BSc Artificial Intelligence and Computer Science
2455269

A systematic approach to improving outfit suggestions

by

Hasan Shariff.

School of Computer Science
College of Engineering and Physical Sciences

Final Year Project Dissertation

Supervisor: **Rami Bahsoon**
Consultant: **Michael Oakes**

University of Birmingham 2024-25

I declare that this dissertation, which has been completed as part of my Undergraduate studies at the University of Birmingham, has been completed independently. All relevant resources and information sources have been listed.

University of Birmingham, 17.05.2025 Hasan Shariff

Word Count:

Contents

1 Acknowledgments

I would like to thank...

2 Abstract

3 Key Words

This is a list of key words and terminology used throughout this paper.

1. UI - User Interface

2. UX - User Experience

4 Introduction

4.1 Overview

The primary objective of this project is to generate outfits for users based on what they have in their wardrobe. The user will be able to take photos of items in their wardrobe and upload them to the system's database. Once the user is happy with the images that they have uploaded then they can generate outfit based on what they have.

4.2 Introduction

The fashion industry is one of the fastest growing sectors in the world with it currently representing 1.65% of the global gross domestic product. This industry is expected to see a compound annual growth rate of 2.81% between 2025 and 2028 [1]. Driven by the rise of e-commerce couple with social media platforms such as Instagram and TikTok these platforms have revolutionised access to the latest clothing trends and items. While on the surface, this scale of accessibility appears to be a substantial benefit. In reality this presents a considerable challenge and concerning patterns of over consumption and wardrobe underutilisation. This claim was supported by a study conducted in 2022 by the UK charity WRAP. WRAP revealed that at least one quarter, 26-31% of the average person's wardrobe remains unworn for at least 1 year [2]. This claim was further supported by the UK based retailer Marks and Spencers who further confirmed this issue. They found that in the average wardrobe which consists of 152 items only 44% of those items are worn regularly [3]. This widespread underutilisation demonstrated by both studies suggests that is a significant problem representing significant environmental challenges through increased carbon footprints, economic waste and textile accumulation.

This project will aim to address this challenge by developing an intelligent outfit recommendation system with the main purpose to help the user rediscover, maximise and optimise their existing wardrobe. The proposed system will leverage several different domains of artificial intelligence and machine learning to aid in the generation process of outfits based on the users wardrobe while keeping the main aim of maximising the user's wardrobe. By aiming to consistently achieve this aim project seeks to reduce unnecessary purchases and their associated environmental impact while working towards the goal of

maximising the potential of the user's wardrobe.

This proposed system will leverage several artificial intelligence capabilities including object detection for accurate classification of the item, background removal for clean image processing, feature extraction to identify and utilise the key features of the item of clothing and finally adaptive learning to refine recommendations based on user preferences and feedback. These technologies work hand in hand with each other to create a comprehensive proposed solution that intelligently analyses the user's wardrobe to suggest optimal outfit combinations.

This report will detail the information about the motivations behind the project and how the project was managed over time. In addition to this the report will also explore the fundamental system, requirements necessary and the design principles and decisions taken. This report will also explore the implementation of different artificial intelligence aspects as well as the user centric design. Furthermore it will delve into how all of the different aspects tie together, finally culminating in the demonstration of the success of achieving the main principle and aim of maximising and fully utilising the user's wardrobe.

5 Literature Review and Research

5.1 Object Detection and Image Classification

The object detection algorithm presented by Lao and Jagadeesh in their paper presents a comprehensive CNN-based framework for fashion classification and object detection across different domains of challenges. This implementation demonstrates the effectiveness of CNNs in fashion classification with a validation accuracy of 93.4% [4]. However, the methodology exhibits some limitations, such as struggling to accurately define two classes which are difficult to visually distinguish. This implementation also predominantly uses controlled datasets, which are not comparable to real-world scenarios where insufficient lighting, poor image quality, or complex backgrounds predominate. This work does, however, provide valuable methodological insights for implementing object detection within fashion applications.

Feng et al.(2018) presented an object detection implementation using the YOLOv2-opt system which is an enhancement on the YOLOv2 architecture. The demonstrated system achieves high levels of accuracy and precision while also maintaining its speed. The model was trained on a dataset containing five categories (trousers, skirts, coats, T-shirts and bags) which varied in detection performance. The model performs better on items with more well-defined borders which results in higher precision 93.5% [5]. Despite these promising results, this model does however present some limitations such as misidentifying objects with more complex backgrounds. This includes the model thinking a dark region within an image was a bag. The authors suggest that "the model needs to be enhanced in processing complex images". Another limitation in this paper is the relatively small dataset as a significant constraint. This research demonstrates the potential of deep learning approaches for fashion detection.

5.2 Optimised Based Matching System

Cross and Hancock present a methodologically sophisticated framework through stochastic optimisation which offers valuable insights for outfit recommendation systems. Optimisation-based matching can be applied to outfit recommendation as it allows for discovering globally optimal combinations in highly complex style compatibility spaces

where stable matching fails to capture nuanced fashion relationships. The authors, Cross and Hancock, compare multiple stochastic optimisation strategies such as genetic algorithms. Their genetic algorithms achieve "rapid and uniform convergence to a global optimum" [6] whereas deterministic methods become trapped in local optima. These findings by Cross and Hancock suggest that optimisation-based matching has profound implications for outfit recommendation systems where it avoids inconsistent suggestions. Furthermore, the genetic approach in this paper maintains the use of weighted matching configurations which draws parallels with multiple style combinations with varying degrees of compatibility. This paper was initially intended for aerial imagery matching however the optimisation framework can be adapted to model relationships between different items of clothes where, similar to a decision tree nodes represent items and edges denote style compatibility. This ultimately results in more robust and globally optimal outfit suggestions.

Mills-Tettey et al. present the dynamic Hungarian Algorithm which is an advancement on the standard implementation of the Hungarian algorithm formulated by Kuhn-Munkres. This dynamic approach efficiently aims to solve and repair existing solutions achieving $O(kn^2)$ complexity where K is the number of modifications. The authors prove that their solution achieves optimality faster through empirical testing that their method runs "orders of magnitude more efficiently" [7]. This implementation also presents promise to optimising outfit suggestions where outfit compatibility scores may fluctuate depending on situational changes such as an alteration to the wardrobe. This can then be applied to outfit suggestions where each node in the Hungarian algorithm represents an individual item and the edges represent compatibility. This optimisation framework set out by Mills-Tettey allows for efficient and responsive outfit recommendation systems.

5.3 Adaptive Learning for Outfit Recommendations

Majeed et al's reinforcement learning implementation for personalised clothing recommendations establishes feasibility for personalised outfit recommendations via real time user interaction mechanisms. The research aims to refine the recommendation process over time but also increase user satisfaction with more personalised outfits. The authors want to ensure that the clothes that are recommended cater to the individual and

their style preferences based on the responses from the user [8]. Despite achieving notable accuracy this implementation does exhibit some limitations, such as using a specific dataset which encompasses only a limited set of pants and shirts. The experimental design also predominantly emphasises parameter optimisation rather than incorporating multiple user attributes such as preference hierarchies or gender specific considerations. Another limitation resides in the absence of user acceptance testing leaving the system unevaluated in real world applications. Despite these challenges this paper does lay the foundations for using reinforcement learning and adaptive learning in recommendation domains.

The machine learning based outfit recommendation system developed by Kokane et al leverages a comprehensive dataset of clothing items while also incorporating body-shape analysis [9]. This system does present promise in increasing personalised fashion recommendations using several domains of artificial intelligence. While the system does demonstrate promise in personalising outfit recommendations there some limitations that should be addressed. Firstly the dataset struggles in diversity and representativeness which may skew the system. In addition the implementation could be susceptible to ethical issues as body shape analysis requires the use of biometric data and the report does not mention how this information is stored and protected. This research does suggest that a machine learning approach represents a significant advancement for personalised fashion recommendation systems.

5.4 Existing Theoretical Works

Stan and Mocanu's implementation of an intelligent personalised fashion recommendation system utilises dual convolutional networks which are used for object classification and attribute identification. This implementation proposes a scoring based matching system which takes into account the compatibility of the proposed outfit. It also strongly considers user preferences which should dynamically update over time. The system uses both of these dynamic scores to then produce an outfit. This implementation is very accurate when it comes to precise image classification however the system has some significant limitations. The system is firstly limited by the two layer convolutional network as the process becomes computationally expensive requiring two large sets of weights to

be added. These weights result in a 10 second time minimum when adding a new item. This extended processing time makes the implementation difficult to implement in the real world [10]

De Divitiis et al. presents a style based outfit recommendation system which builds on Kobayashi's work to further improve outfit recommendations. This theoretical framework incorporates a two stage implementation with a style classifier which is trained on colour triplets which is then used for a network based outfit recommendation system. This work does present significant accuracy resulting in 97% accuracy over 60 iterations. Despite this unique approach to outfit recommendations the system is limited by the imbalanced dataset which it uses [11].

