

tc

HackNiche 2024  
PS-3

# **Tensionflow**

## **empowering developers**

### **with Tension\_Code**

DEVANSH RATHOR  
ABHAY MATHUR  
NAMAN LAKHANI  
SHRUTI SHAH

# OVERVIEW

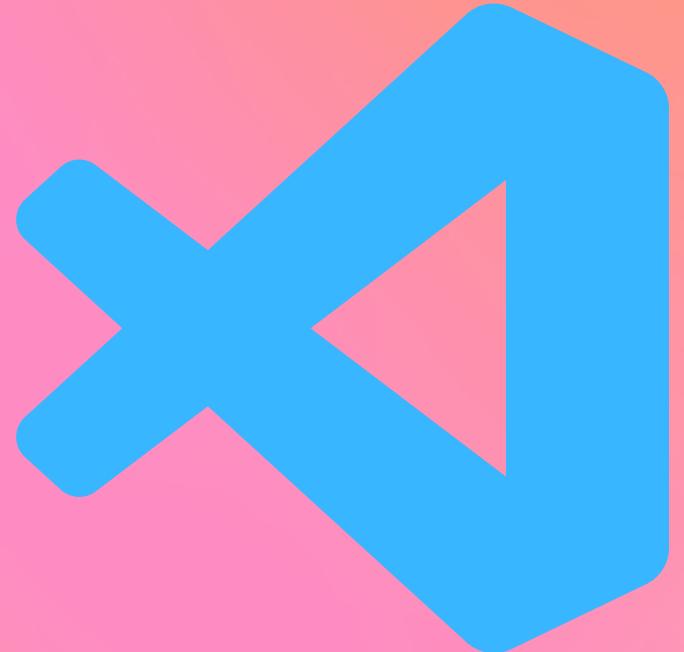
Welcome to the future of software development with TensionCode. We've developed a cutting-edge platform that streamlines the entire development process, from schema extraction to code generation and evaluation, all while offering seamless integration with popular development tools.

# **NEED FOR TENSION\_CODE**

Traditional development workflows are often cumbersome and time-consuming, requiring developers to manually translate database schemas into functional code. This process is prone to errors, inefficiencies, and inconsistencies across different databases and programming languages. Moreover, maintaining version control and providing code explanations can be challenging, leading to reduced productivity and code quality.

# Tech-Stack

~~NEXT~~.js



Flask

 LangChain

Gemini

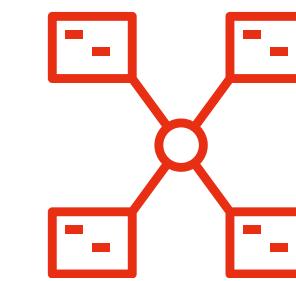
 OpenAI

# Salient Features of Tension\_Code



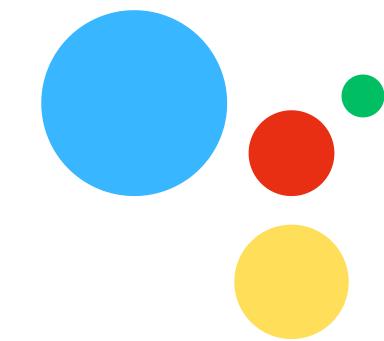
**01**

Function Code  
Generation  
and  
Evaluation



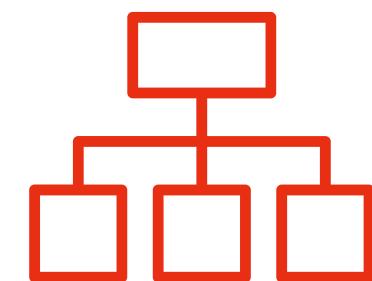
**02**

Schema from  
E-R diagram



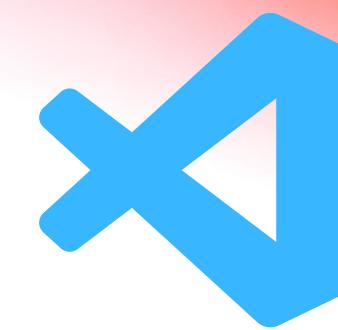
**03**

Google  
Extension



**04**

Flowchart  
Generation.



