Bilkent University
Department of Computer Engineering

Senior Design Project
T2327
Capsule

Detailed Design Report

22003186, Ali Emir Güzey, emir.guzey@ug.bilkent.edu.tr;
22003229, Alp Afyonluoğlu, alp.afyonluoglu@ug.bilkent.edu.tr;
22003158, Ceren Akyar, ceren.akyar@ug.bilkent.edu.tr;
22003530, Deniz Mert Dilaverler, mert.dilaverler@ug.bilkent.edu.tr;
22002379, Mehmet Kağan İlbak, kagan.ilbak@ug.bilkent.edu.tr

Supervisor: Asst. Prof. Hamdi Dibeklioğlu
Course Instructors: Dr. Atakan Erdem & Mert Bıçakçı

15.03.2024

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

constra

1.1 Purpose of the system

Capsule's purpose, and thus the key functionalities, are outfit recommendation, outfit tracking (logging) and planning, and event dress code coordination. Capsule provides outfit suggestions tailored just to serve the users' preferences and the occasion of the event. With outfit tracking, users can track what and when they wear something. With Capsule's advanced analytics view, they can see additional information, such as the least worn clothing items and the most worn color. With event dress code coordination, Capsule is a collaborative platform for event and event dress code planning. It allows seamless outfit sharing and feedback.

In short, Capsule serves the purpose of a transformative tool that helps users make informed wardrobe decisions, streamline their dressing routines, and enhance their overall style experience.

1.2 Design goals

This subsection outlines the critical performance, security, and quality criteria that Capsule must adhere to.

1.2.1. Usability

The application should be intuitive to use. It should not have complex user interface elements and paths that could confuse the user. The user should be able to learn and navigate the basics of the app in under 5 minutes. Furthermore, some application features should be usable without an internet connection. Features such as viewing wardrobe and previous outfits should be available offline.

1.2.2. Reliability

The application should have a reliable infrastructure and should not crash. In the worst case, the backend should not be down for more than 10 minutes. The application should also produce reliable outfits. Test accuracy results of the machine learning models should be higher than 85%.

1.2.3. Performance

The application's user interface should be fast to maximize the user experience. Rendering each component should take less than 100ms. Depending on the network overhead, the machine learning models should produce decent results in under 2-3 seconds. Furthermore, backend resource consumption should always be less than 80%. This would measure usage spikes, which could halt the backend.

1.2.4. Supportability

The application structure should be supportable to receive further development in the future. It should be supportable for at least a year due to potential brand contracts. The application should not have predefined instructions that could hinder its development in the future.

1.2.5. Scalability

The application should be scalable to over 100,000 users without a major financial burden. Expanding the capacity to over 1,000,000 users should make it easy to serve the incoming users.

1.3 Definitions, acronyms, and abbreviations

Capsule Wardrobe: A capsule wardrobe is a minimalist and versatile collection of essential clothing items designed to simplify and enhance one's style while promoting sustainable fashion. The goal is to create the maximum number of outfits with the minimum number of items.
GDPR: The EU General Data Protection Regulation
IEEE: The Institute of Electrical and Electronics Engineers
KNN: K-Nearest Neighbors
KVKK: Kişisel Verileri Koruma Kanunu
PEP8: Python Enhancement Proposal 8
PII: Personally Identifiable Information
UUID: Universally unique identifier. A 128-bit identifier that is standardized and used to identify information in computer systems uniquely.

1.4 Overview

Capsule is a mobile wardrobe solution powered by image processing and machine learning. With Capsule, users can track when and what they wear and get recommended outfits according to user preference and clothing item matching. By tracking users' clothing preferences, we will be able to recommend outfits to wear and sell certain items since they haven’t been worn in a while. The user will also use our platform to log in and plan outfits. Our knowledge about the user's outfit preference will enable us to direct clothing retailers to their preferred demographics via advertisements and/or affiliate marketing.

In addition to the core feature, Capsule will provide a platform for users to discuss and set dress codes for events they attend with friends. For example, if you are going out to a fast food restaurant or going out for drinks, you and your friends can decide on an outfit to wear on the platform. The event feature will be more useful than a regular instant messaging app like “WhatsApp” as the user will have their outfits and clothes integrated with the messaging section and can seamlessly send outfits/clothes and get feedback.

Our vision is to create a mobile application to track and advise users on their wardrobe usage. To achieve our vision, a lot of image processing and machine learning is needed. We need to segment the outfit from a given picture, match it with an item from your wardrobe, label it with

appropriate properties, and use the combination of multiple properties to generate an optimal outfit.

2. Current software architecture

App Name	Capsule (Our Project)	Whering	Pronti	Pureple
Outfit Recommendation	Yes	Yes	Yes	No
Manual Outfit Builder	Yes	Yes	Yes	Yes
UI Quality	Good	Good	Cluttered and Unintuitive	Bad
Wardrobe Analytics	Yes	Yes	No	No
Shopping Recommendation	Yes (with Turkish retailers)	Yes, but inaccurate and no Turkish retailers	Yes, but no Turkish retailers	No
Outfit Collaboration for Events	Yes	No	No	No
Business Plan	Affiliate Marketing, Sponsored Content	No ads or subscriptions, likely affiliate marketing	No ads or subscriptions, likely affiliate marketing	₺28/month on Android, ₺99/month on iOS, Unsubscribed use is nearly impossible due to overwhelming ads

Table 1: Competitive market analysis for Capsule

While the smart wardrobe market isn't new, we believe that Capsule can do things better than the competitors, especially in the Turkish market. We have picked 3 of the most prevalent competitors in the market. Pureple was the worst out of all the smart wardrobe solutions we have seen. They went with a subscription model, which isn't worth the price, and without a subscription, it is impossible to use the app as every click causes a 5 to 30-second advertisement to be run.

Another competitor of ours is Pronti. Unlike Pureple, they have redeemable qualities, and it is usable. However, Pronti's UI and UX aren't desirable. It is hard to navigate and all around ugly.

Whering is the main concern of ours when it comes to market competition. They have a clean UI with a good feature set; however, with the event feature and the better implementation of the core features, we believe we can steal a significant market share from them. Despite its

qualities, Whering doesn't have much presence or support for the Turkish market, which allows us to start from the Turkish market and grow our customer base.

3. Proposed software architecture

3.1 Overview

Capsule follows two architectural styles: layered architecture and microservices architecture. User Interface Layer, Business Layer, and Data Layer are the three layers that form the application. The ML engines for outfit recommendation, clothing item segmentation and labeling, and matching, together with the Django web server, form a microservices architecture inside the Business Layer.

3.2. Subsystem decomposition

Figure 1 below depicts the subsystem decomposition of Capsule.

