EUROPEAN UNIVERSITY OF LEFKE

FACULTY OF ENGINEERING

Graduation Project 2

AI Fashion Assistant

Duygu Gücüyetmez

21120382

Short Introduction;

My project is shortly a fashion website. It's a place that you can look at other fashion brand products with filters and they can track discounts or stock of this products also users can upload their outfits and others can see them and with filters the other users can take inspirations as they want. The main idea is that users can look at products and take inspirations that how can they combine their outfits in the same place.

Users can use this application for looking products from different brands at the same place, using filters and see the same pieces in different types or prices. Also with similarity filter they can found a product with an image like exact same or similars so no more losing time with searching the products they see somewhere. They can track the products stock or discount to buy the product when their desired price come or before it out of stock. Also they can get inspired by other users outfits or they can upload their outfit and show their style. Also when they want to style a clothes item with similarity filter they can see the outfits with similar piece. The main goal is making a website for easy shopping and effortless outfit creation.

Supervisor

Vesile Evrim

09.06.2025

Table Of Contents

1. Introduction

1.1 Problem definition

My project briefly is giving users a place that they can see a lot of different brands products at the same place, if they like show their favorites and give them tracking options like stock or discount, also let them to upload their outfits for sharing other people so they can give inspiration or take inspiration from others. The most important part is a user can use product similarity filter so when she likes an clothes item somewhere else she can found it or similar and also with outfit similarity filter she can found outfits that made of with that clothes item.

First of all, we must accept that many people shape their shopping habits according to popular culture, so like they make a summer shopping based on what’s popular for this summer. In fact shopping has become a popular culture, trends always changing and you should buy what’s popular (Melek, 2012) . The most important medium that creates this popular culture is social media and the influencers in it. According to a survey with 300 people, %89 of people said influencers sharings in social media shape their fashion shopping behavior (Pınar,2025) . My website has features that allow people to directly search for products so no more wasting time to try to found the product link in your favorite influencer or wait the time she share the link. Also you don’t need to buy the linked product which cost too much. According to researchs, similarity searches with images allow users to make faster decisions meaning they don’t need to waste time with shopping, %74 of people say that they’re using image searching when they’re shopping (Joydeep) .

Another feature of the website is that users can upload an image of a product and with the outfit similarity filter she can explore outfits that built with that product. The outfits will be uploaded by users so my website can appeal to people of all clothing styles. So no more time wasting for thinking how can i combine it or buying something that you can actually couldn’t combine. Under the influence of popular culture , people can make unnecessary purschases for example they can buy products that are not their style or they cannot combine an outfit with that product because their closet is not enough for that. (Fatih & İpek, 2022). Due to the Diedrot effect , people feel psychologically the need to buy other products that they can combine with that clothes item and they are forced to buy even more products just because they bought a product without thinking (Wikipedia, 2025)

Lastly with my stock and discount tracking system users notified automatically when their desired price came or the product in the stock or lower the stock. So shopping will be more easy and cheaper.

All my project aim is giving users a fashion place like they can found everything they want like a big online store that has a lot of different brands so a lot of different options. And this online store has a fashion assistant that will notify you when your desired price come or the one clothes item you like comes to the store. And when you show an clothes item to her she is show you the exact same and similars with different options so you can buy a product more cheap or better material. Also she shows how you can combine this outfit too so you don't have to waste your time with thinking your outfits or thinking can i combine this product with the products in my closet.

So my application gives users a great way to shop more conveniently, smartly and without wasting time.

Example-Problems :

- First of all generally a lot of brands have their own websites , and the applications maded with a lot of different brands will give the user the same product a little higher. Like trendyol is a website with having products with a lot of different brands but users have a lot of complaint about a same product is more expensive in the Trendyol. But we're directly giving the brand's website url so users don't need to worry about it .

- My website users can search the product with similarity filter and see outfits that builded with that product. So if the outfits with the product isn't their style or they don't own other pieces for making an outfit they will be not buying. Other brands using the diedrot effect for making their profit more by the power of social media and influencer advertisements. When users buy something they actually couldn't combine with their products or just brand or social media suggest that with the product you buy, this other products will be combined very beatiful. So user have to buy them too for feel more comfortable and shopping cycle never ends (Cyberclick, 2024).

- A lot of different shopping website doesen’t have tracking system for users selected outfits. So my website has this feature a user can start to track their wanted products, they can also select what the price they want to for getting notified.

1.2 Goals

- **Goal:** Making Visually Outfit Find Filter

  Purpose: The purpose of this goal is when a user want to buy a product or want to combine her product give outfits that built with the same product. There is a place in the website user can upload images for using similartiy filter, in that place there’s also different filters like style category etc. So user can use the other filters for making a more spesific search. So they can see different outfit choices with the the product.

  Benefits: Users shouldn’t give a lot of time for combining a product or finding an outfit to wear. Also when users want to buy a product they can look at the outfits that builded with this product and the outfits doesen’t include their owned products or not their style they will be not buying an unecessary products.

- **Goal:** Making Visually Product Finder Filter

  Purpose: The purpose of this goal is taking different clothing brands data’s and show it at one place. Then there is a place that users can upload images for this similarity filter, there is different filters too like price, color etc. So user can use this filters too for more spesific search. So they can find product’s same or similar versions they’re looking for

  Benefits: Prevent users from wasting time trying to finding which site sells a product that they see somewhere and like. Also giving similar choices for the same product so they can buy cheaper or more quality version of the product.

- **Goal:** Making Stock and Discount Tracker

Purpose: The purpose is when a user favorited a product she can also start discount or stock tracking. They can give the discount price range. When stock stiuation change or the discount comes to the users wanted price range they will automatically notified about it.

Benefits: Users don't need to follow discounts or stocks, they don't need to constantly look at other sites for this. With my website they can notified when their desired price came or access the products before they run out.

2. Literature Survey

Trendyol: This place gives a lot different brands products too and it has visual search tool too. But in here they don't give the products actual brand url's and they sell it more expensive from the actual brand. Also yes users can favorite products and get notified but it's only sometimes and it's for stock stiuation but they just warning about it is low in the stock. My website warn about when it's the stock, low in the stock etc and also get notified quickly. Also can track a product discount with their decided price range and they will automatically notified.

Google Lens: In here users can upload some image and look at the places here can they buy it and they have webstie and application too. But we have just website and they don't have tracking system, also when in our website when users try to use similarity filter they can use other filters too. So this way their search will be more practical causr they can just decide the color or price range. But google lens just show the same clothes images and their url's so you need to click every url for selecting the best. But in our website they can just use filters for the best product.

Pinterest Lens : It's has similarity image filter too like our application. But usually it doesen't even provide information about where the product is sold. So you just see the product's exact same or similar images. And if you want to get outfit inspiration usually you can't filter with the product image cause it's just gives the product's similar or exact same versions. U can search style category or write the product's color and category name and get inspiration from the giving outfits. But in my application we have 2 different place for this 2 different thing. So a user can upload a product image and find similar ones with a lot of information. Or user can upload a product for find outfits with built with it.

3. Background Information

3.1 Required & Used software

- **React :**
  For developing my frontend.

- **Node.js:**
  For developing my backend.

- **Python :**
  For developing the artificial intelligences.

- **CLIP:**
  For analyzing the images that users uploaded to generate similarity filters for the uploaded outfits or products. After analyzing the images it gives the embedding of the images

- **FAISS:**
  It compares the embeddings of the products or previous uploaded outfits with the embeddings of the images uploaded by the users for using the similarity filters and it gives the similar results to users.

- **MediaPipe Pose:**
  For when users upload their outfit images it detect the body parts so making sure the uploaded image will be a human and a outfit image.

- **NSFW:**
  For prevent users from uploading inappropriate photos into the website.

- **Visual Studio Code :**
  For managing my development process with a code editor so it will be more easy.

- **MYSQL :**
  For my database.

3.2 Other software

- **Icon8 :**

  For designing icons of my application.

- **Mercurial :**

  Used for repository of my application.

- **Zara API by Apify :**

  For getting product data’s.

- **ZAPSPLAT :**

  For getting my application’s notification sound.

- **Canva:**

  For making the picture poster of my Landing Page.

4. Design Documents

4.1 Data flow diagram

A) This is the data flow diagram of the process that when a user want to uploading an outfit.