**05**

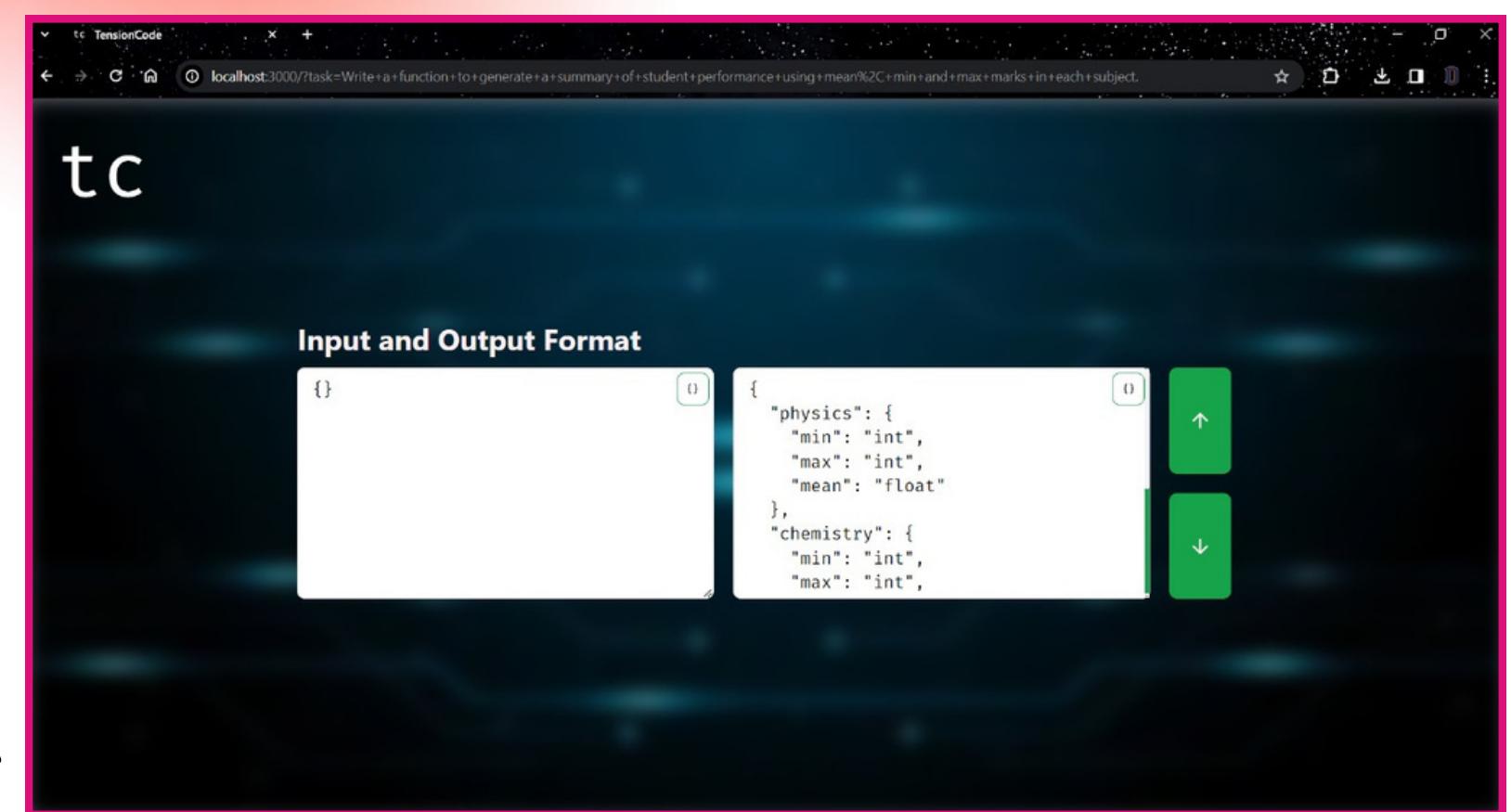
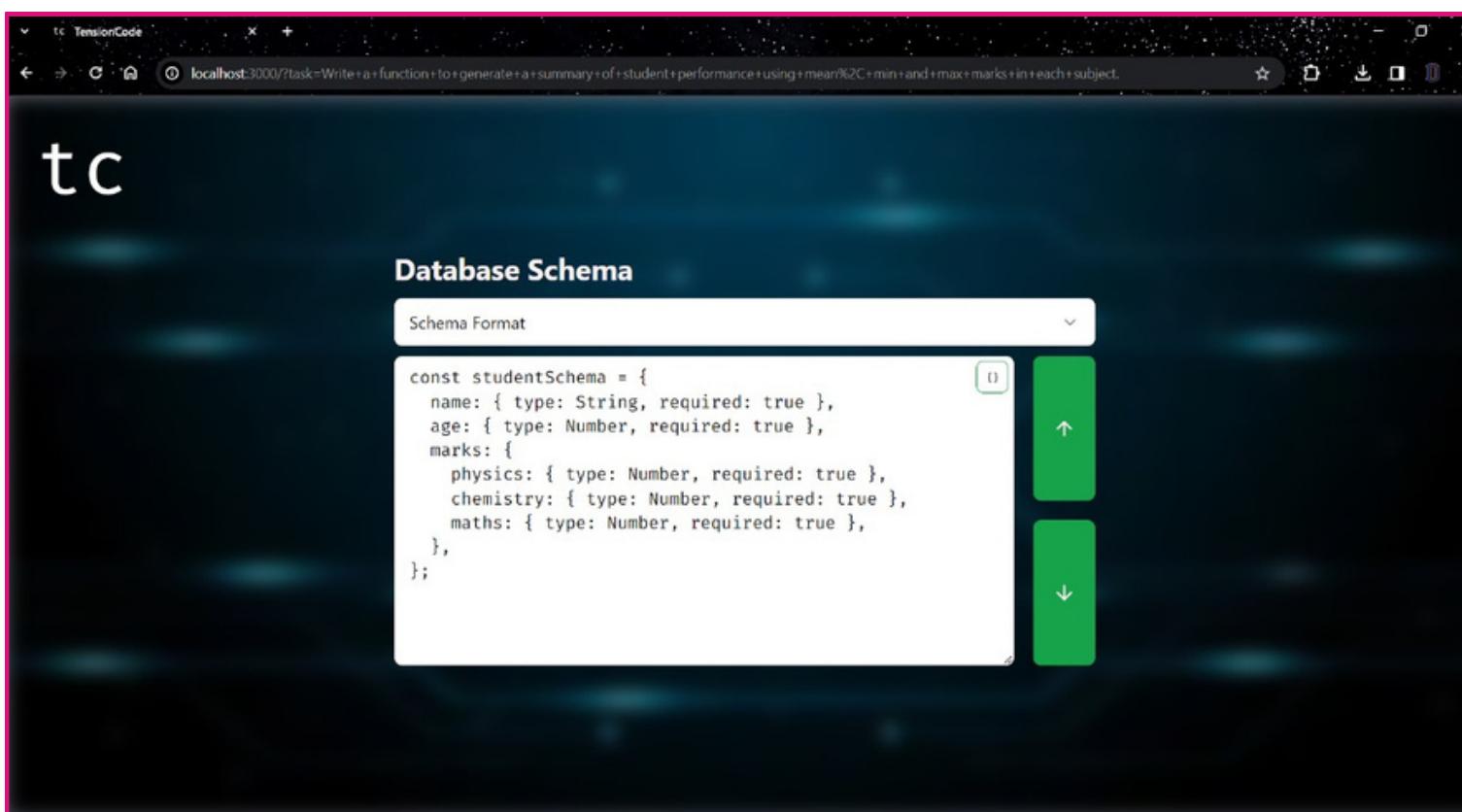
