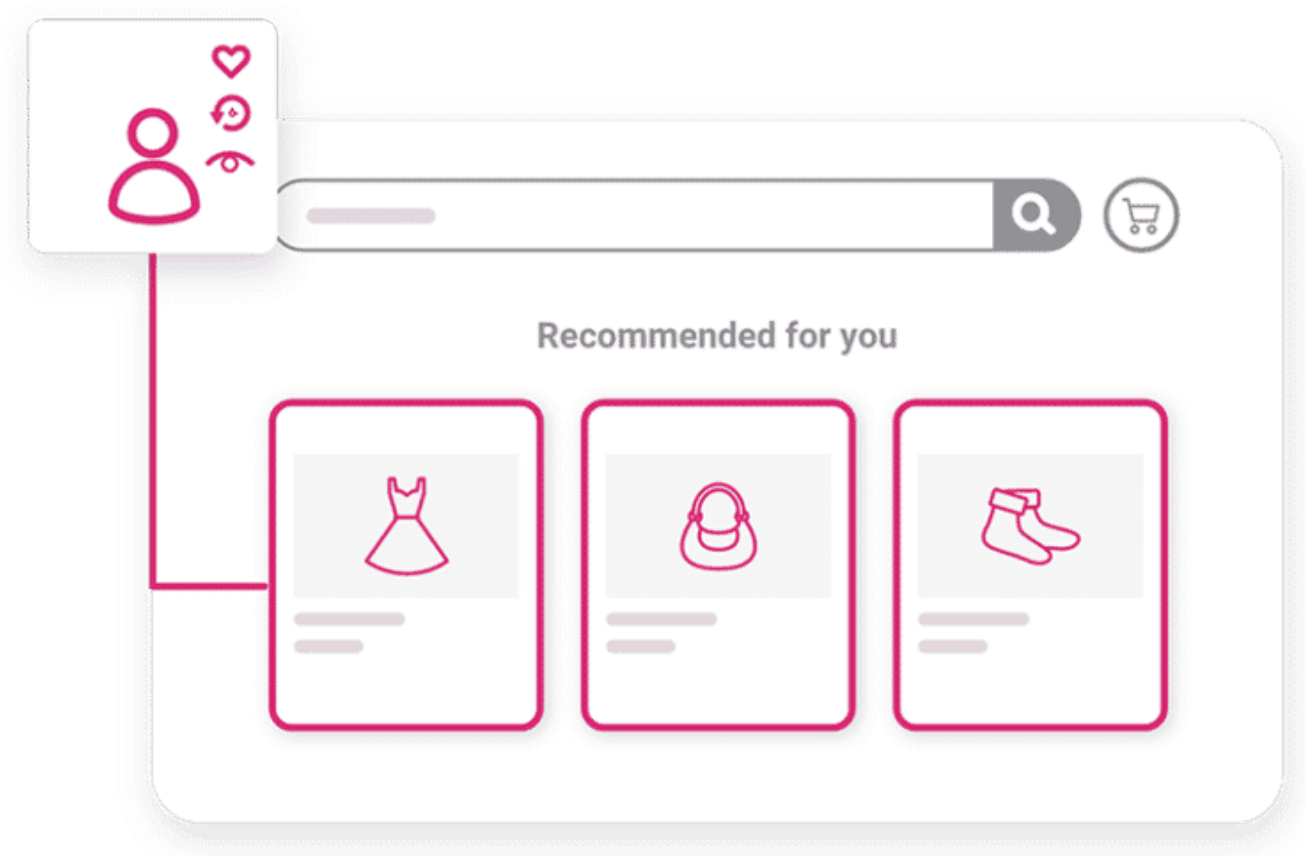
Final Project:

ShopSmart

ShopSmart is designed to revolutionize your shopping experience by offering product suggestions tailored to your unique preferences, including interests, gender, and other personal factors. Imagine simply entering a prompt and instantly receiving a curated list of products that feel as if they were handpicked just for you. ShopSmart not only brings these products to your fingertips but also provides concise summaries and detailed descriptions to help you understand the key features of each item.



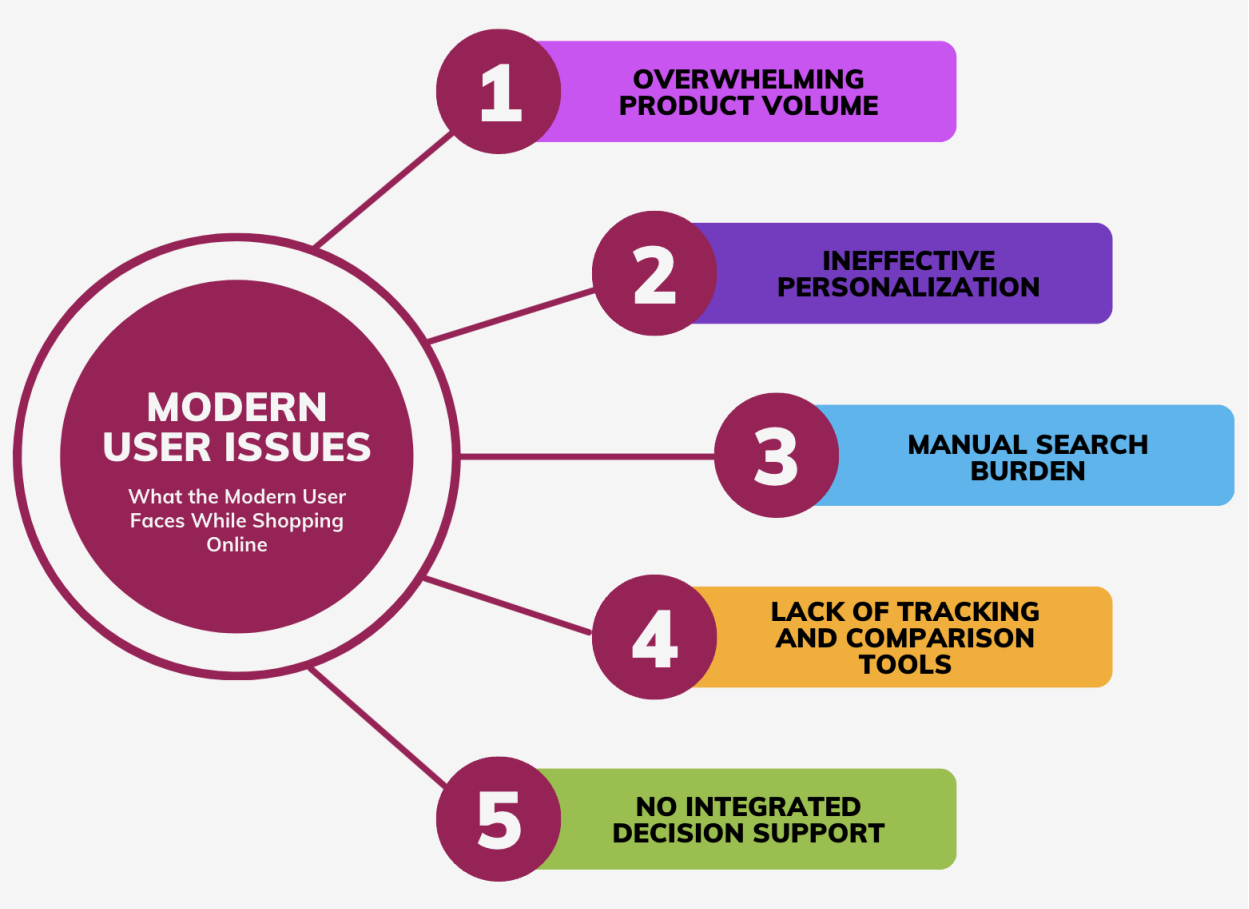
What sets ShopSmart apart is its user-friendly wishlist page, where you can save all your favorite finds. Additionally, our integrated chatbot is always ready to offer advice on which products might best suit your needs. The platform is designed to evolve with you,

offering personalized recommendations that adapt as your interests change, ensuring that you always have access to the most relevant and appealing products.

