Experiment 7

### **Aim**

To deploy a full-stack web application consisting of a Node.js (Express) backend and a React frontend using **Docker**, **GitHub Actions**, and **Vercel** for continuous integration and automated deployment.

### **Theory**

**1. Introduction** DevOps combines **development** and **operations** practices to automate software delivery. With Docker, developers can containerize their applications, ensuring consistent environments across development, testing, and production.  
 Vercel, a cloud hosting platform, simplifies the deployment of frontend and serverless backend applications directly from GitHub. Integrating Vercel with **GitHub Actions** allows for continuous integration and deployment (CI/CD) of containerized or pre-built applications.

**2. Key Concepts**

* **Docker:** Packages code and dependencies into portable containers.
* **GitHub Actions:** Automates building, testing, and deployment of code.
* **Vercel:** Deploys web apps instantly from GitHub repositories.
* **CI/CD Pipeline:** Ensures that every push to the main branch is automatically tested and deployed.

**3. Advantages**

* **Automated Deployments:** Code changes are deployed immediately after passing CI checks.
* **Scalability:** Vercel automatically scales frontend and API routes.
* **Consistency:** Docker ensures identical local and production environments.
* **Speed:** CI/CD reduces manual steps and deployment errors.

### **Procedure**

#### **1. Project Setup**

**Folder structure:**

fullstack-app/

├── backend/ (Node.js + Express)

├── frontend/ (React)

├── docker-compose.yml

└── .github/workflows/vercel-deploy.yml

**Backend Example (Express API)**

const express = require('express');

const app = express();

app.get('/api/hello', (req, res) => {

res.json({ message: 'Hello from backend!' });

});

app.listen(4000, () => console.log('Backend running on port 4000'));

**Frontend Example (React App)**

import React, { useEffect, useState } from 'react';

function App() {

const [msg, setMsg] = useState('');

useEffect(() => {

fetch('/api/hello').then(res => res.json()).then(data => setMsg(data.message));

}, []);

return (<div><h1>Interview Simulator</h1><p>{msg}</p></div>);

}

export default App;

#### **2. Dockerization (for Local Development)**

**Backend Dockerfile**

FROM node:18-alpine

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 4000

CMD ["npm", "start"]

**Frontend Dockerfile**

FROM node:18-alpine as build

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

EXPOSE 3000

CMD ["npm", "start"]

**docker-compose.yml**

version: '3.8'

services:

backend:

build: ./backend

ports: ["4000:4000"]

frontend:

build: ./frontend

ports: ["3000:3000"]

depends\_on:

- backend

Run locally:

docker-compose up --build

#### **3. CI/CD with GitHub Actions + Vercel**

To automate testing and deployment to **Vercel**, add the following workflow in  
 .github/workflows/vercel-deploy.yml:

name: CI and Deploy to Vercel

on:

push:

branches: [ main ]

jobs:

build-and-deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Install dependencies and build

run: |

npm install

npm run build

- name: Deploy to Vercel

uses: amondnet/vercel-action@v25

with:

