AI-Powered Product Recommendation Engine

Project Overview:

This project is an AI-powered product recommendation engine developed for eCommerce platforms. The system utilizes Large Language Models (LLMs) to generate personalized product recommendations based on user preferences (such as price range, categories, and brands) and browsing history. The backend is implemented with FastAPI, and the frontend is developed using React.

The engine integrates meta-llama/Meta-Llama-3.1-8B-Instruct-Turbo, a robust LLM model, to generate context-aware and relevant recommendations for users. This model, part of the Meta-Llama series, leverages transformer architecture and instruction-tuning, enabling it to interpret and respond to diverse user input more effectively.

Features:

- **Personalized Product Recommendations:** The system generates product recommendations tailored to user preferences and browsing history by leveraging AI models and filtering techniques.
- **User Preferences Form:** Users can input their preferences, including price range, categories, and brands.
- **Product Catalog:** Displays a complete list of products across various categories with detailed information.
- **Browsing History Tracking:** Tracks products users have viewed to influence recommendations.

Setup Instructions:

Backend Setup

1. **Clone the repository:**

   git clone <your-repository-url>

   cd backend
2. **Create a virtual environment:**

   python -m venv venv

   venv\Scripts\activate
3. **Install dependencies:**

   pip install -r requirements.txt
4. **Configure environment variables:**

   TOGETHER_API_KEY=your-api-key

   MODEL_NAME=meta-llama/Meta-Llama-3.1-8B-Instruct-Turbo

   MAX_TOKENS=500

   TEMPERATURE=0.7

   DATA_PATH=data/products.json
5. **Run the FastAPI server:**

   uvicorn app:app –reload

Frontend Setup

1. **Navigate to the frontend directory:**

   cd frontend
2. **Install dependencies:**

   npm install
3. **Start the React development server:**

   npm start

API Documentation:

GET /api/products

Description: Fetches the complete product catalog.

Response: Returns an array of product objects, including details like id, name, category, price, brand, etc.

POST /api/recommendations

Description: Generates personalized product recommendations based on user preferences and browsing history.

Request Body


```
{
  "preferences": {
    "priceRange": "0-50",
    "categories": ["Electronics", "Clothing"],
    "brands": ["Nike", "Apple"]
  },
  "browsing_history": ["product123", "product456"]
}
```


Response


```
{
  "recommendations": [
    {
      "product": {
        "id": "product123",
        "name": "Ultra-Comfort Running Shoes",
        "category": "Footwear",
        "price": 89.99,
        "brand": "Nike"
      },
      "explanation": "These shoes are within your preferred price range and match your browsing history.",
      "confidence_score": 9
    }
  ]
}
```

How to Use the Application:

1. **Setting Preferences**: Users can select price range, categories, and brands from the "Your Preferences" section.
2. **Browsing Products**: Users can browse through the product catalog, and their selected items are added to their browsing history.
3. **Getting Recommendations**: Once preferences and browsing history are set, users can click "Get Personalized Recommendations" to view AI-generated suggestions.

Testing:

To run the test cases for backend functionality, execute: **pytest test.py**

This will run the tests and provide feedback on the API's correctness, covering endpoints such as /api/products and /api/recommendations.

