework for JavaScript and React applications.

Setup and Configuration

Generate a new project

npm create vite@latest / npm create vite@4.1.0

Generate a new project

- ❑ Project Name: rts-d5-react-testing
- ❑ > React
- ❑ > TypeScript
- ❑ > cd rts-d5-react-testing
- ❑ > npm run dev

Install packages

```
npm install -D vitest happy-dom @testing-library/react
```

Installing Vitest, happy-dom, React Testing Library

Go to *vite.config.ts* and add the test object with its type definition

TS vite.config.ts ×

TS vite.config.ts > default > test > environment

```typescript
/// <reference types="vitest" />
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  test: {
    globals: true,
    environment: 'happy-dom'
  },
})
```

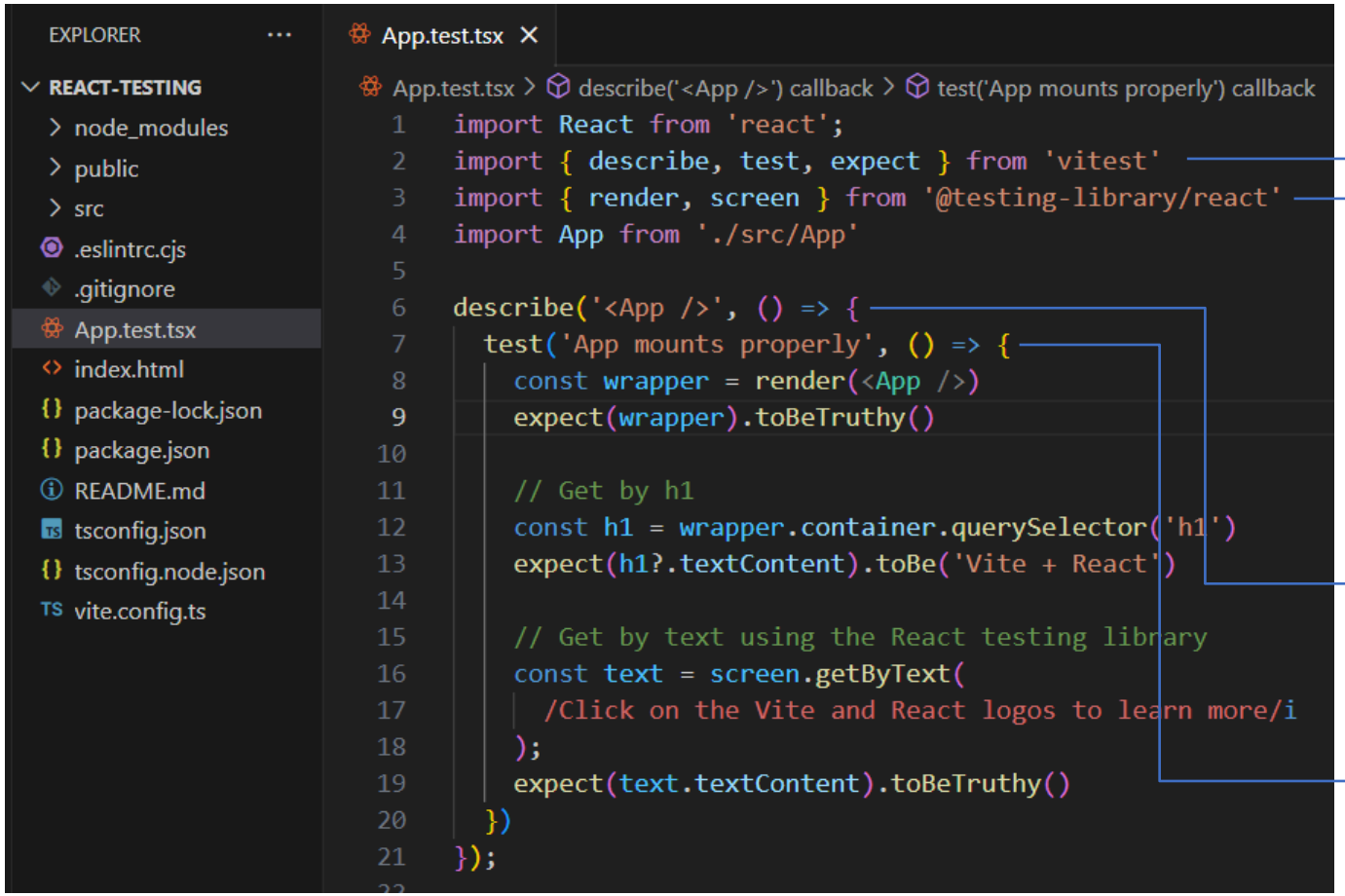
TypeScript reference directive used to specify that this file uses types from the "vitest" type declaration package.

