DevOps Project

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad – 500 043

Certificate

This is to certify that it is a bonafied record of practical work done by Mr. / Ms.

Rohit.M, Rithvik.M, R.Raju bearing the roll no. 23951A057Q,23951A057P, 23951A0578

of B.Tech CSE class Computer Science Engineering branch in

the DevOps laboratory during the academic

year 2024-25 under our supervision.

Head of the department **Lecturer – in charge**

Signature of External Examiner **Signature of Internal Examiner**

DevOps Project On: Dockerize Of weather application

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad – 500 043, Telangana

Department of CSE
Bachelor of Technology
in
CSE

-BY

M.Rohit
23951A057Q

M. Rithvik
23951A057P

R.V.K.S.N. Raju
23951A0578

DECLARATION

I certify that

a. The work contained in this report is original and has been done by me under the guidance of my supervisor (s).
b. The work has not been submitted to any other Institute for any degree or diploma.
c. I have followed the guidelines provided by the Institute for preparing the report.
d. I have conformed to the norms and guidelines given in the Code of Conduct of the Institute.
e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Place: Hyderabad **Signature of the Student**

Date: **Roll No:**

ABSTRACT

In the evolving world of software engineering, deploying frontend applications rapidly and reliably has become crucial. This project presents the complete DevOps lifecycle applied to a **React-based Weather Application**, focusing on its **Dockerization and CI/CD automation** using **GitHub Actions**.

The primary goal is to automate the building, testing, and container-based deployment of the weather app to ensure high reliability and faster delivery cycles.
The project begins with version-controlling the React codebase using Git and GitHub.

It then sets up GitHub Actions to automate the CI/CD pipeline, triggered by events such as pushes or pull requests. The pipeline includes linting, testing (if available), building the app, and creating a production-ready Docker image. Finally, the Docker image is pushed to a container registry or hosted on a server using Docker.

This hands-on implementation of CI/CD with Docker highlights the practical aspects of modern DevOps culture—automated testing, reproducible builds, and seamless deployment. It showcases how containerization ensures consistency across environments while GitHub Actions provides a powerful and integrated way to manage the entire lifecycle of the React weather application.

CONTENTS

Chapter 1 Introduction 1

1.1 About CI/CD pipelining 1

1.2 Requirements 2

1.3 Prerequisites 3

Chapter 2 Methodology 5

Chapter 3 Results and Discussions 8

Chapter 4 Conclusion 11

CHAPTER 1
INTRODUCTION

1.1 About CI/CD Pipelining

This project focuses on a React-based Weather Application that provides real-time weather data using external APIs. The application allows users to search for any city and view its current weather conditions, including temperature, humidity, wind speed, and forecasts. While the frontend logic is handled by React, deployment and delivery are automated using modern DevOps techniques.

To ensure reliable and rapid delivery, a CI/CD pipeline is integrated using GitHub Actions, while the application is containerized using Docker. Continuous Integration ensures every code push triggers automated builds and tests. Continuous Deployment automates the process of shipping the app to production—either as a static site or as a containerized service. This eliminates manual deployment overhead and promotes stable, repeatable delivery across environments.

The project not only streamlines development and deployment but also introduces students to real-world DevOps practices like version control, automation, and containerization. The pipeline guarantees that every change is validated and deployed consistently, reducing risks and improving overall efficiency.

1.2 Requirements

To implement Dockerization and CI/CD for the React-based weather app, the following tools and configurations were required:

1.2.1 GitHub Repository

- The weather app's source code is hosted in a GitHub repository.
- Contains:
  - React source code
  - .github/workflows/ci.yml for GitHub Actions
  - Dockerfile for containerizing the app
  - Static configuration and environment files

1.2.2 GitHub Actions

- YAML workflows define CI/CD stages like install, build, and deploy.
- Triggered by events such as push or pull_request.
- Uses actions like:
  - actions/checkout@v2 to access the repo
  - actions/setup-node for setting up Node.js
  - Custom shell commands to build and Dockerize the app

1.2.3 React Weather App Codebase

- Written in React.js, with Axios or Fetch for API calls.
- Uses npm or yarn for dependency management.
- Includes:
  - package.json for scripts
  - public/ and src/ folders
  - Optionally, basic unit testing with Jest

1.2.4 Docker

- Used for creating consistent deployment environments.
- Key files:

