LLM BENCHMARKING REPORT

Project Structure:-

```
benchmarking/
│
├── app.py
├── requirements.txt
```

Tech Stack

- **Streamlit**: For building an interactive, lightweight web interface without requiring heavy frontend frameworks.

- **Python 3.x**: Core programming language.

- **Concurrent Futures (ThreadPoolExecutor)**: To handle **parallel execution of multiple LLM requests** simultaneously.

API & Backend

- **OpenRouter API**: Unified gateway to multiple LLMs (OpenAI, Gemini, Grok, DeepSeek, Claude).

- **HTTP (requests)**: For making API calls.
- **Local Development**: Via streamlit run app.py

By-
Khushi Chadha

HOW IT WORKS :-

1ST

MODEL SELECTION

User toggles the models (OpenAI, Gemini, Grok, DeepSeek, Claude) at the top.

2ND

PROMPT INPUT

User enters a prompt.

3RD

PARALLEL EXECUTION

Each selected LLM processes the prompt simultaneously.

4TH

SIDE-BY-SIDE DISPLAY

Responses are shown column-wise for easy comparison.

By-
Khushi Chadha

EVALUATION OUTLINE

1 Efficiency – Speed and ability to generate a complete response.

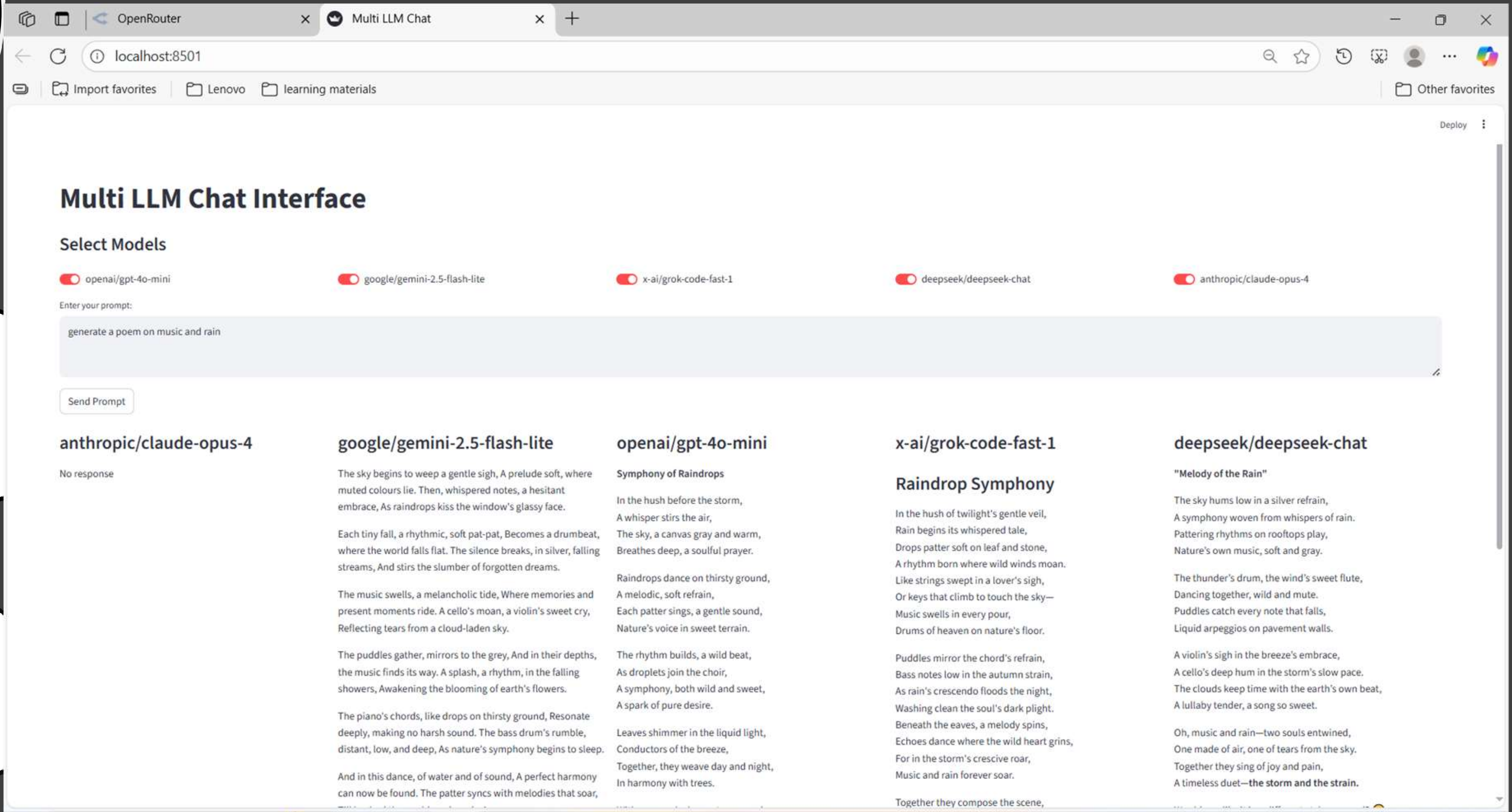
2 Response Quality – Clarity, coherence, and structure of the output.

