AFFINE

COMMAND THE NEW

IntelliCodEx

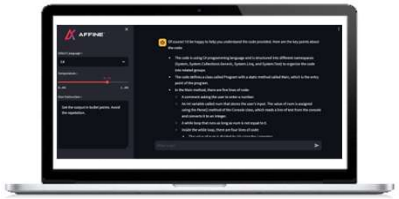
https://affine.ai/

Intelligent Code Explainer: IntelliCodEx

BUSINESS PROBLEM

Mercedes Benz wants to create a smart documentation assistant that would explain and document their existing and legacy codes which will be deployed on their premises

OUTPUT



The output was consumed in the form of a IDE plugin/web tool that facilitates seamless access to upload code files through an intuitive and efficient user interface.

CAPABILITY USED

Prompt Engineering: Prompts are fine-tuned to give the user flexibility to obtain the results in the format they require for different programming languages

Hosting: The model will be hosted on a suitable on-prem device and linked within the private network to make it accessible for anyone in the network to the user query .

User Interface: Built a dedicated user interface that helps to submit the code snippets and perform contextual Q&A if necessary, which can be also integrated to an IDE of client's choice

BUSINESS BENEFITS

Reduced Time to search through the document manually

Increased Scalability to handle multiple queries simultaneously

A centralized and trusted source of the data

Industry

AUTOMOTIVE

Function

DOCUMENTATION

Data Used

Existing and Legacy Codes

Tech Stack

python

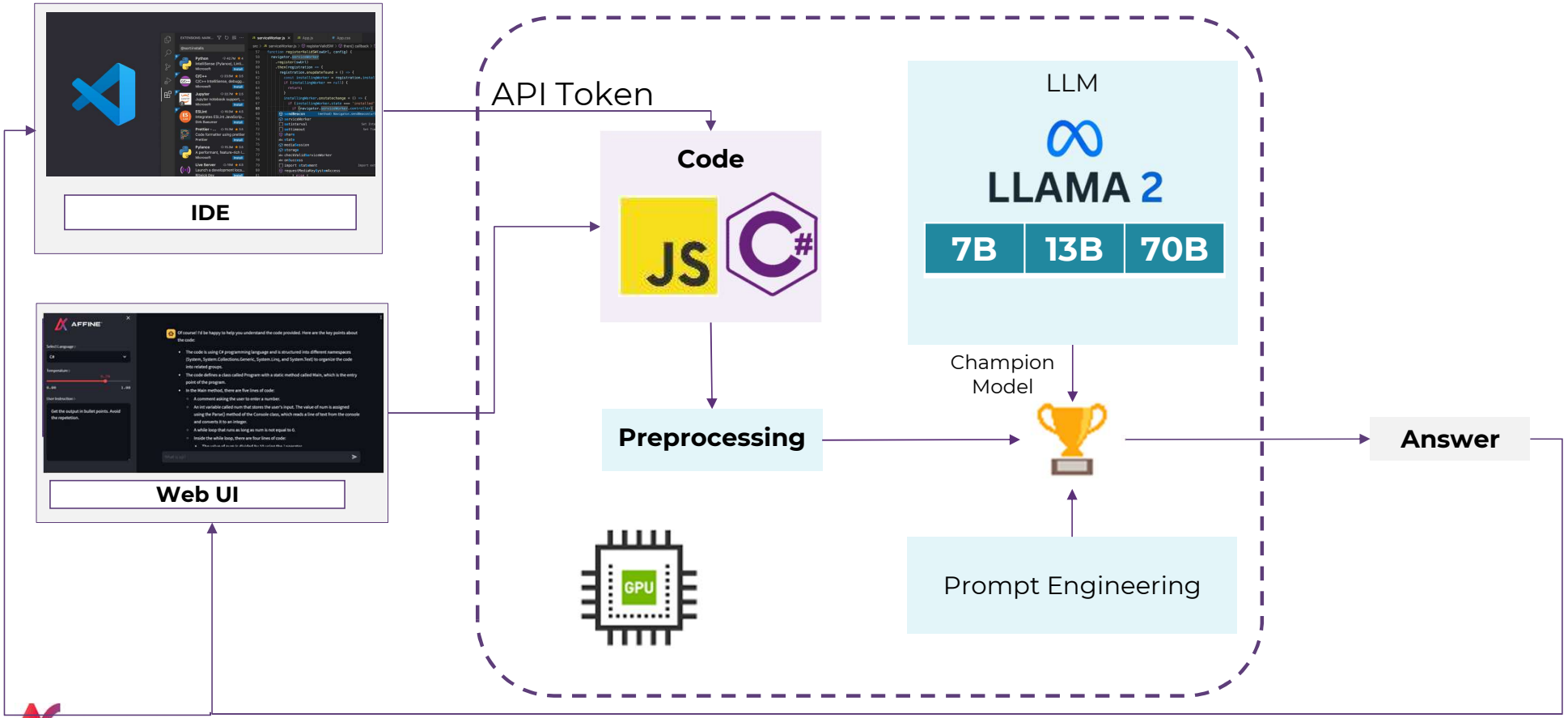
LLAMA 2

JS

C#

Technical Architecture

Illustrative



Sample Results: C#

```
using System;
class Program
{
    static void Main()
    {
        Console.WriteLine("Enter the Base : ");
        double num = double.Parse(Console.ReadLine());
        Console.WriteLine("Enter the First Exponent :");
