Midterm Assignment: Smart Home Technical Support Agent

1. Executive Summary

Our Smart Home Technical Support Agent is designed to assist users in diagnosing and resolving issues with their smart home devices. As technology grows more complex, users often struggle with setup, connectivity, and compatibility problems. Our assistant provides real-time, conversational support to simplify troubleshooting and reduce reliance on human customer service.

- **Purpose & Value Proposition**: To provide 24/7 intelligent support for smart home users, reducing frustration and downtime.
- **Key Functionalities**:
    - Troubleshoot device issues via natural language
    - Provide setup and configuration guidance
    - Recommend solutions based on device logs and user input
- **Target Users**: Homeowners and renters using smart home devices (e.g., Google Nest, Amazon Echo, Philips Hue)
- **Implementation Approach**: A retrieval-augmented (generation) system that uses vector embeddings to match user queries with a curated knowledge base of device manuals, FAQs, and troubleshooting guides.

2. Project Definition

Domain Selection: Smart Home Technical Support

We selected this domain due to the increasing adoption of smart home technology and the frequent technical issues users face. Many users lack technical expertise to resolve problems independently.

Problem Statement

Smart home users often face issues like device malfunctions, connectivity problems, or confusion during setup. Traditional support channels can be slow and frustrating. Our assistant is here to provide quick, accurate, and user-friendly technical support

Core Functionalities

- Natural language troubleshooting for common device issues
- Step-by-step setup instructions for new devices

- Compatibility checks between devices and platforms
- Voice command diagnostics and suggestions
- Escalation recommendations when human support is needed

Target Users

- Tech-savvy homeowners looking for quick fixes
- Elderly or non-technical users needing simplified guidance
- Renters with multi-brand smart home setups

Value Proposition

- Reduces support wait times and user frustration
- Offers consistent, brand-agnostic troubleshooting
- Empowers users to resolve issues independently
- Scales easily across device types and brands

3. Data Requirements Analysis

Data Types

- Unstructured: Device manuals, support articles, user forums
- Structured: Device metadata (model, firmware, compatibility)
- Semi-structured: JSON-formatted troubleshooting flows
- Logs: Simulated device error logs for testing

Data Sources

- Open-source manufacturer documentation (e.g., Philips, Nest, Ring)
- Public support forums (e.g., Reddit, Stack Exchange)
- Synthetic data from simulated device interactions
- Open-source smart home datasets (e.g., Home Assistant logs)

Data Volume & Velocity

- Initial corpus: 100 support entries
- 50KB-1MB text per device
- Updates: Monthly ingestion of new manuals and FAQs

Data Quality Requirements

- Verified solutions from official sources
- Clear, concise, and actionable instructions
- Coverage across major device brands and categories
- Properly labeled with device type, brand, firmware

Potential Data Challenges

- Inconsistent terminology across brands
- Outdated or deprecated device information
- Ambiguity in user-reported issues

Data Schema (Simplified)

Field	**Type**	**Description**
issue_id	String	Unique identifier
device_type	String	e.g., thermostat, camera
brand	String	e.g., Nest, Ring
issue_description	Text	User-reported problem
resolution_steps	List[Text]	Step-by-step fix
tags	List[String]	Keywords (e.g., Wi-Fi, pairing)
source	String	Manual, forum, synthetic

4. Processing Pipeline Design

Data Preprocessing Workflow

1. Ingest documents from manuals, forums, and APIs
2. Clean and normalize text (remove HTML, fix encoding)
3. Extract issue-resolution pairs
4. Generate embeddings using sentence transformers

5. Index data for semantic retrieval

Feature Engineering Approach

- Embeddings for semantic similarity (e.g., BERT, MiniLM)
- Device-brand-topic classification
- Confidence scoring for retrieved solutions

Processing Pipeline Diagram

[Raw Docs] → [Cleaning] → [Issue Extraction] → [Embedding] → [Indexing] → [Query Matching]

Data Transformation Techniques

- Text: spaCy for entity recognition (device names, error codes)
- JSON: Flatten troubleshooting flows
- Logs: Parse and label error messages

Infrastructure Considerations

- Google Colab for development
- FAISS for fast vector search
- SQLite or Firebase for metadata storage
- Batch updates for new content ingestion

5. Implementation Strategy

Technical Stack

- Python, Pandas, spaCy, NLTK
- SentenceTransformers (MiniLM)
- FAISS for retrieval
- Streamlit or Flask for UI prototype
- Langchain
- Gradio

Development Timeline

Week Milestone

1	Finalize domain and collect data
2–3	Build preprocessing and embedding pipeline
4	Implement retrieval and UI
5	Test with simulated queries
6	Final evaluation and documentation

Team Member Responsibilities

- Alex: Data collection and preprocessing
- Jamie: Embedding and retrieval system
- Taylor: UI/UX and user testing
- Morgan: Project coordination and documentation

Resource Requirements

- Google Colab Pro
- Access to device manuals and forums
- 4 team members with Python and NLP experience

Implementation Challenges

- Handling vague or multi-device queries
- Ensuring brand-agnostic responses
- Managing updates to device firmware and features

Integration Approach

- Modular architecture: UI ↔ API ↔ Retrieval Engine
- RESTful API for query handling and response generation

6. Evaluation Framework

Success Metrics

- Resolution accuracy (manual review)

- User satisfaction (survey score ≥ 4/5)
- Retrieval precision and recall

Testing Strategy

- Unit tests for each module
- Simulated user queries for end-to-end testing
- A/B testing for different explanation formats

User Feedback Mechanism

- In-app thumbs up/down on solutions
- Optional feedback form after each session

Performance Benchmarks

- ≥ 80% resolution accuracy on test queries
- ≤ 3 seconds average response time

Continuous Improvement Process

- Weekly review of user feedback
- Monthly ingestion of new support content
- Quarterly model updates and fine-tuning