3 Relevance – How closely the response matches the prompt intent.

By-
Khushi Chadha

TASK 1:-
Poem Generation , checking Quality of response

INSIGHTS :-



Multi LLM Chat Interface

Select Models

● openai/gpt-4o-mini ● google/gemini-2.5-flash-lite ● x-ai/grok-code-fast-1 ● deepseek/deepseek-chat ● anthropic/claude-opus-4

Enter your prompt:

generate a poem on music and rain

Send Prompt

anthropic/claude-opus-4

No response

google/gemini-2.5-flash-lite

The sky begins to weep a gentle sigh, A prelude soft, where muted colours lie. Then, whispered notes, a hesitant embrace, As raindrops kiss the window's glassy face.

Each tiny fall, a rhythmic, soft pat-pat, Becomes a drumbeat, where the world falls flat. The silence breaks, in silver, falling streams, And stirs the slumber of forgotten dreams.

The music swells, a melancholic tide, Where memories and present moments ride. A cello's moan, a violin's sweet cry, Reflecting tears from a cloud-laden sky.

The puddles gather, mirrors to the grey, And in their depths, the music finds its way. A splash, a rhythm, in the falling showers, Awakening the blooming of earth's flowers.

The piano's chords, like drops on thirsty ground, Resonate deeply, making no harsh sound. The bass drum's rumble, distant, low, and deep, As nature's symphony begins to sleep.

And in this dance, of water and of sound, A perfect harmony can now be found. The patter syncs with melodies that soar,

openai/gpt-4o-mini

Symphony of Raindrops

In the hush before the storm,
A whisper stirs the air,
The sky, a canvas gray and warm,
Breathes deep, a soulful prayer.

Raindrops dance on thirsty ground,
A melodic, soft refrain,
Each patter sings, a gentle sound,
Nature's voice in sweet terrain.

The rhythm builds, a wild beat,
As droplets join the choir,
A symphony, both wild and sweet,
A spark of pure desire.

Leaves shimmer in the liquid light,
Conductors of the breeze,
Together, they weave day and night,
In harmony with trees.

x-ai/grok-code-fast-1

Raindrop Symphony

In the hush of twilight's gentle veil,
Rain begins its whispered tale,
Drops patter soft on leaf and stone,
A rhythm born where wild winds moan.
Like strings swept in a lover's sigh,
Or keys that climb to touch the sky—
Music swells in every pour,
Drums of heaven on nature's floor.

Puddles mirror the chord's refrain,
Bass notes low in the autumn strain,
As rain's crescendo floods the night,
Washing clean the soul's dark plight.
Beneath the eaves, a melody spins,
Echoes dance where the wild heart grins,
For in the storm's crescive roar,
Music and rain forever soar.

Together they compose the scene,

deepseek/deepseek-chat

"Melody of the Rain"

The sky hums low in a silver refrain,
A symphony woven from whispers of rain.
Pattering rhythms on rooftops play,
Nature's own music, soft and gray.

The thunder's drum, the wind's sweet flute,
Dancing together, wild and mute.
Puddles catch every note that falls,
Liquid arpeggios on pavement walls.

A violin's sigh in the breeze's embrace,
A cello's deep hum in the storm's slow pace.
The clouds keep time with the earth's own beat,
A lullaby tender, a song so sweet.

Oh, music and rain—two souls entwined,
One made of air, one of tears from the sky.
Together they sing of joy and pain,
A timeless duet—**the storm and the strain.**