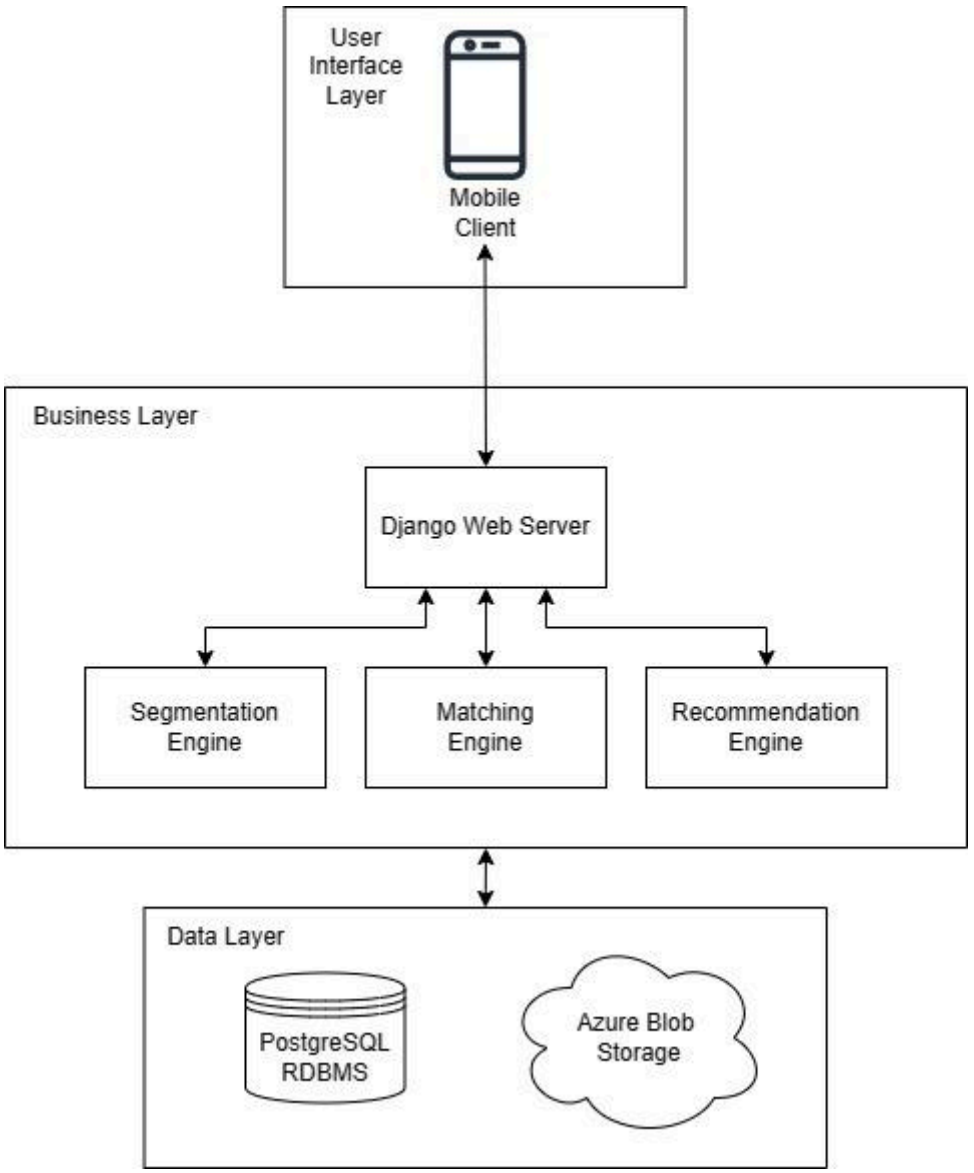


Figure 1: Subsystem decomposition diagram of Capsule

3.2.1. User Interface Layer

The mobile interface will be the only interface through which the user will interact with Capsule.

3.2.1.1. Mobile Interface

Capsule will first be published on iOS platforms; thus, its front end is being developed in Swift for better integration with the Apple ecosystem.

3.2.2. Business Layer

All critical business logic for Capsule will be executed in this layer utilizing a microservices architecture for better encapsulation and decoupling of underlying systems.

3.2.2.1. Django Web Server

The Django web server will be the main business layer component that interacts with the external engines, APIs, and the database. As far as the mobile interface is concerned, the Django web server is the only communication option. All the logic about Authorization and Authentication, users, friends, and events will be handled here. This will also be used as a proxy to interact with the ML engines.

3.2.2.2. Segmentation Engine

Segmentation Engine is the ML model that will crop and label clothes in user images to be used by other engines. These labels are needed to create semantic data from images that are more easily processable. This engine consists of two segmentation models for worn and unworn clothing items and one classification model for labeling segmented images.

3.2.2.3. Recommendation Engine

Recommendation Engine will recommend outfits according to user preference and the outfits the user has in stock. This engine uses the Segmentation Engine's outputs and recommends clothes and outfits to users. This engine is made of multiple models, including the core matching model that matches multiple clothing items to create outfit pairings and the personalization model, which shapes the recommendations based on previous actions and preferences of the user.

3.2.2.4 Matching Engine

Matching Engine will match a picture of the items in an outfit with the ones in the user's wardrobe. This engine functions as a ranking algorithm that extracts features from input images and creates a multidimensional hyperspace accordingly, in which clothing items similar to the provided input image can be found via algorithms like KNN.

3.2.3. Data Layer

The persistent data will be stored and managed in this layer.

3.2.3.1. PostgreSQL Relational Database

As the primary database, a PostgreSQL relational database will be used because of the team's familiarity with the system and due to economic reasons, as it is an open-source system. Unless there is an explicit reason not to store data in a relational database, all data will be stored here.

3.2.3.2 Azure Blob Storage

An Azure Blob Storage account will store and organize the clothing and profile images uploaded by users and those generated by the ML engines.