This block configures the test environment for Vite. Test files should have access to the global scope. This line sets the test environment to "happy-dom."

Add the test script to *package.json*

```
6   "scripts": {
7     "dev": "vite",
8     "build": "tsc && vite build",
9     "lint": "eslint . --ext ts,tsx --report-unused-disable-directives --max-warnings 0",
10    "preview": "vite preview",
11    "test": "vitest",
12    "coverage": "vitest run --coverage"
13  },
```

Create a new file in the root and call it *App.test.tsx*



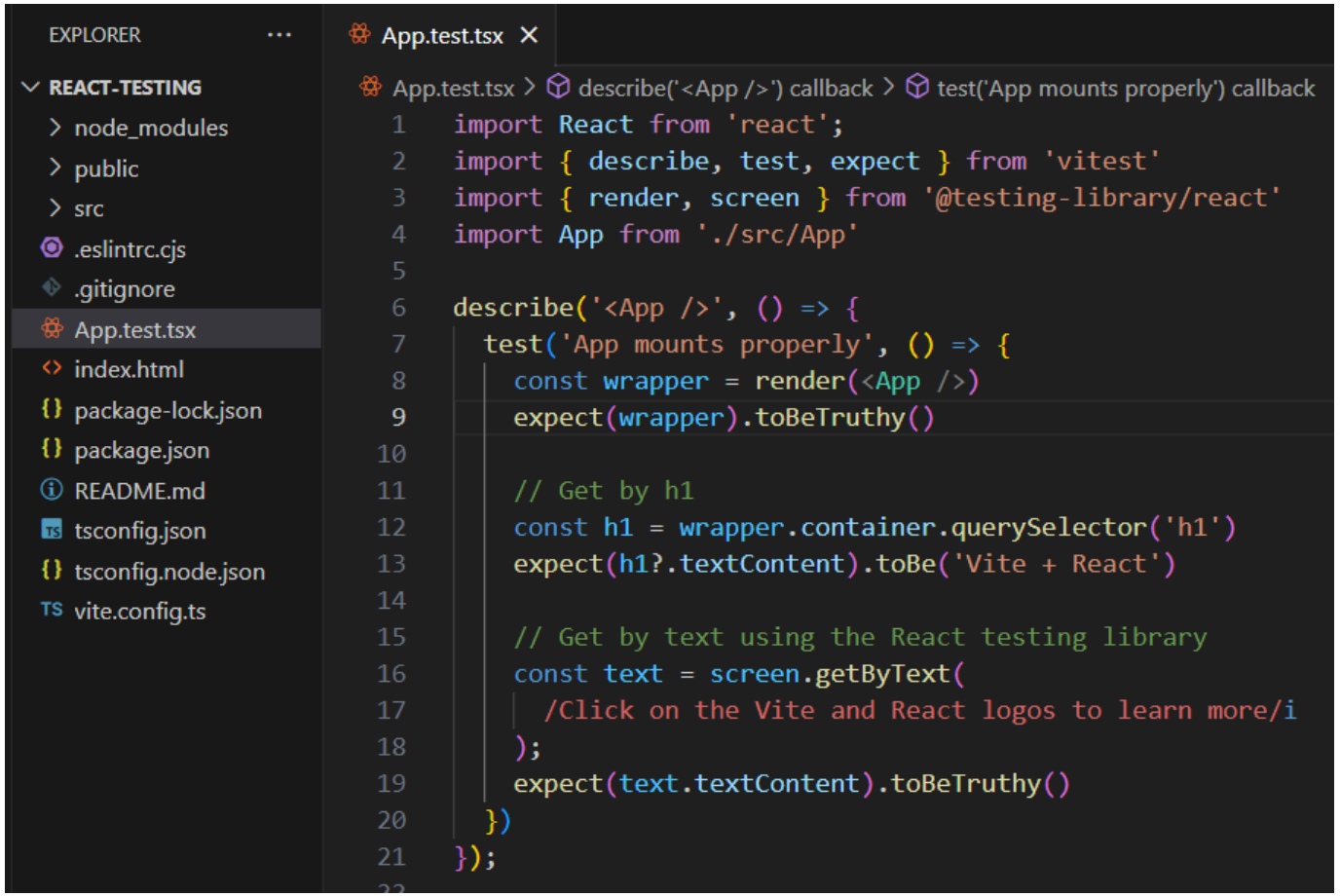
Imports testing utilities from the Vitest testing library.

Imports the *render* and *screen* utilities from the "@testing-library/react" library.

