Çankaya University

Computer Engineering Department
CENG 407 - Software Design Description (SDD)

Multimodal RAG-Based Product Recommendation System

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

Contents

1 Introduction **4**

1.1 Purpose of this document 4

1.2 Definitions, Acronyms, and Abbreviations 4

2 System Overview **5**

3 System Design **5**

3.1 Architectural Design 5

3.1.1 Layered Architecture 6

3.1.2 Key Components 6

3.1.3 Benefits of Layered Architecture 6

3.2 Decomposition Description 7

3.2.1 Presentation Layer 7

3.2.2 Application Layer 8

3.2.3 Data Layer 9

3.2.4 External AI Services 10

3.2.5 Communication Between Layers 10

3.3 System Modeling 11

3.3.1 Activity Diagrams 11

3.3.2 Sequence Diagrams 14

3.3.3 Class Diagram 17

4 User Interface Design **18**

4.1 User Interface of Profile Page 18

4.2 User Interface of Chatbot Screen 19

4.3 User Interface of Personal Recommendations Page 19

4.4 User Interface of Home Page 20

4.5 User Interface of Login Page 21

4.6 User Interface of Trending Page 22

5 References **23**

1 Introduction

1.1 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.

1.2 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.

2 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.

3 System Design

3.1 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.

3.1.1 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.

3.1.2 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.

3.1.3 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.

3.2 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.

3.2.1 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.

3.2.2 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.

3.2.3 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.

3.2.4 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.

3.2.5 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.

3.3 System Modeling

3.3.1 Activity Diagrams

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

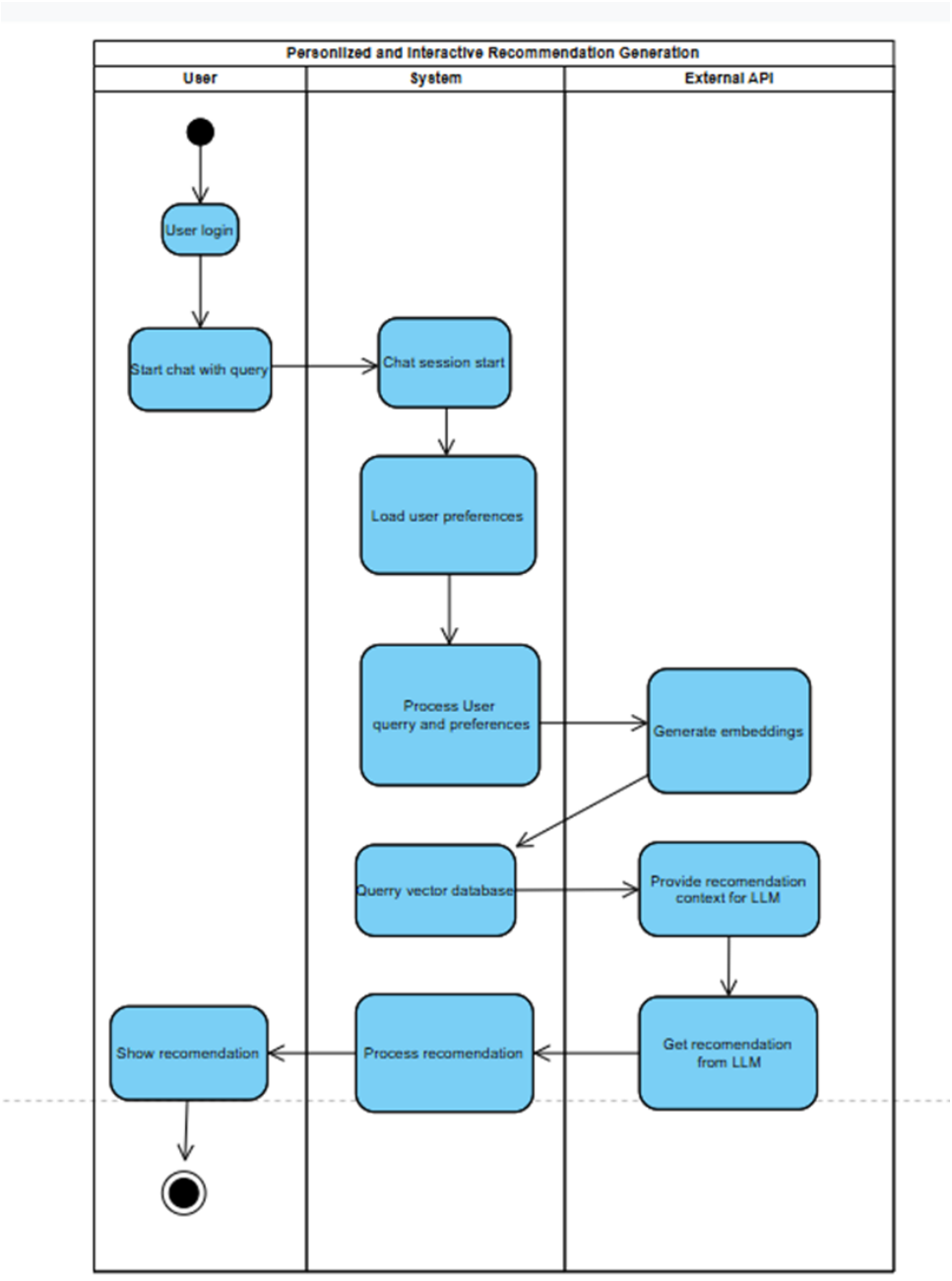

Figure 1: Use Case 2 Activity Diagram

Use Case 3: Trends Review (Activity Diagram)

