ÇANKAYA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING
DEPARTMENT

CENG 408
Project Report

Multimodal RAG-Based Product Recommendation System

Hazal KANTAR - 202111036
Ahmet Doğukan GÜNDEMİR - 202111033
Ali Boran BEKTAŞ - 202111001
Hikmet Berkin BULUT - 202111057

Advisor: Assist. Prof. Dr. Serdar ARSLAN

Contents

Introduction

This project focuses on developing a Multimodal Retrieval-Augmented Generation (RAG) system to provide personalized product recommendations in the fashion and cosmetics industries. By using advanced AI technologies, the system combines user input, product images, and textual descriptions to generate accurate and relevant suggestions. Users can interact with the platform through a user-friendly web interface that supports both text and image-based chats.

The goal is to enhance the user experience by delivering recommendations that align with personal preferences, sustainability criteria, and current trends. The system integrates Large Language Models (LLMs) and external APIs, such as OpenAI, Gemini, and Hugging Face, to process multimodal data effectively. This approach allows for a more comprehensive understanding of user needs and market demands.

This report provides a comprehensive overview of the project, including the Test Plan and Results, Software Requirements Specification (SRS), and Software Design Document (SDD), which detail the system's functionality, architecture, and design components. Together, these documents outline the project's objectives, technical specifications, and implementation roadmap.

Under the Test Plan and Results section we present a structured test strategy that lists every high-value test case for the core functionality, identifies areas intentionally left out of scope, and states the pass, fail, and exit criteria that define completion. The accompanying Test Results Summary demonstrates that those exit criteria have been met, closing the feedback loop between design goals and realised performance. Altogether, the report traces the project's purpose, design journey, verification process, and final outcome in one cohesive narrative.

System Requirements Specification

1 Introduction

1.1 Purpose of This Document

The purpose of this document is to outline the software requirements for the development of a Multimodal Retrieval-Augmented Generation (RAG)-Based Product Recommendation System. This document serves as a reference for stakeholders, including project managers, developers, designers, and testers, ensuring a shared understanding of the project's goals, features, constraints, and technical requirements. It provides a comprehensive framework for system implementation, verification, and maintenance.

1.2 Scope of This Project

This project focuses on designing an intelligent recommendation platform for the fashion and cosmetics domains, using multimodal data sources such as textual descriptions, product images, sustainability certifications, and trend analytics. By utilizing advanced AI technologies, including LLMs and multimodal embeddings, the system provides eco-conscious and trend-aligned product recommendations tailored to individual user preferences. The platform consists of a Flask-based backend, a React-powered frontend, and a vector database for efficient data handling. Its ultimate goal is to promote sustainable consumer choices through a user-friendly, adaptive, and visually engaging interface.

2 General Description

2.1 Glossary (Definitions, Acronyms, and Abbreviations)

- **AI:** Artificial Intelligence.
- **API:** Application Programming Interface, a connection between computers or between computer programs.
- **CLIP:** Contrastive Language–Image Pretraining, a model that creates unified embeddings for text and images.[3]
- **Flask:** A lightweight Python web framework for backend API development.[4]
- **Frontend:** The part of the application that interacts directly with the user.
- **Backend:** The part of an application that is not directly accessed by the user, typically responsible for storing and manipulating data.
- **Gemini:** A suite of AI models developed by Google.[5]

- **Hugging Face:** A company providing natural language processing tools and models.[6]
- **HCI:** Human-Computer Interaction; the study of how people interact with computers and to design technologies that let humans interact with computers in novel ways.
- **LLM:** Large Language Model, a type of computational model designed for natural language processing tasks.
- **LLaMA:** Large Language Model Meta AI; a family of large language models developed by Meta AI.[7]
- **OpenAI:** An AI research and deployment company.
- **RAG:** Retrieval-Augmented Generation, A framework combining retrieval-based methods with generative models for improved contextual output.
- **React:** A JavaScript library for building user interfaces.
- **Vector Database:** A database optimized for storing and retrieving vector embeddings.

2.2 User Characteristics

The primary users of this system are environmentally conscious consumers seeking personalized recommendations for fashion and cosmetic products. They are comfortable with digital platforms and expect seamless, intuitive interactions. Secondary users include fashion retailers and brands interested in promoting sustainable products to a targeted audience.

2.3 Overview of Functional Requirements

The system involves a variety of functionalities to provide a seamless user experience and effective personalized recommendations. It includes user login and registration, enabling secure access to the platform while storing individual preferences and interaction history. Personalized and interactive recommendation generation lies at the core of the system, using multimodal data and advanced AI models to create suggestions tailored to the user's preferences. Additionally, users can review ongoing trends in the fashion and cosmetics domains, offering insights into popular and eco-conscious products.

Real-time recommendation updates ensure that users receive the most relevant suggestions based on current trends and new data. Historical recommendations are accessible through the platform, allowing users to revisit previous suggestions for convenience. Profile management features allow users to update their preferences, adjust their interaction history, and manage saved recommendations. Furthermore, a robust search and filtering mechanism is integrated to help users explore the product catalog more efficiently, ensuring a user-centric and dynamic recommendation platform.

2.4 General Constraints and Assumptions

Constraints

- **Data Availability:** The system's performance relies on continuous access to current and accurate product data, including images, descriptions, availability, and sustainability certifications.
- **Performance Limitations:** The efficiency and responsiveness of the recommendation engine are influenced by the capabilities of the employed AI models and the computational resources at hand, which may affect response times and recommendation quality.
- **Regulatory Compliance:** The system must adhere to data privacy regulations and ethical guidelines, ensuring responsible and transparent handling of user data.
- **Scalability:** Accommodating a growing user base and expanding product catalogs requires efficient data processing and storage solutions to maintain system performance.
- **Compatibility:** The system must ensure compatibility with a range of web browsers, including Chrome, Firefox, Safari, and Edge, to support a diverse user base.
- **Accessibility:** As an online platform, the system must provide uninterrupted access and maintain optimal functionality for users across different devices and network conditions.

Assumptions

- **User Connectivity:** It is assumed that users have reliable internet access and utilize modern web browsers to interact with the platform.
- **Accurate User Input:** The effectiveness of personalized recommendations depends on users providing precise and comprehensive preference information.
- **Retailer Collaboration:** Successful operation relies on effective collaboration with retailers to ensure the availability and accuracy of product sustainability information, which is critical for delivering environmentally conscious recommendations.
- **User Proficiency:** Users are presumed to possess a basic level of digital literacy, enabling them to navigate the platform and utilize its features effectively.
- **Stable Operating Environment:** The system is expected to operate within a stable technological environment, with minimal disruptions due to software updates or hardware failures.

3 Specific Requirements

3.1 Interface Requirements

3.1.1 User Interface

The system shall provide an intuitive and responsive user interface accessible via modern web browsers. The frontend will be developed using React to ensure dynamic content rendering and a seamless user experience. Users will interact with the platform through various components, including:

- **Dashboard:** Displays personalized recommendations, recent trends, and user activity summaries.
- **Search and Filter Panel:** Allows users to search for products and apply filters based on categories, sustainability certifications, price ranges, and popularity.
- **Product Detail View:** Provides comprehensive information about selected products, including images, descriptions, sustainability credentials, and user reviews.
- **User Profile Management:** Enables users to view and edit personal information, manage preferences, and review historical recommendations.

The interface will adhere to accessibility standards to accommodate users with varying needs, ensuring a user-friendly experience for all.

3.1.2 Hardware Interface

The system will integrate with various software components and services to deliver its functionalities:

- **User Devices:** Must support modern web browsers compatible with the React-based frontend.
- **Server Infrastructure:** The backend, developed using Flask, will be hosted on cloud servers (e.g., AWS, Google Cloud Platform, Microsoft Azure) to ensure scalability and reliability.

3.1.3 Software Interface

The system will integrate with various software components and services to deliver its functionalities:

- **Backend Services:** The Flask-based backend will handle API requests, manage business logic, and interface with the database.

- **Database:** A vector database (e.g., ChromaDB, Milvus) will store multimodal embeddings for efficient retrieval.
- **External APIs:** Integration with external services such as OpenAI API, Gemini, and Hugging Face for embedding services will enhance recommendation accuracy.
- **Authentication Services:** Implementation of secure authentication protocols to manage user login and registration.

These software interfaces will be designed to ensure seamless communication between components, maintaining data integrity and system performance.

3.1.4 Communication Interfaces

Client-Server Communication The frontend (React-based) will communicate with the backend (Flask-based) via RESTful APIs. These APIs will manage data retrieval, user requests, and recommendations. All communication will be secured using HTTPS to protect data integrity and confidentiality.

Backend-Database Interaction The backend will interact with the vector database (e.g., ChromaDB, Milvus) using database-specific APIs and libraries. These interactions will include storing, updating, and retrieving multimodal embeddings for recommendation generation.

Integration with External APIs The system will communicate with third-party services like OpenAI API, Gemini, and Hugging Face to access advanced AI models and embedding services. Calls to these APIs will be managed via asynchronous methods to optimize performance and ensure responsiveness.

User Notifications The system will include communication interfaces for delivering notifications to users, such as alerts for new trends, recommendations, or updates. Notifications will be sent through web-based mechanisms such as in-app messages or push notifications.

Error Handling and Logging Communication interfaces will include robust error-handling mechanisms to manage issues such as failed API calls, network disruptions, or database inconsistencies. A logging system will capture detailed logs of communication activities to support debugging and system maintenance.

3.2 Detailed Description of Functional Requirements

3.2.1 User Login and Registration

Name	User Login and Registration
Purpose/Description	Enables users to create accounts, securely log in, and access personalized features of the platform.
Inputs	• User-provided email, username, and password during registration. • Login credentials for authentication.
Processing	• Validation of input fields to ensure compliance with security policies (e.g., strong passwords). • Storage of user credentials using encrypted formats. • Authentication via secure protocols.
Outputs	• Confirmation of successful registration or login. • Error messages for failed attempts (e.g., incorrect credentials or duplicate accounts).
Error Handling	• Display specific errors for invalid inputs or registration conflicts. • Account recovery options in case of forgotten passwords.

3.2.2 Personalized and Interactive Recommendation Generation

Name	Personalized and Interactive Recommendation Generation
Purpose/Description	Generates personalized product recommendations based on user preferences, interactions, and historical data.
Inputs	• User preferences, profile data, and interaction history. • Multimodal product data (e.g., text, images, sustainability metrics).
Processing	• Retrieval of relevant data from the vector database using user embeddings. • Integration of results from AI models (e.g., RAG pipeline) to provide personalized recommendations. • Interactive updates based on user feedback (e.g., refining recommendations).
Outputs	• A list of products tailored to user preferences. • Real-time updates to recommendations when user preferences are adjusted.
Error Handling	• Data Gaps: Provide generic recommendations when user data is incomplete. • User Feedback: Allow users to report inaccurate or irrelevant recommendations. • Fallback recommendations in case of model or API failure.

3.2.3 Trends Review

Name	Trends Review
Purpose/Description	Allows users to review current trends in fashion and cosmetics, including eco-friendly options and popular products.
Inputs	• External Data Sources: Trend data from APIs like NewsAPI, social media platforms, and industry reports. • Internal Product Data: Details of products related to trending keywords, categories, or styles. • User Preferences: Personal interests or saved categories influencing trend presentation.
Processing	• Aggregation and filtering of trend data based on user preferences and sustainability criteria. • Visualization of trend summaries (e.g., trending product categories, seasonal highlights).
Outputs	• Trend Highlights: A list of popular trends, including associated products and descriptions. • Visual Summaries: Graphs showing the evolution and impact of trends over time.
Error Handling	• Data Source Failures: Display a fallback message with alternative suggestions. • Analysis Errors: Log issues in trend extraction and ensure retrying data processing.

3.2.4 Real-Time Recommendation Updates

Name	Real-Time Recommendation Updates

Purpose/Description	Provides users with up-to-date product recommendations that reflect the latest trends, availability, and user interactions, ensuring relevance and timeliness in suggestions.
Inputs	• User Interactions: Real-time data on user behaviors, such as clicks, views, purchases, and feedback. • Product Data: Continuous updates on product details, including availability, pricing, and sustainability certifications. • Trend Information: Current data on fashion and cosmetic trends sourced from industry reports, social media, and market analyses.
Processing	• Data Aggregation: Collect and integrate real-time inputs from various sources to form a comprehensive dataset. • Dynamic Analysis: Utilize AI algorithms to analyze aggregated data, identifying patterns and shifts in user preferences and market trends. • Recommendation Adjustment: Modify existing recommendations based on new insights, ensuring alignment with the most recent data.
Outputs	• Updated Recommendations: Present users with a refreshed list of product suggestions that mirror current trends and personal preferences. • User Notifications: Alert users to significant updates or changes in recommendations.

Error Handling	• Data Latency Management: Handle delays in data updates to ensure smooth functioning without noticeable lags. • Fallback Strategies: Revert to the most recent stable data if real-time data is unavailable or delayed. • User Feedback Integration: Monitor and address user-reported issues for continuous refinement.

