**Lesson 05 Demo 05**

**Using a Pretrained Model to Create an Application Structure**

**Objective:** To create a pre-trained model for generating an application structure based on the type of application

**Tools required:** GitHub Copilot

**Prerequisites:** Basic knowledge of programming languages

Steps to be followed:

1. Create a pretrained model using GitHub Copilot
2. Run the code using GitHub Copilot

**Note:** Please be advised that GitHub Copilot is an artificial intelligence tool that can produce varied outputs even when presented with similar prompts. Thus, you may get different code snippets for the same prompt.

**Step 1:** **Create a pretrained model using GitHub Copilot**

* 1. Click on the **Chat** icon, click on the **three dots** icon, and select the **Open Chat in Editor** option

A screenshot of a computer

Description automatically generated

* 1. Use the following prompt to create a pretrained model class

**“**

**MyStructureGenModel that generates an application structure:**

**You're working on a code generation tool that dynamically creates directory structures for different types of applications. You have a hypothetical machine learning model, MyStructureGenModel, which generates a basic directory structure based on user input. The model takes in user input (user\_input) indicating the type of application structure to be generated (for example, 'web', 'mobile', 'custom').**

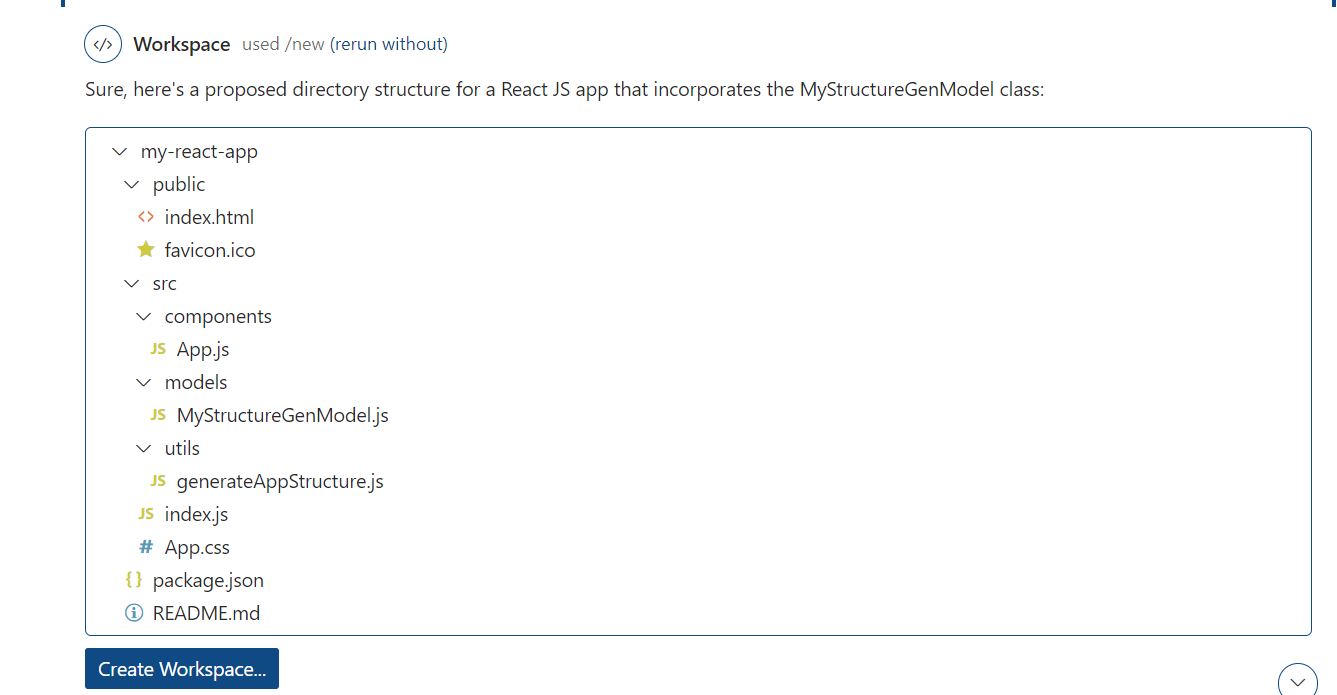
**The MyStructureGenModel class has a static method generate\_structure that processes the user input and generates the application structure in a dictionary format. For instance, if the user input is 'web', it generates a directory structure for a web application with folders named 'web\_app', 'static', 'templates', and 'config'. For 'mobile', it generates 'mobile\_app', 'ui', 'logic', and 'resources'. Otherwise, it creates a 'custom\_app' folder.**