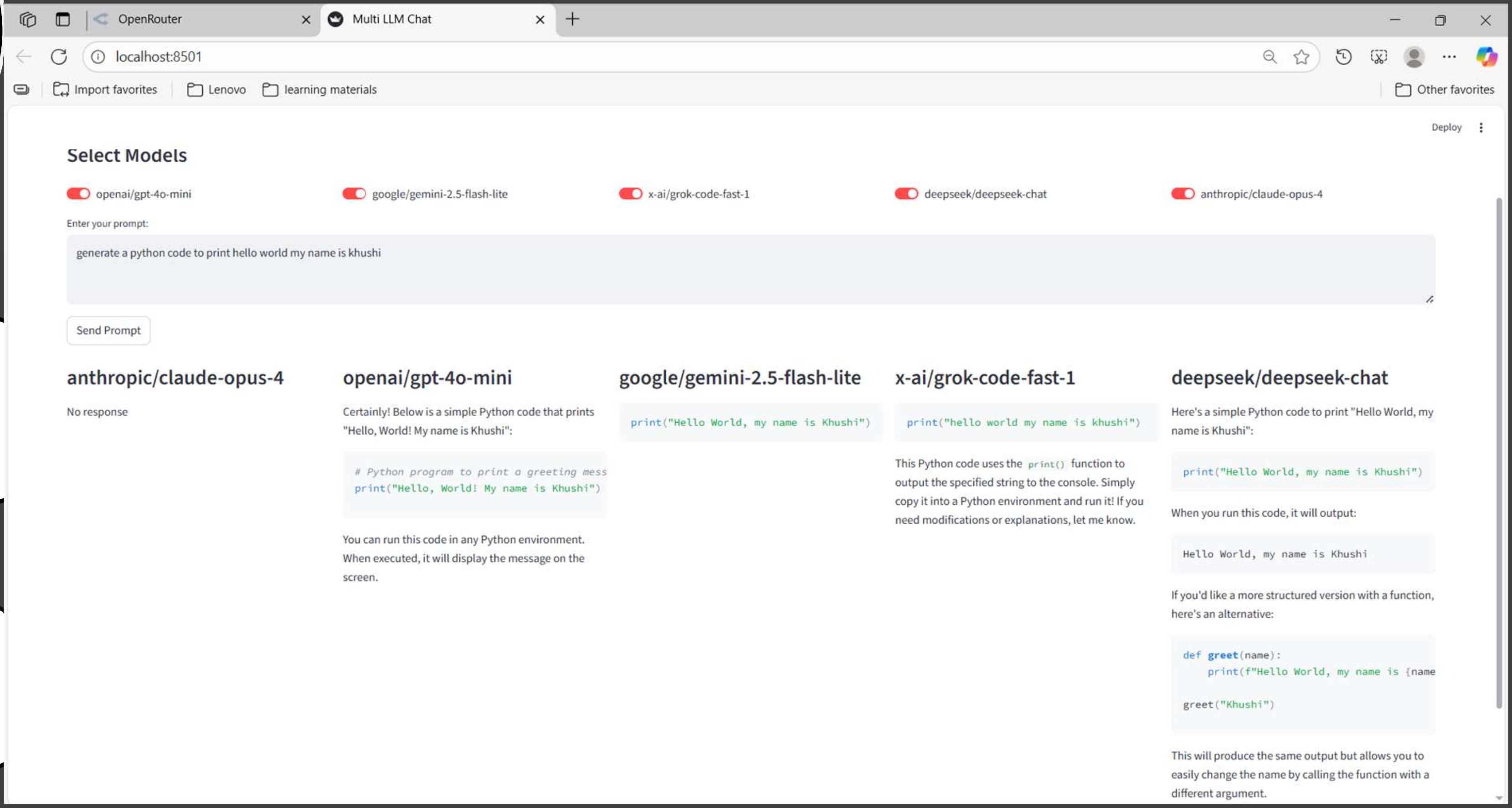
Claude model did not worked

Grok was best here

By-
Khushi Chadha

TASK 2:-
Python Code:
Print Hello
World with
Name

INSIGHTS :-



Select Models

- openai/gpt-4o-mini
- google/gemini-2.5-flash-lite
- x-ai/grok-code-fast-1
- deepseek/deepseek-chat
- anthropic/claude-opus-4

Enter your prompt:

generate a python code to print hello world my name is khushi

Send Prompt

anthropic/claude-opus-4

No response

openai/gpt-4o-mini

Certainly! Below is a simple Python code that prints "Hello, World! My name is Khushi":

```python
# Python program to print a greeting mess
print("Hello, World! My name is Khushi")
```

You can run this code in any Python environment. When executed, it will display the message on the screen.

google/gemini-2.5-flash-lite

```python
print("Hello World, my name is Khushi")
```

x-ai/grok-code-fast-1

```python
print("hello world my name is khushi")
```

This Python code uses the `print()` function to output the specified string to the console. Simply copy it into a Python environment and run it! If you need modifications or explanations, let me know.

deepseek/deepseek-chat

Here's a simple Python code to print "Hello World, my name is Khushi":

```python
print("Hello World, my name is Khushi")
```

When you run this code, it will output:

```
Hello World, my name is Khushi
```

If you'd like a more structured version with a function, here's an alternative:

```python
def greet(name):
    print(f"Hello World, my name is {name

greet("Khushi")
```

This will produce the same output but allows you to easily change the name by calling the function with a different argument.

