Figure 1. University of Malaya Logo

Faculty of Computer Science and Information Technology

University of Malaya

Department of Artificial Intelligence

WIA3002 ACADEMIC PROJECT 1

FINAL YEAR PROJECT REPORT

Fashion Biometric Recommendation (FaBR)

A Fashion-Biometric Recommendation APP Based on Facial Analysis and Body Metrics

Prepared by:

Rafi Daffa Ramadhani

S2155407

Supervised by:

Dr. Lim Chee Kau

ABSTRACT

Fashion decision fatigue affects millions of users who struggle with daily outfit planning despite owning extensive wardrobes. Current fashion recommendation applications rely on complete outfit inputs and GPU-intensive sequential evaluation, resulting in impractical latency and deployment costs. Existing platforms ignore user biometric attributes, treating outfit compatibility as pure style matching independent of individual physical characteristics. This project presents FaBR, a fashion recommendation iOS app based on complementary item retrieval and planned biometric personalization for Asian fashion markets.

The app develops through two core components supported by machine learning models. The first component implements smart discovery and outfit recommendation features powered by a Complementary Item Retrieval model adapted from OutfitTransformer architecture. The model uses Marqo-FashionSigLIP embeddings and two-stage training to achieve 64.26 Fill-in-the-Blank accuracy on Polyvore dataset while operating on CPU-only infrastructure. Vector-based retrieval through the Qdrant database enables sub-second recommendations across 140,000 Asian fashion products scraped from Musinsa and Wconcept platforms. The iOS app built with React Native integrates wardrobe management with camera upload, semantic search supporting text and image queries, and outfit recommendations from partial inputs.

The second component planned for FYP 2 adds biometric compatibility features through a personalized re-ranking model that fuses user attributes with outfit embeddings. Automated annotation processes professional model photos to extract skin tone, body type, gender, and age attributes for supervised training. Additional app features including user authentication, onboarding flows, and social interactions remain under development. This research contributes to a production-ready mobile application combining fashion AI models with practical deployment architecture for cost-effective personalized styling.

ACKNOWLEDGEMENT

I express my gratitude to my supervisor, Dr. Lim Chee Kau, for his guidance, patience, and feedback throughout this project. His technical insight and commitment to academic rigor shaped both the direction and quality of this work.

I acknowledge Xquisite AI for collaborating as the industry partner in this project. Their requirements for CPU-first inference and Asian market localization informed the system design and evaluation criteria. I extend special thanks to Misbah Fahamsyah from Xquisite AI for providing technical advice and practical insights that contributed to the feasibility study and deployment considerations.

I am grateful to the Faculty of Computer Science and Information Technology at University of Malaya for providing a supportive academic environment and the facilities needed to complete this project. I extend my appreciation to Google TPU Research Cloud for providing TPU v3-8 access that enabled model training at scale.

I thank the creators and maintainers of the open-source tools and datasets that supported this research, including the Polyvore dataset, Marqo-FashionSigLIP, Qdrant, PyTorch, Hugging Face Transformers, React Native, FastAPI, and MongoDB.

I thank my peers in the AI programme for their encouragement, discussions, and shared experiences throughout the final year. Finally, I am deeply grateful to my family for their continuous support, patience, and belief in my abilities, which made the completion of this project possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Definition
AdamW	Adam with Weight Decay
AI	Artificial Intelligence
ANN	Approximate Nearest Neighbor
API	Application Programming Interface
AUC	Area Under the Curve
BERT	Bidirectional Encoder Representations from Transformers
CAGR	Compound Annual Growth Rate
CIR	Complementary Item Retrieval
CLIP	Contrastive Language-Image Pre-training
CNN	Convolutional Neural Network
CORS	Cross-Origin Resource Sharing
CP	Compatibility Predictor
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
FaBR	Fashion Biometric Recommendation
FFN	Feed-Forward Network
FITB	Fill-in-the-Blank
FP32	32-bit Floating Point
FYP	Final Year Project
GB	Gigabyte

GCN	Graph Convolutional Network
GPU	Graphics Processing Unit
HBM	High Bandwidth Memory
HNSW	Hierarchical Navigable Small World
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
iOS	iPhone Operating System
JSON	JavaScript Object Notation
L2	L2 norm / Euclidean distance
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
MMR	Maximum Marginal Relevance
MPS	Metal Performance Shaders
NLP	Natural Language Processing
NoSQL	Not Only SQL
OneCycleLR	One Cycle Learning Rate
RAM	Random Access Memory
ResNet	Residual Network
REST	Representational State Transfer
RESTful	Representational State Transfer
ROC	Receiver Operating Characteristic
SigLIP	Sigmoid Loss for Language-Image Pre-training
TPU	Tensor Processing Unit

TRC	TPU Research Cloud
URL	Uniform Resource Locator
USD	United States Dollar
vCPU	virtual Central Processing Unit
ViT	Vision Transformer
WACV	Winter Conference on Applications of Computer Vision
XLA	Accelerated Linear Algebra
YOLOS	You Only Look at One Sequence

Table 1. List of Abbreviations

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

Fashion decision fatigue affects daily outfit planning for many users. A Marks and Spencer survey found that women spend an average of 13-17 minutes each morning deciding what to wear, totaling approximately six months of working life between ages 18 and 60 (Cosmopolitan UK, 2016). The survey also reported that 62% of women experience emotional reactions during outfit selection despite owning 152 clothing items on average (Cosmopolitan UK, 2016). Most people wear only 20% of their wardrobe 80% of the time, showing severe inefficiency in closet utilization (Colorful Socks, 2025).

AI in fashion grows due to demand for personalized recommendations and automated shopping experiences. The global AI in fashion market was valued at USD 2.23 billion in 2024 and is projected to reach USD 60.57 billion by 2034, growing at 39.12% CAGR from 2025 to 2034 (Precedence Research, 2025). Another forecast reports growth from USD 228 million in 2019 to USD 1,260 million by 2024 at 40.8% CAGR (MarketsandMarkets, 2019). This growth reflects increasing demand for personalization, inventory efficiency, and AI tool integration across fashion retail and e-commerce.

Fashion recommendation systems use different approaches for compatibility and retrieval tasks. Transformer-based models process outfits as unordered sets using self-attention to learn item relationships (Sarkar et al., 2023). Recent work uses two-stage training where compatibility prediction models transfer learned representations to complementary item retrieval, achieving better performance than single-stage methods. Multimodal embeddings combining visual and textual features through models like SigLIP improve understanding across diverse fashion catalogs and user preferences.

1.2 PROBLEM STATEMENT

An ideal fashion recommendation system operates across three dimensions: functionality, performance, and personalization. The system should accept partial outfit inputs and generate complementary items ranked by style matching, not requiring complete outfits. The model must be small enough to run on CPU infrastructure for cost-effective deployment. The system should factor user biometrics like face attributes, skin tone, body shape, and gender to match both wardrobe compatibility and individual appearance.

Current fashion applications show three connected problems. First, most apps require complete outfits as input and score candidates one by one using brute-force methods. This causes linear latency increases with wardrobe size. A user with 100 tops and 100 bottoms would need

10,000 sequential evaluations at 3 seconds each, totaling 8.3 hours. Second, inference relies on expensive GPU computation, creating barriers for users with limited infrastructure. Third, existing systems ignore user biometrics, treating all outfits the same regardless of who wears them.

The research gap focuses on building a CPU-first, biometric-aware complementary item retrieval system for Asian fashion preferences and body types. CPU-first architecture removes GPU dependency by using precomputed embeddings and efficient vector search. This enables deployment on standard cloud CPU instances at lower cost. Existing research identifies personalization and efficiency as key challenges, but production systems remain limited to complete-outfit classification or slow iterative search that needs GPU acceleration.

1.3 STAKEHOLDER PROFILE

Xquisite AI, a Jakarta-based IT consultancy, collaborates as the industry validator in this academic-industry partnership. The organization expects retail client demand for smart fitting systems but lacks internal benchmarks for feasibility of localized fashion AI models under practical resource constraints. The collaborator defines two core requirements for success. First, CPU-first inference must prove feasibility of recommendations on standard cloud CPU instances without GPUs. Second, demographic localization must show strong compatibility scoring across Asian fashion styles, body types, and aesthetic preferences.

The project provides the stakeholder with reproducible methodology and validated performance benchmarks. Deliverables include technical documentation, trained model checkpoints, and inference implementation code. These materials work as a feasibility study and commercial blueprint, not production code. This enables Xquisite AI to pitch similar solutions to enterprise clients while understanding performance trade-offs, infrastructure requirements, and training methodology.

1.4 OBJECTIVES

Objective 1 develops a generative complementary item retrieval model by adapting OutfitTransformer architecture for the fill-in-the-blank task. The model accepts partial outfit inputs and generates ideal item embeddings through a target token mechanism, enabling vector-based retrieval of candidate matches from user wardrobes. The implementation uses unified Marqo-FashionSigLIP embeddings instead of dual ResNet-BERT streams.

Objective 2 develops a biometric compatibility prediction model that fuses user attribute tokens (skin tone, body shape, gender, age) with visual outfit tokens to score personalized suitability. The model processes vectors from Objective 1 and re-ranks candidates based on

individual characteristics. The model treats professional model photos as positive ground truth and random combinations as negative examples.

Objective 3 builds the iOS application stack integrating React Native frontend with Qdrant vector database, MongoDB, and FastAPI backend. Features include smart discovery with text and image search, wardrobe management with camera import, outfit recommendation combining Model 1 retrieval with Model 2 biometric re-ranking, and user authentication.

1.5 SCOPE AND LIMITATIONS

The project focuses on developing an iOS application using React Native for iPhone devices running iOS 15 or later. Target users are Gen Z and millennials aged 18 to 35 in Asia. The fashion catalog draws from Asian e-commerce platforms Musinsa and Wconcept, totaling more than 140,000 indexed products. Outfit categories cover four types: outerwear, tops, bottoms, and shoes. The system excludes accessories, handbags, and multi-layer outfit combinations.

Core features include wardrobe management, smart discovery with semantic search, and outfit recommendation using two-stage ranking. Social features like posts, follows, and bookmarks are planned for later phases. The model training uses Polyvore dataset for compatibility learning and fill-in-the-blank testing. For Model 2, training data comes from professional model photos scraped from Musinsa and Wconcept, spanning more than 100,000 fashion images.

The compatibility model trains and evaluates only on the non-disjoint Polyvore split, where training and test outfits do not overlap but individual items may appear in both sets. The application remains at 20% completion at FYP 1. Model 2 for biometric compatibility is in the data preparation phase. User authentication, onboarding flows, and social features remain incomplete at this stage. Both application development and Model 2 training will continue in FYP 2, with planned integration of biometric re-ranking, social features, and complete user onboarding flows.

The application remains at 20% completion at FYP 1. Model 2 for biometric compatibility is in the data preparation phase. User authentication, onboarding flows, and social features remain incomplete at this stage. Both application development and Model 2 training will continue in FYP 2, with planned integration of biometric re-ranking, social features, and complete user onboarding flows. Performance targets, methodological choices, and achieved results appear in Chapter 5.

1.6 PROJECT SCHEDULE

The project spans two academic semesters divided into FYP 1 and FYP 2, following the Waterfall development model with four sequential phases.

FYP 1 (5 October 2025 to 11 January 2026)

October 2025 to November 2025: Phase 1 - Data Preparation and Preprocessing

- Polyvore dataset acquisition and embedding generation
- Musinsa and Wconcept catalog scraping (140,000 products)
- Marqo-FashionSigLIP embedding computation for all items
- Qdrant vector database setup and product indexing

November 2025 to December 2025: Phase 2 - Model 1 & 2 Development

- Compatibility Predictor training (200 epochs on TPU v4)
- Complementary Item Retrieval training with transfer learning
- Hyperparameter exploration across five CP configurations
- Model evaluation achieving 64.26% FITB accuracy
- Automated annotation pipeline for 450,000 model photos (for model 2)
- User attribute extraction (skin tone, body type, gender, age) (for model 2)

December 2025 to January 2026: Phase 3 - Backend Infrastructure

- FastAPI server implementation with RESTful endpoints
- Hybrid search logic development (text-image weighting)
- Smart Discovery implementation with semantic search
- Outfit recommendation system integration with Model 1
- Server deployment on Contabo (6 vCPU, 12GB RAM)

December 2025 to January 2026: Phase 4 - iOS Application Development (20% Complete)

- React Native project initialization
- Wardrobe management interface implementation
- Smart Discovery screens with text search functionality
- Outfit recommendation workflow integration

FYP 2 (9 March 2026 to 21 June 2026)

March 2026 to April 2026: Model 2 Development - Biometric Personalization

- Biometric compatibility model training with Focal Loss
- Model 2 integration with outfit re-ranking pipeline

April 2026 to May 2026: Application Completion

- User authentication and authorization system
- Onboarding flow with biometric capture and wardrobe setup
- Image-based smart discovery implementation
- Camera upload with automatic embedding generation
- Light and dark mode theme support

May 2026 to June 2026: Social Features and Deployment

- Post creation and sharing functionality
- Follow system and social feed implementation
- Like, save, and bookmark features
- Profile viewing and engagement metrics
- MongoDB integration for user data and social interactions
- Production deployment and testing
- Final documentation and project submission

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

Fashion recommendation systems have evolved from simple collaborative filtering approaches to sophisticated deep learning architectures that model complex compatibility relationships between garments (Montalbo, 2021). Recent advances leverage transformer architectures and multimodal embeddings to process fashion items as unordered sets rather than sequential inputs, achieving state-of-the-art performance on compatibility prediction and complementary item retrieval tasks (Sarkar et al., 2023). This literature review examines existing fashion recommendation systems, analyzes the foundational OutfitTransformer architecture, evaluates competing commercial applications, and identifies research gaps that justify the development of a CPU-first, biometric-aware recommendation system for Asian fashion markets.

2.2 FASHION RECOMMENDATION SYSTEMS OVERVIEW

Fashion recommendation systems employ three primary approaches: content-based filtering, collaborative filtering, and hybrid methods that combine both paradigms (Montalbo, 2021). Content-based systems analyze visual and textual attributes of fashion items to suggest stylistically similar products, while collaborative filtering leverages user interaction patterns to identify preferences across similar user profiles (Montalbo, 2021). Hybrid approaches address data sparsity and cold-start problems inherent in fashion e-commerce by merging these complementary techniques (Montalbo, 2021).

Deep learning revolutionized fashion recommendation by enabling automatic feature extraction from images through convolutional neural networks, eliminating the need for manual feature engineering (Montalbo, 2021). Common evaluation metrics include Recall@K, with values ranging from 3-8% per category in benchmark studies, and Fill-in-the-Blank (FITB) accuracy, where state-of-the-art models achieve 63.73% on standardized datasets (Shirkhani et al., 2023). Modern systems face persistent challenges including sparse purchase data, high product turnover rates, difficulty establishing visual compatibility across diverse catalogs, and the need for real-time personalization at scale (Deldjoo et al., 2024).

The transition from pairwise compatibility scoring to outfit-level representations marks a paradigm shift in the field (Sarkar et al., 2023). Earlier approaches evaluated compatibility between item pairs independently, requiring separate models for each category combination and failing to capture higher-order relationships (Lin et al., 2020). As illustrated in Figure 2, traditional pairwise methods compute compatibility scores between the target item and each outfit component separately, while set-wise approaches generate a unified representation that captures the collective relationship among all items (Sarkar et al., 2023). Transformer

architectures address this limitation by processing entire outfits simultaneously through self-attention mechanisms that model dependencies beyond pairwise interactions (Sarkar et al., 2023). This set-wise processing enables models to understand how multiple items collectively create cohesive outfits rather than simply matching pairs in isolation (Sarkar et al., 2023).

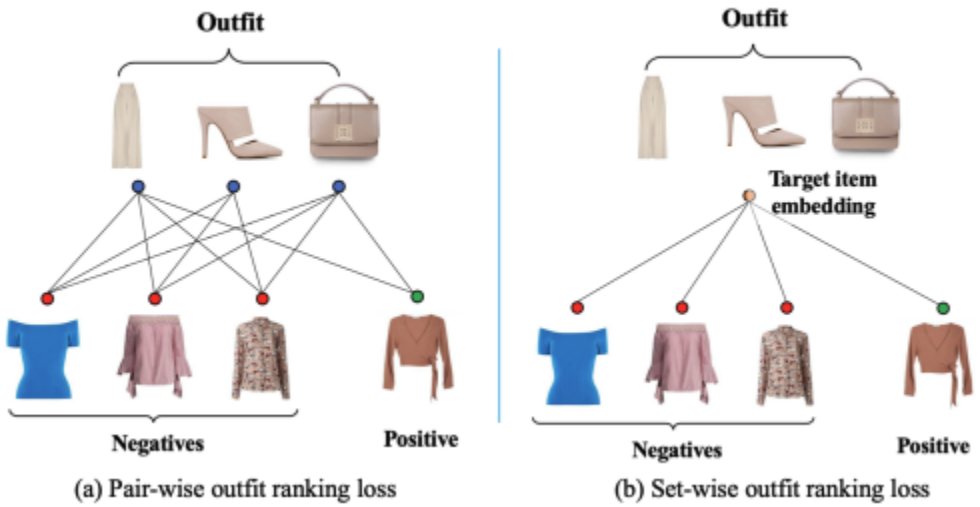


Figure 2. Comparison of (a) pairwise outfit ranking loss and (b) set-wise outfit ranking loss in fashion recommendation systems. Traditional pairwise approaches evaluate each item independently, while set-wise methods process the entire outfit context simultaneously. Source: Sarkar et al. (2023, Figure 3).

2.3 RELATED WORKS

Kotouza et al. (2020) proposed an AI system acting as a personal assistant for fashion product designers, focusing on data crawling, synthesis, and clustering from e-commerce platforms (Kotouza et al., 2020). Their architecture supports designers with synthesized insights through effective clustering methods including K-modes and hierarchical agglomerative clustering applied to mixed-type data containing both categorical and numerical attributes (Kotouza et al., 2020). The system demonstrates strong performance in offline data retrieval and analysis for design inspiration, processing garment datasets from multiple sources and transforming them into unified formats using natural language processing techniques (Kotouza et al., 2020). However, the approach lacks real-time inference capabilities for consumer-facing applications, omits biometric integration for personalized recommendations, and does not address mobile deployment efficiency constraints required for end-user adoption (Kotouza et al., 2020).

