**📄 Final Report: Portfolio Skills Manager**

**Cover Page & Table of Contents**

**Project Title:** Portfolio Skills Manager  
**Author:** Vikram Somai  
**Submission Date:** [Insert Date]

**Table of Contents**

1. Introduction & Objectives
2. Tech Stack & Frameworks Used
3. Project Architecture
4. CI/CD Pipeline Overview
5. User Stories & Requirements
6. Database Schema & API Endpoints
7. Wireframes or UI Prototypes
8. Screenshots of Application
9. Testing Strategy (Optional)
10. Conclusion / Lessons Learned

**1. 🌐 Introduction & Objectives**

The "Portfolio Skills Manager" is a full-stack web application designed to showcase personal projects and manage technical skills. It also allows visitors to contact the portfolio owner through a dedicated form. An admin panel facilitates CRUD operations on skills and displays contact messages.

**Objectives:**

* Showcase developer's portfolio
* Manage categorized skills with levels
* Provide a contact form for collaboration or hiring inquiries
* Automate deployment pipeline with GitHub Actions

**2. 🚀 Tech Stack & Frameworks Used**

**Frontend**

* React 18 (TypeScript)
* React Router DOM
* Tailwind CSS
* Axios

**Backend**

* Node.js + Express.js
* MongoDB + Mongoose
* express-validator
* Helmet, Morgan

**Cloud & DevOps**

* AWS S3 (frontend hosting)
* AWS CloudFront (CDN)
* AWS EC2 (backend hosting)
* GitHub Actions (CI/CD)
* MongoDB Atlas

**3. 🛠️ Project Architecture**

**Frontend:** React app served via AWS S3 and CloudFront.  
**Backend:** Express API hosted on AWS EC2 with PM2.  
**Database:** MongoDB Atlas.

User (Browser)

|

v

React App (Frontend - S3)

|

v

Express.js API (Backend - EC2)

|

v

MongoDB Atlas (Database)

**4. 🛠️ CI/CD Pipeline Overview**

* GitHub Actions used for build, test, and deploy.
* On main branch push, actions run:
  + Install frontend & backend dependencies
  + Lint + test frontend
  + Build React app
  + Upload artifacts
  + Deploy frontend to S3
  + Invalidate CloudFront
  + Deploy backend to EC2 via SSH

**5. 📊 User Stories & Requirements**

* As a **visitor**, I want to view portfolio and skills.
* As a **user**, I want to contact the portfolio owner.
* As an **admin**, I want to add, edit, delete skills.
* As an **admin**, I want to view contact submissions.

**6. 🔠 Database Schema & API Endpoints**

**Skill Schema**

{

name: String,

level: "Beginner" | "Intermediate" | "Advanced",

category: "Frontend" | "Backend" | "Database" | "Cloud" | "DevOps"

}

**API Endpoints**

* GET /api/skills
* POST /api/skills
* PUT /api/skills/:id
* DELETE /api/skills/:id
* GET /api/contacts
* POST /api/auth/login

**7. 🎭 Wireframes or UI Prototypes**

* Figma design created for Admin Panel and Home.
* Responsive design checked in dev tools.

*(Add screenshots of your Figma or wireframes here)*

**8. 📷 Screenshots of Application**

* Home Page
* Admin Login
* Skill CRUD Operations
* Contact Messages View

*(Insert actual screenshots here in the final version)*

**9. 🔎 Testing Strategy (Optional)**

* ESLint used for linting with TypeScript rules.
* API tested using Postman.
* GitHub Actions auto-run tests on each push.

**10. 📖 Conclusion / Lessons Learned**

Through this project, I learned how to:

* Build a production-ready MERN stack application
* Use GitHub Actions for CI/CD
* Host frontend and backend on AWS
* Manage roles and JWT auth
* Handle real-world full-stack problems from design to deployment

**Future Improvements:**

* Add email notifications
* Add user profile and project sections
* Expand authentication to social logins

**Thank you!**  
*Prepared by Vikram Somai*

I'll provide a comprehensive step-by-step guide to create and deploy everything from scratch. This will be a complete walkthrough covering every aspect of the project.