3.2.5 View Historical Recommendations

Name	View Historical Recommendations
Purpose/Description	Enables users to view previously recommended products and revisit their preferences.
Inputs	• User Profile Data: Historical records of recommendations generated for the user, stored in the system database.
Processing	• Data Retrieval: Query the database to fetch historical recommendation records linked to the user profile. • Organization and Sorting: Arrange the retrieved records based on relevance, time, or user-defined criteria for easy navigation.
Outputs	• Historical Recommendations Display: A user-friendly visualization of past recommendations, categorized and sorted for clarity.

Error Handling	• Data Unavailability: Notify the user if no historical data is available and provide guidance on generating new recommendations. • Database Connectivity Issues: Display an error message and retry option if there is a temporary failure in fetching data. • Search and Filter Errors: Offer default views if user-defined filters return no results.

3.2.6 Profile Management

Name	Profile Management
Purpose/Description	Allows users to manage their personal details, preferences, and account settings.
Inputs	• User-Provided Data: Updates to personal information, preferences, and notification settings.
Processing	• Data Validation: Verify the accuracy and completeness of the new information provided by the user. • Profile Updates: Modify and save changes to user data in the database, ensuring secure storage and real-time synchronization. • Preference Integration: Adjust system settings and recommendation algorithms based on updated user preferences.

Outputs	• Updated Profile Information: Confirmation of changes made to user details and settings. • Enhanced Recommendations: Reflect updated preferences in the personalized recommendations provided by the system. • Notification Settings: Configure and display preferences for receiving updates or alerts.
Error Handling	• Input Errors: Prompt users to correct incomplete or invalid data entries. • Database Issues: Display a message if profile updates cannot be saved temporarily, and retry once the issue is resolved.

3.2.7 Search and Filter Options

Name	Search and Filter Options
Purpose/Description	Ensures users can locate items of interest by applying criteria such as categories, brands, and sustainability certification.
Inputs	• Search Query: Keywords entered by the user to locate specific products. • Filter Criteria: User-defined parameters, such as price range, brand, sustainability certifications, and trending status.
Processing	• Query Matching: Search the database for products matching the user's input keywords and filter criteria. • Filter Application: Refine search results based on selected attributes, such as relevance, price, or category.

Outputs	• Search Results: Display a list of products that match the user's search query and filter preferences. • Filter Summary: Show the applied filters alongside the results, allowing users to adjust them for refined searches.
Error Handling	• No Results Found: Notify the user if no products match the search query or filters. • Invalid Inputs: Prompt users to correct incomplete or inappropriate search queries or filter selections. • System Delays: Display a loading indicator if the search process takes longer than expected.

3.3 Non-Functional Requirements

Backend Server Constraints

The backend server must efficiently handle multiple concurrent requests to ensure smooth functionality during high-traffic periods. The database queries shall be optimized using indexing and caching techniques, minimizing data retrieval time to under 50 milliseconds per query on average. This ensures seamless interaction with real-time recommendation generation, even under significant user loads.

System Responsiveness

The system shall generate product recommendations for user queries within an average response time of 5 seconds. This responsiveness will be maintained by employing asynchronous processing and optimized API calls, ensuring user satisfaction and minimizing wait times.

Availability

The system shall maintain at least 95 percent uptime, ensuring consistent availability for users. This high availability will be achieved through the use of cloud-based infrastructure with failover mechanisms and regular monitoring of system health to preemptively address potential issues.

Ease of Use

The user interface shall be intuitive and user-friendly, requiring no more than 10 minutes for a new user to understand the main features. This will be achieved through clean design principles, tooltips, and a comprehensive help section integrated into the web platform.

Privacy

The system shall comply with GDPR standards for handling user data. It will ensure the confidentiality of personal information and allow users to access, modify, or delete their data upon request. Security measures, such as data encryption and secure login protocols, will be implemented to safeguard user information.

Compatibility

The system shall support the latest versions of major web browsers, including Chrome, Firefox, Safari, and Edge, ensuring broad accessibility. Cross-platform compatibility tests will be conducted to identify and address any discrepancies, guaranteeing a consistent user experience.

Ethical Compliance and Bias Mitigation

The recommendation algorithms shall actively monitor and minimize biases in product suggestions, especially concerning gender, ethnicity, or body type. Regular audits of the recommendation engine will be conducted to detect and rectify any unintentional biases, fostering a fair and inclusive platform.

Transparency

The system shall provide clear explanations for its recommendations, enabling users to understand the rationale behind suggested products. A "Why this recommendation?" feature will be integrated into the user interface, offering insights into how user preferences, trends, and sustainability metrics influence the suggestions.

4 ANALYSIS - UML

4.1 Use Cases

4.1.1 Use Case Diagram

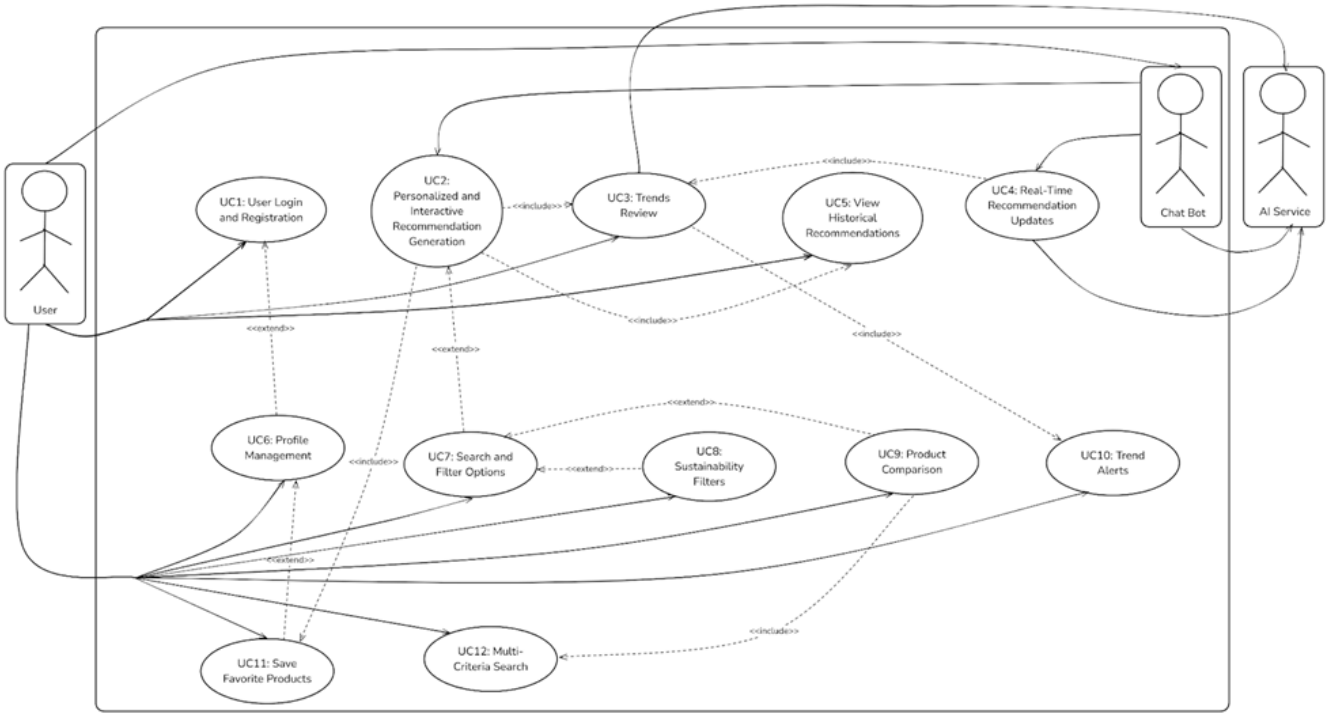


Figure 1: Main Use Case Diagram

4.1.2 Description of Use Cases

Use Case 1: User Login and Registration

Use Case Name	User Login and Registration
Use Case Number	1
Actors	User, System
Description	Users can create an account or log in to access personalized recommendations and manage their profile.

Precondition	The user accesses the system interface.
Scenario	1. The user navigates to the login/registration page. 2. The user enters their credentials or registers by providing required details. 3. The system authenticates the user and grants access.
Postcondition	The user is successfully logged into the system or registered as a new user.
Exceptions	1. Invalid login credentials. 2. User already exists during registration. 3. Network issues preventing communication with the backend.
Related Use Cases	Profile Management (UC6)

Use Case 2: Personalized and Interactive Recommendation Generation

Use Case Name	Personalized and Interactive Recommendation Generation
Use Case Number	2
Actors	User, System
Description	The system generates personalized recommendations based on user preferences, history, and current trends.
Precondition	The user has completed their profile and logged into the system.
Scenario	1. The user logs into their account. 2. The system analyzes user preferences and history. 3. Recommendations are displayed based on real-time trends.
Postcondition	The user receives personalized recommendations tailored to their preferences and trends.

Exceptions	1. Insufficient data in the user profile to generate meaningful recommendations. 2. System errors in retrieving or processing trend data.
Related Use Cases	Search and Filter Options (UC7), Save Favorite Products (UC11)

Use Case 3: Trends Review

Use Case Name	Trends Review
Use Case Number	3
Actors	User, System
Description	Users can view popular trends in fashion and cosmetics based on aggregated data from news sources, social media, and product sales.
Precondition	The system has up-to-date trend data available.
Scenario	1. The user accesses the trends page. 2. The system displays real-time trends. 3. The user interacts with the data for insights.
Postcondition	The user gains insights into current fashion and cosmetics trends.
Exceptions	1. Trend data is outdated or unavailable. 2. The system encounters issues while aggregating data from external APIs.
Related Use Cases	Personalized and Interactive Recommendation Generation (UC2), Real-Time Recommendation Updates (UC4)

Use Case 4: Real-Time Recommendation Updates

Use Case Name	Real-Time Recommendation Updates
Use Case Number	4
Actors	User, System

Description	The system updates recommendations dynamically as user preferences or external trend data changes.
Precondition	User preferences and trend data are updated in the system.
Scenario	1. The system monitors changes in trends or user interactions. 2. Recommendations are updated dynamically. 3. Updated recommendations are displayed to the user.
Postcondition	The user receives real-time recommendations that reflect the latest preferences and trends.
Exceptions	1. Delay in receiving updated trend data. 2. System overload due to high request volume.
Related Use Cases	Trends Review (UC3), Personalized and Interactive Recommendation Generation (UC2)

Use Case 5: View Historical Recommendations

Use Case Name	View Historical Recommendations
Use Case Number	5
Actors	User, System
Description	Users can view their past recommendations to revisit previous suggestions or track changes in trends over time.
Precondition	The system has a log of past recommendations.
Scenario	1. The user navigates to their history page. 2. The system retrieves and displays historical recommendations.
Postcondition	The user views a history of their recommendations.
Exceptions	1. Insufficient data in the history log. 2. System errors in retrieving historical data.

Related Use Cases	Personalized and Interactive Recommendation Generation (UC2)

Use Case 6: Profile Management

Use Case Name	Profile Management
Use Case Number	6
Actors	User, System
Description	Users can manage their personal details, preferences, and settings to customize their experience.
Precondition	The user is logged into their account.
Scenario	1. The user navigates to the profile section. 2. The user modifies preferences or updates personal details. 3. The system saves the updated preferences.
Postcondition	User preferences and settings are successfully updated in the system.
Exceptions	1. System fails to save the updated data due to database errors. 2. User inputs invalid data formats (e.g., non-numeric phone numbers).
Related Use Cases	User Login and Registration (UC1)

Use Case 7: Search and Filter Options

Use Case Name	Search and Filter Options
Use Case Number	7
Actors	User, System
Description	Users can search and filter products using criteria like price, color, brand, and sustainability attributes.
Precondition	The system has products and metadata for filtering.

Scenario	1. The user enters search criteria in the filter options. 2. The system retrieves products matching the criteria. 3. Results are displayed in real-time.
Postcondition	The user views products filtered by their selected criteria.
Exceptions	1. No products match the selected criteria. 2. Errors in retrieving product data due to system or database issues.
Related Use Cases	Personalized and Interactive Recommendation Generation (UC2), Product Comparison (UC9), Sustainability Filters (UC8)

Use Case 8: Sustainability Filters

Use Case Name	Sustainability Filters
Use Case Number	8
Actors	User, System
Description	Users can apply sustainability filters (e.g., eco-friendly materials, cruelty-free certifications) to refine product recommendations.
Precondition	The system contains sustainability data for the available products.
Scenario	1. The user accesses the filter options. 2. The user selects one or more sustainability criteria. 3. The system retrieves and displays filtered results.
Postcondition	Only products meeting the selected sustainability criteria are displayed to the user.
Exceptions	1. No products meet the selected criteria. 2. Incorrect filtering due to incomplete data in the database.

