Design Document: DressUp Recommendation Engine

21PC14 - Hari Ganesh M

My assumption:

1. I assumed users would find a simple, swipe-based UI (like dating apps) intuitive for quickly expressing their fashion preferences.
2. I also assumed that a pre-trained vision-language model (CLIP) would be sufficient for generating meaningful style embeddings without needing to train a model from scratch.

Scope of work:

1. The project's scope was to build a functional mobile prototype with a core user journey: **user authentication**, **an initial style quiz**, **recommendation feed** and a **refinement quiz**.
2. The backend scope included a **data pipeline** to process a large fashion dataset, generate embeddings, and serve API endpoints for the quiz and recommendations.

Experiments:

1. I experimented with the user experience of the swipe gesture. Initially tested ambient glow effects for feedback but found them to be unreliable across devices and visually distracting. Then iterated and settled on clear, explicit text overlays ("LIKE"/"NOPE") as they provided the most intuitive and glitch-free user feedback.
2. Then experimented with randomizing the refinement quiz's dataset to enhance the user's experience by preventing duplicates, which caused a lot of hurdles. Finally settled down with a randomization technique which sometimes may bring in duplicate values in the quiz's dataset.

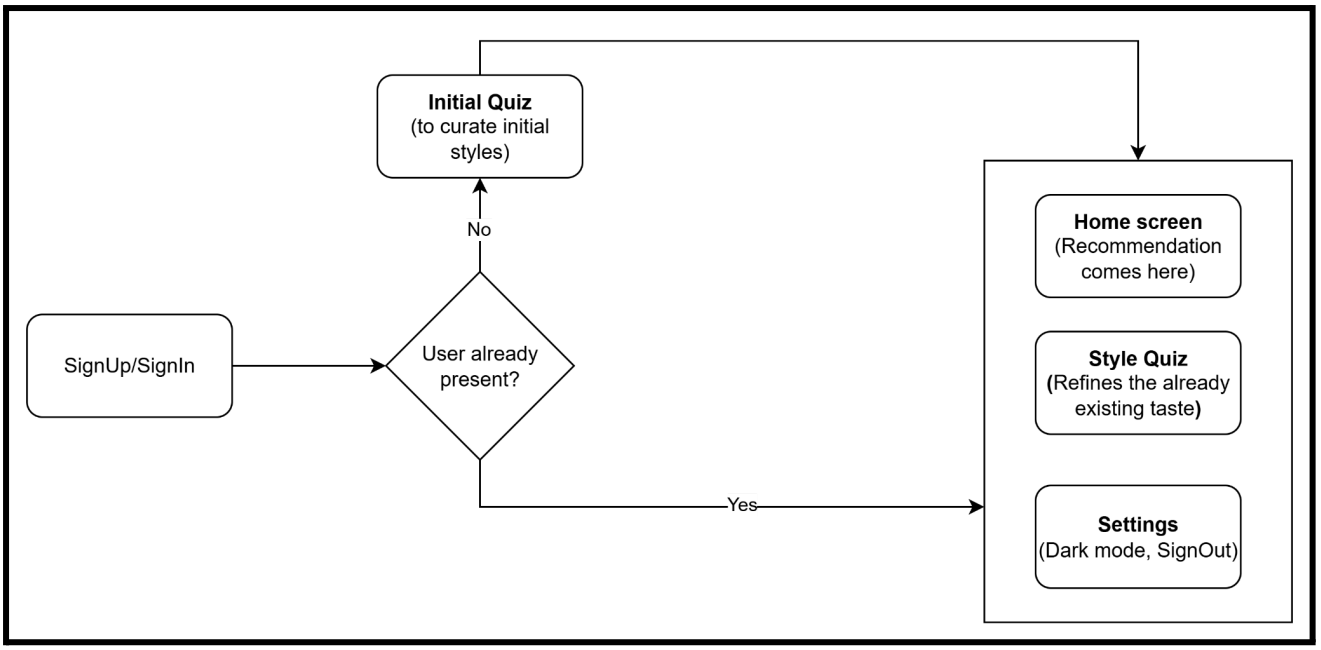
Problem-solving and decision-making approach:

1. I followed a systematic, iterative debugging process. Which enabled me to solve the problems which came by in fast fashion.
2. The time constraint along with the broad problem statement enabled me to focus on my areas of strength within this project.