        double exp1 = double.Parse(Console.ReadLine());
        Console.WriteLine("Enter the Second Exponent :");
        double exp2 = double.Parse(Console.ReadLine());
        double mul;
        mul = exp1 + exp2;
        Console.WriteLine("Result is : {0}^{1} : {2}", num,
mul, Math.Pow(num, mul));
        Console.ReadLine();
    }
}
```

Sure, I'd be happy to help! Here are the main points of the code:

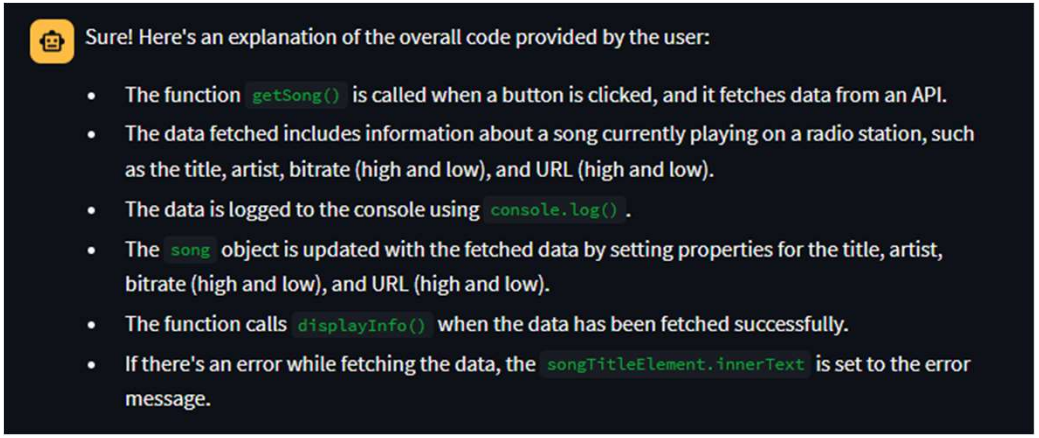
- The code is written in C# and uses the `System` namespace.
- The program prompts the user to enter a base (using `Console.WriteLine("Enter the Base : ")`) and then reads the input using `double.Parse()`. This stores the value of the base in a variable called `num`.
- The program then prompts the user to enter two exponents (using `Console.WriteLine("Enter the First Exponent :")` and `Console.WriteLine("Enter the Second Exponent :")`). Again, these inputs are read using `double.Parse()`. These values are stored in variables called `exp1` and `exp2`, respectively.
- The program then calculates the result of raising the base to the first exponent plus the base to the second exponent using the formula `mul = exp1 + exp2`. This value is assigned to a variable called `mul`.
- Finally, the program prints out the result using `Console.WriteLine("Result is : {0}^{1} : {2}", num, mul, Math.Pow(num, mul))`, where `{0}` and `{2}` are placeholders for the base and the result, respectively. The `Math.Pow()` method is used to calculate the actual value of the result.
- The program ends with a call to `Console.ReadLine()`, which waits for the user to press Enter before continuing.

