ShopWise ChatBot Details

Table of Contents
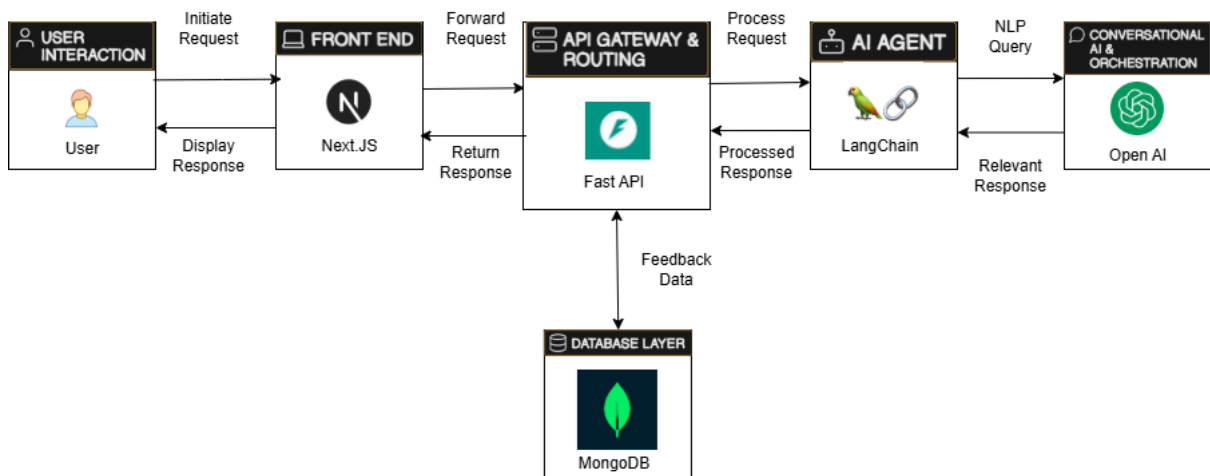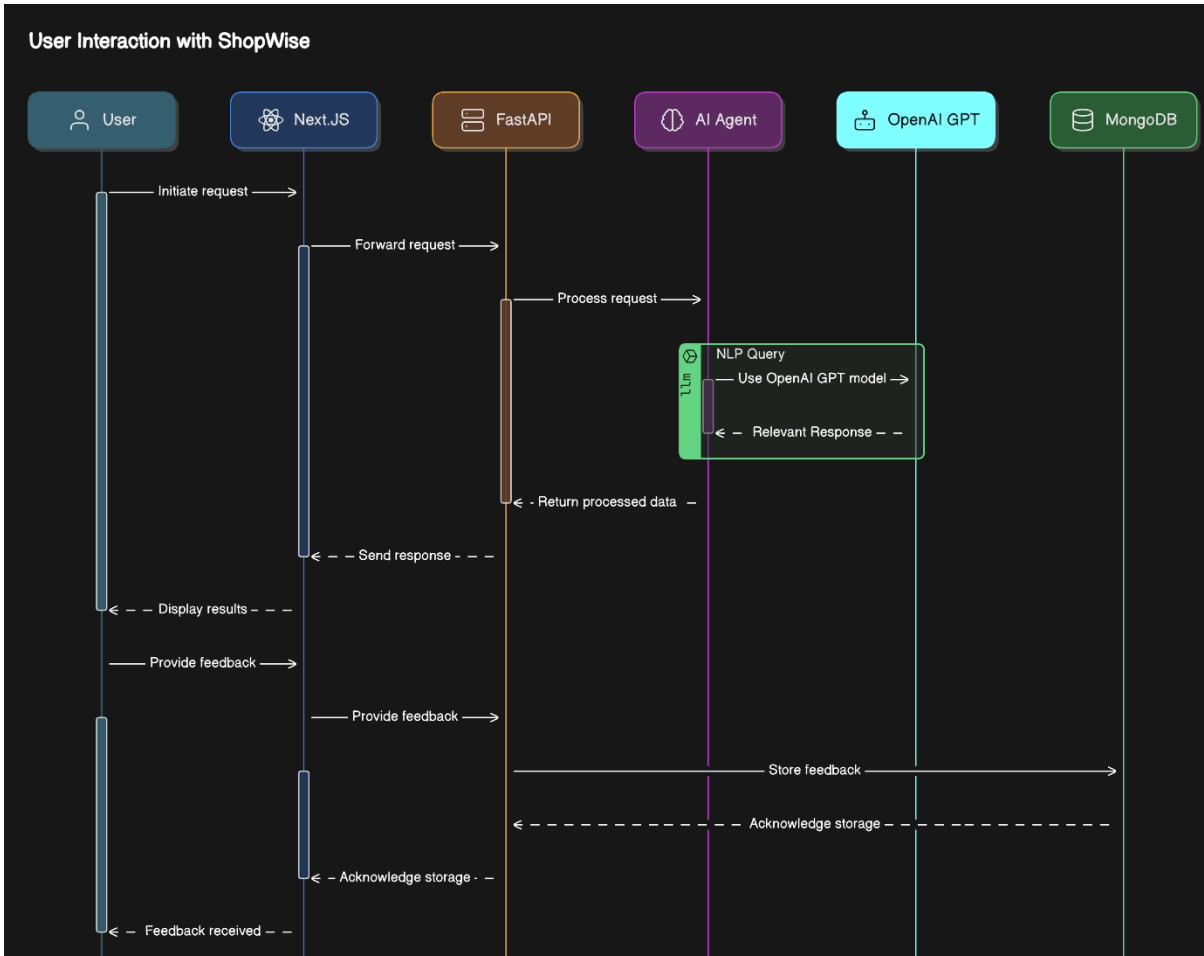
Problem Statement