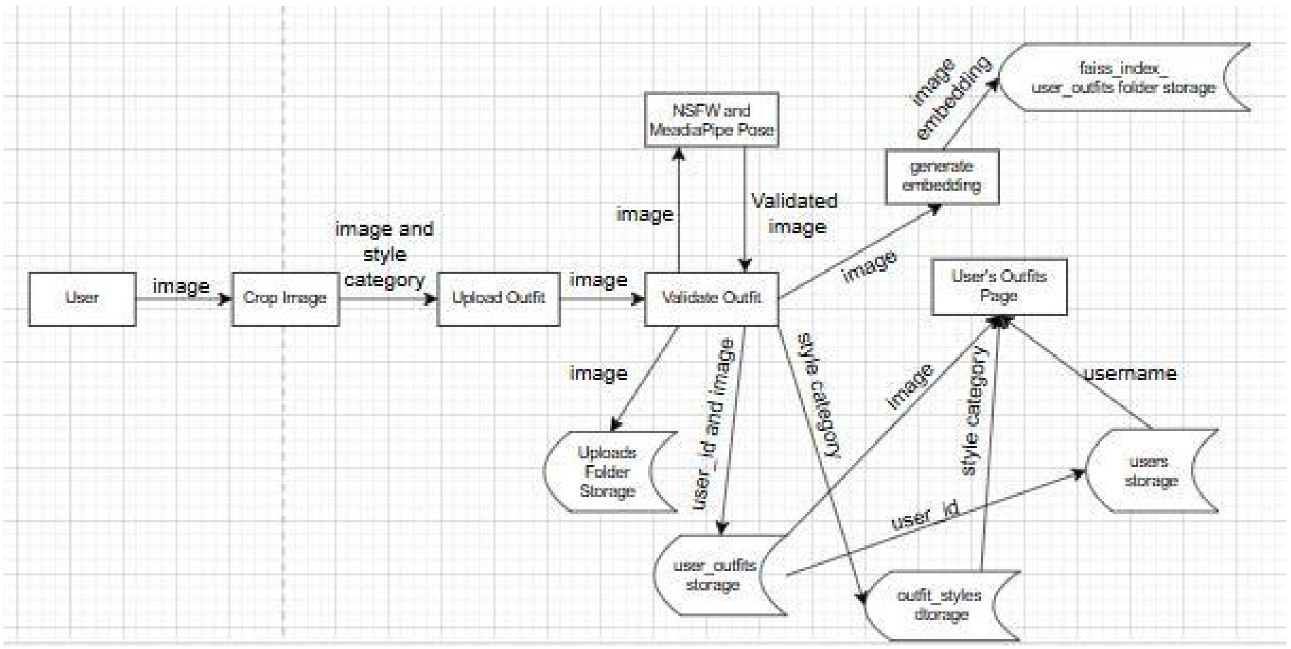
First of all user takes picture or take an image from the gallery. Then automatically cropping feature came if user want to crop the image they can crop. Then before uploading an image users must choose an outfit style. Then user press the upload button.

Then image go to the validate outfit feature in here with nsfw we check the image if it’s inappropriate. And with MediaPipe Pose we check if the image is a human image and is it include an outfit. If the image is okey for this two ai it will be pass the validate outfit process.

Image sended to the generate embedding for taking the image's embedding. Then it's sended to the faiss_index_user_outfits folder for storing because later we will be using it.

We send this image to the uploads folder for giving it a concrete place because the database want url so having an url we should have a concrete place. And user id and image send to the MySQL's user_outfits table. And style category sen to the MySQL's outfit_stles table.

So when an image uploaded by user we can see it in User's Outfit Page. With getting the image from user_outfits, style category from outfit_styles and user_outfits send the user_id to users and finally we get username from users.

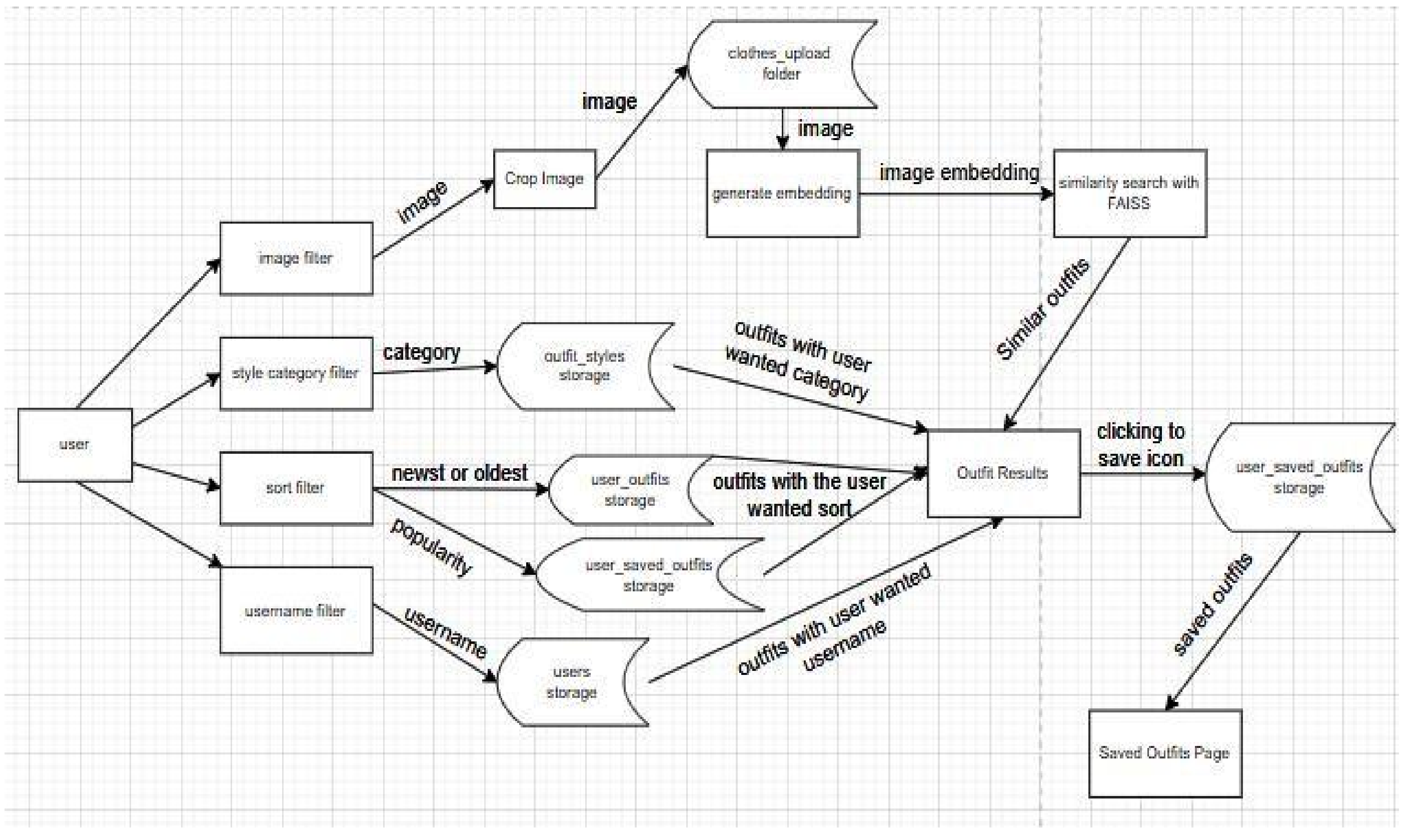


B) This is the data flow diagram of the process that when a user go to the User's Outfits page how she is used filters and saving outfits.

First of all if user image filter. She will takes picture or take an image from the gallery. Then automatically cropping feature came if user want to crop the image they can crop. Then this image goes to the clothes_upload folder. Then we generate the image's embedding. Then FAISS will be looking to the user's uploaded outfit images embeddings and the embedding of the user uploaded image for using the filter. And it will give the most similar outfits. And the result is showed.

The other filters other MySQL database and we will be giving user their wanted outfits. And the result is showed again.