vercel-token: ${{ secrets.VERCEL\_TOKEN }}

vercel-org-id: ${{ secrets.VERCEL\_ORG\_ID }}

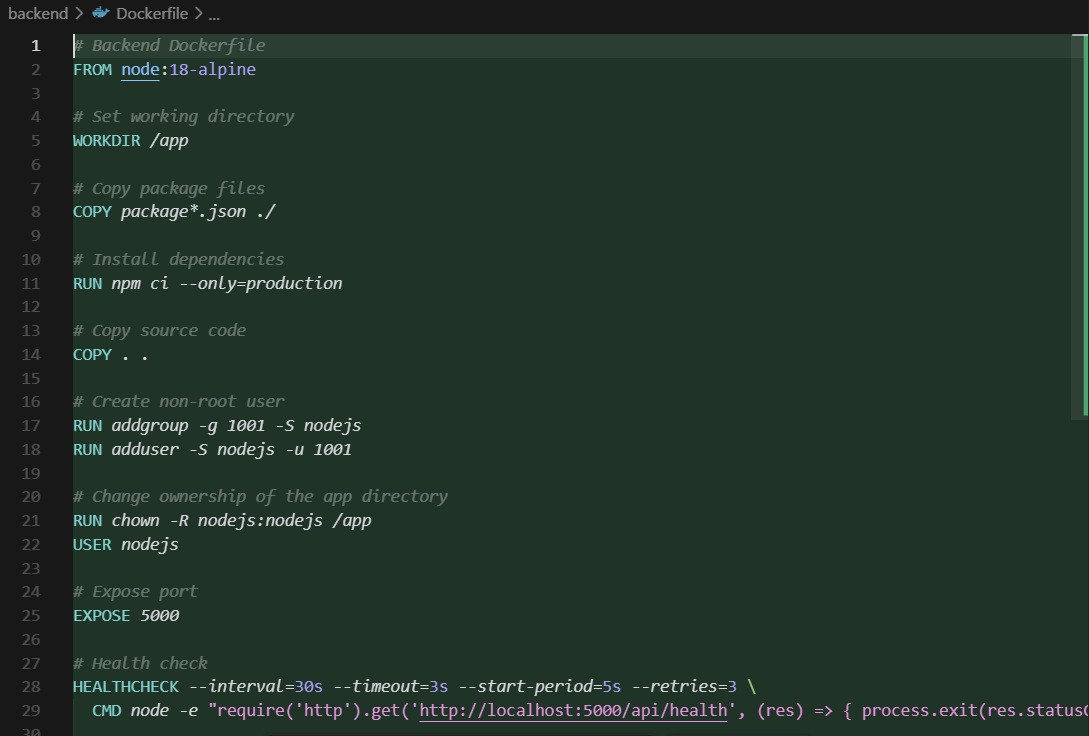
vercel-project-id: ${{ secrets.VERCEL\_PROJECT\_ID }}

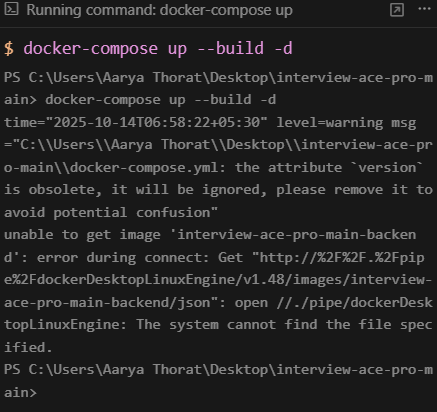
**Secrets Configuration (GitHub → Settings → Secrets → Actions):**

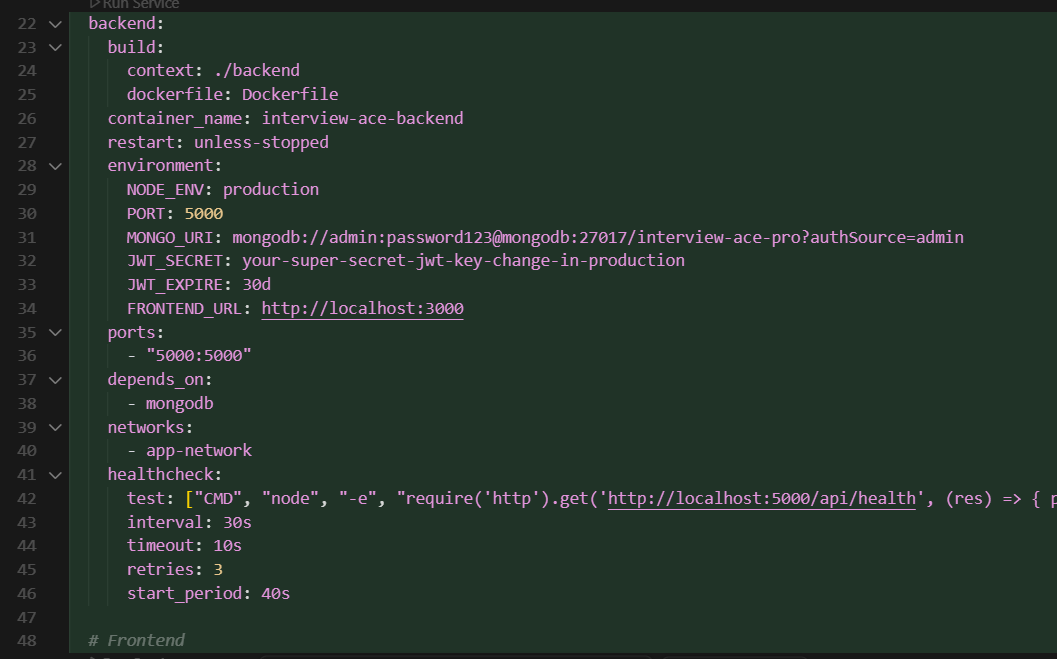
* VERCEL\_TOKEN
* VERCEL\_ORG\_ID
* VERCEL\_PROJECT\_ID