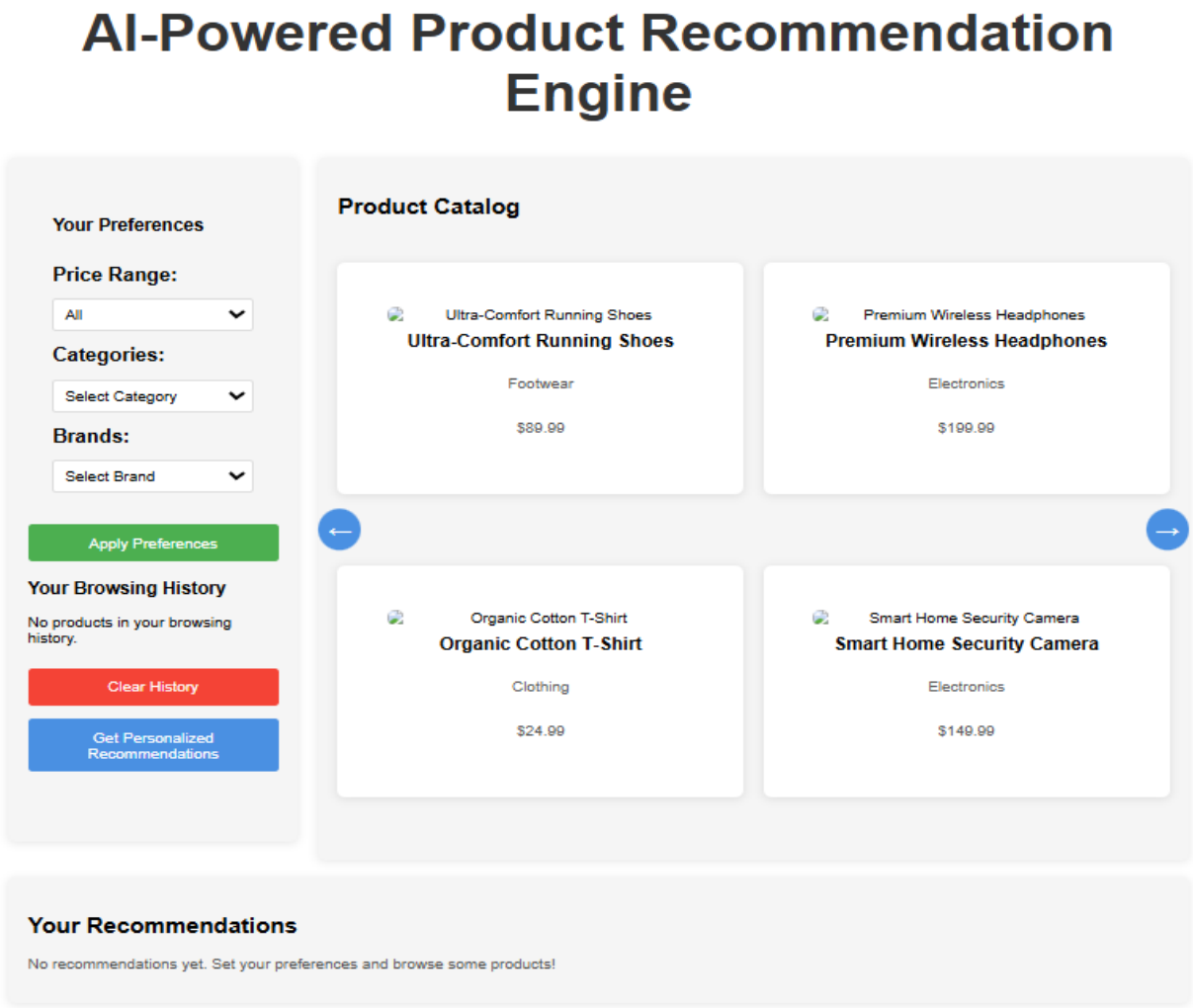
Challenges Faced:

- **Efficient Integration:** Integrating the LLM with backend services to ensure seamless data flow between product catalog filtering, user preferences, and browsing history.
- **Understanding Large Language Models (LLMs)**: As a beginner, comprehending how LLMs work and how they can be effectively used for tasks like product recommendation was initially challenging. I had to familiarize myself with the prompt engineering, the importance of token limits, and how the model interprets and generates responses.

Time Spent on the Assignment:

- **Backend Development:** 3 hours
- **Frontend Development:** 2 hours
- **Testing and Debugging:** 1 hour
- **Documentation:** 1 hour

Results:



Conclusion:

This AI-powered recommendation engine effectively combines user preferences and browsing history with a Meta-Llama-based LLM model for generating personalized product recommendations. The system is designed to enhance user experience on eCommerce platforms by offering real-time, relevant product suggestions. Challenges such as prompt engineering and token management were addressed with thoughtful optimization. The application is fully functional, and its API endpoints have been tested for accuracy.