Secure RAG-Powered Travel Chatbot

SERVERLESS LIMITLESS

Introduction

- Leading Travel & Hospitality Platform serving flight and hotel customers
- Operates at scale with high volumes of customer support interactions
- Focused on delivering fast, accurate, and personalized support experiences
- Adopting AI-driven support automation

Problem Statement

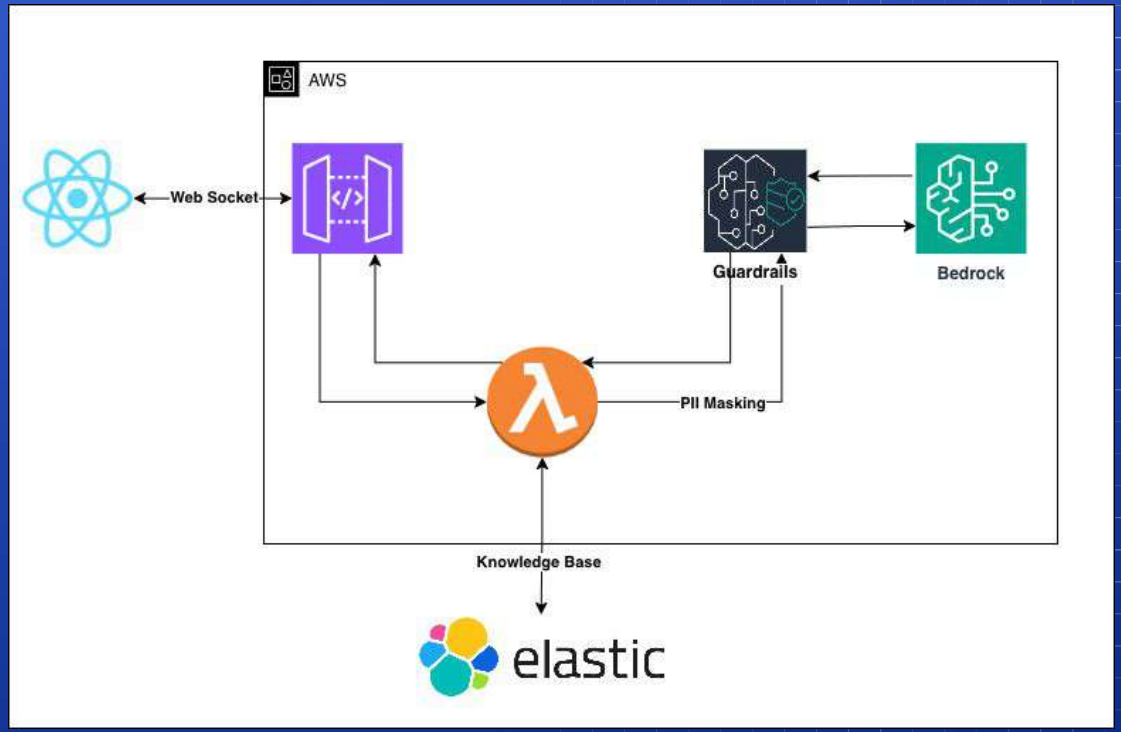
- High volume of **repetitive customer support** queries
- Leverage past support knowledge
- **Generative AI** introduces privacy and compliance risks
- Need accurate, contextual responses without exposing sensitive data
- Maintain customer trust, data security, and response accuracy

Design Goals

- Reuse knowledge from resolved queries without exposing **PII**
- Enforce **privacy-first AI**
- Design for scalability using a **serverless** architecture
- Ensure **compliance** by design

Proposed Architecture

- Web-based chat interface
- AWS Bedrock (Claude)
- RAG retrieval via hybrid search (keyword + semantic)
- PII protection enforced through Bedrock Guardrails
- AWS Lambda with Strands Agent



AWS Bedrock

Bedrock - Brains

- Provides access to foundation models via a single API
- No infrastructure or model management required
- Supports enterprise security and compliance needs
- Integrates natively with AWS services
- Enables safe AI usage with built-in Guardrails, Knowledge Bases, Agents, Evaluation, etc

Claude

Meta Llama 3

Amazon Titan

Mistral AI

Elasticsearch - Knowledge Base

Elasticsearch - Knowledge Base

- Stores resolved customer support queries
- Indexed for both keyword and semantic search
- **Semantic** search captures intent and contextual similarity
- Hybrid scoring improves relevance for customer support queries
- ElasticSearch **Serverless** for managed search and RAG workloads
- Reduces operational overhead while maintaining performance

RAG

- Common RAG Strategies in Elasticsearch
  - Keyword-based retrieval (BM25)
  - Semantic / vector-based retrieval (dense vectors, ELSER)
  - Hybrid retrieval (BM25 + vector search)
  - Metadata-driven filtering and ranking
- What We Used?
  - Hybrid (RRF) - handles both semantic queries and exact

Chunking

- Common Chunking Strategies
  - Fixed-size text chunking (by tokens or characters)
  - Semantic chunking (topic or intent-based)
  - Conversation turn–based chunking
  - Problem–solution based chunking
- What We Used
  - Problem-level chunking
  - Each resolved ticket is treated as a single chunk including metadata, full conversation, etc.

Handling PII Data

Challenge: Handling PII Data

- Customer queries may include sensitive personal data
- Privacy risks increase with scale and automation
- Detects and masks PII in user input
- Provides consistent, automated PII enforcement

Solution: Bedrock Guardrails

- Offers content filtering, denied topic enforcement, **PII redaction**, word filtering, etc
- **No custom model training** or rule maintenance required
- Supports common PII types (names, emails, phone numbers, IDs)
- Integrates **native**ly with AWS Bedrock models
- Enables **centralized governance** and policy updates
- **Reduces security risk** compared to custom PII pipelines

Orchestration

AWS Lambda & Strands Agent

- Acts as the central **orchestration** layer for the chatbot
- **Manages request** flow between chat interface, Elasticsearch, and Bedrock
- Implements agent-based reasoning using Strands Agent library
- Handles **prompt construction and context enrichment**
- Integrates **PII filtering and Guardrails enforcement**
- Enables scalable, event-driven execution without servers

Final Outcomes

- **Context-aware** customer support using RAG
- Reuse of resolved support knowledge
- Automated **PII detection** and masking with Bedrock Guardrails
- **Privacy-first**, compliance-ready AI architecture
- Scalable, serverless design with **low operational overhead**

DEMO

AntStack®

Thank you

Rahul Kumar

Sohan Manju

AntStack - antstack.com