TELECOM FINAL PROJECT DOCUMENTATION TELECOM ASSISTANT

G ANIRUDH(81383)

ABSTRACT:

This project presents Telecom Assistant, a modular AI-driven system designed to handle customer inquiries related to billing, network troubleshooting, service recommendations, and general knowledge retrieval. The assistant integrates multiple AI frameworks—CrewAI, AutoGen, LangChain, LlamaIndex, and LangGraph—to intelligently classify and route user queries to the most suitable processing engine.

PROJECT STRUCTURE:

```
telecom_assistant/
├── app.py                  # Main entry point to run the Streamlit app
├── requirements.txt        # Project dependencies
├── data/
│   ├── telecom.db          # SQLite database for telecom data
│   └── documents/          # Folder for knowledge base documents
│       ├── service_plans.md
│       ├── network_guide.md
│       ├── billing_faq.md
│       └── technical_support.md
├── config/
│   └── config.py           # Configuration settings
├── agents/
│   ├── __init__.py
│   ├── billing_agents.py   # CrewAI implementation
│   ├── network_agents.py   # AutoGen implementation
│   ├── service_agents.py   # LangChain implementation
│   └── knowledge_agents.py      # LlamaIndex implementation
├── orchestration/
│   ├── __init__.py
│   ├── graph.py                # LangGraph orchestration
│   └── state.py                # State management
├── ui/
│   ├── __init__.py
│   └── streamlit_app.py        # Streamlit UI code
└── utils/
    ├── __init__.py
    ├── database.py             # Database utilities
    └── document_loader.py      # Document loading utilities
```

SETUP INSTRUCTIONS:

PREREQUISITES:

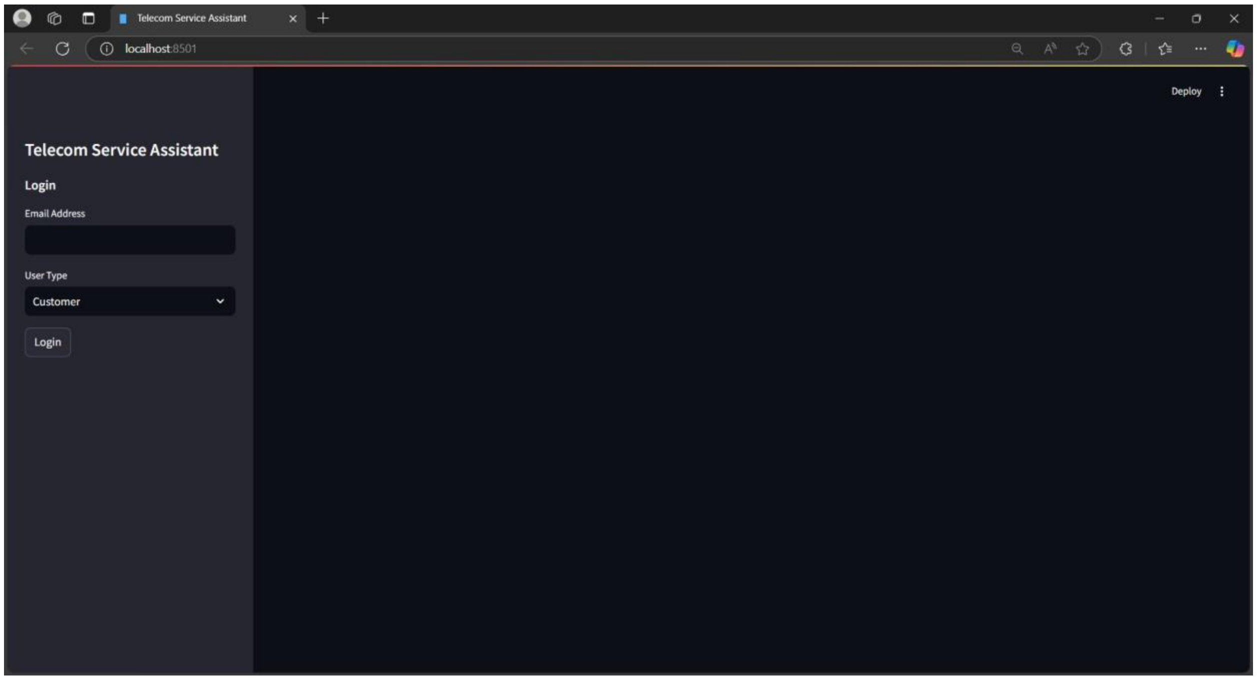
1. Python 3.8 or higher.
2. Poetry (for dependency management)
3. Navigate to the project directory.
4. Install dependencies using Poetry:
   ```
   poetry install
   ```
5. Ensure you have the SQLite database (telecom.db) and knowledge base documents in the data/ folder.