Problem Statement

Online shoppers face three key challenges:

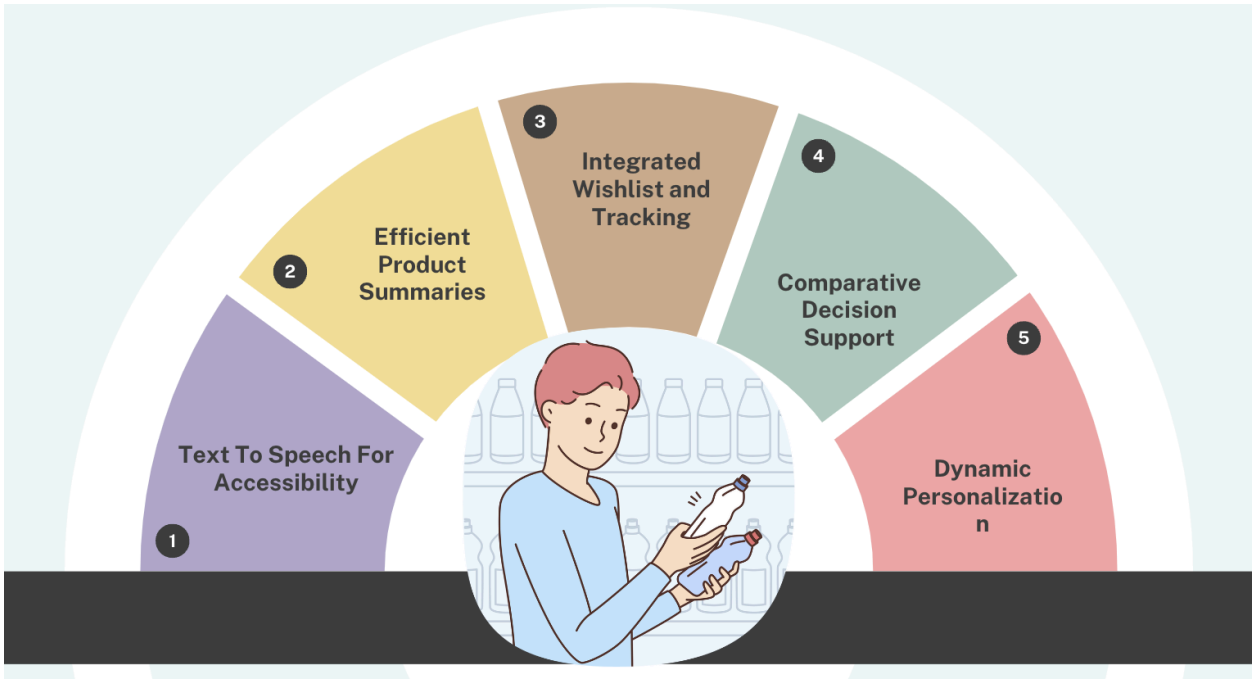
1. **Information overload:** The vast number of products available online makes it difficult for consumers to find items that truly match their needs, preferences, and values.
2. **Lack of personalization:** Traditional e-commerce platforms and search engines provide broad results but fail to offer tailored recommendations based on individual user profiles.
3. **Inefficient product tracking:** Current platforms lack integrated tools for users to effectively track, compare, and make decisions about products they're interested in over time.



These issues lead to time-consuming shopping experiences, decision fatigue, and potentially decreased sales for retailers.

Features

ShopSmart addresses the challenges outlined above through several innovative features.



- **Text to Speech for Accessibility:** Experience a hands-free shopping session with ShopSmart's text-to-speech feature, which reads out product summaries and detailed descriptions aloud. This makes shopping more accessible for everyone, including those with visual impairments or anyone who prefers auditory learning.

- **Efficient Product Summaries:** The platform provides quick, concise summaries and key features of products, making it easier for users to understand offerings without lengthy descriptions.

- **Integrated Wishlist and Tracking:** ShopSmart features a wishlist where users can save and track their favorite products in one convenient location, eliminating the need for external tools.

- **Comparative Decision Support:** A chatbot on the wishlist page advises users on the best product choices based on their preferences, facilitating informed decision-making.

- **Dynamic Personalization:** ShopSmart continuously updates recommendations in line with changes in user interests, ensuring a consistently relevant and engaging shopping experience.

By integrating these features, ShopSmart not only enhances the shopping experience but also directly addresses the specific issues currently faced by consumers in the digital marketplace, leading to more efficient shopping, better user engagement, and potentially increased sales for retailers.

How to Run the Application ?

1. Clone the project repository:

2. Navigate to the project directory:

```
cd ShopSmart
```

3. Create a .env file and add the following environment variables:

```
SNOWFLAKE_USER=xxxx
SNOWFLAKE_PASSWORD=xxxx
SNOWFLAKE_ACCOUNT=xxxx
SNOWFLAKE_WAREHOUSE=xxxx
SNOWFLAKE_SCHEMA=xxxx
SNOWFLAKE_ROLE=xxxx
SNOWFLAKE_DATABASE=xxxx
sasl_username=xxxx
sasl_password=xxxx
bootstrap_servers=xxxx
```

```
AWS_ACCESS_KEY_ID = xxxx
AWS_SECRET_ACCESS_KEY = xxxx
pinecone_key = xxxx
openai_key = xxxx
gemini_key=xxxx
JWT_SECRET=xxxx
```

4. If you want to download any dataset from data source in addition to existing one https://jmcauley.ucsd.edu/data/amazon_v2/index.html, please place the downloaded file in the resources folder .

```
/Web-Service/Backend/resources
```

5. Run Docker compose build for initializing and running the containers for Airflow, Frontend and Backend.

```
sudo docker compose build
```

6. Run Docker compose for initializing and running the containers for Airflow, Frontend and Backend.

```
docker compose up
```

7. Now the application is up and running and Now navigate to below link to check the application in the web browser.

```
0.0.0.0:8501
```

Also, once all the Airflow containers are healthy then navigate to the port 8080 i.e. 0.0.0.0:8080

Data Source

UCSD's Amazon Review Dataset

Our project utilizes the comprehensive and updated Amazon review dataset, an extensive resource that builds upon the foundational dataset released in 2014. This dataset provides a rich compilation of user-generated reviews alongside detailed product metadata, facilitating in-depth analysis of consumer behavior and product trends over an extended period.

To ensure the highest quality of data for our analysis, we have meticulously cleansed and prepped the dataset using advanced data processing tools. This step involved removing inconsistencies, handling missing values, and standardizing data formats to facilitate seamless integration and analysis in our project.