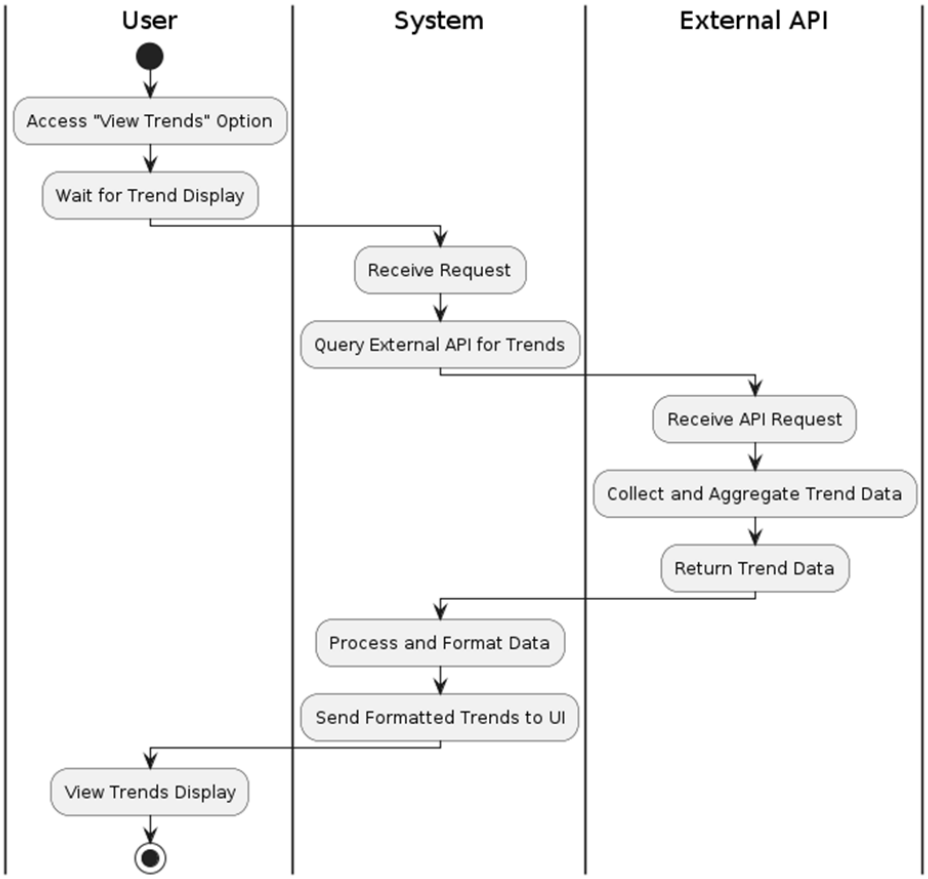

Figure 2: Use Case 3 Activity Diagram

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

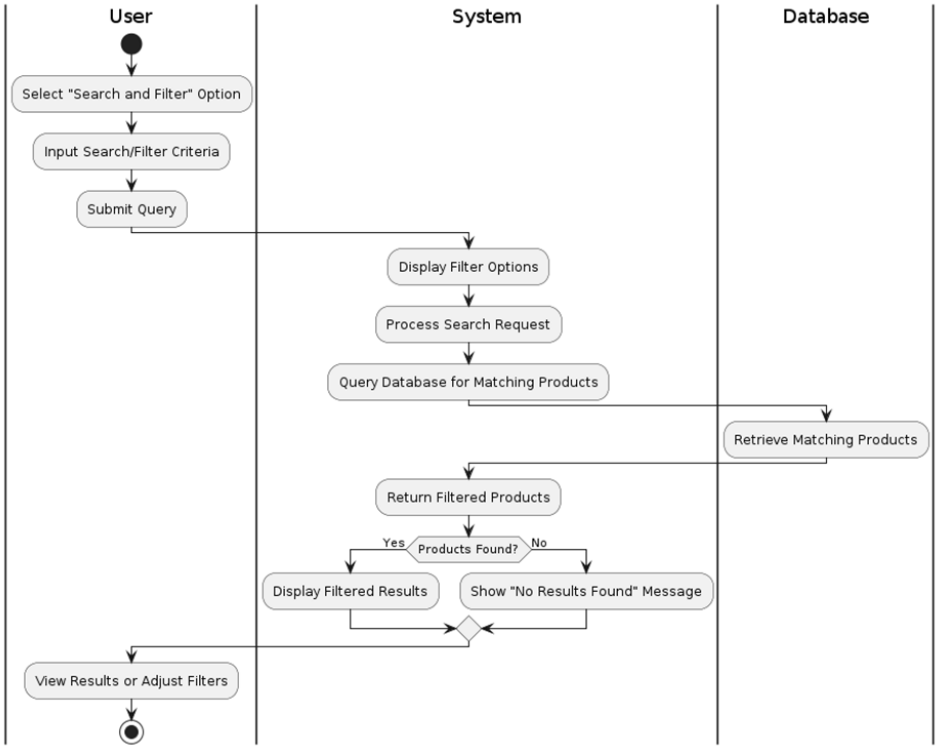

Figure 3 : Use Case 7 Activity Diagram

Use Case 9: Product Comparison (Activity Diagram)

