# Model Selection

- Model selection was a major challenge as not many models were available which could generate good quality code so I started researching **variants of Llama** and got to know about **CodeLlama 7B, 13B and 34B**
- Keeping in mind the **15 GB limit of GPU RAM** available to me in google colab I took CodeLlama 7B but soon realized that normal model was leading to **session crash** due to exceeding **15GB limit of RAM** while finetuning
- So I came accross **sharded models** which divide the model files into smaller sized files which help in **parallel processing** and **less consumption of RAM**
- **TinyPixel/CodeLlama-7B-Instruct-bf16-sharded** was a shared version of CodeLlama tuned to understanding instructions for code generation

# Quantized model, input and problems in Fine-Tuning

- I used the **bitsandbytes library** which provides the 4 bit quantized version(**weights are rounded off basically**) of the sharded model which takes only **¼ th of the space** in the memory without having much effect on accuracy of the model
- Defining the **tokenize_function** was a big challenge and after printing the tokenized output again and again I figured out the way to make the function such that **tokens of the label** act as input and **html code act as output**
- Only the tensors would be a part of the input to the model and not the text columns whose tokenization has taken place so they are removed
- Even after quantization the GPU exceeded 15 GB GPU RAM in fientuning and hence I researched more and got to know about **PEFT(Parameter Efficient Fine Tuning)** technique which uses **gradient checkpointing** to save memory usage
- **LORA(Low rank adaptation)** enhances model ability to fine tune efficently and **accelerate library of transformers with FSDP** shards model parameters across **multiple GPU's**, although I had only 1 GPU, but it would be useful if someone has multiple GPU's
- The references and the predictions were 2-D list so had to debug and convert them to 1-D list to compare and calculate accuracy

# Saving the finetuned model in github and Inferencing

- Tried using **git lfs** to store the **adapter_model.safetensors** file as pointer due to its big size but its limit also exceeded which came with free github, hence uploaded model on Hugging Face Hub
- During the inference part loading the model directly showed error so used the adapter approach with the base model as the original model
- Also there were some problems and import errors due to older versions of some libraries and thus lack of some advanced features in them, so uninstalled the older versions and installed new ones

# Making the API

- I tried using Flask at first but it showed the error of **HOST REFUSED TO CONNECT**
- Tried chainlit which provides the frontend and found that it can be connected with the backend using **ngrok** tunnels which can be created using API key generated bt creating a new account