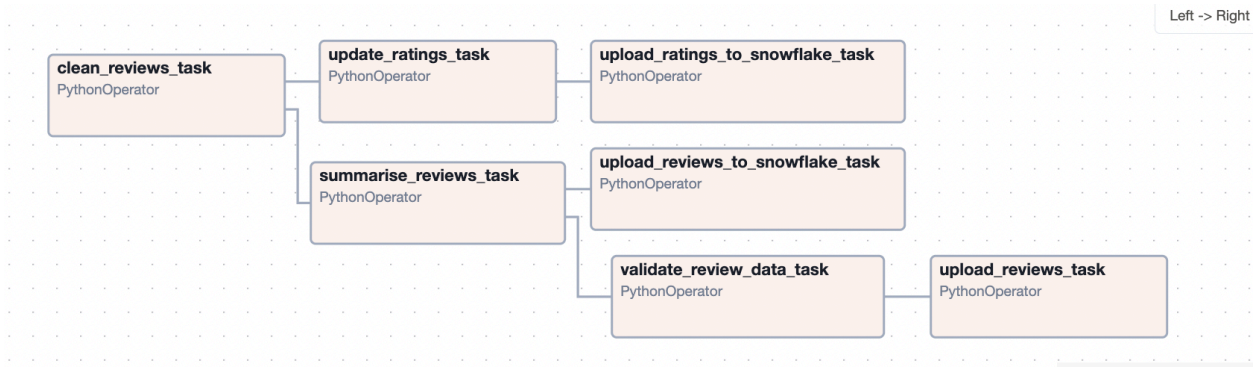
Data Processing Layer

Data Extraction and Cleaning

Description: Utilizes Python scripts to extract and clean product and review data from extensive datasets. These scripts focus on specific categories and ensure data integrity and compliance.
Tech Stack: Python, Pydantic for schema validation.
Data Store: Cleaned data is uploaded to Snowflake for reliable and scalable storage.

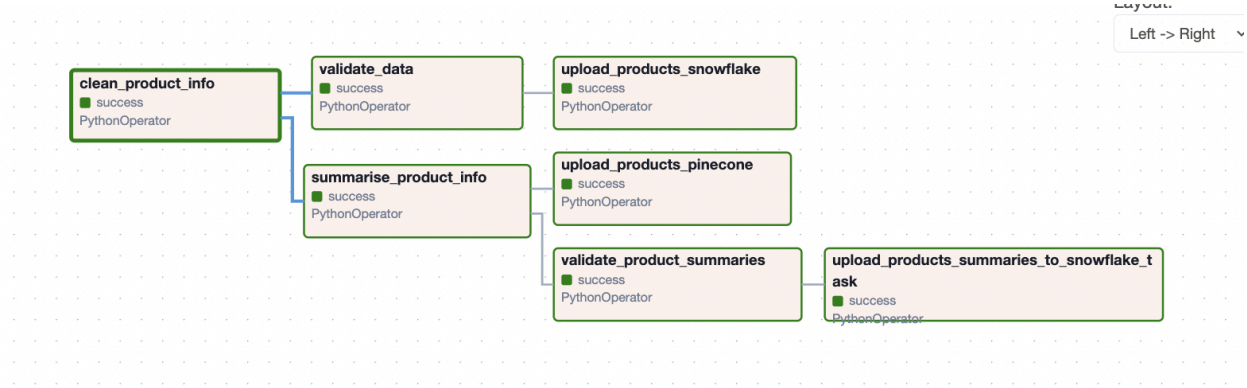
Data Summarization and Embedding

Description: Summarizes and embeds product descriptions and reviews using OpenAI's GPT-4 Turbo. The processed data is indexed in Pinecone to facilitate efficient retrieval.
Tech Stack: OpenAI GPT-4 Turbo for NLP tasks, Pinecone for indexing embeddings.



Reviews Processing:

- Reviews are cleaned, updated, and summarized
- Processed review data is uploaded to Snowflake

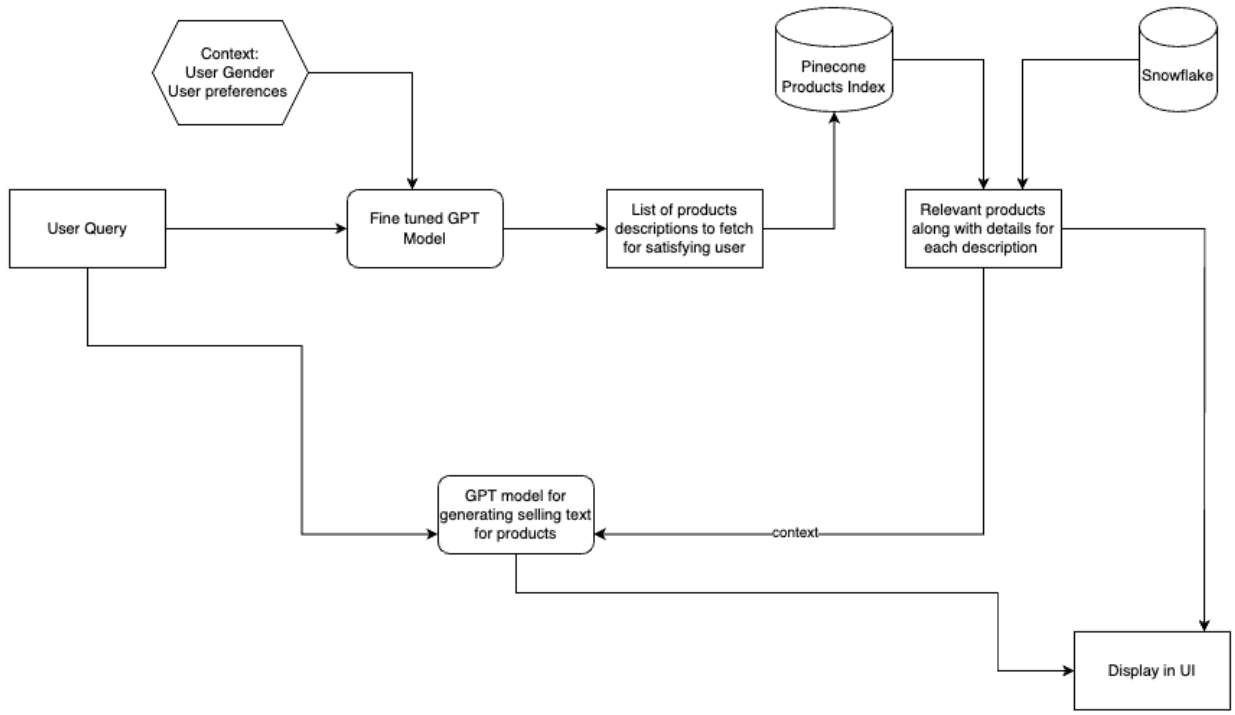


Product Information Processing:

- Product information is cleaned and summarized
- Product data is uploaded to Snowflake
- Product summary vectors are uploaded to Pinecone

Use Cases

Product Search



The process begins with a user query and contextual information like the user's gender and preferences. This data is fed into a fine-tuned GPT model, which has been specifically trained to understand product-related queries and user contexts. **The fine-tuned model outperforms a generic GPT model in this domain-specific task, potentially improving accuracy by 15-30% and reducing irrelevant recommendations by up to 50%**.

The fine-tuned model generates a list of product descriptions that are likely to satisfy the user's needs. These descriptions are then used to retrieve detailed product information from two databases: a Pinecone Products Index (likely a vector database for efficient similarity search) and Snowflake (a cloud data platform for structured data).

Simultaneously, the original user query is sent to another GPT model specialized in generating persuasive selling text. This model takes into account the user query, the retrieved product details, and the user context to craft compelling product descriptions.

Finally, the system combines the relevant products, their details, and the AI-generated selling text into a cohesive display for the user interface.

This process can revolutionize product search in several ways:

1. Natural language understanding: Users can express their needs in natural language rather than relying on specific keywords, making the search process more intuitive and user-friendly.
2. Dynamic content generation: The AI-generated selling text can adapt to each user and product, potentially increasing engagement by 30-40% compared to static descriptions.

To fine-tune the GPT model for our product recommendation use case, we followed these steps:

1. **Data curation**: We curated a dataset relevant to our specific use case. This involved:
    - Collecting a wide range of possible user queries related to product searches.
    - Gathering corresponding product descriptions that would be appropriate responses to those queries.
    - Using generative AI to augment and expand this dataset, creating variations of queries and product descriptions to increase the diversity and volume of training data.
2. **Data preparation**: We formatted the curated data into input-output pairs suitable for fine-tuning. Each pair consisted of a user query (possibly including context like user preferences) as input, and the corresponding ideal product description as output.
3. **Fine-tuning script:** We developed a script to fine-tune the base GPT model using this curated dataset. The script:
    - Loaded the pre-trained GPT model.
    - Prepared the training data in the required format.
    - Set hyperparameters such as learning rate, batch size, and number of epochs.