Attaluri and Lolla present Style-AI which is a deep learning outfit generator, this is a novel outfit system which utilise constructive learning to generate complete outfits. This theoretical framework employs dual embedding spaces the item space and the outfit space. Style-AI is capable of recognising items of clothing with a great deal of accuracy. The system however is faced with some limitations due to over fitting particularly with larger batch sizes. Attaluri and Lolla do note that the traditional accuracy metrics may undervalue the performance of the system. Style-AI does however recommend visually appealing and coherent outfits consistently [12]

5.5 Existing Technological Works

While there is an adequate amount of research on intelligent outfit recommendation systems, there are relatively few digital applications in actuality. These existing applications primarily server as digital wardrobe organisers or tools to enhance e-commerce, however fail to utilise the advanced methods discussed in the literature review. Examples of this include Uniqlo's implementation which lets users combine items from new collections to create outfits manually. On the other hand, Fitted-AI enables users to upload images of their clothing and then randomises outfit suggestions. These implementations demonstrate there is significant scope for technological approaches to outfit recommendations.

5.5.1 Outfit Recommendation for E-Commerce

Uniqlo is one of the world's largest retailers and in order to enhance the e-commerce experience they have implemented an outfit generation system. This system allows users to create outfits manually by selecting items from Uniqlo's website which are in the latest collections. Then based on what the user selects the system will then create an outfit for the user. This is a very simple implementation however, it gives the user full control over how the outfit is generated. This implementation is limited by not fully leveraging artificial intelligence in addition to not having an affective matching system. The dataset used is also very limited as it only focuses on clothes which Uniqlo's sell rather than using items from the user's wardrobe. This implementation serves as a baseline for other methods as there is minimal algorithmic assistance.

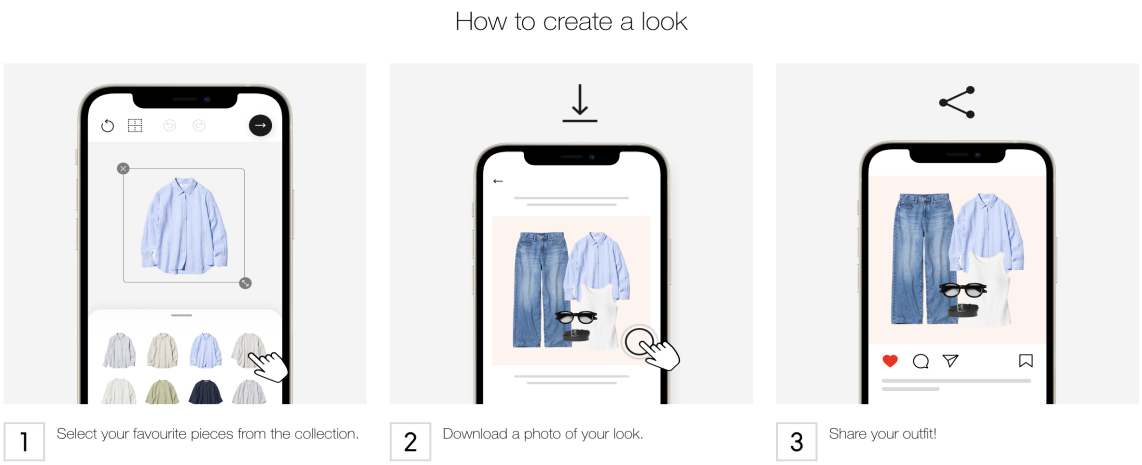

Figure 1: Uniqlo Workflow [13]

5.5.2 Outfit Recommendation for mobile apps

Fitted-AI represents a mobile application implementation for outfit recommendation systems which apply artificial intelligence techniques discussed in the literature review. The app does employ an object detection model to classify images uploaded by the user and it does also automatically label the images. Users are also able to generate outfits using the "create a fit" button which randomly selects four items of clothing across different categories (headwear, top, trousers and shoes). However the implementation does exhibit significant limitations. Firstly, the system lacks any form of robust validation mechanisms, this leads to the system failing to verify if the uploaded images actually contain

items of clothing leading to non-clothing items being used In outfit generation. Furthermore Fitted-AI heavily relies on simple randomisation rather than style compatibility algorithms, this limits the user from expressing individuality and style preferences. In addition to this the practical utility of the app is limited as the application limits the wardrobe size before implementing a pay wall. Despite these limitations Fitted-AI does demonstrate the viability of mobile-based outfit recommendation systems.

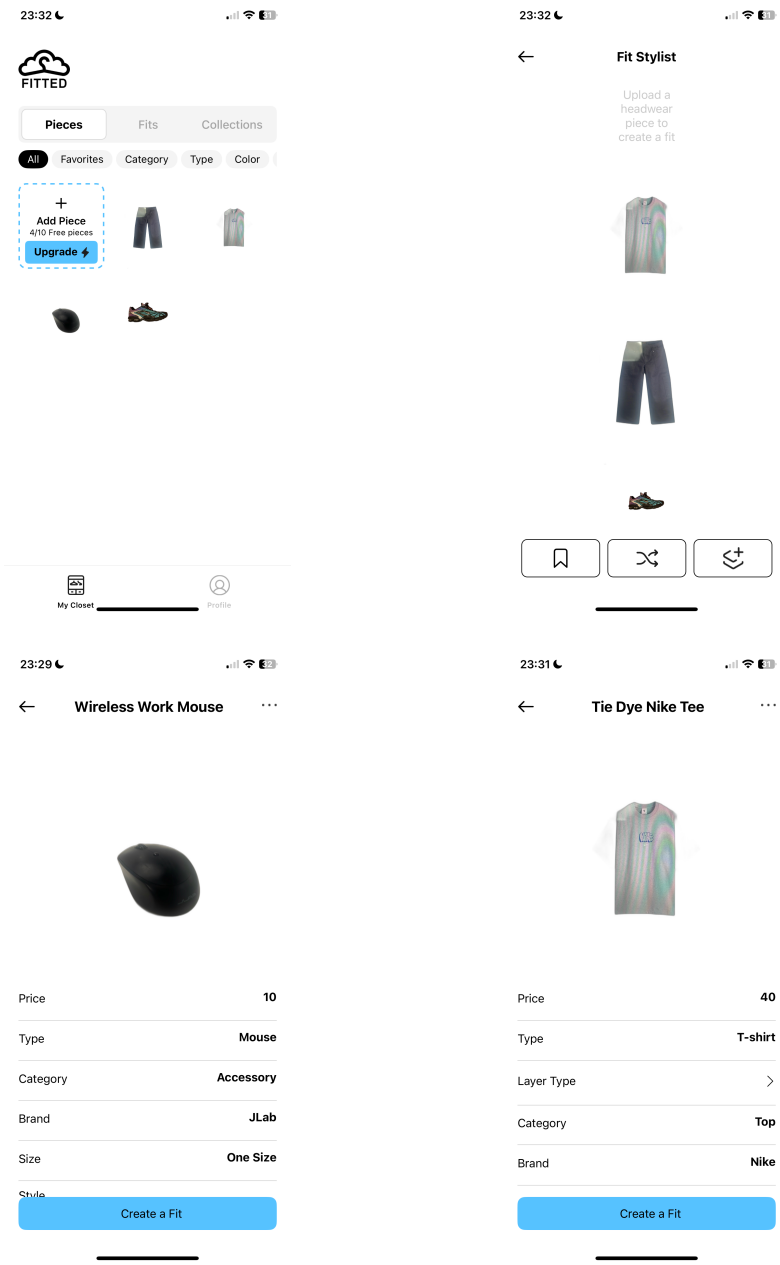

Figure 2: Fitted-AI [14]

5.6 Research Summary

After conducting the necessary research into the key technologies, problems and potential solutions it is evident that there is a lack of substantial existing solutions for intelligent outfit recommendation systems.

The theoretical works examined demonstrate that there is sustenance for outfit recommendation from a theoretical stand point. These papers provide valuable insights for my proposed system as it shows the importance of balancing recommendation accuracy with computational efficiency. While these papers present solid groundwork there are still some inconsistencies between theory and practical implementation

Investigating more thoroughly into the relevant technologies such as object detection, optimisation frameworks, computer vision and adaptive learning suggest to me that these technologies are fundamental to the success of this proposed system. Furthermore analysis of existing applications mentioned in the literature review demonstrate a significant gap between depth of academic research and practical implementation. This suggest that my proposed intelligent outfit recommendation system which uses optimisation matching algorithms represents meaningful advances over current solutions.

To conclude the research emphasises the need to develop a robust outfit recommendation system with sophisticated object detection capable of operating in unconventional environments, optimisation based matching with personalised mechanisms and a clear and attractive user interface. Developing a comprehensive matching system is essential in maintaining a solid user experience. The evidence collected in this literature review confirms that my intended approach of combining advanced AI techniques with user-centric design principles will address significant gaps within current existing solutions.

6 Initial Research

7 Project Management

Due to the nature of this project, a significant amount of research had to be conducted to fully comprehend the working components of this proposed outfit suggestion system. Subsequently, initial user research was then conducted to fully understand what potential users wanted to gain from the app and to gain more insight into what functional and non-functional requirements were necessary. These requirements laid the foundations for the implementation stage of production. After this, the user interface was then designed with a user-centric design process. During the development phase, it became very clear that some features had to be developed concurrently, whereas other features could be developed sequentially. However, most of the design and implementation process happened concurrently. After development was complete, tests were carried out to measure the efficacy of the implemented system. Such tests included unit tests and user acceptance testing.

This Gantt chart in Figure 3, shows the timeline taken for the completion of the project as well as all of the milestones which were achieved along the way.

