Style Quiz App

1. Introduction

The Style Recommendation Quiz App is a mobile-first application designed to provide personalized outfit recommendations based on user preferences. Users interact with a visual quiz where they like or dislike outfit images, allowing the system to infer their style preferences and generate tailored recommendations.

The project focuses on building a clean Minimal Viable Product (MVP) that balances user experience, engineering clarity, and recommendation logic, while remaining easy for reviewers and recruiters to run locally.

2. Assumptions

The following assumptions guided the design and implementation:

- Users are willing to answer a short quiz
- Visual preferences are a strong indicator of personal style
- Binary feedback (like/dislike) provides a cleaner signal than ratings
- Mobile-first UX is preferred over desktop or web
- Recommendations should be explainable and transparent

3. Scope of Work

This project is a mobile application built using Expo (React Native) that implements a gender-based quiz flow to understand user style preferences. Users interact with the quiz through intuitive like and dislike icons, allowing preferences to be progressively aggregated from their responses. A rule-based recommendation system processes these inputs to score and rank relevant products. The results screen presents each recommended item with a product image, a concise explanation of why it matches the user's preferences, and a confidence score indicating relevance. The application features a clean, custom-themed user interface and is designed to be fully runnable locally, supported by clear setup and usage documentation.

4. Problem-Solving & Decision-Making Approach

4.1 Choice of Recommendation Technique (TF-IDF + Similarity)

A traditional collaborative filtering approach was not suitable for this project due to the absence of historical user behavior and the requirement to generate meaningful recommendations immediately. To address this cold-start scenario, a content-based recommendation system was implemented using TF-IDF vectorization combined with cosine similarity. The dataset used for this project was sourced from Kaggle and contains approximately 45,000 product records, each described by structured textual attributes such as category, subcategory, article type, color, season, and product name.

TF-IDF (Term Frequency–Inverse Document Frequency) was selected because it effectively captures the relative importance of these attributes while minimizing the influence of overly common terms, making it well-suited for matching user preferences against product metadata in a scalable and interpretable manner. User preferences collected through the visual quiz are converted into a textual representation and transformed into TF-IDF vectors using the same trained vectorizer as the product catalog. Cosine similarity is then applied to measure how closely each product aligns with the user's expressed preferences, enabling the system to rank and recommend the most relevant items.

4.2 Hybrid Rule-Based Filtering

To ensure recommendation accuracy and prevent irrelevant results, **hard rule-based filters** were applied before similarity scoring. These include strict constraints on attributes such as gender and subcategory derived from quiz responses. This hybrid approach combines deterministic filtering with probabilistic ranking, ensuring both correctness and personalization.

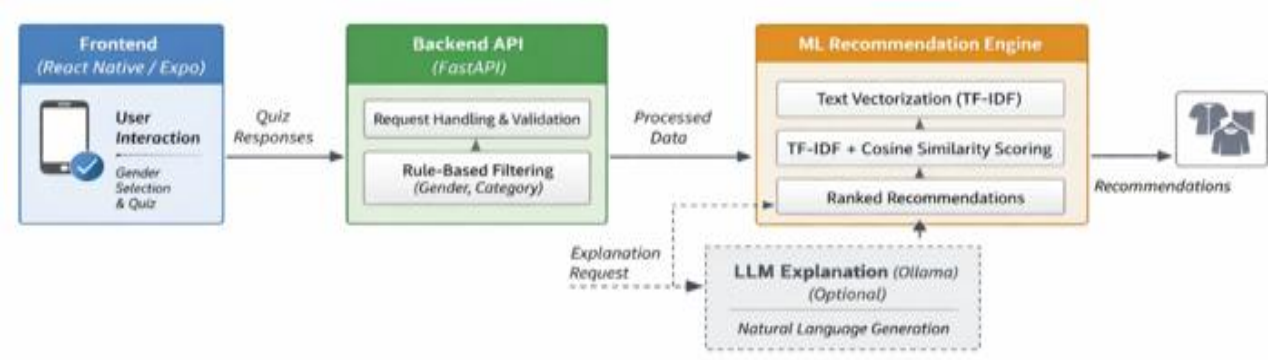
4.3 Backend Architecture & Technology Choices

The backend was implemented using **Python with a lightweight API layer** (FastAPI), enabling fast iteration, modular design, and easy integration with machine learning components. Python was selected due to its mature ecosystem for data processing and machine learning, including libraries such as pandas and scikit-learn.