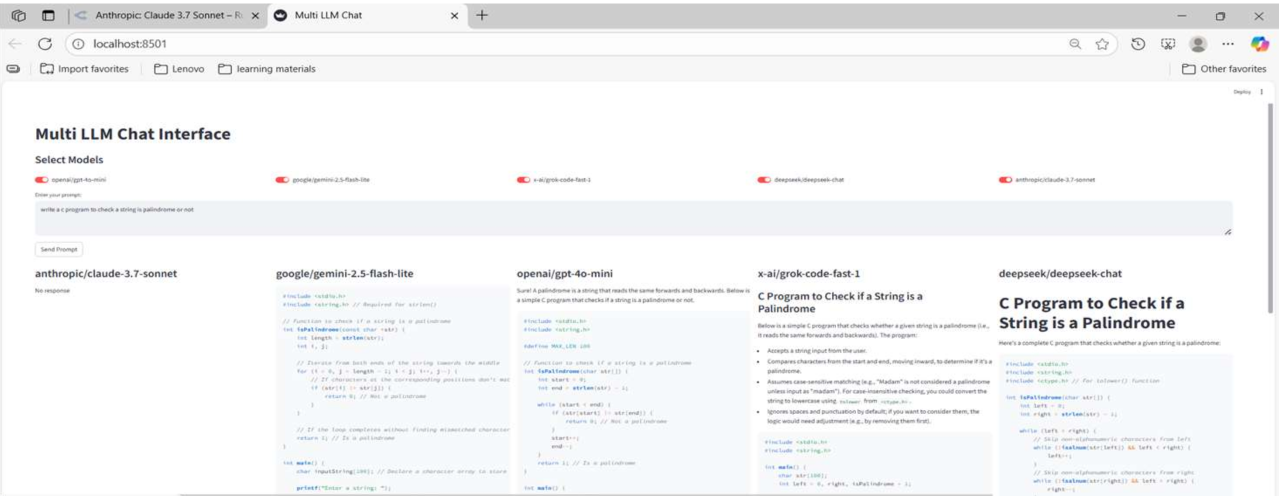
Claude model did not worked

Grok was best here

By-
Khushi Chadha

TASK 3:-
C Program Logical and structural Question

INSIGHTS :-



Multi LLM Chat Interface

Select Models

- openai/gpt-4o-mini
- google/gemini-2.5-flash-lite
- x-ai/grok-code-fast-1
- deepseek/deepseek-chat
- anthropic/claude-3.7-sonnet

Enter your prompt:

write a c program to check a string is palindrome or not

Send Prompt

anthropic/claude-3.7-sonnet

No response

google/gemini-2.5-flash-lite

```c
#include <stdio.h>
#include <string.h> // Required for strlen()

// Function to check if a string is a palindrome
int isPalindrome(const char *str) {
    int length = strlen(str);
    int i, j;

    // Iterate from both ends of the string towards the middle
    for (i = 0, j = length - 1; i < j; i++, j--) {
        // If characters at the corresponding positions don't mat
        if (str[i] != str[j]) {
            return 0; // Not a palindrome
        }
    }

    // If the loop completes without finding mismatched character
    return 1; // Is a palindrome
}

int main() {
    char inputString[100]; // Declare a character array to store

    printf("Enter a string: ");
```

openai/gpt-4o-mini

Sure! A palindrome is a string that reads the same forwards and backwards. Below is a simple C program that checks if a string is a palindrome or not.

```c
#include <stdio.h>
#include <string.h>

#define MAX_LEN 100

// Function to check if a string is a palindrome
int isPalindrome(char str[]) {
    int start = 0;
    int end = strlen(str) - 1;

    while (start < end) {
        if (str[start] != str[end]) {
            return 0; // Not a palindrome
        }
        start++;
        end--;
    }
    return 1; // Is a palindrome
}

int main() {
```

x-ai/grok-code-fast-1

C Program to Check if a String is a Palindrome

Below is a simple C program that checks whether a given string is a palindrome (i.e., it reads the same forwards and backwards). The program:

- Accepts a string input from the user.
- Compares characters from the start and end, moving inward, to determine if it's a palindrome.
- Assumes case-sensitive matching (e.g., "Madam" is not considered a palindrome unless input as "madam"). For case-insensitive checking, you could convert the string to lowercase using `tolower` from `<ctype.h>`.
- Ignores spaces and punctuation by default; if you want to consider them, the logic would need adjustment (e.g., by removing them first).