    - Dockerfile defining the build instructions
    - Optional .dockerignore to reduce image size
- Docker commands (build, run, push) used in CI/CD pipeline

1.3 Prerequisites

1.3.1 Technical Knowledge

- **Git & GitHub:** For version control and remote collaboration
- **React.js:** Understanding of component-based frontend development
- **YAML:** To configure GitHub Actions workflows
- **Docker:** Basics of containerization, including Dockerfile syntax

1.3.2 Tooling Setup

- **Git** installed on local machine
- **VS Code or similar IDE** for development
- **Docker Desktop** for local container testing
- **GitHub account** to host code and enable Actions

1.3.3 Application Readiness

- Fully functional React frontend
- API key and integration for weather data (e.g., OpenWeatherMap)
- Clean project structure
- Optional: Basic testing scripts for validation during CI

CHAPTER 2
METHODOLGY

This project implements the DevOps lifecycle focusing on CI/CD and containerization for the React Weather App. The methodology is structured around key DevOps principles:

Phases of Implementation

1. Code Development and Version Control

- Developers write modular React code and test UI components.
- Git is used to manage source code and handle branches.
- Code is regularly pushed to GitHub.

2. CI/CD Workflow Configuration with GitHub Actions

- Created .github/workflows/ci.yml with the following stages:
    - Install Node.js
    - Install dependencies using npm install
    - Run build with npm run build
    - (Optional) Run tests using npm test
    - Build Docker image
    - (Optional) Push Docker image to Docker Hub or deploy container

3. Dockerization

- Dockerfile created with multistage build:
    - Stage 1: Build React app using Node
    - Stage 2: Serve static files using nginx
- Example Dockerfile:

Dockerfile

CopyEdit

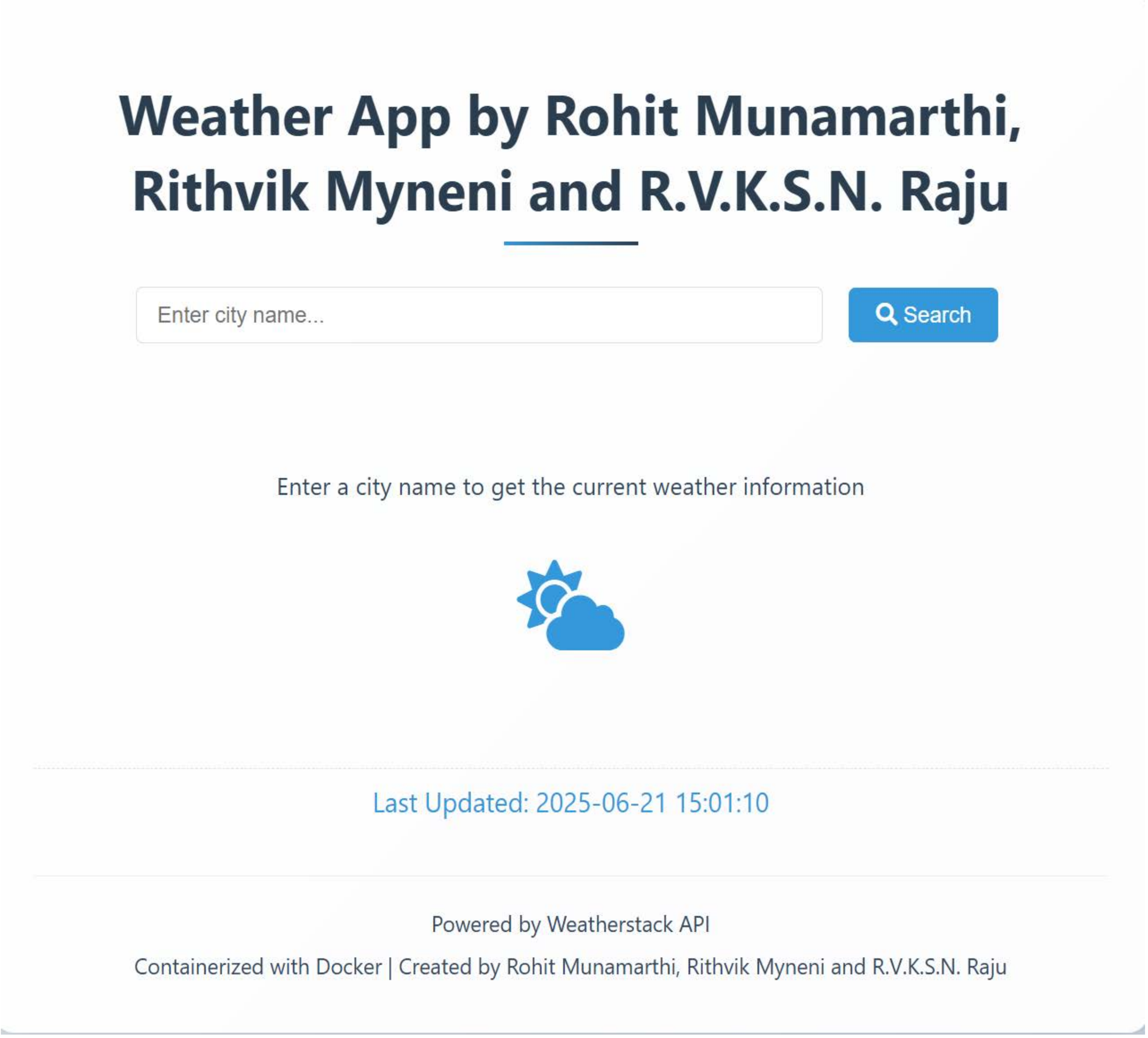
```
FROM node:18-alpine as build
WORKDIR /app
RUN npm install && npm run build

FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
```