In addition to using the Gantt chart to illustrate the project schedule, the project was also managed through the use of Agile principles most notably Kanban cards. These cards were created to track and maintain different tasks which were vital to the production process of this project. All resources and notes taken during the development of the project were recorded using Pages. In order to maintain the security and integrity of the project GitHub was utilised. This allowed for effective commits of new updates to the project as well as insightful comments to label each commit. GitHub was also used for version control enabling the ability to reverse anything. Git however did struggle as it was unable to receive the object detection model due to its size. Finally, meetings with my supervisor were conducted each week to get valuable feedback and insights into the design process from my supervisor and colleagues. These meetings were also used to report progress each week.

7.1 Gantt Chart

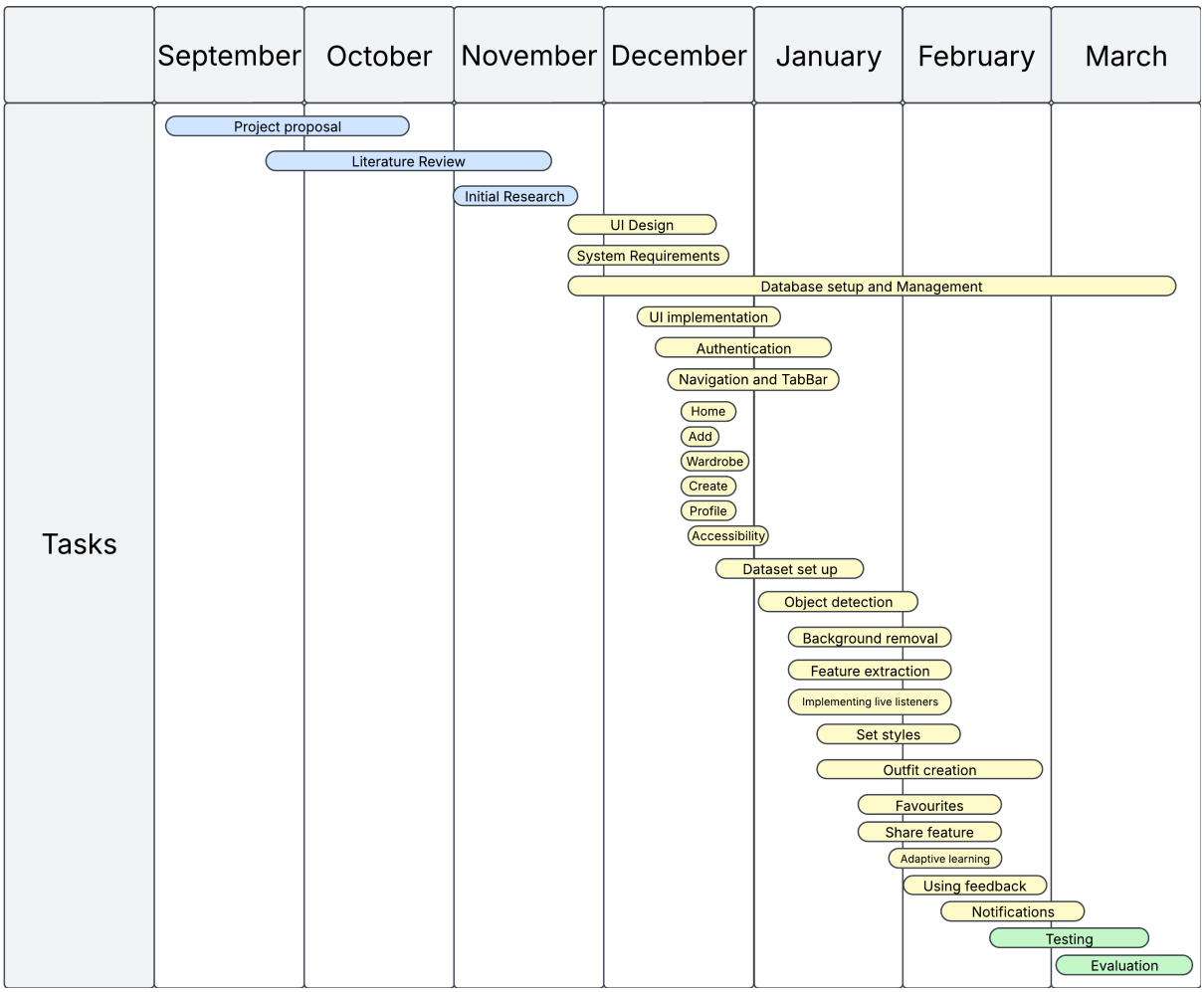

Figure 3: Gantt Chart

8 System Requirements

8.1 Aim

The aim of this section is to layout the fundamental system requirements that the app must adhere to, in order to ensure a strong implementation of the proposed solution. The requirements were developed to guarantee the app is designed in a user-centric manner. The following functional and non-functional requirements were developed when conducting the research phases of this project.

8.2 Requirements

The outlined functional and non-functional requirements were categorised using the MoSCoW framework situated in requirement engineering. This framework which is industry standard denotes the priority of the given requirements using a predetermined criteria "must have", "should have", "could have" and "will not have" [15]. The different requirements will be assigned a status based on their requirement for the project.

8.2.1 MoSCoW Categories

The MoSCoW prioritization framework consists of four distinct categories which are strictly followed to classify a requirement.

- **Must Have (Mo):** These are critical requirements for the project that must be included.
- **Should Have (S):** Important requirements with a high priority which should be included.
- **Could Have (Co):** Requirements that could be added as an extra consideration to further improve the project, however not crucial to the outcome.
- **Won't Have This Time (W):** Requirements which are mainly considered for future works and will not be implemented in the current version.

8.3 Functional Requirements

The functional requirements listed in Table 1 detail the methods and processes which enable users to fully interact with all features within the app. The proposed app requires user authentication functionality enabling users to sign up or log in while sharing necessary personal details.

For the digital wardrobe to realise its full potential users must be able to capture images of their desired clothing items within the app after granting camera permissions. These images will then be uploaded to the database once they have undergone extensive object detection processes other to accurately classify each garment. Each image will also undergo background removal and a degree of feature extraction. User's are able to add information to these images post processing by labelling the images before saving to the database. All of the user data which is collected throughout must be stored in a database with appropriate access restrictions.

Once a sufficient number of clothes have been added to the user's digital wardrobe the user is then able to create outfits. This step enables style selection and outfit creation. The outfits are generated using optimisation methods and matching methodologies to best match the outfit according to a selected style. One critical requirement of the system is the user must be able to interact with the suggested outfits. Should the user reject specific items then the system must adapt its recommendations accordingly.

8.3.1 Functional Requirements Table

ID	Section	Priority	Requirement
1.1	Authentication	Must	User must be able to sign up for an account with first name last name email and password
1.2	Authentication	Must	User must be able to login with correct email and password
1.3	Authentication	Must	The app must deny access if user details are incorrect
1.4	Authentication	Should	User should be able to edit account details

ID	Section	Priority	Requirement
1.5	Authentication	Should	User should be able to reset password
2.1	Home	Must	The latest t shirt that has been added must be displayed
2.2	Home	Must	The latest bottoms that have been added must be displayed
2.3	Home	Must	The latest shoes that have been added must be displayed
2.4	Home	Must	User must be able to able to swipe between the 3 items
2.5	Home	Must	User must be able to navigate to other screens from Home
2.6	Home	Should	User can navigate to favourite outfits section
3.1	Add	Must	User must be able to press "Add Items"
3.2	Add	Must	A set of rules must be added with a "continue"
3.3	Add	Must	User must be prompted to enable camera permissions
3.4	Add	Must	User must be able to take a photo
3.5	Add	Should	There should be the option of closing or retaking the image
3.6	Add	Must	The system must automatically classify clothing items by type using an object detection algorithm
3.7	Add	Must	User must be able to draw a bounding box around the image and confirm their selection
3.7.1	Add	Must	Must accurately identify t shirts
3.7.2	Add	Must	Must accurately identify bottoms
3.7.3	Add	Must	Must accurately identify shoes

ID	Section	Priority	Requirement
3.8	Add	Must	The background must be removed from the image
3.9	Add	Must	Feature extraction must take place recognising RGB values and HSV values
3.10	Add	Should	User should be able to choose retake or confirm which saves the image
3.11	Add	Must	If saved user must be able to enter some identifiable information, brand and size
3.12	Add	Must	The image must save to the database
4.1	Wardrobe	Must	All of the items in the user's wardrobe must be displayed
4.2	Wardrobe	Should	The user should be able to filter by type
4.3	Wardrobe	Should	User should be able to search by brand or size
4.4	Wardrobe	Should	User should be able to press on an item to see more details
4.5	Wardrobe	Should	User should be able to delete item in more details
4.6	Wardrobe	Should	User should be able to toggle between grid view and list view for wardrobe items
5.1	Create	Must	User must be able to choose between 4 different styles casual, streetwear, sandwich method or random
6.1	Outfit	Must	The app must display the outfit for the chosen style
6.2	Outfit	Must	The user must have the choice to reject or save the outfit

ID	Section	Priority	Requirement
6.2.1	Outfit	Must	If the user presses save must be prompted to enter a name. The same outfit or name cannot appear twice
6.2.2	Outfit	Must	If the user rejects then each items rejectionCounter increases by 1. Also a new outfit should be displayed
6.2.3	Outfit	Should	The user is able to reset the rejectionCounter for all items
6.2.4	Outfit	Must	If rejectionCounter = 3 for an item then user is prompted to keep delete or donate the item.
6.2.5	Outfit	Should	The system must decrease the amount of times an item is suggested based on how many times it has been rejected.
6.3	Outfit	Could	User can choose to lock an item. 3 locks prompts the user to save the outfit.
7.1	Favourites	Must	Each saved outfit must be displayed
7.2	Favourites	Must	User must be able to delete outfit
7.3	Favourites	Should	User should be able to share outfit
7.4	Favourites	Should	User should be able to search for an outfit
8.1	Profile	Must	User must be able to logout
8.2	Profile	Could	User could visit links to donate clothes
9.1	General	Must	If the user exits the app the user must be logged out

Table 1: Functional Requirements.