To build an AI-powered product support assistant for ShopWise

Architectural Diagrams:

a) Architectural Diagram


b) Sequence Diagram

Tech Stack:

| Layer	Technology	Purpose
Front-End	Next.js	Building the user interface for the chatbot
Back-End	Python (FastAPI)	API development and handling backend logic
Large Language Model (LLM)	OpenAI GPT (ChatGPT)	Generating human-like responses to user queries
AI Agent	LangChain Framework (create_csv_agent)	Handling CSV data for advanced query processing
Database	MongoDB	Storing response details and user details
Hosting	Microsoft Azure	Hosting the front-end and back-end services
Version Control	Git & GitHub	Source code management and collaboration

Key Features:

URL of the application: https://talktodata.valtech.co.in/

Front End:

This interface for an e-commerce platform called "SHOPWISE". It is a generative AI application that makes it easier to query about the orders made by users on the application or on all products present. The main UI related components are as follows:
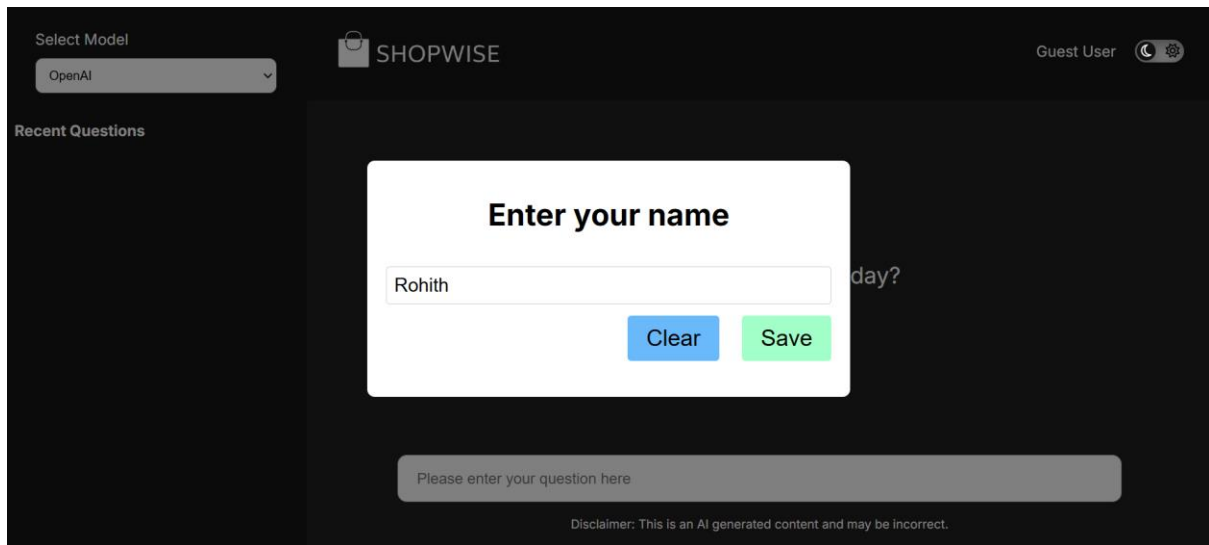
User when opens the application, can give their name thereby getting personalized answers and recommendations for all the orders made by them or on any products.
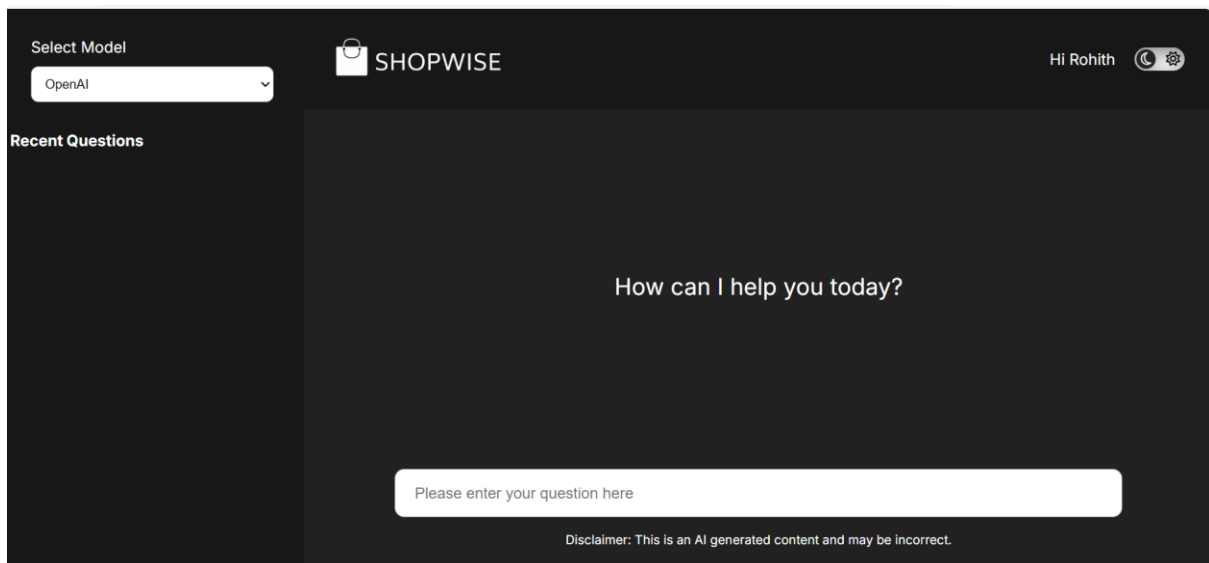
1. Header Section:

- Logo: The "SHOPWISE" logo with a shopping bag icon

- Model Selector: A dropdown menu showing "OpenAI" as the selected option

- User Info: Shows "Hi Rohith" with dark mode toggle, where in user can switch between dark and light modes

2. Left Sidebar:
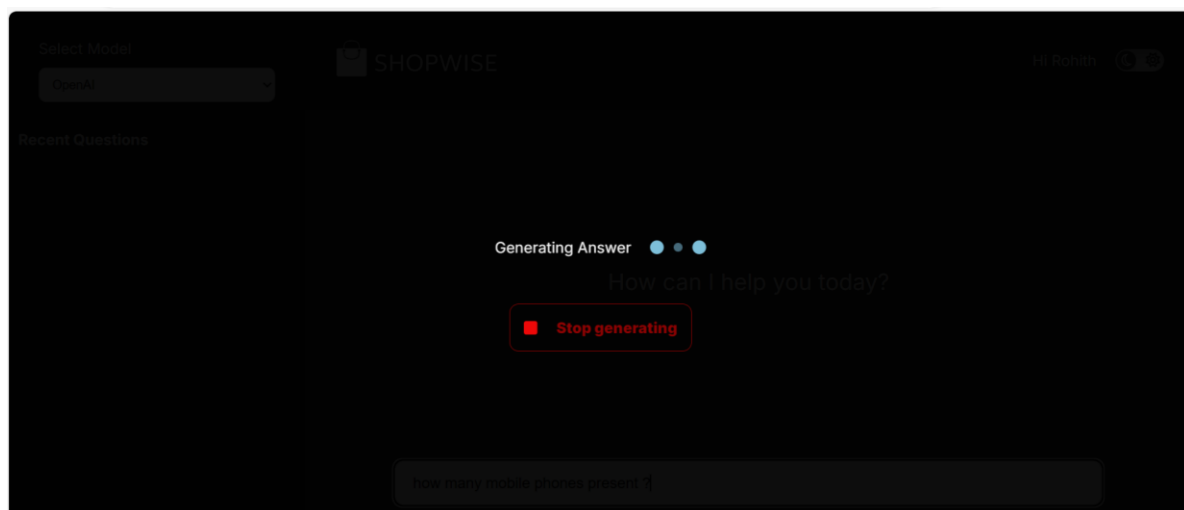
- Title: "Recent Questions"

- This shows all the user questions which are queried to our model.

3. Main Content Area:
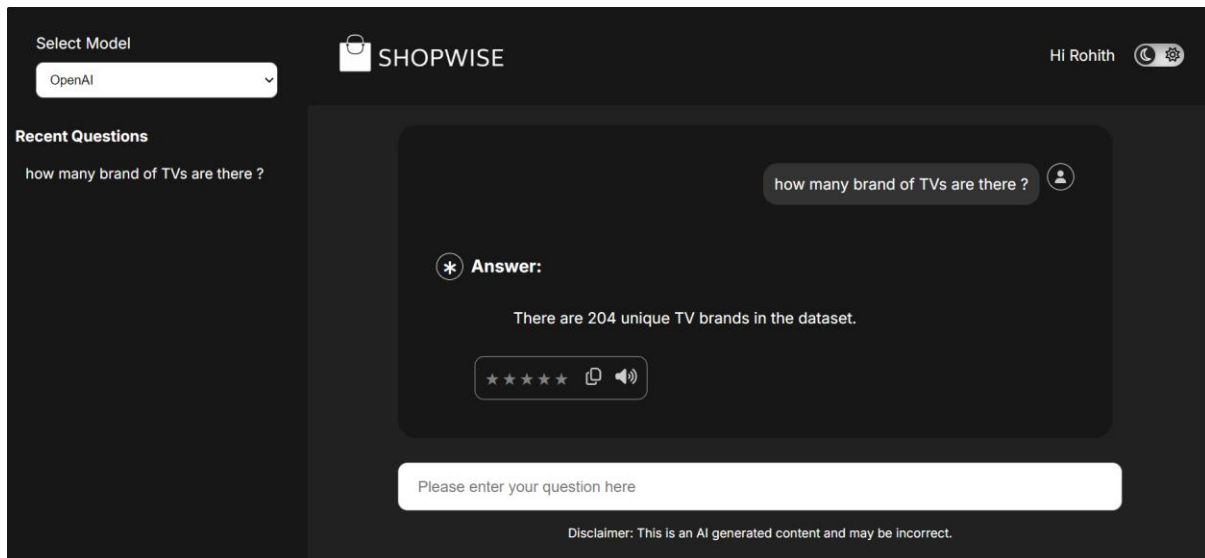
- A chat interface which show all user questions and answers about any orders or products related.

4. Footer:

- Disclaimer text stating "This is an AI generated content and may be incorrect."

User can stop generation of answer in between after a question being asked.

After getting answer, user can listen to the answer, copy the answer contents to clipboard and can give star ratings of 1 being lowest to 5 being highest.