4. Deployment

- Docker image can be run locally or on cloud platforms.
- (Optional) Deployment via services like:
  - GitHub Pages (for static app)
  - Docker Hub + server
  - AWS EC2, Heroku, or DigitalOcean

CHAPTER 3
RESULTS AND DISCUSSION

Execution:



Weather App by Rohit Munamarthi, Rithvik Myneni and R.V.K.S.N. Raju

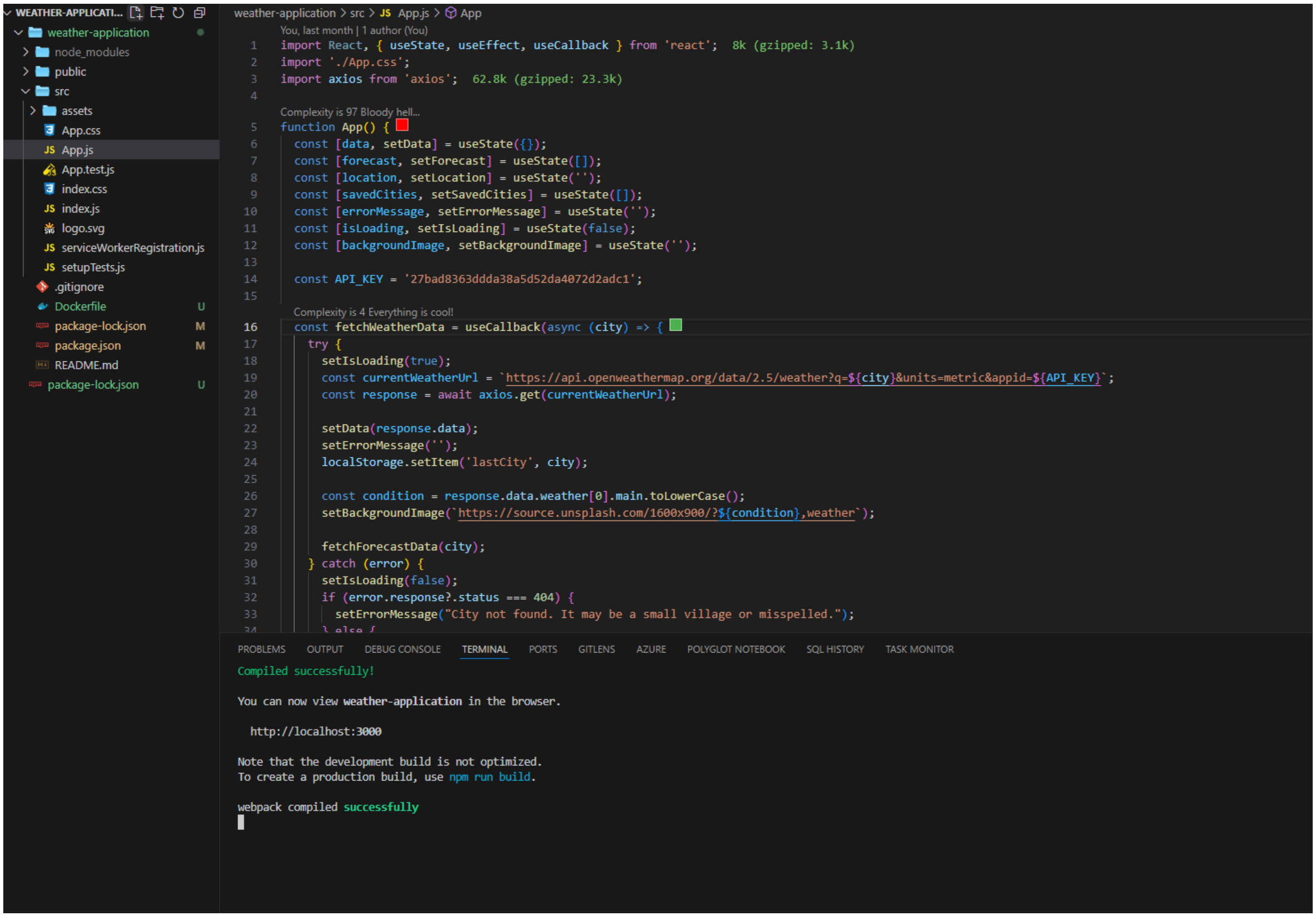
Enter a city name to get the current weather information