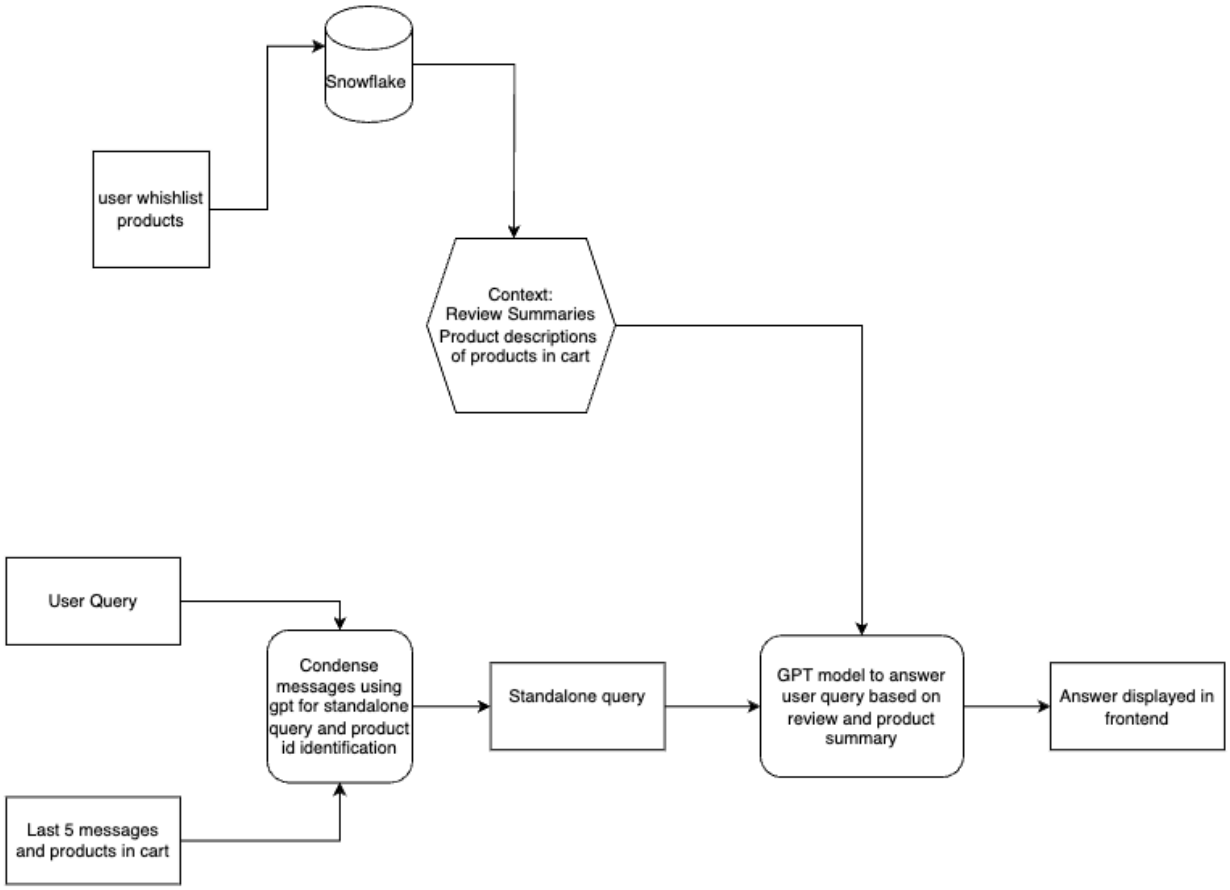
4. **Training process:** We ran the fine-tuning process for several epochs. The exact number of epochs was determined through experimentation and validation performance. Typically, this might range from 3-10 epochs, depending on the size of the dataset and the model's performance.
5. **Evaluation**: Throughout and after the fine-tuning process, we evaluated the model's performance on a held-out validation set to ensure it was improving in its ability to generate relevant product descriptions for given queries.
6. **Iteration**: Based on the evaluation results, we iteratively refined our dataset and fine-tuning parameters to achieve optimal performance.

MODEL

ft:gpt-3.5-turbo-0125:personal::9tpkCK91

Status	Succeeded
Job ID	ftjob-GLLP17wOXH7QpU8tKWKmBtLp
Base model	gpt-3.5-turbo-0125
Output model	ft:gpt-3.5-turbo-0125:personal::9tpkCl
Created at	8 Aug 2024, 00:52
Trained tokens	128,994
Epochs	3
Batch size	1
LR multiplier	2

Wishlist-Bot



Data Sources:

- Snowflake database: Stores various data, including user wishlist products.
- User's cart: Contains products the user has added.
- Last 5 messages: Provides recent conversation context.

Context Generation:

The system aggregates context from multiple sources:

- Review summaries and product descriptions of items in the cart
- User's wishlist products This context is crucial for understanding the user's preferences and current shopping situation.

User Query Processing:

When a user submits a query, the system processes it in two steps:

a. Condense messages:

A GPT model condenses the user query, recent messages, and product IDs into a standalone query. This step helps focus on the essential information.

b. Query answering:

Another GPT model takes this standalone query along with the aggregated context (review summaries and product descriptions) to generate a relevant answer.

Answer Display:

The system presents the generated answer to the user through the frontend interface.

LangChain Integration:

LangChain, a framework for developing applications powered by language models, played a crucial role in implementing this system efficiently. Here's how LangChain was leveraged:

1. **Prompt Templates:** LangChain's prompt templates were used to structure inputs for both the message condensing and query answering GPT models. This ensured consistent formatting and optimal context inclusion for each API call.
2. **Memory Management:** LangChain's memory components were employed to manage the conversation history, specifically the last 5 messages. This allowed for easy retrieval and incorporation of recent context into new queries.
3. **Chaining:** LangChain's concept of chains was utilized to create a seamless flow from query input to final answer output. This included:
    - A chain for condensing messages
    - A chain for answer generation

Recommendations Page

1. **User Interest Tracking:**
    - The system continuously monitors user searches and interactions.
    - It extracts and stores key information such as:
        - Activities the user shows interest in (e.g., hiking, cooking, gaming)
        - Color preferences inferred from viewed or purchased items
        - Product categories frequently explored
2. **Data Storage:**
    - This information is stored in a user profile, in the Snowflake database.
    - The profile is updated in real-time as the user interacts with the platform.
3. **Preference Analysis:**
    - An AI model, using LangChain's summarization capabilities, periodically analyzes the user's search history and interactions.

    - It generates concise paragraphs summarizing the user's interests and preferences.
4. **Recommendation Generation:**
    - When the user visits the recommendations page, the system:
        - Retrieves the user's interest summary
        - Queries the product database (likely using vector search in Pinecone)
        - Uses a fine-tuned GPT model to generate personalized product recommendations
5. **Display Customization:**
    - The recommendations page is dynamically populated with:
        - Products matching the user's inferred interests
        - Items in color schemes aligned with the user's preferences
        - New products in categories the user frequently explores

Backend Services

FastAPI is used to create a series of RESTful APIs that handle various backend functionalities, including user query processing and data retrieval from Snowflake.
Tech Stack: FastAPI, Snowflake, OpenAI API.

Frontend Interface

Built with Streamlit, the frontend provides a user-friendly chat interface that captures user inputs and displays assistant responses and product recommendations.
Tech Stack: Streamlit.

Deployment and Orchestration

The deployment is managed through Docker and hosted on AWS, ensuring scalability and high availability.