Sample Results: Javascript

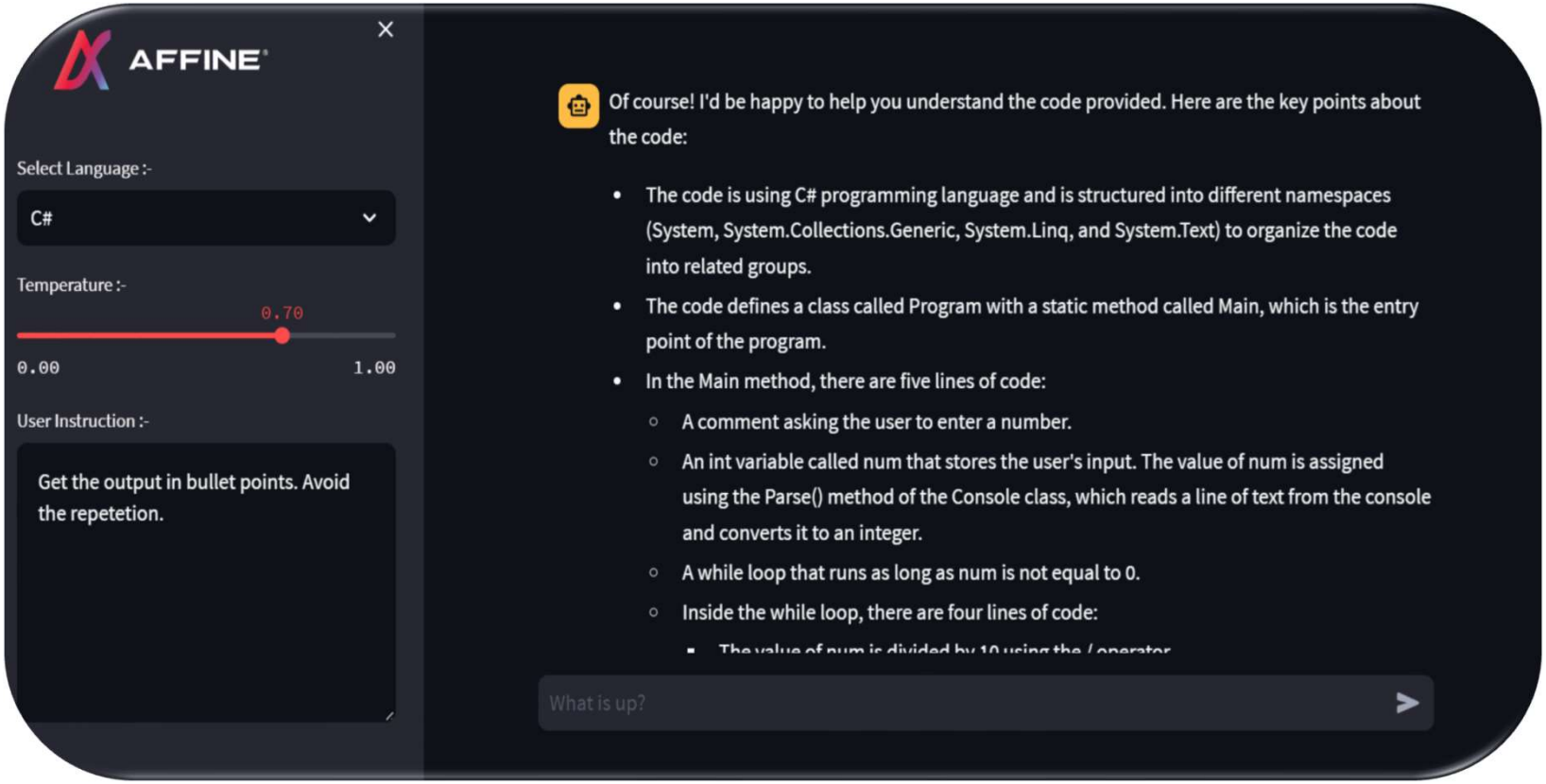
```
function getSong(){
  fetch(api)
    .then(response => {
      return response.json();
    })
    .then(data => {
      console.log(data);
      song.title = data.now_playing.song.title;
      song.artist = data.now_playing.song.artist;

      song.bitrate.high = data.station.mounts[0].bitrate;
      song.url.high = data.station.mounts[0].url;

      song.bitrate.low = data.station.mounts[1].bitrate;
      song.url.low = data.station.mounts[1].url;
    })
    .then(() => {
      displayInfo();
    })
    .catch(error =>{
      songTitleElement.innerText = error.message;
    });
}
```



UI Snapshot



Notes

1. Option to choose a programming language
2. Flexibility to user instruction to change document style
3. Option for parameter setting to tune the model
4. Enter the code base
5. View, Save or export the document from executed instruction

Key Features Of Our Solution

LARGE CODE BASE HANDLING
Large code base can be preprocessed to be fit into token limits to handle large code at one go

OUTPUT FLEXIBILITY
Provides the capability to showcase in bullet list or paragraph as the user requires, can also be summarized function-wise. Also Can be converted to different versions of the language

NL QUERY ENGINE
Translates the natural language query by identifying the accurate intent & context to generate the meaningful explanation of the code

EDIT, REFINE & REVIEW
Capability to edit & refine generated document as needed, allowing amalgamation of AI & human expertise to create functional document in time efficient manner with user review on generated results.

EXECUTE & SAVE
Allows execution of the generated script against the required data table to collect data, capability to save the query for future reference & reveiw purposes

SUMMARIZATION OF CODE
Inbuilt capabilities to perform code summarization

SECURE & SAFE
Easy to integrate with active directory, can support multi-factor authentication to avoid unauthorized access. Multi-tenant architecture to allow the creation of projects & restrict its access

THANK YOU

https://affine.ai/

AFFINE