3.3. Persistent data management

All persistent application data except images will be stored in a PostgreSQL relational database. Image files will be stored in an Azure Blob Storage account on the cloud for ease of maintenance. Three containers will be created on a single storage account for raw clothing images uploaded by users, processed clothing images generated by the Segmentation Engine, and profile pictures. Each image will have a UUID as the file name, making it virtually

impossible to access without having the URL directly sent by the server. Each raw image will have a tag corresponding to the ID of the user who uploaded the image for possible future data processing actions. The URL template for accessing images is as follows.

```
https://<storage-account-name>.blob.core.windows.net/<container-name>/
                                <uuid>.[jpg|png]
```

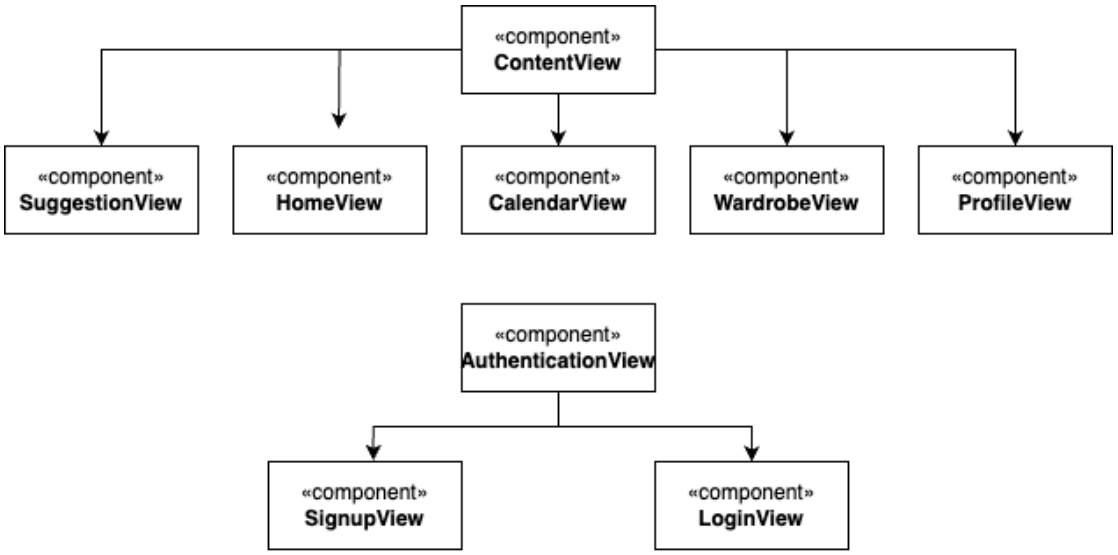
3.4. Access control and security

For authorization and authentication, we are using the JWT bearer token after the user authenticates themselves either with a password and username or Google single sign-on, the client retrieves an access and refresh token. Access tokens are what the user sends the request with and have a shorter lifespan, whereas refresh tokens have a higher lifespan, and the client uses them to get a new access token without logging in again.

Every user is authorized to see their clothes, outfits, and logs. Users can only see other users' outfits if they enable it from their profiles. Users can also modify their entities without access to other users' outfits. For events, users can only see the contents of event chats they are a part of and are blocked from seeing other user's events.

4. Subsystem services

4.1. Mobile Client



AuthenticationView: This view is the parent of SignupView and LoginView. If the user is authenticated, they are redirected to ContentView.
SignupView: This view has several options such as email, Apple, or Google signup.
LoginView: This view has several options such as email, Apple, or Google login.
ContentView: The view is the parent of other views when the user is logged in. While this view is initializing, initial data fetching and updating is done.
HomeView: This is the view that the authenticated user is presented with. It has a swipe feature for outfit recommendations.

WardrobeView: This view has a virtual representation of the user's clothes. It contains both individual items, as well as saved outfits.
ProfileView: This view is the standard profile section of the user. They can change their preferences, update their personal information, or view key metrics such as the most worn outfit.
CalendarView: This view allows users to log their outfits daily. This allows them to keep track of their outfits.
SuggestionView: This view allows users to pick certain options such as the occasion, formality, color combination, or fit to generate outfit recommendations.

4.2. Django Web Server

The Django web server plays a pivotal role as the primary business layer component, facilitating interactions with external engines, APIs, and the database. Its significance extends to the mobile interface, where it serves as the exclusive communication channel. Within the Django web server, critical functions such as Authorization and Authentication, user management, friend connections, and event handling are centralized. Additionally, the Django server acts as a proxy, facilitating interactions with machine learning (ML) engines, further enhancing its role as the central hub of the application's functionality.

4.3. Segmentation Engine

The segmentation engine consists of two segmentation models: a single item model for segmenting unworn clothing items and a multi-item model for extracting multiple clothing items worn on a person. Both models use a U-net structure. After using a separate model to detect whether the input is an image of a person, the image is directed to the corresponding segmentation model. Segmented images are then passed to a classification model for labelling.

4.3.1. Single-item segmentation model

The single-item model uses pre-trained weights already fine-tuned for binary segmentation of clothing items over the iMaterialist Fashion dataset. The dataset contains 46 categories of clothing items, as seen in Table 2, allowing the model to detect and segment all those items [3]. The model uses “timm-efficientnet-b3” encoder and can semantically segment clothing items without classification into a single class [4]. After segmentation, the resultant mask is used to cut the clothes out of the input image. The resultant masked image is then passed to the classification model.

Table 2: Clothing item categories included in the iMaterialist Fashion dataset [3]

ID	Name	Category
0	shirt, blouse	upperbody
1	top, t-shirt, sweatshirt	upperbody
2	sweater	upperbody
3	cardigan	upperbody
4	jacket	upperbody
5	vest	upperbody
6	pants	lowerbody
7	shorts	lowerbody

8	skirt	lowerbody
9	coat	wholebody
10	dress	wholebody
11	jumpsuit	wholebody
12	cape	wholebody
13	glasses	head
14	hat	head
15	headband, head covering, hair accessory	head
16	tie	neck
17	glove	arms and hands
18	watch	arms and hands
19	belt	waist
20	leg warmer	legs and feet
21	tights, stockings	legs and feet
22	sock	legs and feet
23	shoe	legs and feet
24	bag, wallet	others
25	scarf	others
26	umbrella	others

27	hood	garment parts
28	collar	garment parts
29	lapel	garment parts
30	epaulette	garment parts
31	sleeve	garment parts
32	pocket	garment parts
33	neckline	garment parts
34	buckle	closures
35	zipper	closures
36	applique	decorations
37	bead	decorations
38	bow	decorations
39	flower	decorations
40	fringe	decorations
41	ribbon	decorations
42	rivet	decorations
43	ruffle	decorations
44	sequin	decorations
45	tassel	decorations

4.3.2. Multi-item segmentation model

The multi-item model is a collection of segmentation sub-models following a hierarchical structure. Each model fine-tunes a U-net structure that uses “resnet152” as an encoder and is pre-trained on the imagenet dataset. The number of required sub-models is determined automatically according to the selected dataset. The current model uses the Clothing Co-Parsing (CCP) Dataset [5]. Upon manual analysis of the dataset, 59 dataset labels are categorized, and categories are fit into a hierarchical tree structure, as seen in Figure 2, in which red labels represent ignored and removed parts, green labels represent segmented and usable parts, and blue labels represent intermediate images that require further processing before being used. The mapping of dataset labels and chosen hierarchical category labels can be seen in Table 3.