Related Use Cases	Search and Filter Options (UC7), Personalized and Interactive Recommendation Generation (UC2)

Use Case 9: Product Comparison

Use Case Name	Product Comparison
Use Case Number	9
Actors	User, System
Description	Users can compare multiple products based on attributes such as price, popularity, and sustainability.
Precondition	The user has added products to the comparison list.
Scenario	1. The user selects products for comparison. 2. The system retrieves the attributes of the selected products. 3. The system displays a comparison table.
Postcondition	The user views a detailed comparison of the selected products.
Exceptions	1. Insufficient product data for comparison. 2. System errors while retrieving product attributes.
Related Use Cases	Multi-Criteria Search (UC12), Search and Filter Options (UC7)

Use Case 10: Trend Alerts

Use Case Name	Trend Alerts
Use Case Number	10
Actors	User, System
Description	Users can subscribe to alerts for new trends in the fashion and cosmetics industry based on their preferences.
Precondition	The user has an active subscription and trend data is available.

Scenario	1. The user enables trend alerts in their profile settings. 2. The system monitors trend data. 3. Alerts are sent via notifications or emails.
Postcondition	The user is notified about relevant trends.
Exceptions	1. Trend data unavailable due to API errors. 2. Notifications fail to send due to connectivity issues.
Related Use Cases	Trends Review (UC3)

Use Case 11: Save Favorite Products

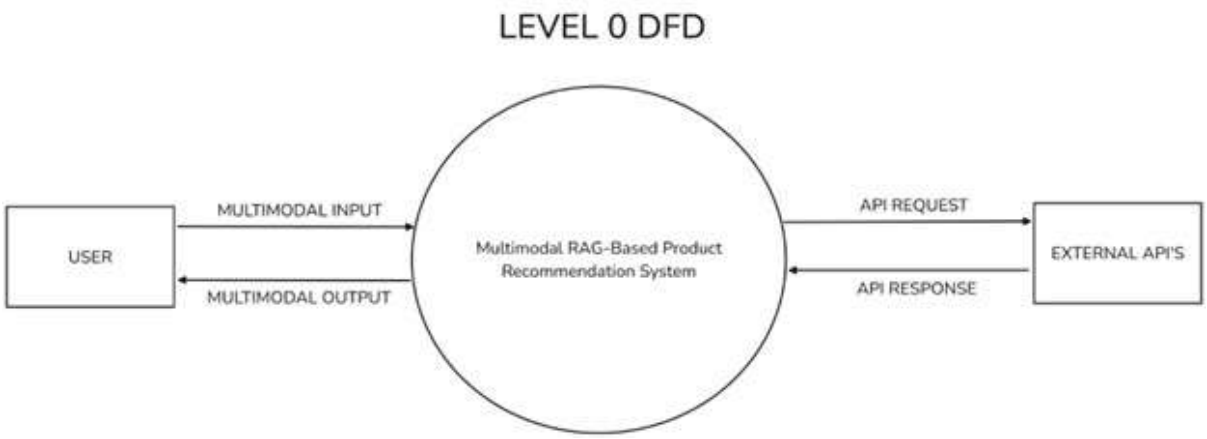
Use Case Name	Save Favorite Products
Use Case Number	11
Actors	User, System
Description	Users can save products to their favorites list for easy access and future reference.
Precondition	The user is logged into their account.
Scenario	1. The user clicks the favorite icon for a product. 2. The system saves the product to the user's favorites list. 3. The user accesses their favorites later.
Postcondition	The selected product is added to the user's favorites list.
Exceptions	1. Errors saving the product to the favorites list due to database issues. 2. The system fails to retrieve the favorites list when requested.
Related Use Cases	Personalized and Interactive Recommendation Generation (UC2), Profile Management (UC6)

Use Case 12: Multi-Criteria Search

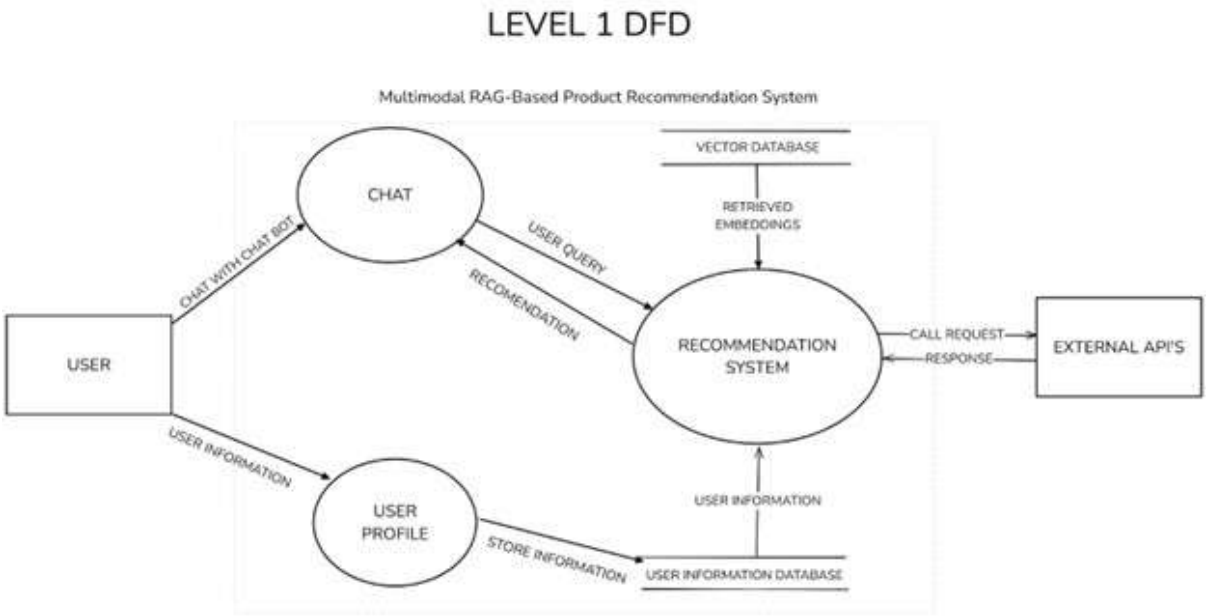
Use Case Name	Multi-Criteria Search
Use Case Number	12
Actors	User, System
Description	Users can perform searches by combining multiple criteria, such as price range, brand, product category, and sustainability certifications.
Precondition	The system has a comprehensive database of product metadata and attributes.
Scenario	1. The user inputs multiple criteria in the search form. 2. The system processes the criteria and retrieves matching products. 3. Results are displayed.
Postcondition	The user views a filtered list of products matching the selected criteria.
Exceptions	1. No products match all the selected criteria. 2. Errors in data retrieval or processing.
Related Use Cases	Search and Filter Options (UC7), Product Comparison (UC9)

4.2 Functional Modeling and Data Flow Diagrams (DFD)

4.2.1 Level - 0 Data Flow Diagram (Context Diagram)



4.2.2 Level - 1 Data Flow Diagram



5 CONCLUSION

The Multimodal Retrieval-Augmented Generation (RAG) system developed in this project represents a significant advancement in personalized recommendations for the fashion and cosmetics industries. By integrating textual descriptions, visual data, sustainability certifications, and real-time trend analysis, the system delivers accurate, user-centered, and environmentally conscious suggestions. Built on a scalable architecture using Flask, React, and vector databases, and powered by advanced AI services like OpenAI and Gemini, the platform ensures responsiveness and precision. This innovative system not only addresses critical gaps in traditional recommendation models but also promotes sustainable consumer behavior, setting a new benchmark for ethical and trend-aware AI applications in the industry.

System Design Document

6 Introduction

6.1 Purpose of this document

This Software Design Document (SDD) describes the architecture and system design of our project, Multimodal RAG-Based Product Recommendation System. The intended audience includes software developers, system architects, project managers, and stakeholders involved in our project. This document aims to provide a detailed and clear understanding of the platform's design.

The SDD covers the architectural design of our project Multimodal RAG-Based Product Recommendation System, providing a detailed look at the architecture. In addition to architectural details, this document includes activity diagrams and sequence diagrams of use cases to illustrate the dynamic aspects of the system. These diagrams help in understanding the flow of activities and interactions within the system, providing a clear visualization of how different components collaborate to achieve specific functionalities.

Furthermore, the SDD presents the UI design of our project, showcasing the visual and interactive elements that users will engage with. This ensures that stakeholders have a comprehensive view of the system's design, from backend architecture to user interface, facilitating better decision making and alignment throughout the development process.

6.2 Definitions, Acronyms, and Abbreviations

- **Multimodal RAG (Retrieval-Augmented Generation):** A system architecture that integrates retrieval mechanisms and generative models to process and combine multiple data modalities, such as text and images, to generate contextual outputs.
- **LLMs (Large Language Models):** Advanced machine learning models that are capable of understanding and generating human-like text based on large-scale datasets.
- **Vector Database:** A database for storing and retrieving high-dimensional data embeddings, often used in recommendation systems to compare and retrieve relevant results efficiently.
- **SDD:** Software Design Document, a document detailing the architectural and system design of the software.
- **UI/UX:** User Interface/User Experience, the overall experience of a user when interacting with a product, including its usability, accessibility, and aesthetics.
- **API (Application Programming Interface):** A set of protocols and tools that allow different software applications to communicate and exchange data.

- **Sustainability Filters:** Criteria applied to product data to identify and recommend eco-friendly items.
- **Embedding:** A mathematical representation of data (e.g., text, images) in a dense vector space, allowing the system to process multimodal data seamlessly.
- **Ethical AI Compliance:** The practice of ensuring the system adheres to guidelines minimizing bias and maintaining fairness, transparency, and user privacy.

7 System Overview

The Multimodal RAG-Based Product Recommendation System is designed to provide personalized recommendations in the fashion and cosmetics domains by leveraging state-of-the-art artificial intelligence technologies. This system integrates retrieval-augmented generation (RAG) architectures with multimodal data processing, combining user preferences, product metadata, and sustainability criteria to deliver tailored suggestions. It also supports dynamic trend analysis and real-time updates, ensuring the recommendations align with current market dynamics and user interests.

The system consists of several core components, including a user-friendly frontend built with React, a Flask-based backend for managing data processing and business logic, and a vector database for efficient embedding retrieval. External APIs, such as those provided by OpenAI, Gemini, and Hugging Face, are used to enrich the system with real-time trend data and advanced recommendation capabilities. The combination of text and image embeddings ensures a holistic understanding of user preferences and product attributes.

The architecture ensures high performance and scalability, allowing it to handle multiple user interactions and complex recommendation queries simultaneously. Users can explore recommendations through features like search and filter options, sustainability insights, and product comparisons, all of which are accessible via a seamless web interface. Furthermore, the system integrates ethical AI principles, such as bias mitigation and transparent decision-making, to enhance user trust and satisfaction.

By combining cutting-edge AI technologies, multimodal data handling, and user-centric design, this system aims to bridge the gap in sustainable and personalized recommendations, setting a benchmark in the fashion and cosmetics industry.

8 System Design

8.1 Architectural Design

The architecture of the Multimodal RAG-Based Product Recommendation System follows a layered three-tier design, ensuring modularity, scalability, and separation of concerns. The system is divided into three primary layers: Presentation Layer, Application

Layer, and Data Layer. Each layer has different roles and interacts with the other layers to provide a seamless user experience.

8.1.1 Layered Architecture

The layered architecture is a design pattern that separates the system into distinct, interconnected layers, where each layer is responsible for a specific aspect of the application. This architectural style promotes modularity, scalability, and maintainability by organizing components based on their functionality, allowing for independent development and testing.

In the context of our Multimodal RAG-Based Product Recommendation System, the layered architecture enables clear separation of user interactions, business logic, and data management. This separation ensures that changes made to one layer, do not disrupt the entire system, thereby reducing the risk of cascading failures.

We chose this architecture because it aligns with industry best practices for AI-driven web applications and effectively supports the processing and storage needs required for multimodal product recommendation systems. The clear division of responsibilities allows our development team to manage different system components simultaneously, increasing productivity and reducing development time.

8.1.2 Key Components

Our project's layered architecture consists of three primary layers, each fulfilling a distinct role within the system. These layers interact to deliver personalized recommendations, manage user data, and ensure the system operates efficiently. Components include:

- **Presentation Layer:** This layer manages user interaction and visualization. Built with React, it captures user inputs and displays real-time recommendations, product comparisons, and sustainability data.
- **Application Layer:** The core processing unit, implemented in Flask, handles business logic, API requests, and AI model integration. This layer processes user inputs, manages sessions, and communicates with external AI services to generate recommendations.
- **Data Layer:** Responsible for data storage and retrieval, this layer uses a database for structured data (user profiles, logs) and a vector database for vector embeddings. It ensures fast and scalable storage of product embeddings, allowing efficient search and recommendation generation.

8.1.3 Benefits of Layered Architecture

- **Modularity:** Each layer can be updated or replaced without affecting the entire system.