8.4 Non-functional Requirements

Aside from the functional goals which have been mentioned it is also essential to define the behaviour of the app in order to ensure a positive experience for the user. The following non-functional requirements listed in Table 2, encompass a range of topics which affect user experience. These requirements should be considered during the design and implementation phase in order to maintain a user-centric design throughout. As this project uses an external database it is reliant on a strong connection for optimal performance. The performance goal in the app is to read, write, delete and update to and from the database within 1.5 seconds, with loading indicators for longer processes. This indicators are incorporated to ensure the user remains focused on the app. The image classification algorithm will achieve at least 80% accuracy. The background removal will execute instantaneously as soon as the item has been accurately classified. The system will exhibit some multiprocessing because while the background is being removed there is also a high level of feature extraction which occurs on the image. Security is paramount throughout this project as sensitive data is handled such as personal information which is used for authentication. The way the data is handled must be in accordance with GDPR policies, this allows users to delete their accounts as and when requested. The user interface will remain consistent across all iOS devices. Resource management aims to keep each image under 1.5MB which will prevent memory leaks and minimisation any battery consumption during computationally expensive tasks. The code will follow the MVVM architecture pattern and it will be well commented throughout to ensure maintainability.

8.4.1 Non-Functional Requirements Table

ID	Section	Priority	Requirement
1.1	Performance	Must	All database related actions should execute in less than 2 seconds
1.2	Performance	Must	Loading indicators must be displayed to represent the time taken for a database action

ID	Section	Priority	Requirement
1.3	Performance	Must	Image classification must achieve at least 80% confidence rating
1.4	Performance	Must	Background removal must execute after accurate image classification
1.5	Performance	Must	Feature extraction must occur in parallel with background removal
1.6	Performance	Must	Image classification should work in an optimal way to minimise memory and battery consumption
2.1	Database	Must	The app must maintain database connection when possible
2.2	Database	Must	The app must reflect changes in the database within 2 seconds of updates
3.1	Security	Must	All user data must be stored securely
3.2	Security	Must	Personal data must be handled in accordance with GDPR regulations
3.3	Security	Must	Users must be able to delete their accounts and any associated data
3.4	Security	Must	The database must be inaccessible to unauthorised users
4.1	Usability	Should	User interface must be compatible with all iOS devices
4.2	Usability	Must	User interface must stay consistent
4.3	Usability	Must	All user interfaces must be responsive and have a quick response time under 2 seconds
5.1	Maintainability	Must	Code must follow MVVM architecture
5.2	Maintainability	Must	Code must be well commented

ID	Section	Priority	Requirement
6.1	Reliability	Must	Error messages should prompt the user to take the correct action
6.2	Reliability	Must	Errors should not damage the user experience
6.3	Reliability	Must	All of the errors should be logged
6.4	Reliability	Must	The app should not fail if there is an error
6.5	Reliability	Could	The app works without internet connection
7.1	Scalability	Must	Database must be able to handle large wardrobes 100 plus items
7.2	Scalability	Must	Image processing must maintain performance across various clothing types

Table 2: Non-Functional Requirement.

8.5 Accessibility

The current guidelines in the United Kingdom strongly advise that every single public sector app must meet the international WCAG 2.2 AA accessibility standard [16]. This standard ensures that the app can cater to the widest range of user's possible. The WCAG operates under 4 key guidelines which are POUR (perceivable, operable, understandable and robust) [17]. These guidelines and principles are established to ensure the app caters to the widest user base possible. In addition to this the app must meet AA success criteria in order to fully conform with UK guidelines and WCAG principles. The app will utilise iOS functionality by taking advantage of the VoiceOver accessibility function. This function will announce all of the actions taken by the user e.g. navigating or swiping to inform the user should they be visually impaired. In addition details regarding the garment will also be announced to help the user gain a different perspective on their clothes.

8.5.1 Accessibility Table

ID	Section	Priority	Requirement
1.1	Perceivable	Must	Provides text to speech through the use of voiceOver within the app
1.2	Perceivable	Must	Provides captions for different forms of media
1.3	Perceivable	Must	Content can be presented in different ways
1.4	Perceivable	Must	User can toggle between light and dark modes
1.5	Perceivable	Must	Text size can be altered within a given range
2.1	Operable	Should	User is able to enable some functionality from the keyboard
2.2	Operable	Should	None of the content will cause seizures or physical harm
2.3	Operable	Must	Navigation is very simple throughout the app
3.1	Understandable	Must	All text is readable and understandable
3.2	Understandable	Should	Helps users avoid and correct mistakes
4.1	Robust	Could	Maximise compatibility with other platforms and tools

Table 3: Accessibility Requirements.

9 Design

9.1 Aim

In this section, the complete design phase of the application will be established primarily focusing on functionality and user experience. This section will cover the design process of the application, database and the user interface. Culminating in a fully transparent blueprint for the project which will cover the entire development process and iterative design for meeting requirements.

9.2 System Design and Architecture

When developing this application I decided to use the three-tier architecture which organises the application into three logical computing tiers. The application processing steps are divided into 3 main sections presentation layer, application layer and the data layer. The presentation layer is where the UI is presented. The application layer is considered the heart of the application this handles user actions while also querying the database constantly. The data layer contains access to the back-end or the database. As this is a three tier architecture all of the communication within the app goes through the application tier. The presentation tier and the database are not able to communicate with each other to maintain the apps functionality [18] [19].

The layout of this architecture allows for compartmentalisation of the functionality across the three different tiers. The lack of communication between the presentation layer and the data layer adds a layer of security to the application while also maintaining the functionality of the app. Due to the set up of this architecture it also allows for concurrent programming which improves development. For small mobile applications like this one the three tier architecture presents an attractive solution. The distribution of the different features and the potential to use multiple different machine learning components was one of the main reasons why I chose this architecture. This project is well geared to using this robust architecture as it maintains simplicity while also enabling independent development of each layer. Both of the presentation tier and the application tier will use SwiftUI as this application has been developed for iOS devices. The data tier will use Firebase as an external database.

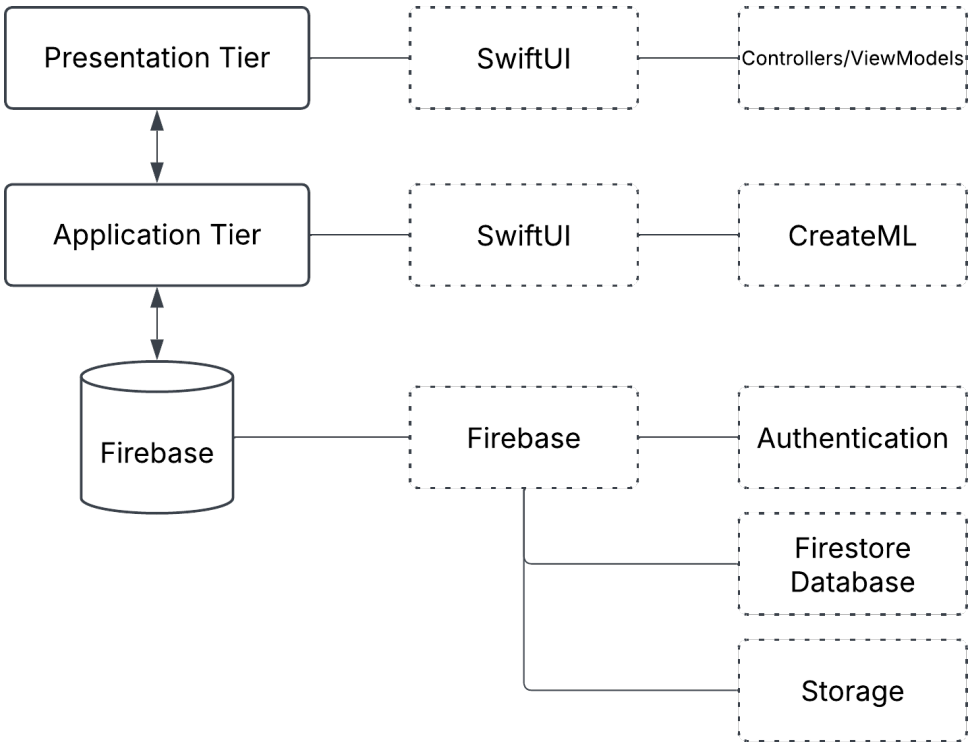

Figure 4: Three-tier architecture of the outfit suggestion application

As shown in Figure 4 the presentation layer utilises SwiftUI in conjunction with the controller/viewModels that manage them. This follows the MVVM pattern (Model-View-ViewModel) which is an inherent property of SwiftUI. The application layer also uses SwiftUI however it uses CreateMl as well in order to incorporate on-device machine learning and other artificial intelligence capabilities. The data layer uses a range of services from Firebase including authentication, Firestore database for data storage and Storage for handling images of items of clothing.

