**EXPERIMENT 10- CI/CD Deployment with GitHub Actions + Render/Vercel**

## 

### **Aim**

To implement Continuous Integration and Continuous Deployment (CI/CD) for the Interview Simulator Website using GitHub Actions and deploy it automatically to cloud hosting platforms like Render and Vercel after successful testing.

### **Theory**

**1. Introduction to CI/CD**  
 CI/CD (Continuous Integration and Continuous Deployment) is a modern DevOps practice that automates the process of integrating code changes, testing, and deploying applications. It ensures that every code commit is tested, built, and deployed automatically, reducing manual errors and accelerating release cycles.

**2. Continuous Integration (CI)**  
 CI focuses on merging all developer code into a shared repository several times a day. Each merge triggers an automated pipeline that installs dependencies, runs lint checks, and executes test cases to ensure code quality.

**3. Continuous Deployment (CD)**  
 CD automates the release of validated code to production or staging servers. Once the CI pipeline passes all tests, the CD stage deploys the latest build automatically to a hosting service (e.g., Render or Vercel).

**4. GitHub Actions**  
 GitHub Actions is a workflow automation tool integrated into GitHub. It allows developers to define YAML-based workflows that can build, test, and deploy applications directly from the repository. Each workflow consists of jobs and steps triggered by events like push, pull\_request, or workflow\_dispatch.

5. **Render and Vercel Overview**

* Render: A cloud hosting platform that supports web services, background workers, and databases. It provides a deployment API to trigger builds programmatically.
* Vercel: A platform optimized for frontend frameworks like React, Next.js, and static sites. It integrates directly with GitHub and supports pre-built deployments through GitHub Actions.

**6. Benefits of CI/CD Integration**

* Faster and more reliable deployment cycles
* Early bug detection through automated testing
* Consistent environments between staging and production
* Improved developer collaboration and productivity

### **Procedure**

#### **1. Prerequisites**

1. Create a GitHub repository and push your project code.
2. Add necessary secrets in GitHub → Settings → Secrets and Variables → Actions.  
   * Render Deployment  
     + RENDER\_API\_KEY → from Render dashboard
     + RENDER\_SERVICE\_ID → Render service ID (e.g., srv-xxxxx)
   * Vercel Deployment  
     + VERCEL\_TOKEN
     + VERCEL\_ORG\_ID
     + VERCEL\_PROJECT\_ID

#### 2. CI + Deploy to Render

Step 1: Create Workflow File  
 Create .github/workflows/render-deploy.yml

name: CI and Deploy to Render

on:

push:

branches: [ main ]

jobs:

build-and-deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout repository

uses: actions/checkout@v4

- name: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: 18

- name: Install dependencies

run: npm install

- name: Run tests

run: npm test

- name: Deploy to Render (only if tests pass)

if: success()

uses: johnbeynon/render-deploy-action@v1

with:

serviceId: ${{ secrets.RENDER\_SERVICE\_ID }}

apiKey: ${{ secrets.RENDER\_API\_KEY }}

Step 2: Verify

* When a commit is pushed to the main branch, GitHub Actions runs tests.
* If all tests pass, Render API is triggered automatically to deploy the latest version.

#### 3. CI + Deploy to Vercel

Step 1: Create Workflow File  
 Create .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: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: 18

- 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 }}

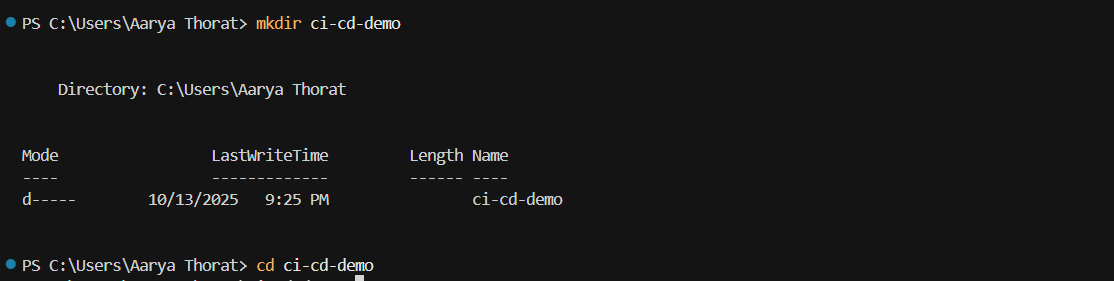
vercel-org-id: ${{ secrets.VERCEL\_ORG\_ID }}

