📘 Lecture 4: BUILD AI Agents :

🔷 AI Agent क्या होता है?

AI Agent एक ऐसा smart system होता है जो:

- User की query को समझता है
- उसे structured format में बदलता है
- फिर proper tool/function को call करता है
- और output user को देता है

> 🧠 **AI Agent खुद calculations या logic execute नहीं करता** — वो predefined tools या functions से काम करवाता है।

🔷 Common Problems जो AI Agent Solve कर सकता है:

✅ 1. Sum of Two Numbers

```
function sum(num1, num2) {
  return num1 + num2;
}
```

📝 **Use Case:**
"7 और 5 का जोड़ बताओ" → Output: 12

✅ 2. Prime Number Checker

```
function prime(num) {
  if (num < 2) return false;
  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) return false;
```

```
  }

  return true;

}
```

📝 **Use Case:**
"11 prime है या नहीं?" → Output: `true`

✅ 3. Cryptocurrency Price Checker (API से)

```
async function cryptoPrice(coin) {

  const response = await
fetch(`https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&
ids=${coin}`);

  const data = await response.json();

  return data;

}
```

📝 **Use Case:**
"Bitcoin की current price बताओ"

⏳ 4. Weather Checker (Future Scope)

```
function weather(city) {

  // TODO: Add weather API

}
```

📝 **Use Case:**
"Delhi का मौसम कैसा है?" (To be implemented)

🔷 LLM (Like ChatGPT or Gemini) कैसे काम करता है?

🧠 Step-by-Step Process:

1. **User** बोलता है:
   "7 और 8 का जोड़ क्या है?"

2. **LLM** उस query को structured format में बदलता है:

```
{
  "name": "sum",
  "args": {
    "num1": 7,
    "num2": 8
  }
}
```

3. AI Agent इसे sum function को भेजता है → `sum(7, 8)` → Output: `15`

4. Output user को show किया जाता है

🔷 LLM की Limitations (Boundaries):

❌ Limitation	🔍 Explanation
खुद कोई code नहीं चलाता	Execution external function से होता है
Calculation नहीं करता	Functions से result generate करता है
API या DB access नहीं होता	Tool access ज़रूरी है
नया function दिया तो fail हो सकता है	इसलिए predefined होना ज़रूरी है

🔷 Function Call Format (Structured Form):

🕵️ User Query	🧠 LLM Output
"7 और 8 का जोड़ बताओ"	`{ name: "sum", args: { num1: 7, num2: 8 } }`
"11 prime है?"	`{ name: "prime", args: { num: 11 } }`
"Bitcoin की price बताओ"	`{ name: "cryptoPrice", args: { coin: "bitcoin" } }`

🔷 AI Agent Working Architecture:

```
User Input (Natural Language)
          ↓
LLM (ChatGPT, Gemini, etc.)
          ↓
Structured Format (JSON Style)
          ↓
Tool/Function Call (sum, prime, etc.)
          ↓
Output from Function
          ↓
Final Answer to User
```

🔷 Important Keywords & Concepts:

Keyword	Meaning

`name:`	Function का नाम जो call करना है
`args:`	Function के लिए input parameters
`functionDeclarations`	वो सभी tools जिनका access LLM को है
`systemInstructions`	LLM को guide करने वाले rules
`History`	पुरानी queries और responses की list

✅ Final Summary: Lecture 4

🔑 Concept	✅ Explanation
AI Agent	User की query को समझकर tool को call करता है
LLM	Decide करता है कि कौन सा function call होना चाहिए
Functions	Actual काम करते हैं (जैसे sum, prime)
LLM Execution नहीं करता	वो सिर्फ logic decide करता है
Input Format	Natural language
Output Format	Structured JSON (`name`, `args`)
Tools Needed	sum, prime, cryptoPrice

Must be Declared	Functions को पहले से declare करना ज़रूरी है

💡 Bonus Tip: खुद का AI Agent बनाना चाहते हो?

