Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Capstone Project (01CT0715)	**Implementation- Continuous progress review**	
Implementation	**Date: 25/09/2025**	**Enrolment No: 92200133018**

MRI Impression Generation

Introduction

The project focuses on developing an assistive tool that supports doctors as well as radiologists in generating consistent and time-efficient MRI impressions. The system is not meant to replace medical expertise but to act as a supportive layer that reduces repetitive work and ensures accuracy.

Implementation

The project was built with a focus on **clean, easy-to-understand code**, a **modular structure**, and **smooth performance** so that it fully supports the goals of the MRI impression assistant.

1. Code Quality

The code is written in a **clear and consistent style**. Functions have meaningful names, comments are added wherever logic might be complex, and sensitive keys are kept safe using environment variables instead of being hardcoded. A requirements.txt file is included so that the setup can be easily reproduced, and the project is version-controlled using Git.

app5.py > ...

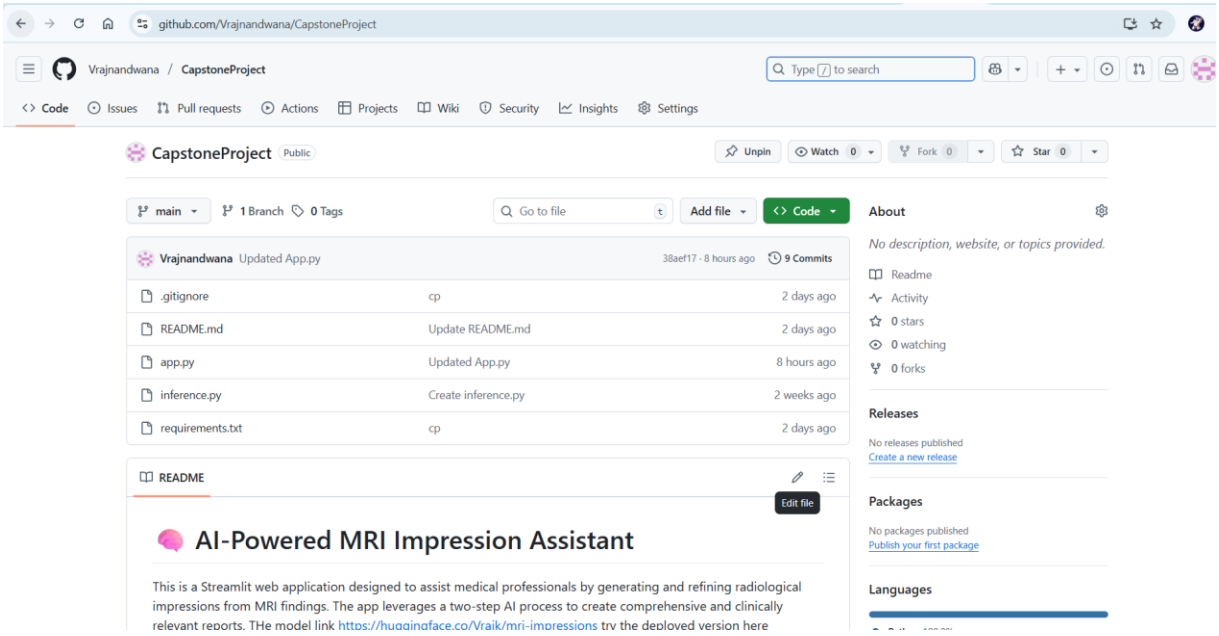
```
83   def enhance_with_gpt(raw_impression: str, original_findings: str):
107
108      **FINAL IMPRESSION:**
109      """
110      response = client.chat.completions.create(
111          model=AZURE_OPENAI_DEPLOYMENT_NAME,
112          messages=[{"role": "user", "content": prompt}],
113          temperature=0.2,
114          max_tokens=300
115      )
116      return response.choices[0].message.content.strip()
117
118  def extract_findings_from_pdf(pdf_file):
119      with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as tmp_file:
120          tmp_file.write(pdf_file.getvalue())
121          tmp_path = tmp_file.name
122      elements = partition_pdf(filename=tmp_path)
123      os.remove(tmp_path)
124      full_text = "\n".join([el.text for el in elements if el.text])
125      impression_keywords = ["Impression", "IMPRESSION"]
126      findings_start_patterns = ["SEQUENCES:", "HISTORY:", "FINDINGS:"]
127      start_idx = 0
128      for pat in findings_start_patterns:
129          idx = full_text.find(pat)
130          if idx != -1:
131              start_idx = idx + len(pat)
132              break
133      end_idx = len(full_text)
134      for kw in impression_keywords:
135          idx = full_text.find(kw)
136          if idx != -1:
137              end_idx = min(end_idx, idx)
138      findings_text = full_text[start_idx:end_idx].strip()
139      return re.sub(r"(\n\s*)+\n+", "\n", findings_text).strip()
```