Last Updated: 2025-06-21 15:01:10

Powered by Weatherstack API

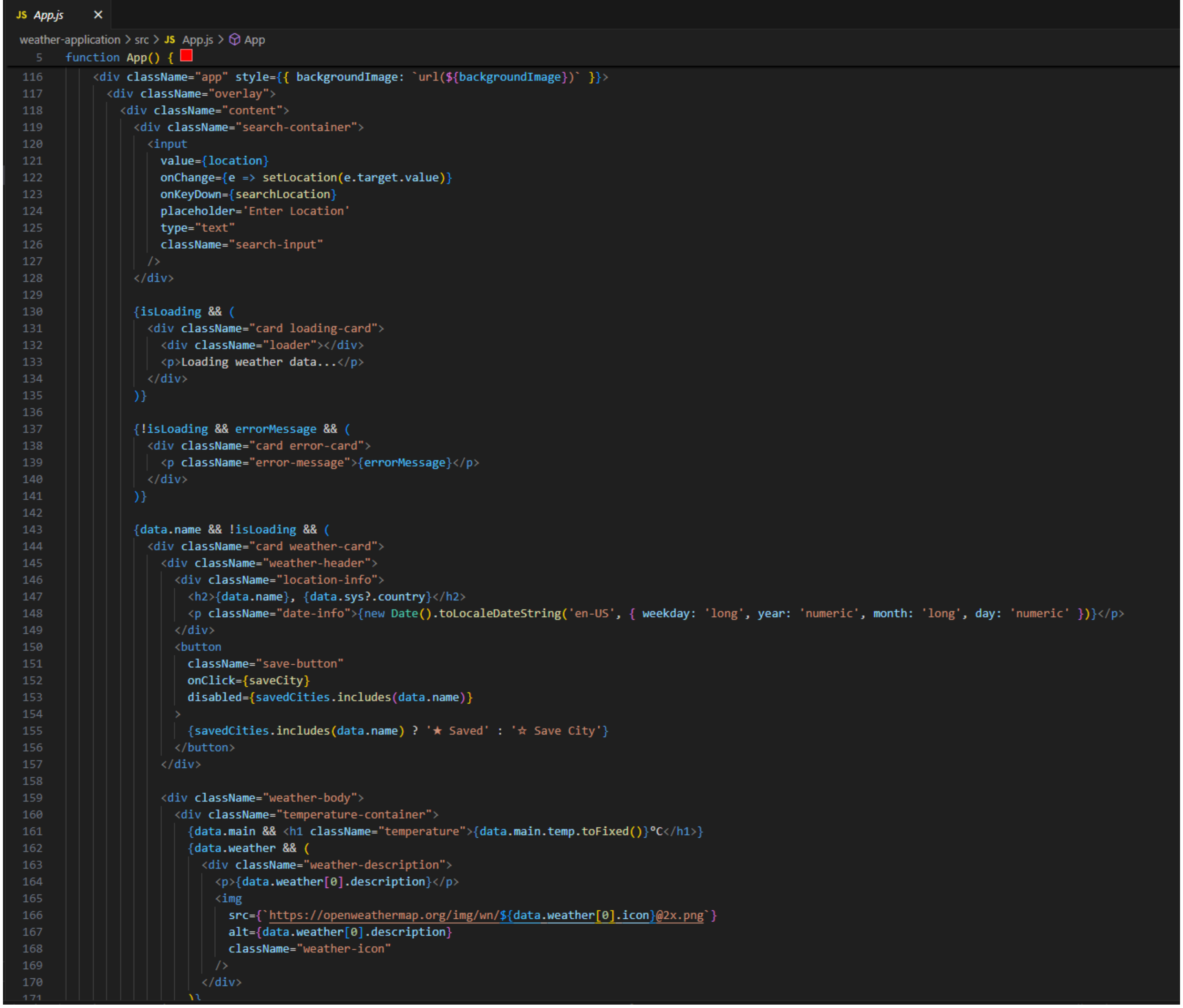
Containerized with Docker | Created by Rohit Munamarthi, Rithvik Myneni and R.V.K.S.N. Raju