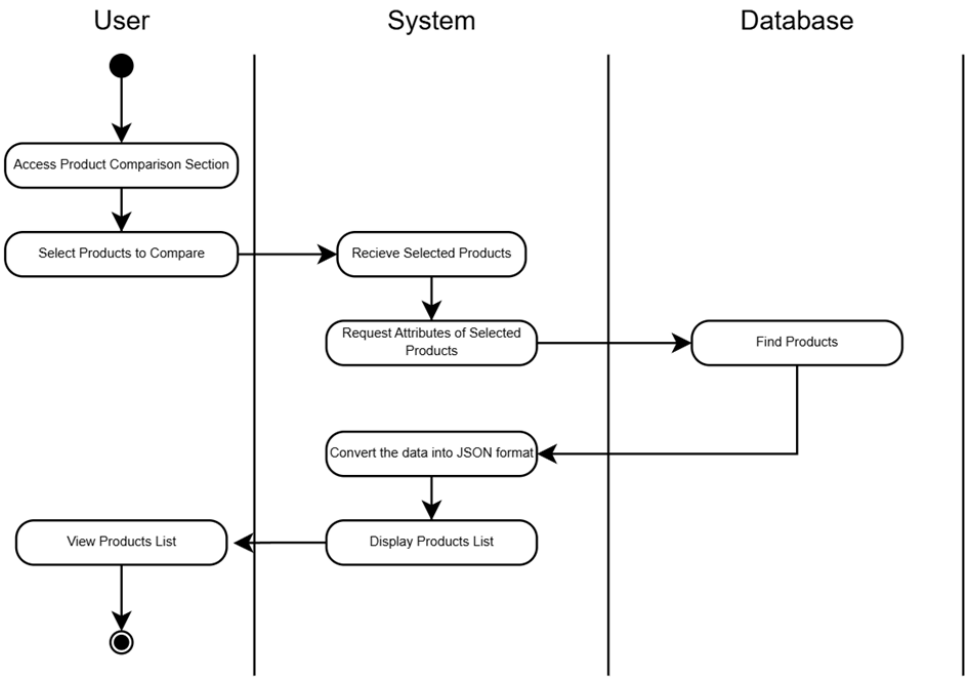

Figure 4: Use Case 9 Activity Diagram

3.3.2 Sequence Diagrams

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

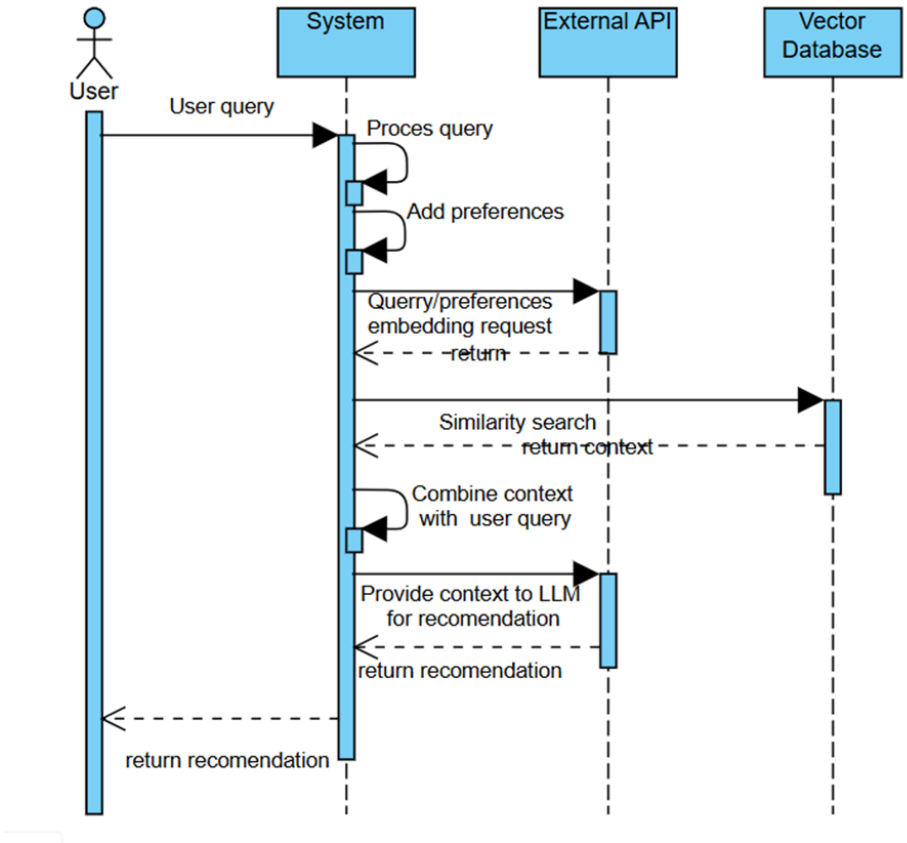

Figure 5: Use Case 2 Sequence Diagram

Use Case 3: Trends Review (Sequence Diagram)