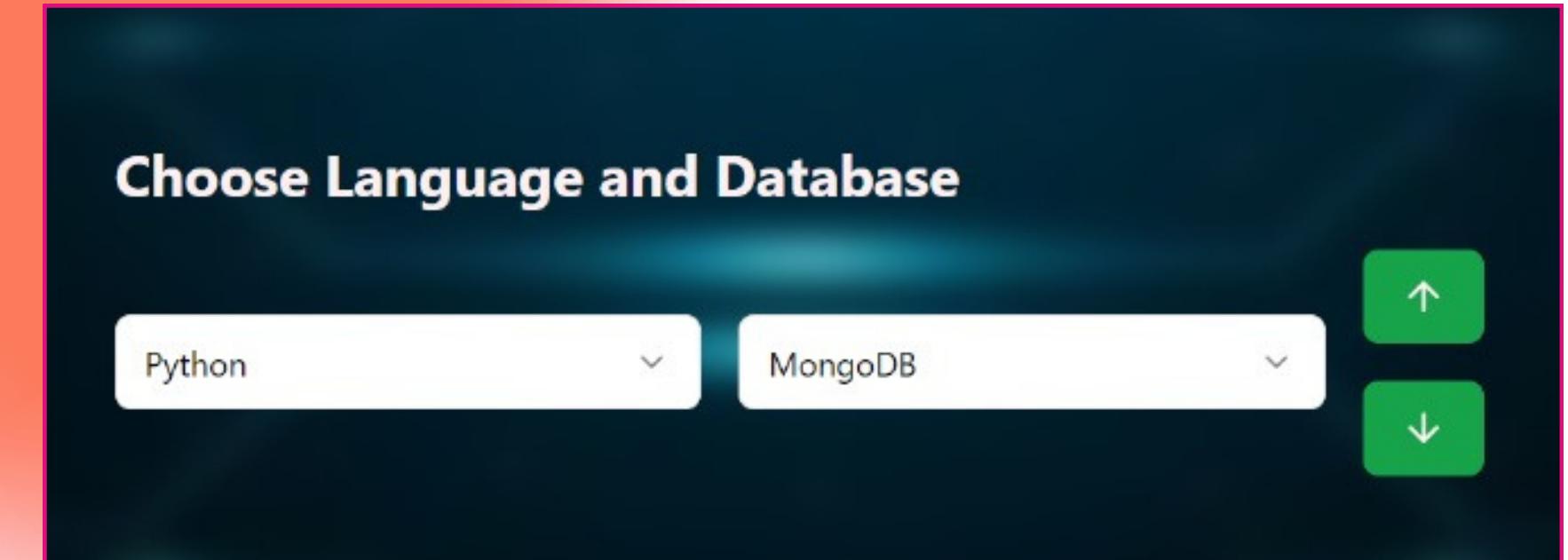
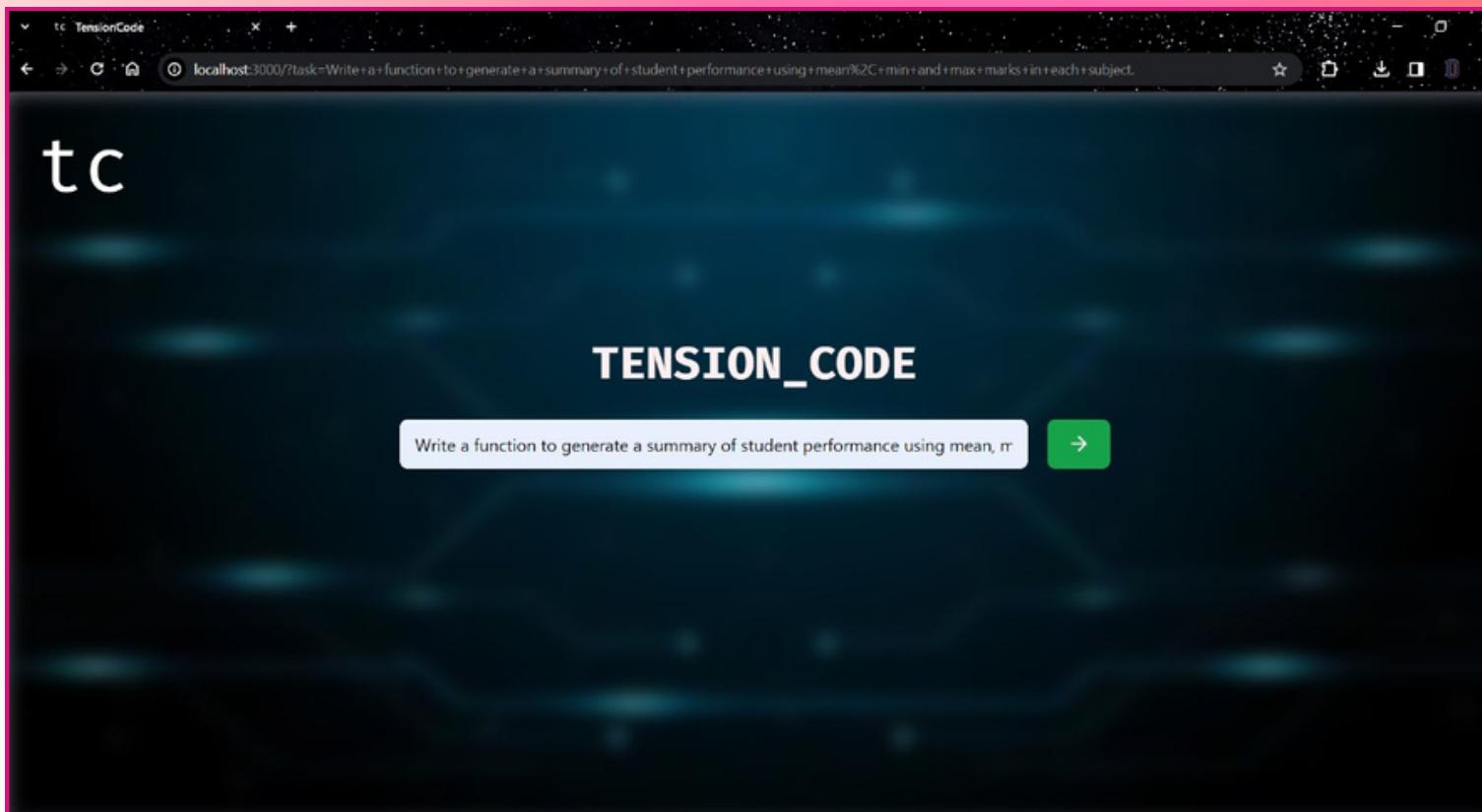
VSCode  
Integration