## Complete Step-by-Step Implementation Guide

### Phase 1: Initial Setup & Environment Preparation

#### Step 1: Install Required Software

\*\*1.1 Install Node.js\*\*

```shellscript

# Download and install Node.js from https://nodejs.org/

# Verify installation

node --version

npm --version

```

\*\*1.2 Install Git\*\*

```shellscript

# Download from https://git-scm.com/

# Verify installation

git --version

```

\*\*1.3 Install Visual Studio Code\*\*

```shellscript

# Download from https://code.visualstudio.com/

# Install useful extensions:

# - ES7+ React/Redux/React-Native snippets

# - Prettier - Code formatter

# - GitLens

# - Thunder Client (for API testing)

```

\*\*1.4 Install AWS CLI\*\*

```shellscript

# Windows (using installer from AWS website)

# macOS

brew install awscli

# Linux

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

# Verify installation

aws --version

```

#### Step 2: Create Project Structure

\*\*2.1 Create Main Project Directory\*\*

```shellscript

mkdir portfolio-skills-manager

cd portfolio-skills-manager

# Create project structure

mkdir frontend

mkdir backend

mkdir docs

mkdir scripts

```

\*\*2.2 Initialize Git Repository\*\*

```shellscript

git init

echo "node\_modules/" > .gitignore

echo ".env" >> .gitignore

echo "build/" >> .gitignore

echo "dist/" >> .gitignore

```

### Phase 2: Frontend Development (React Application)

#### Step 3: Create React Application

\*\*3.1 Initialize React App\*\*

```shellscript

cd frontend

npx create-react-app . --template typescript

```

\*\*3.2 Install Additional Dependencies\*\*

```shellscript

npm install react-router-dom axios

npm install -D @types/react-router-dom

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

```

\*\*3.3 Configure Tailwind CSS\*\*

```typescriptreact project="portfolio-app"

...

```

#### Step 4: Test Frontend Application

\*\*4.1 Start Development Server\*\*

```shellscript

cd frontend

npm start

```

\*\*4.2 Verify Application\*\*

- Open [http://localhost:3000](http://localhost:3000)

- Navigate through all pages

- Test responsive design

- Check console for errors

### Phase 3: Backend Development (Node.js + Express + MongoDB)

#### Step 5: Create Backend Application

\*\*5.1 Initialize Backend\*\*

```shellscript

cd ../backend

npm init -y

```

\*\*5.2 Install Dependencies\*\*

```shellscript

npm install express mongoose cors dotenv helmet morgan

npm install -D nodemon @types/node concurrently

```

\*\*5.3 Create Backend Structure\*\*

```typescriptreact project="portfolio-app"

...

```

#### Step 6: Database Setup and Seeding

\*\*6.1 Install MongoDB\*\*

```shellscript

# For macOS

brew tap mongodb/brew

brew install mongodb-community

# For Windows - Download from MongoDB website

# For Linux

wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

sudo apt-get update

sudo apt-get install -y mongodb-org

```

\*\*6.2 Start MongoDB\*\*

```shellscript

# macOS/Linux

sudo systemctl start mongod

# Or using brew (macOS)

brew services start mongodb-community

# Windows - Start as service or run mongod.exe

```

\*\*6.3 Create Database Seed Script\*\*

```typescriptreact project="Database Setup"

...

```

\*\*6.4 Run Database Scripts\*\*

```shellscript

cd backend

# Seed the database

node scripts/seedDatabase.js

# If you need to clear the database

node scripts/clearDatabase.js

```

#### Step 7: Test Backend API

\*\*7.1 Start Backend Server\*\*

```shellscript

cd backend

npm run dev

```

\*\*7.2 Test API Endpoints\*\*

```shellscript

# Test health endpoint

curl http://localhost:5000/api/health

# Test skills endpoint

curl http://localhost:5000/api/skills

# Test creating a skill

curl -X POST http://localhost:5000/api/skills \

-H "Content-Type: application/json" \

-d '{"name":"Vue.js","level":"Beginner","category":"Frontend"}'

# Test contact endpoint

curl -X POST http://localhost:5000/api/contacts \

-H "Content-Type: application/json" \

-d '{"name":"Test User","email":"test@example.com","message":"This is a test message from the API"}'