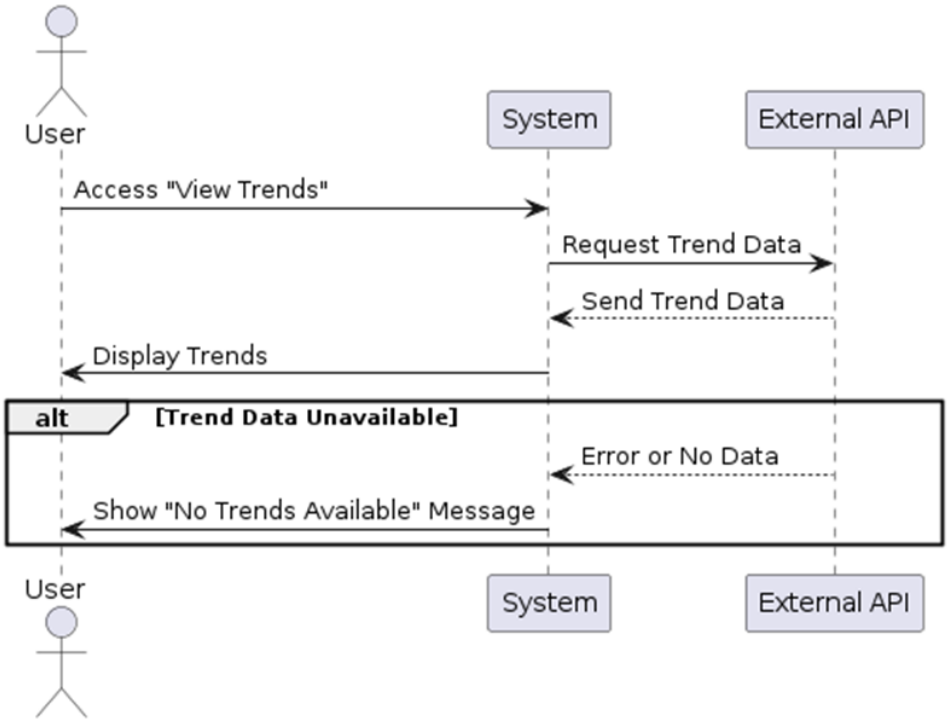

Figure 6: Use Case 3 Sequence Diagram

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

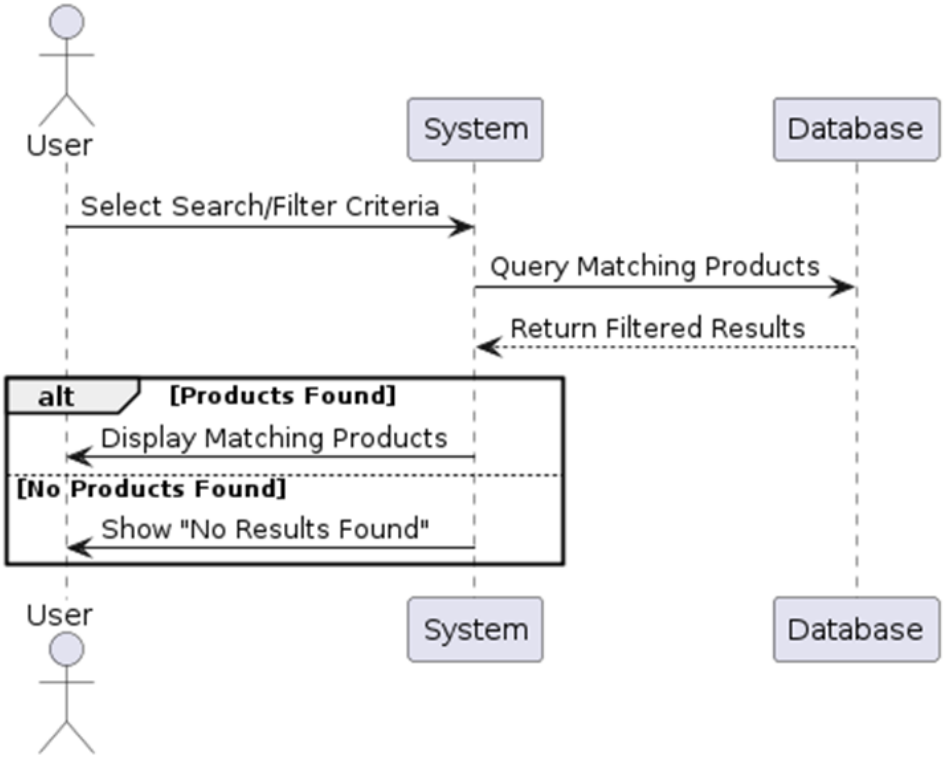

Figure 7: Use Case 7 Sequence Diagram

Use Case 9: Product Comparison (Sequence Diagram)