**06**

Youtube  
Recommendations

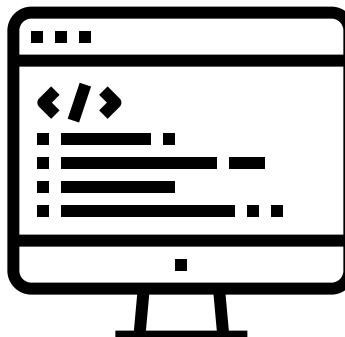
# User Interface



# Function Code Generation and Evaluation

Tension\_code goes beyond code generation by incorporating reinforcement learning techniques to optimize generated code for performance and reliability.

Developers can evaluate generated functions internally within our platform, ensuring correctness and efficiency before deployment.



A screenshot of a web-based code editor titled 'tc TensionCode'. The URL in the address bar is 'localhost:3000/?task=Write+a+function+to+generate+a+summary+of+student+performance+using+mean%2C+min+and+max+marks+in+each+subject'. The editor displays a 'Function Template' block with the following code:

```
function studentPerformance(inputParams) {  
    // Write your code here  
  
    return {  
        data : outputData,  
    }  
}
```

On the right side of the editor, there are two green buttons with white arrows: an upward arrow at the top and a downward arrow at the bottom.

# Function Code Generation and Evaluation

The screenshot shows a dark-themed web application window titled "tc TensionCode". The main area is labeled "Generated Code 1". It contains Python code for generating a summary of student performance across three subjects: Physics, Chemistry, and Maths. The code uses a loop to iterate over students and calculate min, max, and mean marks for each subject. It also includes a statistics module import and a condition to handle cases where some subjects might have no data.

```
physics": {"min": None, "max": None, "mean": None},  
"chemistry": {"min": None, "max": None, "mean": None},  
"maths": {"min": None, "max": None, "mean": None}  
}  
  
for student in students:  
    outputData["physics"]["min"] = min(outputData["physics"]["min"],  
student["marks"]["physics"]) if outputData["physics"]["min"] is not None else  
student["marks"]["physics"]  
    outputData["physics"]["max"] = max(outputData["physics"]["max"],  
student["marks"]["physics"]) if outputData["physics"]["max"] is not None else  
student["marks"]["physics"]  
    outputData["physics"]["mean"] = statistics.mean([outputData["physics"]  
["mean"], student["marks"]["physics"]]) if outputData["physics"]["mean"] is not None  
else student["marks"]["physics"]  
  
    outputData["chemistry"]["min"] = min(outputData["chemistry"]["min"],  
student["marks"]["chemistry"]) if outputData["chemistry"]["min"] is not None else  
student["marks"]["chemistry"]  
  
    outputData["maths"]["min"] = min(outputData["maths"]["min"],  
student["marks"]["maths"]) if outputData["maths"]["min"] is not None else  
student["marks"]["maths"]  
  
Write a function to generate a summary of students with mean marks above 'threshold' marks in total. →
```

The screenshot shows a dark-themed web application window titled "tc TensionCode". The main area is labeled "Generated Code 2". It contains Python code for generating student performance summaries. The code includes docstrings for the function, arguments, and returns. It also includes comments explaining the steps: getting student data from a database, initializing output data, and calculating min, max, and mean marks for each subject.

```
def studentPerformance(inputParams):  
    """This function generates a summary of student performance using mean, median,  
    min and max marks in each subject.  
  
    Args:  
        inputParams (dict): The input parameters.  
  
    Returns:  
        dict: The output data.  
    """  
  
    # Get the student data from the database.  
    students = db.students.find()  
  
    # Initialize the output data.  
    outputData = {  
        "physics": {  
            "min": None,  
            "max": None,  
            "mean": None,  
            "marks_list": [marks],  
            "median": marks,  
        }  
    }  
  
    for student in students:  
        response[subject]["min"] = min(response[subject]["min"], marks)  
        response[subject]["max"] = max(response[subject]["max"], marks)  
        response[subject]["mean"] = (response[subject]["mean"] + marks) /  
        len(response[subject]["marks_list"])  
  
        response[subject]["marks_list"].append(marks)  
        response[subject]["median"] = statistics.median(response[subject]  
["marks_list"])  
  
    return response  
except Exception as e:  
    return {"error": str(e)}  
  
Write a function to generate a summary of students with mean marks above 'threshold' marks in total. →
```

