**Customer Support Chatbot Agent**

**Problem Statement:**

Customers often face long wait times for basic queries (e.g., FAQs, ticket status), which overloads support staff and lowers satisfaction.

**Goal:**

Build an AI chatbot that can autonomously handle routine support requests using Retrieval-Augmented Generation (RAG) and pre-trained LLMs, with seamless fallback to human agents.

**Value Proposition:**

* Reduces ticket volume for human support staff
* Improves resolution time and customer satisfaction
* Operates 24/7 with multilingual support capabilities

**Project Scope:**

**In-Scope:**

* Use of LLMs (e.g., OpenAI, Hugging Face) + vector DBs for RAG
* Integration with Web interface
* Initial MVP with 15–20 FAQs and IT support queries

**Further Improvements:**

* Full CRM integrations
* Voice support
* Escalation handling beyond notification

**Essential Key Lean Metrics:**

* **Lead Time**: Time from query to resolution
* **Throughput**: No. of queries handled/day
* **CSAT**: Post-chat feedback score
* **Escalation Rate**: % of conversations forwarded to human agents

**Sprint Breakdowns and user stories:**

**Sprint Duration:** 1 week

**Team Members:**

AI Engineer, Data Engineer, Front-End/UI Engineer, Testing Engineer

**Sprint 1: Dataset Preparation & Planning:**

**Objective:** Set up foundational datasets and environment

**User Stories**:

* *Data engineer* collect and clean customer support ticket datasets so that the chatbot has high-quality data.
* The whole team should define the MVP scope and success criteria so that we can stay aligned on goals.
* AI Developer should explore prebuilt Q&A datasets to enrich the chatbot's knowledge.

**Deliverables**:

* Curated FAQ & support ticket dataset
* Dataset dictionary and data cleaning scripts
* Project plan and backlog in Agile tool

**Sprint 2: Sprint 1: Dataset Preparation & Planning:**

**Objective**: Set up Retrieval-Augmented Generation architecture

**User Stories**:

* *AI engineer* should build a RAG pipeline using Hugging Face Transformers or Sentence Transformers so that the chatbot can fetch relevant answers from the knowledge base.
* *Data Engineer should* configure a local vector database or dictionary so we can store and query vector embeddings efficiently.
* *A tester* should validate retrieval quality using test queries so we can measure initial performance.

**Deliverables**:

* Functional RAG pipeline
* Configured vector database (e.g., ChromaDB)
* Test results for retrieval quality

**Sprint 3: Chatbot Integration & Front-End UI**

**Objective**: Create chat interface and connect with RAG backend

**User Stories**:

* *Front-end developer* should build a simple web chat UI so users can interact with the bot easily.
* *Data Engineer* Should connect the UI with the RAG pipeline so queries can be handled live.
* *AI Engineer* should show the sources for the responses so that the answers can be trusted.

**Deliverables**:

* Live chatbot UI
* Functional end-to-end integration
* Support for showing sources for response

**Sprint 4: Testing, Feedback, and Refinement**

**Objective: Refine the chatbot based on user testing and performance metrics**

**User Stories:**

* *Tester* should test the chatbot across various FAQ and IT scenarios to ensure *accurate and helpful responses.*
* *Collect feedback from pilot users so we can prioritize improvements.*
* *As a whole team we work to improve weak areas identified in the test phase, so the bot is ready for MVP release.*

**Deliverables:**

* *Test case coverage and bug fixes*
* *User feedback report*
* *Final MVP demo and documentation*

**Continuous Improvement Plan**

* Bi-weekly retrospectives for process tweaks
* User feedback loop for chatbot retraining
* Potential backlog: add CRM APIs, voice interface, or multi-language support

**Free resources to consider:**

**Customer Support & Helpdesk Datasets**

1. **Zendesk Customer Support Ticket Dataset (via Kaggle)**
   * [https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter](file:///Volumes/Essentials/Job_Search_2024/o%09https:/www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter)  
     Twitter-based customer service dataset with support request interactions for training customer support models.
2. **Customer Support Dataset from Amazon (Customer Support Dialogue Corpus)**
   * <https://github.com/amazon-science/customer-support-dataset>  
     Contains multi-turn dialogues from real customer support chats (anonymized).
3. **Ubuntu Dialogue Corpus (IT Support)**
   * <https://github.com/rkadlec/ubuntu-ranking-dataset-creator>  
     Massive dataset for multi-turn technical support conversations.
4. **Microsoft Support Dataset (WebAP and QA datasets)**
   * <https://github.com/microsoft/MSMARCO>  
     Contains real-world customer queries and answers, usable for training retrieval-based support bots.

**Pre-built Q&A & FAQ-style Datasets**

1. **TREC QA Dataset**
   * <https://cogcomp.seas.upenn.edu/Data/QA/QC/>  
     FAQ-style question dataset useful for building baseline QA retrieval systems.
2. **Stack Exchange Data Dumps (e.g., Super User, Ask Ubuntu)**
   * <https://archive.org/details/stackexchange>  
     Offers real Q&A from technical forums – useful for domain-specific support agents.

**Retrieval-Augmented Generation (RAG) & Embedding Tools**

1. **Hugging Face Datasets for Retrieval/QA**
   * <https://huggingface.co/datasets>  
     Search for: squad, hotpotqa, natural\_questions, ms\_marco, and faq.
2. **LangChain Examples & Templates**
   * [https://docs.langchain.com](https://docs.langchain.com/)  
     Offers chatbot and tool-using agent templates for interfacing with vector stores like Pinecone or Chroma.
3. **ChromaDB for Vector Retrieval**
   * <https://www.trychroma.com/>  
     Lightweight open-source vector DB, ideal for local prototyping.