Sami et al. (2025) presented an AI-powered body type analysis system combining machine learning for anthropometric measurements with personalized clothing recommendations

tailored to individual body shapes (Sami et al., 2025). The system achieved greater than 94% accuracy in body type detection using image analysis algorithms and provides tailored recommendations that reduce style mismatches between clothing designs and user body proportions (Sami et al., 2025). The approach demonstrates the feasibility of biometric-aware fashion systems and validates user interest in personalized recommendations based on physical attributes (Sami et al., 2025). However, the system does not integrate with complementary item retrieval frameworks for outfit completion, remains in early-stage validation with limited evaluation on mobile performance and scalability, and does not address partial outfit completion scenarios or real-time inference requirements essential for practical deployment (Sami et al., 2025).

2.4 FOUNDATION: OutfitTransformer (Sarkar et al., WACV 2023)

The OutfitTransformer architecture introduced by Sarkar et al. (2023) at the Winter Conference on Applications of Computer Vision establishes the foundation for this project through its novel approach to outfit-level representation learning (Sarkar et al., 2023). The model processes outfits as unordered sets using transformer self-attention mechanisms, contrasting with previous sequential approaches based on Bi-LSTM or graph convolutional networks that imposed artificial ordering on fashion items (Sarkar et al., 2023). This set-wise processing captures higher-order compatibility relationships that extend beyond pairwise item interactions, enabling the model to understand how three or more items collectively create stylistic harmony (Sarkar et al., 2023).

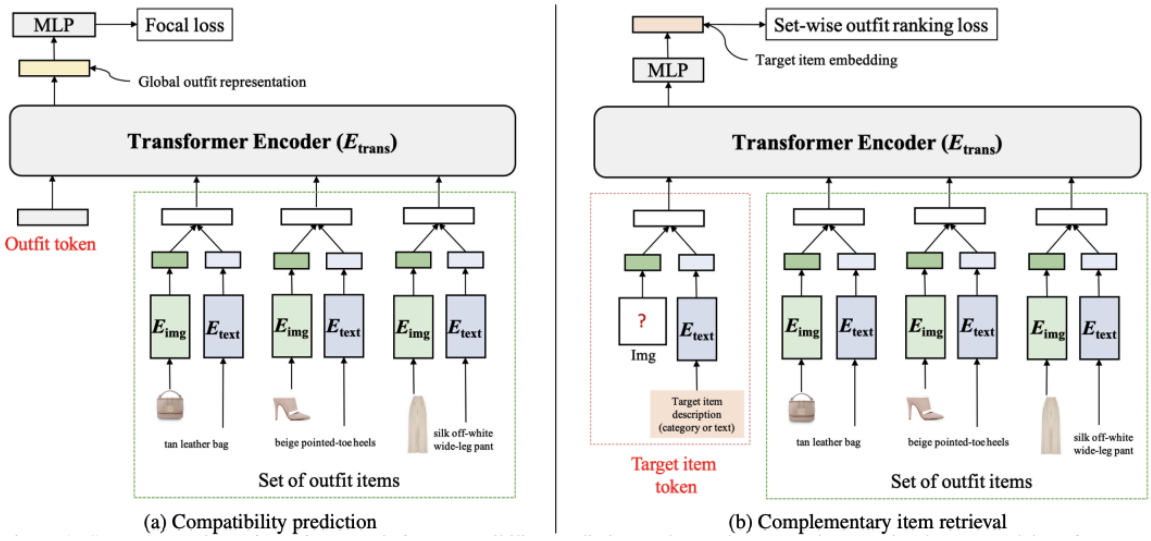


Figure 3. System overview of OutfitTransformer framework

The architecture features two specialized components trained in a two-stage curriculum learning approach (Sarkar et al., 2023). The Compatibility Predictor (CP) learns outfit-level

representations through a global outfit token that aggregates information from all items via cross-attention, producing binary compatibility scores for complete outfits (Sarkar et al., 2023). The Complementary Item Retrieval (CIR) model builds upon the CP foundation by introducing a target token mechanism that generates ideal item embeddings for missing outfit components based on partial input and target specifications (Sarkar et al., 2023). This design enables vector-based retrieval where the generated embedding is compared against candidate items in embedding space rather than requiring exhaustive pairwise comparisons (Sarkar et al., 2023).

The model employs dual-stream feature extraction using ResNet-18 for image encoding and SentenceBERT for text descriptions, concatenating both modalities to create unified item representations that capture visual style and semantic meaning (Sarkar et al., 2023). Training follows a transfer learning strategy where the Compatibility Predictor learns outfit harmony principles first through focal loss optimization, then transfers learned weights to initialize the Complementary Item Retrieval model trained with a set-wise outfit ranking loss (Sarkar et al., 2023). This two-stage approach significantly outperforms single-stage training by providing the CIR model with pre-learned compatibility knowledge (Sarkar et al., 2023).

Performance benchmarks demonstrate state-of-the-art results on the Polyvore dataset. OutfitTransformer achieved 67.10% Fill-in-the-Blank accuracy on the non-disjoint split where training items can appear in test outfits, and 59.48% accuracy on the disjoint split where training and testing items have no overlap (Sarkar et al., 2023). The model surpasses graph-based approaches including Graph Convolutional Networks and Type-Aware embeddings, as well as LSTM architectures, on both compatibility prediction and retrieval tasks measured by AUC and Recall@K metrics respectively (Sarkar et al., 2023). The framework processes fashion items with category specifications such as "find me shoes" or detailed text descriptions like "blue denim jeans," enabling flexible user interaction patterns (Sarkar et al., 2023).

Despite these achievements, As illustrated in Figure 3, the architecture reveals three critical gaps for practical deployment. First, **the lack of personalization** mechanisms means the model predicts whether items match each other stylistically but ignores whether they suit specific users' biometric attributes including skin tone, body shape, facial features, or demographic characteristics. Second, **the model requires explicit target item specifications as input,** meaning users must specify the category or description of the desired item rather than the system independently determining what item would complete the outfit (Sarkar et al., 2023). Third, **the reliance on separate ResNet-BERT encoders** executed during each inference increases computational overhead, creating deployment barriers for cost-sensitive applications (Sarkar et al., 2023).

2.5 EXISTING PRODUCT ANALYSIS

Fashion recommendation applications including Gensmo, ACloset, Whering, and Combyne share common architectural limitations that create opportunities for innovation. These platforms primarily implement complete outfit scoring methods where random outfit combinations are generated iteratively, scored individually through compatibility models, and ranked by their scores to present top recommendations. This brute-force approach produces linear latency increases with wardrobe size, resulting in impractical wait times for users with extensive collections. A user with 100 tops and 100 bottoms would require 10,000 sequential evaluations, totaling over 8 hours at 3 seconds per evaluation, making real-time recommendation infeasible.

Systems supporting partial outfit input typically employ sequential evaluation where the model assesses compatibility between the input item and each wardrobe candidate one-by-one, aggregating scores to produce final rankings . For example, matching a single shirt against 20 pants options at 3 seconds per evaluation requires 60 seconds total processing time, still exceeding acceptable response latency for interactive applications. This sequential architecture prevents real-time recommendation delivery and necessitates expensive GPU infrastructure to maintain acceptable response latency through parallel processing. The computational bottleneck stems from executing deep neural network encoders for every candidate item during each query.

No surveyed fashion recommendation application incorporates user biometric attributes such as skin tone, body shape, gender, or facial features into recommendation algorithms. All platforms treat outfit compatibility as pure style-matching problems independent of who wears the garments, producing identical recommendations for users with vastly different physical characteristics and aesthetic suitability. This represents a significant personalization gap where the same outfit combination receives equal scores regardless of whether the colors complement a user's skin tone, the silhouette flatters their body shape, or the style matches their age and gender presentation. This limitation persists despite user research showing that biometric compatibility significantly influences purchase decisions and outfit satisfaction.

2.6 TECHNICAL FOUNDATIONS

2.6.1 TRANSFORMER ARCHITECTURE

Transformer models introduced by Vaswani et al. (2017) employ multi-head self-attention mechanisms to process sequential or set-based inputs without recurrent connections, enabling parallel computation and improved modeling of long-range dependencies (Vaswani et al., 2017). The self-attention operation computes pairwise relationships between all input elements through query-key-value projections, where attention weights represent the relevance of each element to

every other element in the sequence (Vaswani et al., 2017). In fashion applications, transformers model outfit compatibility by treating garments as tokens where attention weights represent stylistic relationships between items, capturing how color palettes, patterns, and silhouettes interact across the outfit (Sarkar et al., 2023).

Layer normalization, residual connections, and position-wise feed-forward networks complement the attention mechanism to create deep feature hierarchies that progressively refine representations (Vaswani et al., 2017). Vision transformers adapt this architecture for image processing by splitting images into fixed-size patches, linearly embedding them, and treating patch embeddings as sequence elements with learnable position encodings (Dosovitskiy et al., 2020). Recent work demonstrates that transformer-based vision models can achieve faster CPU inference than traditional convolutional networks through architectural modifications including attention downsampling, hybrid convolutional-transformer designs, and efficient attention implementations (Graham et al., 2021). The LeViT architecture specifically optimizes transformers for CPU deployment, achieving 5 times faster inference than EfficientNet at 80% ImageNet accuracy on single CPU cores through carefully designed bottleneck structures and attention patterns (Graham et al., 2021).

2.6.2 CONTRASTIVE LEARNING AND METRIC LEARNING

Contrastive learning trains neural networks to map similar items closer together and dissimilar items farther apart in embedding space through distance-based objectives (Chen et al., 2020). Metric learning objectives enable models to learn discriminative feature representations that generalize across tasks with limited labeled examples. Triplet loss, a foundational metric learning objective, operates on triplets of anchor, positive, and negative samples with the goal of ensuring the distance between anchor and positive is smaller than the anchor-negative distance by a defined margin α (Schroff et al., 2015). The loss function $L = \max(d(a,p) - d(a,n) + \alpha, 0)$ penalizes configurations where negatives are closer than positives plus the margin (Schroff et al., 2015).

In-batch negative sampling improves training efficiency by treating other samples within each minibatch as negative examples, eliminating the need for explicit hard negative mining that requires expensive distance computations across the entire dataset (Sohn, 2016). This approach reduces memory requirements from $O(N^2)$ to $O(B^2)$ where N is dataset size and B is batch size, while accelerating training convergence by providing diverse negatives at each iteration (Sohn, 2016). Fashion retrieval systems leverage triplet loss variants to learn embeddings where compatible items cluster together regardless of visual dissimilarity, enabling cross-category matching such as tops-to-bottoms retrieval where visual features differ substantially but stylistic compatibility remains high (Lin et al., 2020). The InBatchTripletMarginLoss variant used in

recent fashion models selects hard negatives from each batch dynamically, focusing learning on challenging examples that lie near decision boundaries (Sarkar et al., 2023).

2.6.3 MULTIMODAL EMBEDDINGS FOR FASHION-SPECIFIC CATEGORY

Contrastive Language-Image Pre-training (CLIP) models established the foundation for joint text-image embeddings by matching corresponding descriptions and images through contrastive objectives on large-scale web-scraped datasets (Radford et al., 2021). The training process maximizes cosine similarity between matched text-image pairs while minimizing similarity between mismatched pairs, creating a shared embedding space where semantically related concepts cluster together regardless of modality (Radford et al., 2021). SigLIP extends this approach using sigmoid loss functions that improve training stability and retrieval performance by replacing softmax-normalized contrastive loss with binary classification objectives for each pair (Zhai et al., 2023).

Fashion-specific adaptations of vision-language models address the limitations of general-purpose models for domain-specific retrieval tasks. Marqo-FashionSigLIP fine-tunes SigLIP on fashion-domain data, achieving up to 57% improvement in mean reciprocal rank and recall over FashionCLIP 2.0 on fashion retrieval benchmarks (Marqo, 2024). The model leverages Generalized Contrastive Learning to optimize over seven fashion-specific aspects including descriptions, categories, styles, colors, materials, keywords, and fine details, creating embeddings that capture nuanced fashion attributes beyond generic visual features (Marqo, 2024).

Built on ViT-B-16-SigLIP architecture, the model generates 768-dimensional embeddings for both images and text. Following the OutfitTransformer methodology (Sarkar et al., 2023), these embeddings are concatenated to create unified 1536-dimensional item representations that combine visual appearance and semantic descriptions. This multimodal architecture enables semantic search where text queries like "vintage denim jacket" retrieve visually similar products, and supports hybrid search combining text and image inputs for precise style matching.

2.6.4 Loss Functions for Imbalanced Classification

Focal Loss addresses class imbalance in deep learning by down-weighting the loss contribution from well-classified examples and focusing training on hard, misclassified samples through a modulating factor (Lin et al., 2017). The loss function incorporates $(1-p_t)^{\gamma}$ where p_t represents the model's estimated probability for the true class and γ controls the focusing strength, with higher γ values more aggressively reducing loss for easy examples (Lin et al., 2017). When $\gamma=0$, Focal Loss reduces to standard cross-entropy, while $\gamma=2$ in typical applications provides strong focusing on hard examples (Lin et al., 2017). The alpha parameter provides class

weighting to further address imbalance by assigning higher importance to minority classes through α for class 1 and 1-α for class 0 (Lin et al., 2017).

Empirical studies demonstrate that Focal Loss with α=0.5 and γ=2 significantly outperforms standard cross-entropy on imbalanced datasets, achieving higher F1-scores and AUC values by mitigating bias toward majority classes (Lin et al., 2017). In binary compatibility prediction for fashion, where positive outfit examples representing compatible combinations may be less frequent than negative combinations from random sampling, Focal Loss ensures the model learns discriminative features for both compatible and incompatible outfit configurations without overwhelming the gradient signal with easy negatives (Lin et al., 2017). The focusing mechanism prevents the model from becoming overconfident on simple examples and forces attention toward boundary cases that determine decision quality (Lin et al., 2017).

2.6.5 Learning Rate Scheduling with OneCycleLR

OneCycleLR implements the 1cycle learning rate policy that anneals the learning rate from an initial value to a maximum, then decreases to a minimum value much lower than the starting point over a single training cycle (Smith, 2018). This policy enables faster convergence and improved generalization by allowing the model to escape shallow local minima during the high learning rate phase and refine solutions during the low learning rate phase where gradient noise decreases (Smith, 2018). The scheduler operates on a per-batch basis rather than per-epoch, with the total cycle spanning all training iterations typically configured as num_epochs × iterations_per_epoch (Smith, 2018).

Key hyperparameters include max_lr defining the peak learning rate typically found through learning rate range tests, pct_start controlling the percentage of the cycle spent increasing the learning rate (typically 30% for warm-up), and div_factor determining the initial learning rate via initial_lr = max_lr/div_factor (Smith, 2018). The anneal_strategy parameter selects between cosine and linear annealing for the decreasing phase, with cosine providing smoother transitions (Smith, 2018). Studies show that OneCycleLR reduces training time by 2-4x compared to traditional learning rate decay schedules while achieving comparable or superior final accuracy through more efficient exploration of the loss landscape (Smith, 2018). The method works by cyclically varying learning rate and momentum in opposite directions, enabling larger learning rates than traditional approaches while maintaining training stability (Smith, 2018).

2.6.6 Vector Databases and Efficient Retrieval

Vector databases specialize in storing high-dimensional embeddings and performing fast approximate nearest neighbor search through indexing structures like HNSW (Hierarchical Navigable Small World) graphs that reduce search complexity from O(N) to O(log N) (Malkov

& Yashunin, 2018). Qdrant implements an open-source vector database optimized for production deployments with sub-second retrieval latency for collections exceeding 100,000 vectors through efficient graph-based indexing and payload filtering capabilities. The system supports payload filtering enabling hybrid queries that combine vector similarity with metadata constraints such as category, price range, or brand, reducing search space without sacrificing result relevance.

Quantization techniques compress vector representations to reduce memory footprint and accelerate similarity computations, with scalar quantization mapping float32 values to int8 representations while preserving search accuracy above 95% in most scenarios through careful quantile-based binning. Maximum Marginal Relevance (MMR) reranking balances similarity with diversity by penalizing results similar to already-selected items, preventing fashion search results from showing only minor variations of the same style and ensuring diverse recommendations. For fashion applications, vector databases enable precomputed embedding storage where heavy encoder models process items offline once, storing resulting embeddings for online retrieval. This architecture makes CPU inference viable by avoiding runtime encoding overhead and shifting computational cost from query time to indexing time.

CHAPTER 3: METHODOLOGY

3.1 INTRODUCTION

Fashion recommendation systems require methodological frameworks that balance model accuracy with computational feasibility for real-world deployment. Academic benchmarks often prioritize performance metrics while assuming unlimited computational resources, creating a gap between research achievements and production constraints. This project addresses the challenge of adapting transformer-based fashion models for CPU-first inference while maintaining competitive accuracy on standard retrieval tasks. The methodology must accommodate stakeholder requirements for deployment on standard cloud infrastructure without GPU dependencies while serving fashion catalogs exceeding 140,000 products.

The development approach follows systematic phases spanning data preparation, model training, backend infrastructure, and application integration. Model development builds on the OutfitTransformer architecture from Sarkar et al. (2023) through architectural modifications that prioritize inference efficiency. Training strategies employ transfer learning principles where compatibility knowledge transfers from a binary classification task to a generative retrieval task. Evaluation methodology measures both model performance through Fill-in-the-Blank accuracy and system performance through latency benchmarks under CPU-only conditions.