```

### Phase 4: GitHub Setup and Version Control

#### Step 8: Create GitHub Repository

\*\*8.1 Create Repository on GitHub\*\*

1. Go to [https://github.com](https://github.com)

2. Click "New repository"

3. Name: `portfolio-skills-manager`

4. Description: `Full-stack portfolio management application with React, Node.js, MongoDB, and AWS deployment`

5. Make it public

6. Initialize with README

\*\*8.2 Clone and Setup Local Repository\*\*

```shellscript

# Go back to project root

cd ..

# Clone the repository

git clone https://github.com/yourusername/portfolio-skills-manager.git

cd portfolio-skills-manager

# Copy your existing code

cp -r ../frontend ./

cp -r ../backend ./

# Create project structure

mkdir docs

mkdir scripts

```

\*\*8.3 Create Comprehensive .gitignore\*\*

```shellscript

# Create .gitignore

cat > .gitignore << 'EOF'

# Dependencies

node\_modules/

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

# Production builds

build/

dist/

# Environment variables

.env

.env.local

.env.development.local

.env.test.local

.env.production.local

# IDE files

.vscode/

.idea/

\*.swp

\*.swo

\*~

# OS generated files

.DS\_Store

.DS\_Store?

.\_\*

.Spotlight-V100

.Trashes

ehthumbs.db

Thumbs.db

# Logs

logs

\*.log

# Runtime data

pids

\*.pid

\*.seed

\*.pid.lock

# Coverage directory used by tools like istanbul

coverage/

# AWS

.aws/

# Temporary files

tmp/

temp/

EOF

```

\*\*8.4 Create README.md\*\*

```markdown

# Portfolio Skills Manager

A full-stack web application for managing personal portfolio and skills, built with React, Node.js, MongoDB, and deployed on AWS.

## 🚀 Features

- \*\*Portfolio Display\*\*: Showcase projects and experience

- \*\*Skills Management\*\*: CRUD operations for technical skills

- \*\*Contact Form\*\*: Allow visitors to send messages

- \*\*Admin Panel\*\*: Manage skills and view contact submissions

- \*\*Responsive Design\*\*: Mobile-first approach with Tailwind CSS

- \*\*API Integration\*\*: RESTful API with Express.js and MongoDB

- \*\*Cloud Deployment\*\*: AWS S3, CloudFront, and EC2

- \*\*CI/CD Pipeline\*\*: Automated deployment with GitHub Actions

## 🛠️ Technology Stack

### Frontend

- React 18 with TypeScript

- React Router for navigation

- Tailwind CSS for styling

- Axios for API calls

### Backend

- Node.js with Express.js

- MongoDB with Mongoose ODM

- Express Validator for input validation

- Helmet for security headers

- Morgan for logging

### Cloud & DevOps

- AWS S3 for static hosting

- AWS CloudFront for CDN

- AWS EC2 for backend hosting

- GitHub Actions for CI/CD

- MongoDB Atlas for database hosting

## 📋 Prerequisites

- Node.js (v16 or higher)

- MongoDB (local or Atlas)

- Git

- AWS CLI (for deployment)

## 🚀 Quick Start

### 1. Clone the repository