Back End:

1. Context Understanding - Interpret and respond accurately based on the current user queries, and previous interactions

2. Product Comparison - compare details of at least 2 different products

3. Alternative Suggestions - suggest or recommend products matching a given criterion

4. Product Availability – tells whether product asked is available or not

5. Nested Requests – answers users query which is under another query

6. Order Status Check – gives status of the order

7. Tracking and Shipping Information – which location and shipping details are given

8. Return Eligibility – checks whether a product can be returned or not after delivery

9. Order History – users can see all previous orders details

10. Shipping Delay Inquiry – after delivery delay, next when products will be delivered

11. Order Assistance for Multiple Items – gives response for more than one query in a single interaction

12. Customer Feedback Collection -- encourages customers to leave reviews and ratings

Code Structure**:**

Inside **Oreilly-ShopWise-Chatbot** folder following folders are available:

backend –

 controller –

   feedbackController - submit_feedback api which adds or updates feedback data given by user

   generateAnswerController - generate_answer api gives response to users query based on the choice of

                              the LLM

data –

            synthetic-orders-data and **synthetic-product-data** csv source data files

chatbot.py – generate_response, uses a LangChain CSV agent to process a user query against specified CSV

            files, returning a structured response with the query, agent's answer, and sources.

llm.py – contains all details of llm used including its parameters

main.py – initializes a FastAPI application with CORS middleware to allow requests from any origin and

            includes routers from generateAnswerController and feedbackController to handle specific API

            endpoints.

requirements.txt – all dependencies libraries are mentioned

doc -> architecture -> decisions – contains all architecture decision records for each component of application

made during the design process
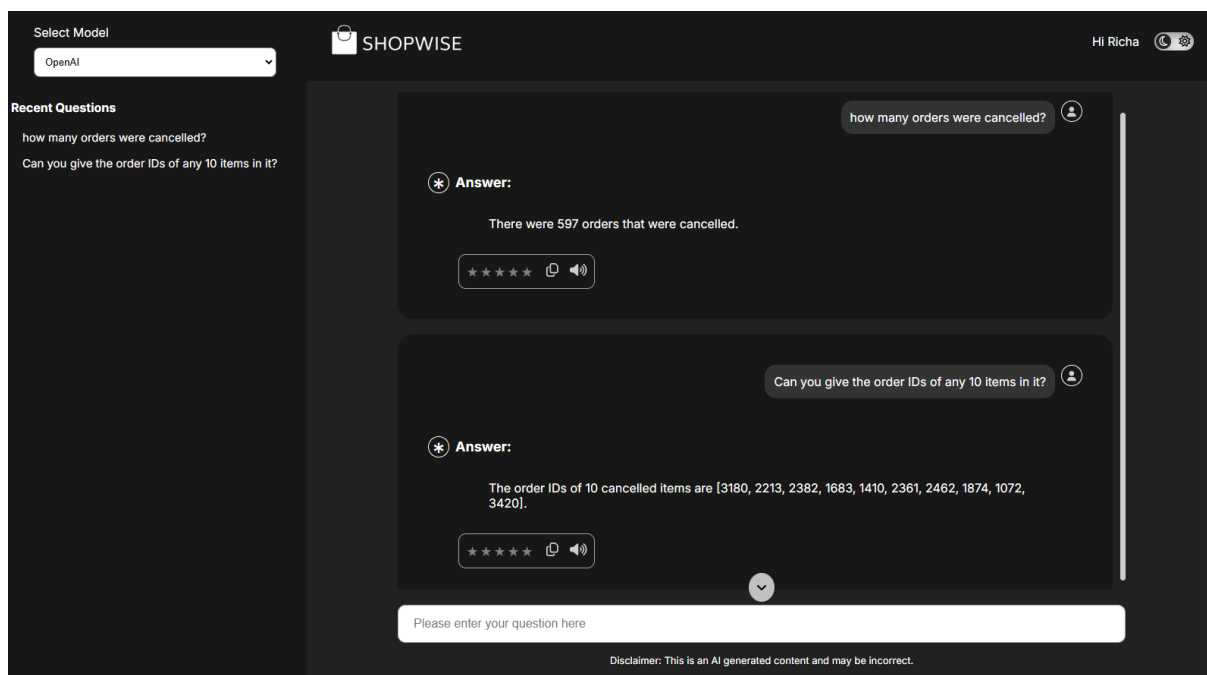
frontend –

src -app –

- **page.jsx** – contains all UI based components code
- **Style.css** – contains all style sheet code for components

Evaluation Scenarios:

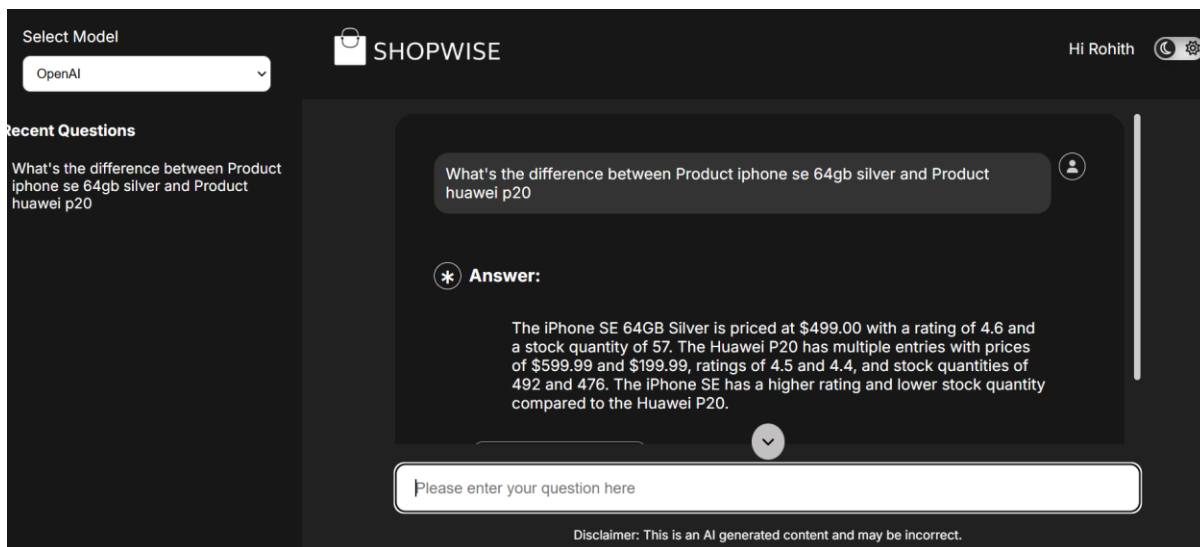**1. Context Understanding**

Query 1: "how many orders were cancelled?"

Query 2: "Can you give the order IDs of any 10 items in it?"
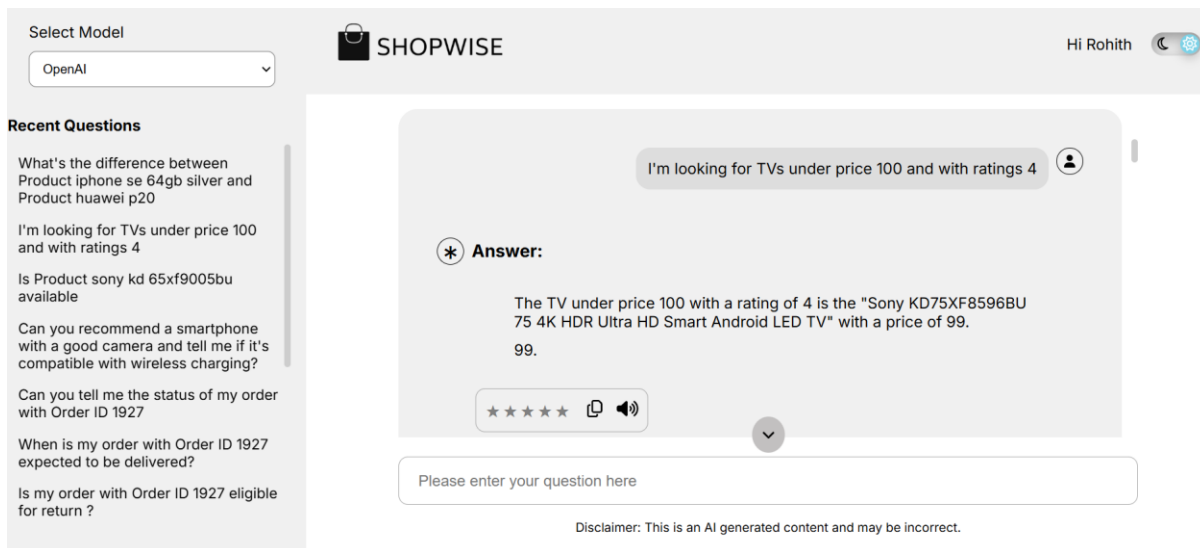

2. Product Comparison

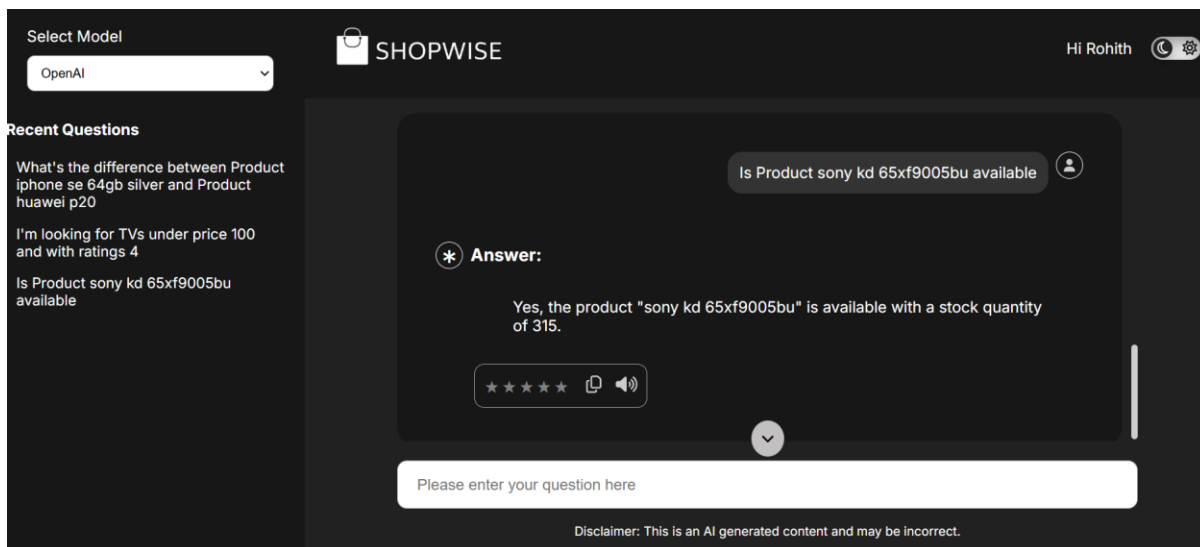Query: "What's the difference between Product iphone se 64gb silver and Product huawei p20?"