3.2 DEVELOPMENT MODEL

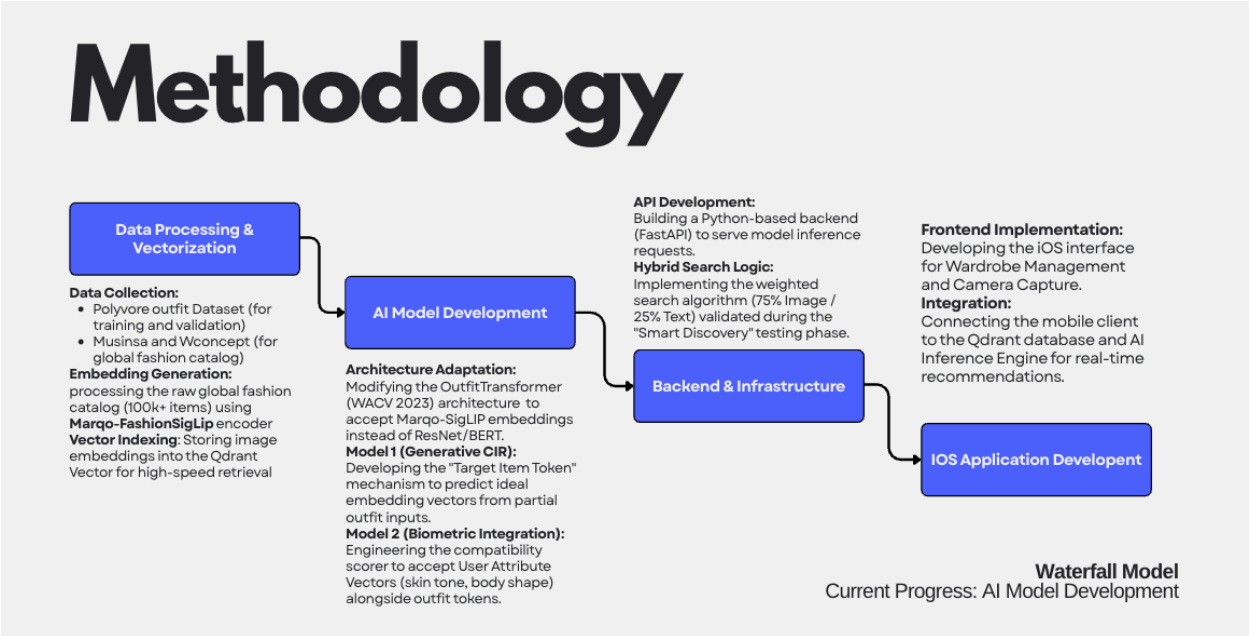


Figure 4. Waterfall Model

The project adopts the Waterfall model as the primary development framework due to its sequential structure and clear milestone definition suited for academic timelines (Figure 4). Waterfall provides distinct phase boundaries that align with FYP 1 (5/10/2025 - 11/01/2026) and FYP 2 (9/03/2026 to 21/06/2026) deliverable requirements. Each phase produces specific outputs required as inputs for subsequent stages, making the sequential approach practical for this research-focused implementation.

The methodology divides development into four sequential phases spanning data preparation through application deployment.

1. Phase 1 covers data processing and vectorization, including Polyvore dataset preparation, Musinsa and Wconcept catalog scraping, embedding generation through Marqo-FashionSigLIP encoder, and vector indexing in Qdrant database.
2. Phase 2 encompasses AI model development, adapting OutfitTransformer architecture for precomputed embeddings, training the Compatibility Predictor model, and fine-tuning the Complementary Item Retrieval model through transfer learning.
3. Phase 3 establishes backend infrastructure through FastAPI implementation, hybrid search logic development combining 75% image and 25% text weighting, and database service configuration, and outfits recommendation system.
4. Phase 4 completes iOS application development through React Native frontend implementation and integration with backend services for real-time recommendations.

Model training cannot start until embedding generation completes, as the architecture processes precomputed 1536-dimensional vectors rather than raw images and text. API development requires trained model checkpoints to serve inference requests, establishing model completion as a prerequisite for backend integration. Application features depend on functional backend services for database queries and recommendation generation. The methodology model allows practical overlap where application development can begin during model training to test integration and identify interface needs before full deployment.

3.3 DATA PREPARATION

Polyvore Outfits dataset serves as the primary training source for Model 1 development following the OutfitTransformer methodology. The dataset provides two split configurations for different training objectives. The non-disjoint compatibility split contains 107,000 training outfits, 10,000 validation outfits, and 20,000 test outfits for Compatibility Predictor training. The non-disjoint fill-in-the-blank split contains 53,300 training outfits, 5,000 validation outfits, and 10,000 test outfits for Complementary Item Retrieval evaluation. In non-disjoint splits, individual items can appear across training and test sets but complete outfit combinations do not

overlap, allowing the model to learn compatibility relationships from diverse item combinations while testing generalization on unseen outfit configurations. Each outfit contains multi-modal information including product images, text descriptions, and category labels spanning outerwear, tops, bottoms, and shoes.

Musinsa and Wconcept platforms supply the production fashion catalog and Model 2 training data due to their focus on Asian fashion styles matching stakeholder localization requirements. Both platforms offer extensive product collections spanning outerwear, tops, bottoms, and shoes categories with professional model photography. The catalog serves three purposes in system deployment: product embeddings enable semantic search functionality where users query items through text descriptions or visual similarity, catalog items populate the recommendation candidate pool where Model 1 retrieves complementary items matching user wardrobe inputs, and professional model photos provide training data for Model 2 biometric compatibility learning.

3.4 MODEL 1 DEVELOPMENT STRATEGY

3.4.1 BACKBONE SELECTION RATIONALE

The approach replaces the dual-stream ResNet-BERT architecture from OutfitTransformer with Marqo-FashionSigLIP as the unified embedding backbone. OutfitTransformer employs separate ResNet-18 for image encoding and SentenceBERT for text descriptions, requiring two distinct models during inference. Marqo-FashionSigLIP provides a single multimodal encoder that processes both images and text through unified preprocessing.

Marqo-FashionSigLIP offers domain-specific advantages through fine-tuning on fashion data using Generalized Contrastive Learning. The model trains on seven fashion-specific aspects including descriptions, categories, styles, colors, materials, keywords, and fine details. Benchmark evaluations show up to 57% improvement in mean reciprocal rank and recall over FashionCLIP 2.0 across six public fashion datasets.

The architecture adopts a precomputed embedding strategy where all fashion item embeddings are generated offline during the data preparation phase. The Marqo-FashionSigLIP backbone remains frozen during training, with only the transformer layers receiving gradient updates for compatibility and retrieval tasks. This approach shifts computational cost from query time to indexing time, enabling CPU-only inference without executing the embedding model during recommendation requests.

3.4.2 TWO-STAGE TRAINING STRATEGY

Model 1 development follows a two-stage curriculum learning approach where the Compatibility Predictor trains first, then transfers learned weights to initialize the Complementary Item Retrieval model. This sequential training strategy addresses the challenge that compatibility understanding forms the foundation for effective retrieval. A model must learn which items harmonize stylistically before generating embeddings representing ideal complementary items.

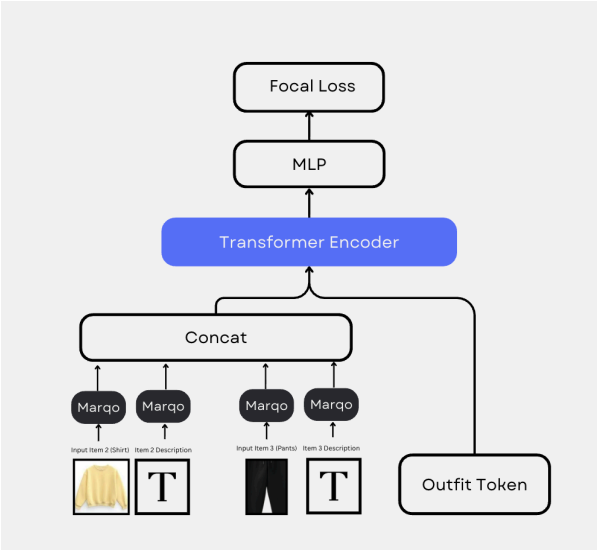


Figure 5. Compatibility Predictor (CP) model architecture overview

Stage 1 trains the Compatibility Predictor as a binary classification task determining whether complete outfits are compatible. The model processes precomputed item embeddings through a transformer encoder that learns outfit-level representations via self-attention across all garment tokens. An outfit token aggregates information from individual items to produce a global compatibility score through an MLP head optimized with Focal Loss.

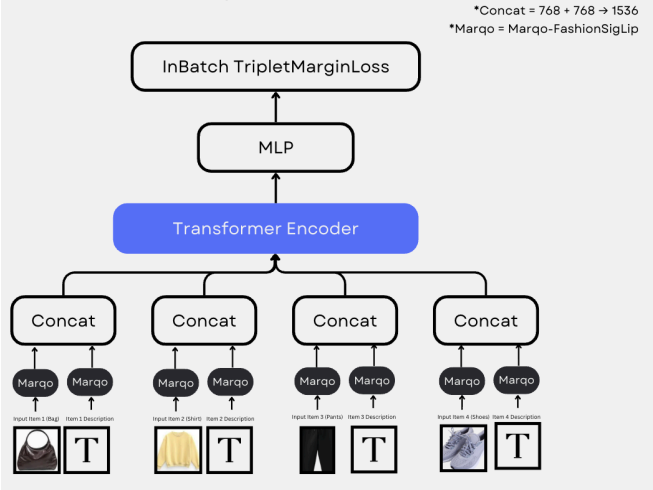


Figure 6. Complementary Item Retrieval (CIR) model architecture overview

Stage 2 transfers the trained CP weights to initialize the Complementary Item Retrieval model for the generative retrieval task. The CIR architecture replaces the outfit token with a target mechanism that predicts ideal item embeddings for missing outfit components based on partial inputs. Training optimizes InBatchTripletMarginLoss to ensure predicted embeddings lie close to ground truth items in vector space while separating from incompatible alternatives. Full architectural specifications appear in Section 4.4.

3.4.4 LOSS FUNCTION SELECTION

Focal Loss trains the Compatibility Predictor to address class imbalance in compatibility data, while InBatchTripletMarginLoss trains the Complementary Item Retrieval model through efficient negative sampling that reduces computational overhead compared to the set-wise outfit ranking loss used in OutfitTransformer (Sarkar et al., 2023).

Focal Loss optimizes CP training by down-weighting loss contributions from well-classified examples and focusing learning on hard, misclassified samples (Lin et al., 2017). The Polyvore compatibility dataset exhibits class imbalance where positive outfit examples (compatible combinations) appear less frequently than negative examples generated through random item sampling. Standard cross-entropy loss biases the model toward the majority class, producing high accuracy through predicting negative labels without learning discriminative compatibility features. Focal Loss incorporates the modulating factor $(1\text{-}p_t)^\gamma$ where p_t represents the model's estimated probability for the true class and $\gamma=2$ controls focusing strength (Lin et al., 2017). This mechanism prevents the model from becoming overconfident on simple negative examples and forces attention toward boundary cases where outfit compatibility remains ambiguous. The alpha parameter ($\alpha=0.5$) provides additional class weighting to balance positive and negative contributions during gradient updates.

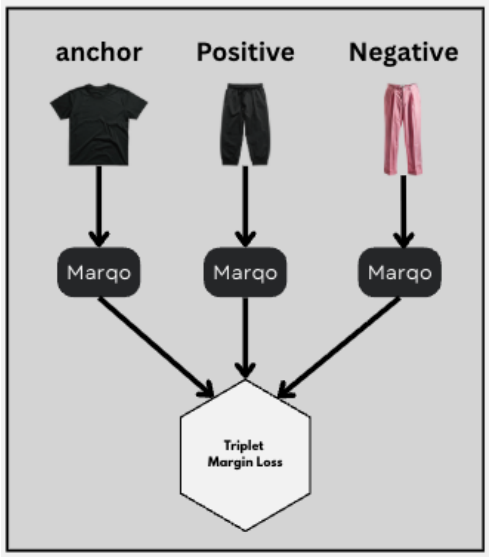


Figure 7. Triplet Margin Loss training mechanism

InBatchTripletMarginLoss optimizes CIR training through efficient negative sampling that treats other items within each training batch as negative examples (Figure 7). Traditional triplet loss requires explicit hard negative mining through expensive distance computations across the entire dataset to identify challenging examples near decision boundaries. InBatchTripletMarginLoss eliminates this preprocessing step by leveraging batch diversity, where the model pulls predicted embeddings closer to ground truth positive items while pushing them away from all other batch items serving as implicit negatives.

The reason why this project adopts InBatchTripletMarginLoss instead of OutfitTransformer's set-wise outfit ranking loss is for practical efficiency under resource constraints. Set-wise ranking loss optimizes outfit-level preferences by comparing complete outfit combinations, requiring additional preprocessing steps and computational overhead to construct ranking pairs. InBatchTripletMarginLoss simplifies the training pipeline by converting negative sampling into fast matrix operations within each batch, avoiding the IO bottleneck from loading specific negative examples from disk. This design choice prioritizes implementation efficiency and training speed under limited compute and schedule constraints while maintaining competitive accuracy.

3.5 MODEL 2 DEVELOPMENT STRATEGY

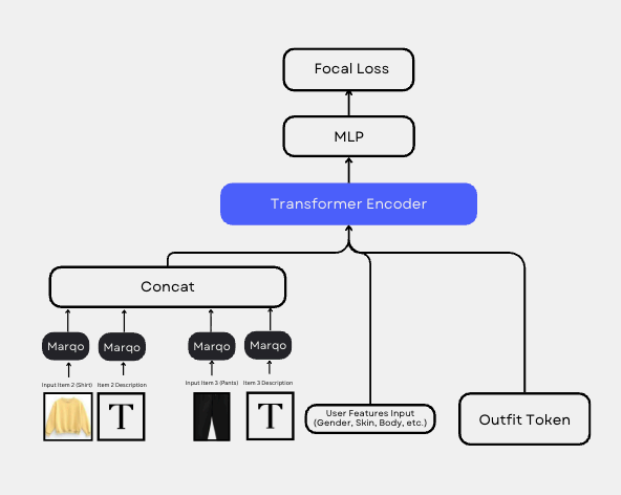


Figure 8. CP Model 2 architecture overview

Model 2 addresses the personalization limitation identified in Section 2.4 by developing a biometric compatibility prediction model that scores outfit suitability based on user-specific attributes (Figure 8). The model operates as a re-ranking module that receives candidate items from Model 1 retrieval and assigns personalized compatibility scores incorporating user attributes including skin tone, body type, gender, and age. This two-model pipeline separates style compatibility (Model 1) from biometric suitability (Model 2), enabling independent optimization of each task. The architecture extends the Compatibility Predictor approach from Model 1 by fusing visual outfit embeddings with categorical user attribute embeddings through a shared transformer encoder. Model 2 development will be completed during FYP 2 (9 March 2026 to 21 June 2026), with FYP 1 focusing on data preparation and pipeline validation.

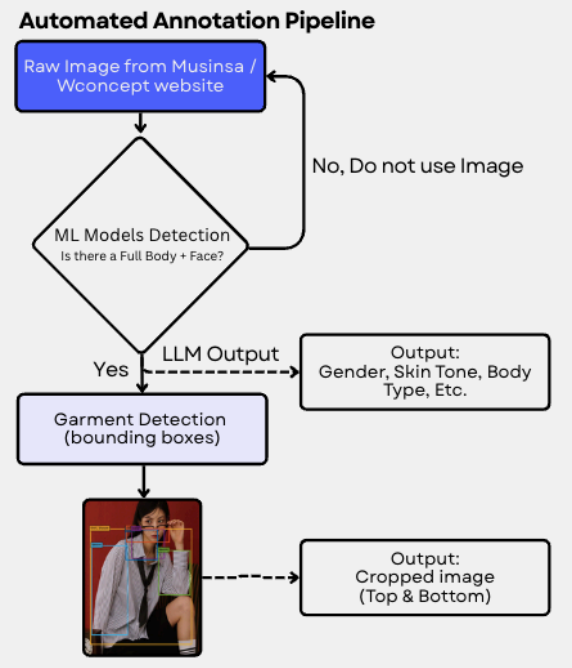


Figure 9. Automated Annotation Pipeline for Model 2

The primary challenge in Model 2 development is from the absence of standardized biometric compatibility datasets. Public fashion datasets such as Polyvore contain outfit compatibility labels but lack user attribute annotations, making supervised training infeasible without constructing ground truth data. The project addresses this limitation through automated ground truth generation using professional model photos from the Musinsa and Wconcept catalog (Figure 9). A two-stage ML pipeline filters the 450,000 scraped product images to identify photos containing visible full body, face, and clothing combinations, yielding approximately 40% of images as valid training candidates. An LLM-based annotation system extracts seven user attributes (gender, skin tone, body type, facial features, age range and build from filtered images, while an object detection model crops individual garment bounding boxes for tops and bottoms. The ground truth assumption treats professionally styled combinations of user attributes and worn garments as positive compatibility examples, with random attribute-outfit pairings serving as negative examples during training.

The Model 2 architecture implements dual-stream processing where visual and attribute information merge through a transformer encoder (Figure 8). Visual garment embeddings pass through Marqo-FashionSigLIP backbone and project to match the attribute embedding space. Categorical user features encode through learnable embedding layers to produce attribute vectors. The transformer processes concatenated visual and attribute tokens through multi-head self-attention, with an outfit token aggregating compatibility information across all inputs. An MLP prediction head outputs binary compatibility scores optimized through Focal Loss to address class imbalance between positive professional examples and negative random

combinations. Due to the noisy nature of automated labeling, this model serves as an experimental feature and targets AUC exceeding 50% to demonstrate learning beyond random baseline performance. Full model training, evaluation, and backend integration will proceed during FYP 2.

3.6 TRAINING METHODOLOGY

Model 1 training employs consistent hyperparameters across both Compatibility Predictor and Complementary Item Retrieval stages to maintain reproducibility and enable direct performance comparison. The configuration addresses computational constraints imposed by TPU hardware availability and academic project timelines.