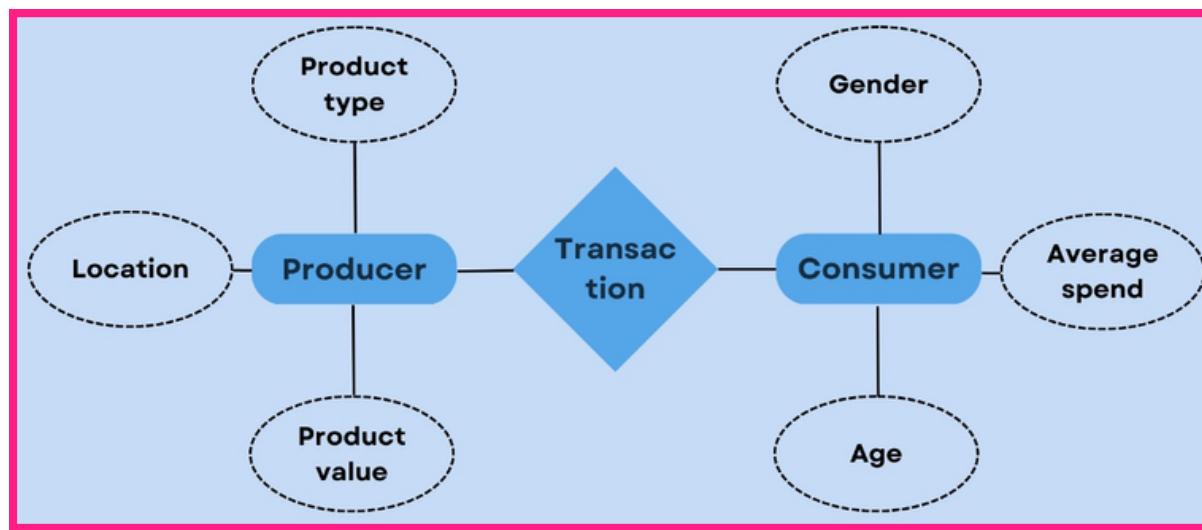
The screenshot shows a dark-themed web application window titled "tc TensionCode". The main area is labeled "Generated Code 3". It contains Python code for generating student performance summaries. This version appears to be a more refined or optimized version of the previous code, possibly using list comprehensions and more concise loops.

```
physics": {"min": marks,  
"max": marks,  
"mean": marks,  
"marks_list": [marks],  
"median": marks,  
}  
}  
else:  
    response[subject]["min"] = min(response[subject]["min"], marks)  
    response[subject]["max"] = max(response[subject]["max"], marks)  
    response[subject]["mean"] = (response[subject]["mean"] + marks) /  
    len(response[subject]["marks_list"])  
  
    response[subject]["marks_list"].append(marks)  
    response[subject]["median"] = statistics.median(response[subject]  
["marks_list"])  
  
return response  
except Exception as e:  
    return {"error": str(e)}  
  
Write a function to generate a summary of students with mean marks above 'threshold' marks in total. →
```