**You need to create a React JS application that utilizes this model to generate the application structure based on user input. Additionally, when the script generates the structure, it should display the generated directory tree to the console.**

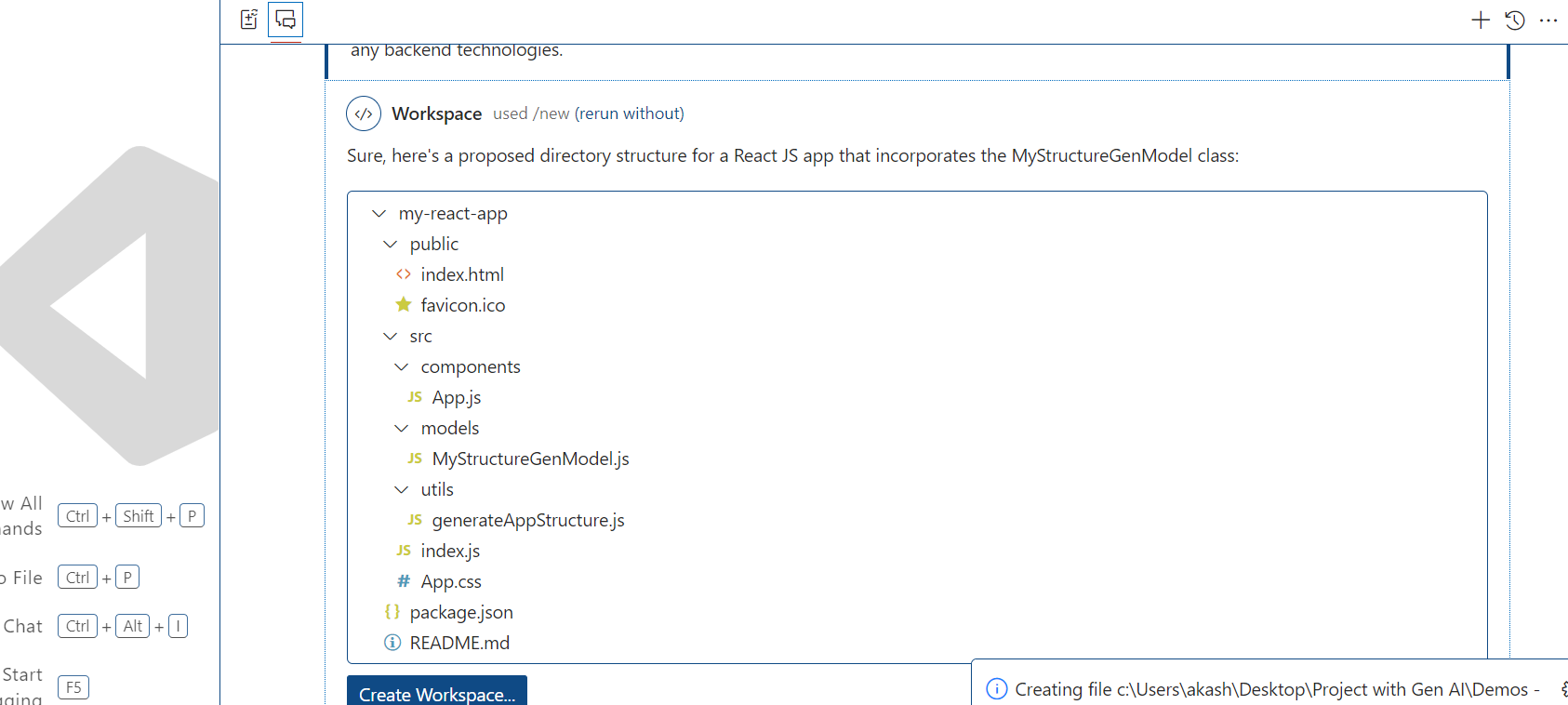
**Write React JS code that incorporates the MyStructureGenModel class and includes the necessary functions to create the directory structure based on the generated dictionary output. The react js should prompt the user to enter the type of application structure they want to generate ('web', 'mobile', 'custom'), use the model to generate the structure, create the directories accordingly, and display the directory structure in the browser.**