Focuses in <App /> Component

Defines a test case with the description "App mounts properly"

Create a new file in the root and call it *App.test.tsx*



Test case logic goes here

Run the test script

```
npm run test
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\admin\Desktop\TRAINOSYS\reactjs-typescript-training\day-5\code-alongs\test-proj\react-testing> npm run test

> react-testing@0.0.0 test
> vitest


 DEV  v0.34.1 C:/Users/admin/Desktop/TRAINOSYS/reactjs-typescript-training/day-5/code-alongs/test-proj/react-testing

 ✓ App.test.tsx (1)
   ✓ <App />  (1)
     ✓ App mounts properly

 Test Files  1 passed (1)
      Tests  1 passed (1)
   Start at  00:42:19
   Duration  844ms (transform 63ms, setup 0ms, collect 247ms, tests 20ms, environment 221ms, prepare 125ms)


 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Run the coverage script

```
npm run coverage
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

> react-testing@0.0.0 coverage
> vitest run --coverage


 RUN  v0.34.1 C:/Users/admin/Desktop/TRAINOSYS/reactjs-typescript-training/day-5/code-alongs/test-proj/react-testing
      Coverage enabled with v8

 ✓ App.test.tsx (2)
   ✓ <App />  (2)
     ✓ App mounts properly
     ✓ Click the button

 Test Files  1 passed (1)
      Tests  2 passed (2)
   Start at  01:26:21
   Duration  984ms (transform 65ms, setup 0ms, collect 258ms, tests 28ms, environment 219ms, prepare 138ms)

 % Coverage report from v8
----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |     100 |      100 |     100 |     100 |
 App.tsx  |     100 |      100 |     100 |     100 |
----------|---------|----------|---------|---------|-------------------
PS C:\Users\admin\Desktop\TRAINOSYS\reactjs-typescript-training\day-5\code-alongs\test-proj\react-testing>
```

Let’s test user events

App.test.tsx

App.test.tsx > describe('<App />') callback

```
20    });
21
22    test('Click the button', () => {
23      const wrapper = render(<App />)
24      const button = wrapper.container.querySelector(
25        'button'
26      ) as HTMLButtonElement;
27
28      // button mounts with count in 0
29      expect(button.textContent).toBe('count is 0')
30
31      fireEvent(
32        getByText(button, 'count is 0'),
33        new MouseEvent('click', {
34          bubbles: true
35        }),
36      )
37
38      // The count hook is working
39      expect(button.textContent).toBe('count is 1')
40    })
41  });
42
```

```
2  import { describe, test, expect } from 'vitest'
3  import { render, screen, fireEvent, getByText } from '@testing-library/react'
4  import App from './src/App'
```

ACTIVITY 1

Activity 1

- ❑ Continue the Code Along in Testing
- ❑ Create separate buttons for Increment and Decrement
- ❑ Display the Count value in a another
- ❑ Write test cases

TRAINOSYS

Advance TypeScript for React

HYBRID
TECHNOLOGY
COURSES

Type Guards

```
interface Person {
  name: string;
  age: number;
  address?: {
    street: string;
    city: string;
    state: string;
  }
}

function hasAddress(person: Person): person is Person & { address: object } {
  return person.address !== undefined;
}

function getAddress(person: Person): string {
  if (hasAddress(person)) {
    return `${person.address.street}, ${person.address.city}, ${person.address.state}`;
  }
  throw new Error("Person does not have an address.");
}
```

Unions

```typescript
interface Circle {
  kind: "circle";
  radius: number;
}

interface Square {
  kind: "square";
  sideLength: number;
}

interface Rectangle {
  kind: "rectangle";
  width: number;
  height: number;
}

type Shape = Circle | Square | Rectangle;

function area(shape: Shape): number {
  switch (shape.kind) {
    case "circle":
      return Math.PI * shape.radius ** 2;
    case "square":
      return shape.sideLength ** 2;
    case "rectangle":
      return shape.width * shape.height;
  }
}
```

Optimization techniques

src > TS App.tsx > App

```typescript
import React, { useState, useMemo } from 'react';

const App: React.FC = () => {
  const [number, setNumber] = useState<number>(5);

  function calculateFactorial(num: number): number {
    console.log(`Calculating factorial of ${num}`);
    if (num === 0 || num === 1) return 1;
    return num * calculateFactorial(num - 1);
  }

  const factorial: number = useMemo(() => calculateFactorial(number), [number]);

  const handleChange = (event: React.ChangeEvent<HTMLInputElement>): void => {
    setNumber(parseInt(event.target.value));
  };

  return (
    <div>
      <h1>Factorial Calculator</h1>
      <div>
        <label>Enter a number: </label>
        <input type="number" value={number} onChange={handleChange} />
      </div>
      <div>
        <p>Factorial of {number} is: {factorial}</p>
      </div>
    </div>
  );
}

export default App;
```

useMemo() Hook

```
const AppRouting = () => {
  const isLoggedIn = useAppSelector(selectAuthLoggedIn);