Optimisation and  
evaluation of code  
using Autogen

# Schema Extraction from ER-Diagram

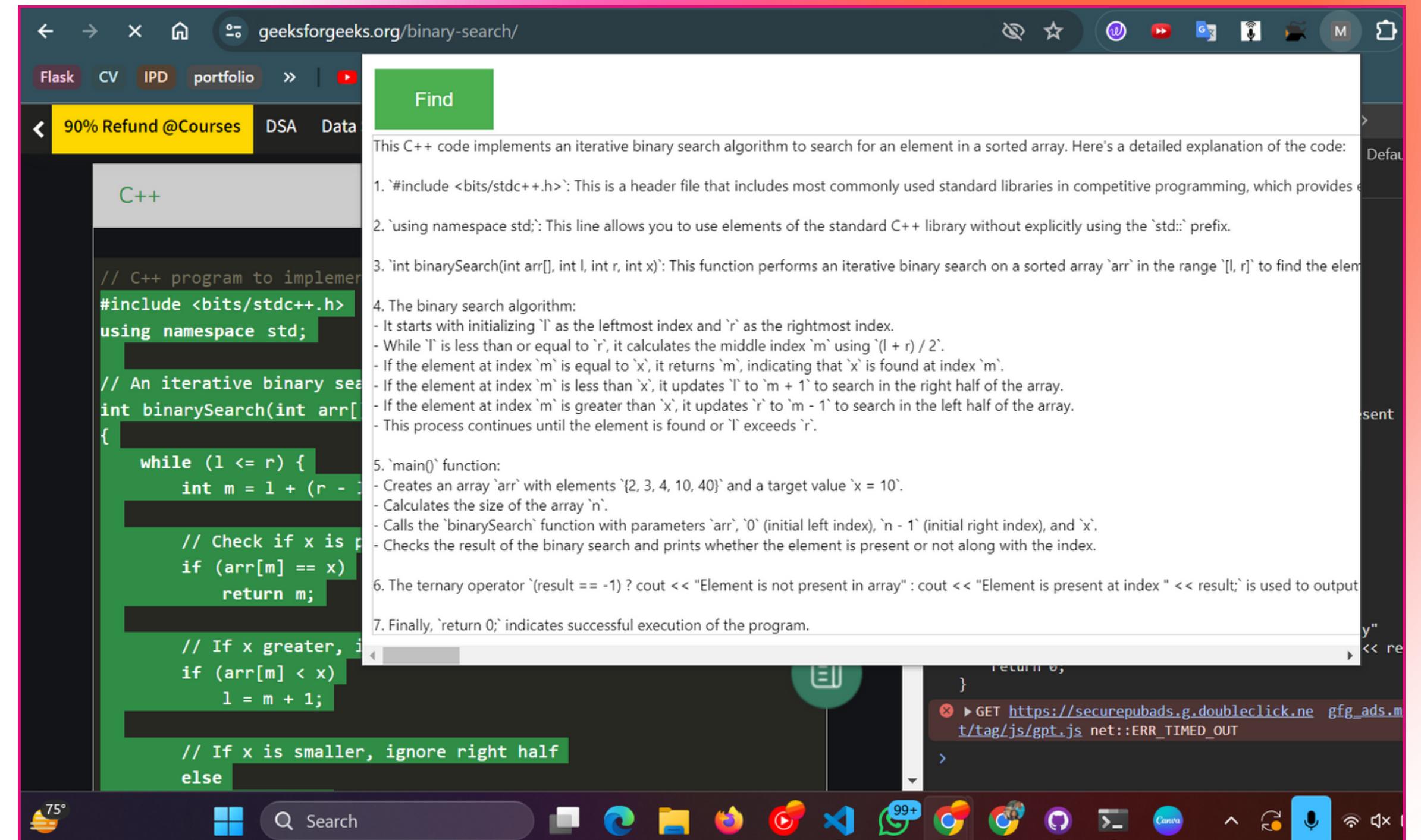
**Tension\_Code** leverages advanced algorithms to extract schemas from Entity-Relationship diagrams and automatically generates clean, efficient code tailored to the specific requirements of different databases and programming languages. Our integration with Gemini 1.0 Pro Vision ensures accurate image-to-text conversion for seamless schema extraction.



```
PowerShell
```
{
  "entities": [
    {
      "name": "Product",
      "attributes": [
        {
          "name": "product_type",
          "type": "string"
        },
        {
          "name": "product_value",
          "type": "integer"
        }
      ]
    },
    {
      "name": "Producer",
      "attributes": [
        {
          "name": "location",
          "type": "string"
        }
      ]
    },
    {
      "name": "Consumer",
      "attributes": [
        {
          "name": "gender",
          "type": "string"
        },
        {
          "name": "age",