Rationale for API/Technology Choices:

1. I chose **FastAPI** for the backend due to its high performance.
2. For the recommendation engine, I chose to generate embeddings using the **CLIP model**. This was a strategic decision to move beyond simple keyword matching and leverage a model that understands the semantic relationship between images and text, allowing for more nuanced style comparisons.

Overall design:



Hybrid Architecture:

1. The system uses a hybrid architecture that combines a scalable cloud database with a high-speed local vector store.
2. User profiles, authentication, and dynamic quiz data are managed in **Supabase**, while the large, static inventory of fashion items is pre-processed into a **FAISS vector index** for millisecond-fast similarity searches.
3. This design provides both performance and scalability.

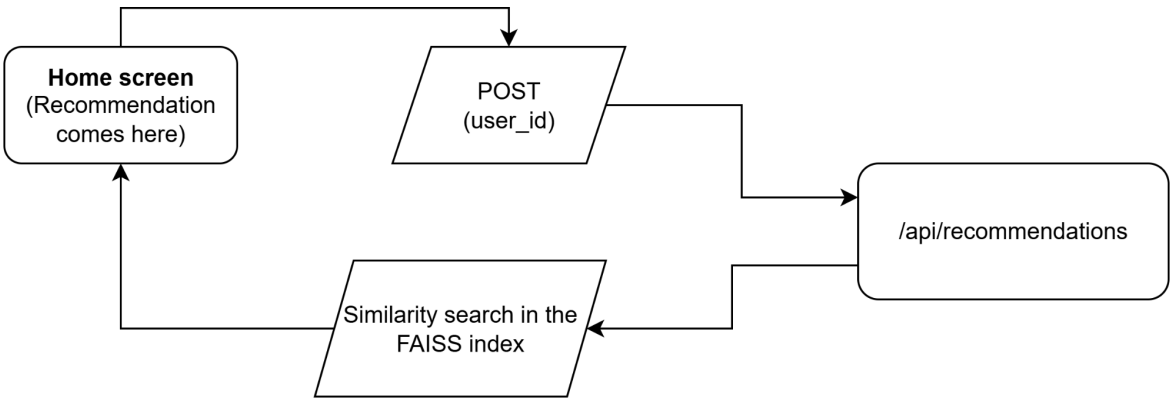
Data Flow:

The recommendation process is as follows:

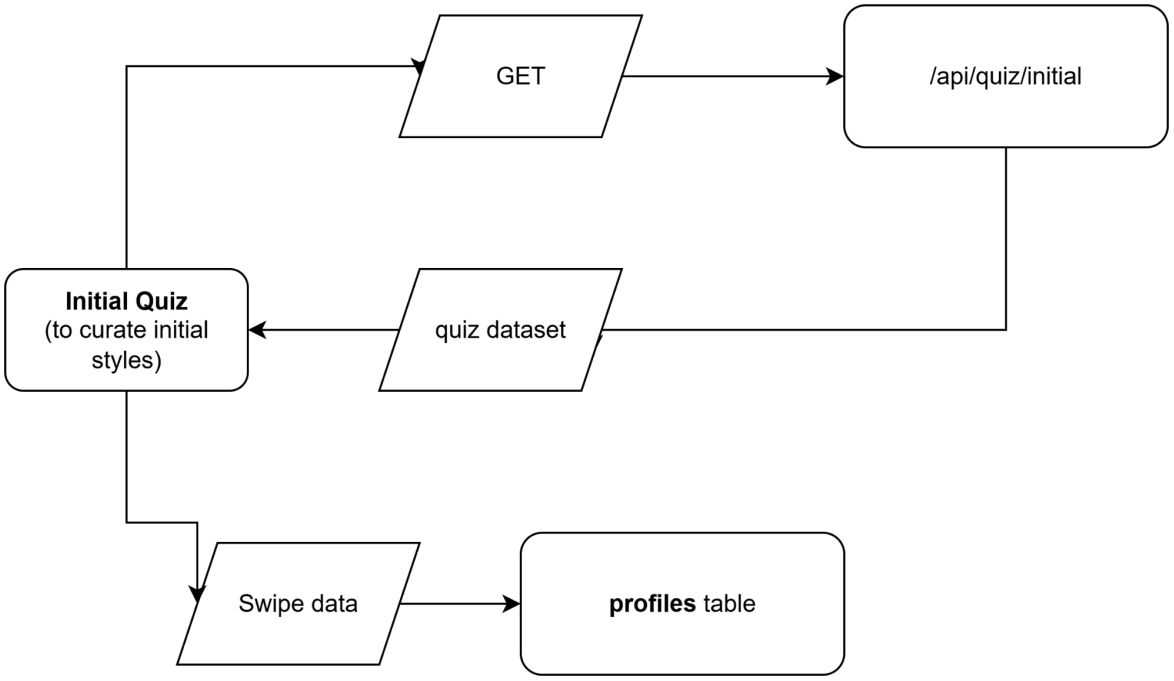
1. The user's liked swipes are fetched from their Supabase profile.
2. The backend generates a single "taste vector" by averaging the embeddings of the metadata from these liked items.
3. This taste vector is used to query the local FAISS index, which finds the k most similar item vectors from our entire fashion inventory. The metadata for these top k items are then returned to the user as their recommendations.