- **Scalability:** Layers can scale independently based on demand, ensuring optimal performance even during peak usage.
- **Security:** Sensitive user data is isolated in the Data Layer, enhancing security and preventing unauthorized access.
- **Maintainability:** Clear separation of concerns simplifies debugging and system enhancements.
- **Flexibility:** New features or external services can be integrated into specific layers, ensuring adaptability.

8.2 Decomposition Description

The Multimodal RAG-Based Product Recommendation System is designed using a modular three-tier architecture, with each layer handling distinct responsibilities. This decomposition ensures the system's scalability, maintainability, and flexibility. By dividing the system into functional layers Presentation Layer, Application Layer, and Data Layer the design allows for independent development and enhancement of each component, promoting efficient workflows and easier debugging.

8.2.1 Presentation Layer

The Presentation Layer acts as the user interface for the system, handling all interactions between the user and the application. It is responsible for capturing user inputs, displaying product recommendations, and visualizing real-time data, making it the most visible part of the architecture.

Responsibilities:

- Collects and processes user inputs such as login credentials, search queries, filter options, and profile settings.
- Displays interactive data, including product recommendations, trend insights, and sustainability details.
- Provides responsive and intuitive user interfaces that can adapt to different web browsers.
- Implements dynamic components using React, allowing for real-time updates without reloading the page.
- Facilitates secure communication with the backend via REST APIs for retrieving or submitting data.
- Ensures accessibility across browsers (Chrome, Safari, Firefox) for wider usability.

Key Features:

- **Authentication and Authorization:** Login and registration interfaces ensure secure access to personalized recommendations.
- **Recommendation Visualization:** Dynamic product recommendations are displayed with detailed trend analysis.
- **Search and Filter:** Users can search products using multi-criteria filters (e.g., price, sustainability).
- **Responsive Design:** Ensures seamless experiences across browsers and screen sizes.

8.2.2 Application Layer

The Application Layer is the core processing engine, managing the system's business logic and orchestrating data flow between the Presentation Layer and Data Layer. It handles AI model integration, user requests, and recommendation generation through advanced retrieval-augmented generation (RAG) techniques.

Responsibilities:

- Processes user requests, manages product searches and generates personalized recommendations.
- Hosts the Multimodal RAG framework, which combines text and image embeddings to enhance recommendation accuracy.
- Communicates with external AI services (e.g., OpenAI, Gemini, Hugging Face) to retrieve embeddings and analyze product trends.
- Manages user sessions, handles profile data, and updates product preferences based on real-time interactions.
- Implements Flask as the primary backend framework for creating and managing API endpoints.
- Facilitates interaction with external APIs to fetch data, analyze trends, and generate personalized recommendations.
- Incorporates error handling mechanisms to ensure system reliability and fallback solutions for API failures.

Key Features:

- **AI Model Integration:** Seamless integration with LLaMA, OpenAI CLIP, and Gemini for advanced product recommendations.

- **Real-Time Recommendations:** Continuously updates recommendations based on user interactions and external trend data.
- **Trend Monitoring:** Aggregates product trend data from APIs and processes it through multimodal embeddings.
- **Security Measures:**Implements user authentication and session management.

8.2.3 Data Layer

The Data Layer manages the storage, retrieval, and processing of all system data, including user profiles, embeddings, and product metadata. This layer plays a critical role in ensuring the system's responsiveness and scalability.

Responsibilities:

- Stores structured data such as user preferences, product information, and recommendation history.
- Manages high-dimensional embedding data for product recommendations using vector databases.
- Aggregates and stores trend data, sustainability certifications, and product details from external sources.
- Provides efficient data indexing and retrieval for fast recommendation generation.
- Maintains logs of user interactions to improve the accuracy of future recommendations.

Key Features:

- **Relational Database:** Stores essential user and product data, providing structured, fast retrieval.
- **Vector Database:** Manages embeddings for similarity searches and efficient product matching.
- **Metadata Storage:** Aggregates and stores external data, such as sustainability ratings, product reviews, and sales trends.
- **Data Caching:** Frequently accessed data is cached to reduce latency and improve response times.

8.2.4 External AI Services

To enhance the system's recommendation capabilities, the Application Layer integrates with external AI services that provide embedding generation, large language model (LLM) processing, and multimodal data analysis.

Services Possible to Use:

- **OpenAI (CLIP, GPT):** Generates embeddings for text and image data.
- **Gemini API (Google):** Provides real-time trend and sustainability data.
- **Hugging Face API:** Pre-trained models for NLP and sentiment analysis.
- **LLaMA (Meta AI):** Enhances recommendations through advanced LLM capabilities.

8.2.5 Communication Between Layers

The three layers communicate through REST APIs and database queries to facilitate data exchange. This modular design ensures that updates to one layer do not disrupt the others.

- **Presentation Layer to Application Layer:** RESTful API endpoints handle data requests and responses.
- **Application Layer to Data Layer:** SQL queries and embedding lookups manage interactions with databases.
- **Application Layer to External Services:** API calls to fetch embeddings and process product data.

8.3 System Modeling

8.3.1 Activity Diagrams

Use Case 2: Personalized and Interactive Recommendation Generation (Activity Diagram)

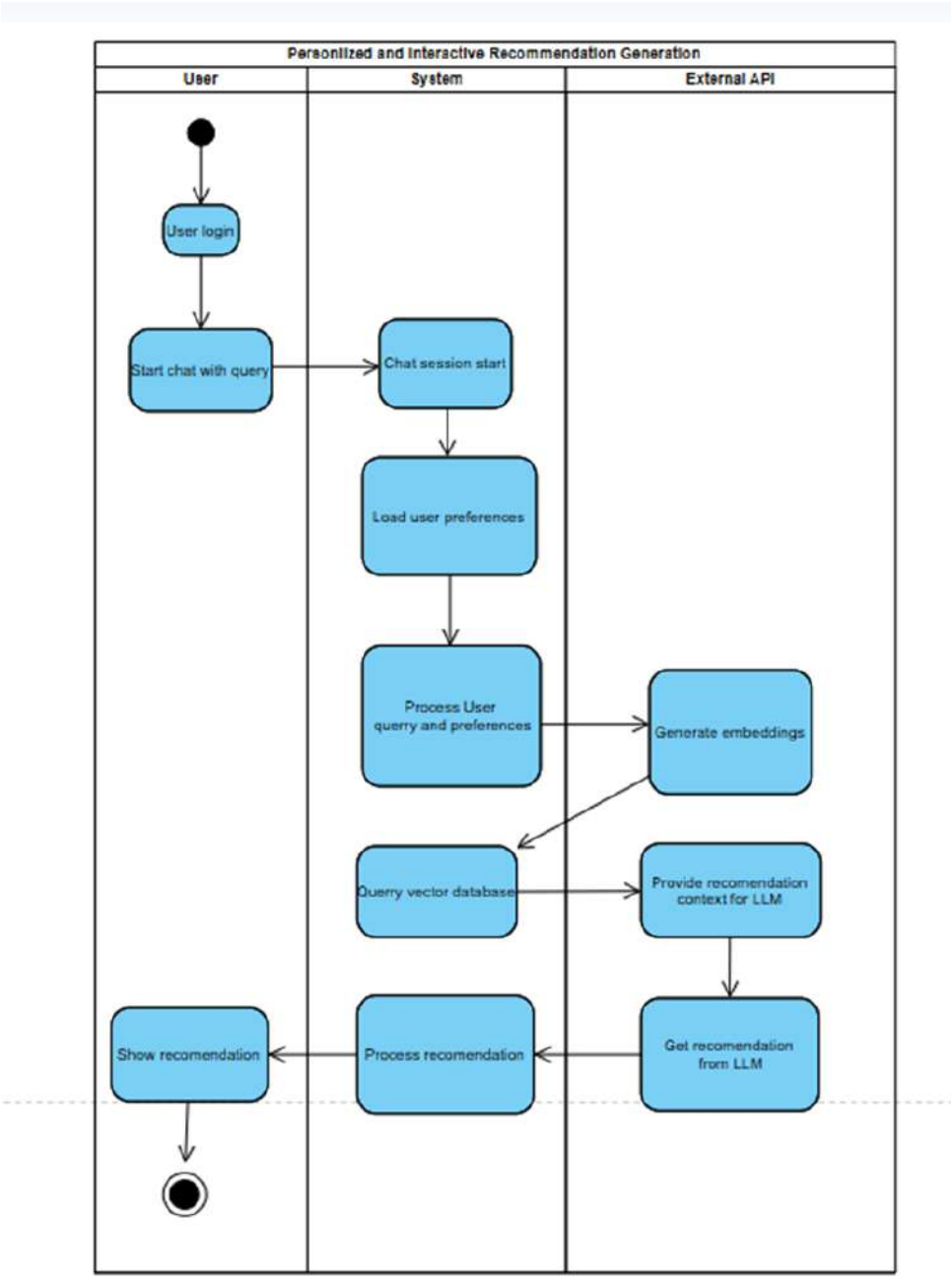


Figure 1: Use Case 2 Activity Diagram

Use Case 3: Trends Review (Activity Diagram)

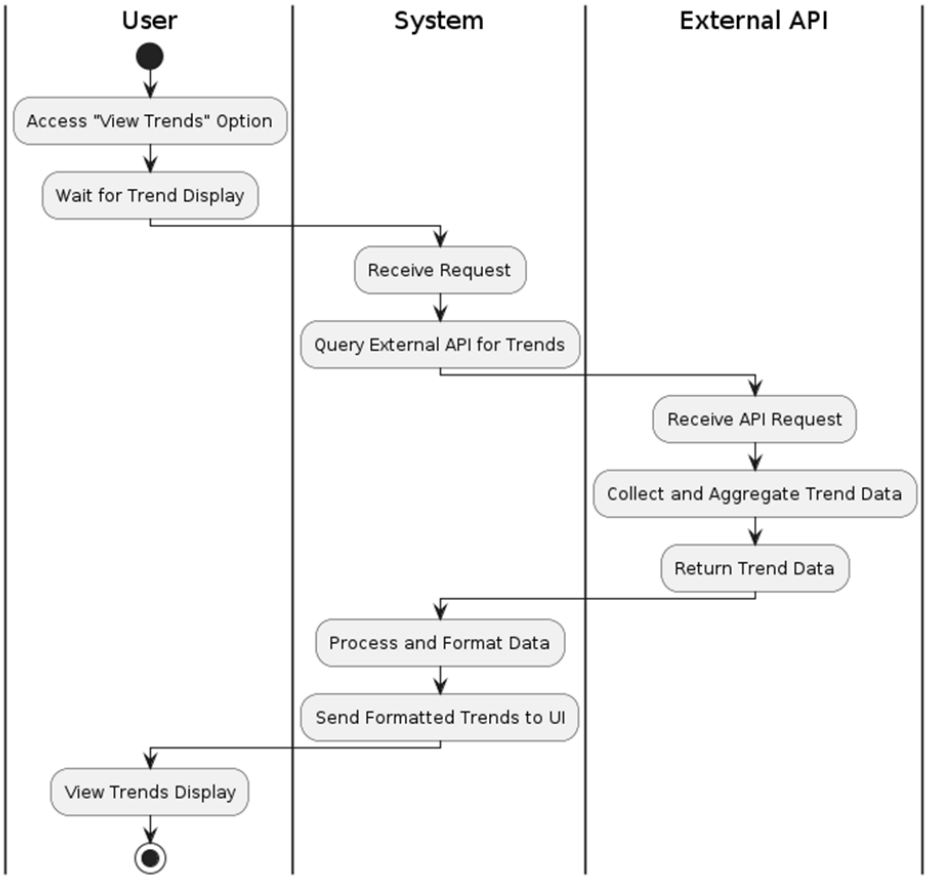


Figure 2: Use Case 3 Activity Diagram

Use Case 7: Search and Filter Options (Activity Diagram)