Model artifacts such as the trained TF-IDF vectorizer are loaded at runtime, ensuring consistency between training and inference. The recommendation logic is encapsulated in a dedicated module to maintain separation of concerns and improve maintainability.

4.4 Explainability Layer (LLM Integration)

To enhance user trust and improve recommendation transparency, a natural language explanation layer was introduced. A locally hosted LLM (via Ollama) generates concise, human-readable explanations for each recommendation based on the user's preferences and product attributes. This design choice avoids external API dependencies while showcasing applied LLM usage in a controlled, cost-free environment.



5. Technology Choices & Rationale

5.1 Frontend - Expo (React Native)

Expo was chosen to enable rapid mobile development with minimal configuration while maintaining a high-quality user experience. Its cross-platform support allows the same codebase to run on both Android and iOS, making it ideal for prototyping and demos. Integration with Expo Go enables reviewers to instantly preview the application without complex setup. The framework also provides a rich ecosystem, strong documentation, and built-in tooling that accelerates development.

5.2 Routing - expo-router

expo-router was used for navigation due to its file-based routing model, which keeps screen structure clean and intuitive. It simplifies parameter passing between screens (such as gender selection and quiz results) and improves maintainability by enforcing clear separation of concerns between routes.

5.3 Icons & UI - @expo/vector-icons (Ionicons)

Ionicons were used for UI interactions such as like/dislike and gender selection. This library requires no manual asset management, provides platform-consistent icons, and keeps the UI lightweight and performant while enhancing user engagement through visual feedback.

5.4 Backend - Python (FastAPI)

The backend was implemented using Python with FastAPI to serve recommendation results. FastAPI was chosen for its high performance, clean API design, and native support for JSON-based workflows. It allows easy integration with machine learning components and provides a scalable structure for future enhancements such as authentication, personalization, or analytics.

5.5 Machine Learning - TF-IDF, Logistic Regression & Cosine Similarity

TF-IDF (Term Frequency–Inverse Document Frequency) was used to convert product metadata-such as category, color, season, and product description-into numerical vectors. This approach was selected because it is efficient, interpretable, and well-suited for structured text-heavy product datasets. Logistic Regression was employed during the model training and experimentation phase as a baseline supervised learning model. Its purpose was to validate feature relevance and assess whether the textual attributes meaningfully captured style-related patterns within the dataset. Logistic Regression was intentionally excluded from real-time inference, as the system operates in a cold-start environment where fixed labels per user session are not available.

For live recommendations, cosine similarity was used to compare user preference vectors-derived dynamically from quiz responses—with product vectors. This similarity-based approach is more appropriate for session-based personalization and avoids unnecessary model complexity while remaining scalable. During experimentation, the model achieved an overall accuracy of approximately **95%** across the full dataset, while the average relevance score for individual recommended products was around **60%**. The reduction in per-item accuracy is primarily due to the high diversity and breadth of the dataset, which includes a wide range of styles, categories, and attributes, making fine-grained personalization more challenging in a cold-start setting.

5.6 Recommendation Strategy - Hybrid Rule-Based + ML Approach

A hybrid approach was adopted to balance precision and interpretability. Hard filters (such as gender and preferred subcategory) ensure correctness and prevent irrelevant recommendations, while cosine similarity provides flexible ranking within the filtered set. This combination ensures reliable results while remaining explainable and easy to debug.

6. Recommendation System Design

6.1 Input Signals

Each quiz interaction captures:

- Like / Dislike signal
- Gender
- Base colour
- Article type
- Master category
- Subcategory
- Season

6.2 Preference Aggregation

- Liked attributes increase preference weights
- Disliked attributes reduce weights
- Preferences accumulate across all quiz responses

6.3 Scoring Mechanism

Each product is scored based on:

- Attribute overlap with user preferences
- Frequency of liked attributes
- Penalties for disliked attributes

Scores are normalized to ensure consistency and readability.

