



Mastering AI Agents: Building Production- Ready Applications



Getting started with Amazon
Bedrock AgentCore

▼ Prerequisites

At an AWS Event (Setup)

Self paced (Setup)

Sagemaker AI Studio

Amazon Bedrock AgentCore
Fundamentals (Optional)

Lab 1: Create the Agent Prototype

Lab 2: Enhance your Agent with
Memory

Lab 3: Scale with Gateway and
Identity

Lab 4: Deploy the Agent to
production with Observability

**(Optional) Lab 5: Build a
Customer-Facing Frontend
Application**

Lab 6: Clean up

AgentCore

AgentCore Documentation

▼ AWS account access

[Open AWS console](#)
(us-west-2)

[Get AWS CLI credentials](#)

Exit event

[Event dashboard](#) > (Optional) Lab 5: Build a Customer-Facing Frontend...

(Optional) Lab 5: Build a Customer-Facing Frontend Application

Overview

In the previous labs, we've built a comprehensive Customer Support Agent with memory, shared tools, and production-grade deployment. Now it's time to create a user-friendly frontend that customers can actually use to interact with our agent.

In this lab, we'll create a **Streamlit-based web application** that provides customers with an intuitive chat interface to interact with our deployed Customer Support Agent. The frontend will include:

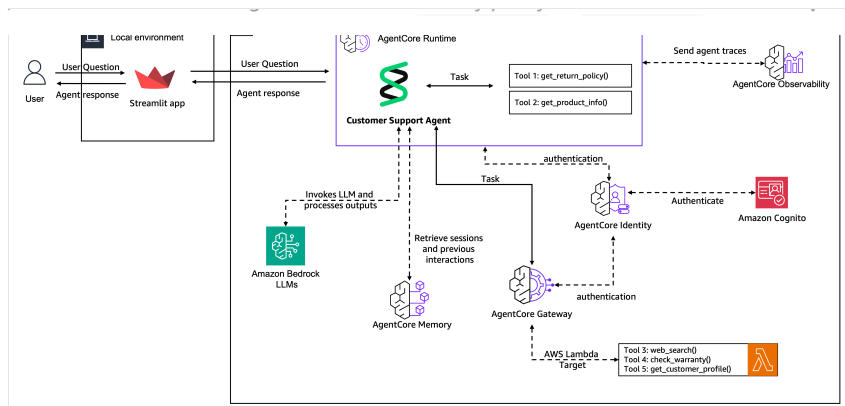
- **Secure Authentication** - User login via Amazon Cognito
- **Real-time Chat Interface** - Streamlit-powered conversational UI
- **Streaming Responses** - Live response streaming for better user experience
- **Session Management** - Persistent conversations with memory
- **Response Timing** - Performance metrics for transparency

What You'll learn

- How to integrate Secure Authentication with a frontend.
- How to implement real-time streaming responses
- How to manage user sessions and conversation context
- How to create an intuitive chat interface for customer support

Architecture Diagram

The architecture showcases the composability of Amazon Bedrock Agentcore features tied together with a front end web based application for the customer support agent:



Steps to follow

Step 1: Understanding the Frontend Architecture

Our Streamlit application consists of several key components:

Core Components

1. main.py - Main Streamlit application with UI and authentication
2. chat.py - Chat management and AgentCore Runtime integration
3. chat_utils.py - Utility functions for message formatting and display
4. sagemaker_helper.py - Helper for generating accessible URLs

Authentication Flow

1. User accesses the Streamlit application
2. Amazon Cognito handles user authentication
3. Valid JWT tokens are used to authorize AgentCore Runtime requests
4. User can interact with the Customer Support Agent securely

Step 2: Launch the Customer Support Frontend 🚀

Now let's start our Streamlit application. The application will:

1. Generate an accessible URL for the application
2. Start the Streamlit server on port 8501
3. Connect to your deployed AgentCore Runtime from Lab 4
4. Provide a complete customer support interface

Important Notes:

- The application will run continuously until you stop it from the notebook (Ctrl+C)
- Make sure your AgentCore Runtime from Lab 4 is still deployed and running
- The Cognito authentication tokens are valid for 2 hours

