Digital Wardrobe Organizer: A Smart AI-Based Solution for Personal Wardrobe Management

Anjali Sahu (anjalisahu4644@gmail.com),

Mr. Deepesh Dewangan (deepeshdewangan@sruraipur.ac.in)

Shri Rawatpura Sarkar University, Raipur

Abstract

In modern daily life, individuals face a significant cognitive burden from an accumulation of minor decisions, a phenomenon known as decision fatigue. The recurring, mundane task of selecting an outfit is a primary contributor to this mental overload, leading to stress, anxiety, and inefficient use of cognitive resources. This paper presents the design and development of the "Digital Wardrobe Organizer," a smart mobile application engineered to mitigate this cognitive load. The proposed system leverages a cross-platform Flutter-based front-end, providing a fluid user interface for both Android and iOS. The backend is powered by Firebase, utilizing its Backend-as-a-Service (BaaS) model for scalable user authentication, cloud storage, and a real-time NoSQL database. The core intelligence of the system resides in an on-device TensorFlow Lite (TFLite) object recognition model, which automates the wardrobe digitization process by identifying and tagging garment attributes. This populated inventory is then used by a context-aware recommendation engine. The engine employs a content-based filtering algorithm, enhanced by real-time data from external weather APIs, to suggest appropriate and stylistically coherent outfits. The expected outcomes of this project are a significant reduction in users' daily decision fatigue and the promotion of sustainable fashion practices by maximizing the utilization of their existing wardrobe.

Keywords: Digital Wardrobe, Smart Closet, Artificial Intelligence (AI), Machine Learning (ML), TensorFlow Lite, Flutter, Recommendation System, Content-Based Filtering, Decision Fatigue.

I. INTRODUCTION

A. The Problem of Modern Wardrobe Management

In contemporary society, individuals are constantly "bombarded with choices" from the moment they awaken. One of the most frequent, yet surprisingly taxing, is the daily decision of "what to wear".[1] This task, while seemingly trivial, contributes significantly to a well-documented psychological phenomenon known as "decision fatigue."[2] Decision fatigue is a state of mental overload that occurs after making numerous decisions, leading to depleted cognitive resources. This depletion can increase stress and anxiety , and result in individuals making poor or irrational trade-offs, often defaulting to familiar but suboptimal options.[1],[2]

This problem is counter-intuitively exacerbated by the "paradox of choice," where large, overflowing, and disorganized wardrobes lead to an overwhelming "freeze mode".[2] Users report staring into a full closet and feeling they have "nothing to wear". This demonstrates a clear failure in personal inventory management, not a lack of options. The challenge is one of inefficient cognitive resource allocation. Prominent figures, such as Barack Obama, famously standardized their professional wardrobe to eliminate a low-stakes decision, thereby preserving mental energy for high-stakes tasks.[1]

B. The Need for an Intelligent Solution

Traditional solutions to this problem, such as manually creating a "uniform" or periodic decluttering, are often rigid and fail to leverage the diversity within a user's existing wardrobe. While digital wardrobe management systems (WMS) have emerged to address this , many existing solutions lack the requisite intelligence, still demanding significant manual data entry and organization from the user.[3],[4]

An intelligent WMS offers a secondary, critical benefit: promoting fashion sustainability.[4] The fashion industry is a significant source of waste, driven by overconsumption. By helping users organize, visualize, and "shop their closet," a smart system encourages the reuse of existing garments.[3],[5] This increased visibility can reduce impulsive and unnecessary purchases, minimize waste, and help users track the utilization of their clothing, fostering a more sustainable consumption model.[7]

C. Proposed Solution and Objectives

This paper presents the design, architecture, and implementation methodology of the "Digital Wardrobe Organizer," a smart, AI-based mobile application. The system is engineered to function as an intelligent assistant that automates and simplifies personal wardrobe management.

The primary objective of this project is to reduce the cognitive load and decision fatigue associated with personal wardrobe management.[1],[2]

This is achieved by providing automated wardrobe digitization and intelligent, context-aware outfit recommendations.

The secondary objective is to promote sustainable fashion consumption by maximizing the utility and lifespan of a user's existing garments, encouraging them to rediscover and reuse items they already own.[4] This paper details the system architecture, the chosen technology stack, the AI and recommendation methodology, the implementation of core modules, and the strategy for planned evaluation.