9.3 Component Diagram Design

This component diagram illustrates the architectural structure of my application, WardrobeBuddy. As stated during the design process of the system and architecture we will follow a three tier architecture which focuses on the presentation, application and data layer. This tier system has been used to construct the component diagram of the system. The separation between the different layers ensures code maintainability, testability and also scalability.

The presentation layer which is implemented in SwiftUI encapsulates all of the user interface components necessary to enhancing the user experience. This different views

strictly follow the presentation layer criteria as they delegate logic and information to the application layer and the relevant viewModels found within the application layer. The application follows the MVVM (Model-View-ViewModel) structure which is an inherent property of SwiftUI. This structure enables effective management of state transitions as well as processing user interactions. The application layer also handles operations such as outfit generation and accessibility services. The data layer displays the interactions that the application has with the external database, Firebase. WardrobeBuddy utilises different features of Firebase such as authentication, Firestore and storage which are all used for secure data management. The relationships between the different components are illustrated with dotted lines, these relationships facilitate controlled communication between the tiers. This approach enhances maintainability as well as scalability, testability and reusability which are also advantages of using the MVVM structure [20].

9.3.1 Component Diagram

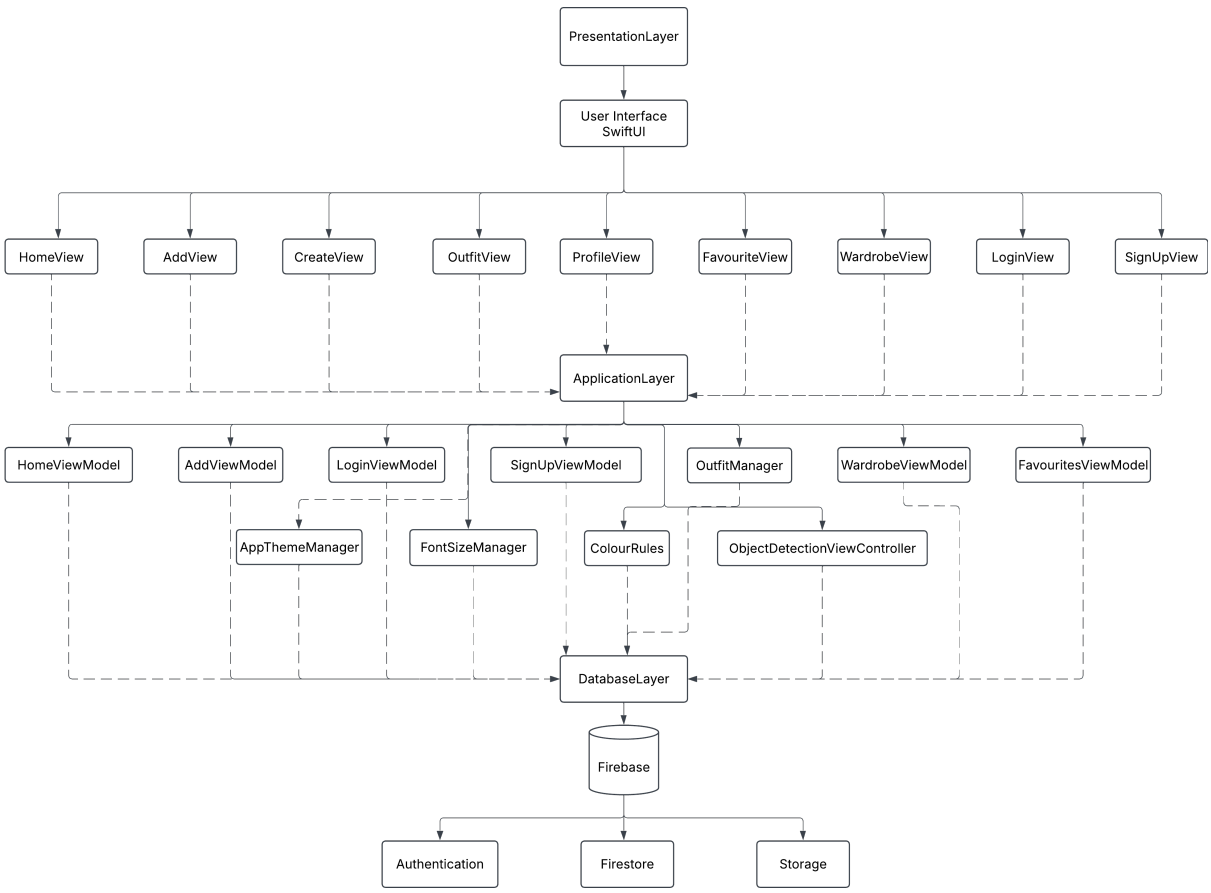

Figure 5: Component Diagram for different components of the app.

9.4 Class Diagram

The class diagram displayed here presented a comprehensive view of WardrobeBuddy's application architecture while also emphasising the inter class relationships between the various different components and classes throughout the distinct layers. We evaluate how the different relationships interact with each other by establishing multiplicities to record relationship types such as one-to-one, many-to-one, one-to-many, etc. We can also augment these relationship multiplicities based on the requirements for the class. The authentication layer allows the user to log in via Firebase's services. The main layer implements the MVVM (Model-View-ViewModel) pattern. This pattern instantiates clear one to one compositions between each of the views as well as their respective view models. The navigation layer shows how the user interacts with the app and the possible routes for the user to take to effectively traverse through the app. The class diagram shows the HomeView as the central hub as this is the first page the user sees when they login. As mentioned before we can use augmented relationships between objects this can be seen when generating an outfit as we use a 1 to 3 relationship between outfits and their constituent clothing items. This class diagram shows the various relationship types which precisely communicate how the different classes interact. The diagram has also been labelled with multiplicities to provide quantitive context for the different interactions, this creates a cohesive representation of the systems architecture facilitating insight into the core functionality.

9.4.1 Class Diagram Table

This table will explain all of the classes which have been utilised during the development of this app. As a lot of the initial classes which follow the MVVM architecture do the same things they will be explain in the same way. However classes that are more complex and crucial to the understanding of the app will be explained in more detail.

ID	Class Name	Function within the app
1	ViewClasses	These ViewClasses are situated in the presentation layer and provide the user interface. The ViewClasses have a one to one relationship and facilitate the navigation between screens
2	ViewModelClasses	These ViewModelClasses are situated in the application layer and they handle the business logic and the application states. Each of the views connect to a data source and they aid in managing data operations.
3	Photo	Represents the images taken of the different items of clothing with metadata. The photos are used and represented in different ViewModels
4	Category	This defines the different categories used for filtering through the items within wardrobe.
5	WardrobeItem	Represented the items of clothing with the different assigned properties such as type, brand and colour. This again is used by multiple different ViewModels
6	WardrobeCount	Keeps track of how many items of each type are in the wardrobe. This is utilised by CreateView to ensure minimum requirements.
7	StyleOption	Defines the different rules for matching items of clothes according to the different defined styles.
8	RGBValues	Stores the extracted colour information for all of the clothing items when the image is uploaded.

ID	Class Name	Function within the app
9	GeneratedOutfit	Displays a complete outfit combination containing top, bottom and shoe. Has a one to three relationship with wardrobeItem
10	SavedOutfit	Enables the user to save an outfit and represents a saved favourite outfit.
11	OutfitItem	This is a simplified version of WardrobeItem however this is used for the saved outfit items. This is only used in SavedOutfit.
12	StoredOutfit	Database representation of outfits.
13	ColourRules	Implements colour matching algorithms for the different styles which are available to the user.
14	ImageCache	Stores and retrieves images to avoid redundant network requests. Uses threading and queues.
15	NetworkService	Handles the downloading of images from the database URL's. Used in conjunction with ImageCache to load images efficiently.
16	CameraView Controller-Representable	This is a SwiftUi wrapper for the UIKIT camera controller. This creates ImageClassifierViewController
17	ImageClassifier ViewController	This handles all camera functionality such as image capture, classification, background removal and feature extraction. This is in direct communication with the database.
18	AppThemeManager	Manages different view settings such as light or dark mode across the app,
19	FontSizeManager	Manages different text size settings across the app.

ID	Class Name	Function within the app
20	Firestore	This provides a real time No-SQL database and functionality. This will store and retrieve all of the applications data.
21	FirebaseStorage	This provides blob storage for the images of varying size. Any image uploaded will be saved under files for each user.
22	Firebase_Auth	This handles user authentication and account management effectivley. User sessions and security concerns are passed through this.