Flow chart of each component:

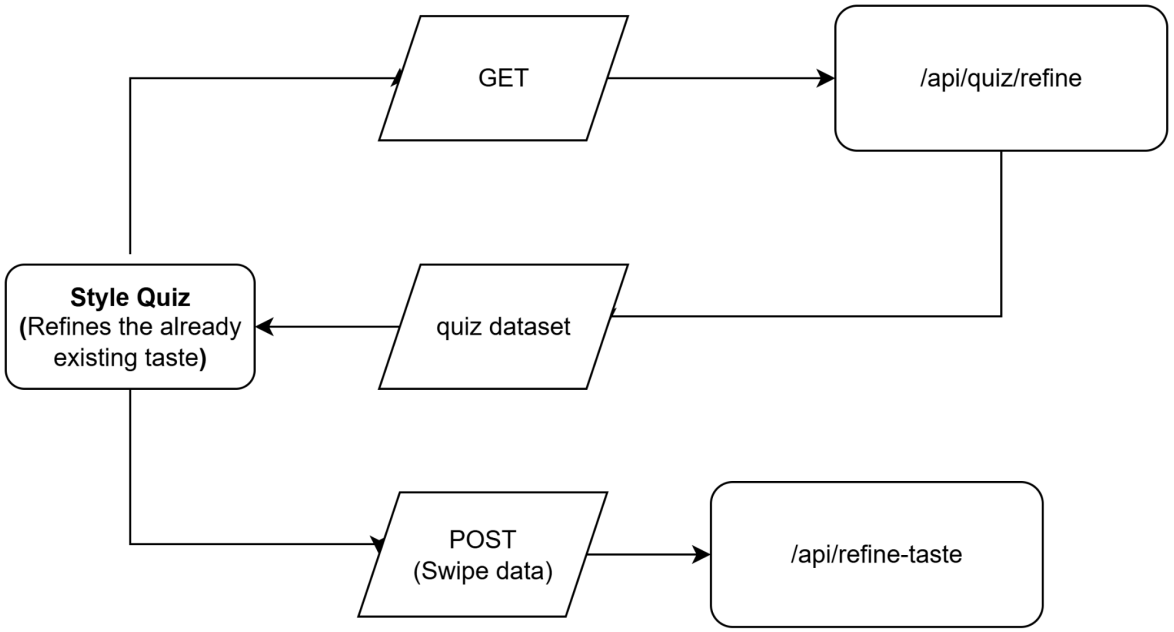
1. **Home Screen** (aka recommendation screen):



2. **Initial Quiz:**



3. **Refinement Quiz:**



Future improvements:

1. **Web scraping:** This will enhance the liveliness of the content in the recommendation as the products and brought in from the live internet.
2. **Automated Data Pipeline for New Items:** Need to build a real-time data ingestion pipeline using a task queue. This would allow the system to automatically scrape new fashion items from websites, use computer vision to extract attributes, generate embeddings, and add them to the recommendation inventory without manual intervention.

Interesting tidbits:

1. Secure Profile Management
2. Scalable contents handling approach