

# **BankBot AI**



## **Chatbot Smart Conversational Banking System**

Milestone 1 + Milestone 2 + Milestone 3 + Milestone 4

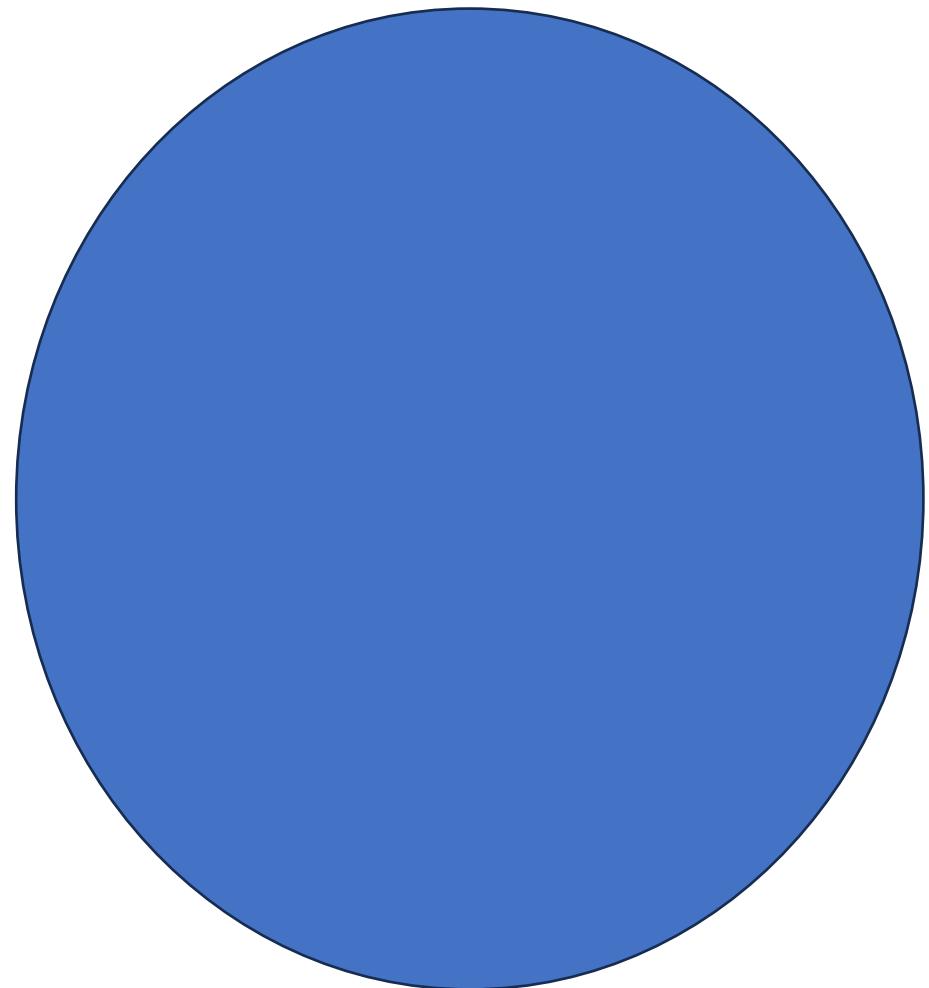
### **Problem :**

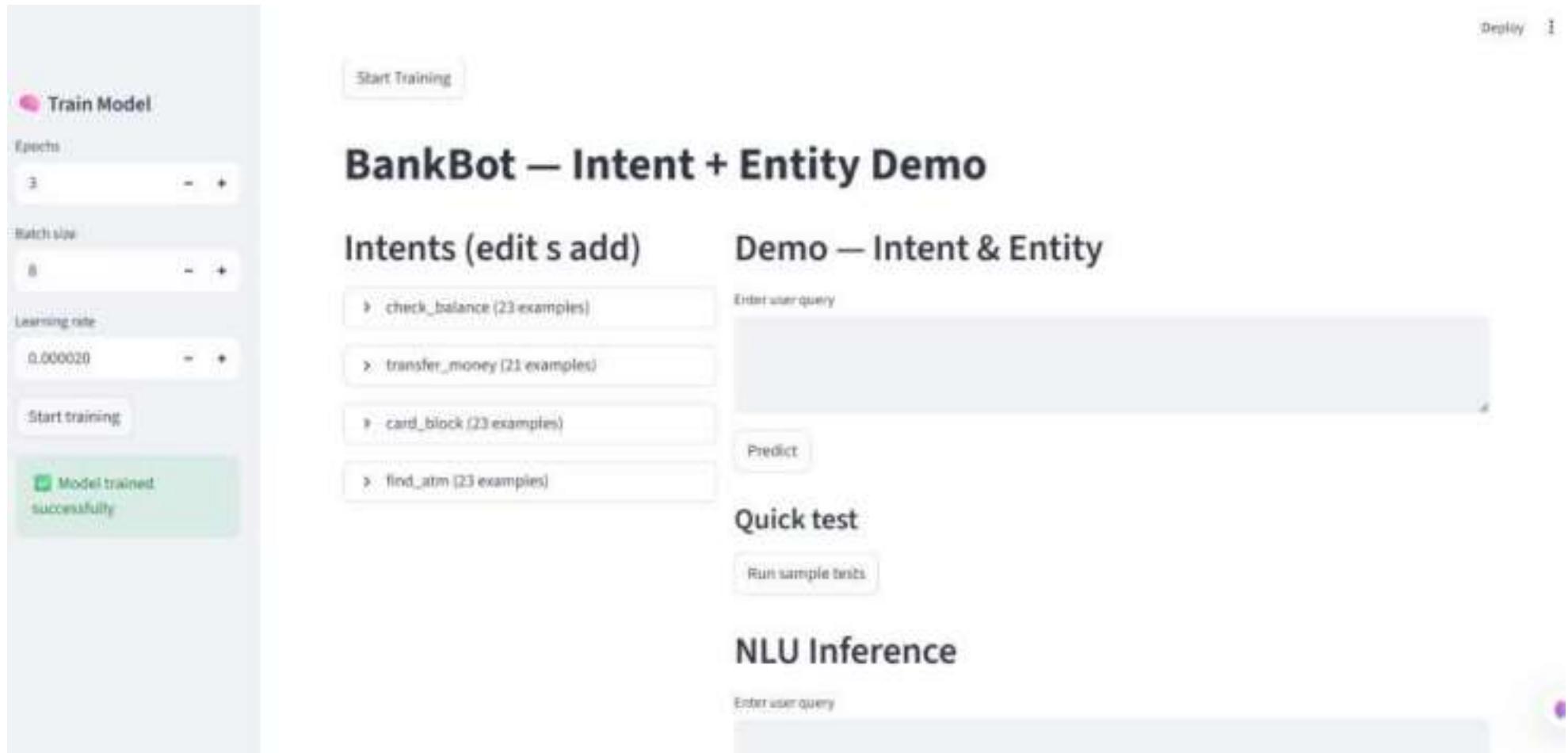
Customers often have recurring banking queries related to account balance, interest rates, loan eligibility, transaction status, branch details, etc. Relying on manual customer service teams leads

to delays and high operational costs. This project aims to build an intelligent chatbot that can handle these questions through natural conversations. It will support both rule-based and machine learning approaches to ensure accuracy and scalability. The chatbot will be accessible via web interface and possibly integrated into mobile apps or WhatsApp.

## Milestone 1

- This is mainly revolve around finding entity and intent recognition .
- Handles intents like balance check, loan inquiry, card blocking, branch locator, and more.
- Interference using streamlit .
- Backend using SQlite .
- 
- 