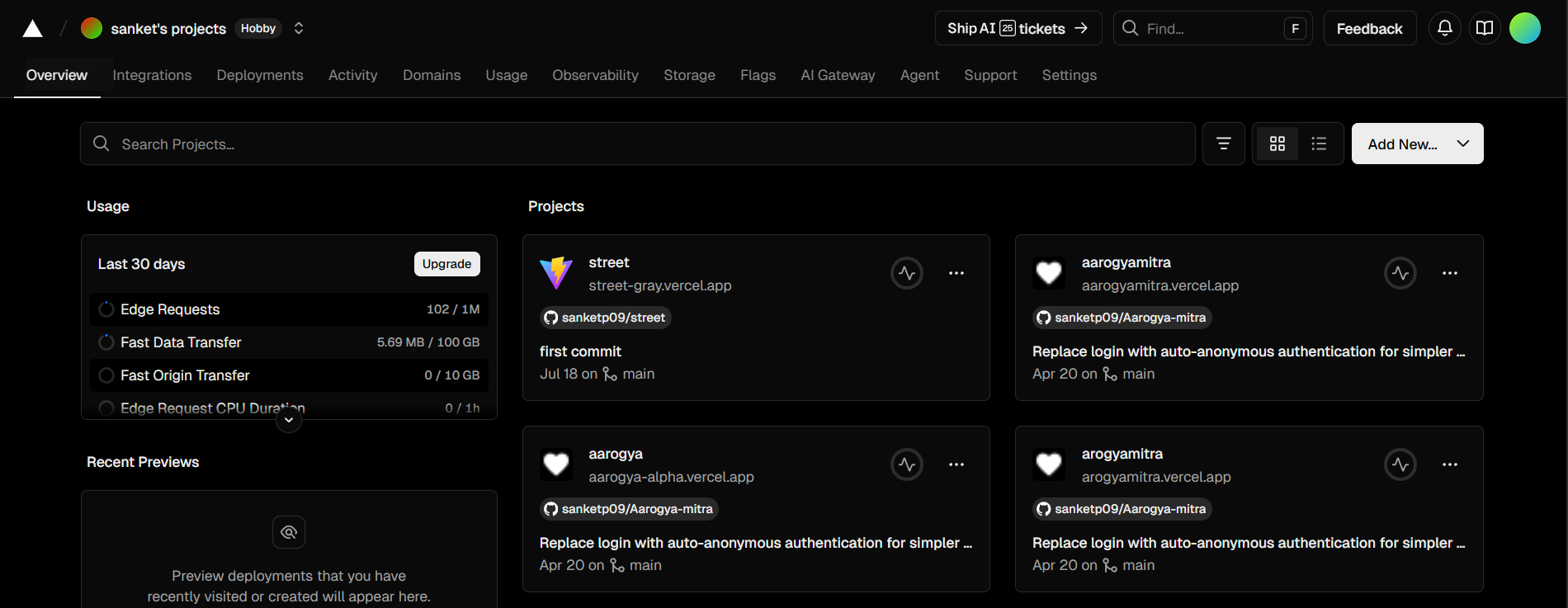
Whenever code is pushed to the **main** branch:

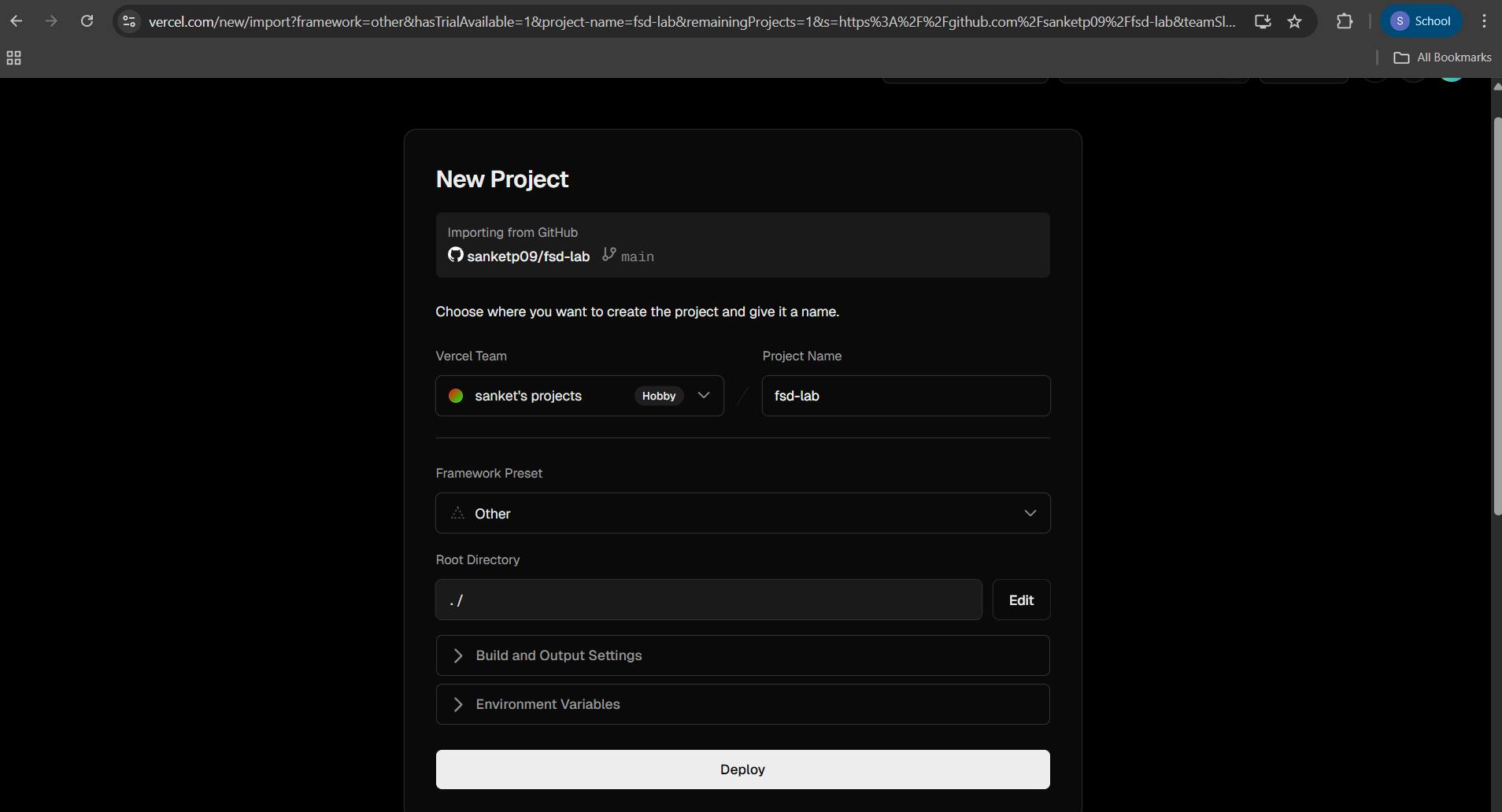
1. GitHub Actions installs dependencies and builds the app.
2. If successful, it deploys automatically to **Vercel**.