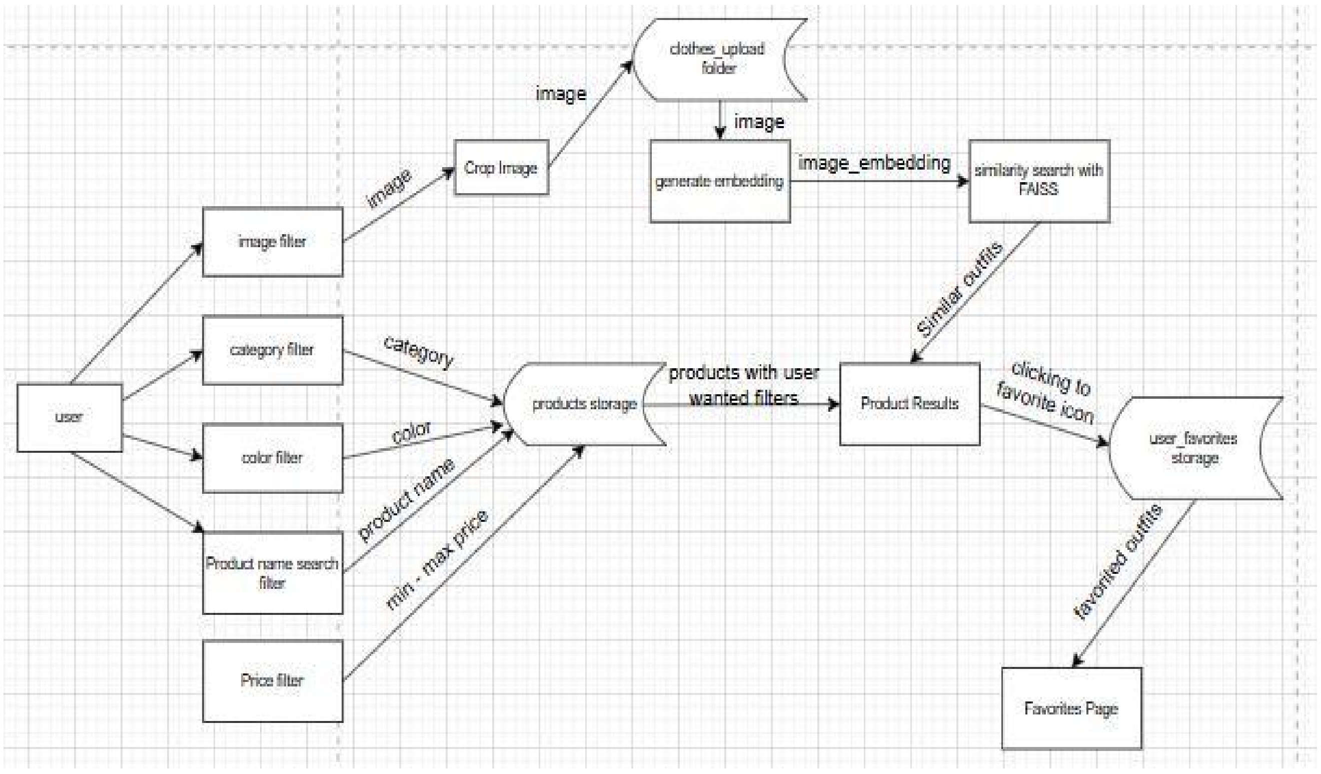
When we see the outfit results we can click the saving icon and it's go to the user_saved_outfits with that we can see our saved outfits in a place what is name is Saved Outfits page.



C) This is the data flow diagram of the process that when a user go to the Find Clothes page how she is used filters and saving outfits.

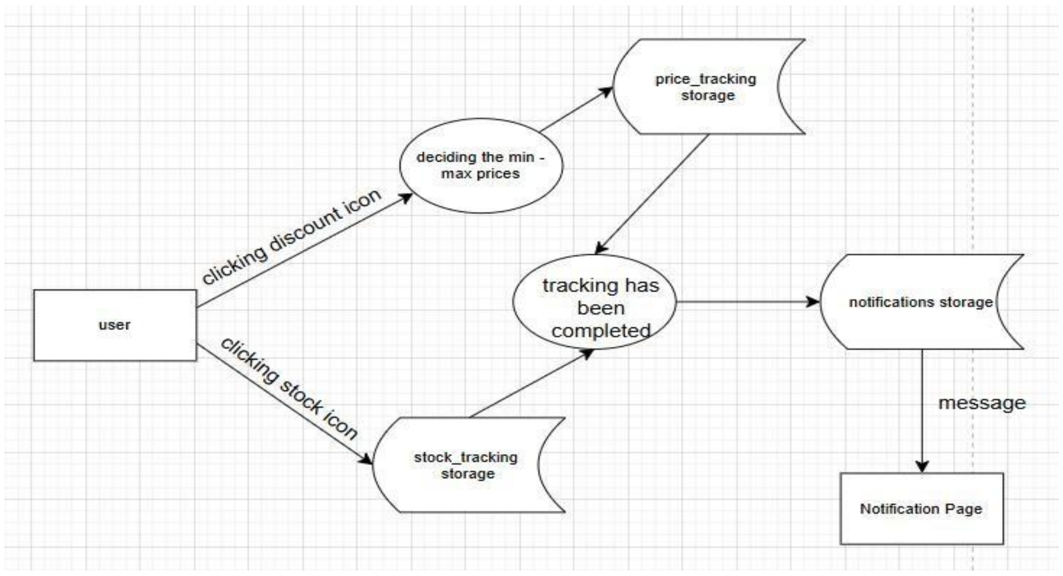
We will be doing the same thing in this place. Just right now our datas will be products not outfits. And just the similarity search's data's will be different because in here FAISS will be looking to the products embeddings.

The other filters names are different. All the datas comes from products table for giving the user wanted products. Then the icon is different for this page it is favorite icon. When u clicked it is go to the user_favorites table so you can look at them in Favorites page.

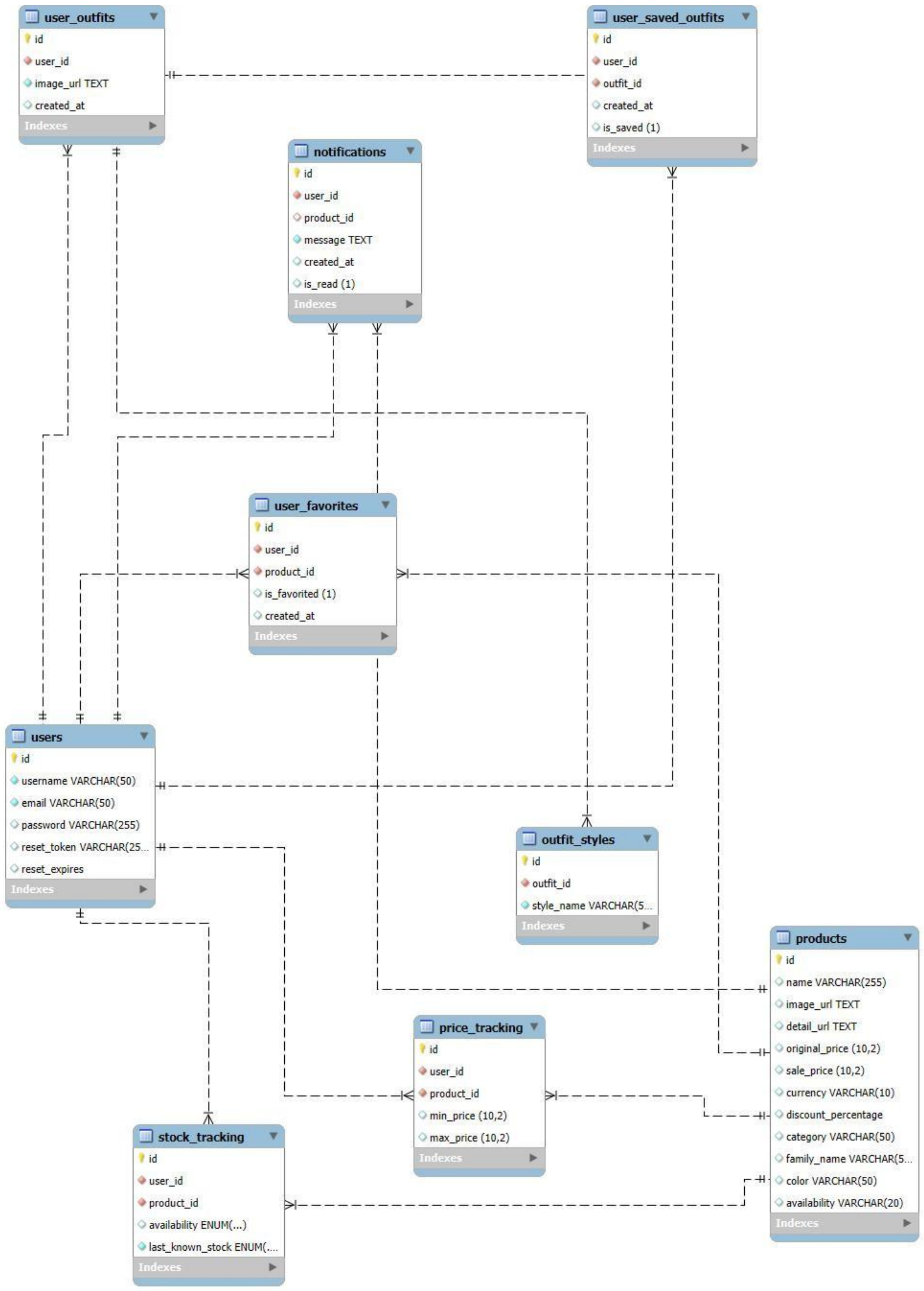


D) This is the data flow diagram of the process that when a user go to the Favorites - Notifications pages how she is can use trackings and get notified.