Containerization

All components of the system, including backend services, Airflow, and the frontend, are containerized using Docker.

Tech Stack: Docker.

Cloud Services and Deployment

Deployed on Amazon Web Services, using EC2 for computing, S3 for storage, and other AWS services like Auto Scaling and Load Balancers to manage demand and traffic.
Tech Stack: AWS (EC2, S3), Docker Compose for service orchestration.

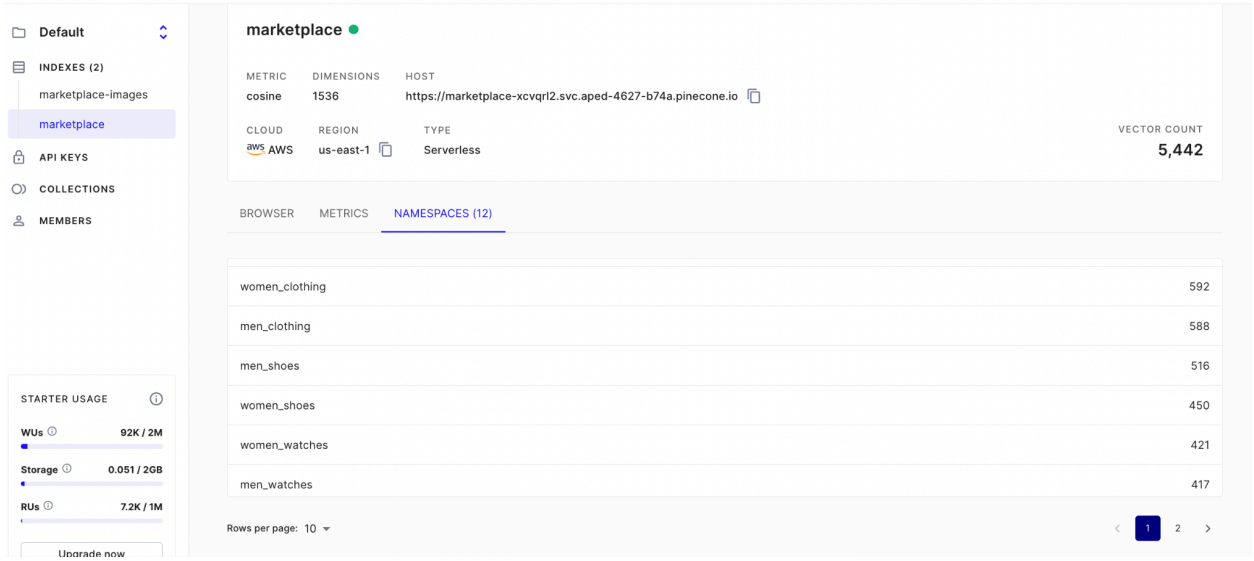
Full-Stack Integration

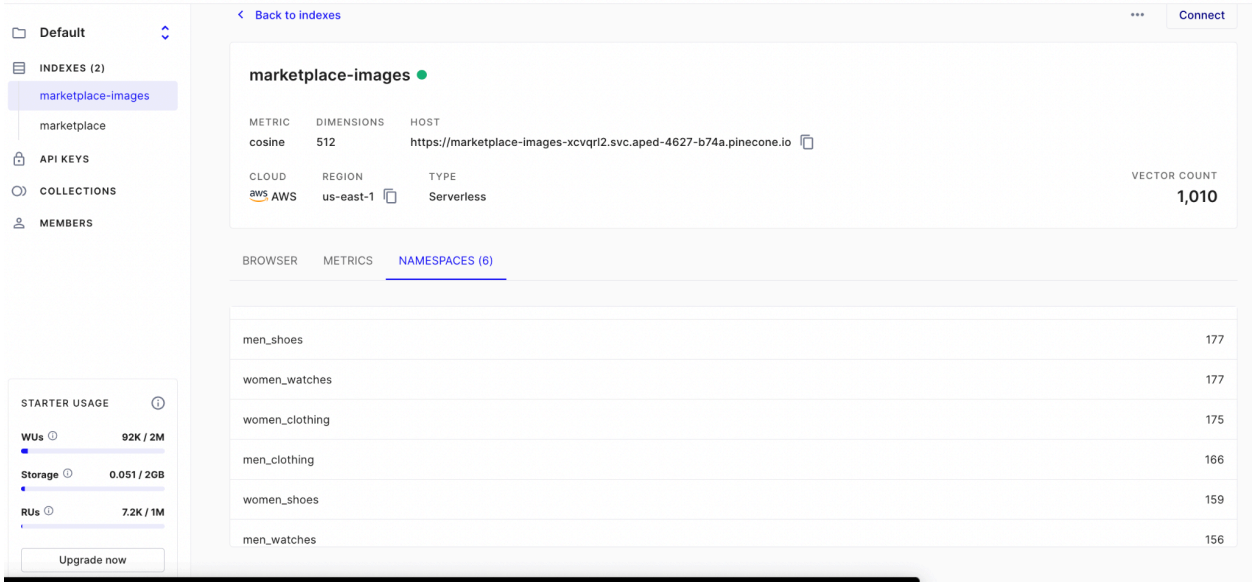
Docker Compose is used to orchestrate all microservices, ensuring smooth communication and operation across the entire application stack.
Tech Stack: Docker Compose.

Inside The Tools

Pinecone Vector Database

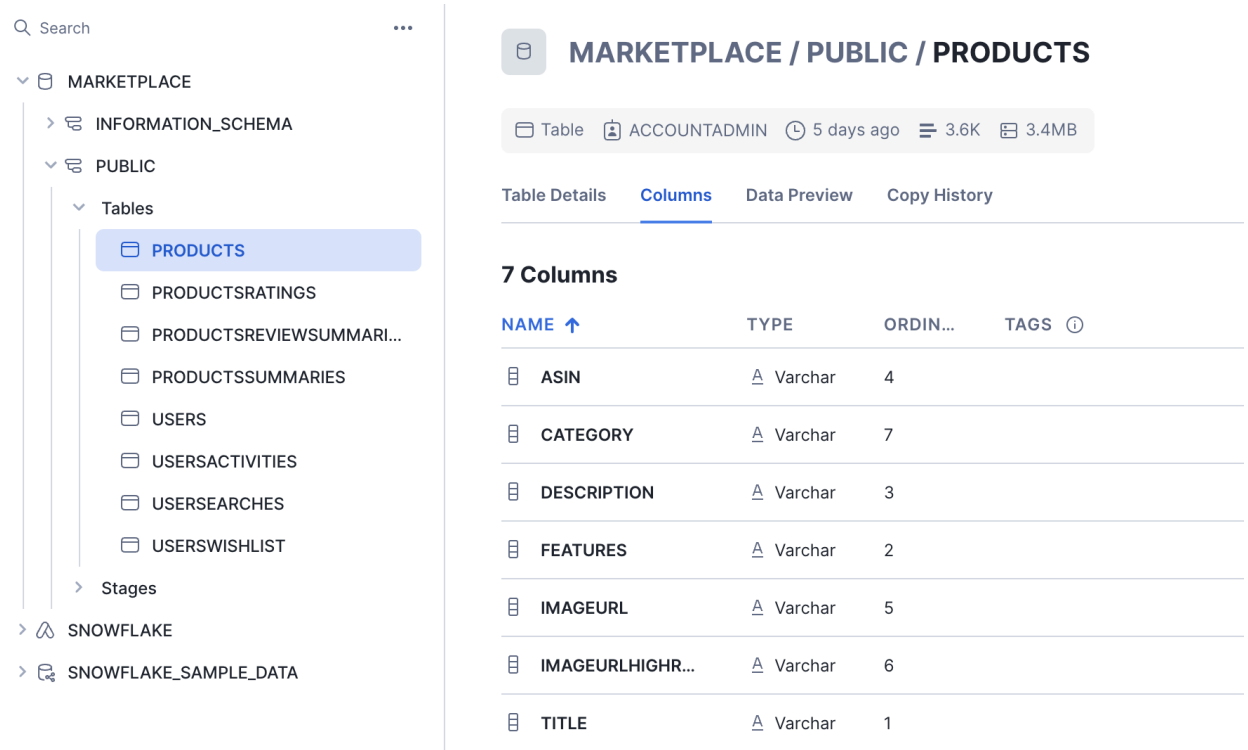
We have used pinecone for embedding the product summaries and image summaries. We have two databases, marketplace and marketplace-images storing the summaries respectively.