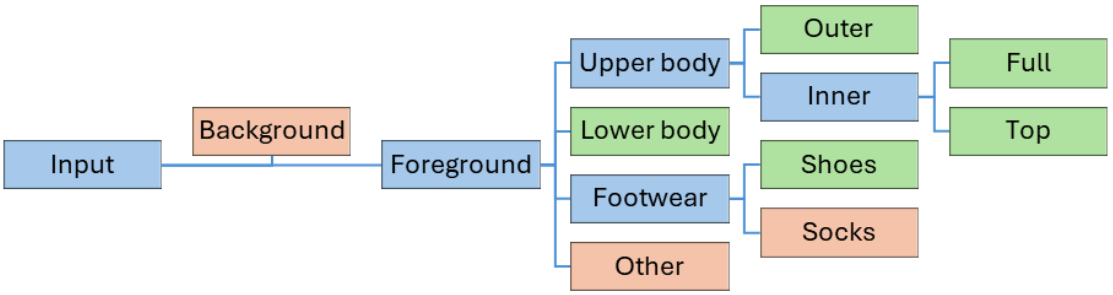


Figure 2: Hierarchical label category structure of the CCP Dataset

Table 3: Mapping of dataset labels and hierarchical category labels

Hierarchical category	Dataset labels
Background	null
Foreground/Upperbody/Outer	cape, blazer, coat, jacket
Foreground/Upperbody/Inner/Full	bodysuit, dress, intimate, romper, swimwear
Foreground/Upperbody/Inner/Top	blouse, bra, cardigan, hoodie, jumper, shirt, suit, sweater, sweatshirt, t-shirt, top, vest
Foreground/Lowerbody	jeans, leggings, panties, pants, shorts, skirt, tights

Foreground/Footwear/Shoes	boots, clogs, flats, heels, loafers, pumps, sandals, shoes, sneakers, wedges
Foreground/Footwear/Socks	socks, stockings
Foreground/Other	hat, hair, bag, purse, wallet, accessories, belt, gloves, scarf, tie, glasses, sunglasses, bracelet, earrings, necklace, ring, watch, skin

A separate sub-model is created for every point in Figure 2 where input is divided into multiple branches. For the CCP dataset, five models are created, and the dataset, consisting of 1004 images and corresponding annotations, is processed for each model [5]. Sample output of layers can be seen in Figure 3, which also reflects how the dataset is processed for each sub-model.

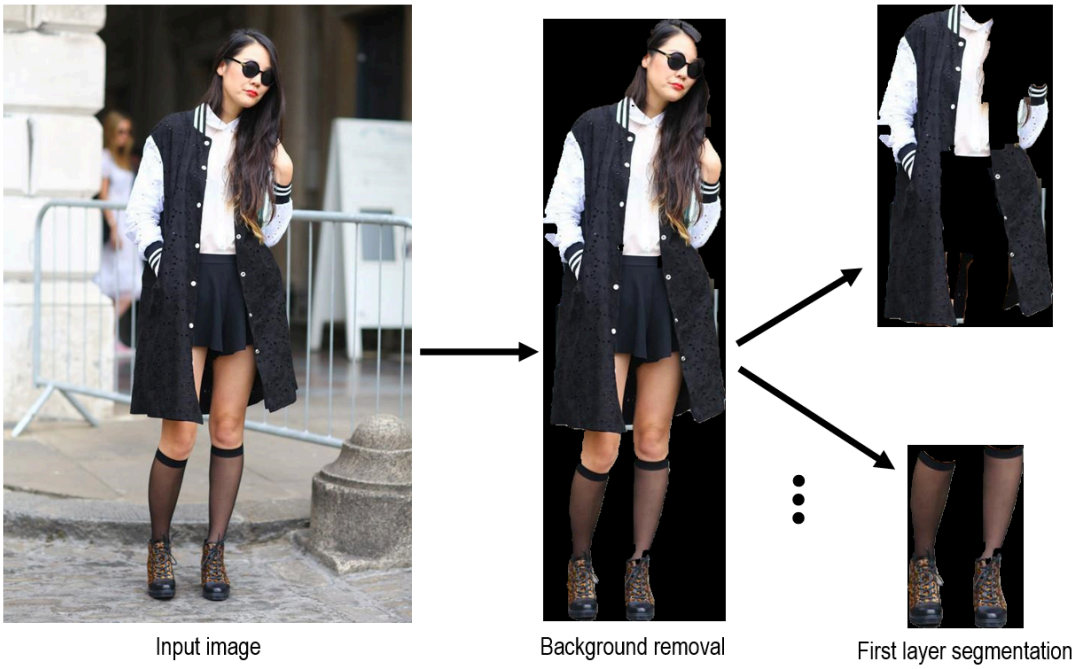


Figure 3: Progression of sample input image through the model

Further details about sub-models and their accuracy levels can be seen in Table 4.

Table 4: Short description and accuracy level of sub-models

Sub-model	Task/Segmented Parts	Test Accuracy
Sub-model 1	Background removal	0.986
Sub-model 2	Upper body, Lower body, Footwear, Other	0.905
Sub-model 3	Outer, Inner	0.819

Sub-model 4	Full, Top	0.894
Sub-model 5	Shoes, Socks	0.946

4.3.2. Classification model

The output of both segmentation models passes through the classification model for labeling clothing items in greater detail. For this purpose, a fine-grained classification model is constructed, the accuracy of which continues improving as different network structures are used to experiment. The model is trained on a custom dataset collected from the web via our custom web scraper. The dataset contains 4400 images having 5 categories and 22 subcategories. Categories and subcategories are used to create a hierarchical labeling tree, which is used to train the model on different layers, which directs the behavior of different parts of the model and increases overall accuracy. The test accuracy of the model is 0.7364, and the related confusion matrix can be seen in Figure 4.

