Standardized Clinical Data (SCD) Chatbot – High Level Documentation

# 1. Introduction

The Standardized Clinical Data (SCD) Chatbot is an AI-powered conversational assistant designed to simplify access to standardized clinical data by allowing users to interact using natural language. Instead of manually writing complex SQL queries or going through raw datasets, users can simply type questions, and the chatbot responds with accurate, real-time insights.  
  
The solution is powered by Generative AI (LLM via AWS Bedrock – Claude 3 Sonnet) and deployed securely within AWS infrastructure. It supports healthcare and life sciences workflows where quick, accurate, and compliant data access is crucial.

## Key Objectives:

* Automate querying of SCD datasets – Reduce manual SQL writing.
* Provide intelligent conversational responses – Make interaction more user-friendly.
* Integrate with AWS services – Leverage security, scalability, and reliability.
* Support containerized execution – Ensure portability across environments (Dev, QA, Prod).

# 2. Architecture Overview

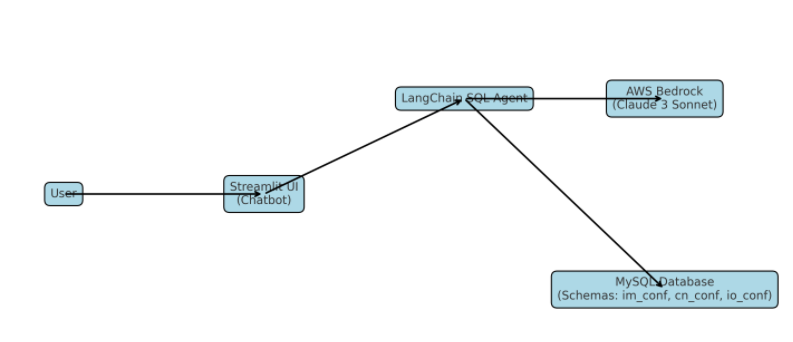
## 2.1 Components Used

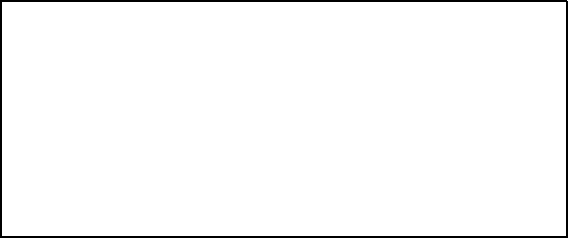
* LLM (AWS Bedrock – Claude 3 Sonnet): Natural language processing and response generation.
* LangChain Framework: Acts as middleware between LLM and database, handles query generation.
* MySQL Database: Stores SCD schemas (im\_conf, cn\_conf, io\_conf).
* Docker: Containerization ensures consistency across environments.
* Cloud9 (AWS IDE): Used for development, testing, and packaging.
* AutoPTP: Deployment automation pipeline, builds and deploys containers.
* AWS Secrets Manager: Securely stores and retrieves database credentials and sensitive keys.

## 2.2 High-Level Flow

1. User Interaction – A user types a query in the chatbot UI (Streamlit-based).
2. Processing via LangChain + LLM (Bedrock) – The chatbot interprets intent.
3. SQL Generation (if needed) – Translates the query into an SQL statement for MySQL.
4. Database Execution – Fetches results from SCD schemas.
5. Response Formatting – LLM refines the output for readability.
6. Deployment – The chatbot runs inside a Docker container and is deployed using AutoPTP + Cloud9.

* Block diagram here showing → User → Chatbot UI → LangChain/LLM (Bedrock) → MySQL → Response → Docker/AutoPTP/AWS)\*





# 3. Deployment & Setup

## 3.1 Development Environment

- AWS Cloud9 IDE used for:  
 - Writing Python + Streamlit chatbot application.  
 - Integrating LangChain with AWS Bedrock API.  
 - Managing secure database connections using AWS Secrets Manager.

## 3.2 Containerization with Docker

Created a Dockerfile to package the chatbot with dependencies. Built image using:

docker build -t scd-chatbot .

docker run -p 8501:8501 scd-chatbot

Ensures the chatbot can run consistently across environments.

## 3.3 AutoPTP Deployment

CI/CD automation via AutoPTP. Pipeline includes:  
1. Build – Docker image created.  
2. Publish – Image pushed to AWS container registry.  
3. Deploy – Automated deployment to target AWS environment.

# 4. Security & Access Control

* AWS IAM Roles – Provide secure access to Bedrock, Secrets Manager, and MySQL.
* AWS Secrets Manager – Used to securely store and retrieve database credentials and sensitive keys.
* Authorized User Access – Restricts chatbot to internal users only.
* Read-only Database Access – Prevents accidental data changes.

# 5. Features of the Chatbot

* Natural language query interface for SCD data.
* Secure database querying with role-based access.
* Fast responses (<10 seconds average).
* Dockerized for portability.
* Integration with Claude model (AWS Bedrock) for intelligent replies.
* Scalable deployment using AutoPTP.

# 6. Lessons Learned / Best Practices

* Use AWS Secrets Manager and IAM roles for better security.
* Containerization ensures environment consistency.
* AutoPTP reduces manual deployment and enables DevOps best practices.
* Optimizing database queries and using caching significantly improves response time.

# 7. Next Steps / Future Enhancements

* Add SSO authentication for corporate user login.
* Integrate with Tableau/Spotfire dashboards for visualization.
* Extend support for multi-turn conversations with memory.
* Expand chatbot coverage for more clinical data schemas.
* Implement monitoring & logging for better observability.

**8. References / Useful Links**

* GitHub Repository: [bms-corp/SCD-ChatAgent](https://github.com/bms-corp/SCD-ChatAgent)
* Multiverse Project Dashboard: [BMS - AWS Multiverse Platform Management](https://multiverse.web.bms.com/index.html#/)