***Natural Language Processing***

*members: Haroon Khan 22I-0516*

*Usaid Azeem 22I-0515*

*Jawad Khan 22I-0507*

**semester project**

**problem statement:**

*To produce a containerized microservice that runs locally and performs story-to-audio using any available tts model.*

*The microservice is expected to:*

* *Have gRPC API with at least 1 core endpoint (e.g., /generate, /infer)*
* *Handle multiple requests concurrently (basic async/concurrency)*
* *Support minimal error handling (invalid input, server errors)*
* *Return JSON responses with status codes*
* *Should be thoroughly tested (test cases to be submitted)*
* *Should be demo-able through Postman + a minimal frontend using Gradio or StreamLit.*

*additionally, the service should handle input as a storyline, and produce a realistic sounding narration at its end-point.*

**methodology:**

*model used:*

*The model used is a 1.6B parameter heavy tts inference engine called Dia that was released within a few days of the assigned project. Due to its recent release, the model architecture has not been completely disclosed. However, due to its highly expressive audio output and high performance (compared to the industry leading eleven labs) it is highly likely that it is based on a transformer architecture.*

*Dia was created by Nari labs; a Korean laboratory that specializes in transformer and deep ANN models. The model weights and safetensors are open source and available for commercial and experimental use under the apache 2.0 license.   
  
The model also allows for voice tuning and produces much more realistic and natural sounding dialogue (with multiple speaker support) including mimicking coughs, laughter, screams etc.*

*microservice:*

*The service is based on the following production flow:*

*[TTS\_model + gRPC\_server (SERVER) -> gRPC(client) + Gradio (FRONT-END)] docker\_cont*

1. *server side:*
   1. *dia-tts base model loaded on the server side*
   2. *gRPC server exposes port 50051 to connect to clients*
2. *client side:*
   1. *gRPC client connect to 50051 to communicate with server*
   2. *Gradio opens front-end for UI on port 7860*
3. *docker:*
   1. *docker wraps everything in a container for easy redeployment on any*

*linux system with gpu/cuda enabled*

*problems:*

1. *The model is compute intensive; around 10 gigs of VRAM required for long form generation*
2. *On low-end GPUs, the model hallucinates during long form audio generation; for short form, it may not produce as high quality output as is expected. So, ideally you have to find a ‘sweet spot’ of num\_tokens for a good balance-performance ratio.*