RUNNING THE APPLICATION:

1. Activate the Poetry environment:
   ```
   poetry shell
   ```
2. Run the Streamlit app:
   ```
   poetry run streamlit run app.py
   ```
3. Access the application in your browser at http://localhost:8501

FRONT-END IMPLEMENTATION:

The frontend is built using Streamlit, providing an interactive interface for users



LOGIN PAGE

ADMIN DASHBOARD:

KNOWLEDGE-BASED MANAGEMENT:

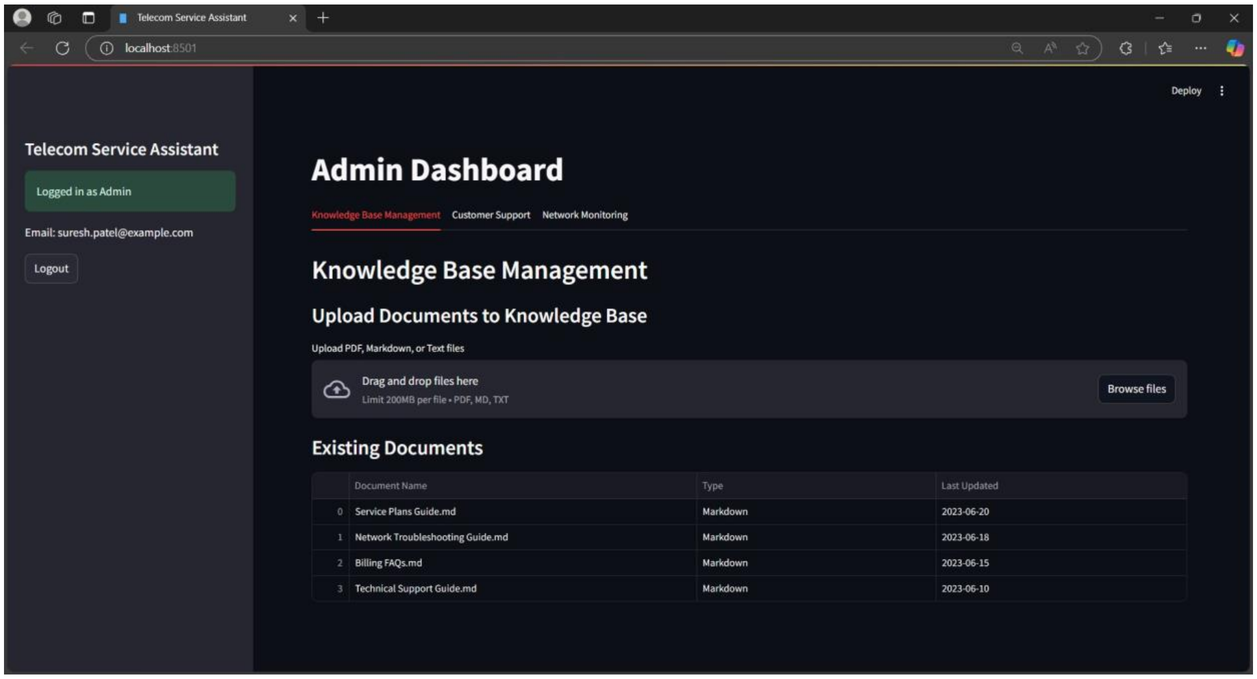
• Document Upload

- Administrators can upload multiple files simultaneously.
- Supported file types: PDF, Markdown, and Text files.
- Each uploaded file is processed and added to the knowledge base. ● A success message confirms the addition of each file.

• Document Processing

- The document_loader() function handles the backend processing of uploaded documents:
- Uses SimpleDirectoryReader to load documents from a specified directory.
- Document ProcessingTUses, SimpleDirectoryReader to load documents from a
- specified directory.
- Employs OpenAIEmbedding for creating embeddings of the document content.

- Utilizes ChromaVectorStore for efficient storage and retrieval of document vectors



Telecom Service Assistant

Logged in as Admin

Email: suresh.patel@example.com

Logout

Admin Dashboard

Knowledge Base Management | Customer Support | Network Monitoring

Knowledge Base Management

Upload Documents to Knowledge Base

Upload PDF, Markdown, or Text files

Drag and drop files here
Limit 200MB per file • PDF, MD, TXT

Browse files

Existing Documents

	Document Name	Type	Last Updated
0	Service Plans Guide.md	Markdown	2023-06-20
1	Network Troubleshooting Guide.md	Markdown	2023-06-18
2	Billing FAQs.md	Markdown	2023-06-15
3	Technical Support Guide.md	Markdown	2023-06-10

CUSTOMER DASHBOARD

1. BILLING AND ACCOUNTING QUERIES (CREW AI)

Handles queries related to **billing and account management**.