6.4 Output

Each recommendation includes:

- Product image
- Product name
- Explanation for why it matches the user
- Confidence score

7. LLM / API Usage (Optional Enhancement)

7.1 Purpose

An optional LLM layer was explored to:

- Generate natural-language explanations
- Improve transparency and user trust
- Demonstrate ML and AI integration awareness

7.2 Why Ollama

- Runs locally (no API keys required)
- Privacy-friendly
- No vendor lock-in
- Suitable for offline experimentation

7.3 Design Decision

LLM usage is **non-blocking** and optional:

- The app functions fully without it
- Treated as an enhancement layer, not a dependency
- Demonstrates architectural extensibility

8. UX & Design Decisions

- Soft, friendly color themes to reduce fatigue
- Emoji/icon-based actions for intuitive interaction
- Large touch targets for mobile accessibility
- Minimal text during quiz to maintain flow
- Clear progress indicators to reduce drop-offs

9. Future Improvements

If more development time were available, the following enhancements would be prioritized to improve recommendation quality, product depth, and system scalability.

9.1 Expanded Quiz & Preference Capture

Currently, the quiz flow is limited to gender-based selection. In the future, the quiz can be extended to explicitly ask users what types of products they are interested in, such as shirts, t-shirts, pants, trousers, sarees, tops, and other categories. Based on these selections, subsequent quiz questions can be dynamically tailored to show only relevant product images. This would significantly reduce noise in user feedback and result in more precise recommendations by focusing only on the user’s intended purchase categories.

9.2 Personalization & Advertising Use Cases

As the application evolves into a full-fledged clothing platform, user preferences collected through the quiz can be stored and reused across sessions. These insights can be leveraged to personalize the home feed, recommend seasonal collections, and tailor advertisements shown to users. By understanding preferred colors, categories, and styles, ads and promotions can be aligned with individual taste rather than generic targeting, improving both user experience and conversion rates.

9.3 Recommendation Quality Improvements

The current recommendation logic uses TF-IDF and cosine similarity combined with rule-based filters. This can be enhanced by introducing dense vector embeddings for both users and items, enabling richer semantic understanding of preferences. Collaborative filtering techniques can be added to recommend items based on similar users, while user clustering can help identify style personas. Over time, a fully hybrid system combining ML-based ranking with business rules can be developed to balance accuracy and control.

9.4 Machine Learning Enhancements

Visual understanding can be significantly improved by introducing image embeddings using models such as CLIP, allowing the system to learn from visual style patterns rather than text alone. Fine-tuned recommendation models can be trained using accumulated user interactions, and feedback loops can be introduced to continuously refine recommendations based on likes, dislikes, and post-click behavior.

9.5 Product & UX Enhancements

Future product features can include user profiles, saved favorites, and recommendation history, enabling continuity across sessions. A/B testing different quiz flows and question orders can help optimize engagement and recommendation effectiveness. These enhancements would also provide valuable analytics for improving both the model and the user experience.

10. Additional Notes & Interesting Tidbits

One notable design decision in this project was the choice of **Ollama (local LLM)** for generating natural language explanations instead of a hosted LLM such as ChatGPT.

10.1 Why Ollama Was Chosen Over Cloud LLMs

Since this project is intended as a **demo and portfolio MVP**, cost efficiency and ease of local execution were key considerations. Cloud-based LLMs like ChatGPT typically involve per-request or per-token costs and require API key management, which can create friction for reviewers attempting to run the project locally.

Ollama was chosen because it:

- Runs **fully locally** with no external API dependency
- Is **free to use** for development and experimentation
- Does not require API keys or billing setup

- Ensures consistent behavior across environments
- Aligns well with offline and privacy-friendly development

This allows anyone reviewing the project to clone the repository, run the backend, and immediately see LLM-generated explanations without additional configuration or cost.

10.2 Trade-offs & Transparency

While cloud-hosted LLMs may offer higher accuracy or larger models, Ollama provides an excellent balance for prototyping and demos. The architecture is intentionally designed so that Ollama can be **swapped with a cloud LLM** (such as OpenAI or Anthropic) in the future with minimal changes, if production deployment or scale becomes a requirement.

10.3 Practical Engineering Mindset

This decision reflects a broader engineering principle followed throughout the project:
choose tools that are fit-for-purpose, especially for early-stage prototypes. By prioritizing developer experience, reproducibility, and cost control, the project remains accessible, testable, and easy to evaluate-key qualities for an MVP.

11. Conclusion

This project demonstrates the ability to translate an open-ended, ambiguous problem into a clear and structured solution while making thoughtful technology trade-offs along the way. It showcases the development of a complete, end-to-end working product that balances user experience, engineering practicality, and machine learning considerations. Overall, the application serves as a strong foundational MVP that can be iteratively enhanced and scaled into a production-grade recommendation system.