Table 4: Classes used in the WardrobeBuddy application

9.4.2 Class Diagram Illustration

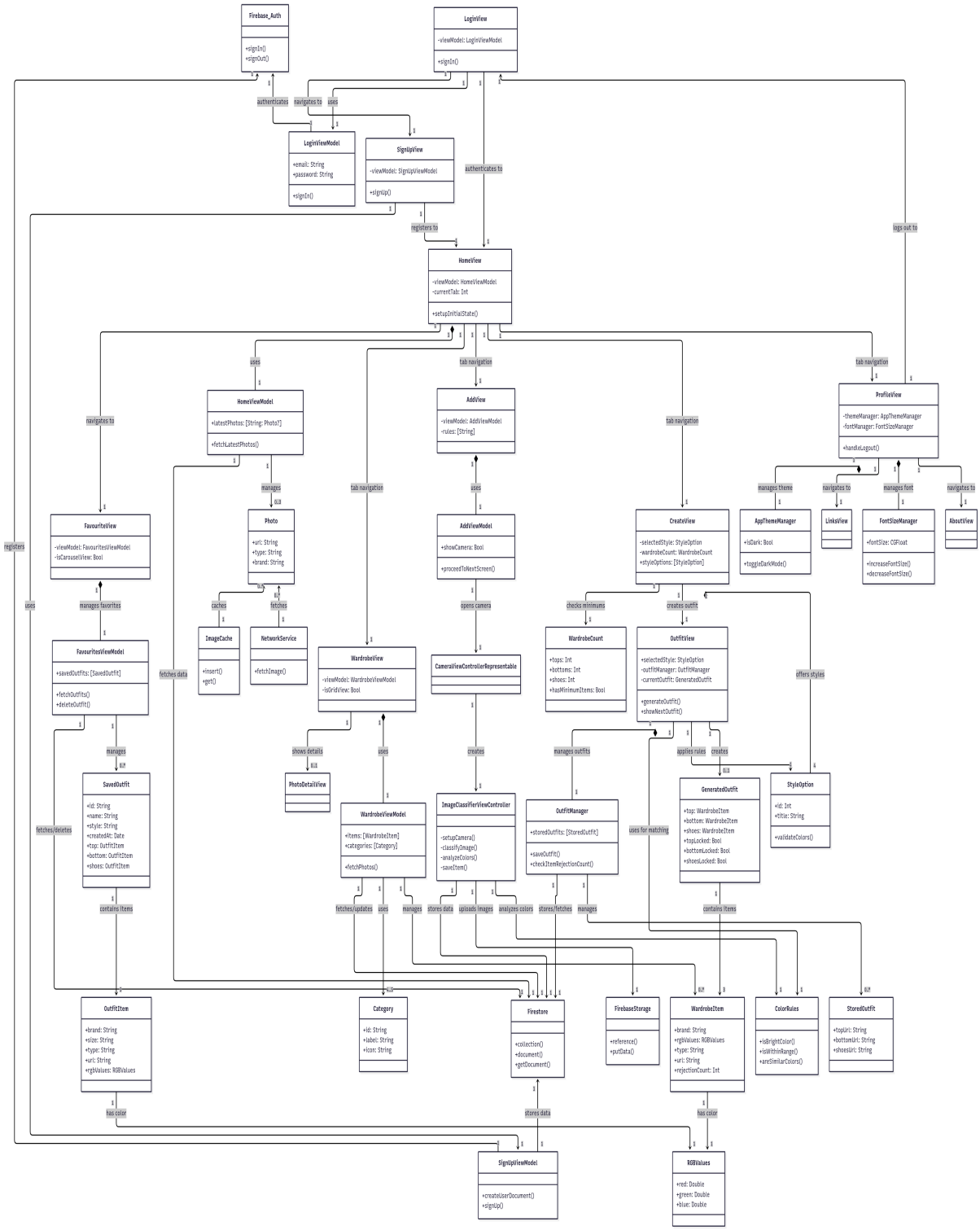

Figure 6: Class Diagram

9.5 Database Design

The database for this project was designed and developed using Firebase which offers real time database synchronisation along with secure and robust authentications services. For small to medium sized applications such as this outfit recommendation system, Firebase is an ideal solution due to its seamless integration with mobile apps. It must be noted however that Firebase is limited in terms of relational data as it does not use an SQL database or an alternative [21].

9.5.1 Entity Relationship Diagram

Figure 7 shows the Entity-Relationship diagram which depicts how the database has been designed in Firebase presenting a comprehensive overview of the database. Each user is able to store personal information as well as images which can be used for outfit generation.

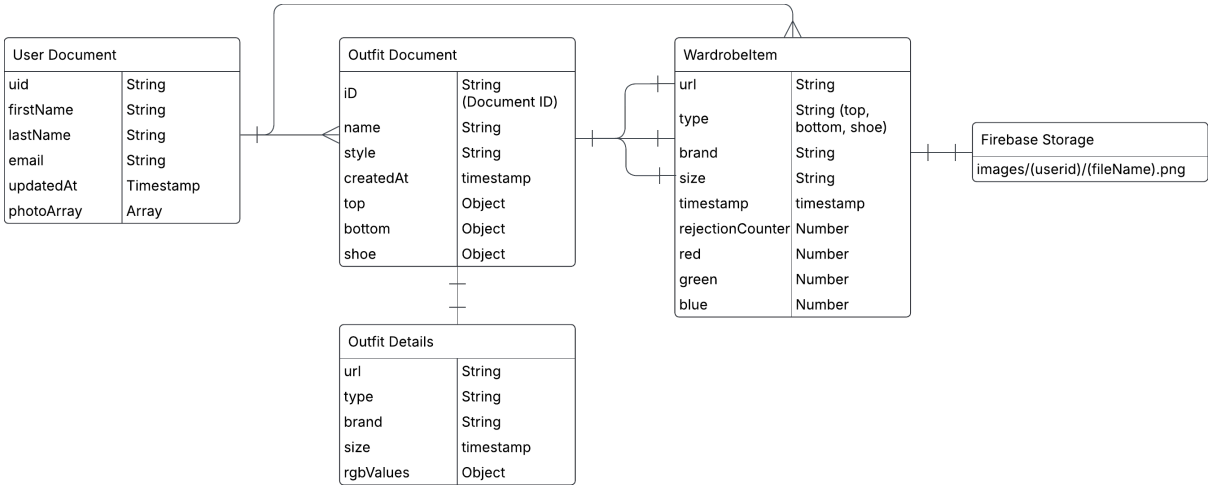

Figure 7: Entity relationship diagram for Firebase database

This Entity-relationship diagram illustrates the different interactions between components of the database structure. Some entities exhibit one to one relationships where as other entities exhibit one to many relationships. The diagram displays the key relationships in this project. One thing that should be noted about the diagram is that some of the entities displayed show how the collections are stored in the database they are not used in this manner. The aim of this diagram is to provide a holistic view of the different data structures used within the project.

9.6 User Interface

10 Implementation

10.1 Technology Stack

10.1.1 Programming Language and IDE

This mobile app was developed specifically for Apple devices by leveraging the native iOS ecosystem and its robust development framework and technologies available. During the design process I chose to use Xcode as the primary integrated development environment dues to its comprehensive debugging tools and performance capabilities tailored to iOS devices. In addition to this Xcode was the most suitable option as it has seamless device integration which was another key factor as this is imperative for software development. Apple's high level programming language SwiftUI is the most suitable UI framework and should be implemented as the primary programming language when constructing the user interface and functionality. Another reason why SwiftUI and Xcode were chosen was due to the importance of testing on a physical iPhone. During the development process continuous testing was required to ensure the app was meeting expectations and SwiftUI enabled me to do this easily. This technology stack provides direct access to on device machine learning frameworks which are heavily utilised in this app. Continuous testing on my physical iPhone was necessary as it would provide an accurate assessment of the efficacy of the object detection algorithm, camera functionality and database synchronisation under real world conditions rather than simulated. SwiftUI also provides type safety and various memory management features which are vital when implementing complex feature extraction algorithms and also optimising the matching system on the physical iPhone. Other potential tech stacks such as Flutter or React Native were rejected as while they do provide cross platform compatibility, Swift provides better performance natively on more computationally intense tasks such as image processing.

10.1.2 Database

Firebase is a Google backed platform which is a back end as a Service (BaaS). Firebase serves as the backbone of this applications database architecture which has been designed to resolved around user specific collections which have been mutated to store comprehensive wardrobe data. Due to the ease of integrating Firebase into this app it

gave me more time to augment the database to my specific needs for this app and spend more time developing the front end. For WardrobeBuddy each user is given a unique user identification number which anonymises the user to ensure no data risks. Each user is then assigned a document which will contain a photoArray, this is used to store metadata for clothing items including URL's, type of clothing, brand information, size and the main RGB values for each item. As this project uses live listeners throughout it means that the app is able to listen for any changes to the database in real time and will update through the app accordingly. Another part of Firebase which is used within this app is Authentication which is implemented using email and password verification. For sign-ups the user is prompted to enter a first and last name as well. Authentication will enable the system to create user specific buckets that are used to store images. These buckets are queried throughout the system most nominally when generating outfits. The outfit generation system will query the database to filter items by type or attribute. The database is also used in rejection counting where the system will incrementally update the items metadata to refine future recommendations. Firebase also offers a storage feature which allows images to be stored in conjunction with the user ID and also a filename. This implementation of Firebase within the app allows for ensuring the data is available locally even when not connected as the data is cached on device. In addition to all of this error handling is also implemented throughout Firebase with appropriate user feedback and tracking [22] [23].

10.2 Object Detection *(Image Processing)*

Within the realms of machine learning and artificial intelligence there are numerous industry standard options that can be selected for object detection such as TensorFlow.js or Google's ML Kit. However as this is an app build for the iOS ecosystem it makes sense to leverage Apple's native technologies such as Create ML and Core ML. These native technologies offer benefits in terms of performance, integration and optimisation. Create ML enables the creation and training of customise models on a given dataset. While Core ML facilities seamless integration and exportation into iOS applications. Object detection is one of the key features of this app as it firstly serves as a checking mechanism on the image that the user wants to upload but it also aids with other key features such as background removal and feature extraction.