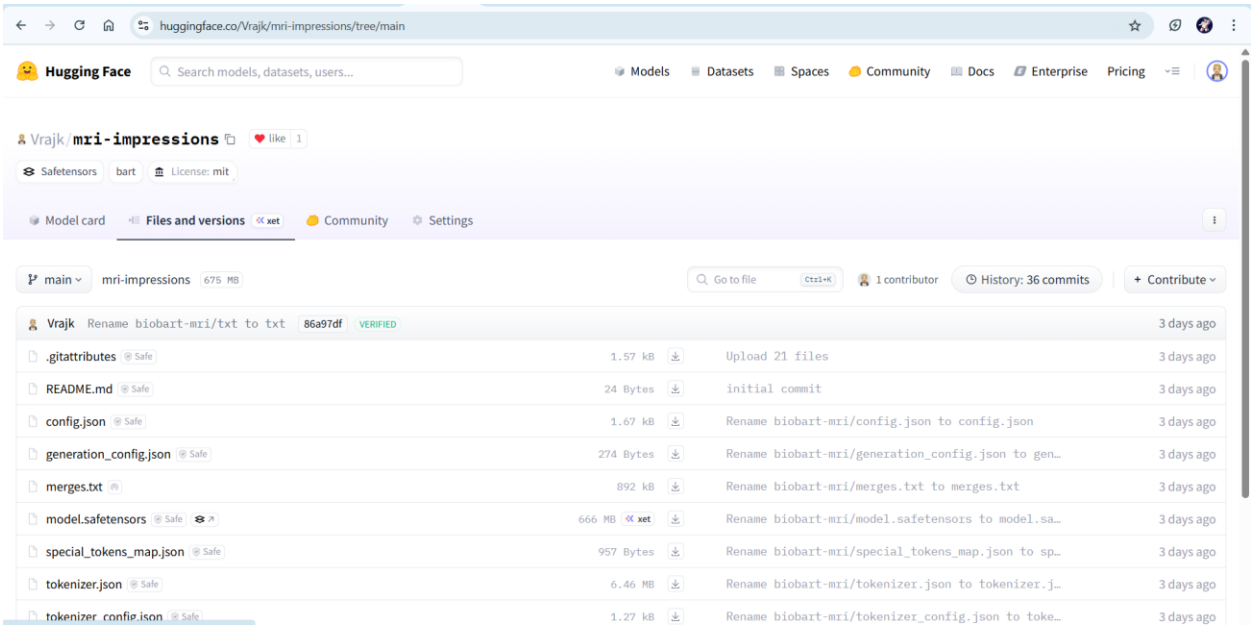
Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Capstone Project (01CT0715)	**Implementation- Continuous progress review**	
Implementation	**Date: 25/09/2025**	**Enrolment No: 92200133018**

Sample output :

Ceck on gitub link: https://github.com/Vrajnandwana/CapstoneProject/blob/main/app.py