Source Code:



```javascript
import React, { useState, useEffect, useCallback } from 'react';  8k (gzipped: 3.1k)
import './App.css';
import axios from 'axios';  62.8k (gzipped: 23.3k)

function App() { 🟥
  const [data, setData] = useState({});
  const [forecast, setForecast] = useState([]);
  const [location, setLocation] = useState('');
  const [savedCities, setSavedCities] = useState([]);
  const [errorMessage, setErrorMessage] = useState('');
  const [isLoading, setIsLoading] = useState(false);
  const [backgroundImage, setBackgroundImage] = useState('');

  const API_KEY = '27bad8363ddda38a5d52da4072d2adc1';

  const fetchWeatherData = useCallback(async (city) => { 🟩
    try {
      setIsLoading(true);
      const currentWeatherUrl = `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${API_KEY}`;
      const response = await axios.get(currentWeatherUrl);

      setData(response.data);
      setErrorMessage('');
      localStorage.setItem('lastCity', city);

      const condition = response.data.weather[0].main.toLowerCase();
      setBackgroundImage(`https://source.unsplash.com/1600x900/?${condition},weather`);

      fetchForecastData(city);
    } catch (error) {
      setIsLoading(false);
      if (error.response?.status === 404) {
        setErrorMessage("City not found. It may be a small village or misspelled.");
      } else {
```

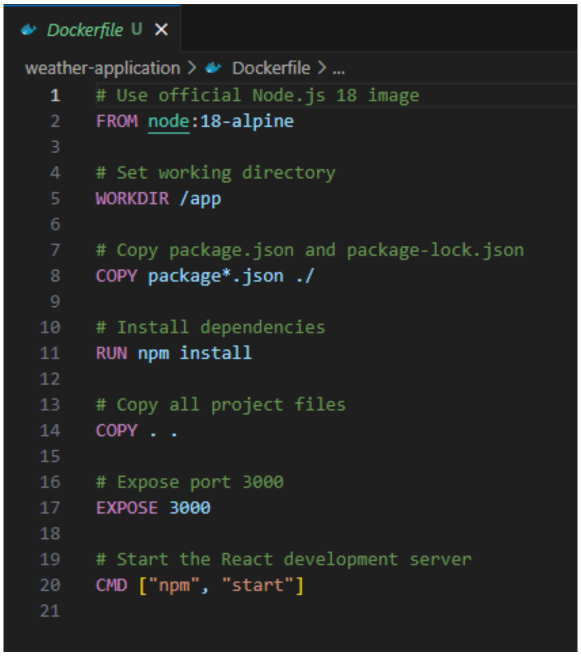
```
Compiled successfully!

You can now view weather-application in the browser.

  http://localhost:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
█
```



