Assignment: Deploy a FastAPI-based ASR Application Using NVIDIA NeMo

Objective:

Build and containerize a **FastAPI-based Python application** that serves an **ASR (Automatic Speech Recognition) model** built using **NVIDIA NeMo**. The model should be optimized for inference using either **ONNX** or **TorchScript**, and must be able to transcribe audio clips of **5–10 seconds**. Note that we will consider even partial submissions but with the caveat!

Requirements:

✅ 1. Model Preparation

- Use the ASR model https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_hi_conformer_ctc_medium/files from **NVIDIA NeMo**.
- Optimize the model for inference using **ONNX**.
- Ensure the model can handle single audio files of **5–10 seconds**, sampled at 16kHz.

✅ 2. FastAPI Application

- Create a FastAPI server with an endpoint:
  `POST /transcribe`
  that accepts an audio file (`.wav`) and returns the transcribed text as JSON.
- Include basic input validation (e.g., file type, duration).
- Use an async-compatible model inference pipeline (or document why it isn't).

✅ 3. Containerization

- Create a **Dockerfile** to containerize the application.
- Ensure the container installs all required dependencies and starts the FastAPI server on port `8000`.
- Use best practices for lightweight image size (e.g., using a Python slim base).

✅ 4. Documentation

- Provide a short **README.md** with:
  - Instructions to build and run the container.
  - Sample `curl` or `Postman` request to test the `/transcribe` endpoint.
  - Any design considerations you have taken into account.

✅ 5. Communication skills

- Provide a short **Description.md** with:
  - List of features you were successfully able to implement.
  - Issues you ran into during your development.
  - Why you may have not been able to implement a particular component? What was the limitation?
  - How will you overcome the challenges mentioned above?
  - Any known limitations or assumptions of your deployment.

Deliverables:

Please submit a link to a GitHub/GitLab repo that includes:

- Source code for the FastAPI app
- Dockerfile
- Preprocessing/inference code with model loading logic
- Code to create the optimized ONNX model
- README.md with setup and usage instructions
- Description.md with a good documentation of what you have done
- Optional: A test audio clip and sample response

Evaluation Criteria:

Area	Points
Code structure & readability	10
Communication about issues and limitations	20
FastAPI implementation	20
Model optimization & usage	20
Containerization (Docker)	10
Documentation & UX clarity	10
Bonus: Async inference, CI/CD, or testing	10

Time Estimate:

Submit by **26/05/2025, 1:00 p.m.** using google form.