

SnapEnhance: AI-Powered Image Processing with CI/CD

01.02.2025

—

Team Binary\_Girls

Farhana Akter Suci

Rifah Sajida Deya

Department of Computer Science and Engineering

Jagannath University, Dhaka

## **Project: AI-Powered Image Processing Pipeline with CI/CD**

### Tech Stack

✅ Backend: Flask (Python) or Node.js (Express)  
✅ Frontend: React.js  
✅ Cloud: Railway/Render (backend), Vercel (frontend)  
✅ CI/CD: GitHub Actions  
✅ Docker: For containerization

# Goals

This project aims to combine AI, cloud computing, and automation to create an efficient and scalable image processing pipeline. Here are the main goals:

### 1️⃣ **Cloud-Native & Scalable Architecture** 🌍

✅ Deploy a fully cloud-based solution without relying on local servers.  
✅ Use free-tier services (Render, Railway, Vercel) to ensure accessibility and scalability.  
✅ Containerize the entire project using Docker for consistent deployment across different environments.

### 2️⃣ **AI-Powered Image Processing** 🖼️

✅ Develop an API that applies AI-based image processing (e.g., grayscale, edge detection, background removal).  
✅ Implement real-time processing so users get instant results.  
✅ Keep the backend efficient and optimized to handle multiple users.

### 3️⃣ **Automation with CI/CD (DevOps Implementation)** ⚙️

✅ Set up GitHub Actions for automated testing & deployment.  
✅ Continuous Integration (CI): Ensure all code changes pass tests before deployment.  
✅ Continuous Deployment (CD): Automatically deploy new changes to Render (backend) and Vercel (frontend) without manual intervention.

### 4️⃣ **User-Friendly Interface & Experience** 🎨

✅ Create a simple and intuitive UI (React/Next.js) where users can upload images easily.  
✅ Provide a progress indicator so users know when the image is being processed.  
✅ Allow users to download the processed image once it’s ready.

### 5️⃣ **Security & Reliability** 🔒

✅ Use environment variables (.env) to secure API keys and configurations.  
✅ Ensure the system is error-proof with proper validation and exception handling.  
✅ Implement basic authentication (optional) to prevent spam or abuse.

### 

### **Final Outcome 🎯**

Goal is to create a fully functional, cloud-hosted AI image processing web app that is:  
✔ Automated (CI/CD pipeline)  
✔ Scalable (Deployed on cloud services)  
✔ User-friendly (Easy UI for uploading and downloading images)  
✔ Impressive (Perfect for showcasing DevOps, AI, and cloud skills)

**Plan**

1. Set up the backend API (Image upload & processing)
2. Dockerize the backend (Containerize for easy deployment)
3. Deploy (Backend: Railway)
4. Add database (MongoDB)
5. Re-deploy Backend
6. Build the frontend (User uploads image, sees processed result)
7. Dockerize the frontend
8. Deploy (Frontend: Vercel)
9. CI/CD with GitHub Actions (Automate deployment)

# Tools to Use

## **GitHub Actions** →For CI/CD automation

## **Docker** →For containerization

## **Railway**→ For backend deployment

## **Vercel** → For frontend hosting

## **MongoDB (e.g., MongoDB Atlas)** → For Database

**Implementation:**

Step-1: Set up the backend API (Image upload & processing)

* Create the backend server with Flask(Language: Python). Elask: Python server
* Add the DL model which is previously trained, to generate image to sketch
* Add python code to take an image from user and upload it in a folder of server and let user choose which effect to apply
* Change the image by applying effects
* Save the processed image in a folder.

Step-2: Dockerize the backend (Containerize for easy deployment)

* Build the Docker image [Build Docker Container with Custom Name. code: docker build -t snapenhance]
* Run the Container with a Custom Name [code: docker run --name snapenhance -p 5000:5000 snapenhance]
* Check the running Containers [code: docker ps. Should see snapenhance in the list.]
* Stop & Remove the Container (If Needed) [to stop use: docker stop snapenhance]; [to remove use: docker rm snapenhance]

Step-3: Deploy (Backend: Railway)

* After Dockerization deploy in railway
* First install and login [code: curl -fsSL https://railway.app/install.sh | sh]
* Login using [code: railway login]
* Then go to the backend-directory
* Deploy [code: railway init]
* After deployment, the API for the project is: <https://snapenhance-backend-production.up.railway.app/>
* Use this in browser or test in Postman