Supplementary Document

Choice of LLM :

- I use the “**deepseek-ai/deepseek-coder-1.3b-instruct**” code gen model for demo in this assignment.
    - Free to use
    - Accurate Outputs.
    - Pre-trained on 1.3 billion tokens.
    - Light and fast for google colab.
    - Multiple coding language supported
- Right now the best model for Code generation is OpenAI’s **Codex.**
- Codex has all the functionalities, it is GPT 3.5 powered.
- Github Co-pilot uses Codex model for generation
- If money is not a concern we should definitely use Codex from OpenAI for our tool.It is not open source and OpenAI charges for it.

Free Alternatives :

- Models from HuggingFace Hub.
- Codegen by Salesforces is a Open source model that is a good alternative for Codex.

Prompt Engineering :

Prompt 1:

- Write a **Python function** (context for context processor) named calculate_average that takes a list of numbers as input, calculates the mean, and returns a string formatted as "The average is: {average}".

Prompt 2:

- Implement a **JavaScript function** to sort an array of strings alphabetically in descending order, ignoring case sensitivity.

User Input Prompts:

Prompt 3:

- Create a button in **React** that displays the text "Submit" and triggers a function named Submission when clicked.

Prompt 4:

- Design a text input field in **HTML** that accepts only numbers, has a maximum length of 10 characters, and displays a validation message if invalid input is entered.

Prompt 5:

- Write a C++ class named ImageFilter with a method applyFilter that takes an image as input and applies a grayscale filter, returning the modified image.

Prompt 6:

- Implement a Python function that simulates a simple game of dice roll, where two dice are rolled, and the sum is displayed.

Be as Specific to the model as you can , don't try to make prompts shorter instead make it longer and elaborate for nuances and details for better output.

Error Handling :

Validation: Check for syntax errors, type mismatches, and logical inconsistencies in the generated code.

User Feedback Integration: Allow users to report incorrect or inefficient code, incorporating their feedback into future prompts.

Multiple Code Generation: Generate several solutions and let users select the most efficient or maintainable one.

Limitations:

- Model Performance.
- Model Selection - Availability of many models and everybody has some unique feature makes it hard to select the most efficient model.
- Data set for Fine tuning - the input format for Fine tuning the model may vary model to model.
- Limited Dataset: Limited languages support as per the dataset.
- LLM Imperfections: No LLM is perfect. Acknowledge limitations in accuracy, efficiency, and understanding of complex problems.