You can click discount icon you will be giving min-max price. It’s go to the price_tracking table. When the tracking is completed like the clothes item comes to the stock again it goes to the notification page and we will be take the notification message at the notification page.


price_tracking storage
deciding the min - max prices
clicking discount icon
user
tracking has been completed
notifications storage
clicking stock icon
message
stock_tracking storage
Notification Page

4.2 ER Diagram



5. Methodology

System Architecture:

In this project i use modern software languages, Iterative SDLC method and multi-layer architecture.

Iterative SDLC Method:

I use it on this project. I choice this because it reduce risk because you do one thing and then testing so you see the errors early. In every step i made;

1. Database table
2. Frontend
3. Backend
4. Testing until eveything seems perfect
5. I keep repeating the steps between 1-4 then did one last testing
6. I make AI integration’s into my project
7. Testing until everyhing is okey

Layers:

I use multi-layer architecture for this project cause every layer do their jobs and when i want to change something it will be more easy to handle like in the short it’s more easy to manage and develop.

1. Database Layer
2. Backend Layer
3. Frontend Layer
4. AI Integration Layer

General Structures Of Layers:

1. Database Layer:

This is the place that keeps our datas. The database and it’s tables are maded in MysSQL. It’s controlled by backend like when a user upload something backend take the uploading information and then give it to the database layer. Or when we need to show the datas in the frontend the backend give it from the database layer.

In here some tables will related to each other to making sure the data consistency. Like a lot of table is related with users tables because everything that user does is gets to data with user_id. Whenever we want to show this data into the frontend we sure this datas for that user. Like users can save outfits, so every user’s saved outfit page is should be show different datas.So right now we will be looking to the tables:

A) users table:
This is the place where we hide user informations. It’s id is related with most of the other tables like user_outfits,notifications,user_saved_outfits, user_favorites, stock_tracking and price_tracking tables user_id is related with users id .

B) user_outfits table:
This is the place where we keep the data of outfits except the outfit style data, when user upload an outfit the infos comes to this page. It’s id is related with user_saved_outfits and outfit_styles tables outfit_id. And of course user_id in this table related with users table’s id.

C) outfit_styles table:
In here we just keep the outfit style data of that uploaded outfit. It’s outfit_id is related with user_outfits table’s id.

D) user_saved_outfits table:
In here when a user save an outfit the data of it keeps in here. If is_saved is 1 the outfit saved and if it’s 0 then the outfit saved first but we removed it from the saveds. It’s user_id is related with users table’s id and the outfit_id is related with user_outfits table’s id.

E) products table:
In here we keep the products data that we took from as json format from an application. It keeps a little too much data because we keep it from as an json format from somewhere else but the important data’s are id, image_url, detail_url (where the products are actually sold), name, sale_price, currency(TL or USD), category ,color and availability (it’s the stock’s stiuation). But i keep the other data’s because i want to get upgraded my project later anyway. It’s id related with a lot of tables products_id like user_favorites,notifications,price_tracking and stock_tracking tables.

F) user_favorites table:
In here we keep the user’s favorited products. If is_favorited is 1 then it is favorited if it’s 0 it was favorited before but later we remove it from the favorites. It’s user_id is related with users table’s id and it’s product_id is related with products table’s id.

G) stock_tracking table:
In here if a user start stock tracking of a product we keep the stock stiuation it’s if last_known_stock and availability is different then the tracking will be completed and we notify the user. It’s related with products and users table’s id.

H) price_tracking table:

In here if a user start discount tracking of a product we keep the desired price range and if the price of a product cames between something in the range we notify the user. It's related with products and users table's id.

I) notifications table:

In here we keep the notification informations. Message is prepared in the backend code for different stiuations when a tracking complete the message of it, the time of that notification store and the information that user did read the notification message or not in is_read (if 0 not readed if 1 is readed) in this table.

2. Backend Layer:

This place is like a bridge of between our frontend and database. It's developed by node.js for AI handling it maded with python.

This code in the below is our main backend code we run the backend in our terminal with this code. So i will be explaining it a little deeply.

```
const express = require("express");

const cors = require("cors");

require("dotenv").config();


const app = express();
```

This is the place where w make the CORS settings. Making our localhost and then deciding which methods could use. And just accepting with the allowedHeaders requests.

```
app.use(

  cors({

    origin: "http://localhost:5173",

    methods: ["GET", "POST", "PUT", "DELETE", "PATCH"],

    allowedHeaders: ["Content-Type", "Authorization"],

    credentials: true,

  })
);
```

This is the place for handling json datas. It can’t be more than 50 mb and it’s parse the json datas so backend can understand.

```
app.use(express.json({ limit: "50mb" }));

app.use(express.urlencoded({ extended: true, limit: "50mb" }));
```

In here with const authRoutes or outfitRoutes place we take the routes from their files and then the place that app.use thing we tell the express server where and how to use them. Like if in the frontend somebody want to see their saved outfits, the server knows exactly which route file to use and with that we give user’s requests.

```
const authRoutes = require("./routes/auth");

const outfitRoutes = require("./routes/outfitbabe");

const savedOutfitRoutes = require("./routes/savedoutfits");

const suggestionsRoutes = require("./routes/suggestions");

const validateRoutes = require("./routes/validate");

const productRoutes = require("./routes/products");

const favoriteRoutes = require("./routes/favorites");

const trackingRoutes = require("./routes/tracking");

const runAllTrackingChecks = require("./utils/checkTracking");

const notificationsRoutes = require("./routes/notifications");

const homeUploadRoutes = require("./routes/homeupload");

app.use("/api/auth", authRoutes);

app.use("/api", outfitRoutes);

app.use("/api", suggestionsRoutes);

app.use("/api", savedOutfitRoutes);

app.use("/uploads", express.static("uploads"));

app.use("/clothes_upload", express.static("clothes_upload"));

app.use("/api", validateRoutes);

app.use("/api", productRoutes);

app.use("/api", favoriteRoutes);

app.use("/api", trackingRoutes);

app.use("/api", notificationsRoutes);

app.use("/api", homeUploadRoutes);
```

It’s controls the tracking datas for every 5 minutes so we can notified users when it’s needed.

```
setInterval(runAllTrackingChecks, 5 * 60 * 1000);
```

So our backend runs in 5000 port we write backend works for see in the terminal that backend is working.

```
const PORT = 5000;

app.listen(PORT, () => {

  console.log(`Backend works !!!`);

});
```

So in short our backend helps to giving data’s or taking new data’s as users requests.

You can see in the below code it’s helps to give the user of their outfits. It takes the datas with the query from database and then send it back it as json format.

```
router.get("/user-outfits", verifyToken, async (req, res) =>

  { try {

    const userId = req.user.id;


    const [outfits] = await pool.query(

      `SELECT

        uo.id AS outfit_id,

        uo.image_url,

        uo.created_at,

        os.style_name

       FROM user_outfits uo

       LEFT JOIN outfit_styles os ON uo.id = os.outfit_id

       WHERE uo.user_id = ?

       ORDER BY uo.created_at DESC`,

      [userId]

    );

    res.status(200).json(outfits);

  } catch (error) {

    res.status(500).json({ error: "Server error." });

  }

});
```

In our backend we have 2 middlewares because it reduce the code repeating.

verifyToken: It’s controls the JWT token and if it’s okey than the user can do the thing they want.

multer: It’s takes the images that get it from users and send it to the uploads folder.

3. Frontend Layer:

This is the place that users see and interact with our system. It’s developed by React for making a user friendly and easy to develop. For making the style of the frontend we use css because it’s a classic and easy to develop. Thşs place takes the user’s requests and sen it to the backend then take the datas from backend and give it to users with a user friendly view way.

This code in the below is our main frontend code we run the frontend in our terminal with this code. So i will be explaining it a little deeply.

In the first place we import all the pages like Log In page , Notifications page etc. Then for router process we import router compenents. And useState for store the notification count and useEffect for controling when the notification count change. So it’s used controling the notification and when a new notification comes it’s play notification sound.