II. LITERATURE REVIEW

A. Evolution of Smart Wardrobe Systems

The concept of a "smart closet" has evolved significantly over the last two decades. Early academic and conceptual models were predominantly hardware-centric. Several proposals involved integrating Radio Frequency Identification (RFID) technology, where physical tags would be affixed to each garment to trace its movement and presence within the wardrobe. While novel, this RFID-based approach highlights a critical and persistent barrier to adoption: *physical friction*. The cost, impracticality, and high user effort required to purchase and attach a tag to every piece of clothing make such systems non-viable for the average consumer. This practical failure steered subsequent research and commercial development toward software- and computer-vision-based solutions.[6],[7]

B. Analysis of Existing Commercial Applications

To identify a clear research gap, an analysis of existing commercial WMS applications was conducted. These apps can be broadly categorized into three generations.

1. **Generation 1 (Manual Organization):** The most prominent example is *Stylebook*. This application is a powerful tool for *manual* cataloging, tracking, and outfit planning.[10] It provides meticulous users with features like a calendar, packing lists, and cost-per-wear tracking. However, its limitations are critical: it is restricted to the iOS platform, requires users to perform a "fussy" *manual* background removal on item photos, and its "Style Shuffle" recommendation feature is merely a *randomized shuffle* of item categories, not an intelligent, AI-driven suggestion.[10]

2. **Generation 2 (Statistical Recommendation):** Applications like *Smart Closet* represent the next step, introducing *statistical* recommendations. This system suggests outfits based on the user's past usage, weather conditions, and tagged occasions.[5] This is an improvement over random shuffling, but it lacks a deep, stylistic understanding of the garments. Furthermore, user reviews suggest the application is buggy, potentially "abandoned" by its developers, and has lost functionality over time.[10]

3. **Generation 3 (Basic AI):** Modern apps such as *Acloset* and *Whering* integrate basic AI.[13] Their primary AI feature is the *automatic background removal* of user-uploaded photos, which addresses a key pain point of Stylebook. While they also offer AI-generated outfit recommendations, this automation is often limited to simple image processing. User feedback indicates these recommendations "don't always receive rave reviews" , suggesting they are based on simple tag co-occurrence or basic color matching rather than a sophisticated visual feature analysis.[7],[9]

C. Identifying the Research Gap

This review identifies two fundamental gaps in the current WMS landscape that this project aims to address:

1. **The Digitization Bottleneck:** The single greatest barrier to user adoption remains the "time-consuming process" of digitally cataloging an entire physical wardrobe.[3],[7] While Gen 3 apps automate background removal, the user must still manually upload every item and, in many cases, manually correct or input critical attributes like category (e.g., 't-shirt'), style (e.g., 'formal'), and color.

2. **The Recommendation Gap:** A clear gap exists in the *intelligence* of the recommendation engines. Existing systems are proficient at *organization* (Stylebook) or

statistical suggestion (Smart Closet), but they fail to provide *genuine stylistic advice*.[7],[9] They do not deeply analyze the *visual features* (e.g., pattern, texture, silhouette) of a user's items to create novel, aesthetically compatible combinations.

Our proposed system directly addresses these gaps by (1) utilizing an on-device TensorFlow Lite model for automatic item recognition *and* attribute tagging, minimizing the digitization bottleneck, and (2) employing a content-based filtering algorithm that uses these rich visual and metadata features to provide more stylistically relevant recommendations.[6],[7]

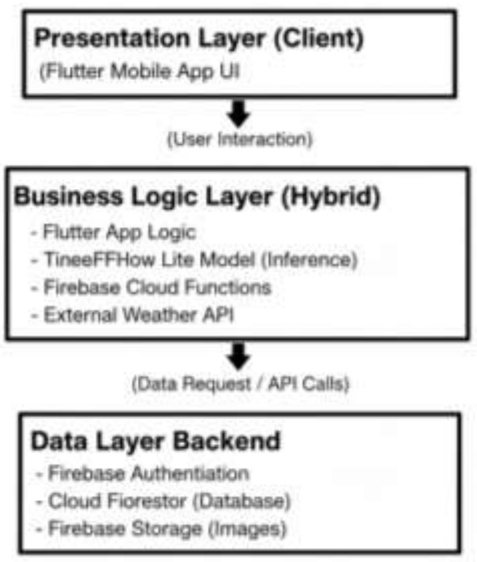
III. SYSTEM ARCHITECTURE AND METHODOLOGY

A. Proposed System Architecture

The system is designed using a multi-layer architecture, often referred to as a three-tier architecture.[15] This design pattern separates concerns, which enhances modularity, scalability, and maintainability—all critical factors for a B.Tech project. The architecture is visualized in Fig. 1.