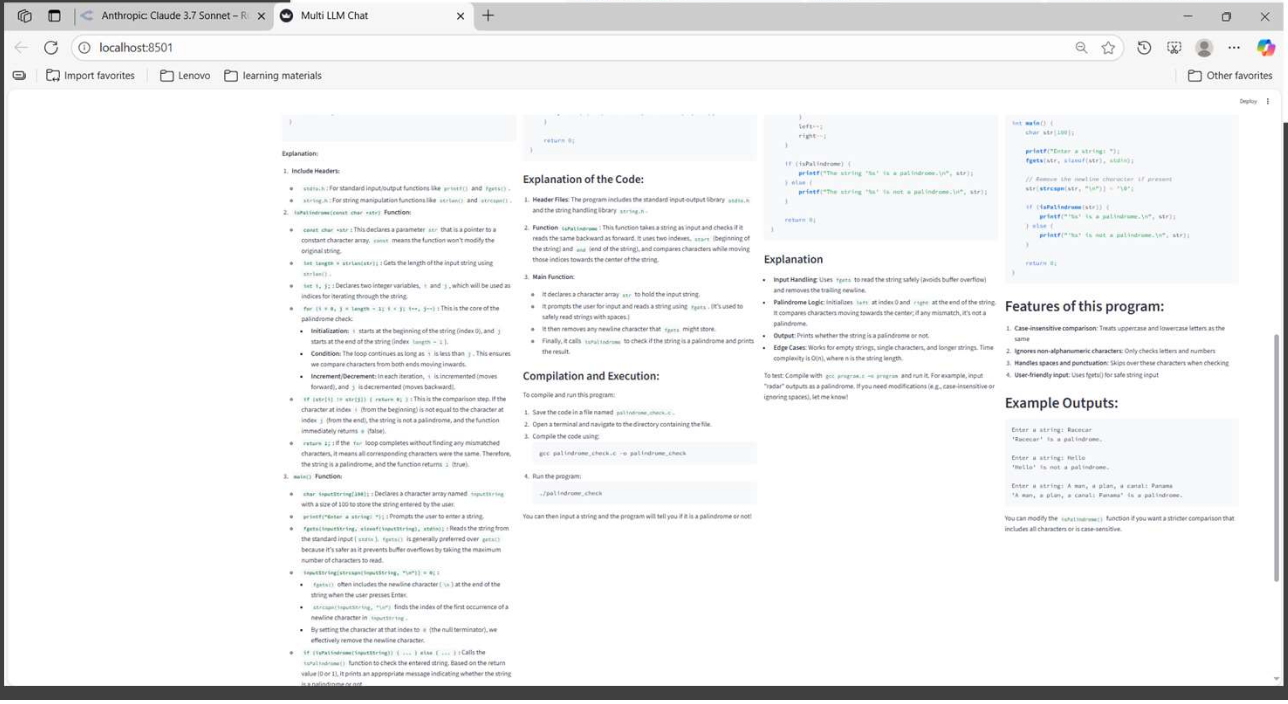
```c
#include <stdio.h>
#include <string.h>

int main() {
    char str[100];
    int left = 0, right, isPalindrome = 1;
```

deepseek/deepseek-chat

C Program to Check if a String is a Palindrome

Here's a complete C program that checks whether a given string is a palindrome:

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h> // For tolower() function

int isPalindrome(char str[]) {
    int left = 0;
    int right = strlen(str) - 1;

    while (left < right) {
        // Skip non-alphanumeric characters from left
        while (!isalnum(str[left]) && left < right) {
            left++;
        }
        // Skip non-alphanumeric characters from right
        while (!isalnum(str[right]) && left < right) {
            right--;
        }

        // Compare characters (case-insensitive)
        if (tolower(str[left]) != tolower(str[right])) {
            return 0; // Not a palindrome
        }

        left++;
        right--;
    }

    return 1; // It's a palindrome
}
```



```c
}
```

Explanation:

1. **Include Headers:**
   - `stdio.h`: For standard input/output functions like `printf()` and `fgets()`.
   - `string.h`: For string manipulation functions like `strlen()` and `strcspn()`.
2. `isPalindrome(const char *str)` **Function:**
   - `const char *str`: This declares a parameter `str` that is a pointer to a constant character array. `const` means the function won't modify the original string.
   - `int length = strlen(str);`: Gets the length of the input string using `strlen()`.
   - `int i, j;`: Declares two integer variables, `i` and `j`, which will be used as indices for iterating through the string.
   - `for (i = 0, j = length - 1; i < j; i++, j--)`: This is the core of the palindrome check:
     - **Initialization:** `i` starts at the beginning of the string (index 0), and `j` starts at the end of the string (index `length - 1`).
     - **Condition:** The loop continues as long as `i` is less than `j`. This ensures we compare characters from both ends moving inwards.
     - **Increment/Decrement:** In each iteration, `i` is incremented (moves forward), and `j` is decremented (moves backward).
   - `if (str[i] != str[j]) { return 0; }`: This is the comparison step. If the character at index `i` (from the beginning) is not equal to the character at index `j` (from the end), the string is not a palindrome, and the function immediately returns `0` (false).
   - `return 1;`: If the `for` loop completes without finding any mismatched characters, it means all corresponding characters were the same. Therefore, the string is a palindrome, and the function returns `1` (true).
3. `main()` **Function:**
   - `char inputString[100];`: Declares a character array named `inputString` with a size of 100 to store the string entered by the user.
   - `printf("Enter a string: ");`: Prompts the user to enter a string.
   - `fgets(inputString, sizeof(inputString), stdin);`: Reads the string from the standard input (`stdin`). `fgets()` is generally preferred over `gets()` because it's safer as it prevents buffer overflows by taking the maximum number of characters to read.
   - `inputString[strcspn(inputString, "\n")] = 0;`:
     - `fgets()` often includes the newline character (`\n`) at the end of the string when the user presses Enter.
     - `strcspn(inputString, "\n")` finds the index of the first occurrence of a newline character in `inputString`.
     - By setting the character at that index to `0` (the null terminator), we effectively remove the newline character.
   - `if (isPalindrome(inputString)) { ... } else { ... }`: Calls the `isPalindrome()` function to check the entered string. Based on the return value (0 or 1), it prints an appropriate message indicating whether the string is a palindrome or not.