https://huggingface.co/spaces/Vrajk

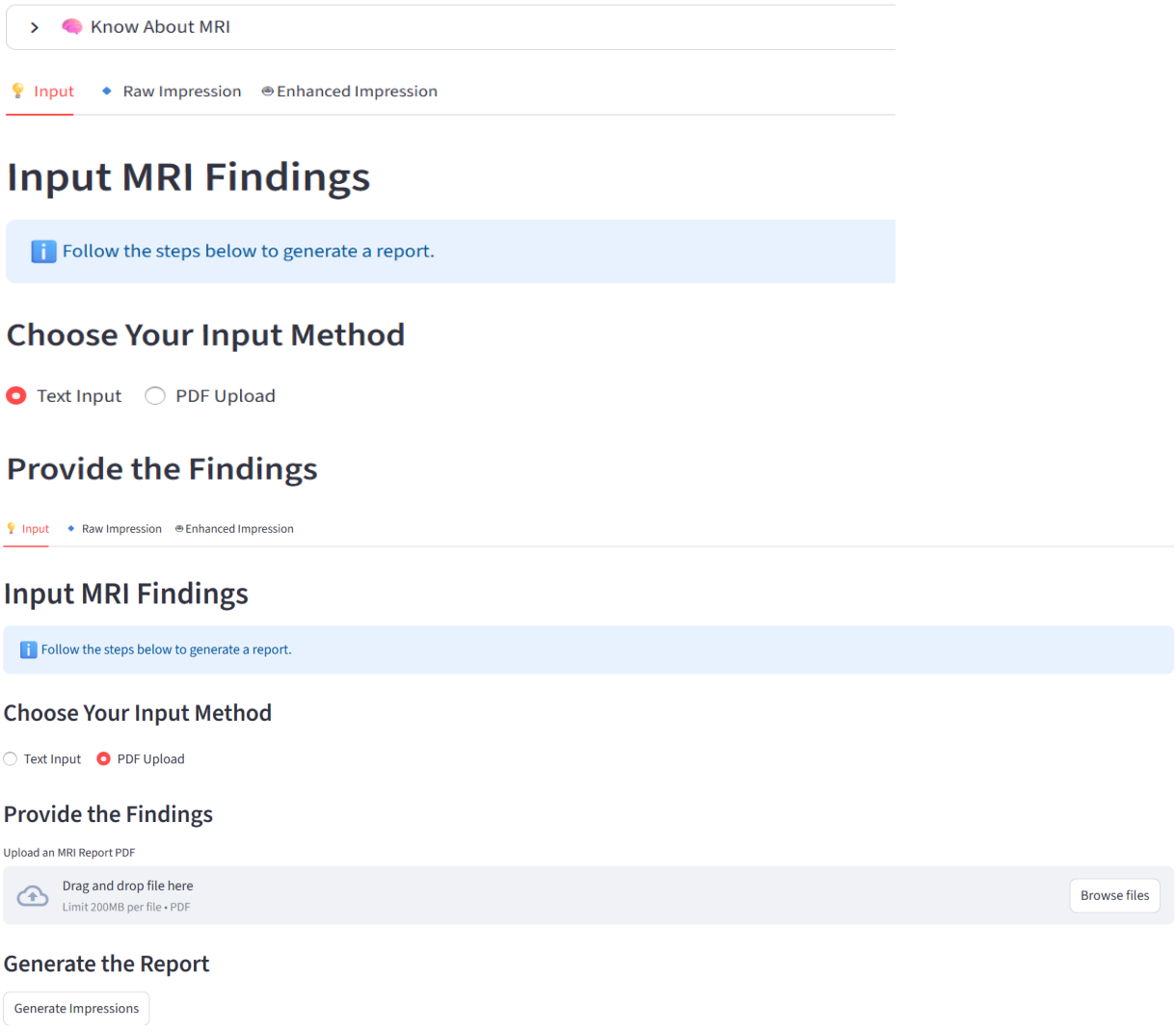




2. Modular Design

The system is broken down into **separate modules**, each handling a specific part of the workflow:

- **Input Handling** – allows users to provide MRI findings either by typing text or uploading PDFs which are related to MRI only as it will not take the reports that are non mri , any other domain reportor any person who is just taking .



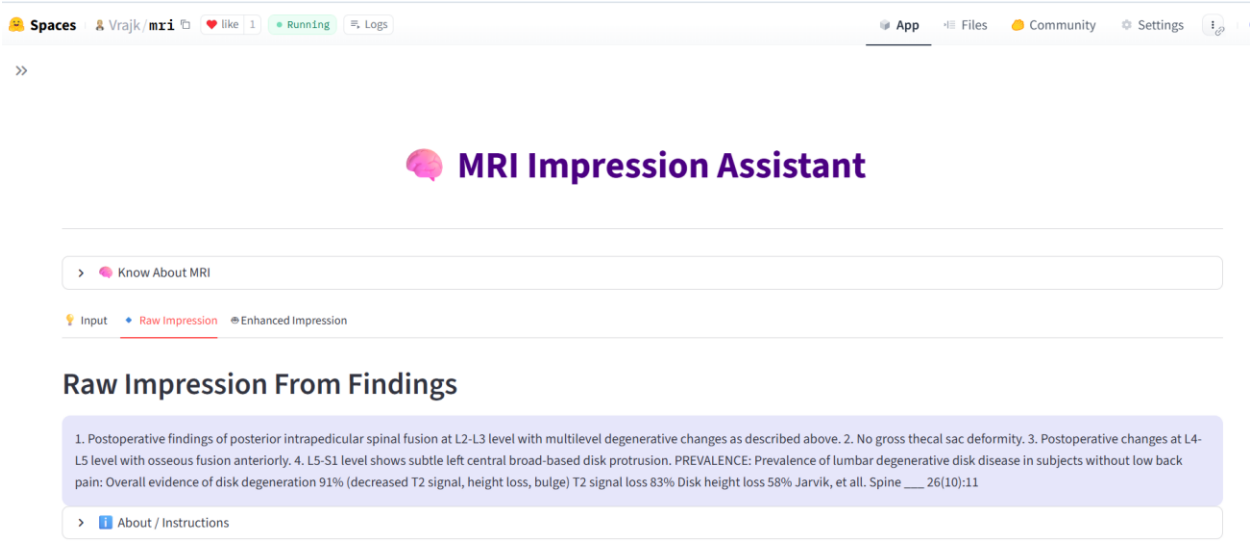
Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Capstone Project (01CT0715)	**Implementation- Continuous progress review**	
Implementation	**Date: 25/09/2025**	**Enrolment No: 92200133018**

- **Model Inference** – uses the fine-tuned BioBART model to generate a first draft impression. The BioBART model is based on the **encoder–decoder (sequence-to-sequence)** architecture. It uses a Transformer backbone where:

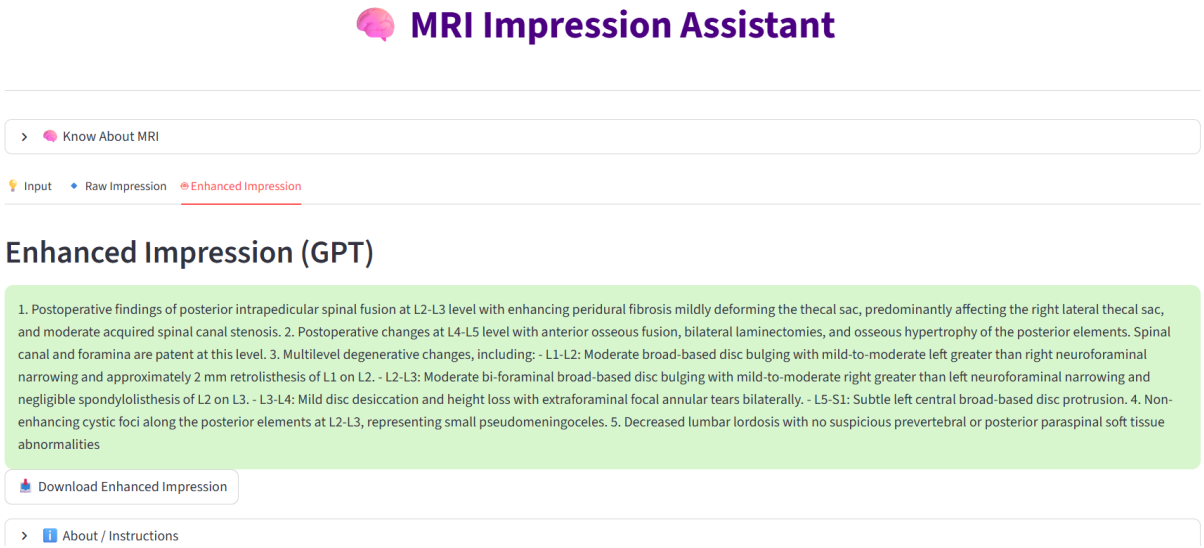
  The **encoder** reads the detailed MRI findings and captures the important clinical context.