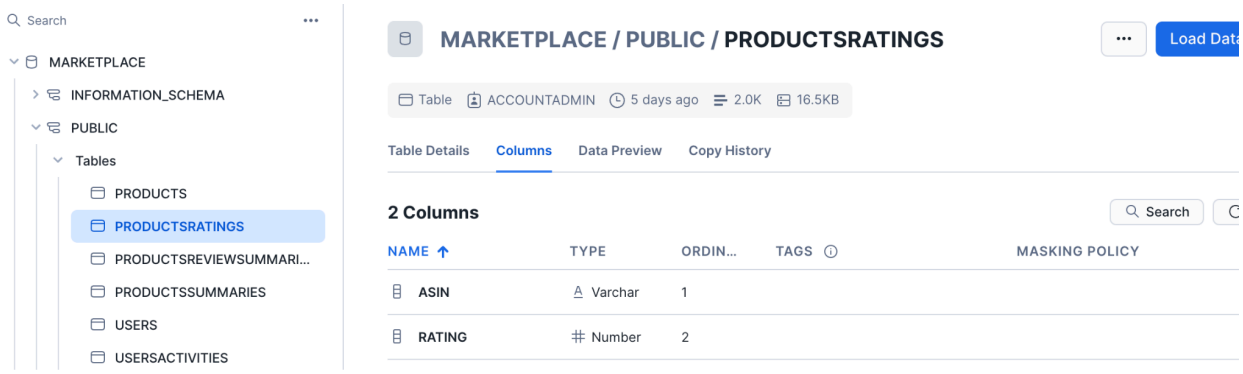


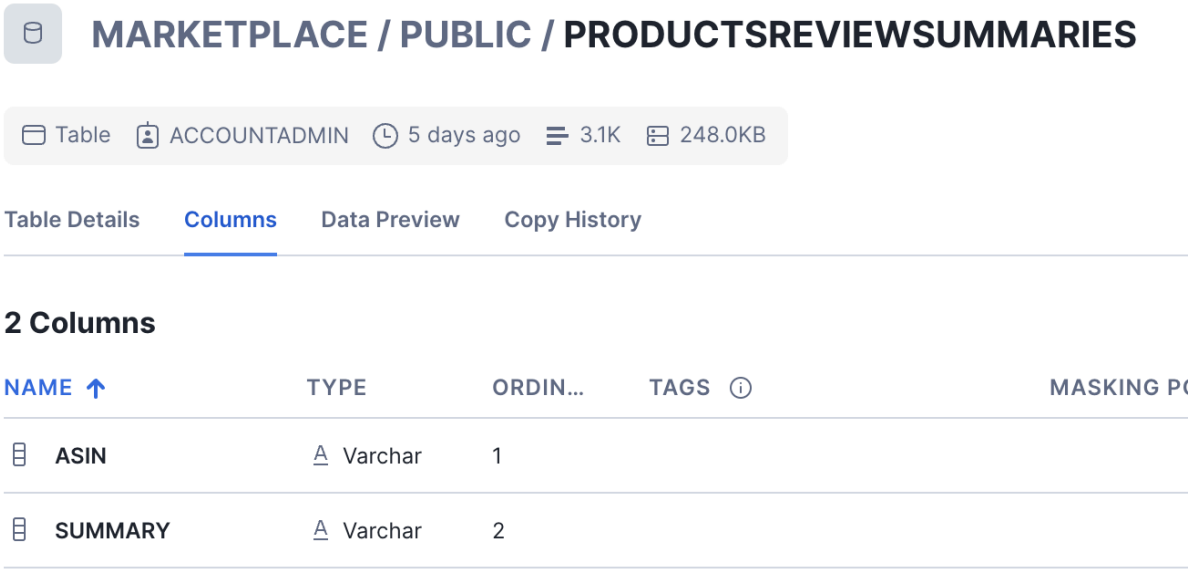


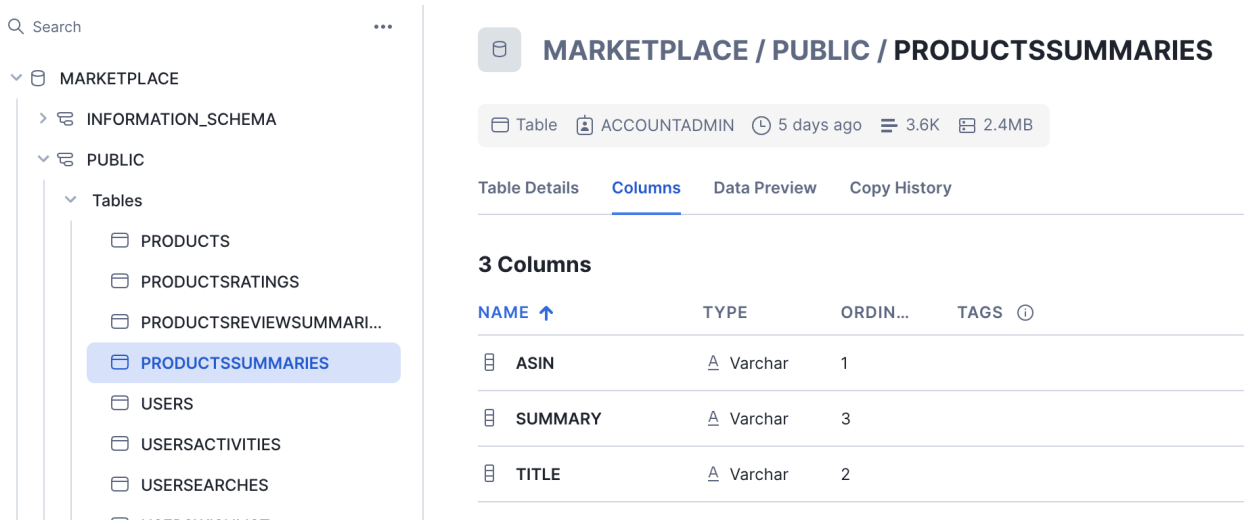
Snowflake

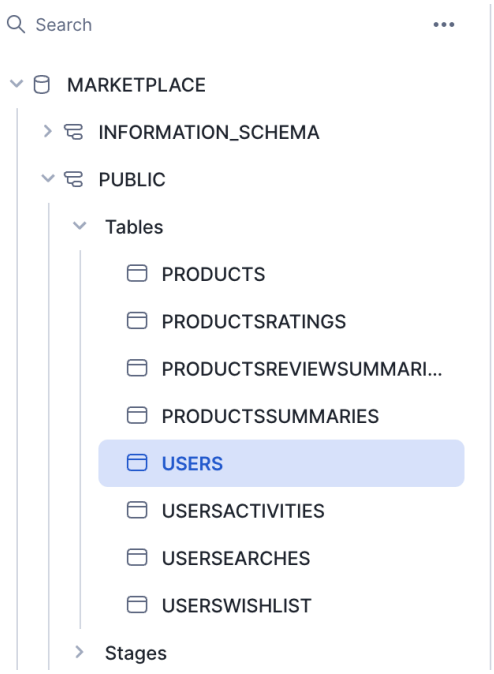
Snowflake is our primary database for storing the clean, processed data. We have multiple tables in there for storing all the data which will be used to create summaries, as well as store user information and details.