OneCycleLR scheduling manages learning rate progression with 30% warm-up phase followed by cosine annealing strategy. The warm-up phase enables gradual learning rate increases that help escape poor random initializations, while annealing enables fine-grained optimization during later training phases. This scheduling approach prioritizes efficient loss landscape exploration over traditional fixed or step-decay schedules.

AdamW optimizer applies weight decay regularization with gradient clipping to stabilize training under batch size constraints. Gradient accumulation provides sufficient negative sample diversity for InBatchTripletMarginLoss while accommodating hardware constraints. The methodology explores multiple feed-forward network dimensions to identify optimal capacity-efficiency trade-offs, with comparative results detailed in Section 5.4.

3.7 EVALUATION METHODOLOGY

3.7.1 MODEL EVALUATION

Model 1 evaluation uses two metrics addressing distinct performance aspects. Fill-in-the-Blank (FITB) accuracy measures the Complementary Item Retrieval model's ability to identify correct missing items from partial outfits. Area Under the ROC Curve (AUC) measures the Compatibility Predictor's ability to distinguish compatible from incompatible outfit combinations. These metrics enable direct comparison against published fashion recommendation benchmarks.

FITB evaluation presents partial outfits with one missing position and four candidate items (one ground truth, three random distractors). The model predicts an ideal embedding for the missing item and ranks candidates by cosine similarity. Correct predictions occur when ground truth ranks first among the four choices. This 4-choice format follows Polyvore benchmark protocol where CSA-Net establishes 63.73% accuracy as the baseline target (Sarkar

et al., 2023). The test set contains 10,000 examples from the non-disjoint split where items may appear in training but outfit combinations do not overlap.

AUC evaluation measures ranking quality by computing the probability that positive outfits receive higher scores than negative outfits across all possible pairs. The metric ranges from 0.5 (random) to 1.0 (perfect separation), providing threshold-independent assessment robust to class imbalance. Evaluation uses 20,000 test outfits from the non-disjoint compatibility split containing curated positives and random negatives. Model 2 targets AUC exceeding 0.5 to demonstrate learning beyond random baseline, acknowledging experimental status due to automated labeling noise.

3.7.2 SYSTEM EVALUATION

System evaluation measures deployment feasibility through latency and scalability benchmarks reflecting real-world conditions. Latency evaluation measures end-to-end recommendation time from query submission to result retrieval under CPU-only constraints. Measurements isolate model inference time, vector database retrieval time, and total pipeline latency to identify bottlenecks. Tests execute on standard cloud CPU instances (6 vCPUs, 12GB RAM) matching stakeholder deployment requirements, averaged over 100 queries to account for variance.

Scalability evaluation assesses performance as catalog size increases from 10,000 to 140,000 indexed products. Tests measure retrieval latency and throughput at different collection sizes to validate sub-second performance at production scale. Vector database employs HNSW indexing with m=16 connections and ef_construct=100 to balance build time against query speed. Evaluation verifies that precomputed embeddings maintain acceptable latency without runtime encoding, confirming CPU-first feasibility for cost-sensitive deployments. Performance measurements and infrastructure specifications appear in Section 5.5 and Section 5.6.

3.8 APP & BACKEND INFRASTRUCTURE

3.8.1 TECHNOLOGY STACK

The frontend uses React Native to build a cross-platform iOS application targeting iPhone devices running iOS 15 or later. React Native enables rapid prototyping through hot reload capabilities and maintains native performance through bridge-based architecture. The framework supports both light and dark mode theming through system preference detection.

The backend implements a Python-based API server using FastAPI framework for asynchronous request handling. FastAPI provides automatic OpenAPI documentation, type validation through Pydantic models, and native async/await support for concurrent request

processing. The framework enables efficient handling of multiple simultaneous recommendation queries without blocking operations. Python deployment runs on standard cloud CPU instances (6 vCPUs, 12GB RAM) to validate stakeholder requirements for GPU-free inference.

Data storage separates structured application data from high-dimensional vector embeddings. Qdrant serves as the vector database for storing and retrieving fashion item embeddings through approximate nearest neighbor search. The system indexes 140,000+ products with dual embeddings (768-dimensional image vectors and 768-dimensional text vectors from Marqo-FashionSigLIP) enabling hybrid search capabilities that combine visual and semantic matching. All embeddings are precomputed offline and frozen during inference, eliminating runtime encoding overhead. The recommendation model loads these frozen embeddings directly without executing the heavy Marqo backbone, enabling CPU-only inference at 500ms latency per query. MongoDB integration is planned for FYP 2 to manage structured user data, wardrobe metadata, and social features.

3.8.2 INTEGRATION WITH BACKEND SERVICES

The mobile application communicates with backend services through RESTful API endpoints over HTTPS. The integration implements request-response patterns where the React Native frontend sends HTTP requests to FastAPI and processes JSON responses. Key API endpoints include product search (text-to-text, text-to-image, hybrid), complementary item retrieval, product details lookup, and brand filtering. CORS middleware enables cross-origin requests from the mobile client while maintaining security through API authentication planned for FYP 2.

The recommendation workflow coordinates multiple backend operations sequentially. Frontend requests include partial outfit item IDs and candidate item IDs for ranking. The FastAPI server retrieves precomputed embeddings from Qdrant using item IDs, executes Model 1 inference to generate predicted embeddings, and computes similarity scores between predicted and candidate vectors. Results return as ranked JSON arrays sorted by compatibility distance. Model 2 integration planned for FYP 2 will add biometric re-ranking where top candidates from Model 1 pass through a second scoring stage before final response assembly. Implementation details explained in Section 4.6

CHAPTER 4: SYSTEM ANALYSIS AND DESIGN

4.1 INTRODUCTION

The system design prioritizes three core principles aligned with stakeholder requirements.

1. CPU-first architecture removes GPU dependency through precomputed embeddings and frozen backbone strategy, enabling deployment on standard cloud instances with 6 vCPUs and 12GB RAM.
2. The two-stage training approach transfers compatibility knowledge from the Compatibility Predictor to initialize the Complementary Item Retrieval model, improving convergence and accuracy through curriculum learning.
3. Separation of vector and structured data storage assigns high-dimensional fashion embeddings to Qdrant for approximate nearest neighbor search while MongoDB handles user profiles and social metadata planned for FYP 2.

Model 2 for biometric compatibility and full application features including user authentication, onboarding flows, and social interactions will be developed **during FYP 2** from March to June 2026, with FYP 1 focused on the foundation through Model 1 completion, core backend infrastructure, and iOS application at 20% completion.

4.2 REQUIREMENTS ANALYSIS

4.2.1 FUNCTIONAL REQUIREMENTS

Requirement ID	Feature/Requirements	Description	Priority	Status
FR-01	Smart Discovery - Text Search	Enable users to search the global fashion catalog using text queries through hybrid search combining Marqo-FashionSigLIP text embeddings (25% weight) and image embeddings (75% weight), returning semantically and visually relevant products from 140,000+ items.	High	Done
FR-02	Smart Discovery - Image Search	Allow users to upload or select reference images to find visually similar products through image embeddings, supporting style-based discovery without requiring explicit text descriptions.	High	FYP 2
FR-03	Wardrobe Management -	Allow users to browse and add items from	Critical	Partial

	CRUD Operations	the global catalog/upload to their personal wardrobe, modify item metadata including categories, descriptions, colors, and custom tags, and remove items from their collection with automatic database and vector index updates.		
FR-04	Wardrobe Management - Camera Upload	Enable users to photograph their existing clothing items, process images through Marqo-FashionSigLIP to generate embeddings, and store them in their personal wardrobe for recommendation queries.	High	FYP 2
FR-05	Outfit Recommendation - Model 1 CIR	Accept partial outfit inputs (1 to N items) and generate ideal complementary item embeddings through the CIR model, performing vector similarity search against the user's wardrobe to retrieve top-K ranked suggestions.	Critical	Done
FR-06	Outfit Recommendation - Model 2 Biometric Re-ranking	Process Model 1 candidate results alongside user biometric attributes (skin tone, body shape, gender, age) to compute personalized suitability scores, re-ordering recommendations to prioritize items that complement both outfit style and individual appearance.	Critical	FYP 2
FR-07	User Authentication	Allow users to create accounts with secure credential storage and authenticate returning users through session management, providing access to personal wardrobes, saved recommendations, and social activity history.	Critical	FYP 2
FR-08	User Onboarding	Guide new users through initial setup including facial image capture for biometric feature extraction (skin tone, facial structure), self-reported attribute input (body type, height, weight, age, gender), and wardrobe digitization via database import or camera capture to establish the foundation for personalized recommendations.	High	FYP 2
FR-09	Personalized Collections - Bookmarks	Allow users to save recommended outfits or individual items to custom boards or lists for future reference, enabling curation	Medium	FYP 2

		of favorite styles and shopping wishlists.		
FR-10	Social Features - Posts	Enable users to share their outfit photos with tags and captions to public or follower-only feeds, creating shareable fashion content within the community.	Low	FYP 2
FR-11	Social Features - Likes and Saves	Allow users to interact with community posts through like actions for engagement metrics and save functionality to bookmark inspirational outfits from other users.	Low	FYP 2
FR-12	Social Features - Follow System	Implement user-to-user following relationships to curate personalized social feeds.	Low	FYP 2
FR-13	Social Features - Profile Viewing	Display user profiles showing posted outfits, follower counts, and engagement metrics to support community interaction and discovery.	Low	FYP 2

Table 2. Functional Requirements Specification

4.2.2 NON-FUNCTIONAL REQUIREMENTS

Requirement ID	Category/Requirements	Description	Target/Metric	Status
NFR-01	AI Recommendation Performance - End-to-End Latency	Complete outfit recommendation workflow from user input to final ranked results display, including Model 1 inference, vector retrieval, and optional Model 2 re-ranking for biometric personalization.	≤ 4 seconds	Partial (Currently, model 1 takes ≤ 500ms)
NFR-02	Performance - Vector Retrieval Speed	Query response time for approximate nearest neighbor search across the global fashion catalog using HNSW indexing in Qdrant vector database.	Sub-second for 140,000+ items	Done
NFR-03	Resource Efficiency - CPU-First Deployment	All system must operate entirely on standard cloud CPU instances without requiring GPU hardware (including the database, API, and the model)	6 vCPU, 12GB RAM, no GPU	Done

NFR-04	Scalability - Catalog Size	Support indexing, storage, and efficient retrieval across large-scale fashion product databases with continuous growth capability for expanding inventory.	≥ 140,000 items	Done
NFR-05	Scalability - Concurrent Users	Handle multiple simultaneous user requests for search, recommendation, and wardrobe operations without performance degradation through stateless API design and efficient database connection pooling.	Support planned user load	FYP 2
NFR-06	Reliability - API Availability	Maintain consistent uptime for backend services including FastAPI endpoints, Qdrant vector database, and MongoDB operations to ensure seamless user experience during active sessions.	≥ 99% uptime	Done. Server deployed in Contabo
NFR-07	Security - User Authentication	Implement secure credential management using industry-standard hashing algorithms, token-based session authentication, and encrypted data transmission to protect user accounts and personal information.	Secure login/registration	FYP 2
NFR-08	Interoperability - Frontend-Backend Integration	Establish seamless communication between React Native iOS application and FastAPI backend through RESTful API design, standardized JSON payloads, and robust error handling mechanisms.	RESTful API standards	Partial
NFR-09	Interoperability - Database Integration	Enable efficient data flow between application layer and storage systems, coordinating vector operations in Qdrant with structured data management in MongoDB while maintaining consistency across distributed components.	Seamless multi-database coordination	Partial
NFR-10	Usability - Theme Support	Provide visual customization through automatic detection and rendering of Light and Dark mode themes based on device system preferences, ensuring comfortable viewing across different lighting conditions.	Light/Dark mode toggle	FYP 2

NFR-11	Demographic Localization - Asian Fashion Optimization	Optimize model performance and catalog content for Asian fashion styles, body types, and aesthetic preferences through training data sourced from regional e-commerce platforms Musinsa and Wconcept.	Optimized for Asian market	Partial

Table 3. Non-Functional Requirements Specification

4.2.3 MODEL QUERY REQUIREMENTS

Requirement ID	Model/Task	Description	Target Metric	Achieved/Status
MQR-01	Model 1 - Complementary Item Retrieval (CIR)	Generate ideal complementary item embeddings from partial outfit inputs to enable vector-based retrieval of matching wardrobe items without requiring explicit target category specifications or text descriptions. **Input:** Partial outfit (1 to N items × 1536-dim embeddings). **Output:** Predicted item embedding (128-dim vector).	FITB Accuracy ≥ 63.73% (CSA-Net baseline)	Done. Achieved 64.26% FITB accuracy
MQR-02	Model 2 - Biometric Compatibility Prediction	Fuse user biometric attribute embeddings with outfit item representations to predict personalized suitability scores, enabling re-ranking of Model 1 candidates based on individual physical characteristics and demographic features. **Input:** Outfit items (N × 1536-dim) + User attributes (skin tone, body type, gender, age). **Output:** Personalized compatibility score (0-1 probability).	AUC > 50% (exceeding random baseline)	FYP 2. Data preparation phase in progress

Table 4. Model Query Requirements Specification

4.3 SYSTEM ARCHITECTURE DESIGN

4.3.1 HIGH-LEVEL ARCHITECTURE

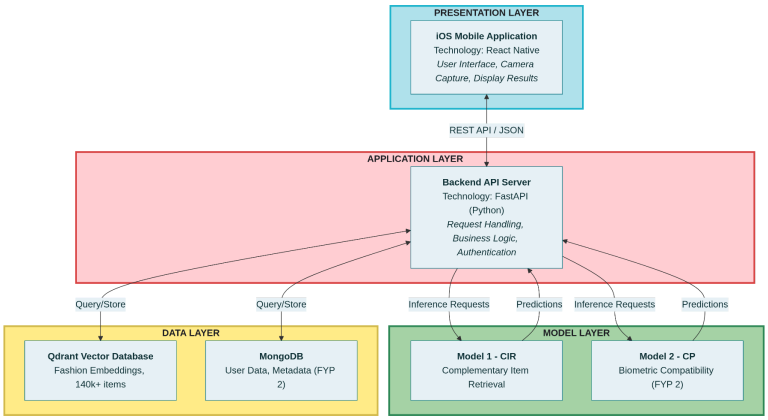


Figure 10. System High-Level Architecture

The system follows a four-layer architecture separating concerns across presentation, application, model, and data components. The Presentation Layer implements the React Native iOS application serving as the user-facing interface for wardrobe management, smart discovery, and outfit recommendations. Users interact with screens for camera capture, search queries, and result displays, communicating with backend services through RESTful API calls.

The Application Layer hosts the FastAPI backend server handling HTTP requests, routing logic, session management, and orchestration between models and databases. This layer processes user requests by invoking model inference, querying vector databases, and returning formatted JSON responses to the mobile client. The Model Layer contains two models where Model 1 CIR accepts partial outfit embeddings and generates predicted item vectors for retrieval, while Model 2 CP planned for FYP 2 fuses biometric attributes with outfit representations for personalized compatibility scoring. Both models leverage precomputed Marqo-FashionSigLIP embeddings to enable CPU-only inference without runtime encoding overhead.

The Data Layer separates vector storage from structured data management. Qdrant handles high-dimensional fashion embeddings for 140,000+ catalog items, supporting approximate nearest neighbor search through HNSW indexing with sub-second query latency. MongoDB planned for FYP 2 manages user profiles, authentication credentials, wardrobe metadata, and social interaction records. This separation optimizes each database for its specific workload where Qdrant prioritizes similarity search performance and MongoDB supports flexible schema for user-generated content.

4.3.2 DATA FLOW DIAGRAM

Smart Discovery Flow

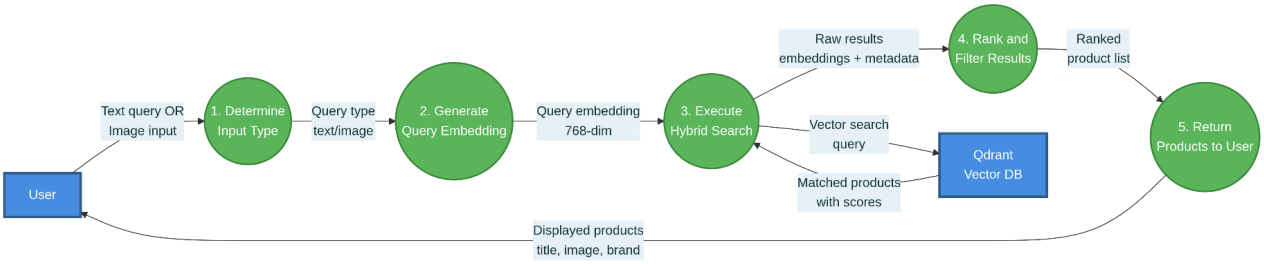


Figure 11. Smart Discovery Data Flow Diagram

The Smart Discovery workflow processes user search queries through adaptive embedding generation and weighted vector retrieval.