  The **decoder** then generates a concise impression, step by step, based on that context.

  Since it is pre-trained on large biomedical text and later fine-tuned on **MRI-specific radiology reports**, the model learns both medical terminology and the style of radiology summaries. This makes it well-suited for transforming long, complex findings into short, meaningful impressions.



- **Ai Refinement** – improves the draft by cross-checking with the findings and producing a polished impression.



- **User Interface** – built with Streamlit, giving a simple and interactive experience.

This separation makes the code easier to maintain, extend, and debug.

3. Meeting Requirements

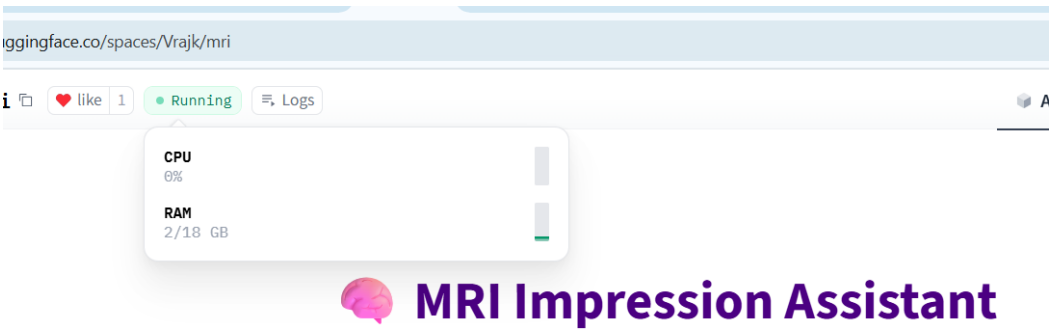
All the planned features are implemented:

- Users can input findings through text or PDFs.
- The system generates both a raw impression and an enhanced final version.
- The enhanced version can be downloaded as a text file.
- Settings like beam size and impression length can be customized from the sidebar.

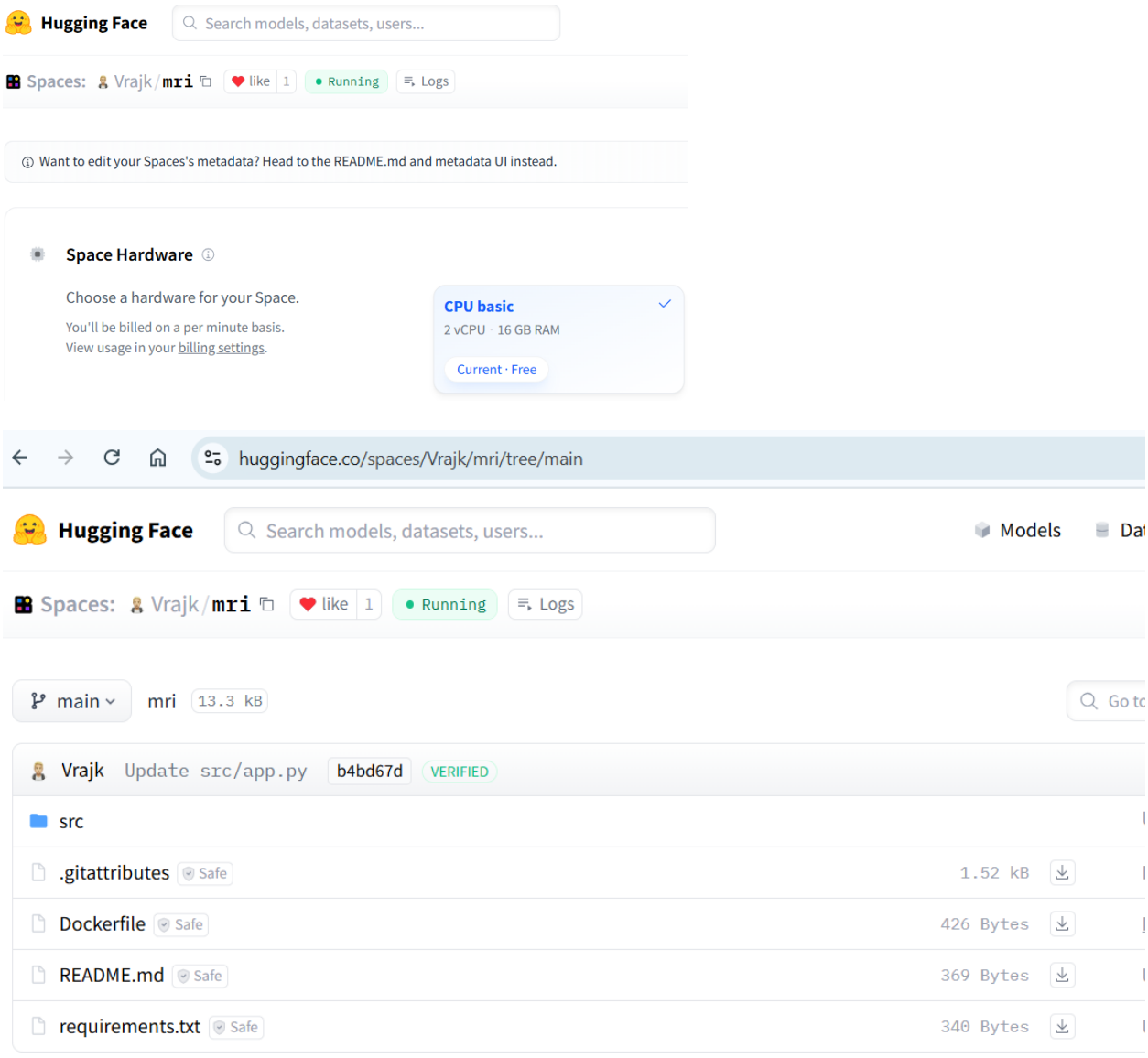
Every requirement is fully covered, and the workflow is easy to follow for the end user.