  const screenComponents = React.useMemo(() => {
    if (isLoggedIn) {
      return (
        <>
          <RootStack.Screen name="Home" component={Homepage} />
          <RootStack.Screen name="LiveRequest" component={LiveRequest} />
        </>
      );
    } else {
      return (
        <>
          <RootStack.Screen name="Signin" component={SignIn} />
          <RootStack.Screen name="Signup" component={SignUp} />
          <RootStack.Screen name="Forgot" component={Forgot} />
          <RootStack.Screen name="ResetPassword" component={ResetPassword} />
        </>
      );
    }
  }, [isLoggedIn]);

  return (
    <RootStack.Navigator screenOptions={RootStackOptions}>
      {screenComponents}
    </RootStack.Navigator>
  );
};

export default AppRouting;
```

useMemo() Hook

```typescript
import React, { useState, useCallback } from 'react';

interface Color {
  name: string;
  hexCode: string;
}

const colorList: Color[] = [
  { name: 'Red', hexCode: '#FF0000' },
  { name: 'Green', hexCode: '#00FF00' },
  { name: 'Blue', hexCode: '#0000FF' },
];

const ColorPicker: React.FC = () => {
  const [selectedColor, setSelectedColor] = useState<Color | null>(null);

  const handleColorSelect = useCallback(
    (color: Color) => {
      setSelectedColor(color);
    },
    []
  );
```

useCallback() Hook

```
  return (
    <div>
      <h1>Color Picker</h1>
      <div>
        {colorList.map((color) => (
          <div
            key={color.name}
            onClick={() => handleColorSelect(color)}
            style={{
              backgroundColor: color.hexCode,
              padding: '20px',
              borderRadius: '5px',
              cursor: 'pointer',
              margin: '10px',
            }}
          >
            {color.name}
          </div>
        ))}
      </div>
      {selectedColor && (
        <div style={{ marginTop: '20px' }}>
          <p>Selected Color: {selectedColor.name}</p>
          <p>HEX Code: {selectedColor.hexCode}</p>
        </div>
      )}
    </div>
  );
}