          "type": "integer"
        },
        {
          "name": "average_spend",
          "type": "decimal"
        }
      ]
    }
  ]
}
```

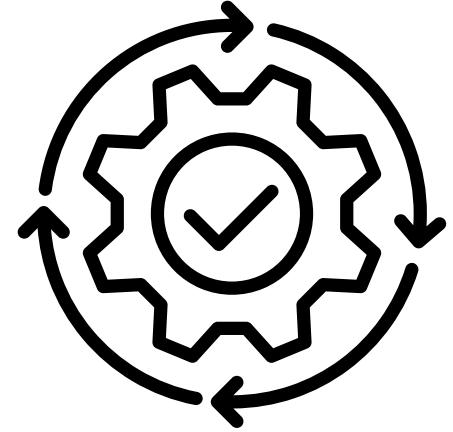
# Google Extension for Code Explanation

Our Google extension enhances developer productivity by providing instant code explanations for **third-party** code snippets, eliminating the need for manual analysis and interpretation.

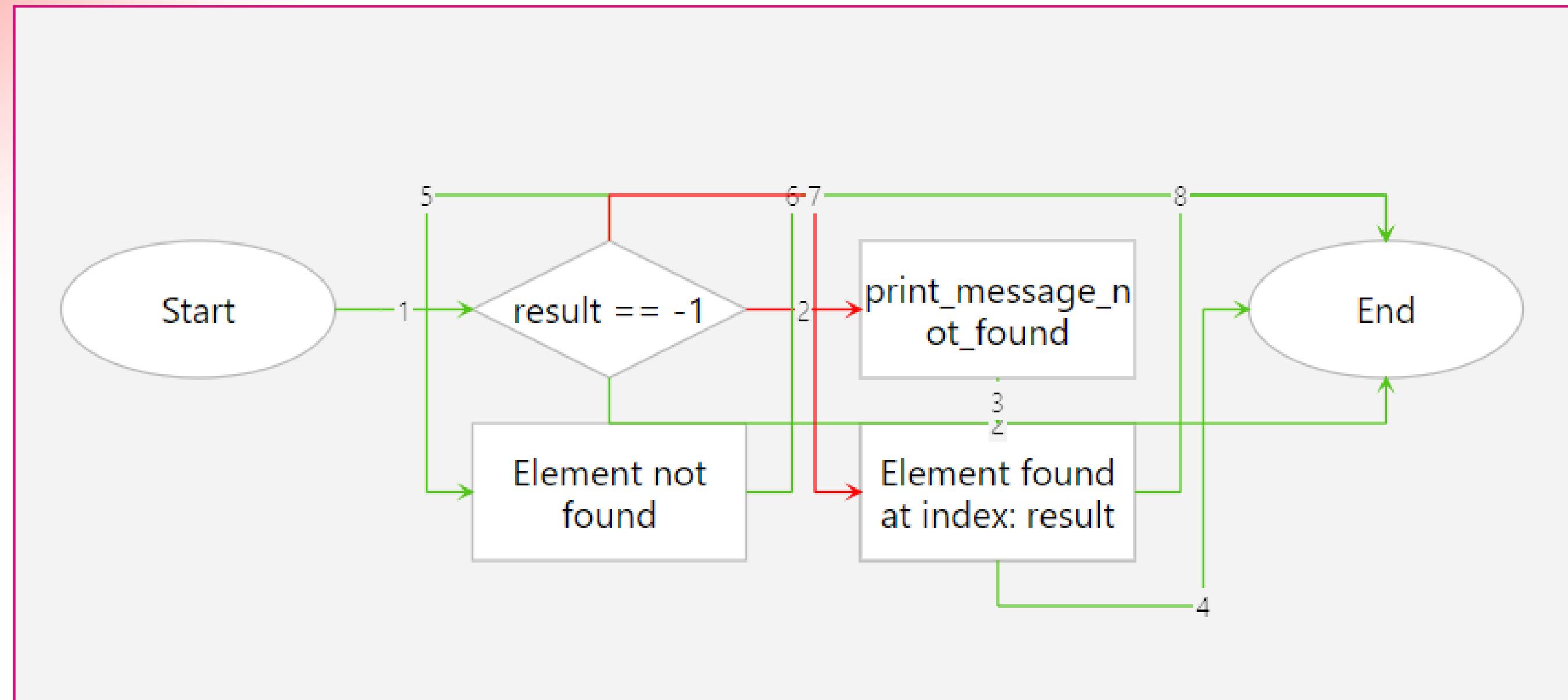


Disclaimer: This tool is **designed for students** who get stuck at some point and **immediately** want the explanation for the code

# Flow-Chart Generation



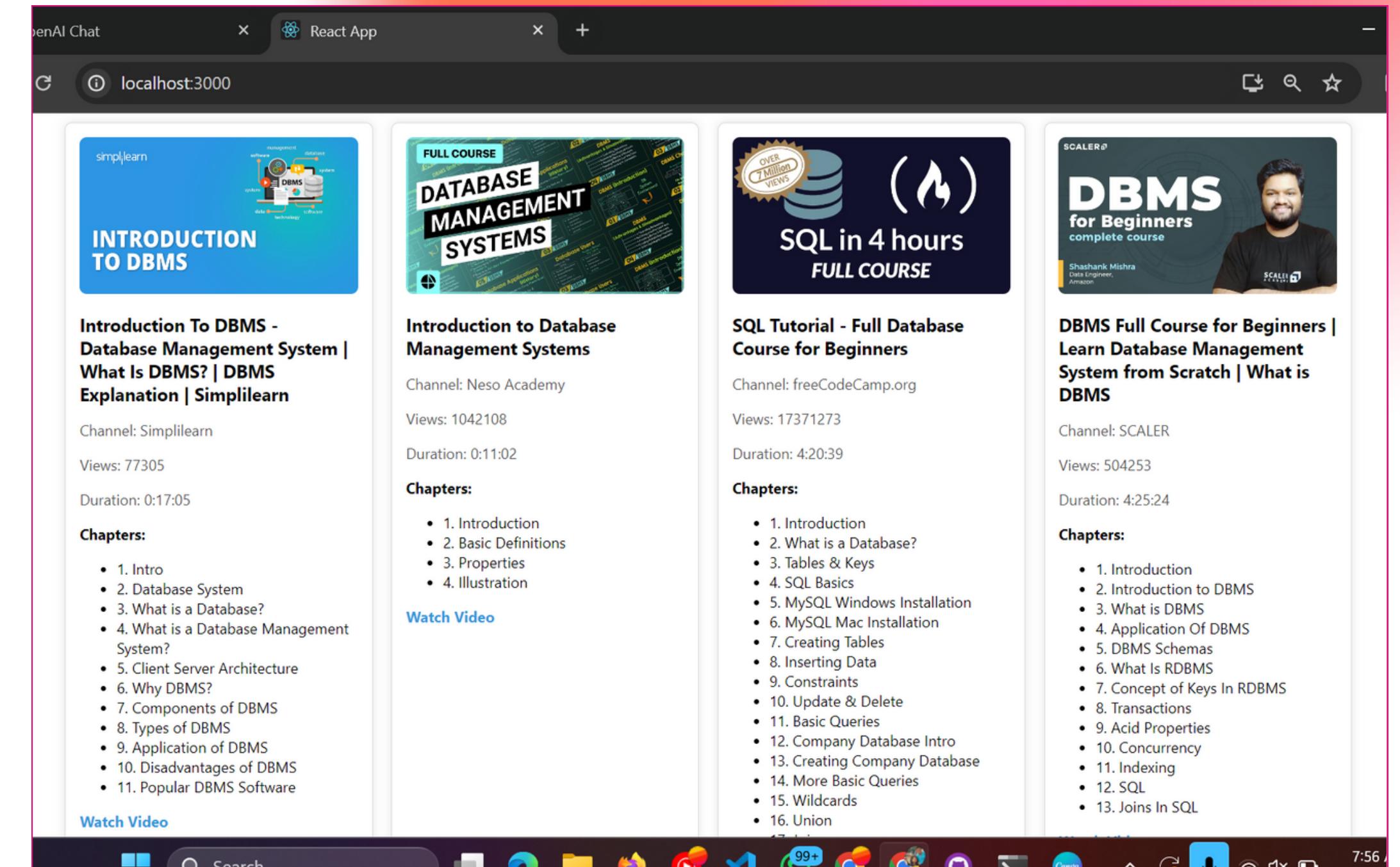
We offer **flowchart generation** from code, aiding developers in visualizing code logic and structure.



Disclaimer: you **JUST** have to input the schema: and you get a very visually appealing flowchart out of it

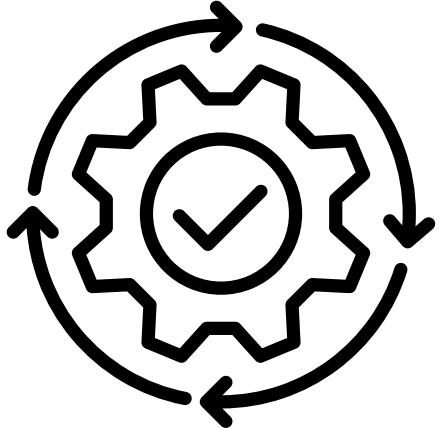
# Youtube Recommendations

Tension\_Code offers tailored educational resources, including [YouTube recommendations](#) for code-related content, empowering developers to continuously enhance their skills.