Firstly one algorithm which is used for object detection specifically in Create ML are convolutional neural networks. These are specialised neural networks which excel at object detection tasks. One benefit of CNN's is that they incorporate multiple specialised layers which process weights and biases through non linear activation functions [24]. In Create ML's implementation the model contains layers such as convolutional, pooling, reshape, slice permute amongst others. These layers work in conjunction to collectively preprocess the input by computing local regions before passing the preprocessed image to NMS for accurate bounding box generation.

Create ML's framework utilised both convolutional neural networks and non-maximum suppression (NMS) to formulate the object detection model which is used to produce accurate classification results. This algorithm, NMS aims to find extreme values and then suppress or discard the remaining values in the neighbourhood, this feature makes it particularly suitable for object detection as it progressively improves recall. The model has the ability to identify all instances of a specific class [25]. NMS can be implemented into object detection and also assign each image a confidence score through the use of a bounding box. The algorithm would select the highest scoring box. Other boxed with significant overlap which is measure by the Intersection over Union (IOU) are then suppressed or removed. This results in more accurate and precise object detection [26].

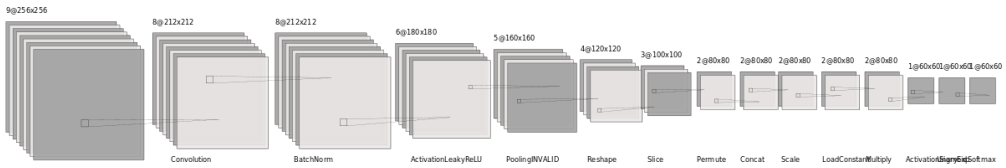

Figure 8: Convolutional Neural Network Diagram of Create ML

10.2.1 Data Collection

The effectiveness of object detection is heavily dependent on high quality datasets which are used for training and testing. One effective strategy for assembling a robust dataset is leveraging a pre-existing dataset however I constructed my own dataset through the use of an inclusion/exclusion strategy. This strategy had three primary conditions, firstly there must be no human presence or identifiable aspects such as hands, body or face. Secondly the strategy would reject images of a low quality and finally images would be rejected if the item was not fully in the frame and was unidentifiable even by a human. This methodical approach while time-consuming yielded data which is tailored to my specific use case. The items In the dataset were sourced from Depop and Vinted as they facilitate for user uploaded content which mirrors the uploading process within this app. Depop and Vinted also replicate the ambiguities in an uploaded photo such as varying angles, poor light or poor image quality which are very likely to occur within this implementation. Once all of the images were collected they were then manually labelled via Roboflow transforming the raw data into a comprehensive dataset that was subsequently exported to CoreML for seamless integration with iOS apps [27].

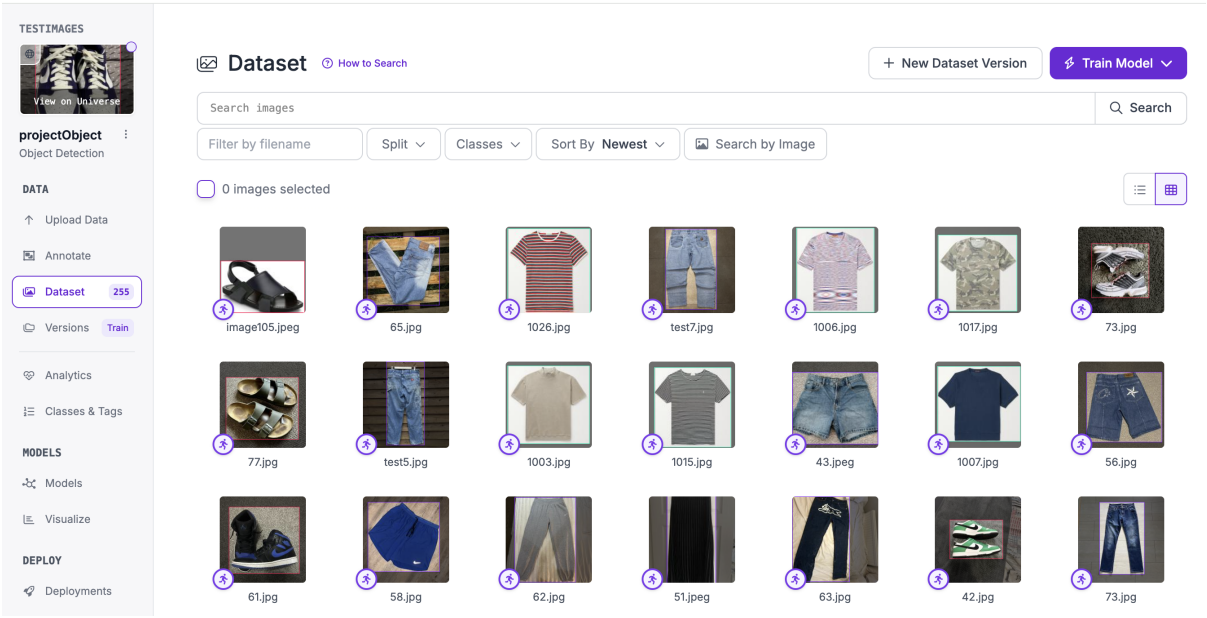

Figure 9: Annotating Images via Roboflow

10.2.2 Model Training

The integration with CoreML facilitated a comprehensive training process by leveraging an API within the Apple ecosystem *VNCoreMLRequest* which is utilised for image processing capabilities. The dataset when training and during the implementation was split into three distinct classes top, bottom and shoe. Where the model was engineered to identify and learn distinct features and attributes specific to each class of clothing. The training process was rigours as it entailed 7000 iterations resulting in a final loss value of 0.823. The achieved loss value indicates that the model does have strong object detection capabilities which are sufficient for the apps requirements. However there is a trade off between the size of the data set and the performance of the model indicating that a larger dataset is required for the loss value to reduce improving detection reliability in real world scenarios. The trained model is then exported and integrated into the main app via *MLModelConfiguration* which facilitates the implementation of machine learning within the iOS environment.

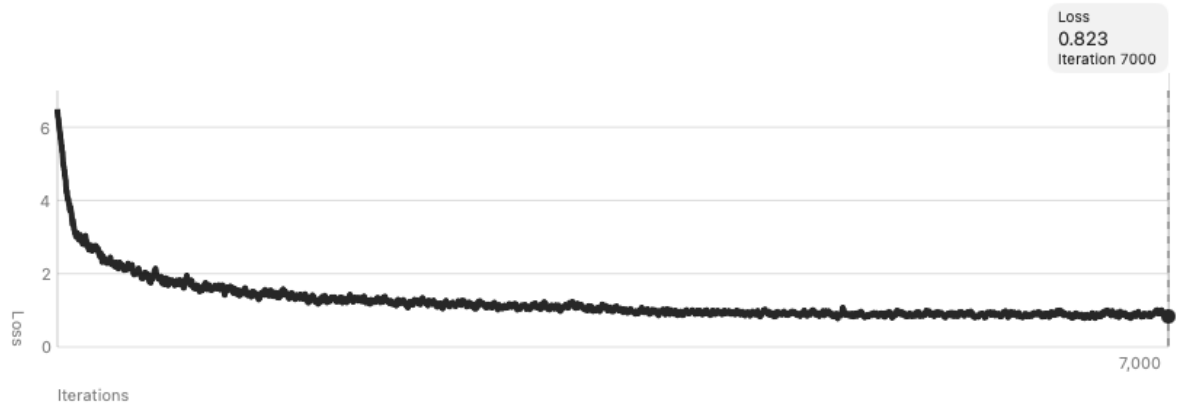

Figure 10: Model results in Create ML

10.3 Background Removal *(Image Processing)*

One of the fundamental aspects related to intelligent outfit generation within this app is the removal of unnecessary information within the image, commonly known as background removal. This process ensures that the user and the system focus solely on the foreground object which is the clothing item, rather than being distracted by any external background elements. In the image that the user uploads the foreground needs to be separated from the background and the background removed leaving just the object in

question. Apple's ecosystem facilitates such requirements via *VNGenerateForegroundInstanceMaskRequest()*. The background removal process consists of three main stages, obtaining a mask from the image, applying the mask to extract the subject and converting the resulting image into a format that is displayed by the user interface.

This process beings with a method called *removeBackground* which takes a CIImage object as an input, This method leverages *VNGenerateForegroundInstanceMaskRequest()*, an API that generates an instance mask of noticeable objects that can be separated from the background. Once the foreground and background images have been identified we can use *generateScaledMaskForImage()* to create a high resolution mask where everything becomes transparent black expect for the specific instances. This high resolution mask is then converted into a CIImage object which is used for further processing.

Once the mask has been generated then the subject is separated from the background as we apply the mask to the original image by using *blendWithMask()* which blends both the images together according to the specifications of the mask. Finally the image must be converted into a format which is compatible with the user interface so that it can be displayed. These steps collectively facilitate the efficient removal of backgrounds from clothing images by leaving only the essential item for further processing and outfit generation [28].