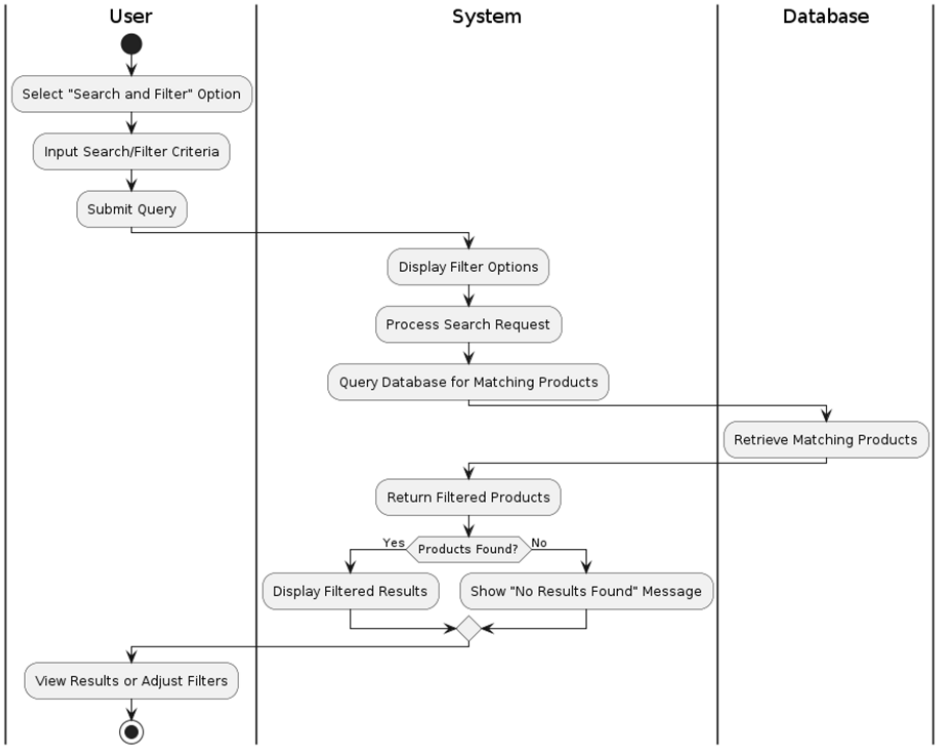


Figure 3 : Use Case 7 Activity Diagram

Use Case 9: Product Comparison (Activity Diagram)

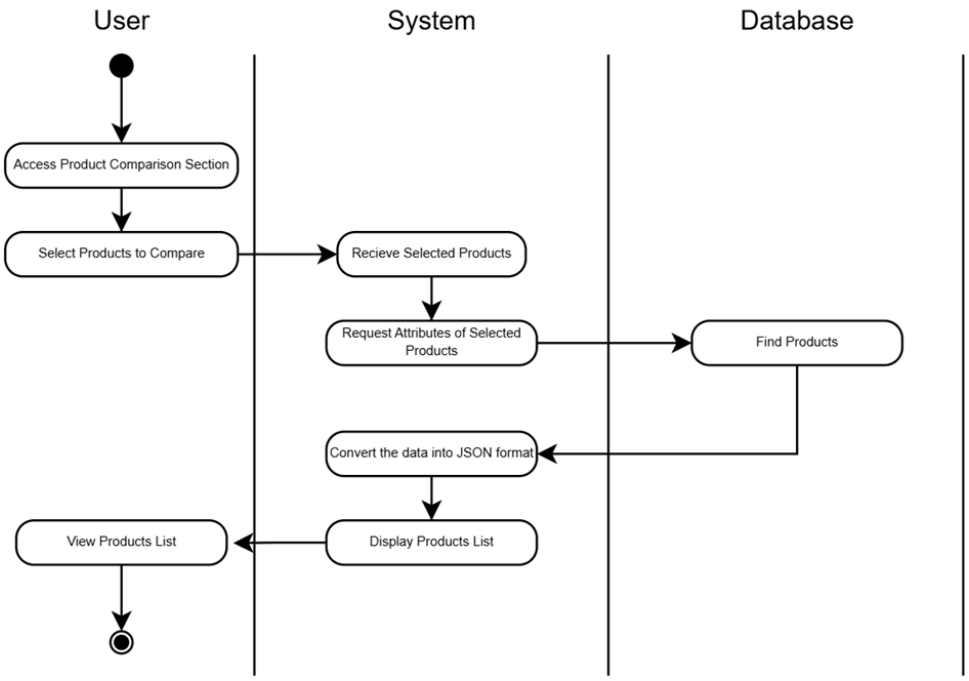


Figure 4: Use Case 9 Activity Diagram

8.3.2 Sequence Diagrams

Use Case 2: Personalized and Interactive Recommendation Generation (Sequence Diagram)

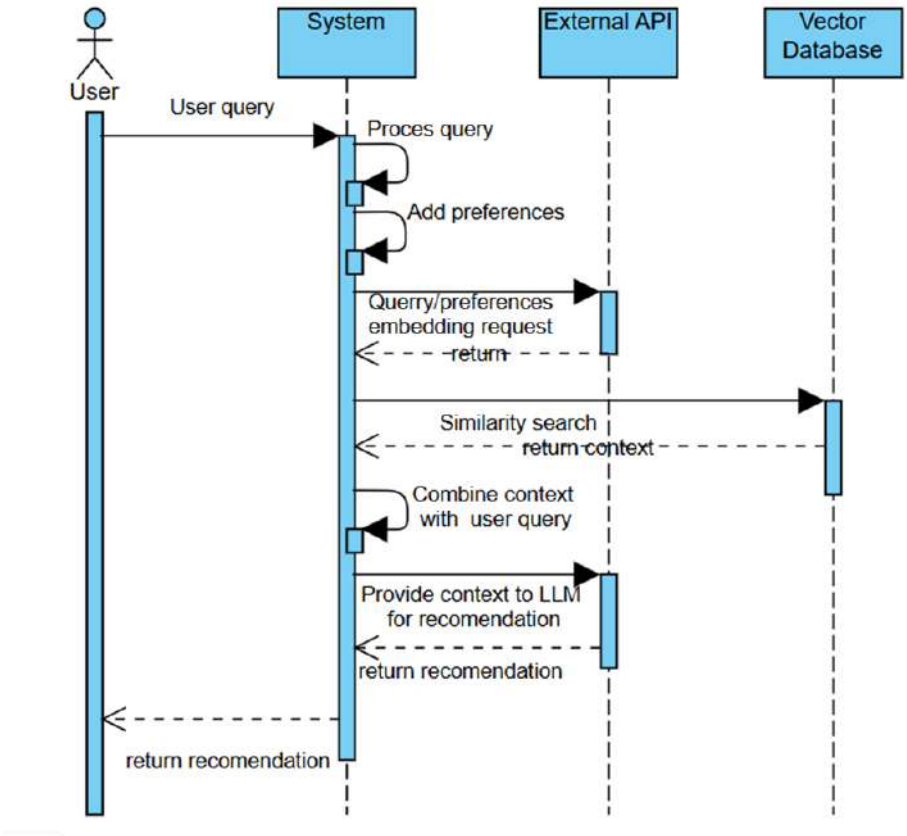


Figure 5: Use Case 2 Sequence Diagram

Use Case 3: Trends Review (Sequence Diagram)

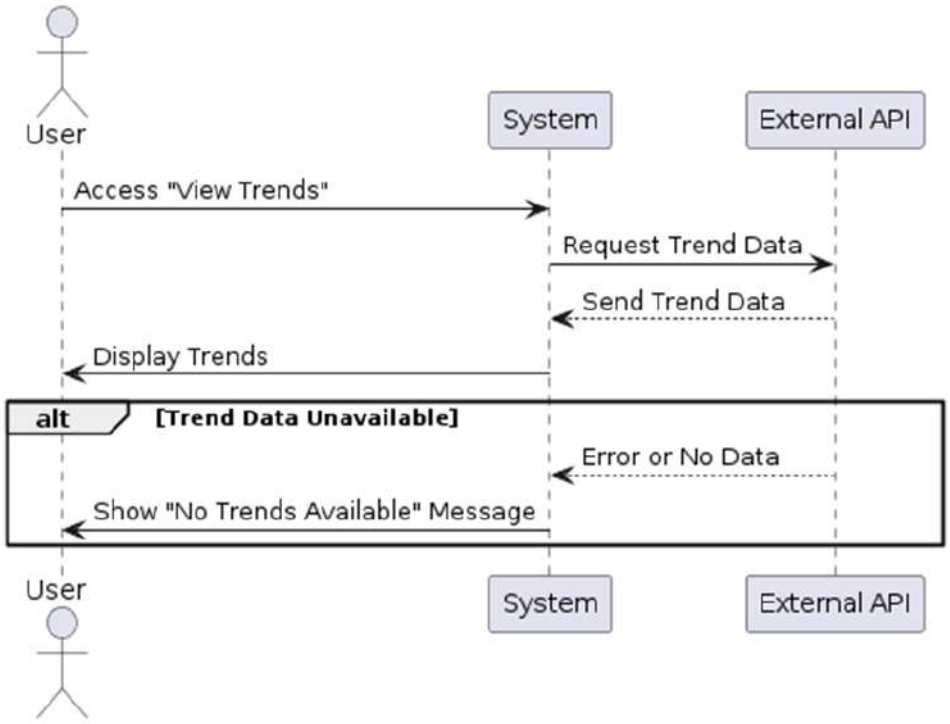


Figure 6: Use Case 3 Sequence Diagram

Use Case 7: Search and Filter Options (Sequence Diagram)

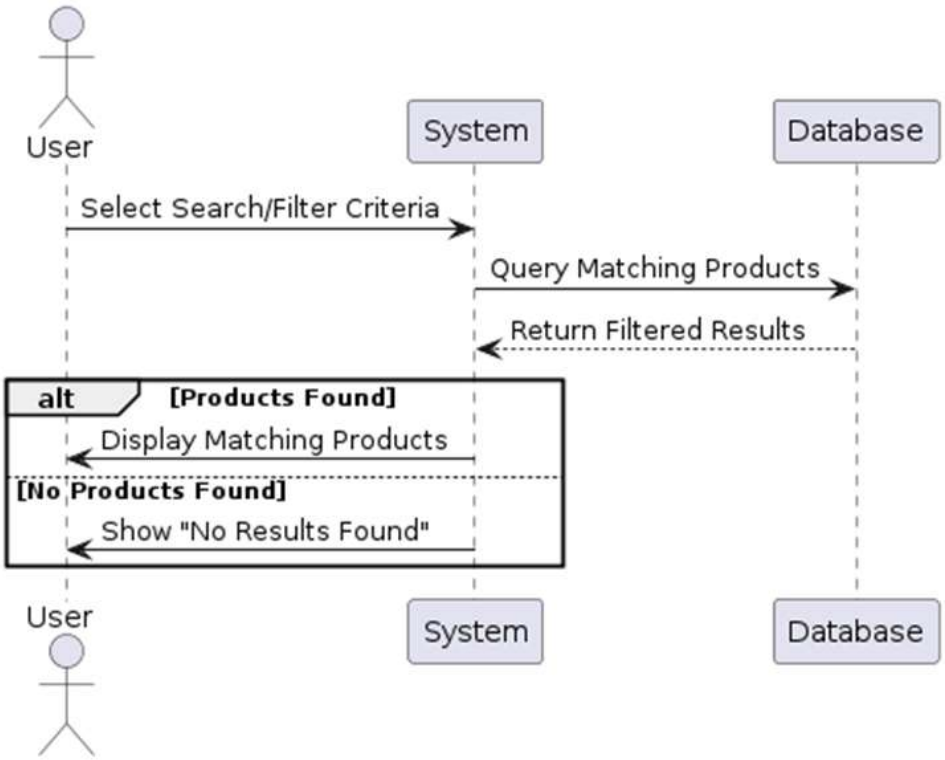


Figure 7: Use Case 7 Sequence Diagram

Use Case 9: Product Comparison (Sequence Diagram)