4. Performance and Reliability

The system runs smoothly even on CPU. To avoid delays, the model is cached after the first load so it doesn't reload repeatedly. Input validation checks help ensure that only proper MRI findings are processed. Errors like missing files, invalid text, or API issues are handled gracefully with clear messages.



Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Capstone Project (01CT0715)	**Implementation- Continuous progress review**	
Implementation	**Date: 25/09/2025**	**Enrolment No: 92200133018**



5. Integration of Components

Each part of the pipeline connects seamlessly to the next:

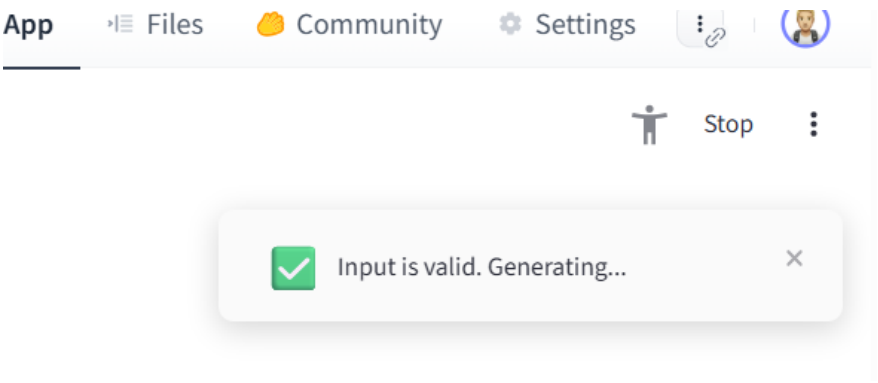
1. Findings are given as input.
2. The text is validated.

3. Our fine ttuned trained BioBART model generates the raw draft.
4. GPT refines it into a professional impression.
5. The result is displayed and can be downloaded.

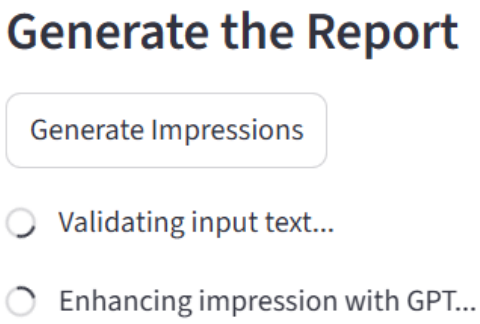
The Streamlit interface keeps users informed through status messages, spinners, and notifications at each step.

6. Testing

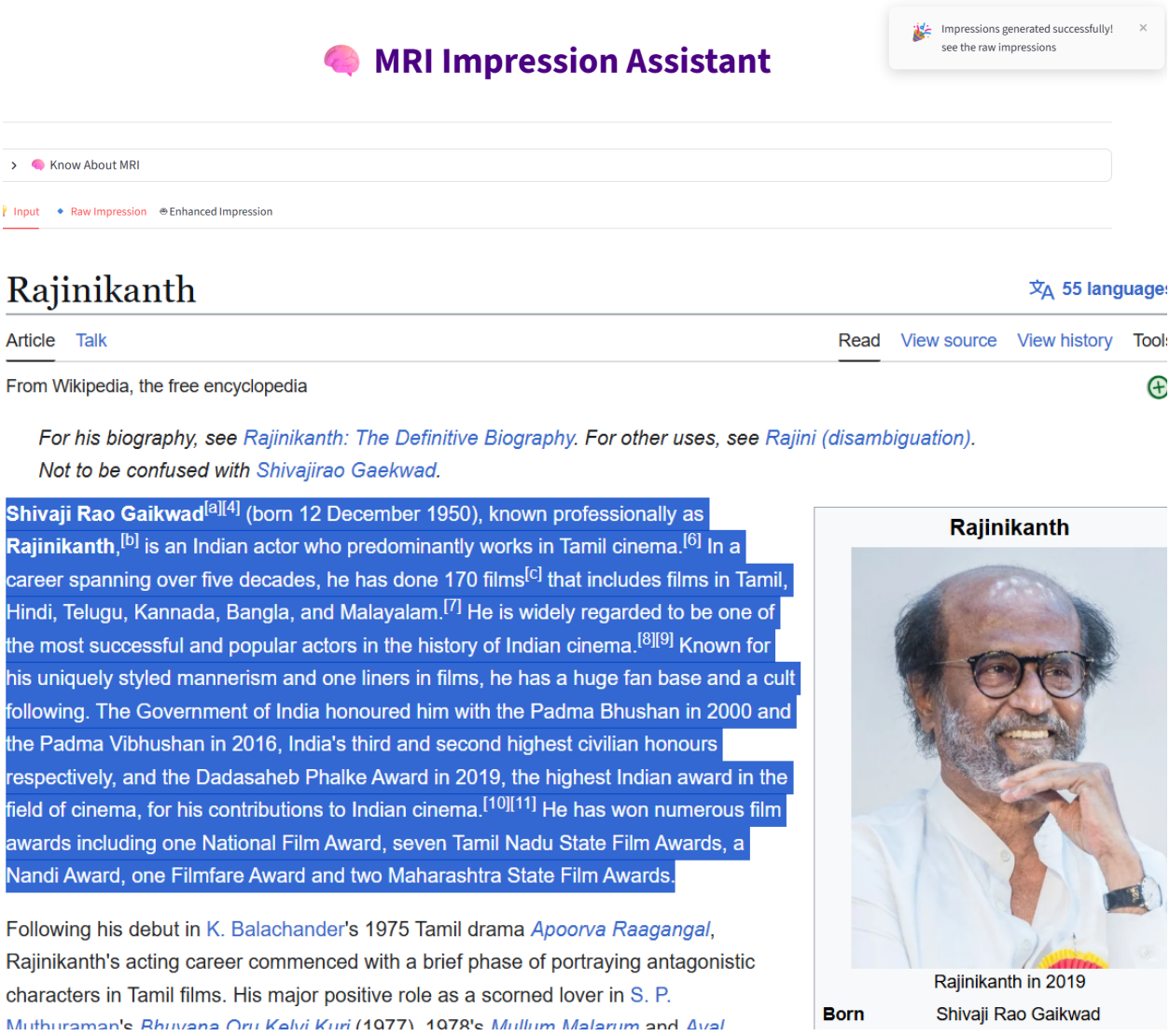
Different parts of the system were tested separately (PDF extraction, input validation, impression generation). End-to-end tests were done by giving actual MRI text and checking the generated impressions. Edge cases like empty inputs or extra-long text were also tested to make sure the tool responds properly.