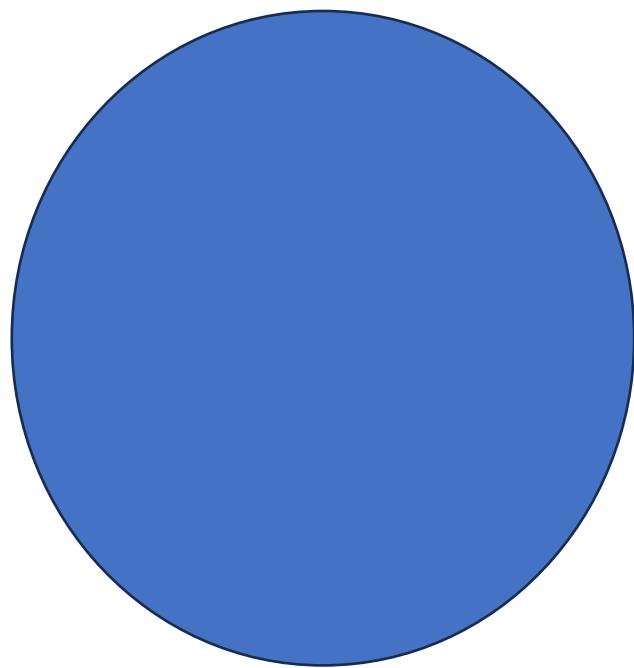


The screenshot shows a Streamlit application for a chatbot named "BankBot – Intent + Entity Demo". On the left, there's a sidebar titled "Train Model" with sliders for "Epochs" (set to 3), "Batch size" (set to 8), and "Learning rate" (set to 0.000020). Below these is a "Start training" button, and a green success message: "Model trained successfully". At the top center is a "Start Training" button. The main area has three sections: "Intents (edit | add)" listing "check\_balance (23 examples)", "transfer\_money (23 examples)", "card\_block (23 examples)", and "find\_atm (23 examples)"; a "Demo – Intent & Entity" section with an "Enter user query" input field and a "Predict" button; and an "NLU Inference" section with an "Enter user query" input field. There are also "Quick test" and "Run sample tests" buttons.

## Milestone 2:

- This is Ai powered chatbot , which uses grog api and streamlit , llm , etc for the interactive interface

- This has a responsive UI with option to create user , login the with the same account
- And there is Chatbot Which is capable to tell details 1)about user ,  
2) type of card ,  
3)amount in account ,  
4)transaction between various user
- This also maintain a Database in the backend using SQLite which contain  
1)user name ,  
2)password(encrypted form ),  
3)type of account ,  
4)transaction,  
5)amount in account



## Problem :

- 1) Traditional banking systems: · 2)Time-consuming manual processes · 3)Limited working hours
- 4)No instant account support 5)Difficult for non-technical users

Output:

[main app](#)[Chatbot](#)[Create Account](#)[Logout](#)

## AI Banking Chatbot

Type here

Send



The screenshot shows a user interface for creating a new bank account. On the left, there's a sidebar with links: 'main app', 'Dashboard', 'Create Account' (which is highlighted in grey), and 'Login'. The main area has a purple header with the title 'Create New Bank Account' and a small bank icon. Below the header is a large white input field. The form fields are as follows:

- Name:** An input field with placeholder text.
- Email Address:** An input field with placeholder text.
- Login Password:** An input field with placeholder text.
- Confirm Password:** A password strength meter consisting of three bars.
- Mobile Number:** An input field with placeholder text.
- Account Type:** A dropdown menu showing 'Savings'.
- Initial Balance:** An input field containing '50000.00' with a minus sign and a plus sign to its right.
- Account Payment:** An input field with placeholder text.

At the bottom is a red button labeled 'Create Account' with a green checkmark icon.

### Milestone 3:

- Milestone is an extended version of the milestone 2 , milestone 2 can

Answer only question related to transfer, cards , account but the milestone 3 can answer the general question , question of the various field mainly

- This use Technology such as Groq api , local llm ,

- This also uses local llm
- This is Domain-aware AI chatbot now earlier it was simple Q&A bot
- LLM generic answer

Problem Faced :

- Simple Q&A bot vs Domain-aware chatbot difference
- Current project me **exact problems**
- `llm_local.py` & `llm_gorg.py` ke context me issue
- Missing **domain classification & routing logic**
- Clean **architecture explanation**

Technology Used:

### 1. LLM Local (`llm_local.py`)

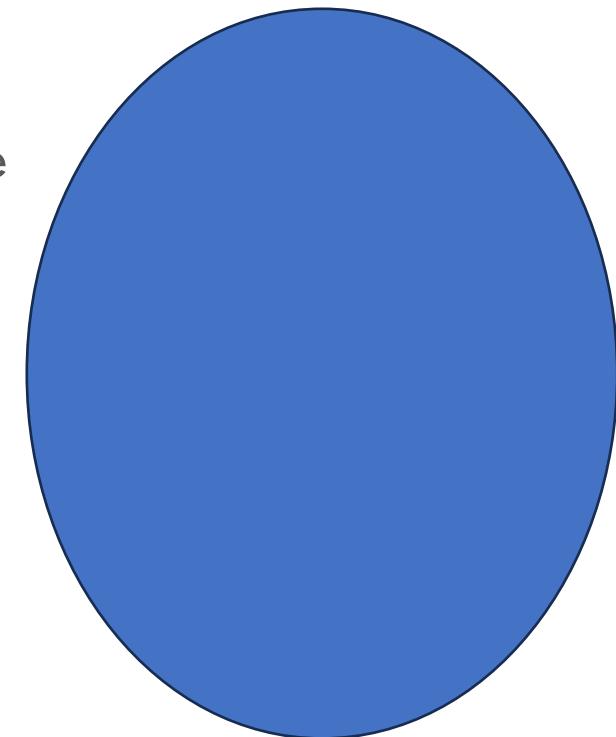
**Definition:**

LLM Local is a **domain-specific intelligence layer** designed to answer questions using **local and structured data sources**.