```bash

git clone https://github.com/yourusername/portfolio-skills-manager.git

cd portfolio-skills-manager

```

### 2. Setup Backend

```shellscript

cd backend

npm install

cp .env.example .env

# Edit .env with your configuration

npm run dev

```

### 3. Setup Frontend

```shellscript

cd ../frontend

npm install

npm start

```

### 4. Seed Database

```shellscript

cd ../backend

node scripts/seedDatabase.js

```

## Project Structure

```plaintext

portfolio-skills-manager/

├── frontend/ # React application

│ ├── public/

│ ├── src/

│ │ ├── components/ # Reusable components

│ │ ├── pages/ # Page components

│ │ ├── services/ # API services

│ │ ├── types/ # TypeScript types

│ │ └── App.tsx

│ └── package.json

├── backend/ # Node.js API

│ ├── models/ # MongoDB models

│ ├── routes/ # API routes

│ ├── scripts/ # Database scripts

│ ├── server.js # Main server file

│ └── package.json

├── docs/ # Documentation

├── .github/workflows/ # GitHub Actions

└── README.md

```

## API Endpoints

### Skills

- `GET /api/skills` - Get all skills

- `POST /api/skills` - Create new skill

- `PUT /api/skills/:id` - Update skill

- `DELETE /api/skills/:id` - Delete skill

### Contacts

- `GET /api/contacts` - Get all contacts (admin)

- `POST /api/contacts` - Submit contact form

- `PUT /api/contacts/:id/status` - Update contact status

## Deployment

### Frontend (AWS S3 + CloudFront)

1. Build the React application

2. Upload to S3 bucket

3. Configure CloudFront distribution

4. Set up custom domain (optional)

### Backend (AWS EC2)

1. Launch EC2 instance

2. Install Node.js and PM2

3. Clone repository and install dependencies

4. Configure environment variables

5. Start application with PM2

### Database (MongoDB Atlas)

1. Create MongoDB Atlas cluster

2. Configure network access

3. Update connection string in environment variables

## CI/CD Pipeline

The project uses GitHub Actions for automated testing and deployment:

1. \*\*Test Stage\*\*: Run unit tests and linting

2. \*\*Build Stage\*\*: Build React application

3. \*\*Deploy Stage\*\*: Deploy to AWS (S3 for frontend, EC2 for backend)

## Monitoring and Analytics

- AWS CloudWatch for application monitoring

- MongoDB Atlas monitoring for database performance

- GitHub Actions for deployment status

## Contributing

1. Fork the repository

2. Create a feature branch (`git checkout -b feature/amazing-feature`)

3. Commit your changes (`git commit -m 'Add some amazing feature'`)

4. Push to the branch (`git push origin feature/amazing-feature`)

5. Open a Pull Request

## License

This project is licensed under the MIT License - see the [LICENSE](LICENSE) file for details.

## Author

\*\*Your Name\*\*

- GitHub: [@yourusername](https://github.com/yourusername)

- LinkedIn: [Your LinkedIn](https://linkedin.com/in/yourprofile)

- Email: [your.email@example.com](mailto:your.email@example.com)

## Acknowledgments

- Built as part of a web development course final project

- Demonstrates modern full-stack development practices

- Showcases cloud deployment and CI/CD implementation

```plaintext

\*\*8.5 Initial Commit\*\*

```bash

git add .

git commit -m "Initial commit: Full-stack portfolio application

- React frontend with TypeScript and Tailwind CSS

- Node.js backend with Express and MongoDB

- Complete CRUD operations for skills management

- Contact form functionality

- Responsive design and modern UI

- Database seeding scripts

- Comprehensive documentation"

git push origin main

```

### Phase 5: AWS Setup and Deployment

#### Step 9: AWS Account Setup

\*\*9.1 Create AWS Account\*\*

1. Go to [https://aws.amazon.com](https://aws.amazon.com)

2. Create new account

3. Verify email and phone

4. Add payment method

5. Choose Basic support plan (free)

\*\*9.2 Configure AWS CLI\*\*

```shellscript

aws configure

# Enter your AWS Access Key ID

# Enter your AWS Secret Access Key

# Default region: us-east-1

# Default output format: json

```

\*\*9.3 Create IAM User for Deployment\*\*

```shellscript

# Create IAM policy for deployment

cat > deployment-policy.json << 'EOF'

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:PutObject",

"s3:DeleteObject",

"s3:ListBucket"

],

"Resource": [

"arn:aws:s3:::your-portfolio-bucket",

"arn:aws:s3:::your-portfolio-bucket/\*"

]

},

{

"Effect": "Allow",

"Action": [

"cloudfront:CreateInvalidation"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"ec2:DescribeInstances",

"ec2:DescribeInstanceStatus"

],

"Resource": "\*"

}

]

}

EOF

# Create IAM policy

aws iam create-policy \

--policy-name PortfolioDeploymentPolicy \

--policy-document file://deployment-policy.json

# Create IAM user

aws iam create-user --user-name portfolio-deployer

# Attach policy to user

aws iam attach-user-policy \

--user-name portfolio-deployer \

--policy-arn arn:aws:iam::YOUR-ACCOUNT-ID:policy/PortfolioDeploymentPolicy

# Create access keys

aws iam create-access-key --user-name portfolio-deployer