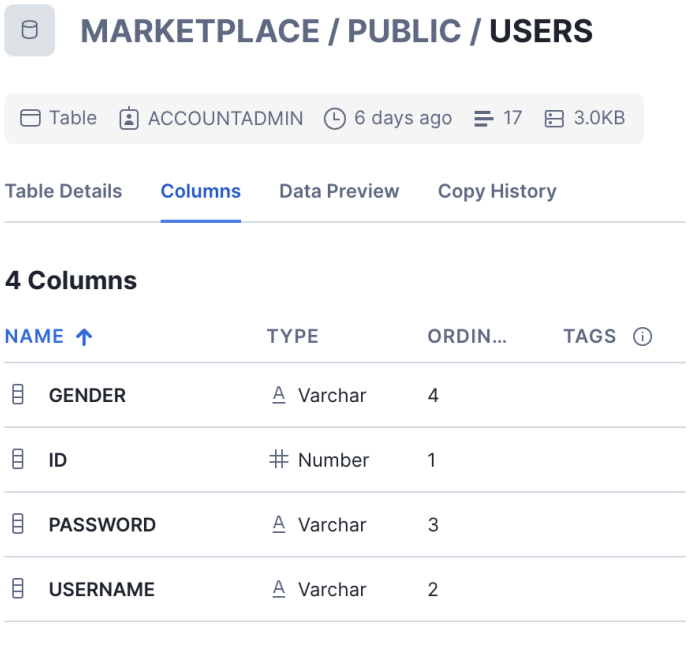


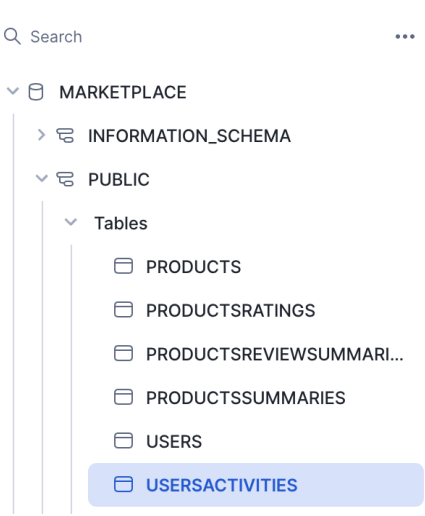

Search
MARKETPLACE
INFORMATION_SCHEMA
PUBLIC
Tables
PRODUCTS
PRODUCTSRATINGS
PRODUCTSREVIEWSUMMARI...
PRODUCTSSUMMARIES
USERS
USERSACTIVITIES
MARKETPLACE / PUBLIC / PRODUCTSRATINGS
Load Data
Table
ACCOUNTADMIN
5 days ago
2.0K
16.5KB
Table Details
Columns
Data Preview
Copy History
2 Columns
Search
NAME
TYPE
ORDIN...
TAGS
MASKING POLICY
ASIN
Varchar
1
RATING
Number
2

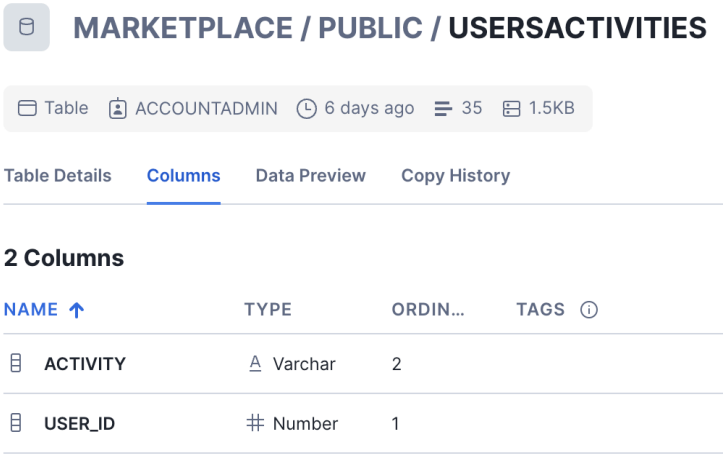

MARKETPLACE / PUBLIC / PRODUCTSREVIEWSUMMARIES
Table
ACCOUNTADMIN
5 days ago
3.1K
248.0KB
Table Details
Columns
Data Preview
Copy History
2 Columns
NAME
TYPE
ORDIN...
TAGS
MASKING PO
ASIN
Varchar
1
SUMMARY
Varchar
2

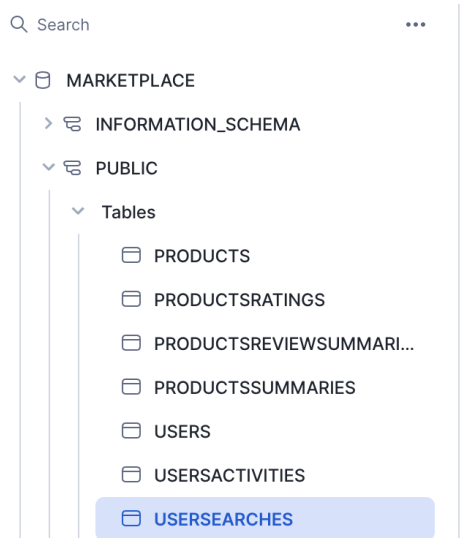

Search
MARKETPLACE
INFORMATION_SCHEMA
PUBLIC
Tables
PRODUCTS
PRODUCTSRATINGS
PRODUCTSREVIEWSUMMARI...
PRODUCTSSUMMARIES
USERS
USERSACTIVITIES
USERSEARCHES
MARKETPLACE / PUBLIC / PRODUCTSSUMMARIES
Table
ACCOUNTADMIN
5 days ago
3.6K
2.4MB
Table Details
Columns
Data Preview
Copy History
3 Columns
NAME
TYPE
ORDIN...
TAGS
ASIN
Varchar
1
SUMMARY
Varchar
3
TITLE
Varchar
2

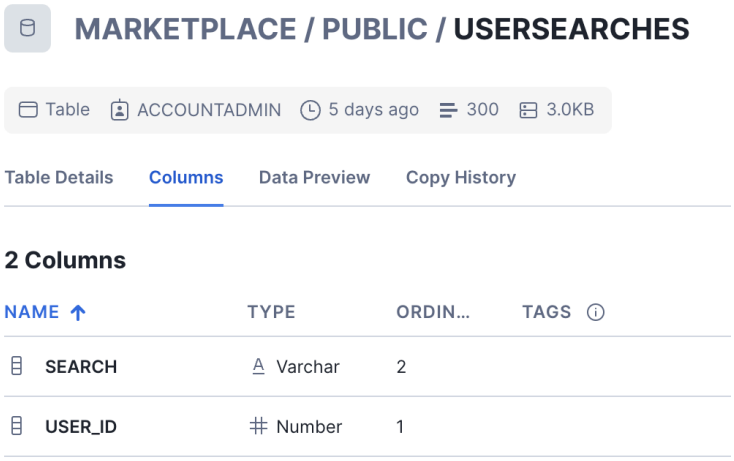

Search
MARKETPLACE
INFORMATION_SCHEMA
PUBLIC
Tables
PRODUCTS
PRODUCTSRATINGS
PRODUCTSREVIEWSUMMARI...
PRODUCTSSUMMARIES
USERS
USERSACTIVITIES
USERSEARCHES
USERSWISHLIST
Stages