JS App.js ×

weather-application > src > JS App.js > App

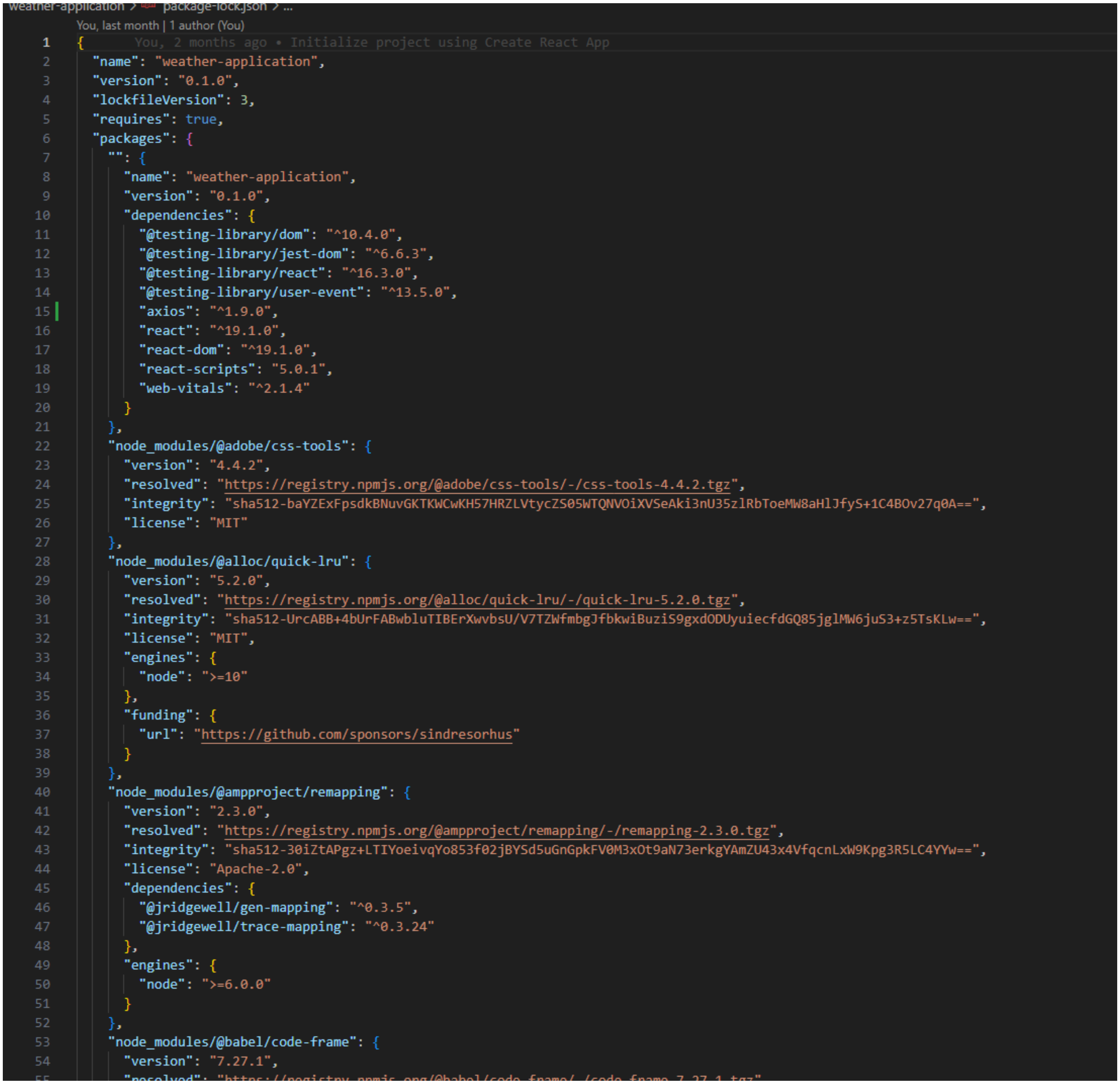
```
function App() {
      <div className="app" style={{ backgroundImage: `url(${backgroundImage})` }}>
        <div className="overlay">
          <div className="content">
            <div className="search-container">
              <input
                value={location}
                onChange={e => setLocation(e.target.value)}
                onKeyDown={searchLocation}
                placeholder='Enter Location'
                type="text"
                className="search-input"
              />
            </div>

            {isLoading && (
              <div className="card loading-card">
                <div className="loader"></div>
                <p>Loading weather data...</p>
              </div>
            )}

            {!isLoading && errorMessage && (
              <div className="card error-card">
                <p className="error-message">{errorMessage}</p>
              </div>
            )}

            {data.name && !isLoading && (
              <div className="card weather-card">
                <div className="weather-header">
                  <div className="location-info">
                    <h2>{data.name}, {data.sys?.country}</h2>
                    <p className="date-info">{new Date().toLocaleDateString('en-US', { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' })}</p>
                  </div>
                  <button
                    className="save-button"
                    onClick={saveCity}
                    disabled={savedCities.includes(data.name)}
                  >
                    {savedCities.includes(data.name) ? '★ Saved' : '☆ Save City'}
                  </button>
                </div>

                <div className="weather-body">
                  <div className="temperature-container">
                    {data.main && <h1 className="temperature">{data.main.temp.toFixed()}°C</h1>}
                    {data.weather && (
                      <div className="weather-description">
                        <p>{data.weather[0].description}</p>
                        <img
                          src={`https://openweathermap.org/img/wn/${data.weather[0].icon}@2x.png`}
                          alt={data.weather[0].description}
                          className="weather-icon"
                        />
                      </div>
                    )}
```