So we check the notification count in every 5 seconds and if it changes the sound play to notificate the user.

```
import LogIn from "./pages/LogIn";

import ForgotPassword from "./pages/ForgotPassword";

import ResetPassword from "./pages/ResetPassword";

import HomePage from "./pages/HomePage";

import { BrowserRouter as Router, Routes, Route, Navigate } from "react-router-dom";

import Profile from "./pages/Profile";

import UploadOutfit from "./pages/UploadOutfit";

import SavedOutfits from "./pages/SavedOutfits";

import OutfitSuggestions from "./pages/OutfitSuggestions";

import LandingPage from "./pages/LandingPage";

import Favorites from "./pages/Favorites";

import Notifications from "./pages/Notifications";

import { useEffect, useState } from "react";




function NotificationListener() {
  const [previousCount, setPreviousCount] = useState(0);

  useEffect(() => {

    const token = localStorage.getItem("token");

    if (!token) return;

    const checkNewNotifications = async () =>

      { try {

        const res = await fetch("http://localhost:5000/api/notifications/unread-count",

          { headers: {

            Authorization: `Bearer ${token}`,

          },

        });

        const { count } = await res.json();

        if (count > previousCount) {

          const audio = new Audio("/notification.mp3");

          audio.play().catch(err => {

            console.error("Voice error:", err);

          });

        }

        setPreviousCount(count);

      } catch (err) {

        console.error("Error checking notifications:", err);

      }

    };

    checkNewNotifications();

    const interval = setInterval(checkNewNotifications, 5000);

    return () => clearInterval(interval);

  }, [previousCount]);

  return null;

}
```

It's for if a user don't have token it's directly send to the landing page. So users can't access the pages with PrivateRoute if they don't log in the website.

```
const PrivateRoute = ({ element }) => {

  const token = localStorage.getItem("token");

  return token ? element : <Navigate to="/" replace />;

};
```

So in this code runs the corresponding page based on which path user goes to. For example if user goes to the /profile the Profile page automatically opens.

```
function App()
  { return (
    <Router>
      <NotificationListener />
      <Routes>
        <Route path="/" element={<LandingPage />}/>
        <Route path="/login" element={<LogIn />} />
        <Route path="/forgot-password" element={<ForgotPassword />} />
        <Route path="/reset-password"element={<ResetPassword />} />
        <Route path="/home" element={<PrivateRoute element={<HomePage />} />} />
        <Route path="/outfit-suggestions" element={<PrivateRoute element={<OutfitSuggestions />} />} />
        <Route path="/profile" element={<PrivateRoute element={<Profile />} />} />
        <Route path="/upload" element={<PrivateRoute element={<UploadOutfit />} />} />
        <Route path="/saved-outfits" element={<PrivateRoute element={<SavedOutfits />} />} />
        <Route path="/favorites" element={<PrivateRoute element={<Favorites />} />} />
        <Route path="/notifications" element={<PrivateRoute element={<Notifications />} />} />
      </Routes>
    </Router>
  );
}
export default App;
```

4. AI Integration Layer

This is the layer that manages artificial intelligences features. In this layer we take the user's request and it's data , then send it to the necessary AI model. It process the data and then we take the output and give the user. We use 3 different artificial intelligence and one embedding search engine which name is FAISS, in this layer so i will explain them one by one.

A) NSFW

This artificial intelligience detects inappropriate images when users try to upload outfits. If it’s a inappropriate image it’s block the uploading process so they can’t upload this image.

The code is here;

First of all we download nsfw model from hugging face.

```
MODEL_ID = "Falconsai/nsfw_image_detection"

PROCESSOR = AutoProcessor.from_pretrained(MODEL_ID)

MODEL  = AutoModelForImageClassification.from_pretrained(MODEL_ID)
```

Then we take the users uploaded image and convert it to rgb format because this model require rgb formated images.

Then process the image and give back as PyTorch tensor. Then model give the solution and with the softmax function we turn output into probabilities.

```
def get_nsfw_score(image_path):

    image = Image.open(image_path).convert("RGB")

    inputs = PROCESSOR(images=image, return_tensors="pt")

    with torch.no_grad():

        outputs = MODEL(**inputs)

        logits = outputs.logits

        probs = logits.softmax(dim=1)


    labels = MODEL.config.id2label
```

The it calculates the NSFW scrore of an image by summing up the probabilities corresponding to the nsfw tag.And give the sum as nsfw_score.

```
    nsfw_labels = ["nsfw"]

    nsfw_score = sum([probs[0][i].item() for i, label in labels.items() if label.lower() in nsfw_labels])

    return nsfw_score

def is_valid_outfit(image_path):

    nsfw_score = get_nsfw_score(image_path)


    print("NSFW SCORE:", nsfw_score)
```

If nsfw score is bigger than 0.95 then it’s return as false so users couldn’t upload the image.

```
    if nsfw_score > 0.95:

        return False
```

B) MediaPipe Pose

It's an artificial that developed by Google. It detects the human body's 33 body parts. In my project we use this ai for making sure the uploaded image is a human and includes an outfit. I decided the outfit image should be at least include human body from shoulders to knee's.

So here is the code;

So we send the image to MediaPipe Pose with turning it into rgb format.

```
def get_visible_body_points(image_path):

    mp_pose = mp.solutions.pose

    image = cv2.imread(image_path)

    if image is None:

        return 0


    with mp_pose.Pose(static_image_mode=True) as pose:

        results = pose.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

        if not results.pose_landmarks:

            return 0
```

This is the visibility function. We make it as 0.5 when the number gets bigger the landmarks detected more harder like maybe if you wear a long plenty skirt with 0.5 it can detect the knee but with 0.8 it's very hard.

```
        def is_visible(part):

            return results.pose_landmarks.landmark[part].visibility > 0.5
```

So making sure that user not only upload an image with her upper or lower body part cause it will not be an outfit image.

```
        upper = is_visible(mp_pose.PoseLandmark.LEFT_SHOULDER) or is_visible(mp_pose.PoseLandmark.RIGHT_SHOULDER)

        lower = is_visible(mp_pose.PoseLandmark.LEFT_KNEE) or is_visible(mp_pose.PoseLandmark.RIGHT_KNEE)

        if not (upper and lower):

            print("Only upper or lower body detected !!!")

            return 0
```

So as we say we took the body parts from shoulders to knee.

```
        required_points =

            [ mp_pose.PoseLandmark.LEFT_SHOULDER,

            mp_pose.PoseLandmark.RIGHT_SHOULDER,

            mp_pose.PoseLandmark.LEFT_HIP,

            mp_pose.PoseLandmark.RIGHT_HIP,

            mp_pose.PoseLandmark.LEFT_KNEE,

            mp_pose.PoseLandmark.RIGHT_KNEE,

        ]
```

We control all the landmark's visibility and if it doesen't seen we didn't count that landmark. We sum the seen landmarks.

```
        count = sum(

            1 for point in required_points

            if results.pose_landmarks.landmark[point].visibility > 0.5

        )

        return count
```

If the seen lanmark count is bigger or equal to 4 we send the image as true. So if the image passed from nsfw and this user can upload the image.

```
def is_valid_outfit(image_path):

    visible_count = get_visible_body_points(image_path)

    print("VISIBLE BODY POINTS:", visible_count)

    return visible_count >= 4
```

C) CLIP

It is an artificial intelligence that developed by OpenAI. It's extract the images embeddings. So later we can use the embeddings found similar products or outfits that build with that product.

We take the image_url and id datas from products table.

```
db = pymysql.connect(
    host="localhost",
    user="root",
    password="123456",
    database="fashion_assistant"
)

cursor = db.cursor()
cursor.execute("SELECT id, image_url FROM products")
rows = cursor.fetchall()
```

We upload CLIP ai model and it's processor.

```
model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
```

```
embeddings_data = []
```

```
headers = {
    "User-Agent": "Mozilla/5.0"
}
```

If the image's format is appropriate it download the image then turn it into rgb format.

```
for row in rows:
    product_id, image_url = row
    try:
        response = requests.get(image_url, headers=headers, timeout=20, allow_redirects=True)
        content_type = response.headers.get("Content-Type", "")
        if not any(t in content_type for t in ["image/jpeg", "image/png", "image/webp"]):
            raise Exception(f"Invalid content type: {content_type}")
        image = Image.open(BytesIO(response.content)).convert("RGB")
```

Make the image like PyTorch tensor then extract the embedding.

```
        inputs = processor(images=image, return_tensors="pt")
        outputs = model.get_image_features(**inputs)
        embedding = outputs[0].detach().numpy().tolist()
```

It’s add the embedding to the product’s id too.

```
        embeddings_data.append({ "i
            d": product_id,
            "embedding": embedding
        })
```

```
        print(f"{product_id} processed successfully.")
    except Exception as e:
        print(f"Error ({product_id}): {e}")
```

Then it write all the product embedding into embeddings.json so we can use them later.

```
with open("embeddings.json", "w") as f:
    json.dump(embeddings_data, f)
print(f"\n embeddings.json file created. Total: {len(embeddings_data)} products processed.")
```

D) FAISS

It’s an embedding search engine that devoloped by Facebook. It takes big embedding lists and when a new embedding come it gives the most similar embedding to the new embedding. In our system we extract the outfit or product data’s embeddings with the CLIP model. Then when users use the visual similarity filter we take the users’s image and extract it’s embeddings too. Then FAISS look it’s embedding storage and give us the most similar ones.