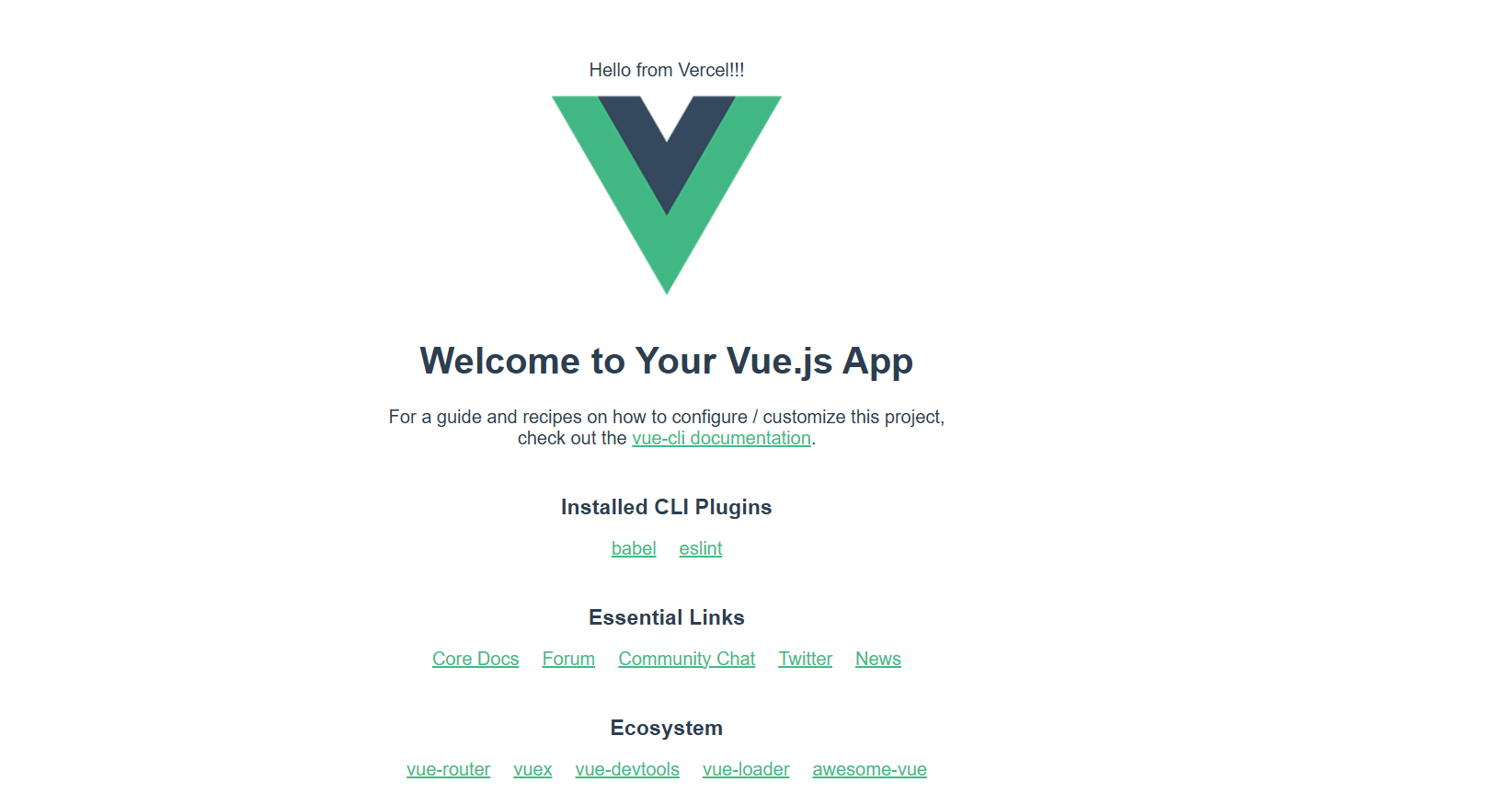
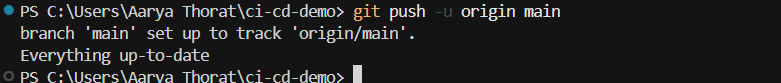
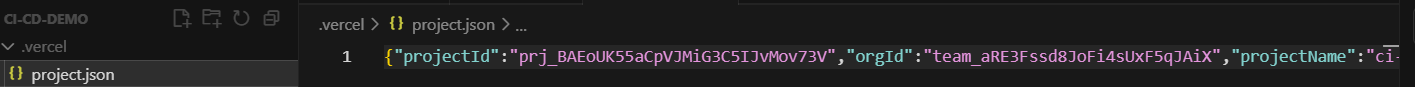
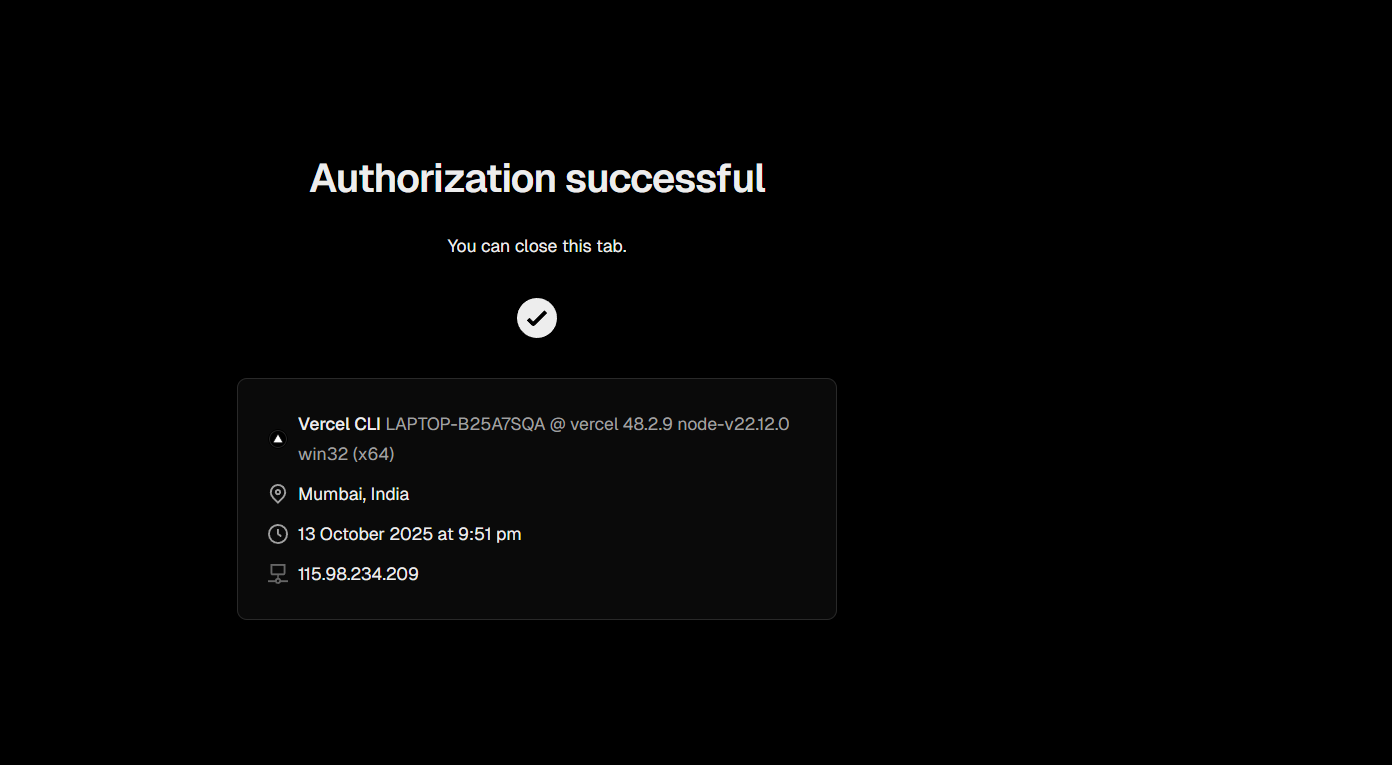
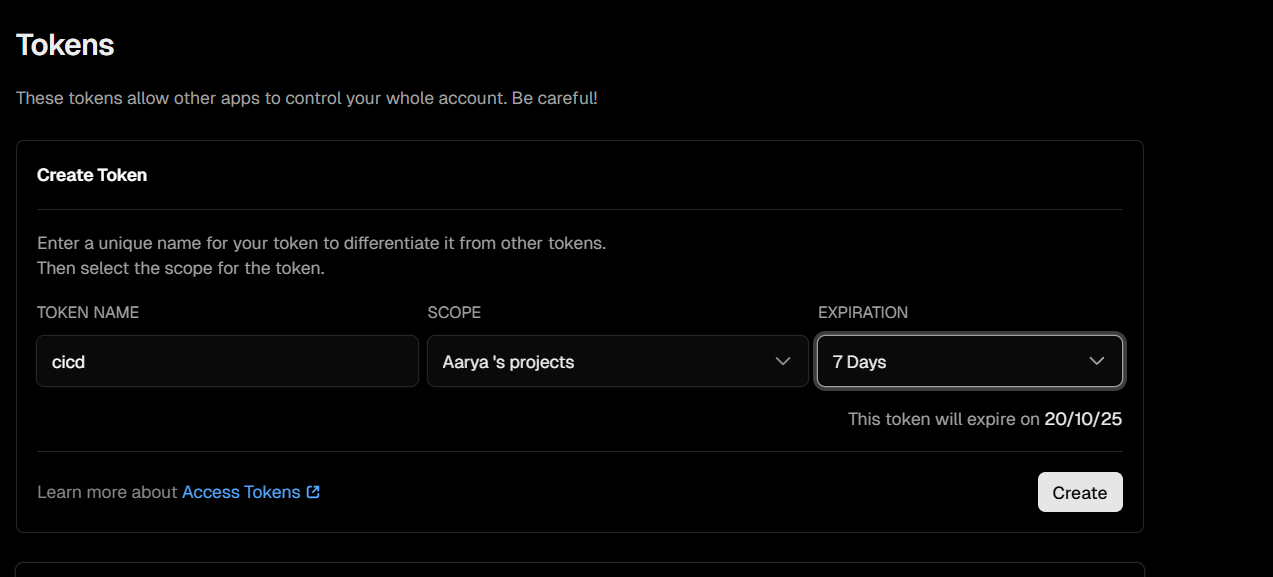