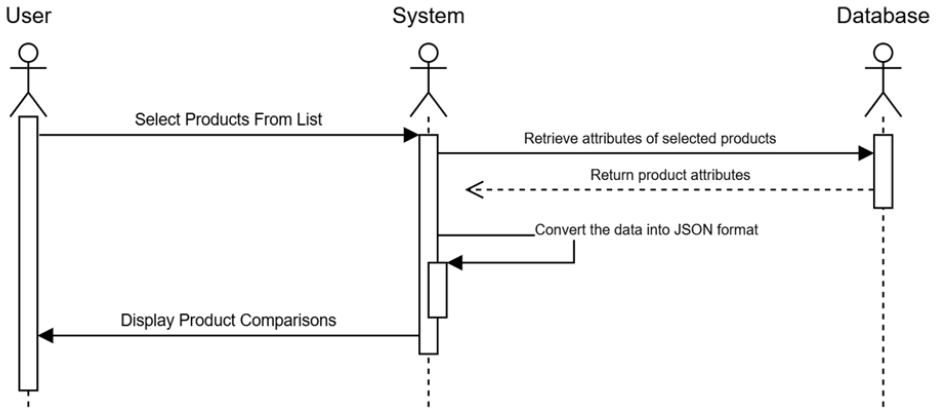

Figure 8: Use Case 9 Sequence Diagram

3.3.3 Class Diagram

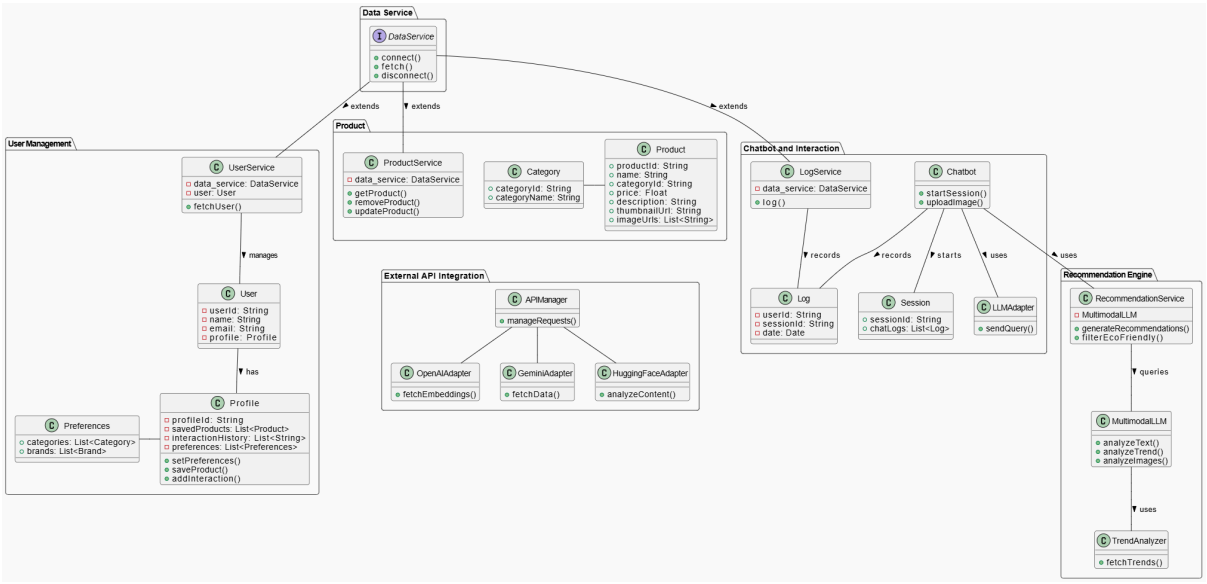

Figure 9: Class Diagram

4 User Interface Design

4.1 User Interface of Profile Page

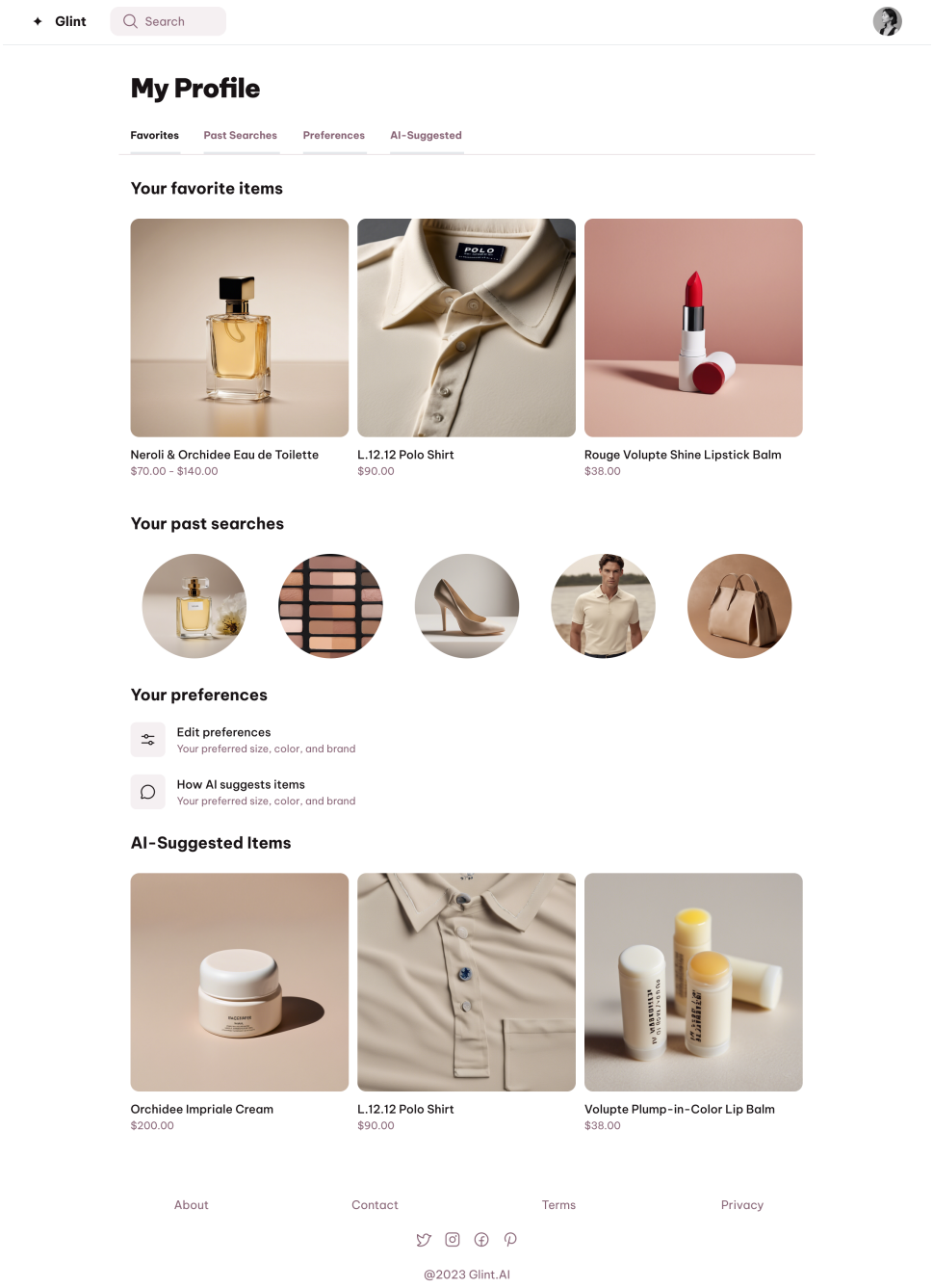

Figure 10: Profile Page UI