For this purpose it calculate the ecludian distance of the one image embeddings with the others. If the ecludian distance is more little it’s more similar.

So here is the code example;

We have a faiss_index folder. In that folder we upload the embeddings.json we extract before. In index.bin file we have the product embeddings and in the id_list file we have the id of that product in same order with index.bin. So here we upload this 2 file for using them in the code.

```
index = faiss.read_index("faiss_index/index.bin")
with open("faiss_index/id_list.npy", "rb") as f:
    id_list = np.load(f)
print("CLIP and FAISS uploaded.")
```

So in here we extract the embedding of user's upload image. Then with reshape we make it as FAISS wanted way. So then we calculate the distances for getting the most similar 25 product with FAISS. And getting similar product embeddings order numbers in the index.

```
try:

        query_vector = get_image_embedding(image_path).reshape(1, -1)

        distances, indices = index.search(query_vector, k=25)

        print("FAISS distances:", distances)

        print("FAISS indices:", indices)

    except Exception as e:

        raise HTTPException(status_code=500, detail="FAISS search error!")
```

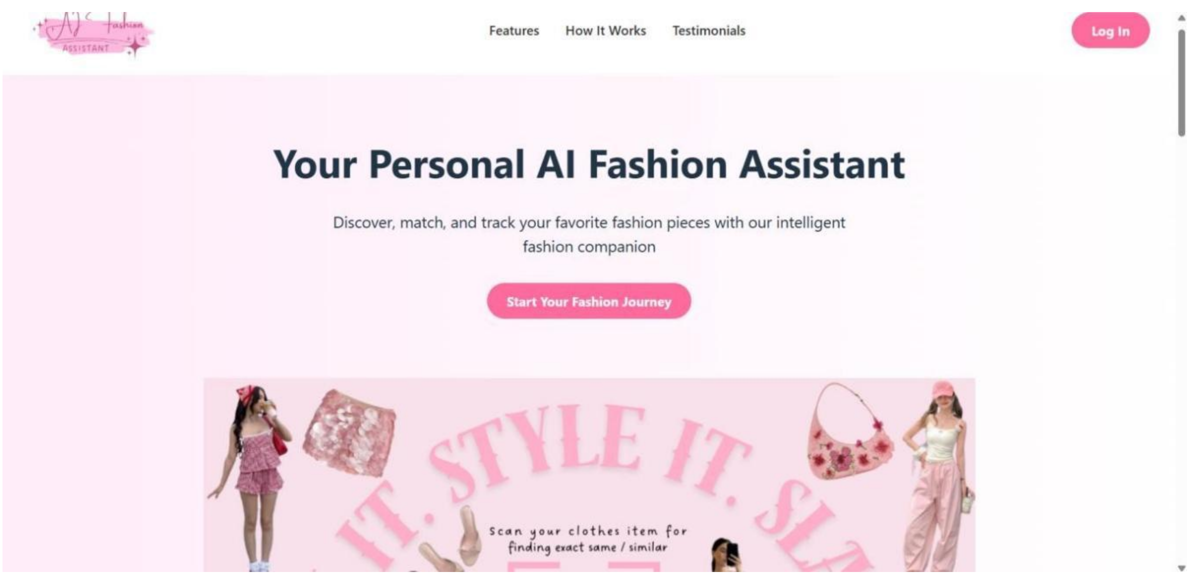
Then with matching the indices and id_list we take the product id too so we can return the products to the user.

```
    similar_ids = [int(id_list[i]) for i in indices[0]]

    print(f"Similar product IDs: {similar_ids}")
```

Website's Pages

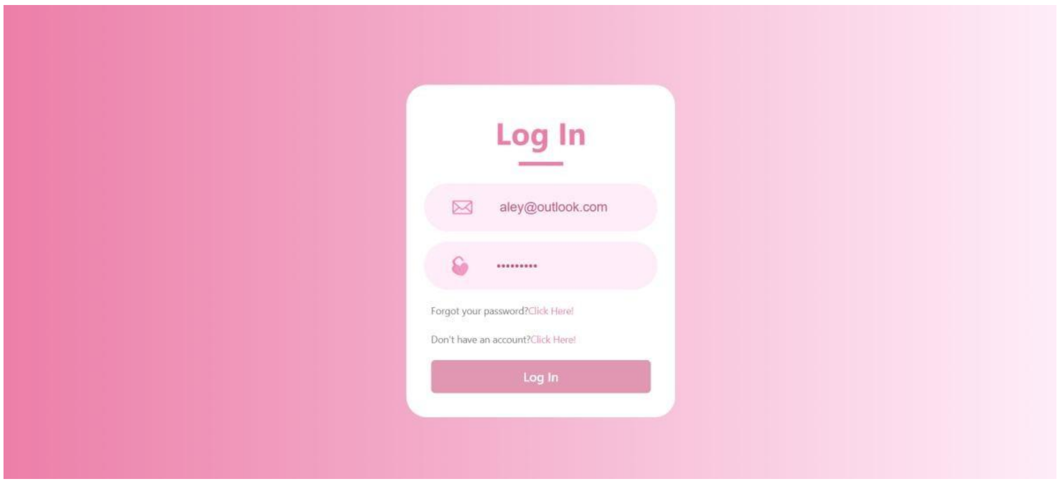
1. Landing Page:

In this page users just have knowledge about my website with clicking log in or start your fashion button they can sign up or log in to the website. In features it's describe our features shortly than in how it works place we tell more information to for how users can use this features. It's basically maded with css and react.


AI Fashion ASSISTANT
Features
How It Works
Testimonials
Log In
Your Personal AI Fashion Assistant
Discover, match, and track your favorite fashion pieces with our intelligent fashion companion
Start Your Fashion Journey
STYLE IT. SLAY IT.
Scan your clothes item for finding exact same / similar

2. Log In / Sign Up / Forgot Password Pages:

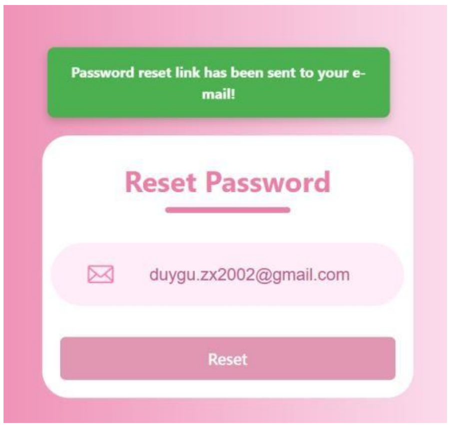
So it's the must have pages. When user click the log in button if user have signed before she can access to our actual website pages. If she don't have an account she can click Don't have an account and create a account with writing username, e-mail and password. The username and of course e-mail should be different from the registered users.


Log In
aley@outlook.com
•••••••••
Forgot your password?Click Here!
Don't have an account?Click Here!
Log In

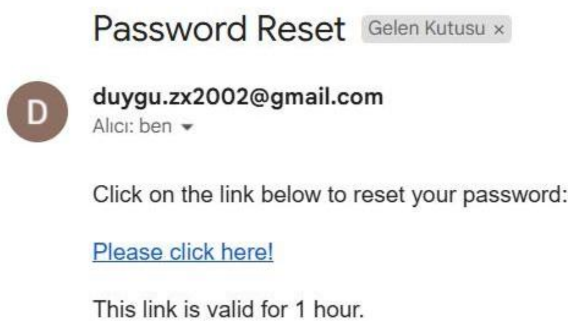
Here is the login route of my log in page

```
router.post("/login", async (req, res) =>

    { const { email, password } = req.body;


    try {

        const [results] = await db.query("SELECT * FROM users WHERE email = ?", [email]);

        if (results.length === 0) {

            return res.status(401).json({ error: "User not found!" });

        }

        const user = results[0];

        if (!bcrypt.compareSync(password, user.password)) {

            return res.status(401).json({ error: "Wrong password!" });

        }

        const token = jwt.sign(

            { id: user.id, email: user.email, username: user.username },

            process.env.JWT_SECRET,

            { expiresIn: "7d" }

        );

        res.json({ message: "Login successful!", token });

    } catch (err)

        { console.error(err);

        res.status(500).json({ error: "Network error!" });
    }
});
```

And if user forgots her password she can click forgot your password. Then the first page will be showed to her. She can write her e-mail but she must be registered before of course than the second image's typed e-mail will be come to her e-mail address. When she clicks the "please click here" she will be to direct the 3rd image page in here she can reset her password.