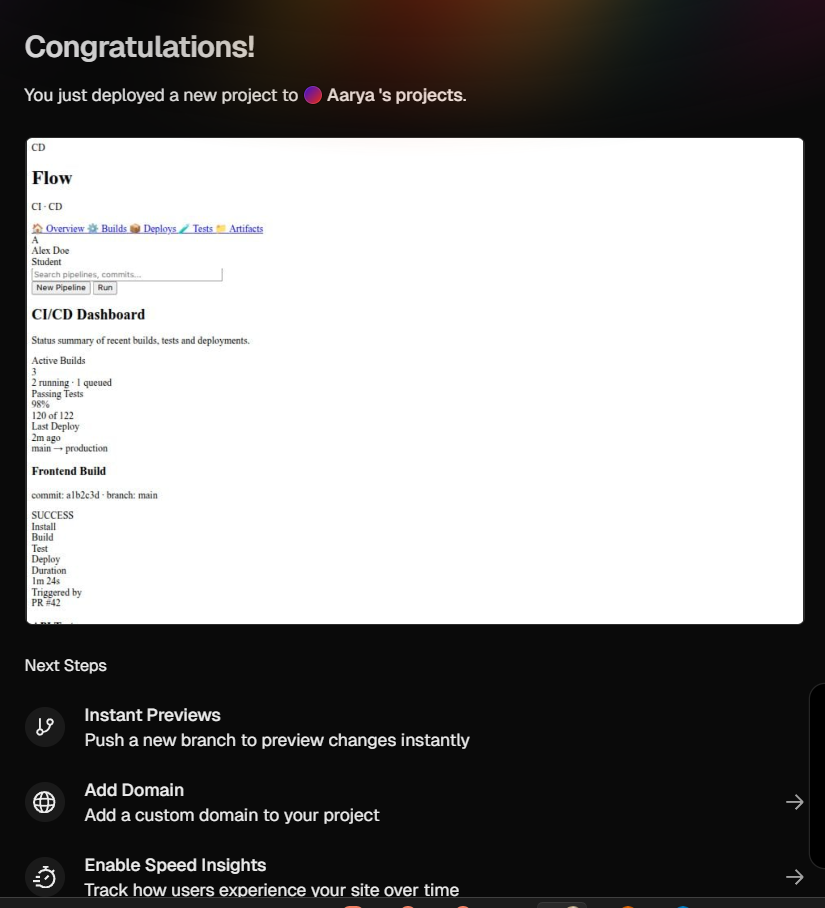
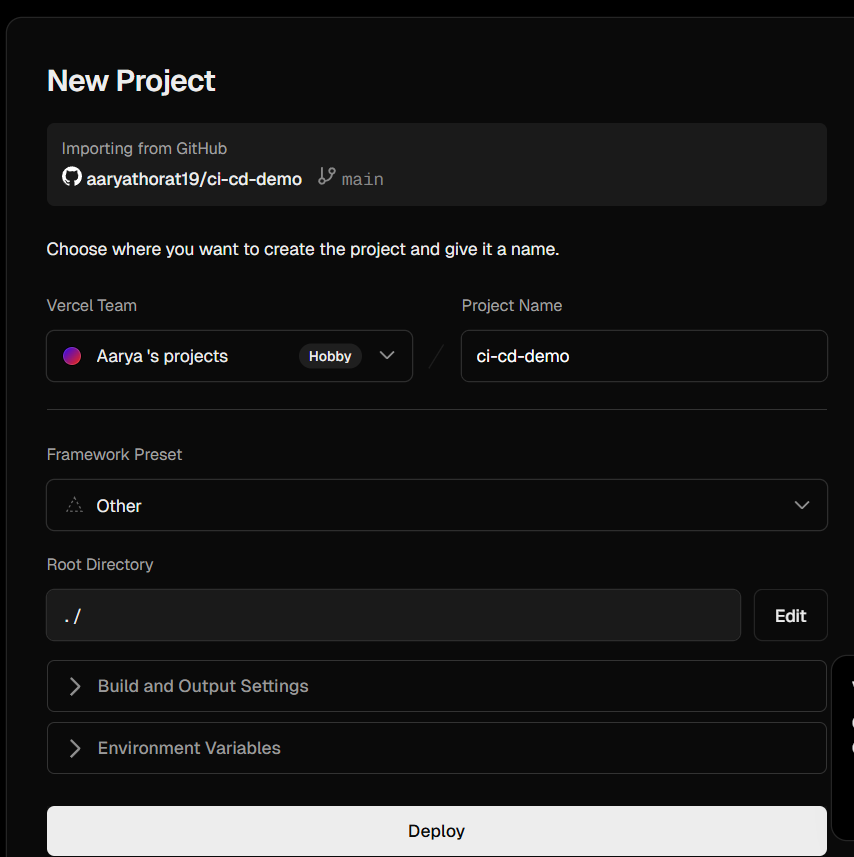
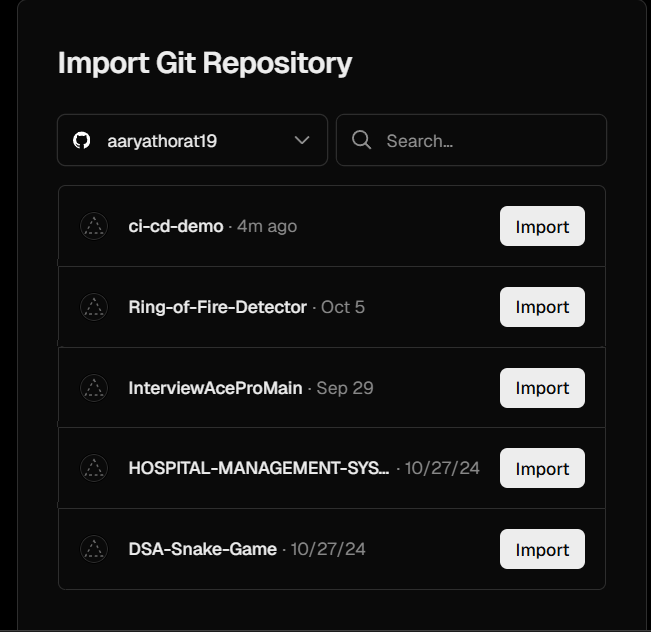
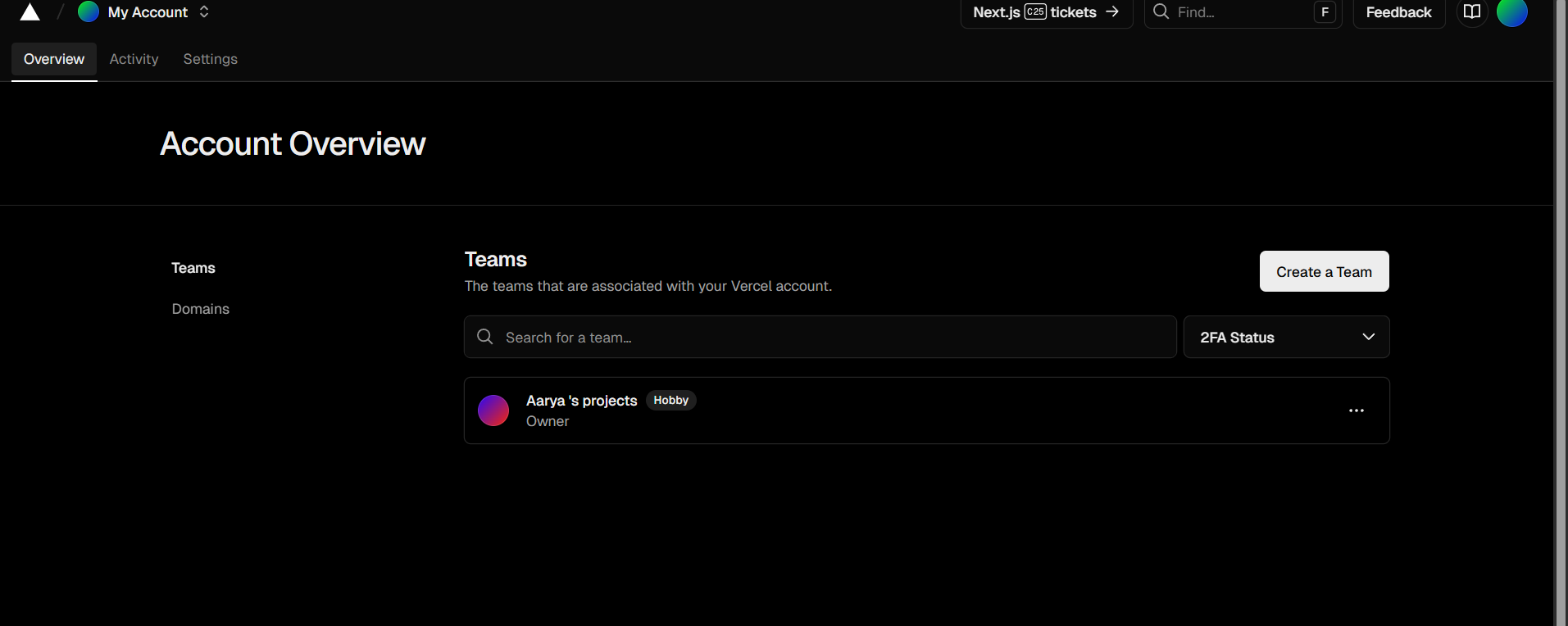
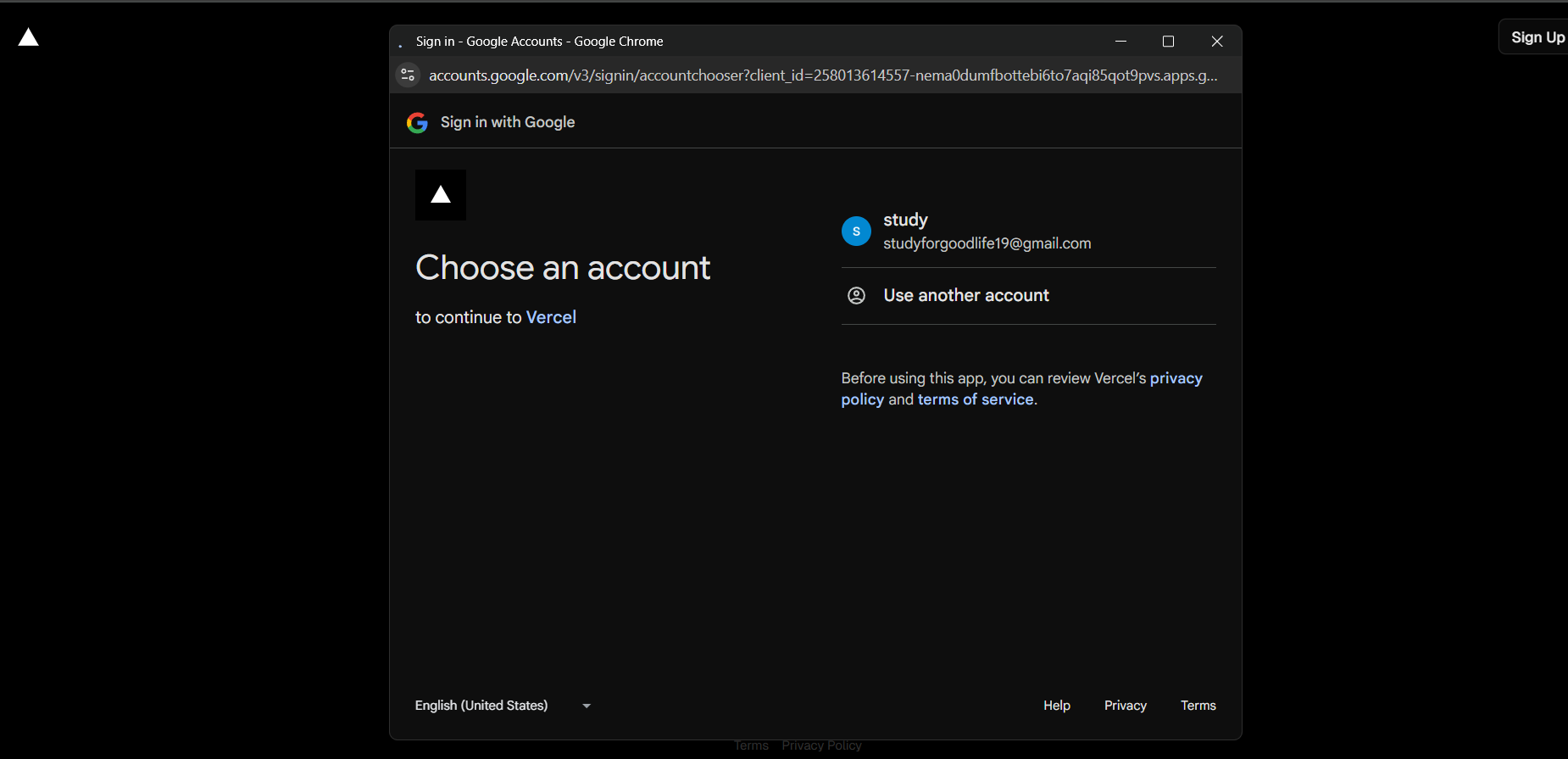