4.2 User Interface of Chatbot Screen

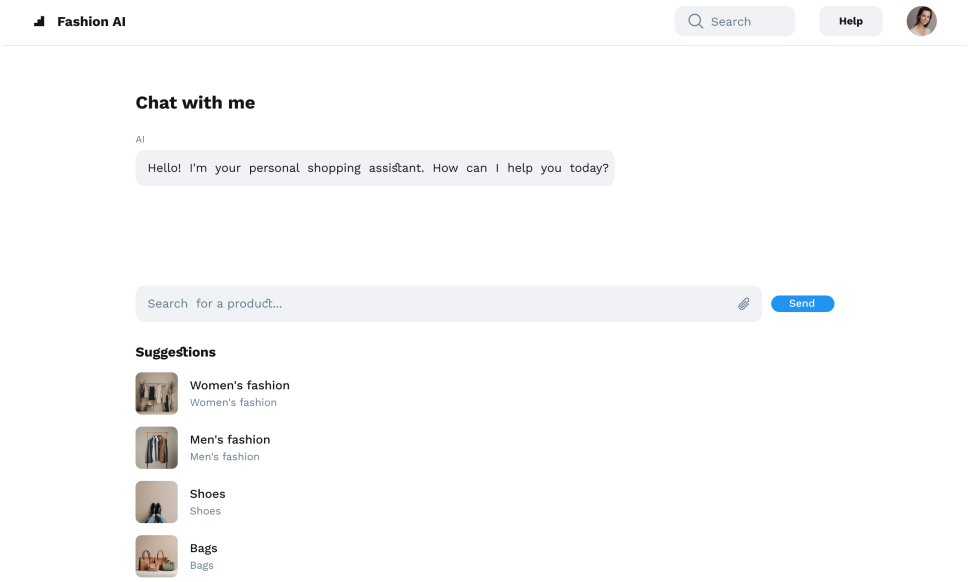

Figure 11: Chatbot UI

4.3 User Interface of Personal Recommendations Page

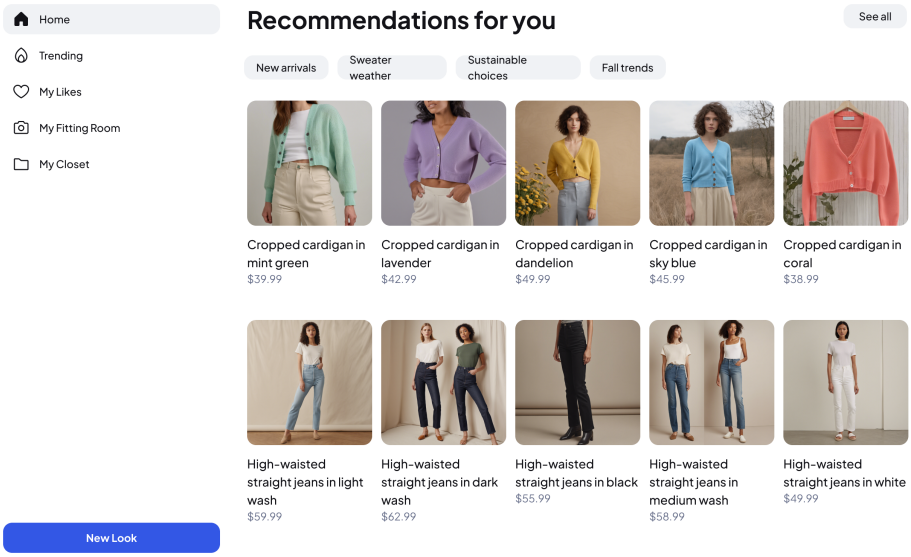

Figure 12: Personal Recommendations UI

4.4 User Interface of Home Page

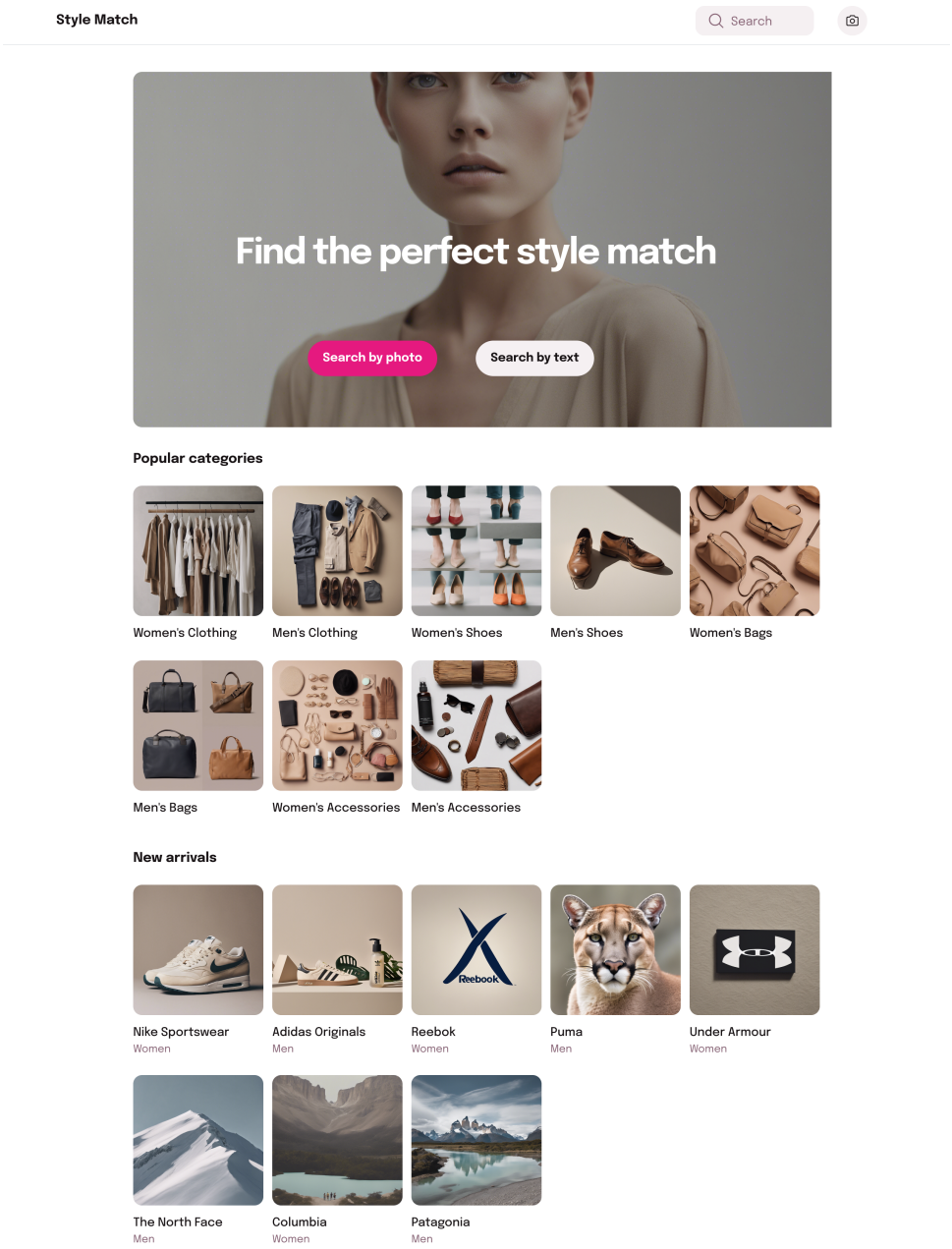