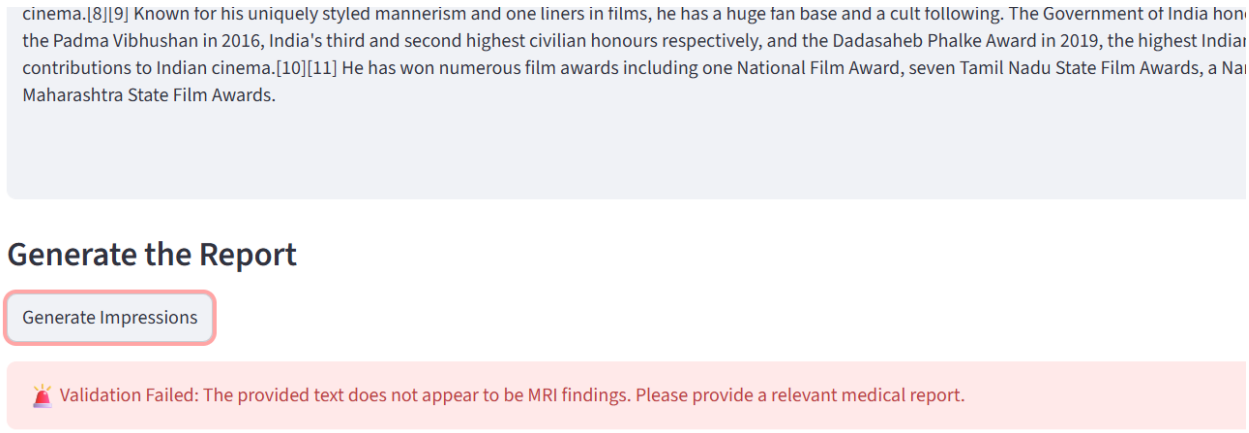
It only generates if input is valid



Then only you see a success message which means that a valid mri report is detected and the impressions aare generated first in our model than with help of gpt its generated the missed impression.



So this is a not valid as it says abour Rajnikant out of context so the validation will invalidate it.



Code Structure & Organization

The system is organized in a clean, modular way to make it easy to maintain, extend, and understand:

- **app.py** – Main Streamlit application that connects all components. Handles UI, user input, output display, and integration of modules.

- **Environment Variables** – All sensitive keys (like Azure OpenAI API) are stored in a .env file to keep the system secure.
- **Model Loading & Caching** – The BioBART model is loaded and cached to prevent reloading for every request, improving performance.
- **Input Handling Module** – Handles text input and PDF extraction, validating that only MRI findings are processed.
- **Model Inference Module** – Uses the fine-tuned BioBART model to generate a raw draft impression.
- **AI Refinement Module** – Uses GPT via Azure OpenAI to enhance and polish the impression for clinical accuracy.
- **User Interface Module** – Streamlit UI handles tabs, sidebar settings (beam size, min/max length), status messages, and download functionality.

