Enhance-It: Building the First Mini Project of CREOS

🚀Overview

Enhance-It is the first mini project under the CREOS initiative, aimed at building powerful AI tools from scratch and integrating them with intuitive user interfaces. The journey of Enhance-It began with manually enhancing images using Real-ESRGAN and evolved into a full-fledged web application combining AI-powered backend and a modern frontend using React.

This document walks through each stage of development — from setup to deployment — with an emphasis on problem-solving, learning, and hands-on execution.

Phase 1: Manual Image Enhancement using Real-ESRGAN

What is Real-ESRGAN?

Real-ESRGAN is a deep learning model that enhances low-resolution images by adding fine details and textures, making them look more realistic and high-quality.

Manual Workflow (Command Line)

1. **Clone the Real-ESRGAN repository**
2. **Install dependencies:**

```
pip install -r requirements.txt
```

3. **Place your input image inside the folder.**
4. **Run enhancement via CLI:**

```
python inference_realesrgan.py -n RealESRGAN_x4plus -i inputs --outscale 4
```

5. **Get output in the results folder.**

What happens behind the scenes?

- The model loads the pretrained weights
- It reads the image, applies a 4x super-resolution transformation
- Then saves the enhanced image

Real-life analogy: It's like feeding a sketch into a painter's mind — it knows how a sharp version should look and paints the details back in.

🛕 Phase 2: Flask Backend Integration

Why Flask?

Flask is a lightweight Python web framework that lets us expose Python functionality (like our Real-ESRGAN script) as a web service.

Flask API Flow

1. User uploads an image via a POST request
2. Flask saves it to the `uploads/` folder
3. The script runs Real-ESRGAN enhancement in the backend
4. The enhanced image is saved in `results/`
5. Flask returns the output URL to the frontend

Sample Flask Code

```
@app.route('/', methods=['POST'])
def enhance():
    # Save image, enhance, return URL
```

Problems Faced

- File path errors (Windows-specific)
- Module errors (`flask_cors`, `uuid`, etc.)
- Running model from within Flask process

Fixes

- Installed missing dependencies
- Handled file naming conflicts
- Debugged and wrapped enhancement in try/except blocks

🚲Phase 3: React Frontend + Flask Backend

Why React?

We wanted a clean, responsive and modern UI with dynamic rendering — React was perfect.

React Frontend

- Upload form using drag-and-drop UI
- Loader while image is enhancing
- Result preview and download button

Frontend Flow

1. User drops image or clicks to upload
2. Image is previewed
3. On clicking "Enhance", React sends image to Flask (`POST /`)
4. Waits for response and shows enhanced image

Backend Flow

1. Receives image
2. Enhances using Real-ESRGAN
3. Sends back result path

Issues We Faced

- Cross-Origin Request Blocking (CORS)
- React not able to parse response
- Real-ESRGAN execution delay

Solutions

- Added CORS using `flask_cors`
- Used `fetch` with proper headers
- Provided loader and error messages on UI

Deployment Planning

We plan to use:

- **Frontend:** Vercel
- **Backend:** Railway, Render or local server

Technologies Used (with Examples)

Real-ESRGAN

- Deep Learning model that acts like an artist fixing a blurred photo.

🛕 Flask

- Like a waiter taking your order, giving it to the chef (Real-ESRGAN), and serving you the enhanced image.

React (Vite + TypeScript)

- Fast UI rendering
- Buttons, image preview, animation, interaction

■ Summary: What We Learned

• How AI models can be executed from terminal and programmatically
• How to build backend services with Flask
• How frontend communicates with backend using REST APIs
• How to style and structure a production-grade React app
• Most importantly, how to troubleshoot errors with ChatGPT and Bolt

■ Credits:
This documentation was created by Rishi Mailoorkar under the AI Research Lab at QD&Co.
Anyone who uses or references this material must credit QD&Co. for the research and development effor