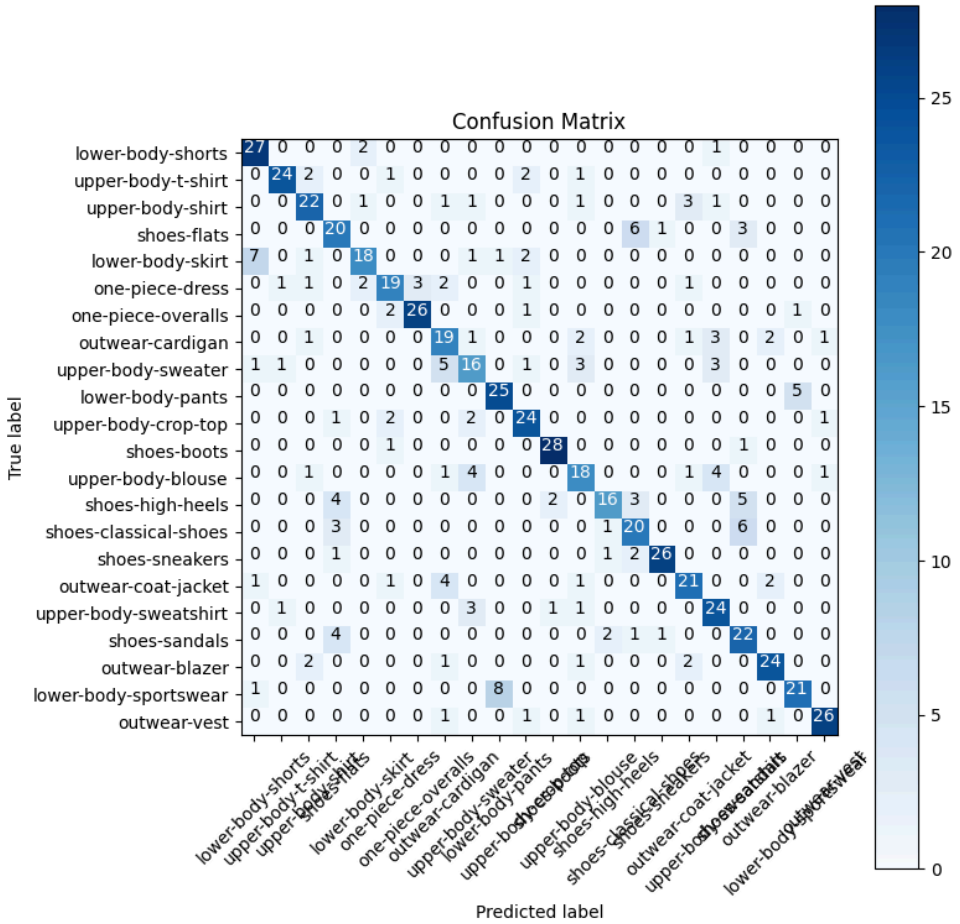


Figure 4: Confusion matrix of the fine-grained classification model

Model structure can be seen in Figure 5, in which the ResNet block represents a resnet18 model pre-trained on the imagenet dataset; all parameters except “layer4” and forward of which are frozen. The last fully-connected layer of the model is removed, making the model headless.

Features extracted via the ResNet block are then passed to Module 1, which uses fully connected layers to output 5 nodes, each of which represents a category. Module 1 output is concatenated with the ResNet output and passed to Module 2, which passes the input through other fully connected layers to output 22 nodes, each representing a subcategory. Both category output and subcategory output are used for loss calculation, making Module 1 and Module 2 specialized in category and subcategory detection, respectively. The current version of the model contains a single fully connected layer in Module 1 and Module 2.

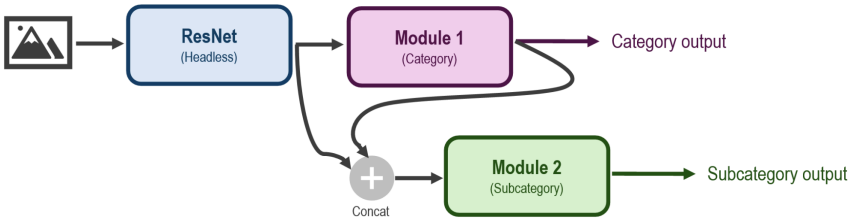


Figure 5: Structure of the fine-grained classification model

4.4. Recommendation Engine

The recommendation engine consists of four parts: collecting user actions, analyzer, recommender, and the final ranker. User actions are collected directly from the user interface based on outfit swipes and manual outfit creation. Then, the analyzer will combine that data with the already existing data pool to create an updated recommendation model. The recommender will output clothing combinations based on each clothing's vector representation. Then, the ranker will pick top recommendations based on user engagements.

We have decided to utilize a widely used recommendation system which employs a multi stage ranking process. The first stage of the process consists of selecting relevant items from multiple data retrieval sources. The main goal of this process is to pass the most relevant clothing pieces to ranking models.

The second stage consists of pre-generated and real-time ML based rankers. Pre-generated model uses personalized pagerank which captures the long-term interest. Real-time model uses two separate neural networks to capture the most recent interactions. One neural network is the user network, capturing the user's interest in outfits. The other network is the item network, producing consistent outfits. Then, similarity search is applied to user and outfit embeddings to propose top ranking outfits.

The last stage of the ranker orders the recommended outfits based on various metrics such as user's recent interest, events, and the weather. Then the results are presented to the user which allows them to like or dislike the recommended outfit. This collection of 0/1 data from users which helps us to train better rankers over time.

4.5. Matching Engine

The matching engine uses the classification model of the segmentation engine for feature extraction. All images in a user's wardrobe are passed through the classification model, during

which a feature vector is extracted by concatenating outputs of the second from the last layer that outputs category and subcategory. The feature vectors are converted to corresponding points on a multidimensional hyperspace. When a new image is inputted, the image is processed just like other items in the user's wardrobe to extract a feature vector. This vector is then projected on the hyperspace and based on the distance between the new image and other points, all clothes in the user's wardrobe are ordered according to similarity. The matching engine returns the closest point as the matching result and several other points as suggestions within the determined limit of max distance.

4.6. Data Storage

The data layer of the application is crucial for storing and managing persistent data. It includes a PostgreSQL relational database, chosen for its familiarity to the team and cost-effectiveness as an open-source system. Unless there are specific reasons against it, all data will be stored here. Additionally, an Azure Blob Storage account will manage clothing and profile images uploaded by users and those generated by the ML engines. The application will store all persistent data, except images, in the PostgreSQL database. Images will be stored in Azure Blob Storage for easier maintenance. The storage account will have three containers: one for raw clothing images, one for processed clothing images from the Segmentation Engine, and one for profile pictures. Each image will be named with a UUID, ensuring security by requiring a direct URL from the server for access. Raw images will also be tagged with the user ID for potential future data processing actions. Figure X below shows the current structure of the PostgreSQL database.

Figure 6: Diagram of relational DB

5. Test Cases

Details of functional and non-functional tests are given in the tables below.