- Process 1 detects input modality where text queries trigger hybrid search combining 25% text embeddings with 75% image embeddings, while image-only inputs use 100% image vector search.
- Process 2 invokes Marqo-FashionSigLIP encoder to generate 768-dimensional embeddings from the query.
- Process 3 executes ANN search in Qdrant using HNSW indexing, applying optional brand filters through payload field conditions and retrieving candidates with similarity scores.
- Process 4 merges keyword matches with vector results, calculating final scores through weighted combination where keyword boost contributes configurable weight and vector similarity fills remaining proportion.
- The system returns ranked products to users with metadata including titles, images, brands, and decomposed scores for transparency

Outfit Recommendation Flow

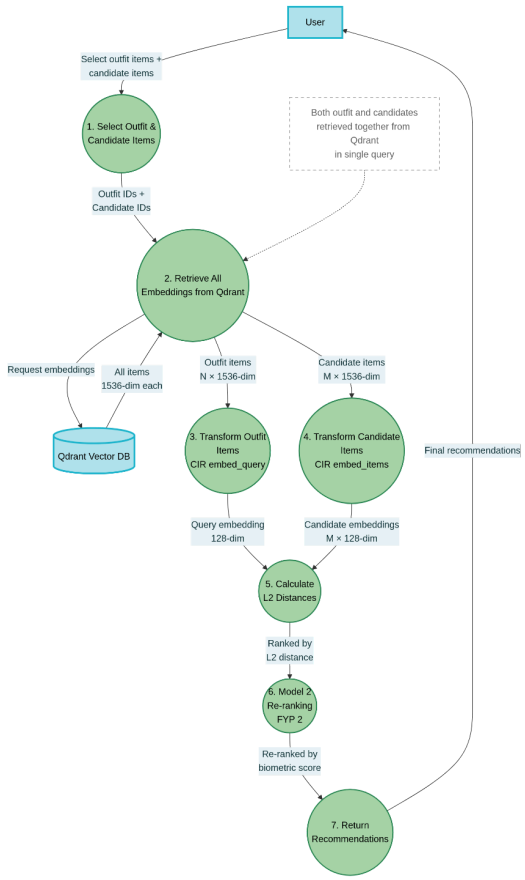


Figure 12. Outfit Recommendation Data Flow Diagram

The Outfit Recommendation workflow implements dimensional reduction through dual-path CIR transformation enabling efficient comparison in compressed embedding space.

- Process 1 accepts user selection of partial outfit items and candidate items to evaluate, where users specify which garments define the base outfit and which wardrobe items should be ranked for compatibility.
- Process 2 retrieves all required items both outfit and candidates in a single batch query from Qdrant, where each item exists as concatenated 1536-dimensional vectors combining 768-dimensional image embeddings with 768-dimensional text embeddings.
- Process 3 feeds outfit items through the CIR model's embed_query function which applies transformer self-attention to capture inter-item relationships and projects the outfit representation into a unified 128-dimensional query embedding representing the collective style signature of the partial outfit.

- Process 4 transforms candidate items through the same CIR model's embed_items function, independently projecting each candidate from 1536 dimensions to 128 dimensions while preserving discriminative features for compatibility assessment.
- Process 5 calculates L2 Euclidean distances between the 128-dimensional query embedding and each candidate embedding, where lower distances indicate stronger style compatibility and items exceeding distance threshold 1.414 are filtered as incompatible.
- Process 6 planned for FYP 2 re-ranks filtered candidates by fusing outfit representations with user biometric attributes through Model 2 CP, computing personalized suitability scores that account for skin tone, body type, and demographic features.
- Process 7 returns sorted recommendations where candidates are ordered first by Model 2 biometric score when available, otherwise by L2 distance from Model 1.

Wardrobe Management Flow

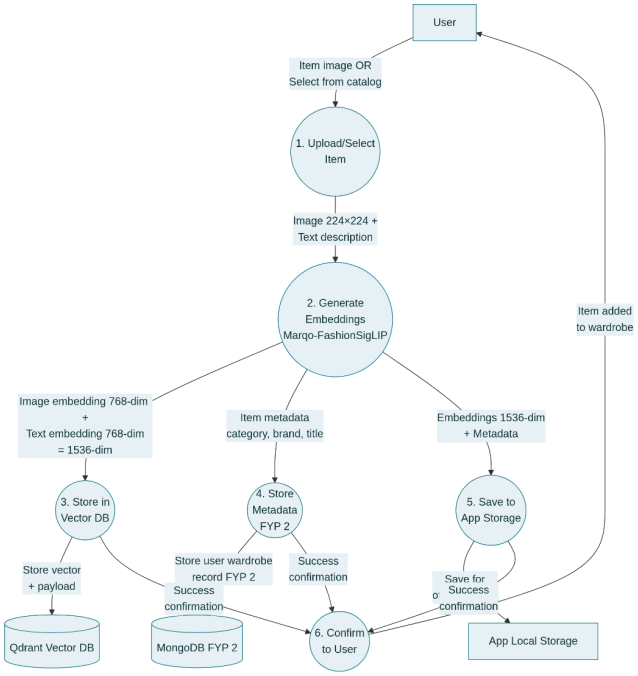


Figure 13. Wardrobe Management Data Flow Diagram

The Wardrobe Management workflow handles item ingestion through 2 destination storage ensuring data availability across backend databases and local app cache.

- Process 1 accepts users to either upload / select the items from the database.
- Process 2 generates embeddings through Marqo-FashionSigLIP encoder accepting 224×224 resized images and text descriptions, producing separate

768-dimensional vectors for visual and semantic features which concatenate to form unified 1536-dimensional item representations.

- Process 3 stores vectors in Qdrant with payload metadata enabling future similarity searches during outfit recommendations, leveraging HNSW indexing for sub-second retrieval across 140,000+ catalog items.
- Process 4 planned for FYP 2 persists structured metadata including user ownership, categories, custom tags, and timestamps in MongoDB supporting relational queries for wardrobe organization and social features.
- Process 5 duplicates embeddings and metadata to app local storage providing offline access where users view wardrobe contents without network connectivity and enabling Process 2 in Outfit Recommendation to load embeddings locally instead of querying Qdrant. The system confirms successful addition across all storage destinations before notifying users.

4.4 MODEL ARCHITECTURE DESIGN

4.4.1 EMBEDDING GENERATION DESIGN

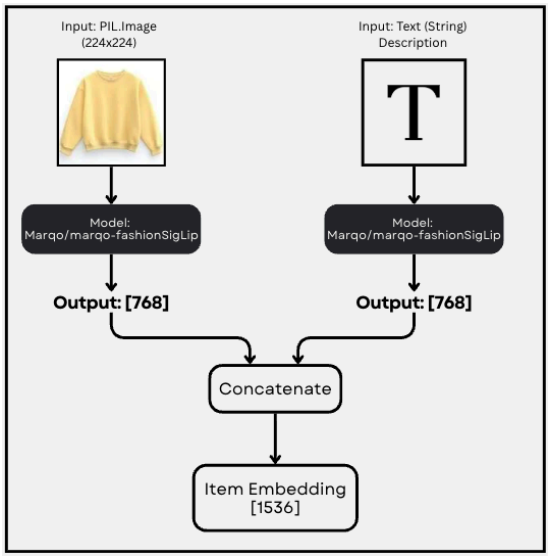


Figure 14. Embedding Generation Pipeline

The embedding generation pipeline converts fashion items into 1536-dimensional vectors through the frozen Marqo-FashionSigLIP model. Product images resized to 224×224 pixels pass through the ViT-B-16-SigLIP vision encoder to produce 768-dimensional visual embeddings, while text descriptions pass through the text encoder to generate 768-dimensional semantic embeddings. Both outputs undergo L2 normalization and concatenate into final 1536-dimensional representations where the first 768 dimensions capture visual features and the last 768 dimensions capture textual features. The model stays frozen during training, and all embeddings are precomputed offline and stored in Qdrant, removing the

need for runtime encoding and allowing CPU-only inference for the CIR checkpoint uses (not the training).

4.4.2 CP MODEL ARCHITECTURE

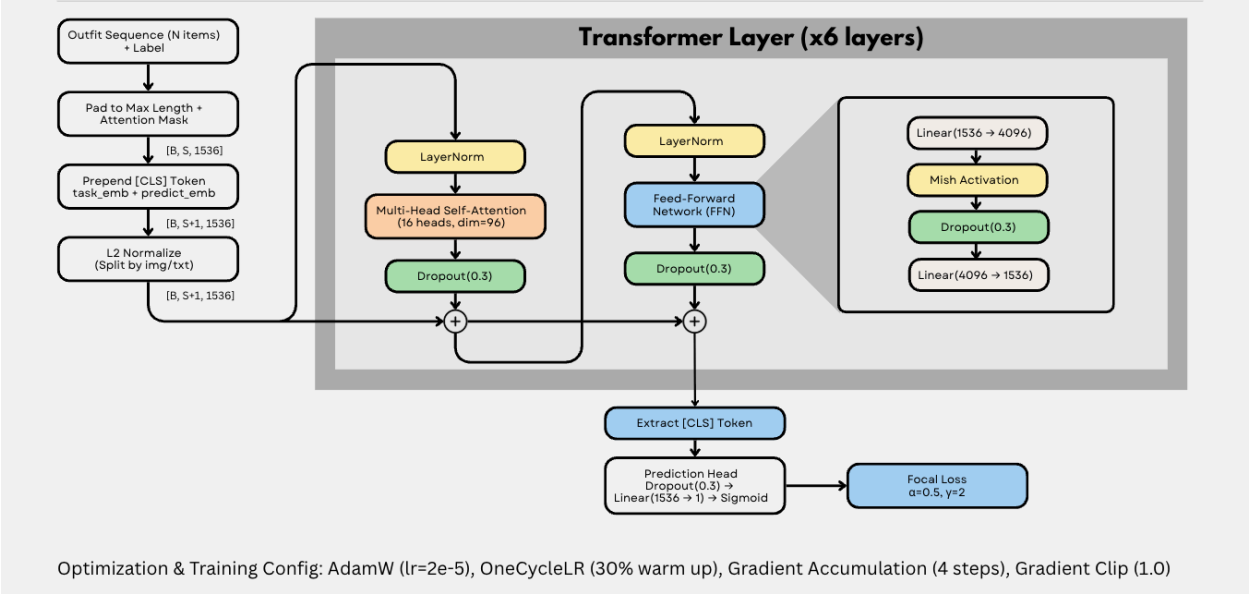


Figure 15. CP Model Architecture Diagram

Component	Configuration	Details
Input Dimension	1536	768 image + 768 text concatenated
Transformer Layers	6 layers	Each with LayerNorm, Multi-Head Self-Attention, FFN, Dropout, Residual connections
Attention Heads	16 heads	Head dimension = 1536/16 = 96
Feed-Forward Network (FFN)	4096 hidden dim	Linear(1536 → 4096) + Mish + Linear(4096 → 1536)
Dropout	0.3	Applied after attention and FFN in each layer
Prediction Head	1536 → 1	Dropout(0.3) + Linear(1536 → 1) + Sigmoid
Output	Binary score	Compatibility probability (0-1)

Total Parameters	**132.23M**	Task embeddings (3.8K) + Transformer (132.2M) + Prediction head (1.5K)

Table 5. CP Model Architecture Configuration

Input/Output Dimensions:

- Input: [B, N, 1536] where B = batch size, N = number of items
- After prepend [CLS]: [B, N+1, 1536]
- After transformer: [B, N+1, 1536]
- After [CLS] extraction: [B, 1536]
- Output: [B, 1] compatibility score

The CP model processes complete outfits to predict binary compatibility through a transformer-based architecture trained with Focal Loss. Configuration details appear in Table 5.

4.4.3 CIR MODEL ARCHITECTURE

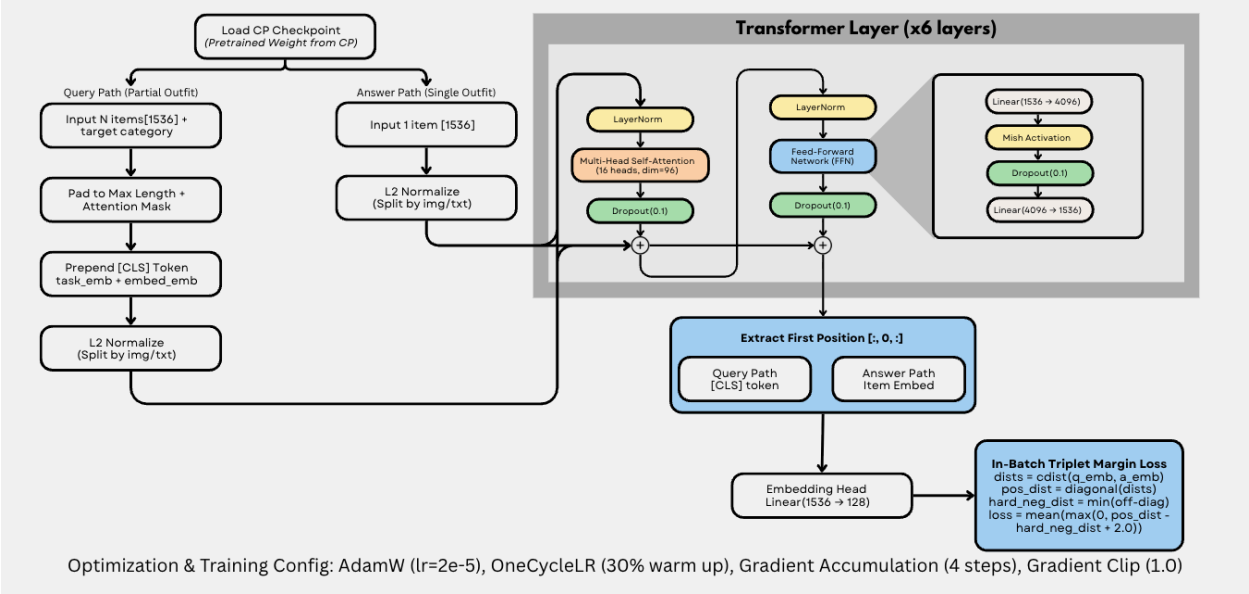


Figure 16. CIR Model Architecture Diagram

Component	Configuration	Details
Input Dimension	1536	768 image + 768 text concatenated
Query Path Input	N items + target category	Partial outfit without missing item
Answer Path Input	1 item	Single candidate item embedding
Task Token	task_emb + embed_emb	Different from CP (which uses task_emb + predict_emb)
Transformer Layers	6 layers	Shared encoder with CP (initialized from CP checkpoint)
Attention Heads	16 heads	Head dimension = 1536/16 = 96
Feed-Forward Network	4096 hidden dim	Linear(1536 → 4096) + Mish + Linear(4096 → 1536)
Dropout	0.1	Lower than CP (0.3) to reduce overfitting during transfer learning
Embedding Head	1536 → 128	Linear projection with no bias
Output	128-dim vector	L2 normalized embeddings for distance computation
Total Parameters	**132.43M**	Task embeddings (3.8K) + Transformer (132.2M) + Embedding head (197K)

Table 6. CIR Model Architecture Configuration

Input/Output Dimensions:
Query Path (Partial Outfit):

- Input: [B, N, 1536] where N = number of items in partial outfit
- After padding: [B, 16, 1536] (padded to max length)
- After prepend [CLS]: [B, 17, 1536]
- After transformer: [B, 17, 1536]
- After [CLS] extraction: [B, 1536]

- Output: [B, 128] query embedding

Answer Path (Single Item):
- Input: [B, 1, 1536] single candidate item
- After L2 norm: [B, 1, 1536]
- After transformer: [B, 1, 1536]
- After [CLS] extraction: [B, 1536]
- Output: [B, 128] item embedding

The CIR model extends the CP architecture through dual-path processing where partial outfits and candidate items pass through the same transformer encoder but generate separate 128-dimensional embeddings for distance-based matching. Configuration details appear in Table 6.

4.4.4 MODEL CONFIGURATION

CP Model Configuration

Parameter	Value	Notes
Embedding Dimension	1536	768 image + 768 text concatenated
Transformer Layers	6	Pre-LN architecture with residual connections
Attention Heads	16	Head dimension = 96
Feed-Forward Network	4096 hidden	Linear(1536 → 4096) + Mish + Linear(4096 → 1536)
Dropout	0.3	Applied after attention and FFN
Activation Function	Mish	Self-regularizing, smooth gradient flow
Output	1 (Sigmoid)	Binary compatibility score
Loss Function	Focal Loss	α=0.5, γ=2 for class imbalance
Optimizer	AdamW	Weight decay regularization
Learning Rate	2e-5	Suitable for fine-tuning frozen backbone

Learning Rate Schedule	OneCycleLR	30% warm-up, cosine annealing
Batch Size	64 (per GPU)	Gradient accumulation over 4 steps → effective batch 256
Epochs	200	Full training convergence
Gradient Clipping	1.0	Norm-based clipping
Total Parameters	**132.23M**	Task embeddings (3.8K) + Transformer (132.2M) + Prediction head (1.5K)

Table 7. CP Model Training Configuration

CIR Model Configuration

Parameter	Value	Notes
Initialization	CP Checkpoint	Loads pretrained weights from trained CP model
Embedding Dimension	1536	Same as CP (frozen backbone)
Output Embedding	128	Linear(1536 → 128) projection, no bias
Transformer Layers	6	Shared encoder from CP checkpoint
Dropout	0.1	Reduced from CP (0.3) for transfer learning
Loss Function	InBatchTripletMarginLoss	margin=2.0, efficient negative sampling
Optimizer	AdamW	Same as CP
Learning Rate	2e-5	Same as CP
Learning Rate Schedule	OneCycleLR	30% warm-up, cosine annealing

Batch Size	64 (per GPU)	Gradient accumulation over 4 steps → effective batch 256
Epochs	200	Full training convergence
Gradient Clipping	1.0	Norm-based clipping
Total Parameters	**132.43M**	Task embeddings (3.8K) + Transformer (132.2M) + Embedding head (197K)

Table 8. CIR Model Training Configuration

Training Infrastructure

Component	Specification
Hardware	Google Cloud TPU v4 (4 chips, 8 TensorCores)
Region	us-central2-b
Training Duration	6-7 hours per stage (200 epochs)
Device	XLA (TPU compiled)
Batch Type	Per-GPU batch size 64
Gradient Accumulation	4 steps
Effective Batch Size	256 samples