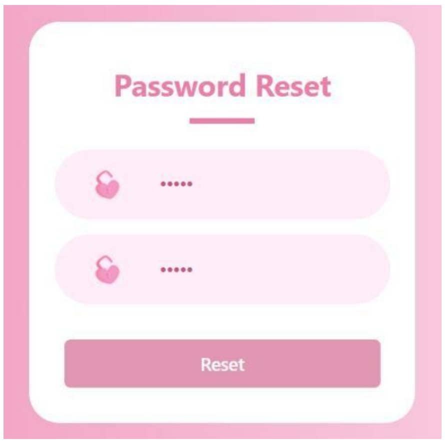
------>


Password Reset
Gelen Kutusu ×
duygu.zx2002@gmail.com
Alıcı: ben

Click on the link below to reset your password:

Please click here!

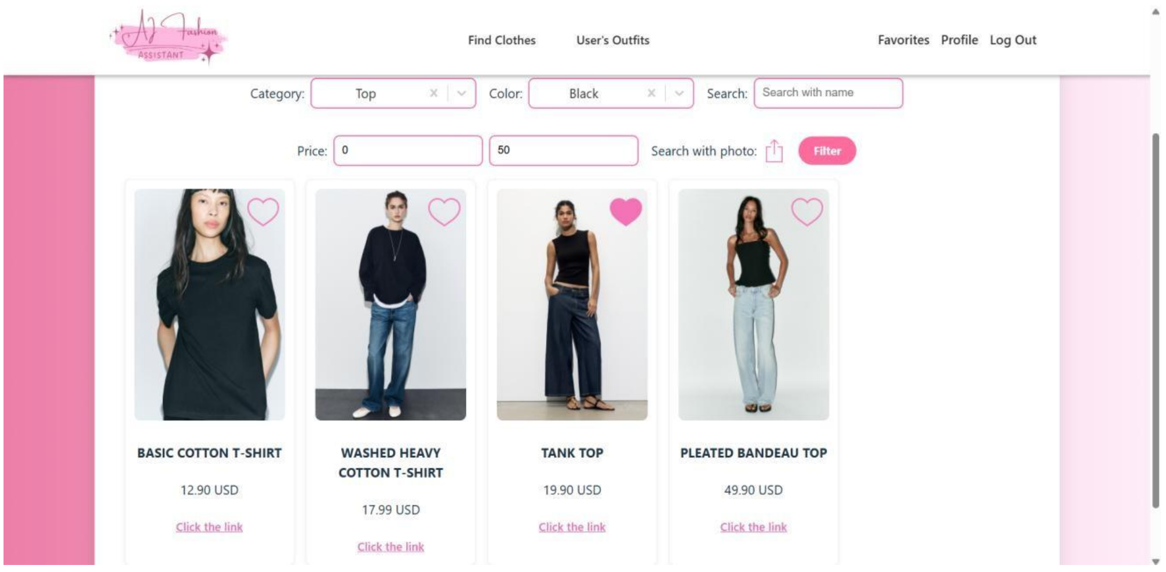
This link is valid for 1 hour.



Here is the forgot password page's route. When a user send to request of changing password the e-mail message is automatically goes.

```
router.post("/forgot-password", async (req, res) =>
    { const { email } = req.body;
    const emailRegex = /^[a-zA-Z0-9._%+-ğüşıöçĞÜŞİÖÇ]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

    if (!emailRegex.test(email)) {
        return res.status(400).json({ error: "Invalid e-mail format!" });
    }
    try {
        const [results] = await db.query("SELECT * FROM users WHERE email = ?", [email]);
        if (results.length === 0) {

            return res.status(404).json({ error: "This e-mail isn't registered!" });

        }

        const resetToken = crypto.randomBytes(32).toString("hex");

        const expireTime = Math.floor((Date.now() + 3600000) / 1000);

        await db.query("UPDATE users SET reset_token = ?, reset_expires = ? WHERE email = ?", [resetToken, expireTime,
email]);

        const transporter =

            nodemailer.createTransport({ service: "gmail",

            auth: {

                user: process.env.EMAIL_USER,

                pass: process.env.EMAIL_PASS

            }

        });

        const mailOptions = {

            from: process.env.EMAIL_USER,

            to: email,

            subject: "Password Reset",

            html: `

                <p>Click on the link below to reset your password:</p>

                <a href="http://localhost:5173/reset-password?token=${resetToken}">Please click here!</a>

                <p>This link is valid for 1 hour.</p>

            `

        };

        await transporter.sendMail(mailOptions);

        res.json({ message: "Password reset link has been sent to your e-mail!" });

    } catch (err)

        { console.error(err);

        res.status(500).json({ error: "Server error!" });

    }

});
```

3. Find Clothes Page:



In this page user can see the products with their name,price and brand website url. When they click to the url they will be directed to the website where the clothing is sold. When they click the heart icons they will be adding the product into the favorites. They can use different filters to see the products they want. The filters are category, color, product name search, price range, and photo similarity filter. The filters can use with together too. In that page for seeing products you must select a category first.

Here is the route for listing the products with the filters that users want to apply. Except photo similarity filter.

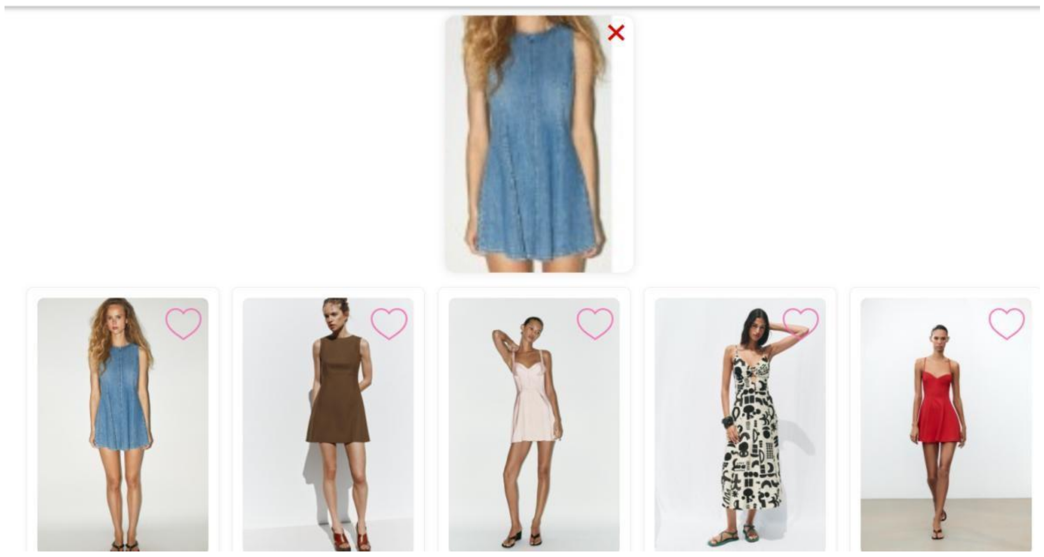
```
router.get("/products", verifyToken, async (req, res) =>
  { const userId = req.user.id;
  const { category, color, min, max, name } = req.query;


  try {
    const [rows] = await pool.query(
      `SELECT
        p.*,
        pt.min_price,
        pt.max_price,
        pt.max_price AS target_price,
        IF(pt.product_id IS NOT NULL, 1, 0) AS is_discount_tracked,
        IF(st.product_id IS NOT NULL, 1, 0) AS is_stock_tracked
      FROM products p
      LEFT JOIN price_tracking pt ON p.id = pt.product_id AND pt.user_id = ?
      LEFT JOIN stock_tracking st ON p.id = st.product_id AND st.user_id = ?
      WHERE 1=1
        AND (? IS NULL OR p.category = ?)
```

```
         AND (? IS NULL OR p.color = ?)

         AND (? IS NULL OR p.sale_price >= ?)

         AND (? IS NULL OR p.sale_price <= ?)

         AND (? IS NULL OR LOWER(p.name) LIKE LOWER(CONCAT('%', ?, '%')))

      `,

      [

        userId, userId,

        category || null, category || null,

        color || null, color || null,

        min || null, min || null,

        max || null, max || null,

        name || null, name || null

      ]

    );

    res.json(rows);

  } catch (err)

    { console.error(err);

    res.status(500).json({ error: "Server error" });

  }

});
```

