Zat Pros Gaznares 239978
Pablo Rodríguez Solans 240027

Universitat Pompeu Fabra Barcelona

# P2-Transcoding Report

## Introduction

During this lab, AI significantly improved the code's structure and scalability. Although there are not many git diff entries for this lab due to us collectively implementing changes without committing regularly, the improvements we implemented deserve detailed attention.

## Exercise 1

Originally, we manually crafted FFmpeg commands for each codec (sources are cited in the code) and named output files manually. AI suggested dynamically constructing filenames to include the codec and original video name (e.g., 'video_vp8.webm'), reducing ambiguity and preventing overwrites. It also recommended explicit error handling for unsupported codecs, enhancing robustness.

Key improvements:

- Dynamic File Naming: Filenames now reflect codec types.
- Centralized Command Handling: Simplified and improved maintainability.
- Enhanced Error Handling: Clear HTTP responses for invalid inputs or process failures.

## Exercise 2

The AI-driven approach replaced our hardcoded, static process with dynamic loops, enabling scalability.

- Resolution Loop: All resolutions are processed iteratively, supporting varied inputs.
- Codec Conversion Loop: Nested loops now handle multiple codecs and resolutions seamlessly.
- Error Isolation: Exceptions are traced to specific steps, ensuring fault tolerance.
- Unified Output: Results are compiled into a clear, structured summary.

AI transformed our approach, making it scalable, adaptable, and easier to maintain.

## Exercise 3

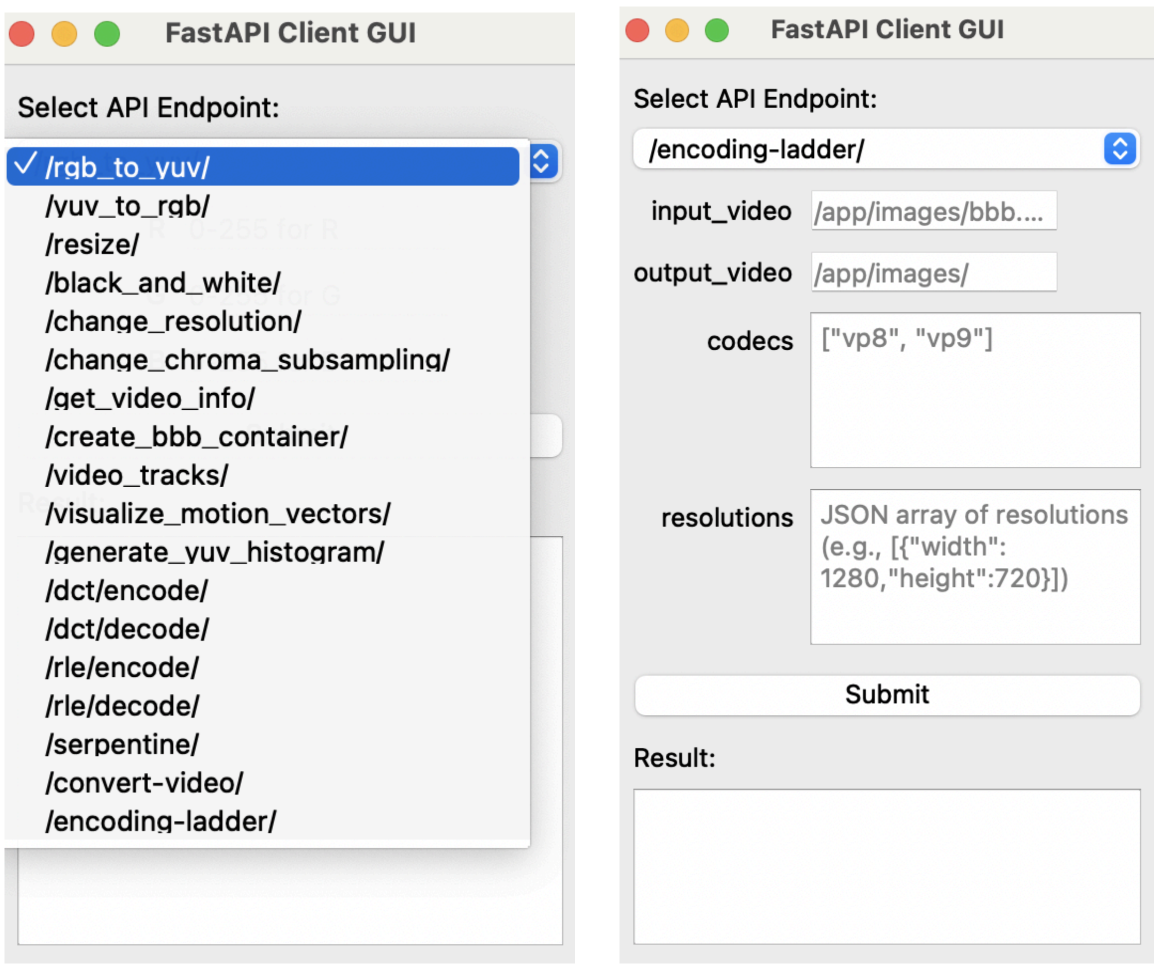
For Exercise 3, we designed a GUI to interface with our Monster API, using PyQt5 as a challenge and an opportunity to explore GUI development for the first time. Despite our lack of prior experience, we managed to create a robust, user-friendly interface that allowed seamless interaction with the API.

Our GUI included a dropdown menu listing API endpoints (see Figure 1), dynamically generated input fields tailored to each endpoint's parameters, and a "Submit" button to send requests and display responses. While AI suggested automating input paths, this approach proved unsuitable for our API. Instead, we opted to hardcode paths and placeholders to ensure clarity and accuracy. For example, selecting `/change_resolution/` provided clearly labeled fields for file paths, width, and height. This deliberate design choice helped minimize user errors and improved overall usability.

Initially, we explored AI-generated code for endpoint testing, but it was unreliable within our Dockerized environment. Instead, we tested each endpoint manually, refining the GUI iteratively.

This hands-on approach proved highly effective, as user testing showed significant improvement—from an initial usability score of 6 to 8 after adding clear reference inputs.

Though time constraints limited further development, we are proud of the GUI's functionality and ease of use. This exercise not only enhanced our understanding of GUI development but also deepened our knowledge of video encoding, Dockers and API design.



/encoding-ladder/

input_video /app/images/bbb....

output_video /app/images/

codecs ["vp8", "vp9"]

resolutions JSON array of resolutions (e.g., [{"width": 1280,"height":720}])

**Figure 1:** PyQt5 Final GUI