• Example Queries:

- *"Why is my bill higher this month?"*
- *"What add-on packs are available for my plan?"*

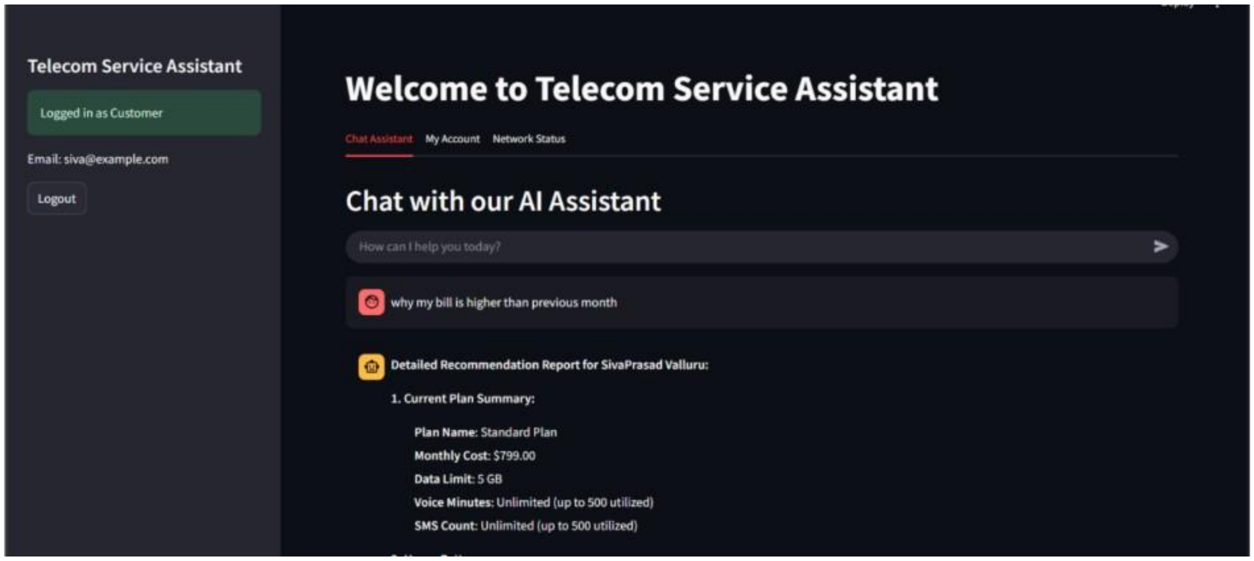
• Flow:

1. **Query classification** by **LangGraph**.
2. Routed to **CrewAI agent** for processing.
3. Response generated by specialized agents like **Billing Specialist** and **Service Advisor**.

• Implementation:

- Located in `agents/billing_agents.py`.

- Uses **SQLDatabaseTool** for accessing billing data.



Welcome to Telecom Service Assistant

Chat with our AI Assistant

why my bill is higher than previous month

Detailed Recommendation Report for SivaPrasad Valluru:

1. Current Plan Summary:

Plan Name: Standard Plan
Monthly Cost: $799.00
Data Limit: 5 GB
Voice Minutes: Unlimited (up to 500 utilized)
SMS Count: Unlimited (up to 500 utilized)

2. NETWORK TROUBLESHOOTING (AUTOGEN)

Assists with resolving **network-related issues**.

• Example Queries:

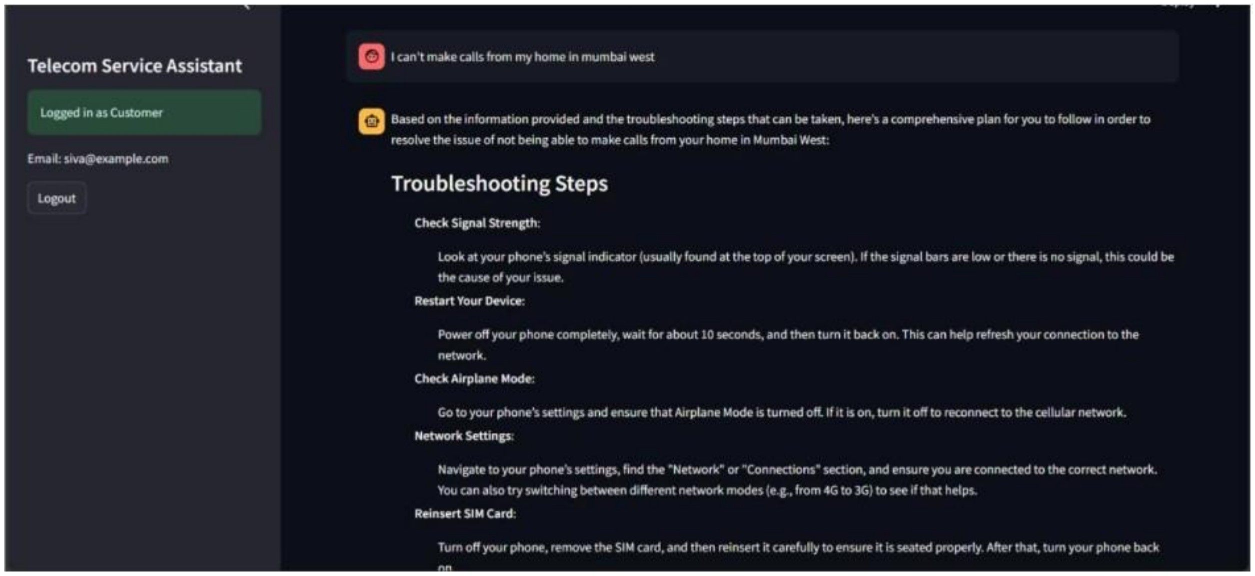
- *"I can't make calls from my home area."*
- *"Why is my 5G speed so slow?"*

• Flow:

1. **Query classification** by **LangGraph**.
2. Routed to **AutoGen agents** like **NetworkDiagnosticsAgent**.
3. **Collaborative troubleshooting** via **GroupChat**.

• Implementation:

- Located in `agents/network_agents.py`.
- Uses both **database and vector store access**.



I can't make calls from my home in mumbai west

Based on the information provided and the troubleshooting steps that can be taken, here's a comprehensive plan for you to follow in order to resolve the issue of not being able to make calls from your home in Mumbai West:

Troubleshooting Steps

Check Signal Strength:

Look at your phone's signal indicator (usually found at the top of your screen). If the signal bars are low or there is no signal, this could be the cause of your issue.

Restart Your Device:

Power off your phone completely, wait for about 10 seconds, and then turn it back on. This can help refresh your connection to the network.

Check Airplane Mode:

Go to your phone's settings and ensure that Airplane Mode is turned off. If it is on, turn it off to reconnect to the cellular network.

Network Settings:

Navigate to your phone's settings, find the "Network" or "Connections" section, and ensure you are connected to the correct network. You can also try switching between different network modes (e.g., from 4G to 3G) to see if that helps.

Reinsert SIM Card:

Turn off your phone, remove the SIM card, and then reinsert it carefully to ensure it is seated properly. After that, turn your phone back on.

3. SERVICE RECOMMENDATION (LANGCHAIN)

Provides **personalized plan recommendations**.

• Example Queries:

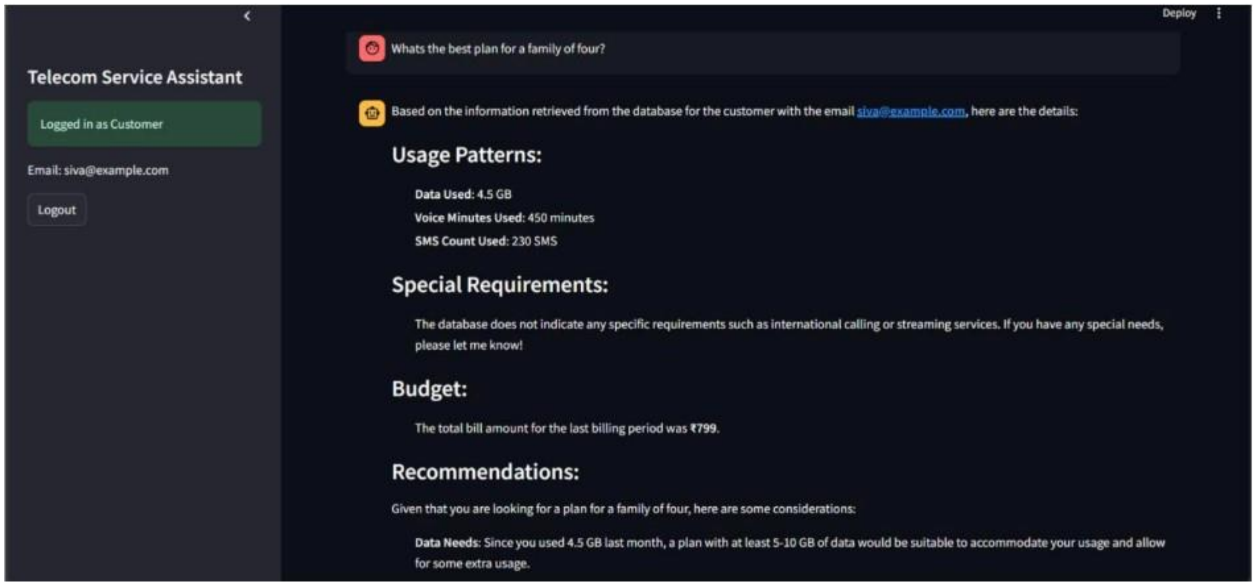
- *"What's the best plan for a family of four?"*
- *"I need a plan with international roaming."*