5.1 Functional Test Cases

FT0001

Title: Create Outfit Manually
type/category: Functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Go to add wardrobe page	Wardrobe page opens		
Press the Add New Outfit button	Opens add new outfit dropdown		
Select create a manual outfit	Opens manual outfit creation page		
Select outfit size: top-bottom and shoes with coat option	Relevant outfits are shown to the user according to the selected size preference		
Select clothing preference for each segment	Clothings are set		
Press continue	The full outfit is displayed		
Enter name and tag	Entered details are set		
Press save	Outfit is saved		

Table 5: Manual Outfit Creation Test

FT0002
Title: Automatic Outfit Generation
type/category: Functional
Severity: Critical

Step	Outcome	Pass/ Fail	Resolution
Go to add wardrobe page	Wardrobe page opens		
Press add new outfit button	Opens add new outfit dropdown		
Select create AI-generated outfit	AI-generated outfit creation page open		
Select ocasion	Outfit recommendations are shown in a swiping view (Tinder-esque)		
Swipe left to say that you	Outfit is removed from the top of		

didn't like that outfit	the stack with an indication that it is disliked		
Press not interested	Outfit is removed from the top of the stack with a pop up indicating that this outfit won't be recommended any more		
Swipe right	Outfit is saved and selected		
Enter tag and outfit name	Added properties are set		
Press save icon	Outfit is saved		

Table 6: Automatic Outfit Generation Test

FT0003
Title: Event Creation
type/category: Functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Press event tab	Opens event page		
Press create new event	Event creation page opens		
Enter date, time and name of event and press create	New event created with said properties. Event should be seen in the week-view		
Open event details and select event dress code	Dress code for event set		

Table 7: Event Creation Test

FT0004
Title: Event Invitation
type/category: Functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Press event tab	Opens event page		
Select a pre-existing event	Details of the event is opened		

Press create invite link	Invite link is generated		
From another user's account, press the invite link	User joins the event		

Table 8: Event Invitation Test

FT0005
Title: Event messaging
type/category: Functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Press event tab	Opens event page		
Select a pre-existing event	Details of the event is opened		
Press chat	Event chat is opened, prior messages should be visible		
Write something and hit send	Message is sent to other users		
Send an image message	Image message is in the chat		
Send an outfit message	Outfit is in the chat		
Send clothing item as a message	Clothing item		

Table 9: Event Messaging Test

FT0006
Title: Register Clothing
type/category: Functional
Severity: Critical

Step	Outcome	Pass/ Fail	Resolution
Open wardrobe tab and press add clothing	Opens dropdown of camera and camera roll		
Select camera and take a picture	App asks for approval of the picture		
Approve the picture	Segmented image is received and with recommended tags and the		

	subcategory the outfit is in		
Edit properties and hit save	The clothing item is saved		
Do the same for addition by camera roll	The clothing item is saved		

Table 10: Register Clothing Test

FT0007
Title: Login
type/category: Functional
Severity: Critical

Step	Outcome	Pass/ Fail	Resolution
Enter user credentials to username and password fields	Signed into the app		
Signout	Opens login page		
Press sign in with Google and select the account prepared for the test	Signed into the app		
Signout	Opens login page		
Press sign in with Apple and select account	Signed into the app		

Table 11: Login Test

FT0008
Title: Signup by email
type/category: Functional
Severity: Critical

Step	Outcome	Pass/ Fail	Resolution
In the login page select signup and choose signup by email	Opens signup by email page		
Enter Email and password and press create	Systems asks for email authentication		
Open the email account and click the auth link	User is directed to personal details step		

Fill the properties and click continue	Account created		

Table 12: Signup by email Test

FT0009
Title: Signup with Google/Apple
type/category: Functional
Severity: Minor

Step	Outcome	Pass/ Fail	Resolution
In the login page select signup and choose signup by Google/Apple (Do the following steps for both) and select your account	Opens signup by email page with name fields filled according to your Single Sign-on provider		
Fill the properties and click continue	Account created		

Table 13: Signup with Google/Apple Test

FT0010
Title: Forgot password
type/category: Functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
On the login page, select I forgot my password.	App prompts you with email		
Enter email and press reset	An email for password reset is sent to your account		
Go to the mail account and press the reset link	Directs the user to a password change page		
Enter the new password and click reset	Password is reset		
Sign In with the old password	Access is denied		
Sign In with the new password	Login to the app		

Table 14: Forgot password Test

FT0011
Title: Select outfit for a day
type/category: Functional
Severity: Critical

Step	Outcome	Pass/ Fail	Resolution
Select today or a future date from the home page	Home page for the set date is opened		
Click add outfit	Dropdown of outfit selection options		
Click select from outfits	List of your outfits are shown		
select one of the outfits	Outfit is added to the date as worn		
Click add outfit	Dropdown of outfit selection options		
Click select from outfits	List of your outfits are shown		
Select build outfit	Redirects you to the outfit creation tab		
Do the same steps in both #T0001 and #T0002	Outfit is created and added as an outfit log to the date		
Click add outfit	Dropdown of outfit selection options		
Click mirror selfie	Opens the camera		
Take a picture of your whole outfit	Shows you the outfits you are wearing and shows you the clothing items in the picture		
Approve the outfit	New outfit is created and added as an outfit log to the said date		

Table 15: Select outfit for a day Test

FT0012
Title: Viewing the wardrobe
type/category: Functional
Severity: Critical

Step	Outcome	Pass/	Resolution

		Fail	
Press wardrobe tab	Opens user wardrobe, should be able to see clothing items (separated by categories) and outfits		
Press favorites icon	Only favorited items and outfits shown		
Press favorites icon again	All items and outfits are again shown		
Click on an outfit	The detailed page of the outfit is opened. Must see tags, category, date it was last worn, is_favorited, how many times it was worn, colors, seasons, texture		

Table 16: Viewing the wardrobe Test

FT0013
Title: Deleting Clothing
type/category: Functional
Severity: Critical

Step	**Outcome**	**Pass/ Fail**	**Resolution**
Press wardrobe tab	Opens user wardrobe, should be able to see clothing items (separated by categories) and outfits		
Click on an clothing that is used in an outfit	Opens outfit detailed view		
Press delete icon	User is prompted with the clothing being in an outfit.		
Click yes	Clothing is deleted		

Table 17: Deleting Clothing Test

FT0014
Title: View analytics
type/category: Functional
Severity: Major

Step	**Outcome**	**Pass/**	**Resolution**

		Fail	
Go to profile	Profile opened		
Press analytics icon	Analytics page opens with the week view. Users should be able to see most/least worn clothes, most worn colors, pie chart of outfit categories in wardrobe, view new outfits/clothes added, wardrobe utilization for the week.		
Press monthly tab	Analytics page opened with a month view, and users should be able to see most/least worn clothes, most worn colors, pie chart of outfit categories in wardrobe, view new outfits/clothes added, wardrobe utilization for the week.		