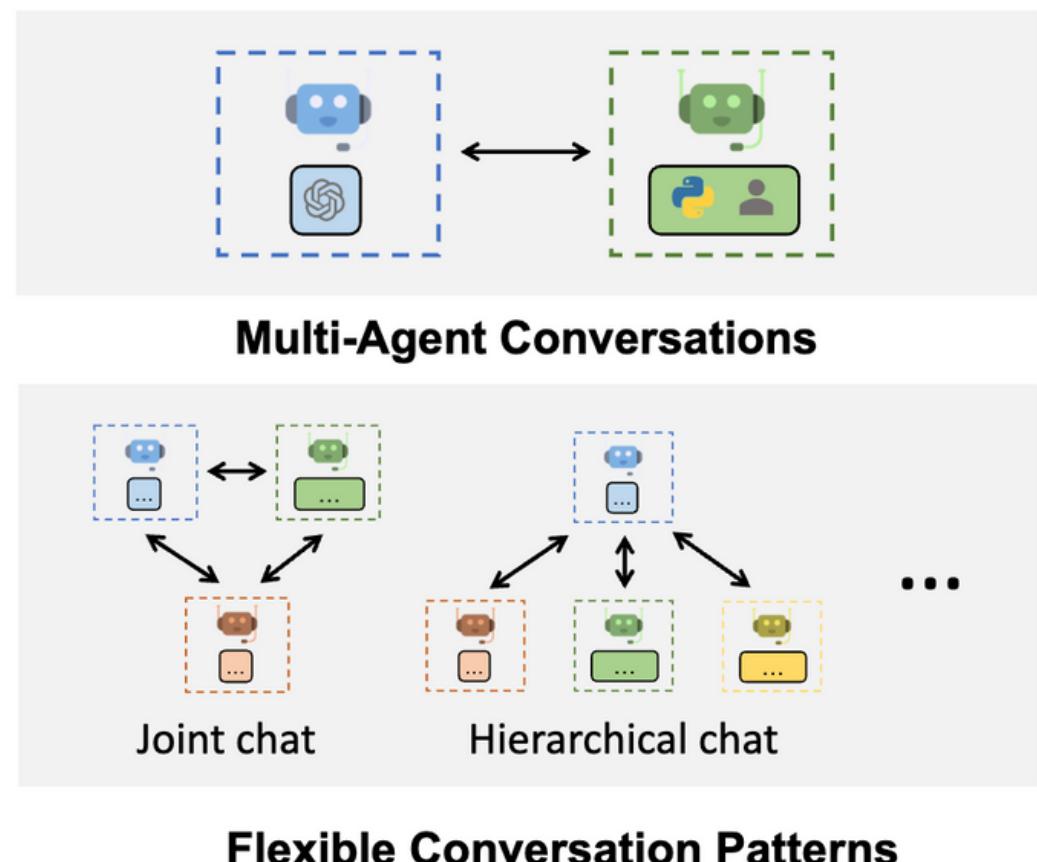
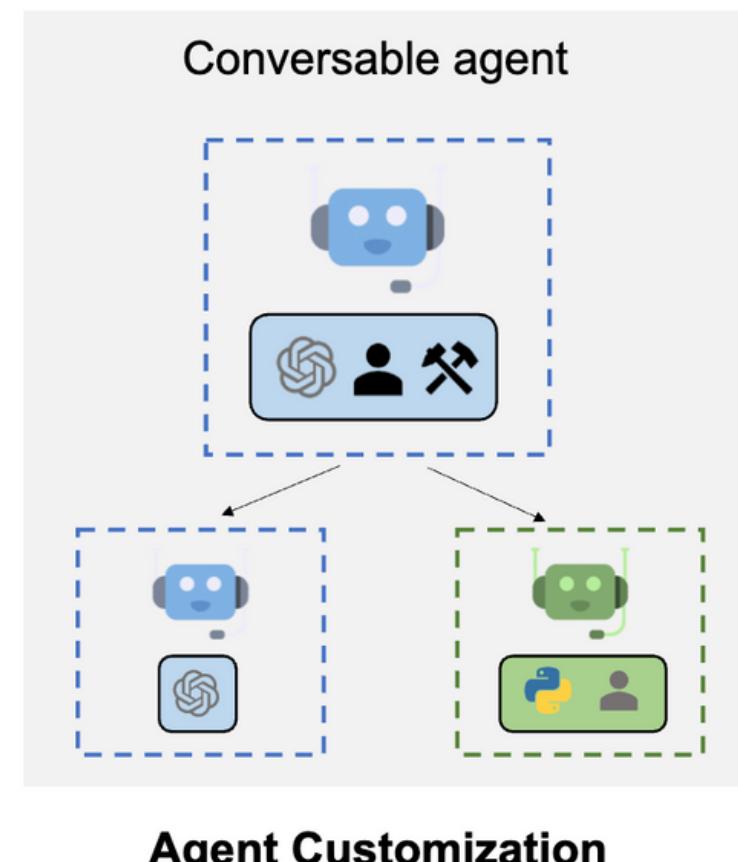


Disclaimer: This tool is **designed for students** who want guided lectures and courses to follow for the same

# Our USPs: Automated Error Handling, Debugging, and Feedback



Utilizing reinforcement learning and **Microsoft's open-source LLM manipulation** agent, **AutoGen**, we aim to seamlessly integrate error handling and debugging mechanisms into code generation. By automating error detection and soliciting user feedback, our system ensures robust, reliable code while continuously refining its performance by **optimising** the code to its **best time and space complexity**.





# Our USPs: Visual Studio Code Integration

Unleash the full potential of our tool with our **Visual Studio Code extension**. Seamlessly leverage advanced capabilities directly within your preferred IDE. Streamline workflows, boost productivity, and empower your coding journey—all effortlessly **integrated into VS Code**. Elevate your development experience today.