• Flow:

1. **Query classification** by **LangGraph**.
2. Routed to **LangChain's ReAct agent**.
3. Matches **user requirements** with **available plans**.

• Implementation:

- Located in `agents/service_agents.py`.
- Uses **SQLDatabaseTool** for **plan information retrieval**.



Whats the best plan for a family of four?

Based on the information retrieved from the database for the customer with the email siva@example.com, here are the details:

Usage Patterns:

Data Used: 4.5 GB
Voice Minutes Used: 450 minutes
SMS Count Used: 230 SMS

Special Requirements:

The database does not indicate any specific requirements such as international calling or streaming services. If you have any special needs, please let me know!

Budget:

The total bill amount for the last billing period was **₹799**.

Recommendations:

Given that you are looking for a plan for a family of four, here are some considerations:

Data Needs: Since you used 4.5 GB last month, a plan with at least 5-10 GB of data would be suitable to accommodate your usage and allow for some extra usage.

4. TECHNICAL DOCUMENTATION RETRIEVAL (LLAMA INDEX)

Fetches **information from technical documentation**.

• **Example Queries:**

- *"How do I enable VoLTE on my phone?"* ● *"What are the APN settings for Android?"*

• **Flow:**

1. **Query classification** by **LangGraph**.
2. Routed to **LlamaIndex's RouterQueryEngine**.
3. Retrieves relevant information using **vector search** or **SQL queries**.

• **Implementation:**

- Located in `agents/knowledge_agents.py`.
- Processes documents into **vector embeddings** for **efficient retrieval**.

LANGGRAPH ORCHESTRATION:

The **LangGraph Orchestrator** serves as the **backbone** of query processing, dynamically coordinating between different modules based on the **query type**.

• Workflow Nodes:

1. **Classification Node** – Identifies the **query type** (e.g., billing, network troubleshooting, service recommendations, or knowledge retrieval).
2. **Routing Node** – Directs the query to the appropriate **framework** (CrewAI, AutoGen, LangChain, or LlamaIndex).
3. **Processing Nodes** – Executes **specialized logic** for each query category.
4. **Response Formulation Node** – Compiles intermediate results into a **user-friendly response**.

GRAPH LOGIC:

The **orchestration logic** is implemented in `orchestration/graph.py`, where queries are classified and routed to the appropriate processing nodes.

```
workflow.add_conditional_edges(
   "classify_query",
route_query,
   {
      "crew_ai_node": "crew_ai_node",
      "autogen_node": "autogen_node",
      "langchain_node": "langchain_node",
      "llamaindex_node": "llamaindex_node",
      "fallback_handler": "fallback_handler"
   }
)
```

- **classify_query**: Determines the query category.
- **route_query**: Directs the query to the correct processing node.

- **Processing Nodes**:
  - **CrewAI** → Handles billing and account-related queries.
  - **AutoGen** → Troubleshoots network issues.
  - **LangChain** → Provides personalized service recommendations.
  - **LlamaIndex** → Retrieves technical documentation.
  - **Fallback Handler** → Manages unclassified queries.

CONCLUSION:

The Telecom Service Assistant streamlines customer support by using advanced AI frameworks for accurate query handling, troubleshooting, and service recommendations. Its modular design ensures scalability and adaptability for various telecom needs. By integrating CrewAI, AutoGen, LangChain, and LlamaIndex, it delivers fast, intelligent responses, reducing manual effort and enhancing user experience. This system provides a strong foundation for AI-powered service automation across industries.

CITATIONS:

1. https://python.langchain.com/docs/introduction/

2. https://microsoft.github.io/autogen/stable/

3. https://langchain-ai.github.io/langgraph/tutorials/introduction/

4. https://docs.crewai.com/introduction

5. https://docs.llamaindex.ai/en/stable/api_reference/agent/

6. https://docs.streamlit.io/develop/api-referenc