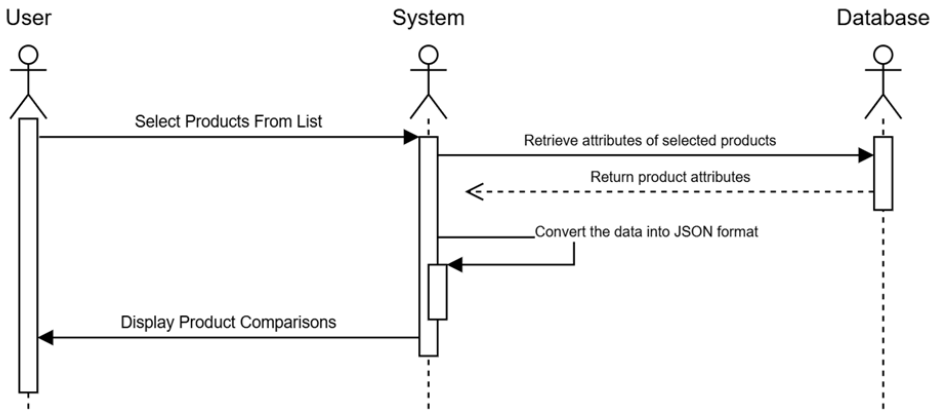


Figure 8: Use Case 9 Sequence Diagram

8.3.3 Class Diagram

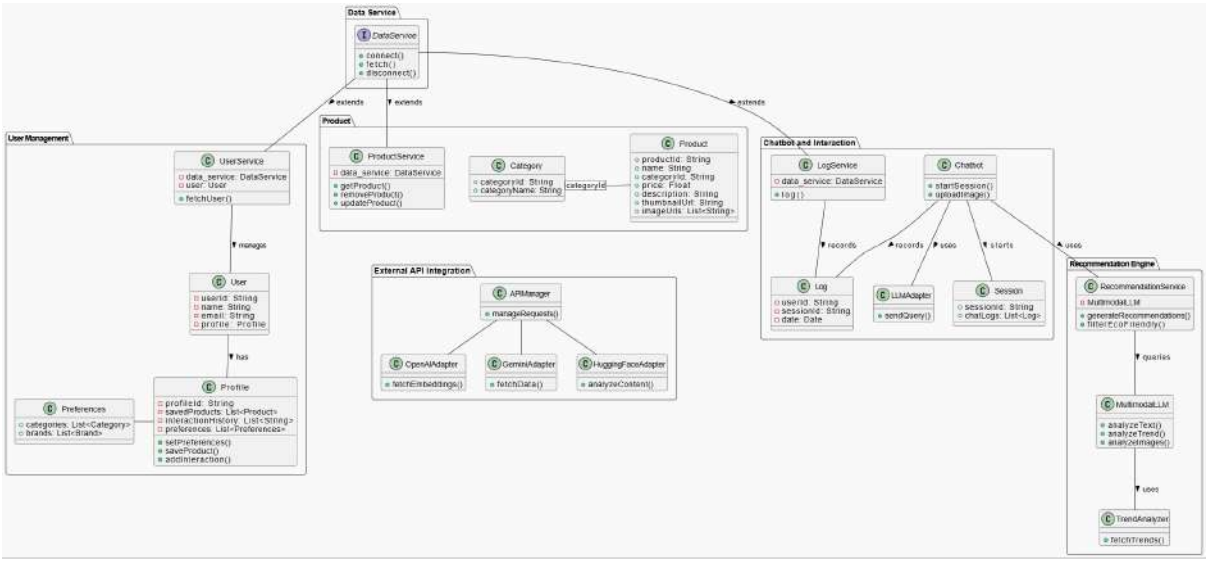


Figure 9: Class Diagram

9 User Interface Design

9.1 User Interface of Profile Page

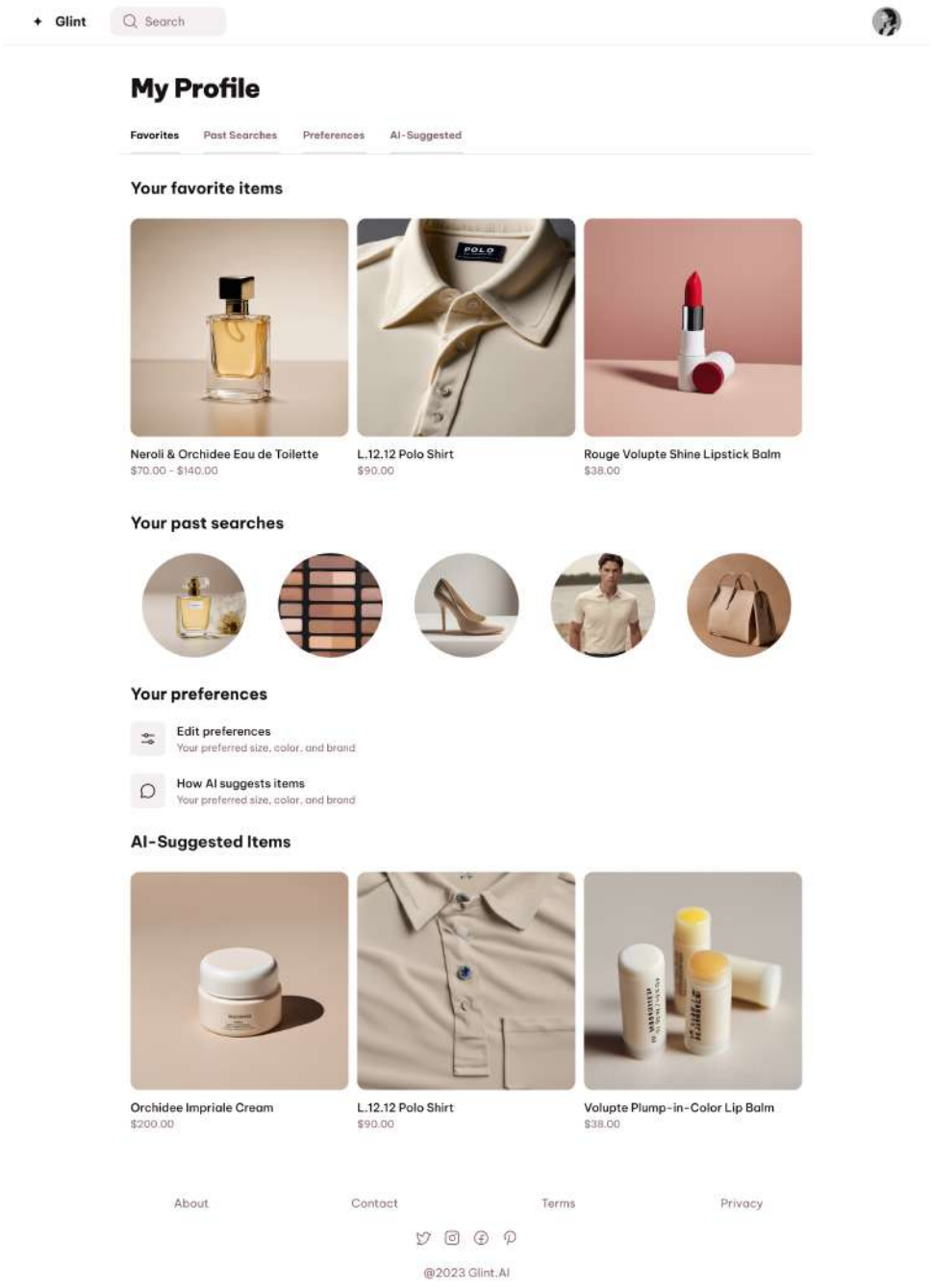


Figure 10: Profile Page UI

9.2 User Interface of Chatbot Screen

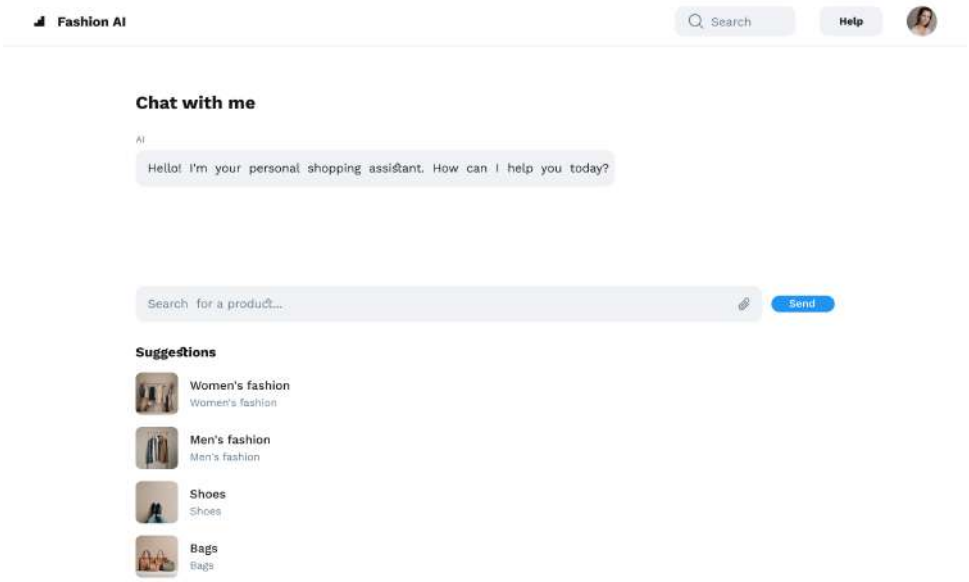


Figure 11: Chatbot UI

9.3 User Interface of Personal Recommendations Page

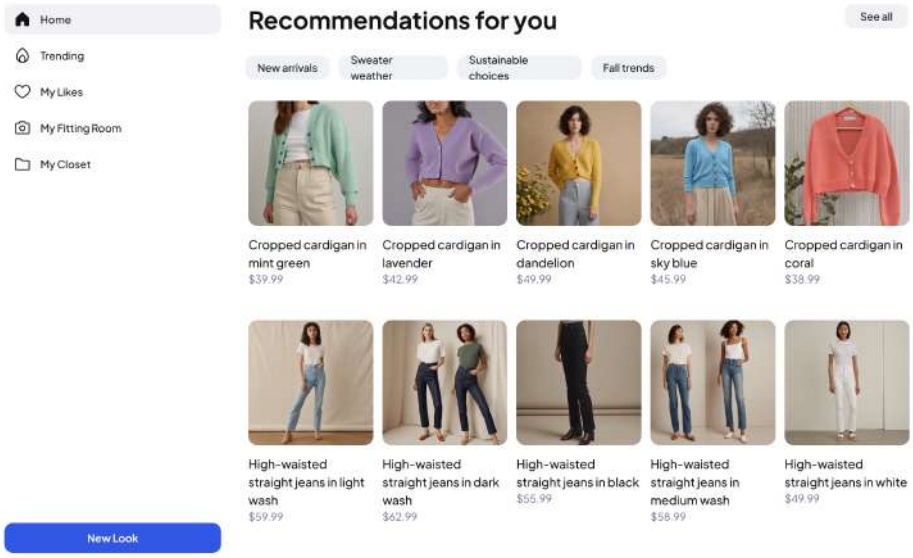


Figure 12: Personal Recommendations UI

9.4 User Interface of Home Page

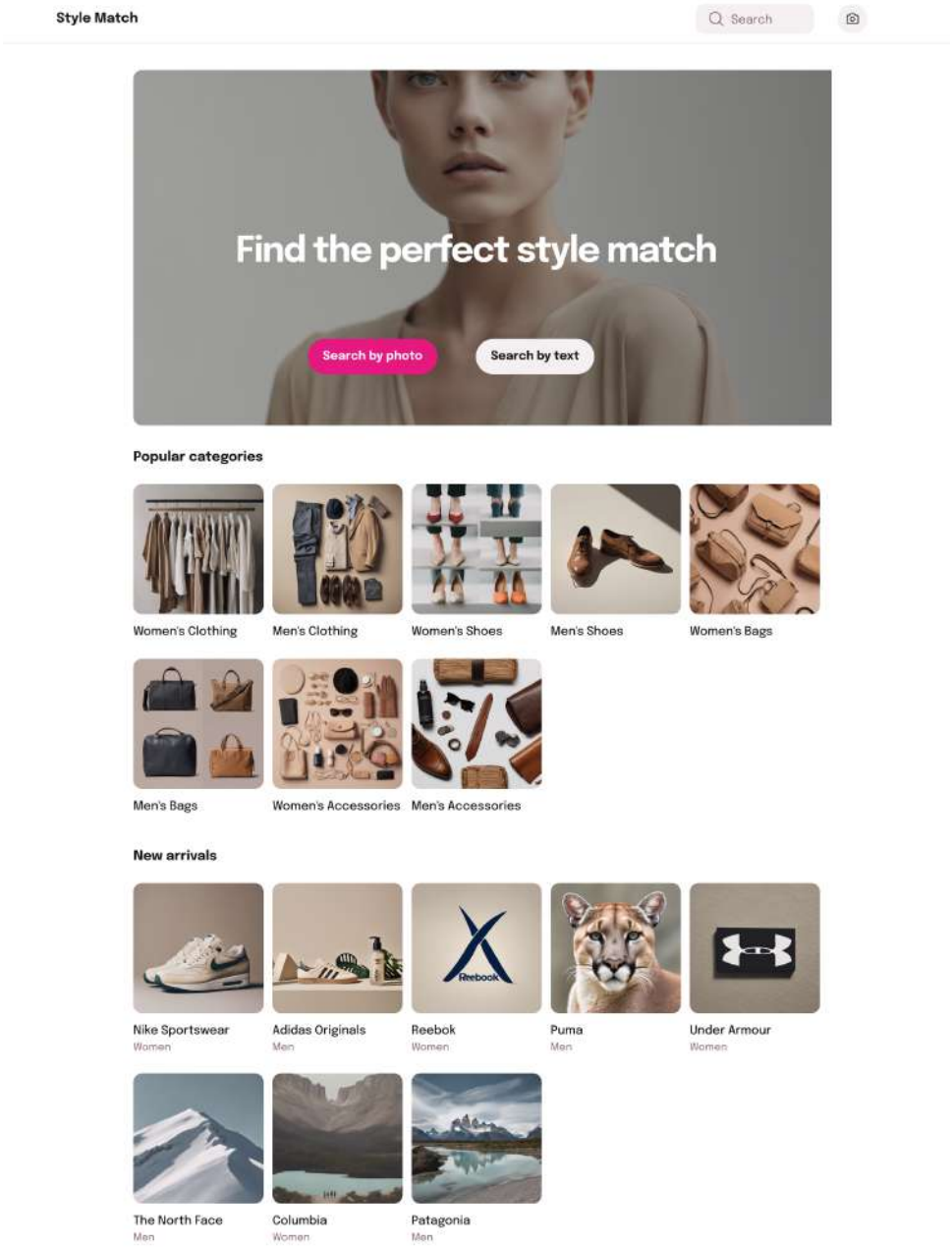


Figure 13: Home Page UI

9.5 User Interface of Login Page

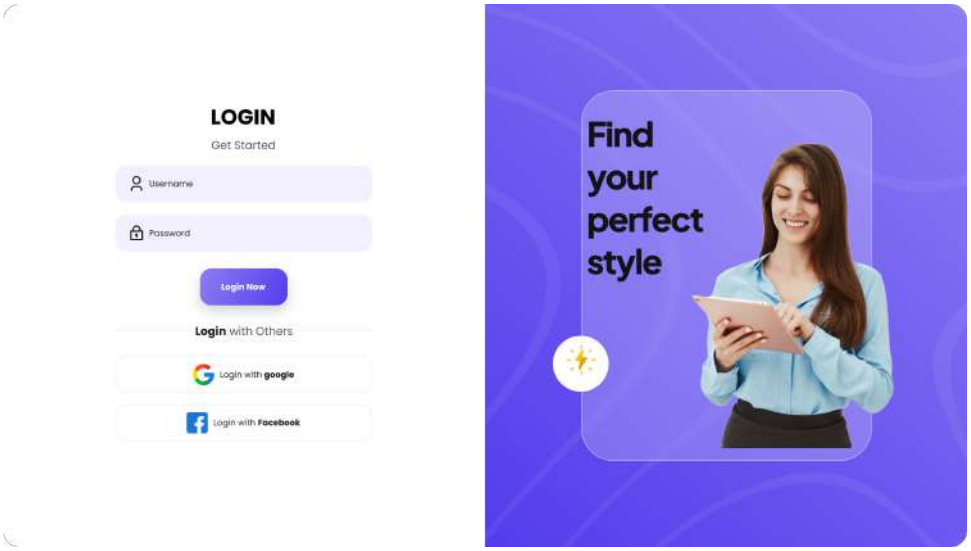


Figure 14: Login Page UI

9.6 User Interface of Trending Page

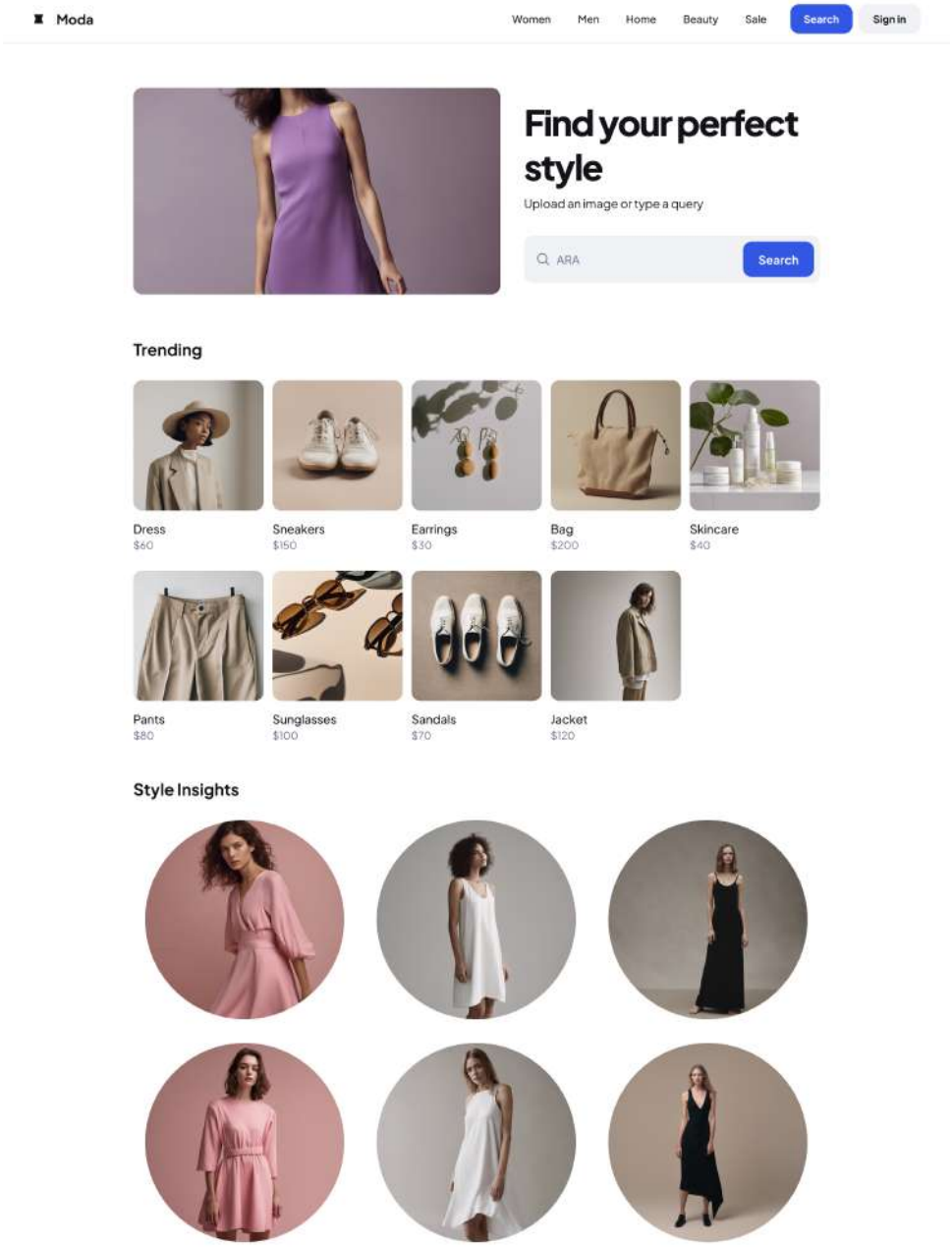


Figure 15: Trending Page UI

10 Test Plan & Result Documents

10.1 Introduction

10.1.1 Version Control

Version No.	Description of Changes	Date
1.0	First version	05 May 2025

10.1.2 Overview

This project tests the effectiveness and functionality of a multimodal AI-driven product-recommendation system that uses Retrieval-Augmented Generation (RAG). The platform accepts both text and image inputs to provide personalised fashion and cosmetic product suggestions.

10.1.3 Scope

The document covers implementation and validation of the RAG engine, interactive chatbot, multimodal input processing, and dynamic filtering. It includes test design, test-case development, feature validation, and pass/fail criteria.

10.1.4 Terminology

Acronym	Definition
SRS	Software Requirements Specification
SDD	Software Design Document
RAG	Retrieval-Augmented Generation
LLM	Large Language Model
HCI	Human–Computer Interaction
API	Application Programming Interface
DB	Database
UI	User Interface

10.2 Features to Be Tested

Chatbot Conversation (CC) This feature enables users to interact with the system using natural language through a chatbot interface. It supports both text and image inputs, providing personalized recommendations and contextual insights. Testing will focus on response accuracy, speed, and user satisfaction.

User Management (UM) This functionality covers secure user registration and account creation. It ensures that new users can successfully sign up, and log out with proper validation and error handling for duplicate entries or invalid data. The tests will verify data integrity and the overall registration process.

Personalised Recommendation Generation (PRG) The system generates tailored product suggestions by processing multimodal inputs (text and images) using an RAG framework. This feature integrates user preferences and real-time trend data to deliver relevant recommendations. Testing will assess the relevance, timeliness, and personalization of the generated recommendations.

Trends Review (TR) This feature aggregates real-time fashion and cosmetic trends from external data sources and displays them to users. It is crucial to provide updated market insights that influence product recommendations. Tests will validate the accuracy and freshness of the trend data displayed.

Search & Filter Options (SFO) Users can search for and filter products based on various criteria such as price, color, styles, categories, sustainability metrics, and different fabrics. This functionality ensures that users can efficiently narrow down product listings to meet their needs. Testing will focus on the accuracy of search results and the functionality of the filtering mechanism.

Product Comparison (PC) Users can compare multiple products using different attributes. This feature supports informed decision-making by presenting a clear comparison. Testing will check the correctness and clarity of the comparison output.

UI Interactions (UI) Users are in interaction with the UI on different pages. This functionality is necessary to ease the navigation and enhance user experience. Testing will focus on the general look of the interactions user has with the UI.

10.3 Features Not to Be Tested

Features not included in this testing phase are the non-critical aesthetic aspects of the user interface, and integration with external third-party services that do not affect the core recommendation functionalities. These components are considered stable and outside the scope of the current testing plan, which focuses on the system's core functionalities and performance.

10.4 Item Pass/Fail Criteria

A test case is considered passed if the actual output of the system aligns with the expected outcome. A test case is considered failed if there is any deviation between the actual and expected results.