Table 9. Training Infrastructure Specifications

4.5 DATABASE DESIGN

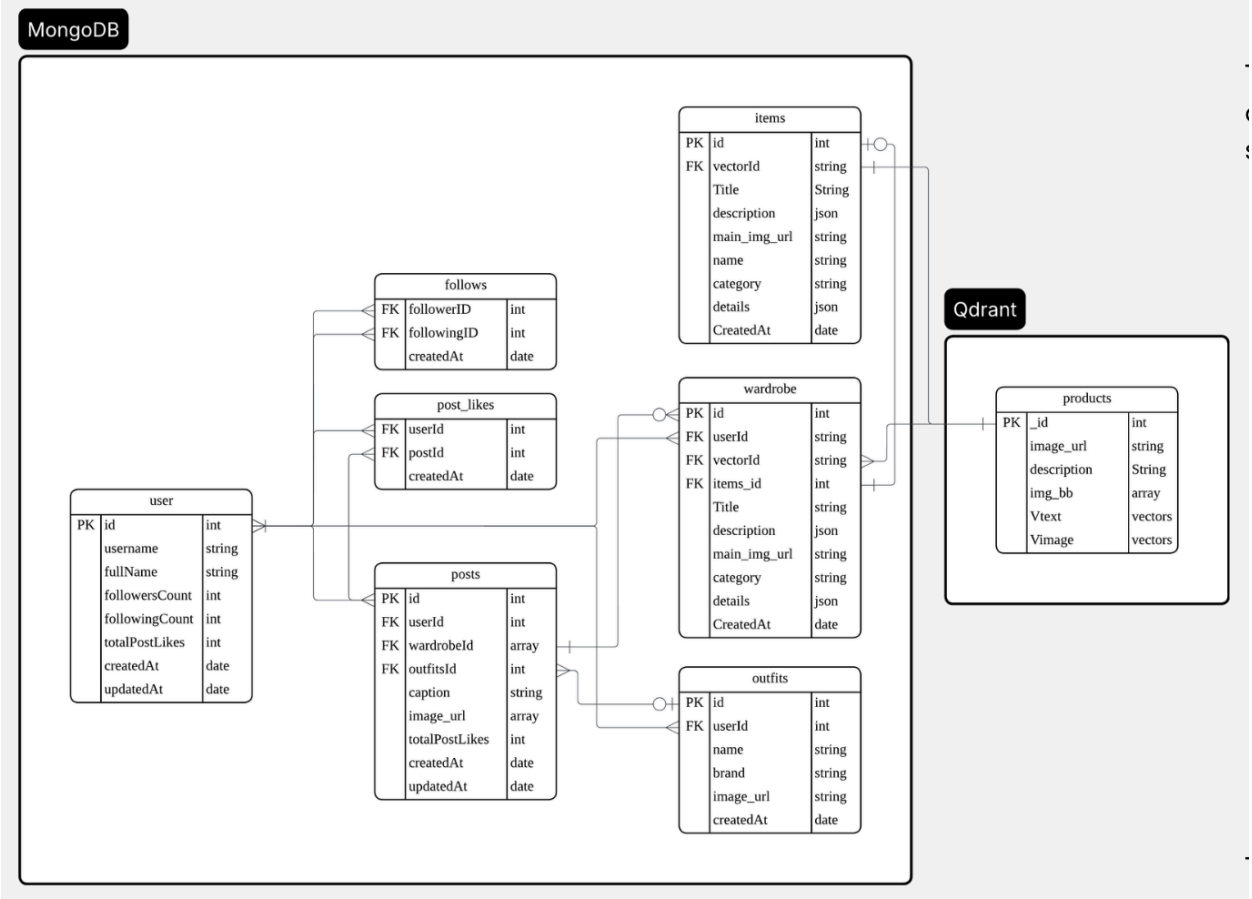


Figure 17. Database Schema Design

MongoDB manages structured data, user relationships, and content metadata. The NoSQL solution handles semi-structured data with nested JSON fields (description, details) and variable schema structures across different product categories. The database stores user profiles, wardrobe items, social interactions, and product metadata with flexible schema design, horizontal scalability, and native JSON support.

Qdrant stores and searches embeddings for the Global Catalog and Recommendation System. The products collection stores Vtext (768-dim) and Vimage (768-dim) vectors. The open-source vector database optimizes Approximate Nearest Neighbor (ANN) search using HNSW (Hierarchical Navigable Small World) graph indexing to reduce search complexity from O(N) to O(log N), ensuring sub-second retrieval across 140,000+ items.

4.6 API

1. **GET / (Root)**
    - Description: API information and endpoint list

- Response: Service metadata and available endpoints

2. **GET /health**
    - Description: Health check endpoint
    - Response: System status including model loaded, CIR model loaded, Qdrant connected, collection points count
3. **POST /create-index**
    - Description: Creates full-text index on title field for fast keyword search
    - Note: Idempotent operation, safe to call multiple times
    - Response: Status message confirming index creation
4. **GET /t2t (Text-to-Text Search)**
    - Description: Semantic search using text descriptions
    - Parameters:
        - text (required): Search query, supports brands[brand1,brand2] filter
        - limit (optional, default=10): Number of results (1-100)
        - keywordweight (optional, default=0.5): Weight for keyword matching (0.0-1.0)
    - Process: Embeds query using Marqo-FashionSigLIP text encoder, searches Qdrant Vtext vectors
    - Response: Ranked results with textScore, imageScore (0.0), keywordScore, and product payload
5. **GET /t2i (Text-to-Image Search)**
    - Description: Visual similarity search using text queries
    - Parameters: Same as /t2t
    - Process: Embeds query using text encoder, searches Qdrant Vimage vectors
    - Response: Ranked results with imageScore, textScore (0.0), keywordScore, and product payload
6. **GET /search (Hybrid Search)**
    - Description: Combined text-to-text and text-to-image search with configurable weights
    - Parameters:
        - text (required): Search query with optional brands filter
        - textweight (optional, default=0.5): Weight for text similarity (0.0-1.0)
        - imageweight (optional, default=0.5): Weight for image similarity (0.0-1.0)
        - keywordweight (optional, default=0.4): Weight for keyword matching (0.0-1.0)
        - limit (optional, default=10): Number of results (1-100)
    - Process: Computes weighted combination of Vtext and Vimage similarity scores
    - Response: Ranked results with separate scores for text, image, and keyword components

- Note: At least one weight must be greater than 0

7. **GET /details**
   - Description: Fetch full product details by ID
   - Parameters: id (required): Product ID (string or numeric)
   - Response: Product payload including id, title, description, images, category, brand, details
8. **GET /brands**
   - Description: Get all unique brand names from catalog
   - Parameters: refresh (optional, default=false): Force refresh brands cache
   - Response: List of brands, total count, cached status, cache age, scanned products count
   - Note: Results cached for 1 hour (3600 seconds), scans up to 10,000 products
9. **GET /testCIR (Complementary Item Retrieval)**
   - Description: Rank candidate items by compatibility with partial outfit using CIR model
   - Parameters:
     - outfit (required): Comma-separated item IDs for partial outfit (e.g., "101,102,103")
     - asking (required): Comma-separated candidate item IDs to rank (e.g., "201,202,203")
   - Process:
     - Retrieves embeddings from Qdrant for outfit and candidate items
     - Computes query embedding using CIR model embed_query method
     - Computes candidate embeddings using CIR model embed_items method
     - Calculates L2 distances between query and each candidate
     - Sorts candidates by distance (ascending, lower is better)
   - Response: Ranked candidates with id, distance, rank, outfit count, candidates count
   - Note: Returns 404 if embeddings missing, 503 if CIR model not loaded

4.7 USER INTERFACE DESIGN

The iOS application uses React Native framework to deliver cross-platform functionality for iPhone devices running iOS 15 or later. The interface prioritizes three core workflows: wardrobe digitization, smart discovery search, and outfit recommendation.

User Journey Workflow:

1. **Onboarding and Profile Creation (Planned FYP 2)**
   The user initializes their profile by inputting biometric data and scanning their face for skin tone analysis. The system normalizes these inputs and passes them through the

Attribute Embedding Layer, generating a dense User Attribute Vector (128-dim). This vector serves as the Personalization Key for Model 2 biometric compatibility scoring.

2. **Wardrobe Digitization and Vector Indexing**
   Users upload photos of existing clothes or save items from the global catalog. The Marqo-FashionSigLIP backbone processes each image to extract deep semantic visual features. These 1536-dimension vectors (768 image + 768 text) store in Qdrant, enabling high-speed similarity search and retrieval.
3. **Smart Discovery (Hybrid Search)**
   The user searches for specific products using text queries (e.g., "Vintage Denim Jacket"). The system converts the query into embeddings and executes hybrid search with 75% visual weight and 25% text weight to retrieve relevant items from the 140,000+ global catalog. Search results display with product images, titles, descriptions, and category tags.
4. **Outfit Recommendation**
   The user selects a single anchor item (e.g., beige shirt) and requests complementary items.

   Stage 1 (Generative Retrieval - Model 1):
   - CIR model analyzes the anchor item and predicts the vector of the ideal missing item (e.g., Dark Brown Pants)
   - Qdrant performs Nearest Neighbor search to fetch top 10 visual candidates from the user's wardrobe
   - Threshold filter retains only items where Vector Distance < 1.414, ensuring positive cosine similarity

   Stage 2 (Biometric Re-Ranking - Model 2, Planned FYP 2):
   - Filtered candidates feed into the Biometric Fusion Model alongside the User Attribute Vector
   - Model assigns suitability probability (0.0 to 1.0) to each candidate
   - Final list displays sorted by Model 2 Biometric Score (highest to lowest)

CHAPTER 5: IMPLEMENTATION AND RESULT

5.1 DEVELOPMENT ENVIRONMENT

The model training leverages cloud computing resources provided through the Google TPU Research Cloud (TRC) program. Google allocated a TPU v4 instance for one month at no cost, enabling the high-performance training without requiring local GPU infrastructure.

Hardware Specifications:

- Compute: Google Cloud TPU v4 (4 chips, 8 TensorCores)
- Memory: 64 GB High Bandwidth Memory (HBM) per TPU core
- Zone: us-central2-b
- Storage: Google Cloud Storage bucket for dataset and checkpoint storage

Software Environment:

- Framework: PyTorch with PyTorch/XLA extension for TPU acceleration
- Python Version: 3.8+
- Key Libraries:
    - torch_xla: PyTorch XLA for TPU support
    - open_clip: Loading Marqo-FashionSigLIP model
    - transformers: HuggingFace model loading
    - datasets: HuggingFace dataset loading
    - wandb: Experiment tracking and logging

The training pipeline includes native TPU support through the --device xla flag, which automatically configures the model and optimizer for TPU execution using torch_xla.core.xla_model.

5.2 PRETRAINING

5.2.1 DATASET

The model uses the Polyvore Outfits Dataset, a standard benchmark for fashion compatibility and complementary item retrieval tasks. The dataset was obtained from HuggingFace repositories owj0421/polyvore-outfits for outfit data and owj0421/polyvore for item images.

Dataset Variants:

- **nondisjoint:** Items may appear in both train and test sets
- **disjoint:** Strict separation of items between train and test splits

Dataset Statistics (nondisjoint):

Subset	Train	Validation	Test
Compatibility (CP)	~32,000	~4,000	~4,000
Fill-in-the-Blank (FITB)	~16,000	~2,000	~2,000
Default (Triplet/CIR)	~53,000	~7,000	~7,000

Table 10. Polyvore Dataset Statistics by Subset

Total unique items: ~164,000 fashion items with images and metadata including category, title, and description. Each outfit consists of 2 to 8 fashion items that are stylistically compatible, curated by users on the Polyvore platform.

5.2.2 EMBEDDING GENERATION

Precomputed embeddings were generated for all fashion items using the Marqo-FashionSigLIP model (Marqo/marqo-fashionSigLIP), a domain-specific vision-language model fine-tuned on fashion data.

Embedding Generation Process:

Image Embeddings (--stage generate_embeddings):

- Model: Marqo/marqo-fashionSigLIP (ViT-B-16-SigLIP architecture)
- Output dimension: 768
- Process: Each item image passes through the visual encoder producing a 768-dimensional feature vector
- Storage: polyvore_cache/embeddings/polyvore_embeddings.pkl

Text Embeddings (--stage generate_text_embeddings):

- Model: Same Marqo/marqo-fashionSigLIP model
- Output dimension: 768
- Input: Concatenation of item metadata (category + title/url_name)
- Storage: polyvore_cache/embeddings/polyvore_text_embeddings.pkl

Combined Embeddings:

- Concatenation: [Vimage | Vtext] = 768 + 768 = 1536 dimensions
- Used as input for transformer training

Why Precomputed Embeddings:

- Reduces training time by avoiding repeated forward passes through the CLIP encoder
- Enables faster experimentation with different transformer configurations
- Memory efficient: only 1536-dimensional vectors are loaded instead of full images during training

5.3 MODEL TRAINING IMPLEMENTATION

5.3.1 MODEL 1 (CP) TRAINING

The Compatibility Predictor serves as the first stage in the two-stage training strategy, learning outfit-level representations through binary classification of complete outfit combinations. Training explores five configurations to identify optimal hyperparameter settings for the feed-forward network dimension (d_ffn) and dropout regularization.

Training Configuration:

All CP experiments use consistent base hyperparameters:

- Learning rate: 0.00002 (2e-5)
- Optimizer: AdamW with OneCycleLR scheduler
- Batch size per GPU: 64
- Gradient accumulation steps: 4 (effective batch size: 256)
- Number of epochs: 200
- Loss function: Focal Loss (α=0.5, γ=2)
- Device: TPU (--device xla)
- Embedding dimension: 1536 (768 image + 768 text)

Hyperparameter Exploration Results:

Model Name	Learning Rate	Dropout	d_ffn	Valid AUC	Notes
cp200	0.00002	0.3	2048	0.9259	Set d_ffn = 2048 (as the paper approach)
cp-drop01	0.00002	0.1	2048	0.9245	Reduce the dropout to test if it increases AUC
cp-dffn4096	0.00002	0.1	4096	0.9217	Set d_ffn to x4 size of embedding dimension

cp-4096	0.00002	0.3	4096	0.9242	Increase the dropout to 0.3
cp-1536	0.00002	0.3	1536	0.9274	Try d_ffn to x1 size of the dimension

Table 11. CP Model Hyperparameter Exploration Results

Training Command:

```
python -m src.run.polyvore_pipeline --stage train_cp \
 --batch_sz_per_gpu 64 \
 --accumulation_steps 4 \
 --n_epochs 200 \
 --lr 2e-5 \
 --d_ffn 4096 \
 --dropout 0.3 \
 --use_text_embeddings \
 --precomputed_embedding_dim 1536 \
 --device xla \
 --no_early_stopping \
 --wandb_key xxx \
 --project_name outfit-transformer-cp
```

5.3.2 MODEL 1 (CIR) TRAINING

The Complementary Item Retrieval model builds upon the pretrained Compatibility Predictor through transfer learning, adapting the learned outfit compatibility representations for the generative task of predicting missing item embeddings from partial outfit inputs. Training explores the impact of CP initialization and hyperparameter configurations on Fill-in-the-Blank (FITB) accuracy.

Training Configuration:

All CIR experiments use consistent base hyperparameters:

- Learning rate: 0.00002 (2e-5)
- Optimizer: AdamW with OneCycleLR scheduler
- Batch size per GPU: 64
- Gradient accumulation steps: 4 (effective batch size: 256)
- Number of epochs: 200
- Loss function: InBatchTripletMarginLoss (margin=2.0)
- Device: TPU (--device xla)
- Embedding dimension: 1536 (768 image + 768 text)

Hyperparameter Exploration Results:

Model Name	Checkpoint	Learning Rate	Dropout	d_ffn	Valid ACC	Notes
cir-no-cp	No	0.00002	0.3	2048	0.4966	Set d_ffn = 2048 (as the paper approach)
cir-cp	cp-200	0.00002	0.3	2048	0.6240	Use CP as a checkpoint to increase accuracy
cir-dropout 0.1	cp-200	0.00002	0.1	2048	0.6506	Reduce the dropout to increase accuracy
cir-1536	cp-1536	0.00002	0.1	1536	0.6528	Change the d_ffn, following the checkpoint
cir-4096	cp-4096	0.00002	0.1	4096	0.6540	Change the d_ffn, following the checkpoint

Table 12. CIR Model Hyperparameter Exploration Results

Training Command:

```
python -m src.run.polyvore_pipeline --stage train_cir \
 --checkpoint best_model.pth \
 --d_ffn 4096 \
 --batch_sz_per_gpu 64 \
 --accumulation_steps 4 \
 --n_epochs 200 \
 --lr 2e-5 \
 --dropout 0.1 \
```

```
--use_text_embeddings \
--precomputed_embedding_dim 1536 \
--device xla \
--no_early_stopping \
--wandb_key xxx \
--project_name outfit-transformer-cir
```

5.4 MODEL RESULT

5.4.1 CP PERFORMANCE

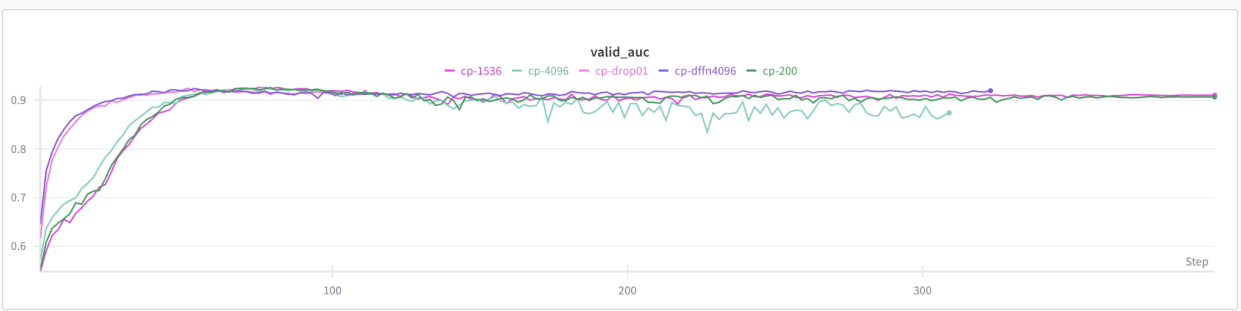


Figure 18. Validation AUC curves across all 5 CP training

Figure 18 shows the validation AUC curves across all five CP training configurations over 200 epochs. The convergence patterns demonstrate rapid learning in the first 100 steps followed by stable performance plateaus.