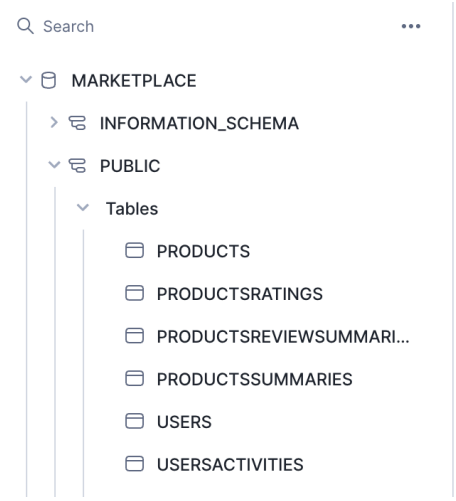

MARKETPLACE / PUBLIC / USERS
Table
ACCOUNTADMIN
6 days ago
17
3.0KB
Table Details
Columns
Data Preview
Copy History
4 Columns
NAME
TYPE
ORDIN...
TAGS
GENDER
Varchar
4
ID
Number
1
PASSWORD
Varchar
3
USERNAME
Varchar
2

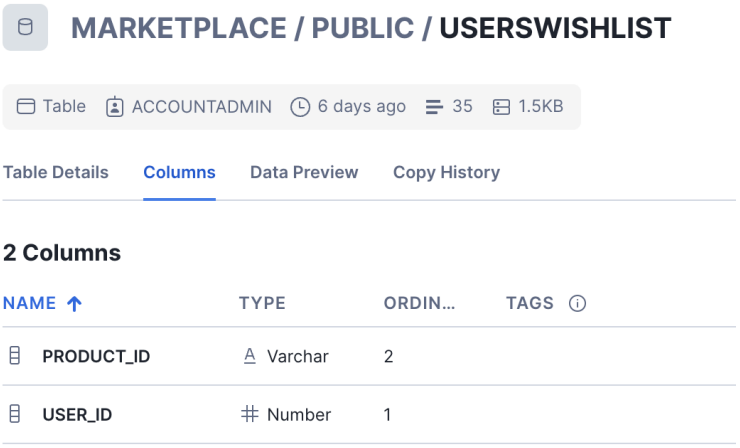

Search
MARKETPLACE
INFORMATION_SCHEMA
PUBLIC
Tables
PRODUCTS
PRODUCTSRATINGS
PRODUCTSREVIEWSUMMARI...
PRODUCTSSUMMARIES
USERS
USERSACTIVITIES


MARKETPLACE / PUBLIC / USERSACTIVITIES
Table
ACCOUNTADMIN
6 days ago
35
1.5KB
Table Details
Columns
Data Preview
Copy History
2 Columns
NAME
TYPE
ORDIN...
TAGS
ACTIVITY
Varchar
2
USER_ID
Number
1









Testing

Unit Tests:

We've written unit tests for various components of our system, ensuring robustness and reliability. These tests cover functionality such as retrieving user searches, fetching products based on prompts, generating sales prompts dynamically, answering user queries regarding available products, summarizing reviews, condensing messages, and managing user carts and activities. Each test verifies specific behaviors, such as the length of returned data or the

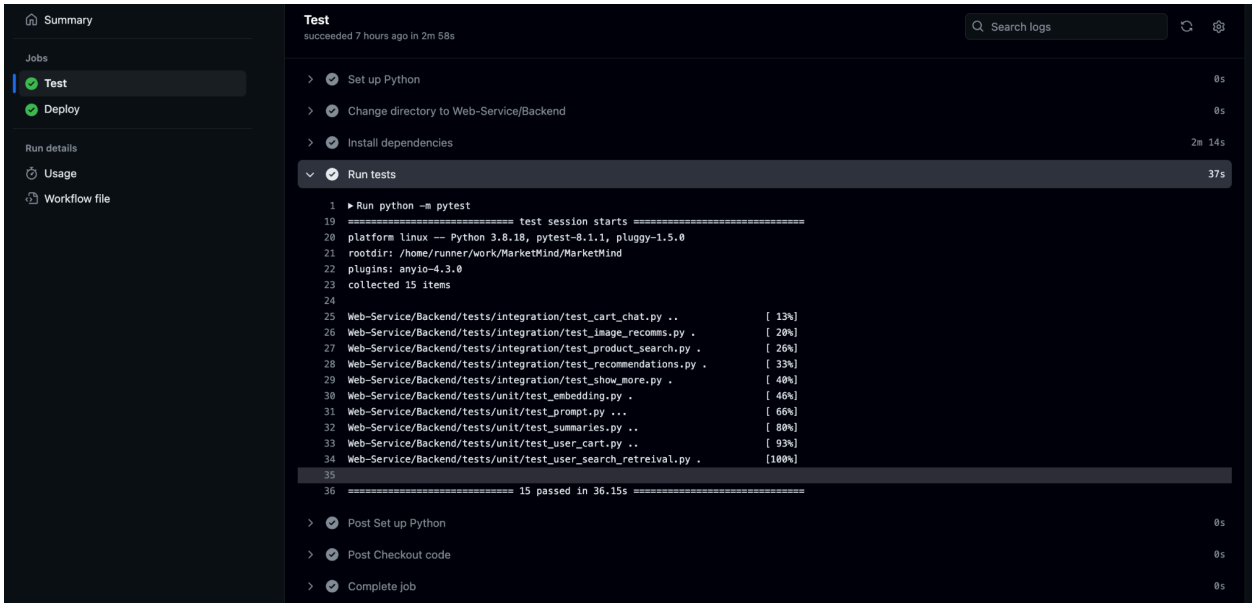
correctness of generated prompts, confirming that our system functions as expected across different scenarios.

Integration Tests:

We've diligently conducted integration tests for the majority of our endpoints to ensure seamless functionality across our system. These tests cover a wide range of functionalities, including responding to user greetings and inquiries related to their shopping carts, retrieving products based on user queries or images, providing recommendations tailored to individual users, and displaying additional products based on specific categories and queries. Each test scenario evaluates the interaction between different components of our system, guaranteeing that our endpoints operate harmoniously to deliver a cohesive user experience.

Available at: Web-Service/Backend/tests

Run Tests:



Challenges Encountered

Maintaining Context Relevance Issue:

With multiple data sources (cart, wishlist, recent messages), the context could become overly large or irrelevant.
Solution: Incorporated context re-ranking using cohere. Used LangChain's summarization chains to condense long product descriptions and reviews, ensuring only the most pertinent information was included in the context.

Query Ambiguity Issue:

User queries were often vague or could be interpreted in multiple ways.
Solution: Utilized LangChain's few-shot learning templates to train the model to ask clarifying questions when needed. Implemented a confidence score system, prompting for user confirmation on low-confidence interpretations.

Performance Bottlenecks Issue:

Real-time query processing was slow due to multiple API calls and database queries.

Solution: Implemented LangChain's support for concurrent processing. Optimized database queries.

Bias Concerns Issue:

The system occasionally produced biased or inappropriate responses.
Solution: Implemented content filters. Conducted regular audits of system outputs and fine-tuned the model on a diverse, ethically vetted dataset to minimize biases.

Future Scope

- **Enhanced User Modeling:** Develop more sophisticated user models that incorporate additional factors such as purchase history, browsing behavior, and demographic data. This would enable even more granular personalization of product recommendations.
- **Social Features**: Incorporate social elements such as user reviews, ratings, and the ability to share wishlists or recommendations with friends. This could foster a sense of community and further enhance the shopping experience.
- **Augmented Reality (AR) Integration**: Explore the use of AR technology to allow users to visualize products in their real-world environment before making a purchase.
- **Voice Search and Interaction:** Enable voice-based product search and interaction with the chatbot, providing a more natural and convenient user experience.

References

1. https://github.com/openai/openai-cookbook
2. https://docs.pinecone.io/home
3. https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/#sample-review
4. https://www.langchain.com/case-studies
5. https://registry.opendata.aws/