export default ColorPicker;
```

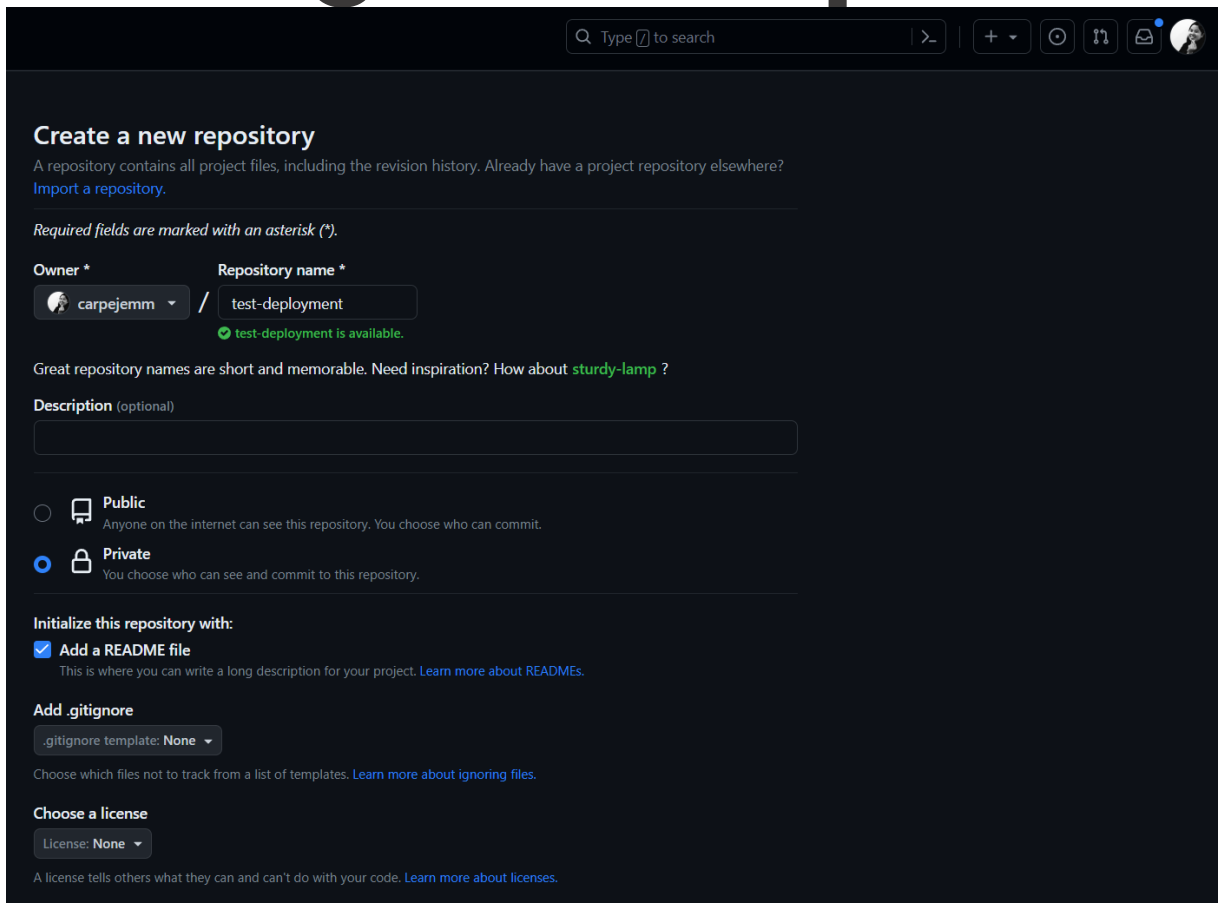
useCallback() Hook

Deployment

Deployment

- Deploying on Vercel – https://vercel.com/docs/getting-started-with-vercel/projects-deployments
- Deploying on Render – https://render.com/docs
- Deploying on Netlify – https://www.netlify.com/blog/2016/09/29/a-step-by-step-guide-deploying-on-netlify/

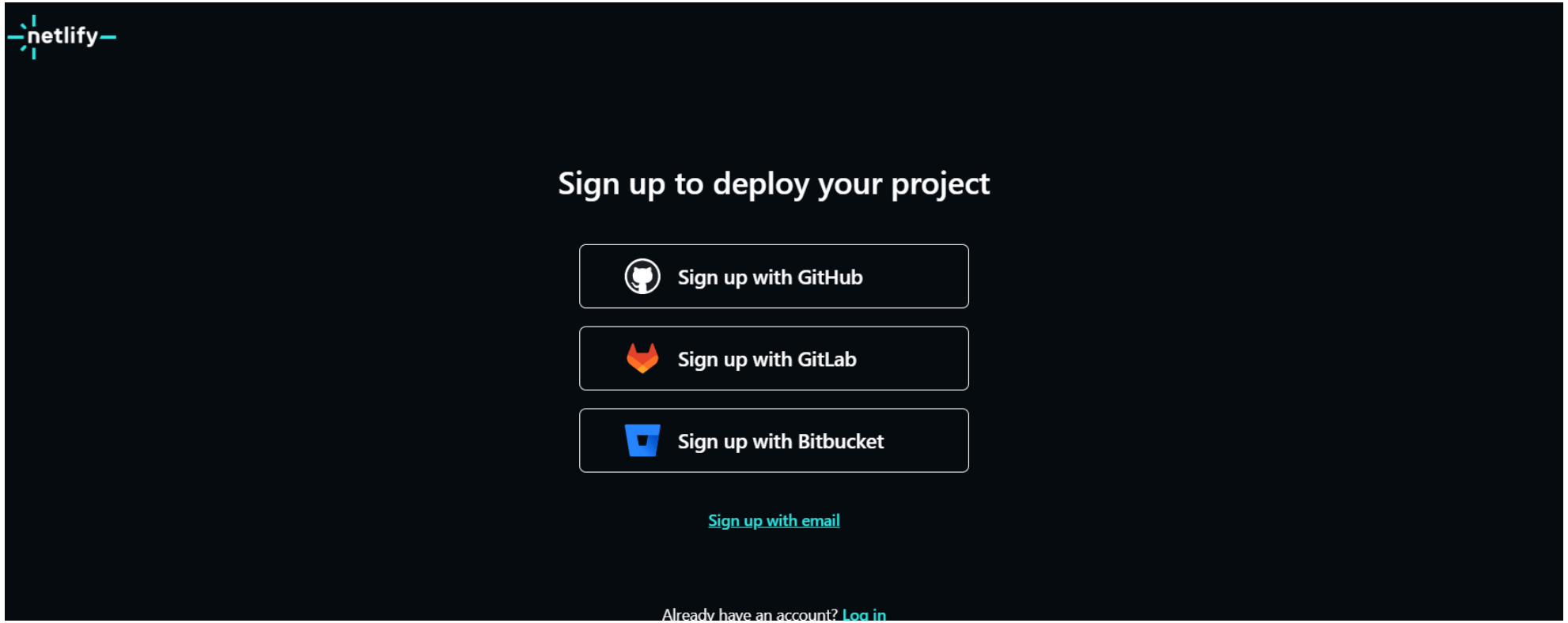
Create a new github repository



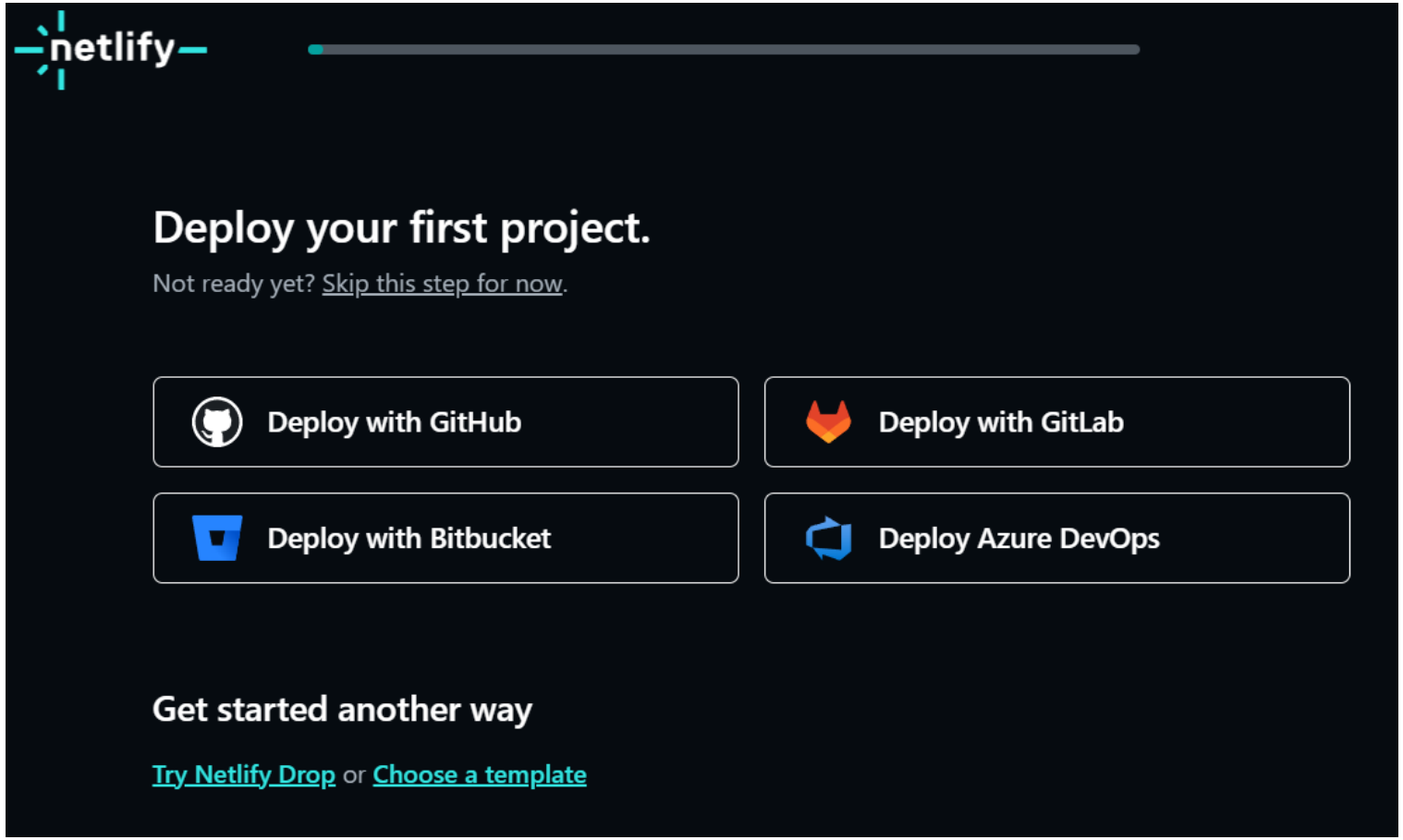