```c
    }

    return 0;
}
```

Explanation of the Code:

1. **Header Files**: The program includes the standard input-output library `stdio.h` and the string handling library `string.h`.
2. **Function** `isPalindrome`: This function takes a string as input and checks if it reads the same backward as forward. It uses two indexes, `start` (beginning of the string) and `end` (end of the string), and compares characters while moving those indices towards the center of the string.
3. **Main Function**:
   - It declares a character array `str` to hold the input string.
   - It prompts the user for input and reads a string using `fgets`. (It's used to safely read strings with spaces.)
   - It then removes any newline character that `fgets` might store.
   - Finally, it calls `isPalindrome` to check if the string is a palindrome and prints the result.

Compilation and Execution:

To compile and run this program:

1. Save the code in a file named `palindrome_check.c`.
2. Open a terminal and navigate to the directory containing the file.
3. Compile the code using:

```
gcc palindrome_check.c -o palindrome_check
```

4. Run the program:

```
./palindrome_check
```

You can then input a string and the program will tell you if it is a palindrome or not!

```c
        }
        left++;
        right--;
    }

    if (isPalindrome) {
        printf("The string '%s' is a palindrome.\n", str);
    } else {
        printf("The string '%s' is not a palindrome.\n", str);
    }

    return 0;
}
```

Explanation

- **Input Handling**: Uses `fgets` to read the string safely (avoids buffer overflow) and removes the trailing newline.
- **Palindrome Logic**: Initializes `left` at index 0 and `right` at the end of the string. It compares characters moving towards the center; if any mismatch, it's not a palindrome.
- **Output**: Prints whether the string is a palindrome or not.
- **Edge Cases**: Works for empty strings, single characters, and longer strings. Time complexity is O(n), where n is the string length.

To test: Compile with `gcc program.c -o program` and run it. For example, input "radar" outputs as a palindrome. If you need modifications (e.g., case-insensitive or ignoring spaces), let me know!

```c
int main() {
    char str[100];

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Remove the newline character if present
    str[strcspn(str, "\n")] = '\0';

    if (isPalindrome(str)) {
        printf("'%s' is a palindrome.\n", str);
    } else {
        printf("'%s' is not a palindrome.\n", str);
    }

    return 0;
}
```

Features of this program:

1. **Case-insensitive comparison**: Treats uppercase and lowercase letters as the same
2. **Ignores non-alphanumeric characters**: Only checks letters and numbers
3. **Handles spaces and punctuation**: Skips over these characters when checking
4. **User-friendly input**: Uses fgets() for safe string input

Example Outputs:

```
Enter a string: Racecar
'Racecar' is a palindrome.

Enter a string: Hello
'Hello' is not a palindrome.