Instructions for Running the System

To set up and run the MRI Impression Assistant:

1. **Clone the Repository**

git clone https://github.com/Vrajnandwana/CapstoneProject.git

cd CapstoneProject

2. **Create a Virtual Environment**

python -m venv env

Activate environment

Windows

.\env\Scripts\activate

macOS/Linux

source env/bin/activate

3. **Install Dependencies**

pip install -r requirements.txt

4. **Add Environment Variables**
   Create a .env file in the root folder with your Azure OpenAI credentials:

AZURE_OPENAI_API_KEY="YOUR_KEY"

AZURE_OPENAI_ENDPOINT="YOUR_ENDPOINT"

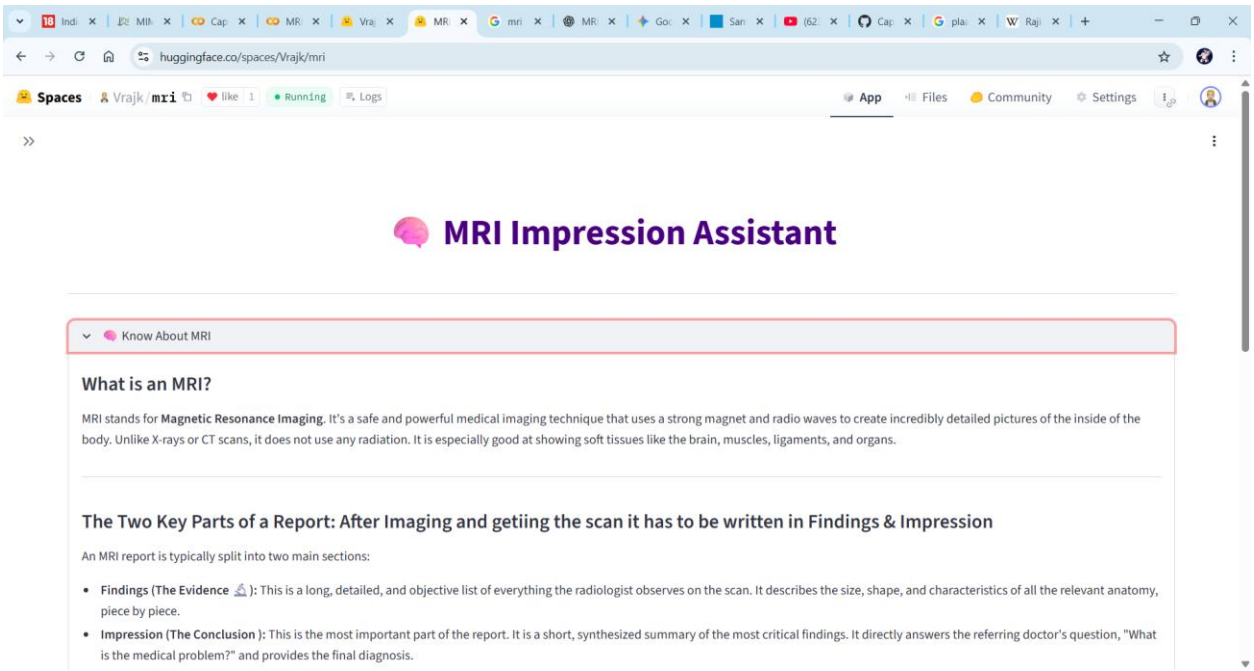
AZURE_OPENAI_API_VERSION="2023-03-15-preview"

AZURE_OPENAI_DEPLOYMENT_NAME="YOUR_DEPLOYMENT"