Dockerfile U ×

weather-application > Dockerfile > ...

```
# Use official Node.js 18 image
FROM node:18-alpine

# Set working directory
WORKDIR /app

# Copy package.json and package-lock.json
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy all project files
COPY . .

# Expose port 3000
EXPOSE 3000

# Start the React development server
CMD ["npm", "start"]
```



weather-application > package-lock.json > ...
You, last month | 1 author (You)
You, 2 months ago • Initialize project using Create React App

```json
{
  "name": "weather-application",
  "version": "0.1.0",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "name": "weather-application",
      "version": "0.1.0",
      "dependencies": {
        "@testing-library/dom": "^10.4.0",
        "@testing-library/jest-dom": "^6.6.3",
        "@testing-library/react": "^16.3.0",
        "@testing-library/user-event": "^13.5.0",
        "axios": "^1.9.0",
        "react": "^19.1.0",
        "react-dom": "^19.1.0",
        "react-scripts": "5.0.1",
        "web-vitals": "^2.1.4"
      }
    },
    "node_modules/@adobe/css-tools": {
      "version": "4.4.2",
      "resolved": "https://registry.npmjs.org/@adobe/css-tools/-/css-tools-4.4.2.tgz",
      "integrity": "sha512-baYZExFpsdkBNuvGKTKWCwKH57HRZLVtycZS05WTQNVOiXVSeAki3nU35zlRbToeMW8aHlJfyS+1C4BOv27q0A==",
      "license": "MIT"
    },
    "node_modules/@alloc/quick-lru": {
      "version": "5.2.0",
      "resolved": "https://registry.npmjs.org/@alloc/quick-lru/-/quick-lru-5.2.0.tgz",
      "integrity": "sha512-UrcABB+4bUrFABwbluTIBErXwvbsU/V7TZWfmbgJfbkwiBuziS9gxdODUyuiecfdGQ85jglMW6juS3+z5TsKLw==",
      "license": "MIT",
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/@ampproject/remapping": {
      "version": "2.3.0",
      "resolved": "https://registry.npmjs.org/@ampproject/remapping/-/remapping-2.3.0.tgz",
      "integrity": "sha512-30iZtAPgz+LTIYoeivqYo853f02jBYSd5uGnGpkFV0M3xOt9aN73erkgYAmZU43x4VfqcnLxW9Kpg3R5LC4YYw==",
      "license": "Apache-2.0",
      "dependencies": {
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.24"
      },
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@babel/code-frame": {
      "version": "7.27.1",
      "resolved": "https://registry.npmjs.org/@babel/code-frame/-/code-frame-7.27.1.tgz",
```

CHAPTER 4
CONCLUSION

This project illustrates how DevOps practices can be applied to a real-world React application through Dockerization and CI/CD automation. Using GitHub Actions, we automated the entire pipeline—from code push to deployment—minimizing human intervention and accelerating the release cycle.

Docker ensured environmental consistency, allowing the weather app to behave identically across development, testing, and production. GitHub Actions enabled seamless integration of automation without third-party CI tools.

The combination of containerization and CI/CD not only enhanced delivery but also introduced scalability and maintainability to the application. The project reflects the core values of DevOps—automation, continuous improvement, and rapid delivery—making it a strong foundation for future projects involving microservices, cloud-native deployments, or infrastructure as code.