```

#### Step 10: Frontend Deployment (S3 + CloudFront)

\*\*10.1 Create S3 Bucket\*\*

```shellscript

# Create S3 bucket (replace with unique name)

BUCKET\_NAME="your-portfolio-$(date +%s)"

aws s3 mb s3://$BUCKET\_NAME --region us-east-1

# Configure bucket for static website hosting

aws s3 website s3://$BUCKET\_NAME \

--index-document index.html \

--error-document error.html

# Create bucket policy for public read access

cat > bucket-policy.json << EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::$BUCKET\_NAME/\*"

}

]

}

EOF

aws s3api put-bucket-policy \

--bucket $BUCKET\_NAME \

--policy file://bucket-policy.json

```

\*\*10.2 Build and Deploy Frontend\*\*

```shellscript

cd frontend

# Create production environment file

cat > .env.production << 'EOF'

REACT\_APP\_API\_URL=https://your-api-domain.com/api

GENERATE\_SOURCEMAP=false

EOF

# Build the application

npm run build

# Deploy to S3

aws s3 sync build/ s3://$BUCKET\_NAME --delete

# Test the deployment

echo "Frontend deployed to: http://$BUCKET\_NAME.s3-website-us-east-1.amazonaws.com"

```

\*\*10.3 Setup CloudFront Distribution\*\*

```shellscript

# Create CloudFront distribution configuration

cat > cloudfront-config.json << EOF

{

"CallerReference": "portfolio-$(date +%s)",

"Comment": "Portfolio Skills Manager Distribution",

"DefaultRootObject": "index.html",

"Origins": {

"Quantity": 1,

"Items": [

{

"Id": "S3-$BUCKET\_NAME",

"DomainName": "$BUCKET\_NAME.s3.amazonaws.com",

"S3OriginConfig": {

"OriginAccessIdentity": ""

}

}

]

},

"DefaultCacheBehavior": {

"TargetOriginId": "S3-$BUCKET\_NAME",

"ViewerProtocolPolicy": "redirect-to-https",

"TrustedSigners": {

"Enabled": false,

"Quantity": 0

},

"ForwardedValues": {

"QueryString": false,

"Cookies": {

"Forward": "none"

}

},

"MinTTL": 0

},

"Enabled": true,

"PriceClass": "PriceClass\_100"

}

EOF

# Create CloudFront distribution

aws cloudfront create-distribution \

--distribution-config file://cloudfront-config.json

```

#### Step 11: Backend Deployment (EC2)

\*\*11.1 Launch EC2 Instance\*\*

```shellscript

# Create key pair for EC2 access

aws ec2 create-key-pair \

--key-name portfolio-key \

--query 'KeyMaterial' \

--output text > portfolio-key.pem

chmod 400 portfolio-key.pem

# Launch EC2 instance

aws ec2 run-instances \

--image-id ami-0c02fb55956c7d316 \

--count 1 \

--instance-type t2.micro \

--key-name portfolio-key \

--security-groups default \

--user-data file://ec2-user-data.sh

```

\*\*11.2 Create EC2 User Data Script\*\*

```shellscript

cat > ec2-user-data.sh << 'EOF'

#!/bin/bash

yum update -y

yum install -y git

# Install Node.js

curl -fsSL https://rpm.nodesource.com/setup\_18.x | bash -

yum install -y nodejs

# Install PM2 globally

npm install -g pm2

# Create application directory

mkdir -p /opt/portfolio

cd /opt/portfolio

# Clone repository (replace with your repo URL)

git clone https://github.com/yourusername/portfolio-skills-manager.git .

# Install backend dependencies

cd backend

npm install --production

# Create production environment file

cat > .env << 'ENVEOF'

NODE\_ENV=production

PORT=5000

MONGODB\_URI=mongodb+srv://username:password@cluster.mongodb.net/portfolio

FRONTEND\_URL=https://your-cloudfront-domain.com

ENVEOF

# Start application with PM2

pm2 start server.js --name "portfolio-api"

pm2 startup

pm2 save

# Configure nginx as reverse proxy

yum install -y nginx

cat > /etc/nginx/conf.d/portfolio.conf << 'NGINXEOF'

server {

listen 80;

server\_name your-domain.com;

location / {

proxy\_pass http://localhost:5000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_cache\_bypass $http\_upgrade;

}

}

NGINXEOF

systemctl start nginx

systemctl enable nginx

EOF