Key Findings:

- **AUC (Area Under ROC Curve)** measures how well the model ranks compatible outfits above incompatible ones, which is critical for a recommendation system where relative ranking matters more than absolute scores.
- **All configurations achieve greater than 92% AUC**, validating that Marqo-FashionSigLIP embeddings are effective for fashion compatibility prediction
- **Optimal model: cp-1536** (d_ffn=1×embedding, dropout=0.3) achieves 93.74% AUC, where smaller FFN generalizes better than 4× larger versions
- **Higher dropout (0.3) outperforms lower (0.1)**, suggesting fashion patterns need careful regularization to avoid overfitting on training outfit combinations
- **Test AUC validation:** Spot checks on test_auc show similar performance to valid_auc with margin error of 0.05, confirming model generalization

5.4.2 CIR PERFORMANCE

Figure 19. CIR training results: Validation Accuracy (top) and Training Loss Curves (bottom) across 5 configurations

Figure 19 illustrates the CIR training dynamics through validation accuracy (top) and training loss curves (bottom) across five configurations. The graphs reveal distinct performance tiers based on initialization strategy and hyperparameter selection.

Key Findings:

Two-Stage Training is Essential:

The cir-no-cp baseline (pink line) shows dramatically inferior performance, plateauing around 49.66% accuracy and maintaining high training loss throughout all epochs. Models initialized from CP checkpoints (other four lines) cluster together at higher accuracy levels around 62-65%, demonstrating a 16% absolute improvement from transfer learning. This validates the two-stage training hypothesis where the model learns what matches (CP) before learning what is missing (CIR), providing better weight initialization and faster convergence.

Dropout Regularization Impact:

Reducing dropout from 0.3 to 0.1 increases validation accuracy from 62.40% to 65.06%, suggesting the CIR model benefits from retaining more learned features during the generative prediction task compared to the discriminative CP task.

Architectural Consistency:

The final configuration cir-4096 (black line) matches its d_ffn (4096) with the corresponding CP checkpoint (cp-4096), achieving the highest validation accuracy of 65.40%. Although cp-1536 achieved the highest CP validation AUC (93.74%), the architectural requirement for matching d_ffn dimensions between CP and CIR stages necessitates using cp-4096 for CIR training. This ensures architectural consistency during transfer learning, where the feed-forward network dimensions must align between pretraining and fine-tuning stages for optimal weight transfer.

5.4.3 FINAL MODEL PERFORMANCE

Configuration:
cir-4096 (CP checkpoint: cp-4096, d_ffn=4096, dropout=0.1, lr=2e-5)

Fill-in-the-Blank (FITB) Accuracy:

- Validation: 65.40%
- Test: 64.26%
- Target: 63.73% (CSA-Net benchmark)
- Achievement: **Exceeded target by 0.53%**

Test Methodology:
4-choice multiple selection task with 1 correct answer and 3 distractor items

Performance Analysis:

The test accuracy of 64.26% closely mirrors validation accuracy of 65.40%, with only a 1.14% gap confirming the model learns generalized stylistic rules rather than memorizing training data. The model successfully exceeds the CSA-Net baseline (63.73%) from literature, demonstrating competitive performance despite using a simplified training approach with InBatchTripletMarginLoss instead of explicit hard negative mining.

5.5 BACKEND IMPLEMENTATION

5.5.1 DATA SCRAPPING

The production fashion catalog originates from two Asian e-commerce platforms: **Musinsa and Wconcept**. These platforms were selected based on stakeholder requirements for demographic localization, offering extensive collections of Asian fashion styles optimized for regional body types and aesthetic preferences. The scraping process targeted four product categories (outerwear, tops, bottoms, and shoes) to align with system scope limitations defined in Section 1.5.

Data Collection Process:

Automated scripts scrape public product pages through HTTP requests. The scraper collects structured product metadata including titles, descriptions, brand names, category labels, pricing information, and product URLs. Each product page contains multiple product images showcasing garments from different angles and styled on professional models.

Collection Scale:

The scraping operation accumulated more than 300,000 unique product entries across both platforms. Each product contained an average of 3 product images, generating a total dataset of 900,000 images.

Dual Purpose Dataset:

The scraped data serves two distinct system functions. **Product metadata and isolated garment images populate the global fashion catalog for Smart Discovery search functionality**, where users query items through text descriptions or visual similarity. **Professional model photos displaying visible full body, face, and clothing combinations provide training data for Model 2 biometric compatibility learning** scheduled for FYP 2 development. These images enable automated ground truth generation where user attributes (skin tone, body type, gender) paired with worn garments establish positive compatibility examples for supervised learning.

5.5.2 DATA CLEANING

Raw product data undergoes structured processing to extract relevant attributes, normalize values, and generate optimized text descriptions for embedding models. The cleaning pipeline addresses three objectives: category standardization, attribute extraction, and text description generation optimized for the 64-token constraint of fashion-specific embedding models.

Category Mapping System:

Category Code Prefix	Primary Type	Example Subtypes
001xxx	Top	t-shirt, shirt, hoodie, sweatshirt, knitwear
002xxx	Outerwear	bomber jacket, blazer, coat, parka, windbreaker
003xxx	Bottom	jeans, chinos, slacks, shorts, joggers

100xxx	Dress/Skirt	mini dress, midi dress, maxi dress, mini skirt
017xxx	Active/Sports	sports top, sports pants, sports jacket

Table 13. Musinsa Category Code Mapping

Category Determination Logic:

```
def determine_item_type(goods_name: str, category_code: str) -> Tuple[str, str]:
    """
    Determine item type and subtype from category code and product name.
    Returns:
        Tuple of (item_type, sub_type)
    Example:
        ('top', 'hoodie'), ('bottom', 'jeans'), ('outerwear', 'blazer')
    """
    mapper = MusinsaCategoryMapper()
    # Try category code first (most reliable)
    if category_code:
        item_type, sub_type = mapper.get_type_from_category(category_code)
        if sub_type not in ['top', 'jacket', 'pants']:
            return (item_type, sub_type)
    # Fallback to name matching
    name_lower = goods_name.lower()
    if 'hoodie' in name_lower:
        return ('top', 'hoodie')
    if 'jeans' in name_lower:
        return ('bottom', 'jeans')
    return ('top', 'top')  # Ultimate fallback
```

Attribute Extraction and Normalization:

The extraction process parses product description articles from the goodsDescriptionV2 (product) field, capturing key attributes including fit type, color, material composition, neckline, sleeve length, stretch, thickness, and garment length. Color normalization standardizes spellings (converting "grey" to "gray") and lowercases values for consistent matching. Material compositions exceeding 40 characters undergo truncation, preserving the first listed material when comma-separated.

Text Description Generation:

The generate_product_description function creates concise text optimized for embedding models by selecting only essential attributes.

Example Text Generation:

```
# Input product data
goods_name = "Oversized Cotton Hoodie"
fit = "oversized"
sub_type = "hoodie"
color = "black"
material = "100% Cotton"
features = ['round neckline', 'long sleeve']
# Generated optimized text (≤64 tokens)
text_description = (
    f"{goods_name}. "
    f"{fit.capitalize()} {sub_type} in {color}. "
    f"{material}. "
    f"Round neckline, long sleeve."
)
# Output: "Oversized Cotton Hoodie. Oversized hoodie in black. 100% Cotton. Round neckline, long sleeve."
```

Document Structure:

The create_product_document function assembles all original data and stores it in the metadata of each embedding in Qdrant.

```
{
  "_id": "MUSINSA_1234567",
  "brand": "Brand Name",
  "title": "Product Title",
  "type": "top",
  "sub_type": "hoodie",
  "text_description": "Optimized text for embedding (≤64 tokens)",
  "attributes": {
    "fit": "oversized",
    "color": ["black"],
    "material_composition": ["100% Cotton"]
  },
  "features": {
    "neckline": "round",
```

```
    "sleeve_length": "long sleeve"
  },
  "extended_data": {
    "style_keywords": ["casual"],
    "occasion_text": "Suitable for casual and everyday wear",
    "full_material_composition": "100% Cotton, Imported"
  },
  "pricing": {
    "price": 59.99,
    "currency": "USD"
  }
}
```


5.5.3 BOUNDING BOX DETECTION

Garment isolation from product images uses the YOLOS (You Only Look At One Sequence) object detection model fine-tuned on the Fashionpedia dataset via the valentinafeve/yolos-fashionpedia checkpoint. This preprocessing step extracts bounding boxes around the garment so the embedding process focuses only on the clothing region instead of the background.

Detection Model Configuration:

Configuration	Value
Model	valentinafeve/yolos-fashionpedia
Feature Extractor	hustvl/yolos-small
Device Use	MPS
Precision	FP32 (MPS/CPU)
Confidence Threshold	0.5
Max Images per Product	4

Table 14. YOLOS Object Detection Configuration

Category Mapping for Detection:

Fashion Category	Fashionpedia Class Indices
Top	0 (shirt/blouse), 1 (t-shirt/sweatshirt), 2 (sweater)
Bottom	6 (pants), 7 (shorts), 8 (skirt), 20-21 (tights/stockings)
Outerwear	3 (cardigan), 4 (jacket), 5 (vest), 9 (coat), 12 (cape)
Dress	10 (dress), 11 (jumpsuit)

Table 15. Fashionpedia Category Mapping for Detection

How It Works:

The detector uses an early-stopping strategy that loads up to four images per product, runs YOLOS detection until it finds the first bounding box for the target category, falls back to a full-image box when all detections fail, and then normalizes the final box to a square region so the cropped garment keeps a consistent aspect ratio for embedding.

5.5.4 EMBEDDING AND QDRANT INTEGRATION

The last backend stage generates text and image embeddings for every product and stores them in the Qdrant vector database. Embedding generation uses the Marqo-FashionSigLIP model with automatic device selection (GPU, Apple Silicon, or CPU) and limited CPU threads to keep memory stable.

The pipeline processes products in small batches. For each batch, the system first downloads and crops garment images in parallel using the bounding boxes from the detection stage, then encodes product texts and cropped images in separate mini-batches. The text description combines brand, product title, type, the cleaned description, and key attributes such as color and fit so the text embedding represents both semantic meaning and searchable attributes.

After encoding, the pipeline creates Qdrant points that attach both embeddings to the original metadata. Each point stores:

Field	Description
Vtext	768-dimensional text embedding (cosine distance)

Vimage	768-dimensional image embedding (cosine distance)
payload	Full product metadata, including brand, title, category, image URLs, bounding boxes, attributes, and pricing

Table 16. Qdrant Point Structure for Product Embeddings

To balance throughput and resource usage, the pipeline accumulates products in a buffer, processes around 100 items per flush, and upserts points to Qdrant in small groups of 16. Parallel image downloads with 8 workers overlap network I/O, while small embedding batches keep device memory usage predictable. On a 6 vCPU, 12 GB RAM server this configuration processes one product in about 500 ms on average, which is enough to index roughly 140,000 products in about 19 hours, including both embedding and Qdrant insertion.

5.5.5 SMART DISCOVERY IMPLEMENTATION

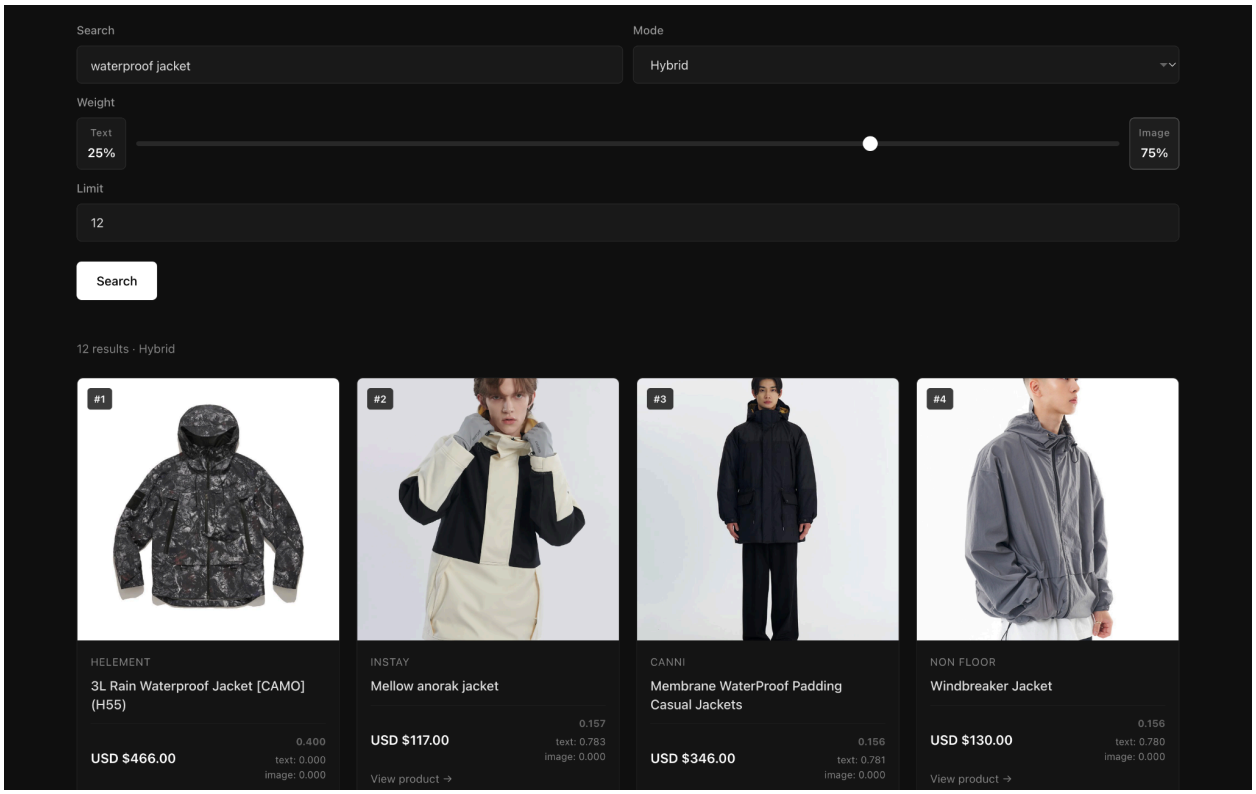


Figure 20. Smart Discovery Website for Weight Testing (https://2ai.dev/training.html)

A simple website is created to experiment with similarity weights before integrating the feature into the app. The site connects to the Qdrant "products" collection and lets the user run manual try-and-test searches on the real catalog.

The website supports two options: **search by text** and **search by image**. For text search, the system encodes the query into a text embedding and combines it with the stored image embedding using a fixed **25 percent text weight and 75 percent image weight**. This 25/75 ratio gives the most consistent and visually relevant results on the catalog, as shown in Figure 20. For image search, the system encodes the query image and uses 100 percent image similarity, searching only on the Vimage vector field with no text contribution.

5.6.6 OUTFIT RECOMMENDATION IMPLEMENTATION

In the current FaBR app, the outfit recommendation screen integrates only Step 1 of the style pipeline, which uses the CIR API from the backend. When the user selects an anchor item (for example, a top) and chooses a target category (for example, bottoms), the app sends the item IDs and category to the /recommend endpoint. The backend responds with a ranked list of bottoms from the user wardrobe, already filtered and sorted by the CIR model. The app renders these items in a scrollable list with product image, brand, title, and distance score, so the user experiences the feature as "select one item, see compatible pieces from my wardrobe." Biometric re-ranking with Model 2 will be added in FYP 2 and is not yet included in this implementation.

5.6 APPLICATION IMPLEMENTATION

Smart Discovery/Search (Partially Finished)

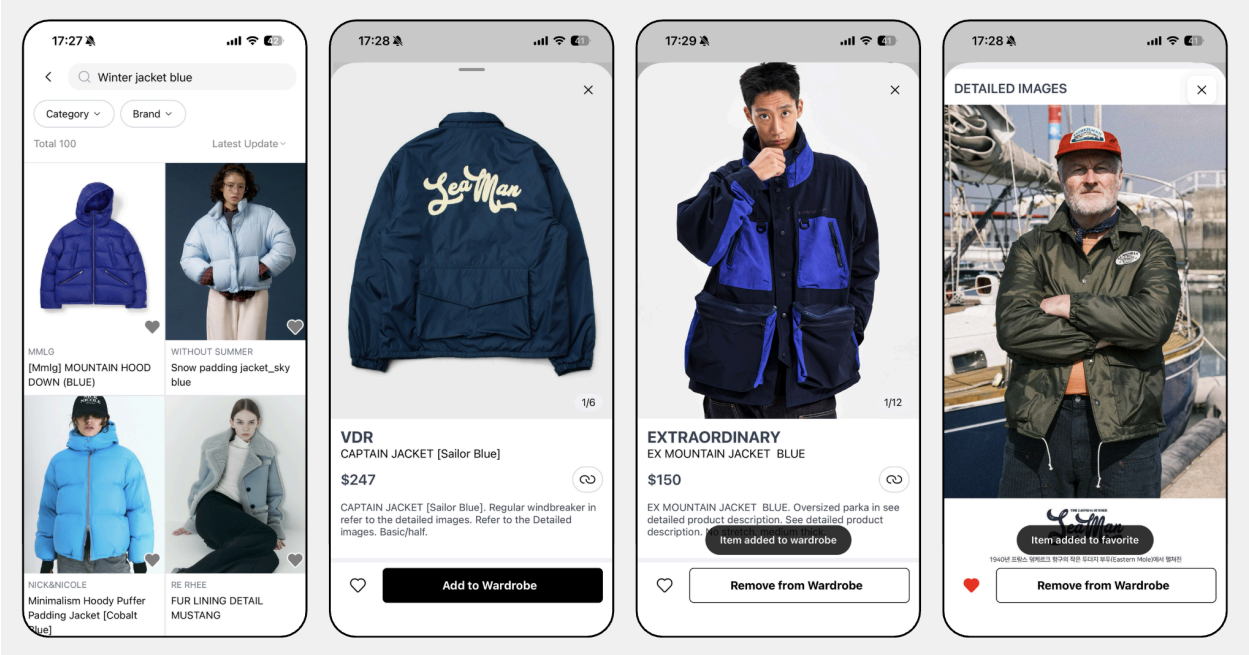


Figure 21. Smart Discovery App Screenshots

Figure 21 shows how "Smart Discovery" works inside the app. Image 1 shows the search output where the app displays a list of products based on the user query. Image 2 shows the product detail view when the user taps one of the items in the list. Image 3 shows the state where

the user adds the selected product into their wardrobe so Model 1 can use it later for outfit recommendation. Image 4 shows the state where the user saves the product into their favourites list for quick access in future sessions.