10.4.1 Exit Criteria

The testing of the Multimodal RAG-Based Product Recommendation System will be considered successful and complete when the following criteria are met:

- 100% of planned test cases executed.
- $\geq$ 80% of test cases pass.
- All **High**- and **Medium**-priority cases pass.

10.5 Test Design Specifications

10.5.1 User Management (UM)

Subfeatures to be tested **Profile Details (UM.PD)** – This subfeature ensures that users can register, log in, and manage their profiles securely, with proper validation and error handling.

TC ID	Priority	Scenario Description
UM.PD.01	High	User registers with valid details and successfully logs in.
UM.PD.02	High	User attempts to log in with an incorrect or blank password, and the system returns an error message.
UM.PD.03	High	User attempts to log in with an incorrect or blank email, and the system returns an error message.

10.5.2 Personalized Recommendation Generation (PRG)

Subfeatures to be tested **Text-Based Recommendations (PRG.TX)** – processes text queries to generate tailored product recommendations using the RAG framework.
Image-Based Recommendations (PRG.IMG) – processes uploaded images to generate visual recommendations based on product attributes and trends.

PRG.TX.01	High	User submits a valid text query and receives relevant product recommendations.
PRG.IMG.01	High	User uploads an image and the system returns matching product suggestions based on visual features.
PRG.TX.02	Medium	User submits an irrelevant text query and receives according answer suggesting user to enter relevant query.
PRG.IMG.02	Medium	User uploads an irrelevant image and the system returns matching product suggestions based on visual features.

10.5.3 Trends Review (TR)

Subfeatures to be tested **Real-Time Trend Analysis (TR.RT)** – This subfeature aggregates real-time fashion and cosmetic trends from external sources and displays them to users.

TR.RT.01	High	User accesses the trends page and sees up-to-date trend data.
TR.RT.02	Medium	System periodically refreshes trend data and displays updated insights.

10.5.4 Search and Filter Options (SFO)

Subfeatures to be tested **Multi-Criteria Search (SFO.MC)** – This subfeature allows users to search for products using multiple criteria such as price, colour, style, category, sustainability metrics, and fabric types.

SFO.MC.01	High	User inputs multiple criteria and the system displays filtered product listings accurately.
SFO.MC.02	Medium	The system displays a 'No Results Found' message when no products match the selected criteria.

10.5.5 Product Comparison (PC)

Subfeatures to be tested **Comparison View (PC.CV)** – This subfeature enables users to compare multiple products side-by-side based on attributes such as price, popularity, and sustainability.

PC.CV.01	High	User selects products for comparison, and the system displays a detailed comparison.
PC.CV.02	Medium	The system shows a warning message when product data for comparison is incomplete or missing.

10.5.6 UI Interactions (UI)

Subfeatures to be tested **Chatbot Conversation (UI.CC)** – This feature enables users to interact with the system using a conversational interface that supports both text and image inputs. It provides personalized recommendations and contextual insights in real time.
Interactive Chat (UI.CC.IC) – Processes text queries, providing clear and relevant responses.
Multimodal Interaction (UI.CC.MI) – Integrates image uploads with text inputs to refine recommendations.

UI.CC.IC.01	High	User sends a valid text query and receives personalized recommendations.
UI.CC.IC.02	Medium	User submits an ambiguous and/or complex query and the system prompts for clarification and/or simplification.
UI.CC.MI.01	High	User uploads an image, and the system generates recommendations based on the image context.
UI.CC.MI.02	High	User sends a combined text and image query, and the system processes them to provide integrated recommendations.

10.5.7 Client-Server Communication (CSC)

Subfeatures to be tested **API Integration (CS.API)** – This feature evaluates the reliability and efficiency of communication between the frontend and backend, including external API calls to AI services for embedding generation and trend analysis.

CS.API.01	High	System successfully retrieves trend data from external APIs and integrates it into recommendations.
CS.API.02	Medium	System handles slow API responses, ensuring no negative impact on user experience.
CS.API.03	High	System creates and retrieves embeddings using AI services from external API calls.