Figure 13: Home Page UI

4.5 User Interface of Login Page

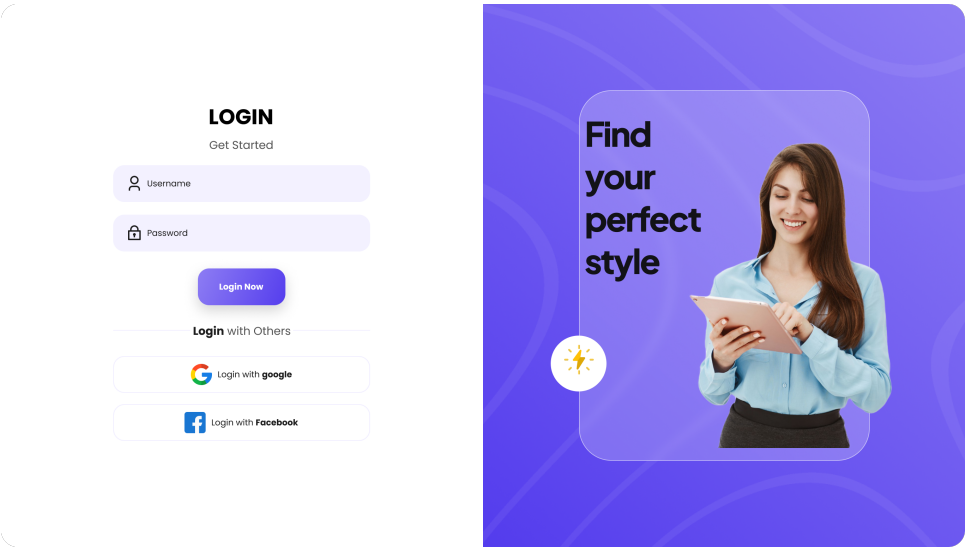

Figure 14: Login Page UI

4.6 User Interface of Trending Page

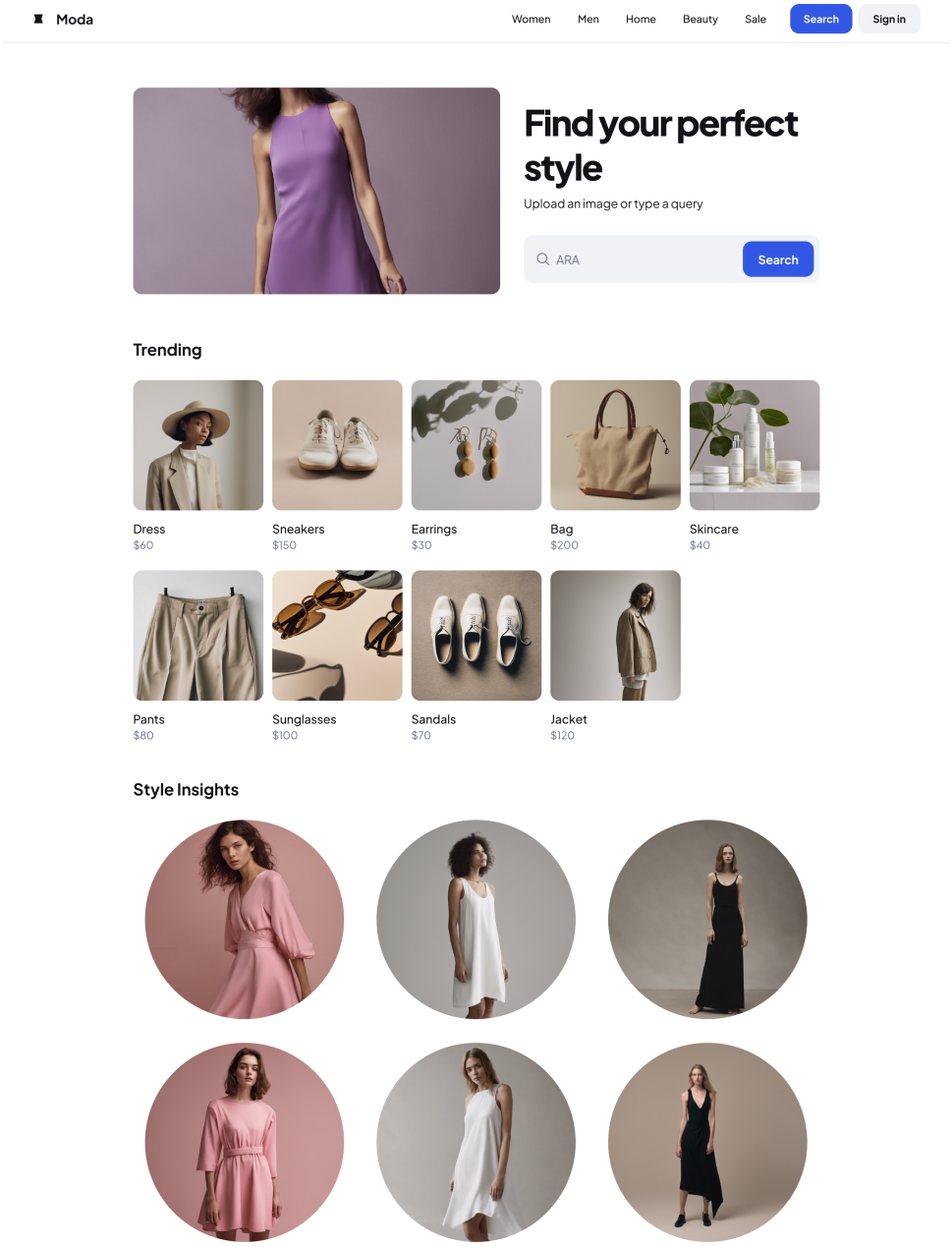

Figure 15: Trending Page UI

5 References

1. 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].

2. 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].

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

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

5. 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].

6. 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].