**Data Sources**

- Local databases (MySQL / SQLite)
- CSV files (e.g., `iris.csv`)
- Internal project-specific data

**How it Works**



- The user query is processed against local data
- Responses are generated using database queries, embeddings, or rule-based logic • It does not rely on external APIs or internet-based knowledge

### **Strengths**

- High accuracy for internal data
- Fast response time
- Data privacy and security
- Ideal for banking, internal tools, and structured datasets

### **Limitations**

- Cannot answer questions outside the available data
- Not suitable for general AI/ML theory or conceptual explanations

### **Example Queries**

- “What is the balance of a customer?”
- “How many features are in the Iris dataset?”
- “Show the last transaction details”

## **2. LLM Groq / General LLM (llm\_gorg.py)**

### **Definition:**

LLM Groq is a **general-purpose language model** used for answering **theoretical and conceptual questions** related to domains like Data Science, Machine Learning, and Deep Learning.

### **Data Source**

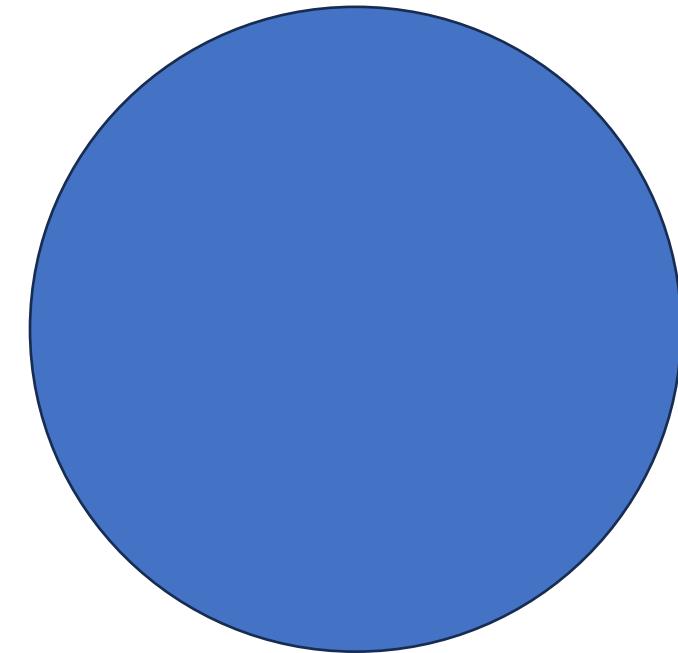
- Pre-trained large language models (Groq / GPT / HuggingFace)
- Trained on large-scale, general-purpose data

### **How it Works**

- Understands the semantic meaning of the question
- Generates explanations using reasoning and language understanding .  
Does not depend on local project data

### **Strengths**

- Explains complex concepts clearly
- Handles unseen and open-ended questions



- Suitable for learning, theory, and comparisons

## Limitations

- Answers can be generic
- May produce hallucinated responses
- No access to internal databases

## Example Queries

- “What is backpropagation in neural networks?”
- “Difference between supervised and unsupervised learning”
- “Explain overfitting with an example”

Output:

The screenshot shows a web-based AI Banking Assistant. On the left, a sidebar menu includes 'main app', 'Chatbot' (which is selected and highlighted in blue), 'Create Account', and 'Logout'. The main content area has a purple header with the title 'AI Banking Assistant' and a subtitle 'Ask something...'. Below this is a white input field containing the text 'What is Deep learning?'. A 'Send' button is located below the input field. The response section starts with 'Your query is Deep learning.' followed by a heading 'Ask: Deep Learning: An Overview'. The text explains that Deep learning is a subset of machine learning involving artificial neural networks trained to analyze and interpret data. It then lists 'Key Characteristics of Deep Learning':

1. Multi-layered architecture: Deep learning models consist of multiple layers of interconnected nodes, allowing them to learn complex patterns and relationships in data.
2. Neural networks: Deep learning models are based on artificial neural networks, which are inspired by the structure and function of the human brain.

## Milestone 4:

So the milestone four revolves around taking user intent query and then classifying the same based the on the intent as

- 1)Transfer money ,
- 2)Card block ,
- 3)Find Atm
- 4)Check balance

To obtain pie chat based on the queries comparison , could be better understood by output.