तो ये steps follow करो:

1. Functions को define करो (`sum`, `prime`, etc.)
2. Gemini/ChatGPT API को integrate करो
3. LLM को instruct करो कौन-कौन से tools available हैं
4. Structured format में output को handle करो
5. हर query के लिए response return करो

📘 Build Your Own AI Agent – Code Explanation (With Comments)

✅ Step 1: Required Imports & Initial Setup

```
import { GoogleGenAI } from "@google/genai";            // Gemini LLM को access करने के लिए

import readlineSync from 'readline-sync';               // CLI से user input लेने के लिए


const History = [];                                     // Chat history को store करता है

const ai = new GoogleGenAI({

    apiKey: "YOUR_API_KEY_HERE"                         // Gemini API key (replace with your own)

});
```

✅ Step 2: Function Definitions (Tools for LLM)

🔷 Sum Function

```
function sum({ num1, num2 }) {
    return num1 + num2;
}
```

दो numbers का जोड़ करता है

🔷 Prime Checker

```
function prime({ num }) {
    if (num < 2) return false;
    for (let i = 2; i <= Math.sqrt(num); i++) {
        if (num % i === 0) return false;
    }
    return true;
}
```

बताता है कि number prime है या नहीं

🔷 Crypto Price (Using API)

```
async function getCryptoPrice({ coin }) {
    const response = await
fetch(`https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&
ids=${coin}`);
    const data = await response.json();
    return data;
}
```

किसी भी crypto coin की live price लाता है (CoinGecko API से)

✅ Step 3: Tool Declarations (LLM को बताना tools के बारे में)

🔷 Sum Declaration

```
const sumDeclaration = {
    name: 'sum',
    description: "Get the sum of 2 numbers",
    parameters: {
        type: 'OBJECT',
        properties: {
            num1: { type: 'NUMBER', description: 'First number' },
            num2: { type: 'NUMBER', description: 'Second number' }
        },
        required: ['num1', 'num2']
    }
};
```

🔷 Prime Declaration

```
const primeDeclaration = {
    name: 'prime',
    description: "Check if number is prime",
    parameters: {
        type: 'OBJECT',
        properties: {
            num: { type: 'NUMBER', description: 'Number to check' }
        },
        required: ['num']
    }
};
```

🔷 Crypto Price Declaration

```
const cryptoDeclaration = {
    name: 'getCryptoPrice',
    description: "Get current price of any crypto currency",
    parameters: {
        type: 'OBJECT',
        properties: {
            coin: { type: 'STRING', description: 'Name of the coin
(e.g., bitcoin)' }
        },
        required: ['coin']
    }
};
```

✅ Step 4: Register Available Tools

```
const availableTools = {
    sum,
    prime,
    getCryptoPrice
};
```

AI Agent को बताता है कि वो किन-किन functions को call कर सकता है

✅ Step 5: Run the AI Agent Logic

```
async function runAgent(userProblem) {
```

```
    History.push({ role: 'user', parts: [{ text: userProblem }] }); // User query को history में add करो



    while (true) {

        const response = await ai.models.generateContent({

            model: "gemini-2.5-flash",                // Gemini का model use किया गया है

            contents: History,                        // पुरानी conversation भेजी जा रही है

            config: {

                systemInstruction: `You are an AI Agent...`,   // Instructions LLM को

                tools: [{

                    functionDeclarations: [sumDeclaration, primeDeclaration, cryptoDeclaration]

                }]

            },

        });



        // अगर LLM ने किसी function को call किया है

        if (response.functionCalls && response.functionCalls.length > 0) {

            const { name, args } = response.functionCalls[0];   // Function name और arguments निकाले



            const funCall = availableTools[name];               // Function को access किया

            const result = await funCall(args);                 // Function को run किया



            const functionResponsePart = {
```

```
                name,
                response: { result }
            };

            // LLM ने जो function call किया वो भी history में जोड़ दो
            History.push({
                role: "model",
                parts: [{ functionCall: response.functionCalls[0] }]
            });

            // जो result आया उसे भी history में जोड़ दो
            History.push({
                role: "user",
                parts: [{ functionResponse: functionResponsePart }]
            });

        } else {
            // अगर function call नहीं है, तो normally text response दो
            History.push({ role: 'model', parts: [{ text:
response.text }] });
            console.log(response.text);
            break;
        }
    }
}
```

✅ Step 6: Main Function – Loop for User Input

```
async function main() {

    const userProblem = readlineSync.question("Ask me anything--> ");
// CLI से user input

    await runAgent(userProblem);
// Agent को input भेजो

    main();
// दुबारा पूछो

}


main();
```

✅ Summary of AI Agent Flow

🔢 Step	🔍 Action
1	User से input लिया जाता है
2	Gemini LLM input को structured format (JSON) में convert करता है
3	सही function को call करने का decision लेता है
4	Output function से लेता है
5	Output को वापस user को दिखाता है
🔁	Process दोबारा चलता है (loop में)

🧠 Example:

🗣️ **User:** "13 prime है क्या?"
🧠 **LLM Converts:**

```
{
  "name": "prime",
  "args": { "num": 13 }
}
```

🔧 **Call:** `prime({ num: 13 })`
📤 **Output:** `true`