**Please implement the code logic inside the generate\_app\_structure\_with\_model function. Consider using the os module to create directories and the os.getcwd() function to get the current working directory. I need the complete instruction step by step to perform this task without any backend technologies.”**

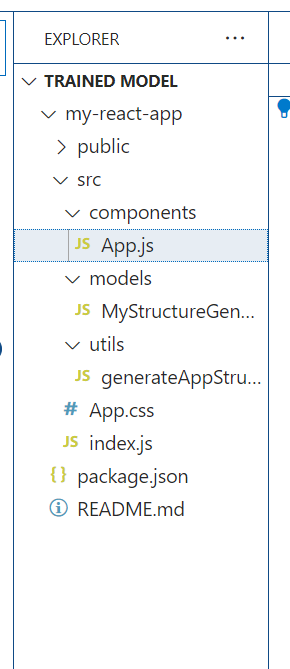
1.3



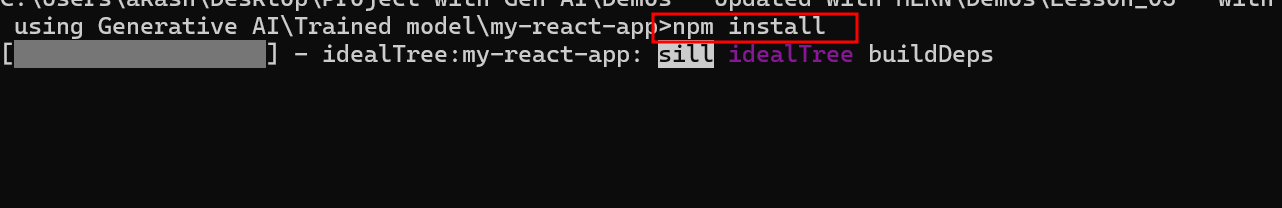
* 1. Now click on create workspace



* 1. It will create the all required file inside a workspace



* 1. now move inside this folder and run the command as **npm install**



* 1. all file generated code

**package.json file**

{

  "name": "my-react-app",

  "version": "1.0.0",

  "private": true,

  "dependencies": {

    "react": "^17.0.2",

    "react-dom": "^17.0.2",

    "react-scripts": "4.0.3"

  },

  "scripts": {

    "start": "react-scripts --openssl-legacy-provider start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": [

      "react-app",

      "react-app/jest"

    ]

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

  }

}

**generateAppStructure.js file part of utils folder**

import MyStructureGenModel from '../models/MyStructureGenModel';

export const generate\_app\_structure\_with\_model = (appType) => {

    const structure = MyStructureGenModel.generate\_structure(appType);

    return structure;

};

**MyStructureGenModel class code part of models folder**

class MyStructureGenModel {

    static generate\_structure(user\_input) {

        switch (user\_input.toLowerCase()) {

            case 'web':

                return {

                    web\_app: ['static', 'templates', 'config']

                };

            case 'mobile':

                return {

                    mobile\_app: ['ui', 'logic', 'resources']

                };

            default:

                return {

                    custom\_app: []

                };

        }

    }

}

export default MyStructureGenModel;

**App.js file code part of component folder**

import React, { useState } from 'react';

import { generate\_app\_structure\_with\_model } from '../utils/generateAppStructure';

const App = () => {

    const [appType, setAppType] = useState('');

    const [directoryStructure, setDirectoryStructure] = useState(null);

    const handleInputChange = (event) => {

        setAppType(event.target.value);

    };

    const handleSubmit = (event) => {

        event.preventDefault();

        const structure = generate\_app\_structure\_with\_model(appType);

        setDirectoryStructure(structure);

        console.log(structure);

    };

    return (

        <div>

            <h1>Application Structure Generator</h1>

            <form onSubmit={handleSubmit}>

                <label>

                    Enter application type (web, mobile, custom):

                    <input type="text" value={appType} onChange={handleInputChange} />

                </label>

                <button type="submit">Generate Structure</button>

            </form>

            {directoryStructure && (

                <div>

                    <h2>Generated Directory Structure:</h2>

                    <pre>{JSON.stringify(directoryStructure, null, 2)}</pre>

                </div>

            )}

        </div>

    );

};

export default App;

**Step 2 : Now run the application**

2.1 npm start