Clone the repository and push any existing proj (example: pokedex proj)

- ❑ > git clone <github link>
- ❑ Inside the folder copy any existing proj like your pokedex
- ❑ > git add .
- ❑ > git commit –m “<commit message>”
- ❑ > git push

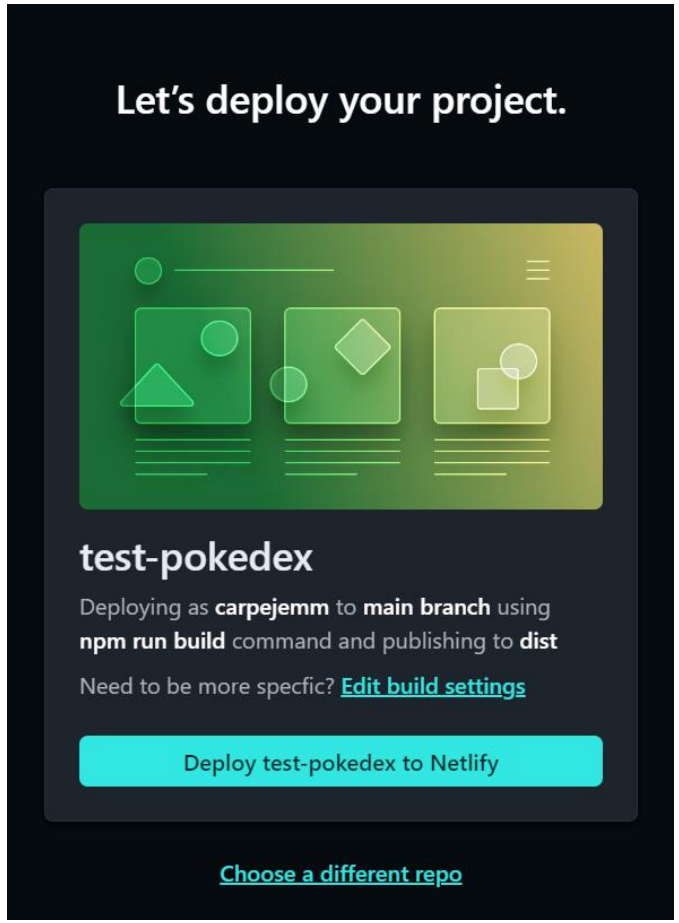
Create an account with Netlify and sign up with github (authorized github)



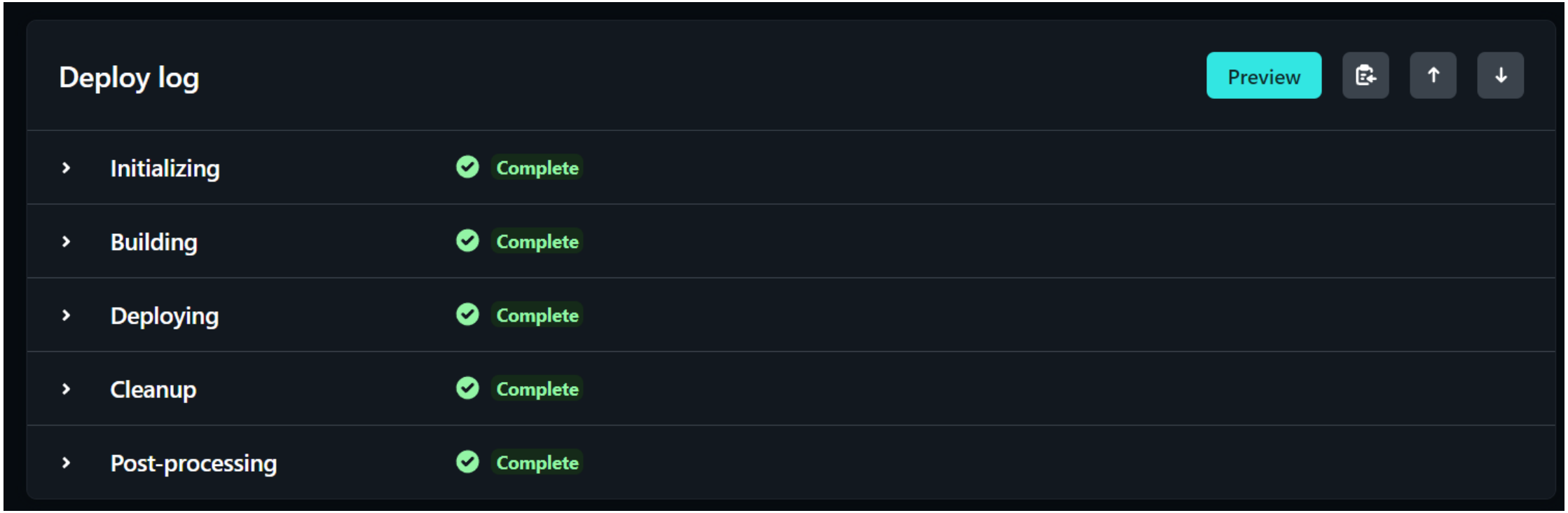
Deploy your first project screen



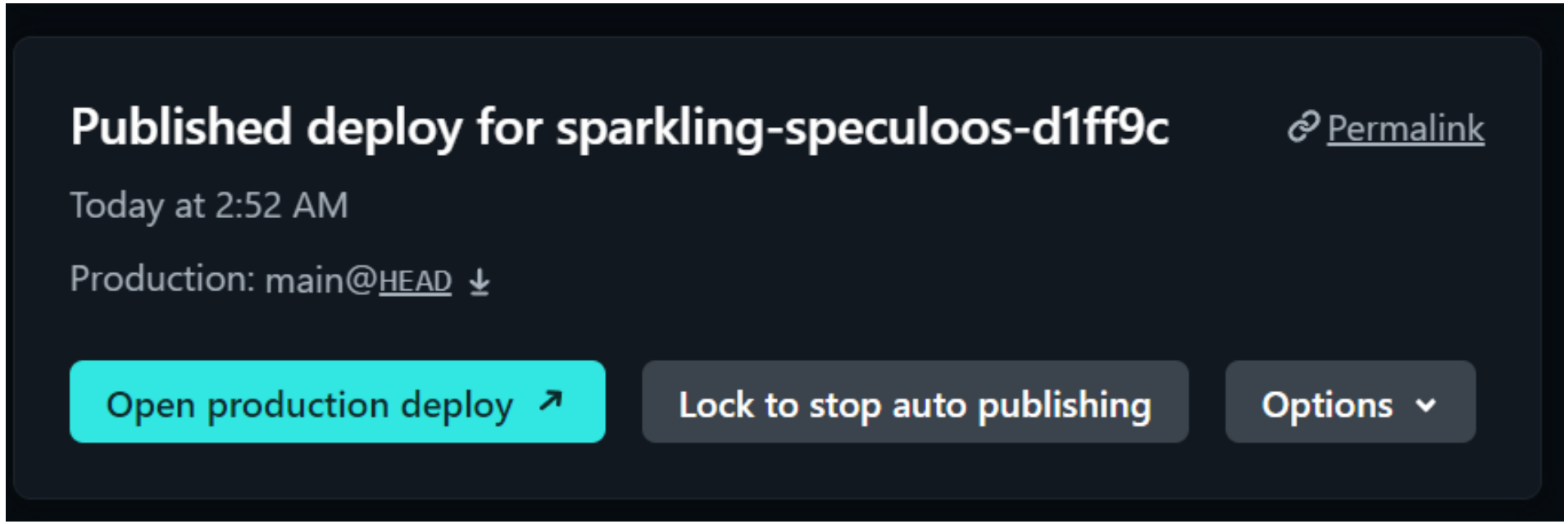
Select your repository and deploy



Deployment logs



Open Link


Published deploy for sparkling-speculoos-d1ff9c
Permalink
Today at 2:52 AM
Production: main@HEAD
Open production deploy
Lock to stop auto publishing
Options

ACTIVITY 2

Deploy Activity 1

ACTIVITY – Kahoot!

TRAINOSYS

CONTINUE DAY 4 ACTIVITY
react-portfolio-project

HYBRID
TECHNOLOGY
COURSES

API for “Status of your Applications”

- https://asia-east2-cstf-v2.cloudfunctions.net/app

TRAINOSYS

React Calendar

HYBRID
TECHNOLOGY
COURSES

Install packages