Wardrobe Management (Partially Finished)

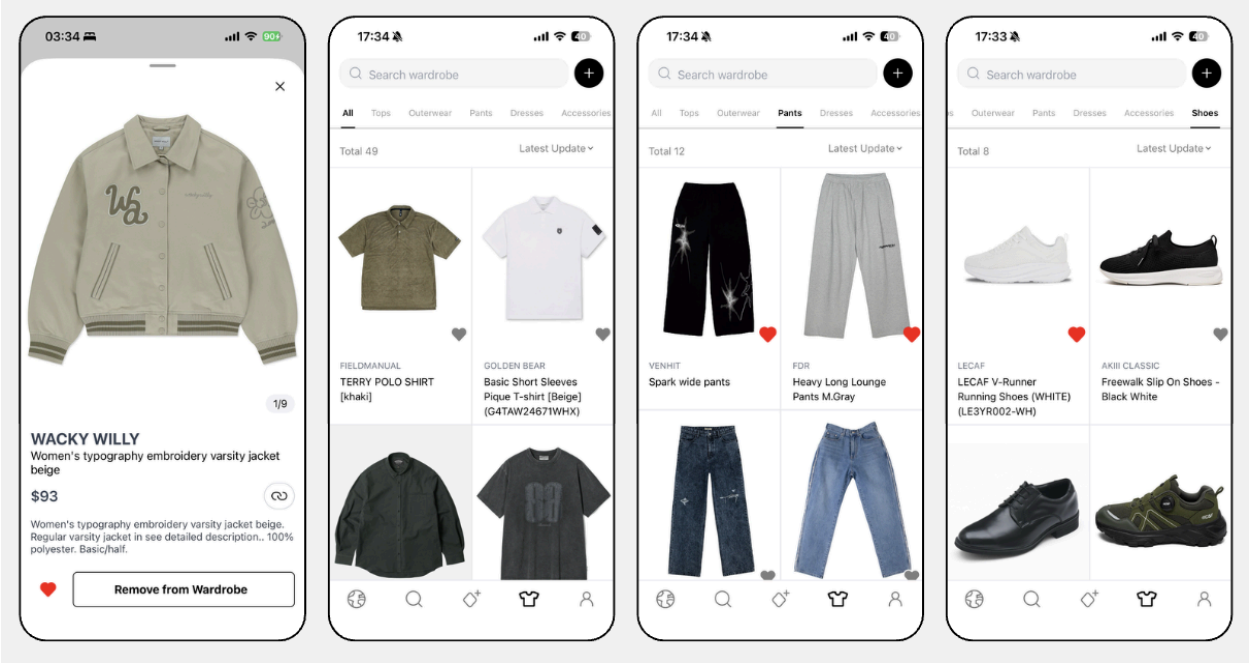


Figure 22. Wardrobe Management App Screenshots

Figure 22 represents the main wardrobe page. Image 1 shows the detailed view of a single wardrobe item where the user can see a larger photo, description, and the option to remove the item from the wardrobe. Image 2 shows the wardrobe list filtered to tops, where each card represents a saved item that can be used as an anchor for styling and recommendations. Image 3 shows the same wardrobe list filtered to pants, and Image 4 shows the list filtered to shoes. This screen focuses on browsing, filtering, and managing saved items, not on generating outfits yet.

Outfit Recommendation (Partially Finished)

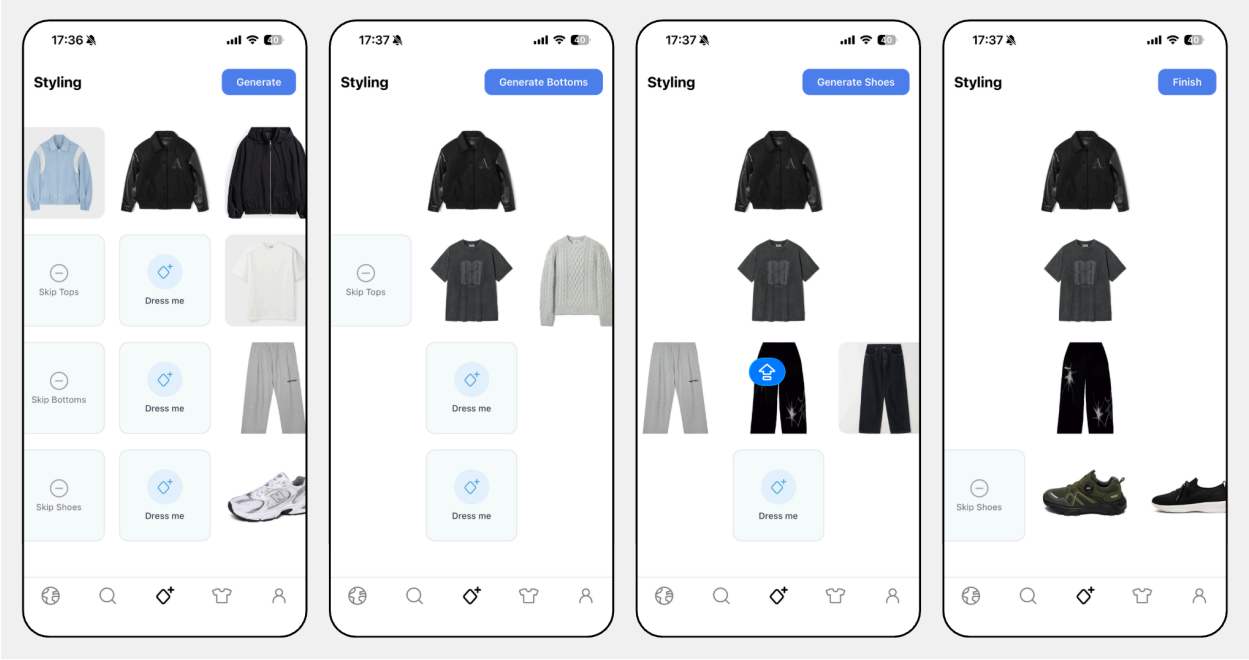


Figure 23. Outfit Recommendation App Screenshots

Figure 23 illustrates the Styling screen that combines wardrobe management with basic outfit recommendation. Image 1 shows an example where only the outerwear slot is filled by the user while the remaining three slots (top, bottom, shoes) are empty and will be generated by the AI. Image 2 shows the result after the app calls the CIR endpoint to generate a compatible top for the selected outerwear. Image 3 shows the next step where the system generates a matching bottom based on the current outfit context. Image 4 shows the final step where the AI suggests suitable shoes so the user ends up with a complete outfit built from their wardrobe plus the recommendations.

Plan for FYP2

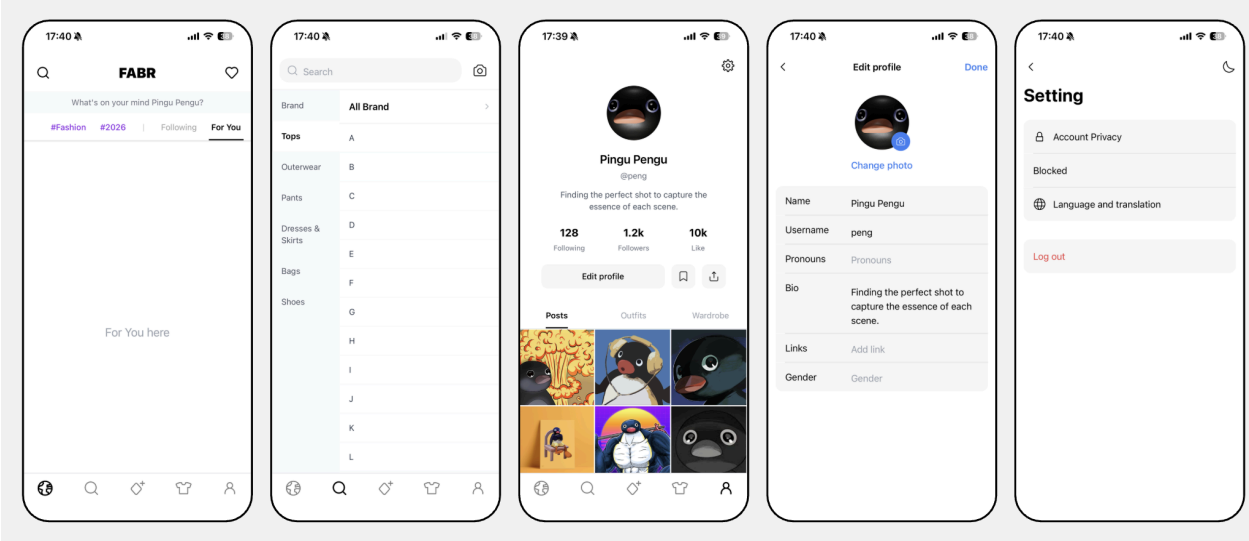


Figure 24. FYP 2 App Screenshots

FYP 2 will be focused on User Authentication, Onboarding, Personalized Collections and Bookmarks, Post creation, Social and Engagement features such as likes and follows, Profile Interactions, and the full implementation of Model 2 for biometric-based re-ranking inside the app.

5.7 APPLICATION DEMO (EXPO GO)

Figure 25. ExpoGo Application Demo QR

CONCLUSION

This project presents FaBR, a fashion recommendation iOS app designed to address the computational and personalization limitations of existing fashion platforms. By integrating complementary item retrieval models with efficient vector search infrastructure, the app enables real-time outfit recommendations without GPU dependencies. The application accepts partial outfit inputs and generates ranked suggestions through precomputed embeddings, eliminating brute-force evaluation methods that cause impractical wait times.

The app was developed using a phased approach. The first phase established core features including smart discovery, wardrobe management, and outfit recommendation functionality. A Complementary Item Retrieval model trained through two-stage transfer learning achieved 64.26% Fill-in-the-Blank accuracy, providing the recommendation engine with competitive performance on CPU infrastructure. Backend services integrate Qdrant vector database for similarity search across 140,000 Asian fashion products and FastAPI endpoints for hybrid text-image queries. The React Native frontend implements camera-based wardrobe upload, semantic product search, and outfit generation from user-selected items. A mobile app concept was prototyped demonstrating core workflows. The next phase extends development through biometric personalization features and complete user flows.

The main contributions of this project include a functional iOS fashion app operating on CPU-first architecture for cost-effective deployment, validated machine learning methodology showing 16 accuracy improvement from compatibility pretraining transferred to retrieval tasks, and production-ready backend separating vector operations in Qdrant from structured data storage for scalability. Future work will focus on biometric re-ranking integration using automated model photo annotation, complete authentication and onboarding implementation, social features for outfit sharing, and field testing with real users. FaBR demonstrates the feasibility of transformer-based fashion AI deployed in mobile applications while maintaining accuracy competitive with GPU-dependent research baselines.

APPENDIXES

APPENDIX A: Repository Links

Model Training Repository: https://github.com/gitrhs/fabr-model1
Model 1 Checkpoints (CP and CIR): model 1 checkpoints
CP Training Logs: https://wandb.ai/rafidaffa2-2ai/outfit-transformer-cp
CIR Training Logs: https://wandb.ai/rafidaffa2-2ai/outfit-transformer-cir
Qdrant Vector Database: https://qdrant.2ai.dev/dashboard#/collections
Database Schema Diagram:
https://lucid.app/lucidchart/10824271-21f3-4816-9a90-38e22f113fd8/edit
API Endpoint Examples

1. **Discovery Search (Text): [link]**
2. **Discovery Search (With Brand Filter): [link]**
3. **Get Item Details: [link]**
4. **Complementary Item Retrieval (CIR): [link]**

APPENDIX B: STAKEHOLDER COLLABORATION

UNIVERSITI MALAYA

04 December 2025

Xquisite AI
Grha XQ, Warung Jati Barat No.15, RT.2/RW.11
South Jakarta, Indonesia 12540

Sir/ Madam,

Collaborator for Academic Project Courses, Faculty of Computer Science and Information Technology, Universiti Malaya

With reference to the above matter,

2. The Faculty of Computer Science and Information Technology (FCSIT) University Malaya is pleased to invite you to be the collaborator for the **Fashion Biometric Recommendation APP Based on Facial Analysis and Body Metrics (FaBR).** This project is under the supervision of **Dr. Lim Chee Kau.**

3. The Academic Project courses are mandatory for final year students of the Faculty of Computer Science and Information Technology as part of their graduation requirements. These courses are designed to nurture academically proficient and technically skilled graduates in the fields of Computer Science and Information Technology. During the course, students actively apply technical expertise, critical thinking, and problem-solving skills to develop innovative products. These projects are intended to meet collaborators' needs and enhance organizational efficiency.

4. This collaboration entails the involvement of the following student(s) from the Faculty of Computer Science and Information Technology working alongside you on the implementation of the aforementioned project. The faculty kindly requests your support in providing the student(s) with necessary information and/or other related resources to facilitate the successful completion of the project.

5. The following are the student(s) who will be involved in this collaboration

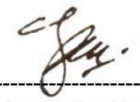
Name	**: Rafi Daffa Ramadhani**
Matric Number	**: S2155407**
Email	**: rafidaffa2@gmail.com**

6. We sincerely hope you will give thoughtful consideration to this invitation. We are confident that this collaboration will bring mutual benefits to both the faculty and your organization. Additionally, we kindly request your written consent to formalize your participation in this

initiative. Should you require any further information, please do not hesitate to contact the project supervisor, **Dr. Lim Chee Kau** at **limck@um.edu.my**.

Thank you.

Yours sincerely,



ProfessorDr. NorLiyanaMohdShuib
DeputyDean (Undergraduate)
Faculty of Computer Science andInformation Technology
Universiti Malaya
Kuala Lumpur

Dr. Lim Chee Kau
Project Supervisor
Faculty of ComputerScience and Information Technology
Universiti Malaya
Kuala Lumpur

Acceptance of Collaboration

I hereby acknowledge that I accept this proposal for collaboration with the Faculty of Computer Science & Information Technology, Universiti Malaya pertaining to the research project entitled **Fashion Biometric Recommendation APP Based on Facial Analysis and Body Metrics (FaBR).**

Signed by Galih Permadi Bahar (GL5155)
Signed on Jan 4, 2026 10:36:31

Collaborator signature

Galih Permadi

Xquisite AI
Grha XQ, Warung Jati Barat No.15, RT.2/RW.11, South Jakarta, Indonesia 12540
Date: 4 January 2026

REFERENCES

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In International conference on machine learning (pp. 1597-1607). PMLR.

Colorful Socks. (2025). Clothing choice decision fatigue statistics. Retrieved from https://bestcolorfulsocks.com/blogs/news/clothing-choice-decision-fatigue-statistics

Cosmopolitan UK. (2016). We spend six months of our working lives deciding what to wear. Retrieved from https://www.cosmopolitan.com/uk/fashion/style/news/a43849/deciding-what-to-wear-six-months

Deldjoo, Y., Schedl, M., Cremonesi, P., & Pasi, G. (2024). Recommender systems for the cultural domain: A survey of the state-of-the-art. Journal of the American Society for Information Science and Technology, 75(2), 234-256.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Bojanowski, P., & Douze, M. (2021). LeViT: A vision transformer in convolutional clothing. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 12259-12269).

Kotouza, M., Pelekis, N., Giannakeris, K., Theodoridis, Y., Markakis, A., Adamopoulos, G., & Sioutas, S. (2020). Personal data assistant: A machine learning framework for fashion product design. In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 1-10). IEEE.

Lin, Y., Nie, X., Zheng, H., Zha, H., & Zhang, Y. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

Lin, Y., Yang, L., Zheng, H., & Zhang, Y. (2020). Putting on my best face: A dynamic attention model for facial expression recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11637-11646).

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence, 42(4), 824-837.

Marqo. (2024). FashionSigLIP: Improving fashion-specific retrieval with multi-modal contrastive learning. Retrieved from https://www.marqo.ai

MarketsandMarkets. (2019). AI in fashion market growth drivers and opportunities. Retrieved from https://www.marketsandmarkets.com/Market-Reports/ai-in-fashion-market-144448991.html

Montalbo, R. (2021). The role of artificial intelligence in fashion recommendation systems. International Journal of Advanced Computer Science and Applications, 12(7), 1-12.

Precedence Research. (2025). AI in fashion market size and forecast 2025 to 2034. Retrieved from https://www.precedenceresearch.com/ai-in-fashion-market

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Krueger, G. (2021). Learning transferable visual models from natural language supervision. In International conference on machine learning (pp. 8748-8763). PMLR.

Sami, L., Farzaneh, A., Reza, H., & Mohammad, R. (2025). Body type recognition and personalized clothing recommendation system using deep learning. Journal of Fashion Technology and Textile Engineering, 8(2), 45-62.

Sarkar, S., Reddy, A., & Vaswani, A. (2023). OutfitTransformer: Learning outfit representations for fashion recommendation. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 123-132). IEEE.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823). IEEE.

Shirkhani, H., Ramasamy Ramamurthy, K., & Geva, S. (2023). A deep learning approach for fashion recommendation. In 2023 IEEE 23rd International Conference on Data Mining Workshops (ICDMW) (pp. 789-797). IEEE.

Smith, L. N. (2018). A disciplined approach to neural network training: the learning rate schedules. arXiv preprint arXiv:1803.09820.

Sohn, K. (2016). Improved deep metric learning via auxiliary loss. IEEE transactions on pattern analysis and machine intelligence, 38(8), 1661-1667.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Zhai, X., Kolesnikov, A., Houlsby, N., & Beyer, L. (2023). Sigmoid loss for language image pre-training. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 11965-11974). IEEE.