Table 18: View analytics Test

FT0015
Title: Manage profile
type/category: Functional
Severity: Minor

Step	Outcome	Pass/ Fail	Resolution
Go to profile	Profile opened		
Press edit	Profile fields are now editable		
Edit body type and style gender and press save	Fields should now be updated		
Click the profile picture	Profile picture opened in the detailed view		
Click edit and add a new picture and hit save	The profile picture is updated		

Table 19: Manage profile Test

FT0016
Title: Clothing Item Recommendation
type/category: Functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Open the outfit recommendation tab	The outfit recommendation tab is opened		
Select pieces of clothing and hit recommend	The app suggests items that would go well with those items.		
Edit body type and style gender and press save	Fields should now be updated		
Click the profile picture	Profile picture opened in the detailed view		
Click edit add a new picture and hit save	The profile picture is updated		

Table 20: Clothing Item Recommendation Test

FT0017
Title: Outfit view
type/category: Functional
Severity: Critical

Step	Outcome	Pass/ Fail	Resolution
Open wardrobe tab	Wardrobe tab opened		
Press outfits	Outfits of user are shown		
Click the topmost outfit	Outfit details are opened		
Press edit and change the tags of the outfit than hit save	Outfit details are edited		
Go back to the outfits view	Outfits of user are shown		
Click the favorited icon	Only the favorited outfits are shown		
Click favorited icon	All outfits are shown		

Table 21: Outfit view Test

FT0018
Title: Settings
type/category: Functional
Severity: Critical

Step	Outcome	Pass/ Fail	Resolution
Press settings icon	Opens settings page		
Turn off/on notifications	Notifications are turned off/on		
Click contact information	Contact information is shown		
Click security preferences and make profile public	Profile is visible to everyone		
Click terms and conditions	Shows terms and conditions		

Table 22: Settings Test

FT0019
Title: Polling
type/category: Functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Open events tab	Event tab opened		
Click the most recent event and open chat.	Event chat opened		
Click poll and select 4 dresscodes for the poll	Poll is created		
Vote in the poll	Vote added to the selection		

Table 23: PollingTest

FT0020
Title: Deleting Outfits
type/category: Functional
Severity: Critical

Step	Outcome	Pass/ Fail	Resolution
Go to wardrobe page	Open wardrobe page		
Click outfits	Outfits tab open		
Click the topmost outfit	Outfit detailed paged opens		
Click the delete icon and	Outfit deleted		

approve it			

Table 24: Deleting Outfits Test

5.2 Non-functional Tests

5.2.1 Usability

All usability tests will be evaluated according to the number of tasks completed successfully, and according to the task times.

$$Effectiveness = \frac{\text{Number of tasks completed successfully}}{\text{Total number of tasks undertaken}} \times 100\%$$

$$Time\ Based\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR}$$

Fig. 7: Usability Metrics [6]

NFT001
Title: Usability Test
type/category: Non-functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
100 users start investigating our 17 functional requirements	At least 90 out of 100 users must complete all 17.		
Calculate time spent for each functional requirement	Users must complete under 5 minutes each task.		

Table 25: Usability Test

5.2.2 Reliability

NFT002
Title: Reliability Test for Backend
type/category: Non-functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Check reliability information of uptime and reliability of AWS server.	The backend should not be down for more than 10 minutes.		

Table 26: Reliability Test for Backend

NFT003
Title: Reliability Test for Machine Learning Models
type/category: Non-functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Run the test batches for each machine learning.	Test accuracy results of the machine learning models should be higher than 85%.		

Table 27: Reliability Test for Machine Learning Models

5.2.3 Performance test

NFT004
Title: Performance test for Outfit recommendation
type/category: Non-functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Setup 100 clothing items and 20 outfits	-		
Start a timer and request an outfit recommendation	Outfits must arrive in less than 2.5s		

Table 28: Performance test for Outfit recommendation

NFT005
Title: Performance test for Wardrobe view
type/category: Non-functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Setup 100 clothing items and 20 outfits	-		

Start a timer and open wardrobe page	Items should load in less than 1.5 seconds		
Rapidly swipe up and down on the page to load different items continuously	The app should scroll smoothly and the new items should be loaded at most in 1 seconds		

Table 29: Performance test for Wardrobe view

NFT006
Title: Performance test for Single item clothing registration
type/category: Non-functional
Severity: Major

Step	Outcome	Pass/ Fail	Resolution
Open clothing registration page	Page opened		
Take a picture of an outfit lying on the bed and send the segmentation request	The segmented and labeled image should return in less than 3 seconds		
Start the timer again and hit save	The item should be saved in less than 1 seconds		

Table 30: Performance test for Single item clothing registration

6. Consideration of Various Factors in Engineering Design

As the owner of a B2C product, we need to take into consideration what constraints are enforced upon us and what limitations we should keep in mind. Furthermore, we must also standardize our work in order to maintain uniformity across multiple members of the project.

6.1 Constraints

6.1.1 Public Health and Safety

We must ensure that our application does not compromise the safety and well-being of our users. For example, we should ensure that user data, including images, are handled carefully and that privacy and security measures are in place.

6.1.2 Data Security

To prevent unauthorized access to user data, data need to be protected both at transit during client-server communication and at rest on the server side. Necessary authentication and authorization systems need to be implemented, and data should be sent to the client after ensuring the client is the owner of the requested data.

6.1.3 Data Privacy and Regulations

Our app should respect the privacy of users' data and comply with related local regulations, such as the General Data Protection Regulation (GDPR) for Europe and, for example, the localized Turkish version, KVKK. Personally Identifiable Information (PII) of users should not be collected unless required to provide our service, and only a minimal amount of information needs to be collected and stored. In addition to the transparency about which data is collected and how it is used, collected information should be permanently deleted upon the request of a user.

6.1.4 Ethical Limitations

Any data collection or processing that may lead to ethical uncertainties should be prevented, and ethical considerations should be in focus throughout the project's development. For example, while users need to send their photos to servers for processing and extraction of clothing-related data, having personal photos of users on the server in a human-readable form would lead to ethical and privacy-related concerns. To overcome this issue, faces can be blurred before being transferred to the server for processing in order to minimize the collected PII, increase user privacy, and resolve ethical concerns..

6.1.5 Computational Limitations

As the processing of user data will either be done locally on the user's device or remotely on our servers, the amount of data to process will be limited by the computational capabilities of those machines. Provided service will depend on the time of computation and the virtual space, memory, or disk space spared for the computations. Recognizing that the AI engine will perform demanding tasks to recognize and categorize user images, this constraint will be significant throughout the development of Capsule.

6.2 Standards

Standards are important for a long lasting software project as it makes it more robust and consistent. Furthermore, it makes it so that all members of the team are on the same page. For Capsule we are using various standards for the reports, diagrams and the codebase.

We are following IEEE standards for requirements documentation, employing IEEE 830-1998 to systematically capture and manage project requirements. We are using IEEE for our citations as well. Additionally, we utilize UML 2.5.1 as the standard modeling language for visualizing, specifying, and documenting various aspects of system design. In terms of code formatting, the "Black" formating standard is used for Python code, ensuring adherence to a strict coding style guide for enhanced readability and maintainability.

6.3 Factors

6.3.1 Sociocultural Factors

Clothing choices can vary greatly across cultures and societies. We need to be sensitive to cultural differences and social norms to provide respectful and appropriate recommendations for a diverse user base.