Enter a string: A man, a plan, a canal: Panama
'A man, a plan, a canal: Panama' is a palindrome.
```

You can modify the `isPalindrome()` function if you want a stricter comparison that includes all characters or is case-sensitive.

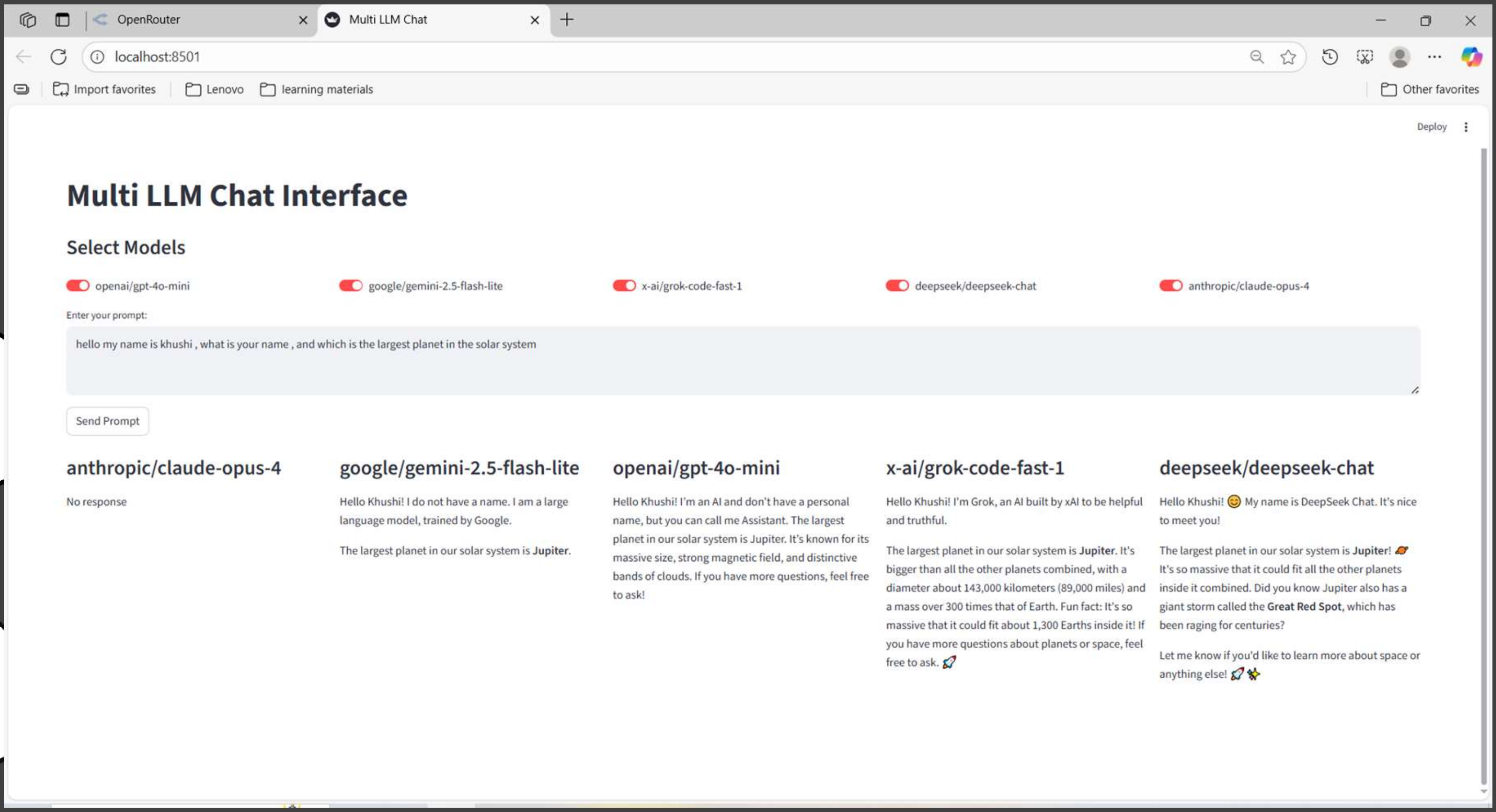
Claude model did not worked

Openai was best here

By-
Khushi Chadha

TASK 4:-
Greetings and factual Question

INSIGHTS :-



Multi LLM Chat Interface

Select Models

● openai/gpt-4o-mini ● google/gemini-2.5-flash-lite ● x-ai/grok-code-fast-1 ● deepseek/deepseek-chat ● anthropic/claude-opus-4

Enter your prompt:

hello my name is khushi , what is your name , and which is the largest planet in the solar system

anthropic/claude-opus-4

No response

google/gemini-2.5-flash-lite

Hello Khushi! I do not have a name. I am a large language model, trained by Google.

The largest planet in our solar system is **Jupiter**.

openai/gpt-4o-mini

Hello Khushi! I'm an AI and don't have a personal name, but you can call me Assistant. The largest planet in our solar system is Jupiter. It's known for its massive size, strong magnetic field, and distinctive bands of clouds. If you have more questions, feel free to ask!

x-ai/grok-code-fast-1

Hello Khushi! I'm Grok, an AI built by xAI to be helpful and truthful.

The largest planet in our solar system is **Jupiter**. It's bigger than all the other planets combined, with a diameter about 143,000 kilometers (89,000 miles) and a mass over 300 times that of Earth. Fun fact: It's so massive that it could fit about 1,300 Earths inside it! If you have more questions about planets or space, feel free to ask. 🚀

deepseek/deepseek-chat

Hello Khushi! 😊 My name is DeepSeek Chat. It's nice to meet you!

The largest planet in our solar system is **Jupiter**! 🪐 It's so massive that it could fit all the other planets inside it combined. Did you know Jupiter also has a giant storm called the **Great Red Spot**, which has been raging for centuries?

Let me know if you'd like to learn more about space or anything else! 🚀✨