```

\*\*11.3 Configure Security Group\*\*

```shellscript

# Get the instance ID

INSTANCE\_ID=$(aws ec2 describe-instances \

--filters "Name=key-name,Values=portfolio-key" \

--query 'Reservations[0].Instances[0].InstanceId' \

--output text)

# Get the security group ID

SECURITY\_GROUP\_ID=$(aws ec2 describe-instances \

--instance-ids $INSTANCE\_ID \

--query 'Reservations[0].Instances[0].SecurityGroups[0].GroupId' \

--output text)

# Allow HTTP traffic

aws ec2 authorize-security-group-ingress \

--group-id $SECURITY\_GROUP\_ID \

--protocol tcp \

--port 80 \

--cidr 0.0.0.0/0

# Allow HTTPS traffic

aws ec2 authorize-security-group-ingress \

--group-id $SECURITY\_GROUP\_ID \

--protocol tcp \

--port 443 \

--cidr 0.0.0.0/0

```

#### Step 12: Database Setup (MongoDB Atlas)

\*\*12.1 Create MongoDB Atlas Account\*\*

1. Go to [https://www.mongodb.com/atlas](https://www.mongodb.com/atlas)

2. Sign up for free account

3. Create new project: "Portfolio Skills Manager"

4. Build a cluster (choose free tier)

5. Choose AWS as cloud provider

6. Select region closest to your EC2 instance

\*\*12.2 Configure Database Access\*\*

```shellscript

# In MongoDB Atlas dashboard:

# 1. Go to Database Access

# 2. Add new database user

# 3. Choose password authentication

# 4. Username: portfolio-user

# 5. Generate secure password

# 6. Grant read/write access to any database

# 7. Go to Network Access

# 8. Add IP Address

# 9. Add your EC2 instance IP

# 10. Add 0.0.0.0/0 for development (restrict in production)

```

\*\*12.3 Update Backend Configuration\*\*

```shellscript

# SSH into EC2 instance

ssh -i portfolio-key.pem ec2-user@YOUR-EC2-PUBLIC-IP

# Update environment variables

sudo nano /opt/portfolio/backend/.env

# Update MONGODB\_URI with Atlas connection string

MONGODB\_URI=mongodb+srv://portfolio-user:YOUR-PASSWORD@cluster0.xxxxx.mongodb.net/portfolio?retryWrites=true&w=majority

# Restart the application

pm2 restart portfolio-api

# Check application status

pm2 status

pm2 logs portfolio-api

```

### Phase 6: CI/CD Pipeline Setup

#### Step 13: GitHub Actions Configuration

\*\*13.1 Create GitHub Secrets\*\*

```shellscript

# In your GitHub repository, go to Settings > Secrets and variables > Actions

# Add the following secrets:

# AWS\_ACCESS\_KEY\_ID: Your AWS access key

# AWS\_SECRET\_ACCESS\_KEY: Your AWS secret key

# S3\_BUCKET\_NAME: Your S3 bucket name

# CLOUDFRONT\_DISTRIBUTION\_ID: Your CloudFront distribution ID

# EC2\_HOST: Your EC2 instance public IP

# EC2\_USERNAME: ec2-user

# EC2\_PRIVATE\_KEY: Contents of your portfolio-key.pem file

```

\*\*13.2 Create GitHub Actions Workflow\*\*

```typescriptreact project="CI/CD Pipeline"

...

```

#### Step 14: Testing and Quality Assurance

\*\*14.1 Add Frontend Tests\*\*

```typescriptreact project="Backend API"

...

```

\*\*14.2 Add Backend Tests\*\*

```typescriptreact project="Backend API"

...

```

To configure the generation, complete these steps:

Add REACT\_APP\_API\_URL Add FRONTEND\_URL Run seedDatabase.js Run clearDatabase.js