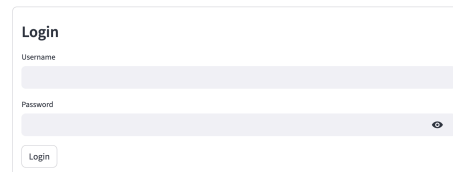
► Code Snippet

Step 3: Testing Your Customer Support Application

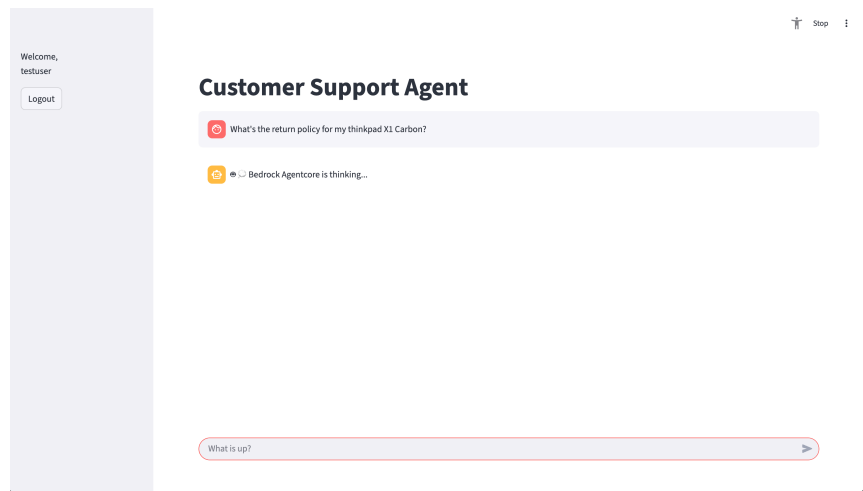
Once your Streamlit application is running, you can test the complete customer support experience:

Authentication Testing

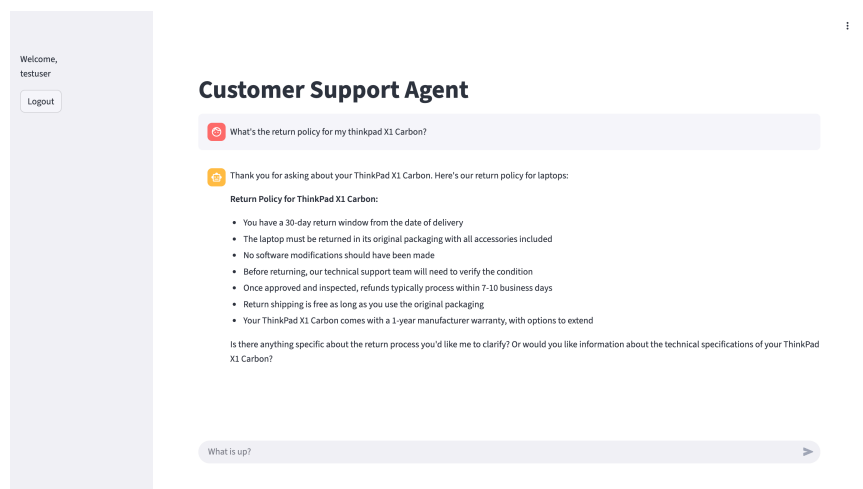
1. Access the application using the URL generated above
2. Sign in with the test credentials provided in the output
3. Verify that you see the welcome message with your username



The screenshot shows a web application interface with a login form. The form is titled "Login" and contains two input fields: "Username" and "Password". The "Password" field has a toggle icon (an eye) to the right of the input box. Below the input fields is a "Login" button. The form is enclosed in a light gray border.



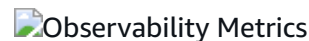
Customer Support Scenarios to Test:



Step 4: Monitoring and Observability

While testing your frontend application, you can monitor the agent's performance using AgentCore Observability on AWS GenAI Observability Dashboard on Cloudwatch console.

Navigate to the Cloudwatch console, and search for the GenAI Observability. You should see the following:



Navigate to X-Ray Traces and turn on the transaction search to see the agent's behavior. You should enable it as shown below:




Congratulations - you successfully completed Lab 5: Building a Customer-Facing Frontend Application!

Try it out

At an AWS event

If you are following the workshop via workshop studio, now go to JupyterLab in SageMaker Studio. In the JupyterLab UI navigate to `lab-05-frontend.ipynb`

Self paced

For the complete working implementation and examples: [Front end Lab Notebook](#) .

Congratulations !! You've completed the workshop and successfully deployed an End-to-End Agentic Solution using AgentCore Services !

You now have a complete, customer support system that includes:

- Intelligent Agent (Lab 1) - AI-powered support with custom tools
- Persistent Memory (Lab 2) - Conversation context and personalization
- Shared Tools & Identity (Lab 3) - Scalable tool sharing and access control
- Production Runtime (Lab 4) - Secure, scalable deployment with observability
- Customer Frontend (Lab 5) - web interface for end users

Key Capabilities Demonstrated:

- Multi-turn Conversations - Agent maintains context across interactions
- Tool Integration - Seamless use of product info, return policy, and web search
- Memory Persistence - Customer preferences and history maintained
- Security & Identity - Proper authentication and authorization
- Observability - Full tracing and monitoring of agent behavior

[Previous](#)

[Next](#)