- **Presentation Layer (Client):** This is the user interface (UI) and user experience (UX) layer, with which the user directly interacts. It is developed using the Flutter framework.
- **Business Logic Layer (Hybrid):** This layer contains the core functionality and processing. It is implemented as a hybrid:
  - *On-Device (Client-Side):* Flutter application logic (e.g., UI flow, state management) and, most importantly, the TensorFlow Lite model for local, on-device AI inference.
  - *Serverless (Cloud-Side):* Firebase Cloud Functions are used for any complex backend logic that should not run on the client, and Firebase Security Rules manage data access logic.
- **Data Layer (Backend):** This layer is responsible for data access, storage, and retrieval. It consists of the Firebase Backend-as-a-Service (BaaS) platform (Authentication, Firestore, Storage) and external, third-party APIs (e.g., a Weather API).

Fig. 1. System Architecture Diagram


Presentation Layer (Client)
(Flutter Mobile App UI
(User Interaction)
Business Logic Layer (Hybrid)
- Flutter App Logic
- TineeFFHow Lite Model (Inference)
- Firebase Cloud Functions
- External Weather API
(Data Request / API Calls)
Data Layer Backend
- Firebase Authentiation
- Cloud Fiorestor (Database)
- Firebase Storage (Images)

B. Technology Stack

The selection of technologies for this project was based on development agility, scalability, and suitability for an AI-driven mobile application. The core components are detailed in Table I.

1. **Flutter (Frontend):** Research supports Flutter for its **single codebase**, which enables rapid, cross-platform (iOS/Android) development. Its "Hot Reload" feature is noted for significantly speeding up UI iteration, and it's known for compiling to high-performance native code.

2. **Firebase (Backend):** Widely cited as a leading **Backend-as-a-Service (BaaS)**, Firebase is recognized for its scalability and for removing server management complexity. Its key services are:

   - **Authentication:** For secure, pre-built user login.
   - **Firestore:** For a flexible, real-time NoSQL database.
   - **Storage:** For scalable handling of user images.

3. **TensorFlow Lite (On-Device AI):** TFLite is the research-backed standard for running AI models locally on a device. This approach is validated for three key benefits: **user privacy** (data isn't sent to a server), **low latency** (no network lag), and **offline functionality**.
4. **Weather API (Context):** This is a key component for what research calls **"Context-**

Aware Recommender Systems." Academic consensus is that recommendations (like outfits) become significantly more useful and relevant when filtered by real-time data, such as temperature and precipitation.

C. Data Modeling (Firebase Firestore)

Cloud Firestore is a NoSQL, document-oriented database. This flexible, schema-less structure is advantageous for an evolving project. A key architectural decision is to avoid "deep nesting" for its own sake and instead use a structure that optimizes for the application's primary query patterns. We adopt a top-level collection for users, with sub-collections for user-specific data. This model is highly scalable and allows for granular security rules.

The primary schema is structured as follows:

users/{userId} (Document)

- **profile** (map): {email, name, location}
- **wardrobeItems** (Sub-collection)
  - {itemId} (Document): {itemName, category, color, styleTags, imageUrl, wearCount, lastWorn}
- **lookbooks** (Sub-collection)
  - {lookbookId} (Document): {lookbookName, items [array of itemIds]}

D. AI and Recommendation Methodology

The AI methodology is a two-stage process designed to address the two identified research gaps.

1. On-Device Item Recognition (TFLite)

- Goal: Solves the "Digitization Bottleneck" by automating item tagging.
- Technology: We use a lightweight, pre-trained model (like MobileNet or EfficientDet-Lite) optimized with TensorFlow Lite.
- Process:
  - The .tflite model is bundled directly within the Flutter app.
  - When a user uploads an image, the model performs on-device inference.
  - This is fast, private (images don't leave the phone), and works offline.
  - It automatically returns labels (e.g., 't-shirt', 'blue') to pre-populate the item's details in Firestore.