```
npm install @fullcalendar/react @fullcalendar/daygrid
@fullcalendar/interaction
```

React Calendar

TRAINOSYS

TS App.tsx

src > TS App.tsx > App

```typescript
import './App.css'
import BookingCalendar from './components/BookingCalendar';

function App() {

  return (
    <>
      <div className="App">
        <h1>Calendar Booking App</h1>
        <BookingCalendar />
      </div>
    </>
  )
}

export default App
```

TRAINOSYS

TS App.tsx | TS BookingCalendar.tsx ×

src > components > TS BookingCalendar.tsx > BookingCalendar

```typescript
import React, { useState } from 'react';
import FullCalendar from '@fullcalendar/react';
import dayGridPlugin from '@fullcalendar/daygrid';
import interactionPlugin from '@fullcalendar/interaction';
import './styles.css';

const BookingCalendar: React.FC = () => {
  const [selectedDate, setSelectedDate] = useState<string | null>(null);
  const [events, setEvents] = useState<any[]>([]);


  const handleDateClick = (arg: any) => {
    setSelectedDate(arg.dateStr);
  };

  const handleBookSlot = () => {
    if (selectedDate) {
      console.log('Booking slot:', selectedDate);

      const newEvent = {
        title: 'New Booking',
        date: selectedDate,
      };
      setEvents([...events, newEvent]);

      setSelectedDate(null);
    }
  };

```

```typescript
  return (
    <div className="calendar-container">
      <div className="book-button-container">
        <button className="book-button" onClick={handleBookSlot}>
          Book Slot
        </button>
      </div>
      <FullCalendar
        plugins={[dayGridPlugin, interactionPlugin]}
        initialView="dayGridMonth"
        events={events}
        editable={true}
        selectable={true}
        dateClick={handleDateClick}
      />
    </div>
  );
};

export default BookingCalendar;
```

TRAINOSYS
Training the Future Today

Reach Us!

Visit Us

12th/F The Trade & Financial Tower Unit 1206
32nd Street & 7th Avenue Bonifacio Global City,
Taguig 1634 Philippines

Email Us

inquiry@trainosys.com

Browse Our Website

www.trainosys.com

TRAINOSYS

Training the Future Today