Building an Agentic AI Finance Assistant Using Amazon Bedrock and Meta LLaMA

🚀 Overview

In this blog, I'll walk you through the journey of building an Agentic AI Proof of Concept (POC) using **Amazon Bedrock**, **Meta's LLaMA model**, **AWS Lambda**, and a **Knowledge Base** powered by **Titan Embeddings**. The goal was to combine Generative AI with real-time logic execution for a finance-related assistant.

The result? An AI assistant that can:

- Answer tax-related questions using a Knowledge Base
- Calculate income tax using Lambda tools
- Dynamically respond based on user input and intent

🎯 Objective

The idea was to build a conversational assistant that:

- Responds to queries like "What are the income tax slabs for FY 2023–24?"
- Calculates income tax based on a given income value
- Leverages both Retrieval-Augmented Generation (RAG) and Lambda-based action tools

---

🧱 Architecture Components

- **Amazon Bedrock Agent** with Meta's LLaMA model
- **Knowledge Base (KB)** using Titan embedding model
- **Lambda Function** for tax calculation
- **S3** to store tax-related PDFs and financial documents
- **Agent Action Group** configured with function schema

🛠️ Step-by-Step Implementation

1. Create and Configure the Agent

- Chose **Meta LLaMA** as the foundational model
    - Defined system instructions:

      If a user asks a question related to finance or taxes:
        - First check the Knowledge Base for relevant information.
        - If the query involves tax calculation, invoke the appropriate tool.

2. Set Up the Knowledge Base

- Uploaded PDF documents about Indian tax slabs, 80C, PPF, etc. to an S3 bucket
- Created a Knowledge Base in Bedrock
- Used **Titan Embed Text v1** for chunking and vectorization
- Attached the KB to the Agent

3. Create the Lambda Tool

- Lambda named `CalculateTaxLambda`
- Function parameters:
    - `income`: number, required
- Input schema set in Action Group
- Logic handles Indian income tax slab calculation for FY 2023–24

4. Tool Response Format (Critical!)

To make Lambda responses compatible with Bedrock Agent:

```
{
  "messageVersion": "1.0",
  "response": {
    "actionGroup": "taxcalculator",
    "function": "taxcal",
    "functionResponse": {
      "responseBody": {
        "TEXT": {
          "body": "Your estimated tax is Rs. 72,500."
        }
      }
    }
  }
}
```

5. Agent Behavior

- For "What is my tax for 8 lakh?" → Extracts parameter and calls Lambda
- For "What is Section 80C?" → Searches KB and returns document-based response

---

🧪 Testing the POC

✅ Lambda Tool Test

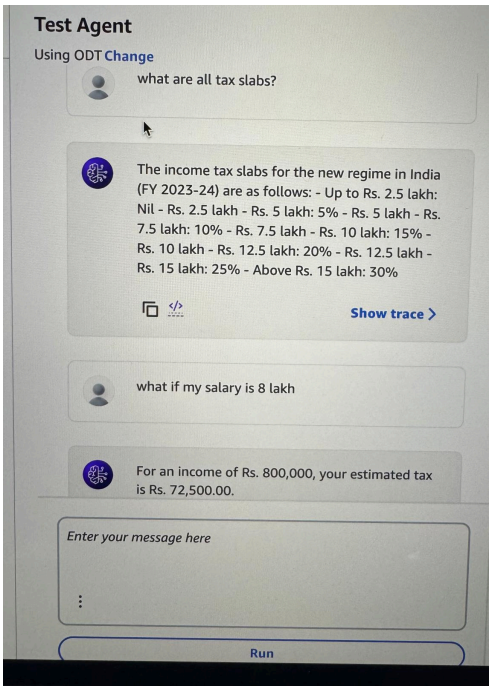
- Input: `{ "income": 800000 }`
- Output: `Your estimated tax is Rs. 72,500.`

✅ KB Search Test

- Input: "What are the slabs under new regime?"
- Output: Extracted from `tax_slabs.pdf`

🧠 Learning Curve

- **toolResult: null** → fixed by using proper `messageVersion: 1.0` format
- **Incorrect event parsing** → fixed by extracting parameters from list format
- **Action not found** → solved by verifying Action Group name and function mapping



📌 Summary

This POC combines:

- Agentic AI with Meta's LLaMA
- Real-time action execution via Lambda
- Retrieval-based intelligence via KB + Titan embeddings

It's a powerful blueprint for building domain-specific, intelligent assistants that are not only chatty but also capable.