10.6 Detailed Test Cases

10.6.1 PRG.IMG.01

TC_ID	PRG.IMG.01
Purpose	Verify that a registered user can upload an image via the chatbot and receive relevant product recommendations based on the image.
Priority	High
Estimated Time Needed	10 Seconds
Dependency	User must be registered and logged in; a valid test image must be accessible.
Setup	Ensure the chatbot interface is active, and the test image file is accessible.
Procedure	[A01] Navigate to the chatbot interface. [A02] Click the "Upload Image" button. [A03] Select a valid image file. [A04] Optionally add a text prompt for clarification. [A05] Submit the input and verify that the system processes the image to generate relevant recommendations.
Validation	[V01] Confirm that the recommendations align with the context of the uploaded image.
Cleanup	Remove the uploaded image from the session and log out.

10.6.2 UI.CC.MI.02

TC_ID	UI.CC.MI.02
Purpose	Verify that the recommendation system processes both text and image inputs together.
Priority	High
Estimated Time Needed	10 Seconds
Dependency	Text and image integration must be functional.
Setup	Ensure the chatbot interface supports multimodal inputs.
Procedure	[A01] Enter a text query and upload an image of a jacket. [A02] Submit the inputs.
Validation	[V01] Confirm that the recommendations consider both the text and image context.
Cleanup	Remove the conversation.

10.6.3 UI.CC.IC.01

TC_ID	UI.CC.IC.01
Purpose	Verify that the recommendations provided by the chatbot are relevant and tailored to the input text query.
Priority	High
Estimated Time Needed	6 Seconds
Dependency	Recommendation algorithms must be operational.
Setup	Ensure the chatbot interface is active and user is logged in.
Procedure	[A01] Navigate to the chatbot interface. [A02] Enter specific text queries. [A03] Submit the query and observe the recommendations provided.
Validation	[V01] Verify that the recommendations closely match the input query.
Cleanup	Log out of the chatbot session.

10.6.4 UI.CC.IC.02

TC_ID	UI.CC.IC.02
Purpose	Verify that the chatbot gracefully handles invalid or unsupported queries, or when the input query is ambiguous.
Priority	Medium
Estimated Time Needed	7 Seconds
Dependency	Error-handling mechanisms must be active.
Setup	Ensure the chatbot interface is accessible.
Procedure	[A01] Enter an invalid or ambiguous query. [A02] Observe the system response.
Validation	[V01] Verify that the system does not crash, and tries to make sense of the query.
Cleanup	Clear session data.

10.6.5 PRG.TX.01

TC_ID	PRG.TX.01
Purpose	Verify that the recommendation engine adapts to user preferences stored in the system.
Priority	High
Estimated Time Needed	15 Seconds
Dependency	User preference data must exist in the database.
Setup	Ensure the system has stored user preferences.
Procedure	[A01] Log in to the chatbot and submit a query. [A02] Observe whether the recommendations align with the user's preferences.
Validation	[V01] Verify that the recommendations reflect the preferences.
Cleanup	None.

10.6.6 SFO.MC.01

TC_ID	SFO.MC.01
Purpose	Verify that multiple filter criteria can be applied simultaneously to refine search results.
Priority	High
Estimated Time Needed	8 Seconds
Dependency	Filter logic must be implemented.
Setup	Populate the search database with products with diverse attributes.
Procedure	[A01] Perform a search query and apply multiple filters. [A02] Verify that the results match all applied filters.
Validation	[V01] Confirm accurate filtering of results.
Cleanup	Reset the filters to default.

10.6.7 TR.RT.01

TC_ID	TR.RT.01
Purpose	Verify that the system aggregates accurate trend data from external APIs and displays it to the user.
Priority	High
Estimated Time Needed	12 Seconds
Dependency	External API integration must be functional.
Setup	Ensure trend data aggregation is enabled and API connections are active.
Procedure	[A01] Access the trends page. [A02] Verify that data is fetched from external sources.
Validation	[V01] Confirm the accuracy and freshness of the displayed trend data.
Cleanup	None.

10.6.8 UI.CC.MI.01

TC_ID	UI.CC.MI.01
Purpose	Verify that the chatbot API processes image inputs and returns recommendations based on image attributes.
Priority	High
Estimated Time Needed	12 Seconds
Dependency	Image processing service and chatbot API must be operational.
Setup	Prepare valid image input and ensure the API endpoint is live.
Procedure	[A01] Send a POST request with image data to the chatbot API. [A02] Observe the API response and verify generated recommendations.
Validation	[V01] Ensure that the response is accurate, complete, and returns a 200 HTTP status code.
Cleanup	Remove uploaded test images from the system.

10.6.9 UM.PD.01

TC_ID	UM.PD.01
Purpose	Verify that the user registration API enforces input validation.
Priority	High
Estimated Time Needed	10 Seconds
Dependency	User validation logic must be implemented.
Setup	Ensure the user registration API endpoint is live.
Procedure	[A01] Submit an invalid POST request. [A02] Verify that the API returns a 400 HTTP status code and appropriate validation errors. [V01] Repeat with valid inputs and confirm successful registration.
Validation	See Procedure.
Cleanup	Delete test user data.

10.6.10 TR.RT.02

TC_ID	TR.RT.02
Purpose	Verify that the trend data API fetches and returns up-to-date data from external sources.
Priority	High
Estimated Time Needed	10 Seconds
Dependency	External APIs and trend aggregation services must be operational.
Setup	Ensure API connections to external trend sources are active.
Procedure	[A01] Send a GET request to the trend API. [A02] Verify that the response contains the latest data with accurate timestamps.
Validation	[V01] Confirm that the response returns a 200 HTTP status code.
Cleanup	None.

10.6.11 PRG.IMG.02

TC_ID	PRG.IMG.02
Purpose	Verify that the system processes irrelevant image uploads and still returns matching product suggestions based on visual features.
Priority	Medium
Estimated Time Needed	20 Seconds
Dependency	Image recognition and recommendation engine must be operational.
Setup	Ensure that the recommendation API is functional and a sample set of product data is available in the database.
Procedure	[A01] Upload an image that is irrelevant to the product database. [A02] Submit the image to the system via the image-based recommendation API. [A03] Observe the system's handling of the image and the generated recommendations.
Validation	[V01] Verify that the recommendations align with recognizable visual features even if the image context is unrelated to the product catalog.
Cleanup	Delete any cached or temporary data related to the test image.

10.6.12 CS.API.03

TC_ID	CS.API.03
Purpose	Verify that the system can create and retrieve embeddings using AI services from external API calls.
Priority	High
Estimated Time Needed	30 Seconds
Dependency	External AI services must be operational and accessible.
Setup	Ensure that API credentials for external AI services are configured correctly and the system has an active connection to the external APIs.
Procedure	[A01] Send a POST request to the external AI service with a valid input. [A02] Observe the response and verify the generated embedding. [A03] Store the retrieved embedding in the local database or memory. [A04] Send a subsequent GET request to retrieve the embedding based on a unique identifier. [A05] Verify that the retrieved embedding matches the originally generated embedding.
Validation	[V01] Confirm that the embedding creation process completes successfully with a 200 HTTP status code. [V02] Ensure that embeddings are retrieved accurately without data loss or corruption.
Cleanup	Clear any test embeddings or temporary files created during the test from the local system and external service.

10.7 Test Results

TC ID	Feature/Subfeature	Description	Result
UM.PD.01	User Management – Profile Details	Register with valid/invalid inputs	Pass
UM.PD.02	User Management – Profile Details	Login with invalid/blank password	Pass
UM.PD.03	User Management – Profile Details	Login with invalid/blank email	Pass
PRG.TX.01	Text-Based Recommendations	Valid text query	Pass
PRG.TX.02	Text-Based Recommendations	Irrelevant text query	Pass
PRG.IMG.01	Image-Based Recommendations	Valid image upload	Pass
PRG.IMG.02	Image-Based Recommendations	Irrelevant image upload	Pass
TR.RT.01	Real-Time Trend Analysis	Up-to-date trend data	Pass
TR.RT.02	Real-Time Trend Analysis	Data refresh	Pass
SFO.MC.01	Multi-Criteria Search	Accurate filtering	Pass
SFO.MC.02	Multi-Criteria Search	"No Results Found" message	Pass
PC.CV.01	Product Comparison View	Attribute comparison	Pass
PC.CV.02	Product Comparison View	Warning on missing data	Pass
UI.CC.IC.01	Interactive Chat	Valid text query	Pass
UI.CC.IC.02	Interactive Chat	Ambiguous query handling	Pass
UI.CC.MI.01	Multimodal Interaction	Image processed	Pass
UI.CC.MI.02	Multimodal Interaction	Combined text+image processed	Pass
CS.API.01	API Integration – Trends	Retrieve trend data	Pass
CS.API.02	API Integration – Latency	Handle slow response	Pass
CS.API.03	API Integration – Embeddings	Create/retrieve embeddings	Pass

All **High** and **Medium** priority cases passed (100 % pass rate), meeting the exit criteria.

10.7.1 Summary of the Test Results

The comprehensive test campaign confirms that the Multimodal RAG-Based Product Recommendation System meets its functional and performance objectives. Every critical feature operates reliably, including multimodal recommendations, real-time trend integration, and robust user management. With a 100 % pass rate for High and Medium priority cases, the system is considered ready for production deployment and future enhancements.

Conclusion

The development of this Multimodal Retrieval-Augmented Generation (RAG) system marks a significant step toward enhancing user experience in the fashion industries. By integrating important AI technologies and multimodal LLMs, the system not only personalizes recommendations but also promotes sustainable and trend-aware purchasing decisions. Users benefit from an interactive interface that seamlessly processes both text and image inputs, delivering highly relevant product insights.

The layered architecture ensures scalability, modularity, and easy maintenance, allowing for future improvements and integration of emerging AI models. Through features like sustainability filters and trend analysis, the system aligns with the growing demand for eco-conscious consumerism.

Quality assurance is documented in the Test Plan and Results section, which defines every high-priority test case, states explicit pass/fail/exit criteria, and presents a summary confirming that those criteria were met. This evidence base demonstrates the reliability of the deployed functionality and provides a repeatable framework for future regression testing as the platform grows. The accompanying SRS and SDD documents provide a detailed breakdown of the system's design, requirements, and functionality, serving as a foundation for continued development and refinement. This project sets the stage for broader adoption of AI technologies in the retail sector.

References

[1] T. N. Prabhu, "Google Trends API," *Google Colab*, 2019. [Online]. Available: https://colab.research.google.com/github/Tanu-N-Prabhu/Python/blob/master/Google_Trends_API.ipynb. [Accessed: Dec. 4, 2024].

[2] Stack Overflow, "What is the difference between functional and non-functional requirements?" *Stack Overflow*, 2013. [Online]. Available: https://stackoverflow.com/questions/16475979/what-is-the-difference-between-functional-and-non-functional-requirements. [Accessed: Dec. 4, 2024].

[3] OpenAI, "CLIP: Connecting Text and Images," *OpenAI*, 2021. [Online]. Available: https://openai.com/research/clip. [Accessed: Dec. 4, 2024].

[4] Pallets Projects, "Flask Documentation (3.1.x)," *Flask*, 2024. [Online]. Available: https://flask.palletsprojects.com/en/stable/. [Accessed: Dec. 4, 2024].

[5] Google AI, "Gemini Models," *Google AI for Developers*, 2024. [Online]. Available: https://ai.google.dev/gemini-api/docs/models/gemini. [Accessed: Dec. 4, 2024].

[6] Hugging Face, "The AI Community Building the Future," *Hugging Face*, 2024. [Online]. Available: https://huggingface.co/. [Accessed: Dec. 4, 2024].

[7] Meta AI, "Llama 3.2," *Meta*, 2024. [Online]. Available: https://www.llama.com/. [Accessed: Dec. 4, 2024].

[8] GeeksforGeeks. "Unified Modeling Language (UML) - Activity Diagrams." GeeksforGeeks, 2024. [Online]. Available: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/. [Accessed: Dec. 20, 2024].

[9] GeeksforGeeks. "Unified Modeling Language (UML) - Sequence Diagrams." GeeksforGeeks, 2024. [Online]. Available: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/. [Accessed: Dec. 20, 2024].

[10] Figma. "UI Design Principles." Figma, 2024. [Online]. Available: https://www.figma.com/resource-library/ui-design-principles/. [Accessed: Dec. 23, 2024].

[11] GeeksforGeeks. "Design Patterns - Architecture." GeeksforGeeks, 2024. [Online]. Available: https://www.geeksforgeeks.org/design-patterns-architecture/. [Accessed: Dec. 22, 2024].

[12] P. Walpita. "Software Architecture Patterns: Layered Architecture." Medium, 2024. [Online]. Available: https://priyalwalpita.medium.com/software-architecture-patterns-layered-architecture-a3b89b71a057. [Accessed: Dec. 22, 2024].

[13] Zeeshan. "The Weaknesses and Strengths of Layered Architecture in Software Development." Medium, 2024. [Online]. Available: https://zeeshan01.medium.com/the-weaknesses-and-strengths-of-layered-architecture-in-software-development-81ba1206a17b. [Accessed: Dec. 22, 2024].