3. Alternative Suggestions

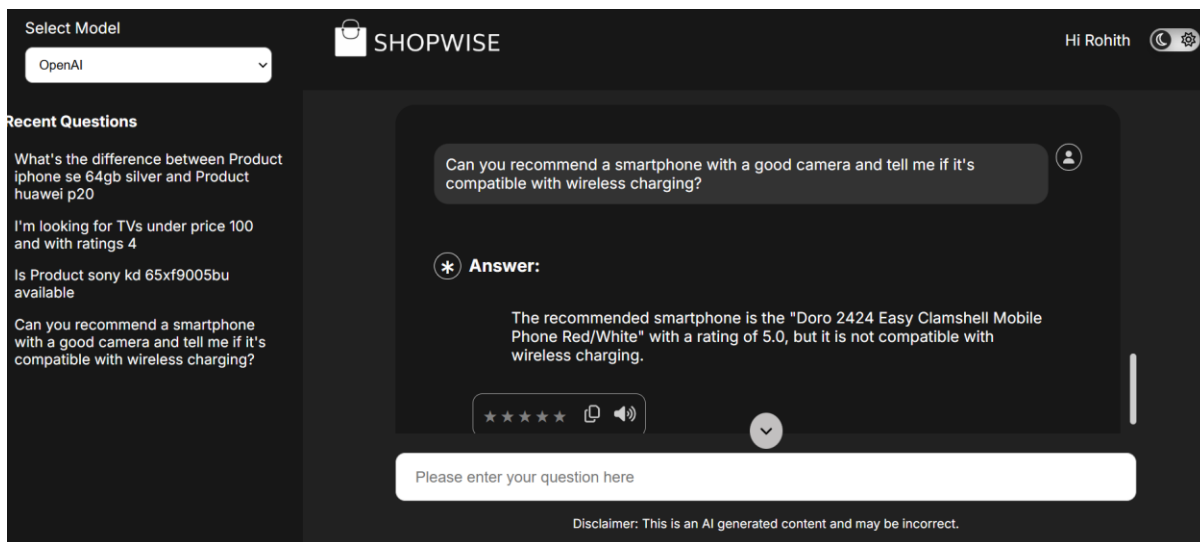Query: "I'm looking for TVs under price 100 and with ratings 4"