Claude model did not worked

Grok was best here

By-
Khushi Chadha

LLM’S BENCHMARKING OVERVIEW REPORT

OpenAI GPT-4o-mini

01

A lightweight yet powerful model optimized for fast response generation and balanced performance. It delivers clear, structured outputs suitable for both creative and technical tasks.

Google Gemini 2.5-Flash-Lite

02

Designed for speed and expressiveness, this model creates rich, human-like content with strong creativity. Performs well in conversational and content-generation tasks. ideal for engaging, natural outputs.

X.ai Grok-Code-Fast-1

03

Focused on quick code generation and concise outputs, making it well-suited for programming-related tasks. While efficient for technical use, it provides minimal creativity for non-coding prompts. Best when speed is a priority and overall was best.

DeepSeek Chat

04

A versatile model delivering consistent and well-balanced responses across varied tasks. It combines dependability with adaptability, handling both creative and technical prompts effectively.

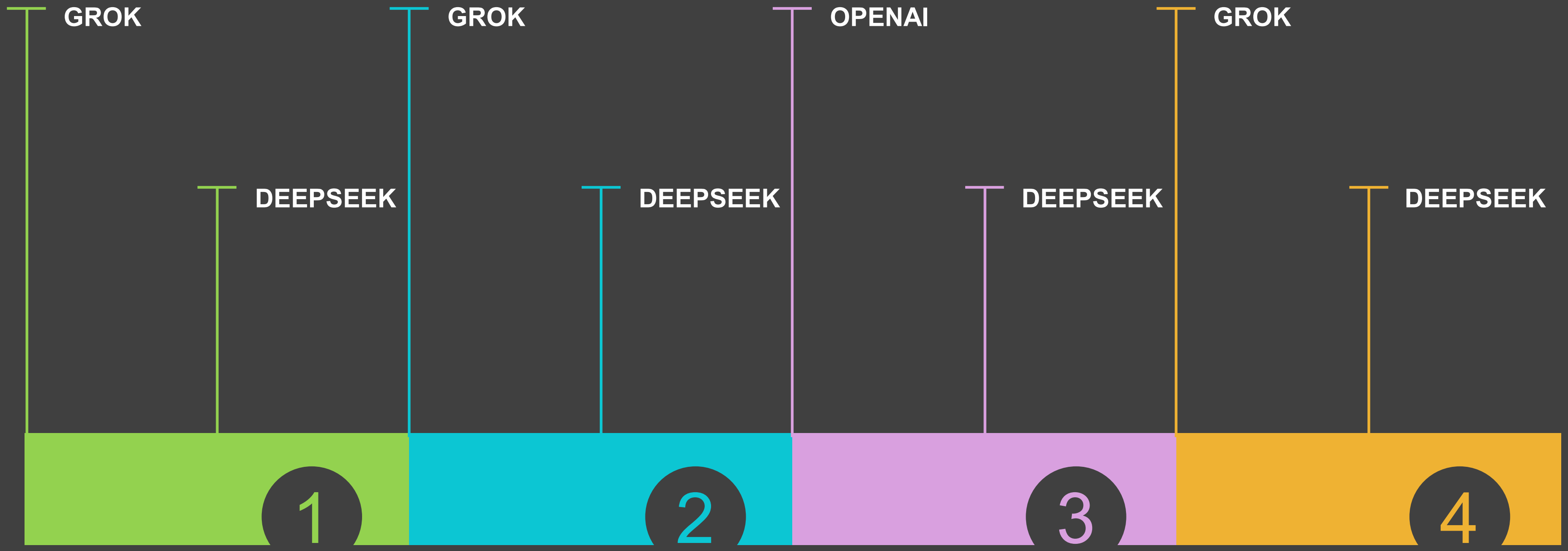
Anthropic Claude Opus-4

05

A high-capacity model with potential for advanced reasoning and creative generation. However, in this benchmark, it failed to produce responses across tasks. Expected to perform well when active

By-
Khushi Chadha

OVERALL RANKING TASK-WISE



GROK

DEEPSEEK

1

GROK

DEEPSEEK

2

OPENAI

DEEPSEEK

3

GROK

DEEPSEEK

4

By-
Khushi Chadha

FINAL RESULTS :-

Model	Efficiency	Response Quality	Relevance	Strengths	Weaknesses
X.ai Grok-Code-Fast-1	High	High	High	Balanced across tasks, less verbose for creative tasks	None significant
DeepSeek Chat	High	High	High	Creative richness, human-like email tone	Sometimes verbose
OpenAI GPT-4o-mini	High	High	High	Fast and functional for coding	Creative/email outputs lacked depth
Google Gemini 2.5-Flash-Lite	High	High	High	Reliable and versatile	Slightly template-like
Anthropic Claude Opus-4	N/A	Low	N/A	–	Failed to produce output

By- Khushi Chadha