**Order Status Assistant - Project Documentation**

**Project Overview**

The Order Status Assistant is an AI-powered customer service application that allows customers to check their order status using natural language queries. Built with LangChain, Streamlit, and Supabase, it transforms traditional order lookup systems into an intelligent conversational interface.

**Key Features**

* **Natural Language Processing**: Customers can ask "Where is my order?" instead of navigating complex menus
* **Multiple Search Methods**: Search by order ID, email, phone number, or customer name
* **Real-time Database Integration**: Instant access to order information from Supabase
* **Conversational AI**: Handles follow-up questions and maintains context
* **User-friendly Interface**: Clean Streamlit web interface with chat and sidebar options

**Technology Stack**

* **Frontend**: Streamlit
* **AI/NLP**: LangChain + OpenRouter API
* **Database**: Supabase (PostgreSQL)
* **Language**: Python

**System Components**

**1. Main Application (main.py)**

* Streamlit interface and session management
* Coordinates all system components
* Handles user interactions and display

**2. Chat Handler (chat\_handler.py)**

* Processes user messages with AI
* Manages conversation context
* Integrates database results with AI responses

**3. Database Module (database.py)**

* Connects to Supabase database
* Executes order lookup queries
* Formats data for display

**4. Query Handling (query.py)**

* Extracts information from user queries (order IDs, emails, phones)
* Prepares database context for AI model
* Text processing and validation

**5. Setup (setup.py)**

* Environment setup
* AI model and database initialization
* System prompt definition

**6. Sidebar (sidebar.py)**

* Quick lookup interface
* Direct database searches
* Alternative to chat interface

**Database Schema**

The Order\_Status table contains comprehensive order information:

**Core Fields**

* order\_id, order\_status, order\_date, total\_amount
* customer\_name, customer\_email, customer\_phone
* customer\_address, city, state, postal\_code
* delivery\_date, delivery\_time\_slot, items\_ordered
* payment\_status, delivery\_driver, driver\_phone
* special\_instructions, priority

**Installation & Setup**

**Prerequisites**

* Python
* Supabase account
* OpenRouter API key

**Quick Start**

1. **Install dependencies**:

bash

pip install -r requirements.txt

1. **Configure environment** (.env file):
2. OPENAI\_API\_KEY=your\_openrouter\_api\_key
3. SUPABASE\_URL=your\_supabase\_url

SUPABASE\_KEY=your\_supabase\_key

1. **Run application**:

bash

streamlit run main.py

**Usage Examples**

**Chat Interface**

* "Check order 12345"
* "Orders for [john@email.com](mailto:john@email.com)"
* "What's the status of my recent order?"
* "When will my order be delivered?"

**Sidebar Quick Lookup**

* Direct order ID search
* Email-based customer lookup
* Phone number search

**Key Benefits**

* **Natural interaction**: No need to remember specific order formats
* **Comprehensive information**: Complete order details in one place
* **Follow-up questions**: Can ask additional questions about the same order
* **Improved satisfaction**: Faster, more accurate responses

**System Workflow**

1. **User Input**: Customer types natural language query
2. **Information Extraction**: System identifies order details, emails, or phone numbers
3. **Database Query**: Searches Supabase for matching orders
4. **AI Processing**: LangChain formats database results with conversational response
5. **Response**: User receives natural language answer with order details

**Project Structure**

├── main.py # Main Streamlit application

├── chat\_handler.py # AI chat processing

├── database.py # Database operations

├── query.py # Text processing and query handling

├── setup.py # Configuration & initialization

├── sidebar.py # Sidebar interface

├── requirements.txt # Dependencies

└── .env # Environment variables

**Conclusion**

The Order Status Assistant demonstrates how modern AI technology can transform customer service operations. By combining natural language processing with robust database integration, it provides an intuitive, efficient solution.