4. Product Availability
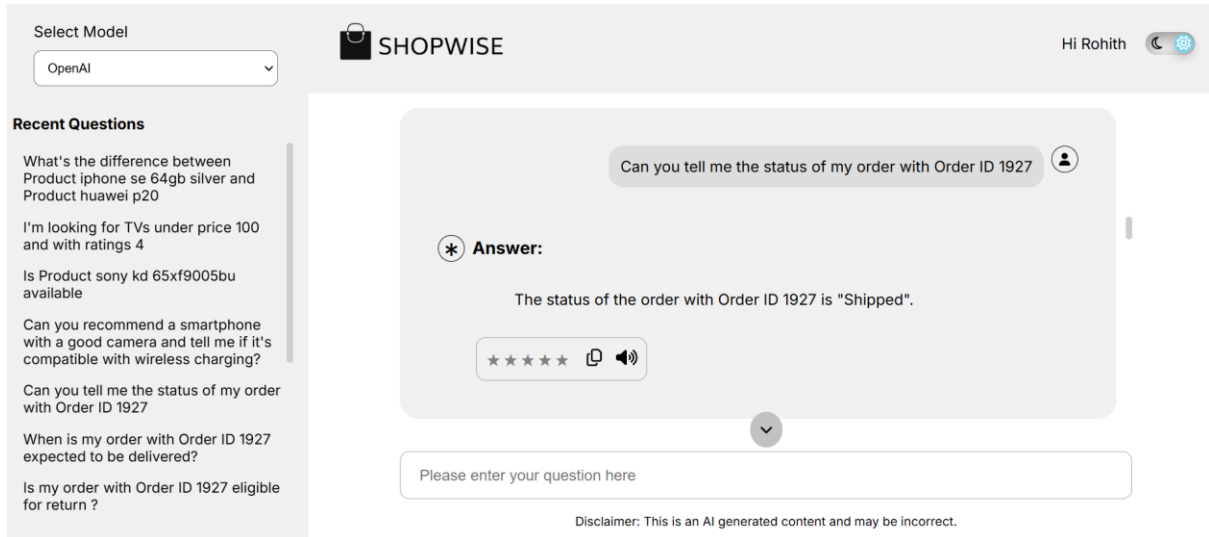
Query: "Is Product sony kd 65xf9005bu available"

5. Nested Requests

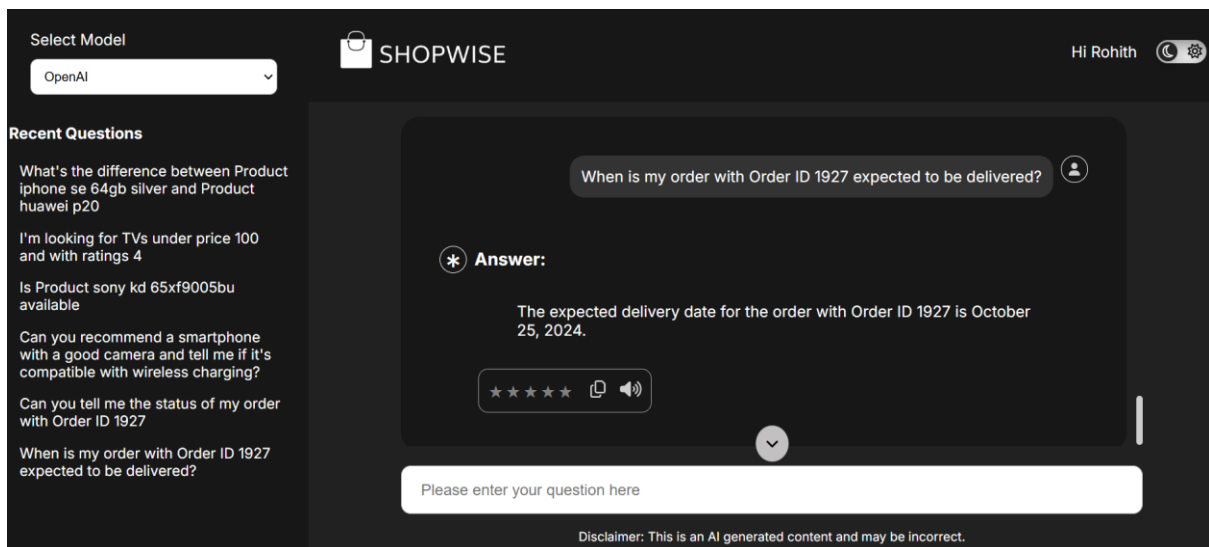Query: "Can you recommend a smartphone with a good camera and tell me if it's compatible with wireless charging?"

6. Order Status Check

Query: "Can you tell me the status of my order with Order ID 1927"
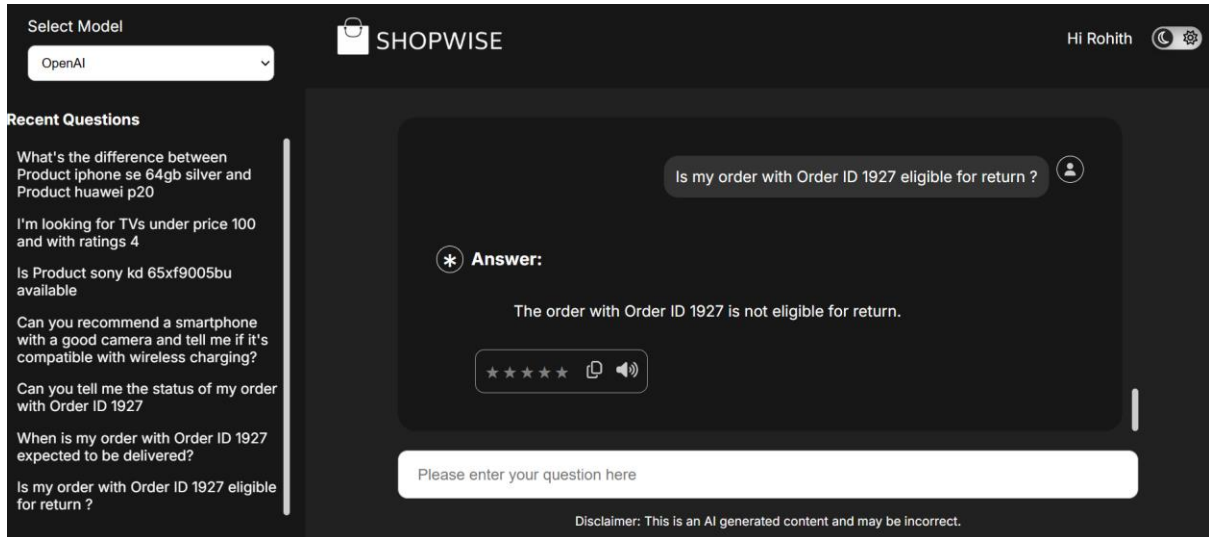
7. Tracking and Shipping Information

Query: "When is my order with Order ID 1927 expected to be delivered?"
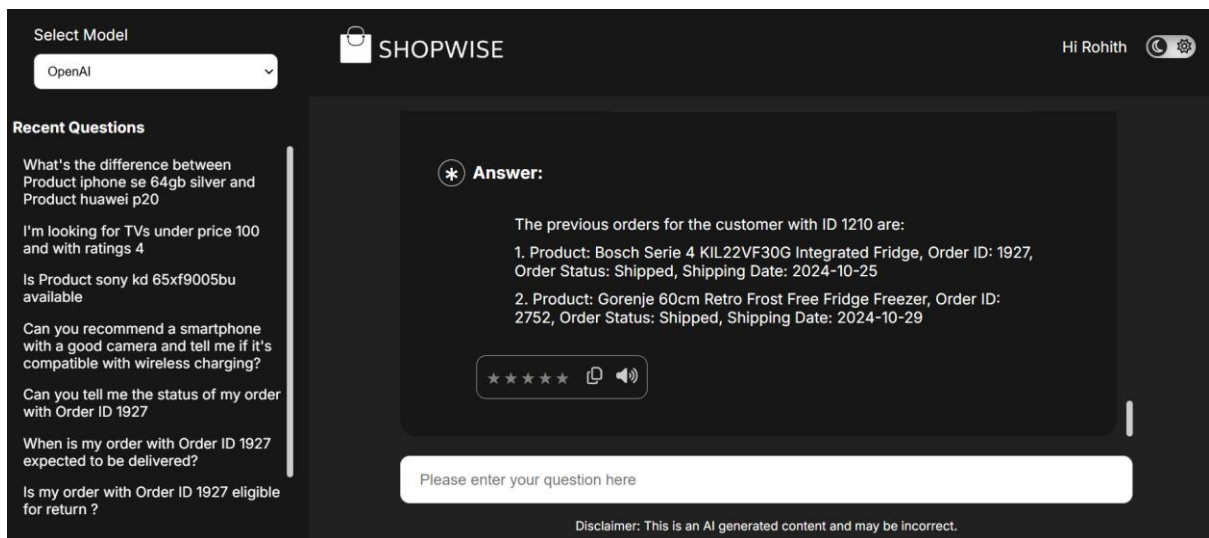