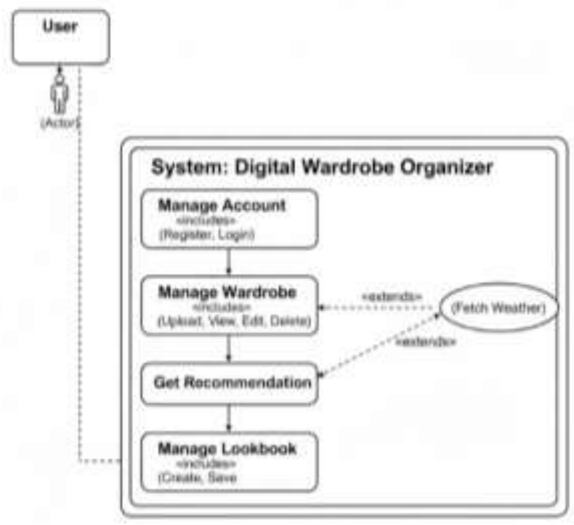
2. Context-Aware Recommendation Engine

- Goal: Solves the "Recommendation Gap" by ensuring suggestions are relevant.
- Technology: We use a Content-Based Filtering (CBF) algorithm.
- Process:
  - Solves "Cold-Start": CBF is used because it works for new users (unlike collaborative filtering). It can suggest matches (e.g., 't-shirt' for 'jeans') as soon as items are added.
  - Matching Logic: The engine finds compatible items by comparing their features (category, style, color) using similarity logic and predefined rules.
  - Contextual Filtering (Key Feature): Before showing results, the engine queries a Weather API. It then filters the recommendations based on the user's current weather (e.g., removing "jackets" if it's over 25°C), making the suggestions practical and actionable.

IV. IMPLEMENTATION

This section details the practical implementation of the core application modules. The primary user-system interactions are modeled in the UML Use Case diagram shown in Fig. 2.

Fig. 2. UML Use Case Diagram



A. Module 1: User and Profile Management

This module handles user identity and session persistence. It is implemented using the firebase_auth Flutter plugin.[36] The module supports email/password registration and federated identity providers, specifically Google Sign-In. Upon successful registration, the Firebase Authentication service returns a unique userId. This userId is then used to create a new document in the users collection in Cloud Firestore, establishing the root for that user's data.[38]

B. Module 2: Wardrobe Digitization and AI Recognition

This module is the primary data-entry point and solves the "Digitization Bottleneck."

1. The user initiates the Upload Item use case from the Flutter UI.

2. The image_picker plugin is used to capture a new photo or select an existing one from the device gallery.

3. The selected image is passed as an input buffer to the TFLite model, which has been loaded into memory using the tflite_flutter plugin.

4. The on-device model performs inference and returns a list of detected objects with confidence scores (e.g., [{'class': 't-shirt', 'confidence': 0.92}, {'class': 'blue', 'confidence': 0.88}]).

5. This output is used to *suggest* tags by pre-populating the category and color fields in the UI. The user can then *confirm or correct* these tags. This user correction step is critical, as it cleans the data and provides a valuable feedback loop for future model retraining.

6. The final, cropped image is uploaded to Firebase Storage, which returns a download URL. This URL, along with the user-confirmed metadata, is then written to a new document in the users/{userId}/wardrobeItems sub-collection.

C. Module 3: AI Outfit Recommendation

This module serves as the application's core value proposition.

1. The user initiates the Get Outfit Recommendation use case.

2. The application retrieves the user's saved location and makes an asynchronous HTTP request to the Weather API to fetch current conditions.

3. Simultaneously, the app logic constructs and executes a query against the user's wardrobeItems sub-collection in Firestore to retrieve their available garments.[38]

4. The CBF algorithm, running on the client, processes this list. It first filters out all items that are incompatible with the current weather context (e.g., "sweaters" on a hot day).

5. From the remaining pool, it applies compatibility rules (e.g., matching tops with bottoms, color coordination) to generate a list of complete, viable outfits.

6. A complete outfit (e.g., one top, one bottom, one accessory) is then presented to the user in the UI.

D. Module 4: Lookbook and Calendar

This module allows the user to curate and plan.