6.3.2 Environmental Impact

The fashion industry has a significant environmental footprint. Capsule aims to reduce unnecessary clothing consumption, but we should also consider the environmental impact of our solution, such as promoting sustainable clothing brands and practices.

6.3.3 Economic Factors

By helping users make more informed clothing choices, we aim to reduce unnecessary spending. However, we should be aware of the potential economic impact on the fashion industry and consider strategies for sustainable growth and partnerships.

6.3.4 Global Reach

Capsule has the potential to reach a global audience. We must take into account the diverse fashion preferences and needs of users from different parts of the world and adapt our system accordingly.

	Effect level (0 - None to 10 - Maximum)	Effect
Public health and safety	9	We must prioritize user data security and privacy to ensure the safety and well-being of our users. Implementation of robust security measures is essential.
Sociocultural factors	7	We need to be sensitive to cultural differences and social norms to provide recommendations that are respectful and appropriate for a diverse user base.
Environmental factors	8	Capsule aims to promote sustainable clothing practices. We must consider the environmental impact of our solution, such as endorsing eco-friendly clothing brands and responsible consumption.
Economic factors	5	While helping users make informed clothing choices, we need to be aware of the potential economic impact on the fashion industry and consider strategies for sustainable growth and partnerships.
Global factors	7	Capsule aims to serve a global audience. We must consider the diverse fashion preferences and needs of users from different parts of the world and adapt our system accordingly.

Table 31: Global Reach

7. Teamwork Details

The capsule team is composed of 5 members from various backgrounds and interests. Our members are listed as follows: Ali Emir Güzey, Alp Afyonluoğlu, Ceren Akyar, Deniz Mert Dilaverler, and Mehmet Kağan İlbak.

7.1 Contributing and functioning effectively on the team

Ali Emir Güzey

He contributes to the backend development of the project. He is also responsible for the infrastructure-related tasks such as Docker deployment and choosing and integrating necessary cloud products.

Alp Afyonluoğlu

He contributed to the machine learning side, working on segmentation models, implementing the hierarchical multi-item segmentation model and related dataset-processing tools, and fine-grained classification model.

Ceren Akyar

She contributed as the UI/UX designer and designed both the landing page and the user interface of the mobile app. Also, she contributes to the development of the mobile app as a frontend developer.

Deniz Mert Dilaverler

Deniz worked on the backend of the system, designing the schema and APIs for the frontend to consume. He worked on the endpoints for user registration, authentication and also wardrobe side of the app like CRUD requests of clothings and outfits.

Mehmet Kağan İlbak

Kağan worked on machine learning tasks and on the mobile application. On the machine learning side, he contributed to the initial segmentation pipeline and the recommendation pipeline. He also helped rewrite the application in platform native APIs.

7.2 Helping creating a collaborative and inclusive environment

Our top priority has always been creating a collaborative and inclusive environment since diverse perspectives and experiences will bring new ideas to light. We have implemented several strategies to foster this environment.

We have set up recurring weekly meetings where each member freely expresses their ideas and feelings about the project. These meetings served as a dedicated space for dialogue where each team member actively participated. Furthermore, we also picked a physical headquarters for our meetings, which is in Dorm 82. This physical gathering place allowed us to have a sense of belonging since we have spent most of our time here. We distributed the

workload according to everyone's relative expertise so that everyone was working on a task they liked.

Overall, with these measures, we have achieved our initial goal of creating a collaborative and inclusive environment. Each team member felt valued, empowered, and motivated to contribute their best work.

Ali Emir Güzey

He helps the team through code reviews. He also encourages his teammates with positive feedback. He has a camera, which he uses to make his teammates happier by recording their memories. He also created a shared keychain on Bitwarden where we keep our secrets (e.g. API keys, environment variables) securely. Lastly, he created and shared a custom Neovim configuration to increase his teammates' productivity.

Alp Afyonluoğlu

He worked 1-on-1 with team members to collaboratively integrate ML models with other modular app parts, such as the backend and the web scraper, He documented ML-related parts of the repo with readme files for maintainability and easier understanding by the teammates. He also helped the team with transportation when possible to prevent unproductive time loss during transportation for the whole team.

Ceren Akyar

She designed the user interface for the team to implement collaboratively. She also tries to motivate unmotivated team members and tries to help them with their problems.

Deniz Mert Dilaverler

Deniz distributes tasks for the backend team and communicates with the frontend team for API creation. Furthermore, he documents the endpoints the team creates for the requests to be easily consumable. He also distributes tasks to the team for each report.

Mehmet Kağan İlbak

Kağan helped with the overall synchronization of various subsystems. He contributed to implementation strategies of subsystems such as ML and mobile. He encouraged team members to conduct code reviews with cute GIFs. He also has a camera, allowing him to contribute to Ali Emir's perspective. He also increased the happiness of team members by removing JavaScript from the project.

7.3 Taking lead role and sharing leadership on the team

Ali Emir Güzey

He is the lead responsible for system infrastructure tasks.

Alp Afyonluoğlu

He is the lead responsible for segmentation and matching parts of ML.

Ceren Akyar

She is the lead front-end developer and UI/UX designer.

Deniz Mert Dilaverler

Working as the lead backend engineer and report manager of the team.

Mehmet Kağan İlbak

He is the lead responsible for ML infrastructure and the recommendation system.

8. References

[1] E. Newbery, “The Average American Spends This Much on Clothes Every Year,” the ascent, Jul. 26, 2022. [Online]. Available: https://www.fool.com/the-ascent/personal-finance/articles/the-average-american-spends-this-much-on-clothes-every-year/. [Accessed: Oct. 15, 2023].
[2] E. Penner, “The Ultimate Guide: How to Build a Capsule Wardrobe,” Modern Minimalism, 2023. [Online]. Available: https://modernminimalism.com/how-to-build-a-capsule-wardrobe/. [Accessed: Oct. 15, 2023].
[3] “Imaterialist (fashion) 2019 at FGVC6,” Kaggle, https://www.kaggle.com/c/imaterialist-fashion-2019-FGVC6/data?select=label_descriptions.json (accessed Mar. 14, 2024).
[4] Ternaus, “Ternaus/cloths_segmentation: Code for binary segmentation of cloths,” GitHub, https://github.com/ternaus/cloths_segmentation (accessed Mar. 14, 2024).
[5] Bearpaw, “Bearpaw/clothing-co-parsing: CCP dataset from "clothing co-parsing by Joint Image Segmentation and labeling " (CVPR 2014),” GitHub, https://github.com/bearpaw/clothing-co-parsing (accessed Mar. 14, 2024).
[6] J. Mifsud, “Usability metrics - a guide to quantify the usability of any system,” Usability Geek, https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/ (accessed Mar. 14, 2024).