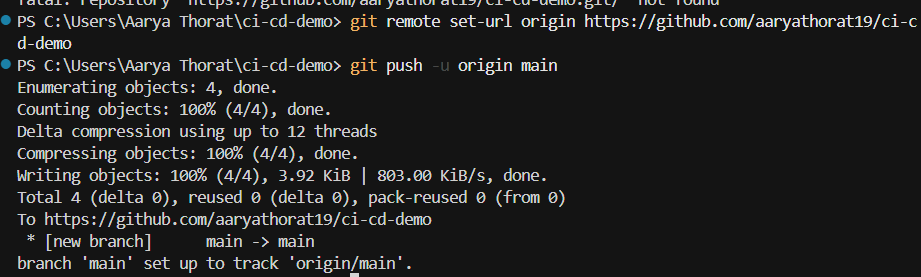
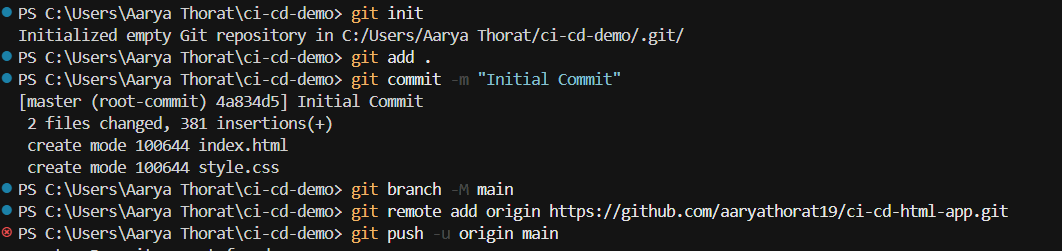
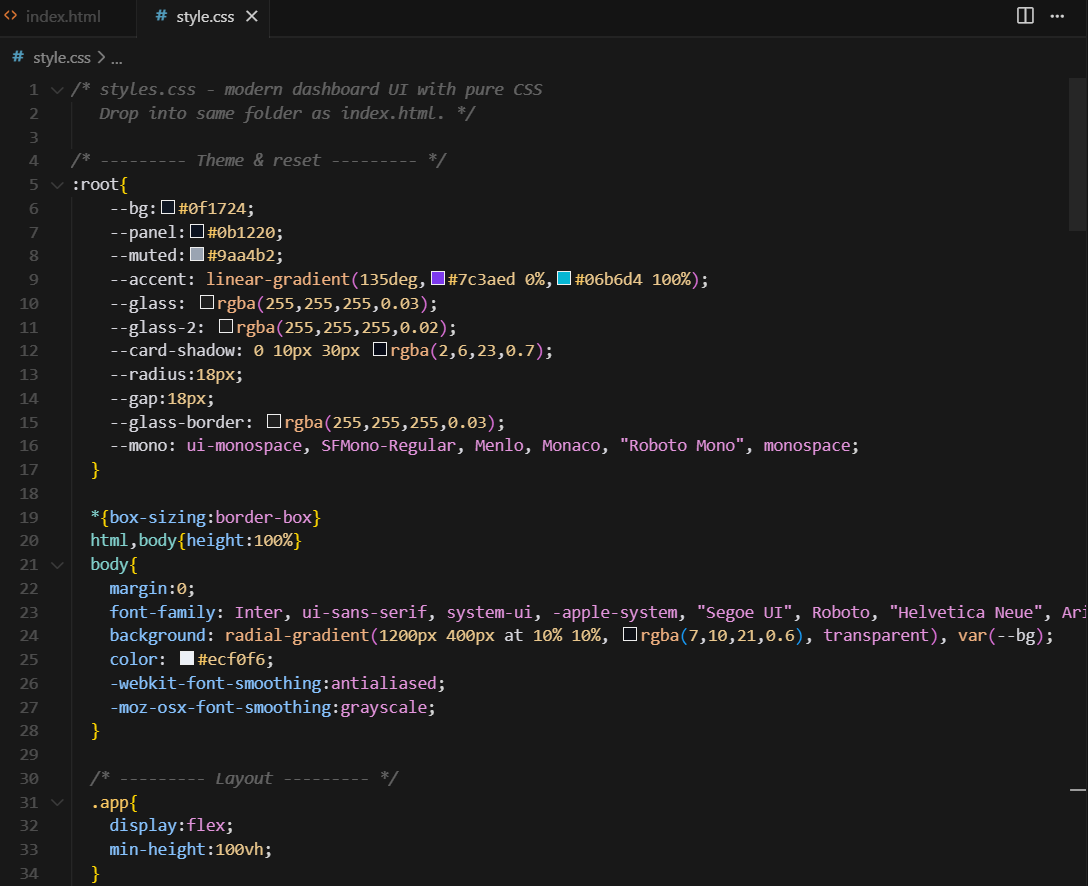
vercel-project-id: ${{ secrets.VERCEL\_PROJECT\_ID }}

working-directory: ./

Step 2: Verify

* On each push to main, GitHub Actions builds the app and deploys the build output to Vercel automatically.
* You can monitor progress under the Actions tab in GitHub.

****

****

### **Conclusion**

Integrating CI/CD pipelines with GitHub Actions and deploying via Render/Vercel automates the entire development workflow — from testing to deployment. For the Interview Simulator Website, this setup ensures that every code update is tested and deployed automatically, maintaining a live, stable, and up-to-date version for users.

This approach improves productivity, reliability, and consistency across environments, embodying modern DevOps best practices.  
 In production, additional enhancements such as code coverage reports, environment-specific workflows, and notifications (via Slack or email) can be added to further optimize the deployment process**.**