This has an option to count the type of the question based on the query and can also export the question count list as per the requirement

Editorial could also be trained .

We can export the file.

## Technology Used:

- Python
- LangChain
- Groq LLM
- Local LLaMA (GGUF)
- SQLite
- Streamlit

## Problem Faced:

### **Intent Misclassification**

Some user queries were ambiguous and matched multiple intents, which led to incorrect classification, especially when queries contained overlapping keywords.

### **Confidence Score Variations**

Similar queries sometimes produced different confidence scores, making it challenging to define a fixed threshold for accurate intent detection.

### **Limited Training Data**

Initial training examples were limited, which affected the accuracy of intent prediction and required frequent updates to the editorial/training data.

### **Visualization Handling**

Managing pie chart and bar chart rendering dynamically in Streamlit based on user selection required careful state and layout handling.

### **Performance Issues with Local LLaMA**

Running the GGUF-based local LLaMA model resulted in higher latency compared to cloud-based models, especially during repeated query analysis.

### **Data Persistence and Logging**

Maintaining consistent query logs in SQLite and syncing them with real-time analytics dashboards required additional handling.

### **Model Switching Complexity**

Supporting both Groq LLM and local LLaMA increased complexity in configuration and response standardization.

## AI Banking Assistant

Go to

- Chatbot
- Admin Analytics
- Logout

## BankBot Admin

Navigation

- Chat Analytics
- Query Analytics
- Training Editor
- Export Logs

## Test Query

Enter user query

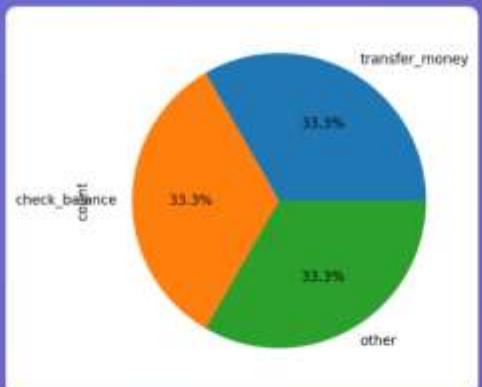
how are you

Analyze Query

# Chat Analytics

Select intent:

- overall
- check\_balance
- transfer\_money
- card\_block
- find\_atm



	question	intent	timestamp
0	transfer 100	transfer_money	2026-01-11 23:59:59
1	check my acc	check_balance	2026-01-11 23:59:59
2	how are you	other	2026-01-11 23:59:59

## AI Banking Assistant

Go to

- Chatbot
- Admin Analytics
- Logout

## BankBot Admin

Navigation

- Chat Analytics
- Query Analytics
- Training Editor
- Export Logs

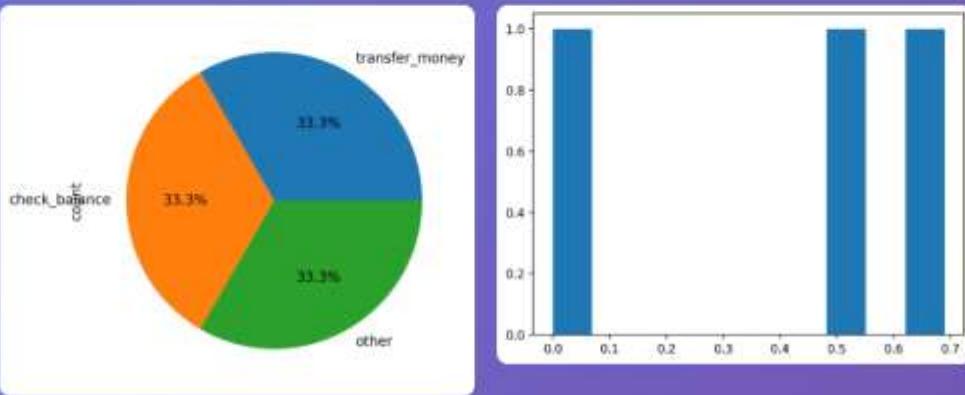
## Test Query

Enter user query

how are you

Analyze Query

## Query Analytics



	question	intent	confidence	timestamp
2	how are you	other	0	2026-01-11 23:24:39
1	check my account	check_balance	0.5	2026-01-11 23:24:32
0	transfer 100	transfer_money	0.69	2026-01-11 23:23:19

Thank You 