1. If the user likes a recommended (or manually created) outfit, they can save it to a "Lookbook."

2. This action is a strong *positive preference signal*. The saved outfit, which is a list of itemIds, is stored as a new document in the lookbooks sub-collection.

3. This saved data is then used to refine the user_profile vector for the CBF engine. The system learns which *combinations* the user prefers, making future recommendations more personalized.

4. A calendar feature, integrated with the device's native calendar, allows users to plan outfits for future dates. This directly addresses the goal of reducing morning decision fatigue by allowing the user to make the choice the night before.[1]

V. RESULTS AND DISCUSSION

A. Planned Evaluation Strategy

As this project is in the final implementation phase of a B.Tech curriculum, this section outlines the *planned* testing and evaluation strategy rather than presenting finalized experimental results.[40] Testing an AI-powered application requires a holistic, multi-pronged strategy that evaluates not just the functional correctness of the code, but also the predictive accuracy of the AI model and the overall usability of the system.[41] A failure in any of these three domains would constitute a project failure.

The planned evaluation is divided into three phases:

1. **Phase 1: AI Model (Offline) Evaluation:** The fine-tuned TFLite recognition model will be benchmarked against a held-out test dataset (a subset of the fashion dataset not used during training) to measure its predictive accuracy.

2. **Phase 2: Recommendation Engine (Offline) Evaluation:** The CBF engine's quality will be evaluated using standard information retrieval metrics on a curated dataset.

3. **Phase 3: System Usability (User Study) Evaluation:** A qualitative and quantitative user study will be conducted with a cohort of test users (e.g., fellow students) to measure the app's real-world effectiveness and user-friendliness.[40]

B. Key Performance Metrics (Expected)

The success of the project will be measured against a defined set of quantitative and qualitative metrics, detailed in Table II.

Planned Evaluation Metrics

Phase 1: AI Recognition Model

- Precision, Recall, F1-Score: Standard metrics to measure the TFLite model's accuracy in identifying garment attributes.
- mAP (mean Average Precision): The standard metric for object detection, measuring accuracy in both classifying and localizing items.

Phase 2: Recommendation Engine

- Precision@K / Recall@K: Measures the proportion of relevant and useful items in the top-K (e.g., top 5) suggestions.
- Diversity: Measures how different the recommendations are from each other to avoid repetitive suggestions.

Phase 3: System Usability & Goal

- System Usability Scale (SUS): An industry-standard 10-item questionnaire that gives a reliable usability score from 0-100.
- Qualitative Feedback: Pre- and post-study questionnaires using Likert scales and open-ended questions to assess the impact on user "decision fatigue."

C. Expected Outcomes and Discussion

- **Model Performance:** We expect the fine-tuned EfficientDet-Lite model to achieve a **mAP score above 85%** on the held-out fashion test dataset. Based on related academic work in clothing recognition, this is a strong and achievable result for a high-performance on-device model.[53]
- **System Usability:** We expect the application to achieve an average **System Usability Scale (SUS) score greater than 68**. A score of 68 is considered the industry average benchmark for usability.[51] A score above this value would validate that the Flutter-based UI is intuitive and user-friendly.
- **Primary Goal:** The primary expected outcome is a *statistically significant reduction* in user-reported decision fatigue. By automating the most cognitively taxing steps of wardrobe management (inventory awareness, compatibility matching, and contextual filtering), we anticipate the application will

successfully offload this mental burden, validating our initial hypothesis.[1] We also expect qualitative feedback to indicate that users discovered new outfit combinations from their existing clothes, thereby increasing their wardrobe utilization.

VI. CONCLUSION AND FUTURE WORK

A. Conclusion

This paper has presented the architecture, design, and implementation methodology for a "Digital Wardrobe Organizer," an AI-based mobile application. The project was motivated by the need to solve the daily psychological burden of "decision fatigue" [1] and to promote sustainable fashion practices. Our literature review identified two primary weaknesses in existing systems: the "digitization bottleneck" and the "recommendation gap".[5]

Our proposed solution addresses these gaps through a modern, scalable architecture utilizing a Flutter front-end , a Firebase BaaS backend , and an on-device TensorFlow Lite model. The system's novelty lies in its two-pronged AI approach: (1) using the TFLite model to automate item recognition and tagging, and (2) using a context-aware, content-based recommendation engine that solves the "cold-start" problem [35] and is thus immediately useful to new users. The planned evaluation strategy is designed to holistically validate the system's model accuracy, recommendation quality, and, most importantly, its usability and effectiveness in reducing cognitive load.