10.3.1 Background Removal Examples

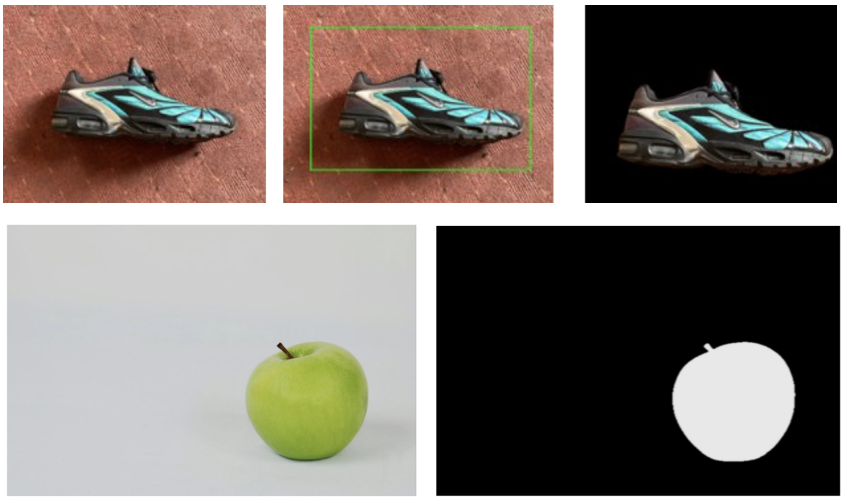

Figure 11: Examples of Background Removal

10.4 Feature Extraction *(Image Processing)*

Another key component in this implementation of WardrobeBuddy, is computer vision based feature extraction. When the user is uploading an image the system needs to be able to conceptualise the characteristics of the image for accurate outfit recommendations. Following on from object detection and background removal, feature extraction for colour information plays a critical role as it enables the system to understand the properties of each item of clothing while also creating harmonious outfit combinations.

The process begins after object detection has accurately classified the items of clothing and background removal has successfully split the item from the background. Once this is done then the system will call on *analyzeColors* which will process the image pixel by pixel in order to extract precise colour information. This method will then iterate through each pixel recording the RGB (red, green, blue) values from all of the non-transparent pixels. The RGB calculation employs a grid based approach which treats each images as a two-dimensional array. The system will explore each x and y coordinate of the image to establish a comprehensive understanding of the colour composition at the pixel level. After this the method will then calculate the average RGB values while providing a representative colour profile of each item. These RGB values are then communicated to the database alongside other metadata.

This colour information which is collected serves as a fundamental input for the outfit generation algorithms particularly in this implementation where outfit generation adheres to specific styles rules that are dependent on colour theory principles. This comprehensive feature extraction process allows the app to visualise the image forming a strong base knowledge for intelligent outfit recommendations.

10.5 Summary of Image Processing

WardrobeBuddy employs object detection, background removal and feature extraction into a cohesive processing pipeline that executes when users upload clothing items. When the user first takes the photo the object detection model is first deployed, classifying the image into one of the three defined classes with a confidence score. If the score is above 50% then the system will proceed with background removal applying *VNGener-*

ateForegroundInstanceMaskRequest() which isolate the item from its background. Simultaneously the feature extraction process analyses the pixels of the item to calculate the average RGB values. These three process work in unison to transform a raw image into a properly processed image which can be used. This seamless integration creates a frictionless user experience despite complex algorithms being executed.

10.5.1 Workflow of Image Processing

11 Evaluation

12 Conclusions

References

[1] UniformMarket. Global apparel industry statistics. https://www.uniformmarket.com/statistics/global-apparel-industry-statistics, March 20 2025. Accessed: March 20, 2025.

[2] WRAP. Nations' wardrobes hold 1.6 billion items of unworn clothes as people open to new ideas, 7 October 2022. Report.

[3] Erica Euse. Men will spend four months of their lives deciding what to wear. https://www.complex.com/style/a/erica-euse/men-spend-four-months-of-lives-deciding-what-to-wear, June 6 2016.

[4] Brian Lao and Karthik Jagadeesh. Convolutional neural networks for fashion classification and object detection. mailto:bjlao@stanford.edu,mailto:kjag@stanford.edu.

[5] Zhihua Feng, Tao Yang, Xin Luo, and Kenji Kita. An object detection system based on yolov2 in fashion apparel. mailto:fengzhihua2012cc@163.com,mailto:yangtao@dhu.edu.cn,mailto:xluo@dhu.edu.cn,mailto:kita@is.tokushima-u.ac.jp, 2018.

[6] Andrew D.J. Cross and Edwin R. Hancock. Relational matching with stochastic optimisation. mailto:ad.cross@york.ac.uk,mailto:edwin.hancock@york.ac.uk, 1995.

[7] G. Ayorkor Mills-Tettey, Anthony Stentz, and M. Bernardine Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. CMU-RI-TR-07-27, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 2007.

[8] Omar Kashif Majeed, Zain ul Abideen, Usama Arshad, Raja Hashim Ali, Abdullah Habib, and Rafay Mustafa. Adaptivecloset: Reinforcement learning in personalized clothing recommendations. Faculty of Computer Science & Engineering, GIK Institute of Engg. Sciences & Technology, Topi, Pakistan; BCMaterials Basque Center for Materials, Applications & Nanostructure, Leioa, Basque Country, Spain; Dept. of Technology & Software Engineering, University of Europe for Applied Sciences, Berlin, Germany, 2025. Corresponding Author: hashim.ali@giki.edu.pk.

[9] Chandrakant D. Kokane, Aadit Rode, Rushikesh Sangale, Jaysri Rathod, Sonu Khapekar, and Vilas V. Deotare. Ml-based outfit suggestion system. Nutan Maharashtra Institute of Engineering and Technology, Talegaon(D), Pune, Maharashtra, India, 2023.

[10] Cristiana Stan and Irina Mocanu. An intelligent personalized fashion recommendation system. *Computer Science Department, University Politehnica of Bucharest*, 2019. Email: cristiana.stan@stud.acs.upb.ro, irina.mocanu@cs.pub.ro.

[11] Lavinia De Divitiis, Federico Becattini, Claudio Baecchi, and Alberto Del Bimbo. Style-based outfit recommendation. *University of Florence*, 2021. Email: lavinia.dedivitiis@unifi.it, federico.becattini@unifi.it, claudio.baecchi@unifi.it, al-

berto.delbimbo@unifi.it.

[12] Nithya Attaluri and Sadhana Lolla. Styleai: A deep-learning powered outfit generator. Email: nithyaa@mit.edu, sadhana@mit.edu.

[13] UNIQLO. How to create a look - uniqlo lifewear collection. Available at: https://www.uniqlo.com/uk/en/special-feature/lifewear-collection/coordinate. Accessed: March 22, 2025.

[14] Fitted-AI. Fitted-ai: Closet. Apple App Store. iOS App version 1.0. Retrieved from `https://apps.apple.com/us/app/fitted-ai-closet/id6596771952`.

[15] Khadija Sania Ahmad, Nazia Ahmad, Hina Tahir, and Shaista Khan. Fuzzy_moscow: A fuzzy based moscow method for the prioritization of software requirements. *Department of Computer Science and Engineering, Faculty of Engineering and Technology, Al-Falah University*, 2017. Department of Computer Science, Imam Abdulrahman Bin Faisal University, Dammam, SAUDI ARABIA.

[16] UK Government. Meet the requirements of equality and accessibility regulations. *GOV.UK*, 2025. Accessed: 27 March 2025.

[17] World Wide Web Consortium (W3C). Web content accessibility guidelines (wcag) 2.2. *W3C Recommendation*, 2023. Accessed: 27 March 2025.

[18] IBM. Three-tier architecture. *IBM Think*, 2025. Accessed: 28 March 2025.

[19] Channu Kambalyal. 3-tier architecture. 2025. Prepared By Channu Kambalyal.

[20] Kalidoss Shanmugam. Mvvm design pattern in ios with swift, 2024. Accessed: April 2, 2025.

[21] Ascensor. The pros and cons of firebase. *Ascensor News*, 2025. Accessed: 28 March 2025.

[22] Pankaj Chougale, Vaibhav Yadav, and Anil Gaikwad. Firebase - overview and usage. *International Research Journal of Modernization in Engineering, Technology and Science*, 3(12):1178, December 2021. Impact Factor: 6.752.

[23] BitCot. Leveraging firebase for your app development: When and why to choose firebase, 2024. Accessed: April 2, 2025.

[24] Reagan L. Galvez, Argel A. Bandala, Elmer P. Dadios, Ryan Rhay P. Vicerra, and Jose Martin Z. Maningo. Object detection using convolutional neural networks. In *Proceedings of the Gokongwei College of Engineering Research Congress*, Manila, Philippines, 2020. De La Salle University.

[25] Gavin. One important concept you must know in object detection: Non-maximum suppression (nms). `https://medium.com/@gavin_ml/one-important-concept-you-must-know-in-object-detection-non-maximum-suppression-n` 2023. Accessed: 2025-04-04.

[26] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS – improving object detection with one line of code. In *2017 IEEE International*

Conference on Computer Vision (ICCV), pages 5561–5569. IEEE, 2017.

[27] Olga Pavlova, Andrii Bashta, and Andrii Kuzmin. Approaches of building a real-world object detector data source. *Computer Systems and Information Technologies*, 2023(3):23–27, 2023.

[28] Matteo Altobello. Removing image background using the vision framework, 2024.