8. Return Eligibility

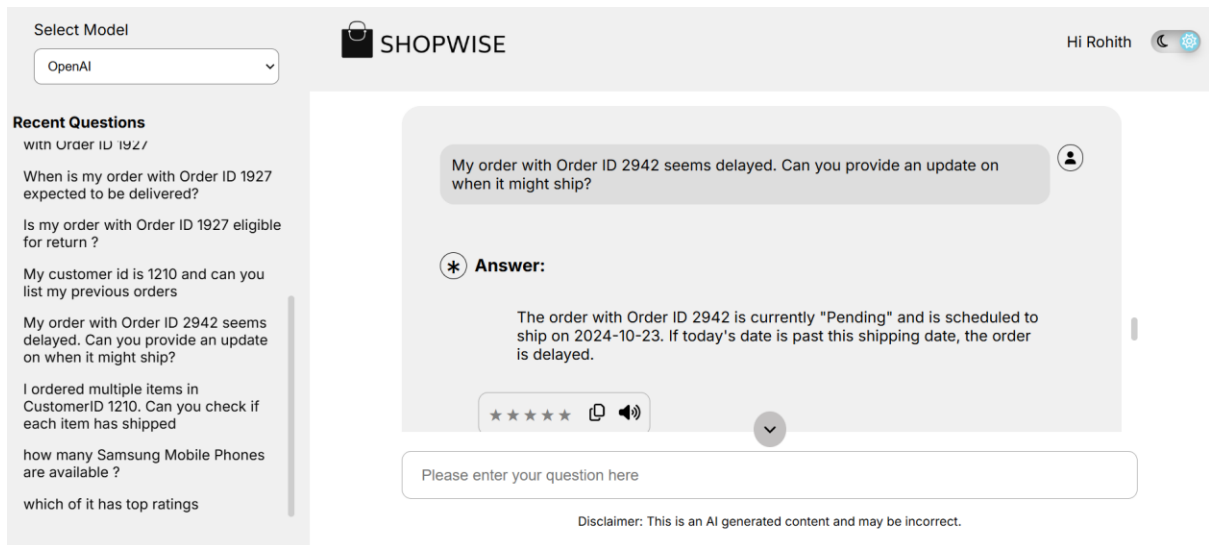Query: "Is my order with Order ID 1927 eligible for return?"

9. Order History

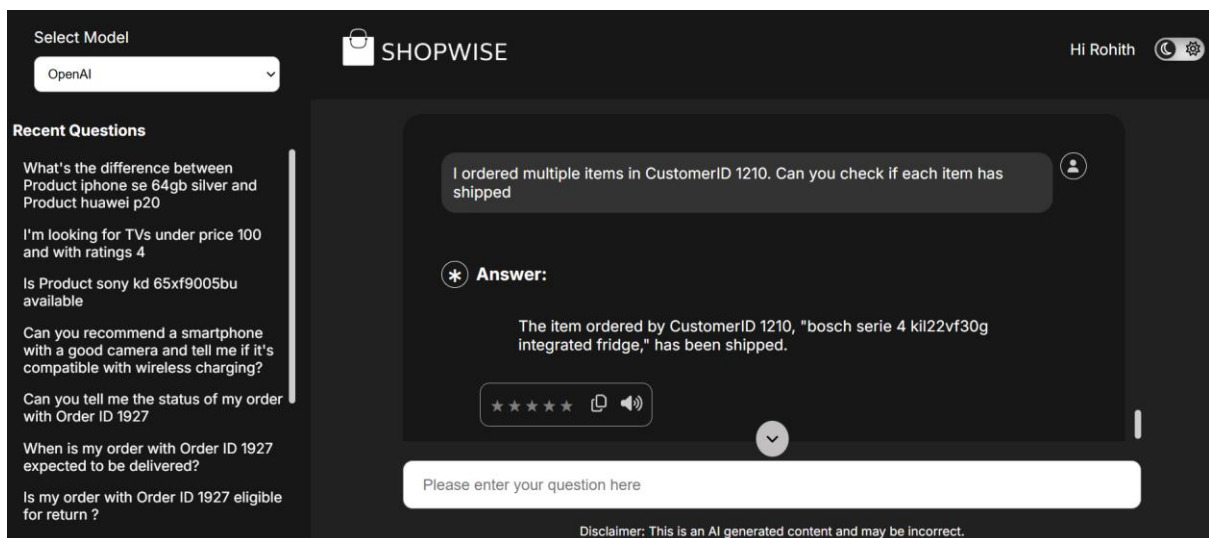Query: "My customer id is 1210 and can you list my previous orders"

10. Shipping Delay Inquiry

Query: "My order with Order ID 2942 seems delayed. Can you provide an update on when it might ship?"

11. Order Assistance for Multiple Items

Query: "I ordered multiple items in CustomerID 1210. Can you check if each item has shipped"

Future RoadMap:

1) Apart from current OpenAI llm, other alternative llms can be used by having a dropdown containing all llms in header section.
2) Dashboard can be created for creating graphs and charts based on orders and products for data analysis.

THANK YOU