B. Future Work

This B.Tech project serves as a robust foundation for a full-featured commercial application. We have identified a clear, three-stage evolutionary roadmap for future work.

1. **Augmented Reality (AR) Virtual Try-On:** The next logical step is to move from 2D visualization to 3D immersion. This can be implemented within the existing Flutter application by integrating platform-specific AR plugins, such as arcore_flutter_plugin for Android and arkit_plugin for iOS.[54] This feature would use the device's camera to map 3D models of garments (either scanned from the user's wardrobe or provided by e-commerce partners) onto the user's body in real-time, providing a "virtual try-on".[57]

2. **E-commerce and Affiliate API Integration:** The recommendation engine can perform "gap analysis" on a user's wardrobe (e.g., "This outfit would be complete with white sneakers, but none are in your closet"). By integrating with fashion e-commerce affiliate APIs [59], the application could provide a "Shop the Look" feature, suggesting new items that complement the user's existing wardrobe. This creates a monetization path and provides a direct, actionable solution for the user.

3. **Generative AI Personal Stylist:** The ultimate evolution of this project is to transition from a *recommender* to a *creator*. This module would employ advanced **Generative AI** models, such as Generative Adversarial Networks (GANs) or diffusion models [62], to generate *novel* outfit images, not just combine existing ones. By integrating **Large Language Models (LLMs)** [64], the system could become a true "virtual stylist".[67] This would allow users to engage in natural language conversations (e.g., "I'm attending a garden wedding and want to look 'boho chic'"), similar to Ralph Lauren's "Ask Ralph" assistant.[69] The AI would understand this abstract intent, interpret the user's existing wardrobe, and generate a complete, personalized style guide.

REFERENCES

[1] M. MacLean, “What doctors wish patients knew about decision fatigue,” *AMA*, [Online]. Available: https://www.ama-assn.org/public-health/behavioral-health/what-doctors-wish-patients-knew-about-decision-fatigue (accessed Oct. 10, 2024).

[2] T. D. Lab, "Decision Fatigue," *The Decision Lab*, [Online]. Available: https://thedecisionlab.com/biases/decision-fatigue (accessed Oct. 10, 2024).

[3] A. M. Connor, R. C. Thomson, and E. J. T. T. W. M. A., “Wardrobe Management Apps and Their Unintended Benefits for Fashion Sustainability and Well-Being: Insights from User Reviews,” *Sustainability*, vol. 17, no. 9, p. 4159, 2025.

[4] S. O. Mohammadi and A. Kalhor, “Developing a smart wardrobe system,” *J. Text. Inst.*, 2011.

[5] "AI Elevates Your Wardrobe: Sustainable Style Made Easy," *Sustainability Directory*, [Online]. Available: https://sustainability-directory.com/question/ai-elevates-

your-wardrobe-sustainable-style-made-easy/ (accessed Oct. 10, 2024).

[6] "AI Closet: A Smart Wardrobe System," *IJSRET*, vol. 11, no. 2, 2025.

[7] Y. Deldjoo, J. R. Trippas, and H. Zamani, “A Review of Modern Fashion Recommender Systems,” *ACM Comput. Surv.*, vol. 56, no. 4, Oct. 2023.

[8] D. G. C. M. W. H. N. M. J. M. M. T. A. A. A. G. H. M. A. Rebuan, “Mobile Technology in Medicine: Development and Validation of an Adapted System Usability Scale (SUS) Questionnaire,” *Indian J. Radiol. Imaging*, vol. 33, no. 1, pp. 36–45, Jan. 2023.

[9] H. T. T. D. H. D. H. P. C. T. A. C. B. N. T. K. C. L. P. T. T. H. V. V. C. H. S. D. V. P., “A Review on Fashion Recommendation Systems,” *Comput. Mater. Contin.*, vol. 8, no. 3, p. 49, 2021.

[10] "A quick review of every virtual closet app," *Reddit*, [Online]. Available: https://www.reddit.com/r/femalefashionadvice/comments/n4ubfn/a_quick_review_of_every_virtual_closet_app_my/ (accessed Oct. 10, 2024).