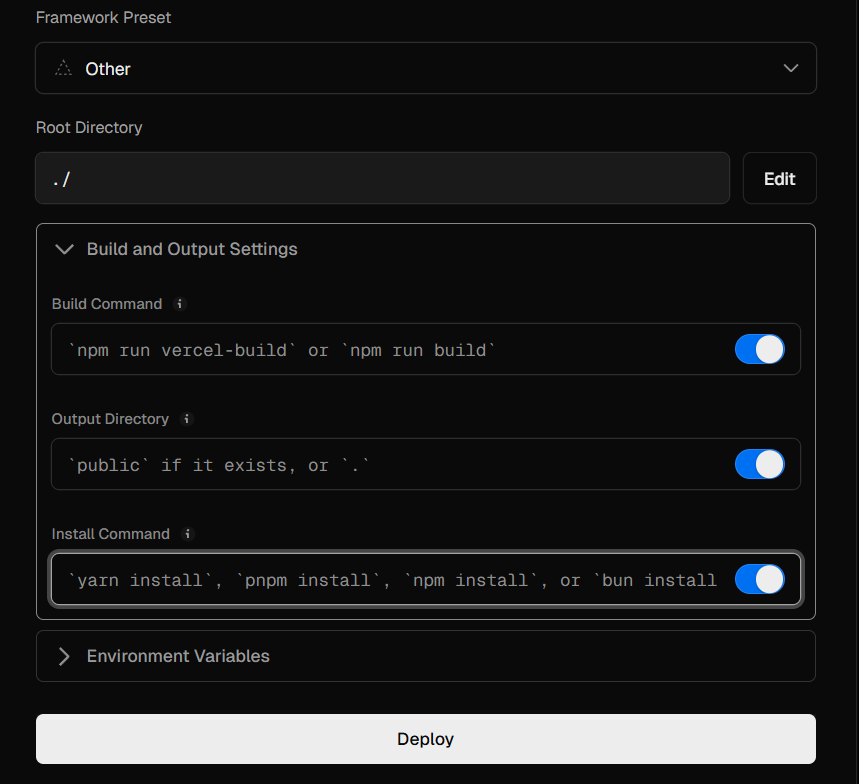


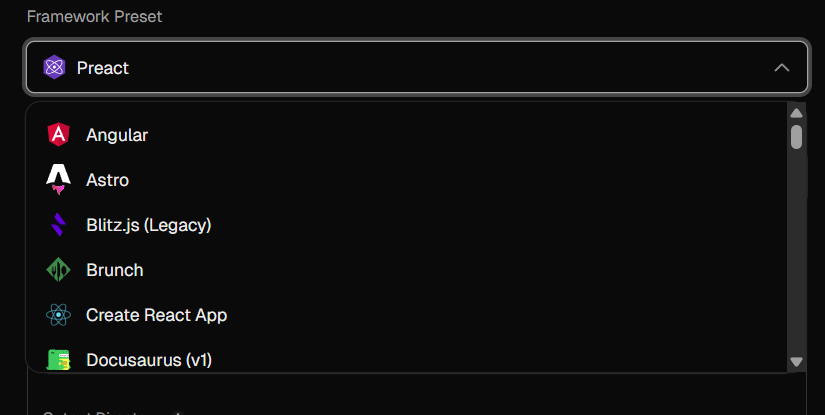


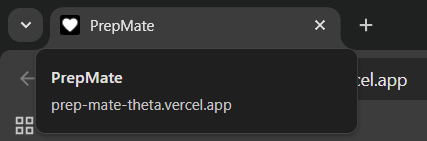


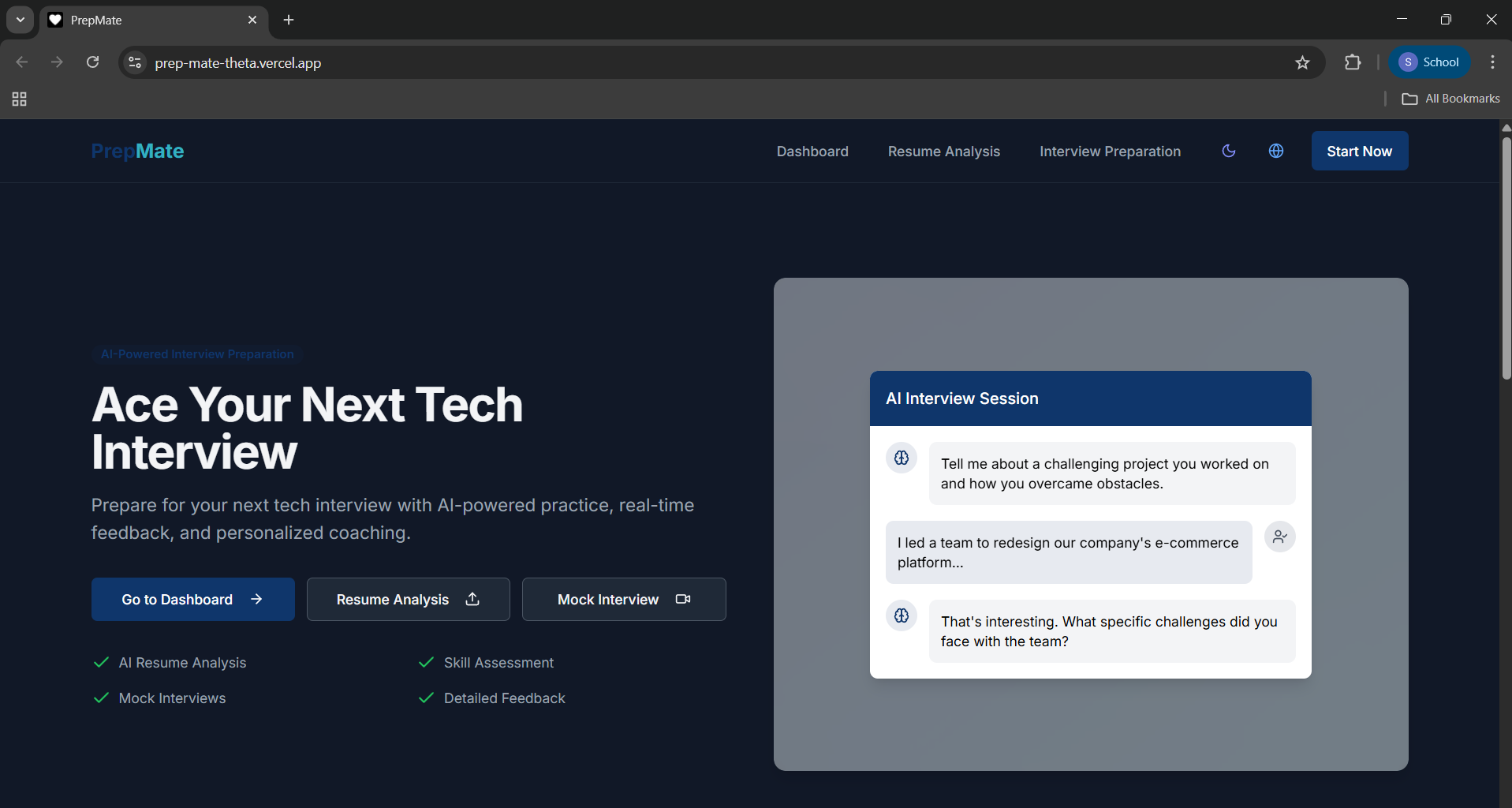












### 

### **Conclusion**

This experiment successfully demonstrated the deployment of a full-stack application using **Docker**, **GitHub Actions**, and **Vercel**.  
 By integrating these technologies, the workflow became **fully automated**, enabling instant deployment upon each push to GitHub.  
 Vercel handled hosting and scaling seamlessly, while Docker ensured consistency between development and production environments.

The result is a **modern, cloud-native CI/CD pipeline** — portable, efficient, and reliable — suitable for full-stack applications like your **Interview Simulator Website**.