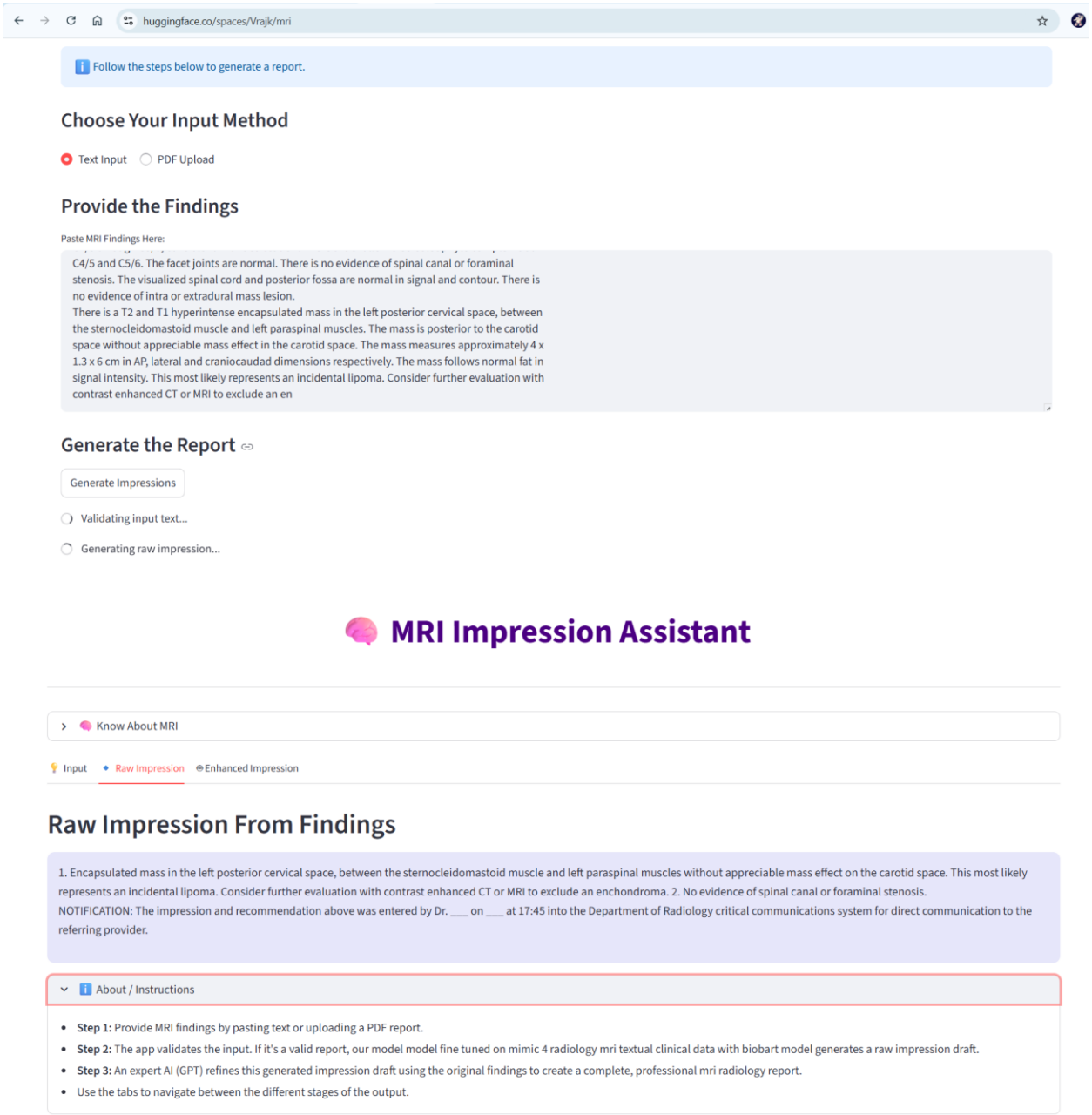
5. **Run the App**

streamlit run app.py

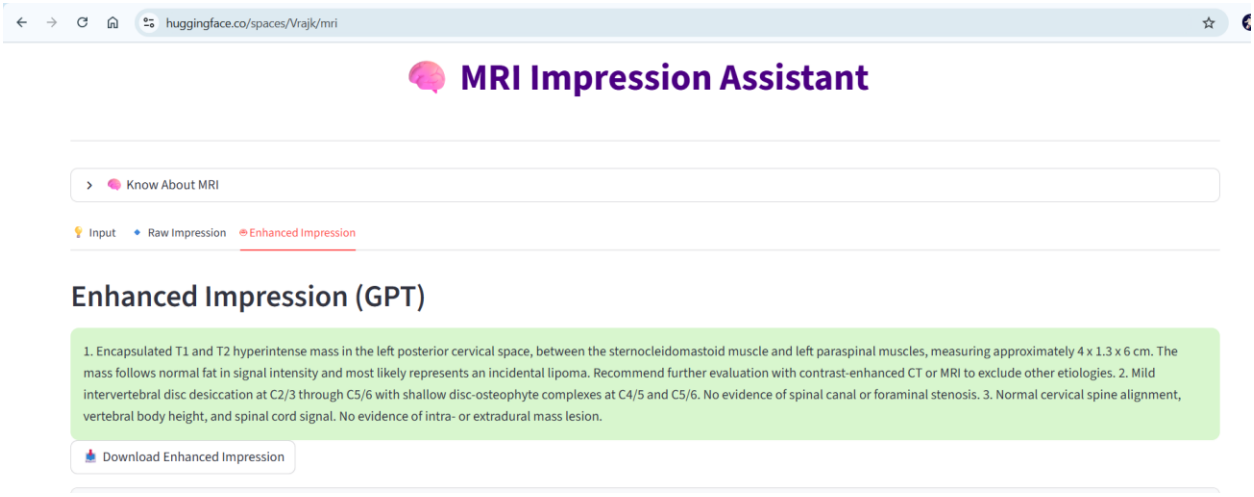
Screenshorts

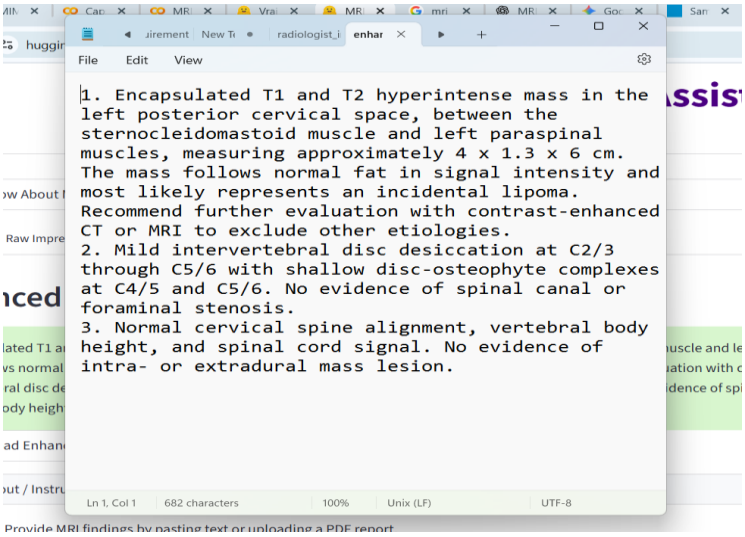


Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Capstone Project (01CT0715)	**Implementation- Continuous progress review**	
Implementation	**Date: 25/09/2025**	**Enrolment No: 92200133018**



Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Capstone Project (01CT0715)	**Implementation- Continuous progress review**	
Implementation	**Date: 25/09/2025**	**Enrolment No: 92200133018**





The MRI Impression Assistant is successfully deployed and publicly accessible.

- Live URL: https://huggingface.co/spaces/Vrajk/mri
- Gitub : https://github.com/Vrajnandwana/CapstoneProject