It’s the code piece of getting the users wanted filtered products. This one is for when user use photo similarity filter with other filters. There is two different code like above and below code because when we are using photo similarity filter we are using CLIP ai with FAISS. So they give us 25 different products. Then we can apply that products the other filter requests not before so we need to make 2 different code for it.

```
category = data.category

    color = data.color

    search_term = data.searchTerm

    min_price = data.minPrice

    max_price = data.maxPrice


    try:

        min_price = float(min_price) if min_price is not None else None

    except (TypeError, ValueError):

        min_price = None

    try:

        max_price = float(max_price) if max_price is not None else None

    except (TypeError, ValueError):
        max_price = None

    try:

        format_ids = ",".join(["%s"] * len(similar_ids))

        query = f"SELECT * FROM products WHERE id IN ({format_ids})"

        params = similar_ids.copy()

        if category:

            query += " AND category = %s"

            params.append(category)

        if color:

            query += " AND color = %s"

            params.append(color)

        if min_price is not None:

            query += " AND sale_price >= %s"

            params.append(min_price)

        if max_price is not None:

            query += " AND sale_price <= %s"

            params.append(max_price)

        if search_term:

            query += " AND name LIKE %s"

            params.append(f"%{search_term}%")

        cursor = db.cursor(pymysql.cursors.DictCursor)

        cursor.execute(query, params)

        products = cursor.fetchall()

        cursor.close()

    except Exception as e:

        print("Database error:", e)

        raise HTTPException(status_code=500, detail="Database query error!")
```

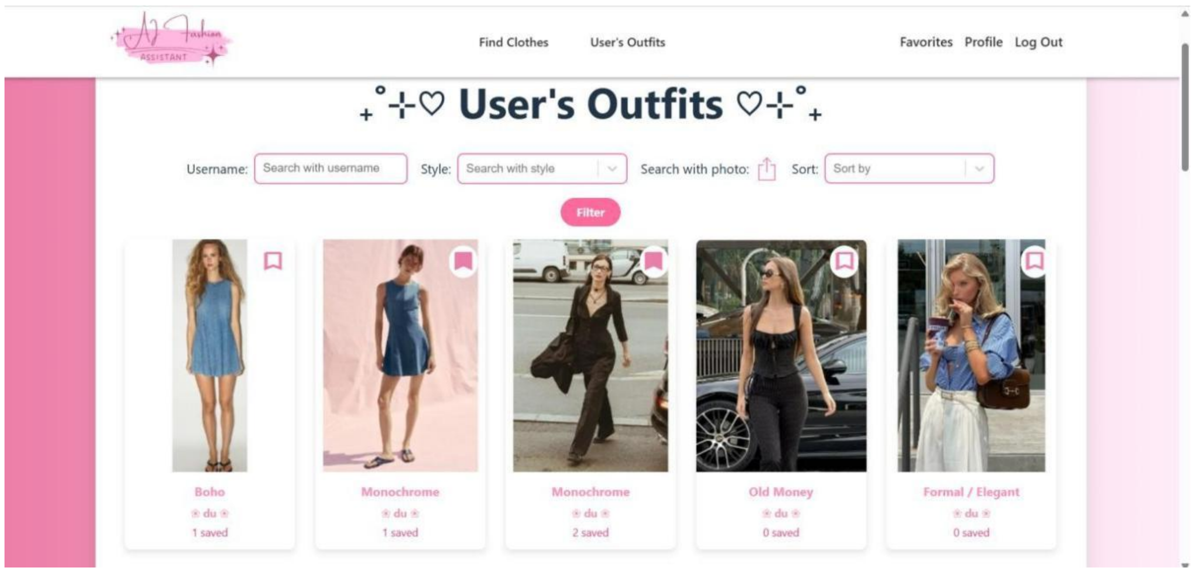
Again the Clothes Finder page image. In here we use photo similarity filter with selecting dress category.



4. User’s Outfit Page:

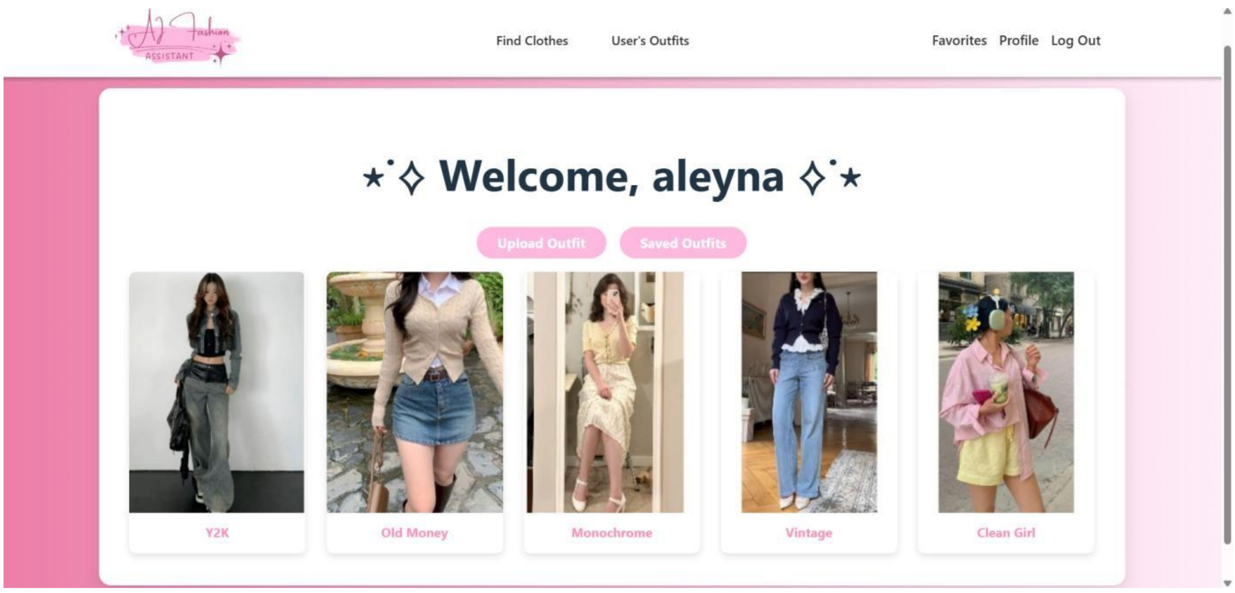
In this page user can see the other user’s outfit sahrings with their username, style category and the count of how many user saved it. When they click the bookmark icons they will be adding the outfit into the saved outfits page. They can use different filters to see the outfits they want. The filters are style category, username, sort and photo similarity filter. The filters can use with together too.

In that page if you want to use different filters at the same time you should write 2 different codes too like the user’s outfits page. Because when we are using photo similarity filter we are using CLIP ai with FAISS. So they give us 15 different products. Then we can apply that products the other filter requests not before so we need to make 2 different code for it.

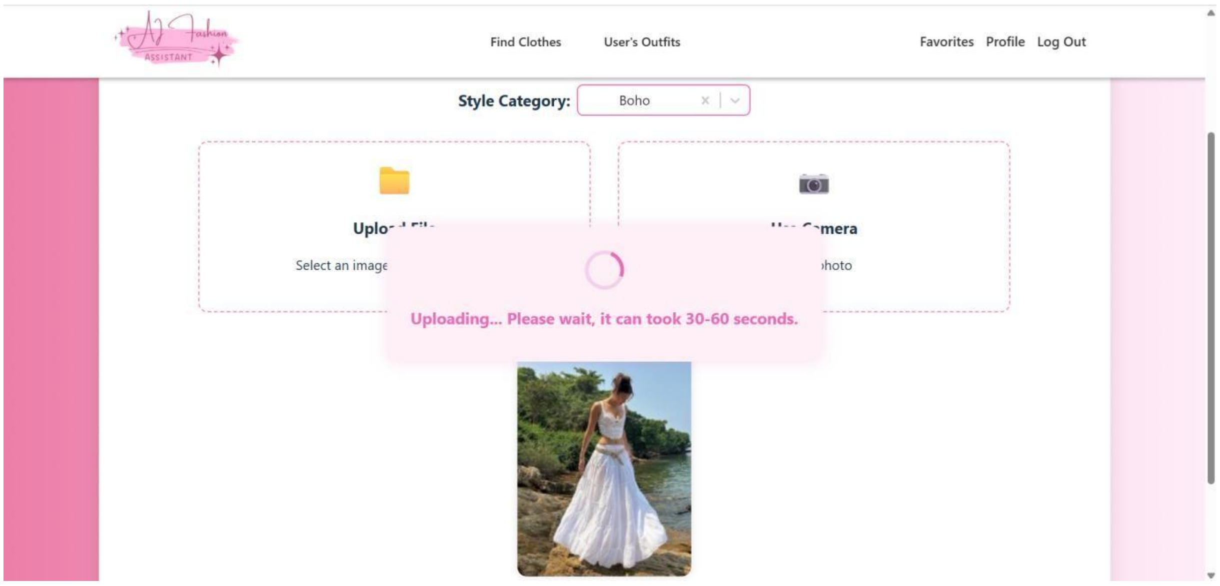


5. Profile Page:

In this page users can see their uploaded outfits with the category names on it. You can click upload outfit or saved outfits buttons to go these pages.



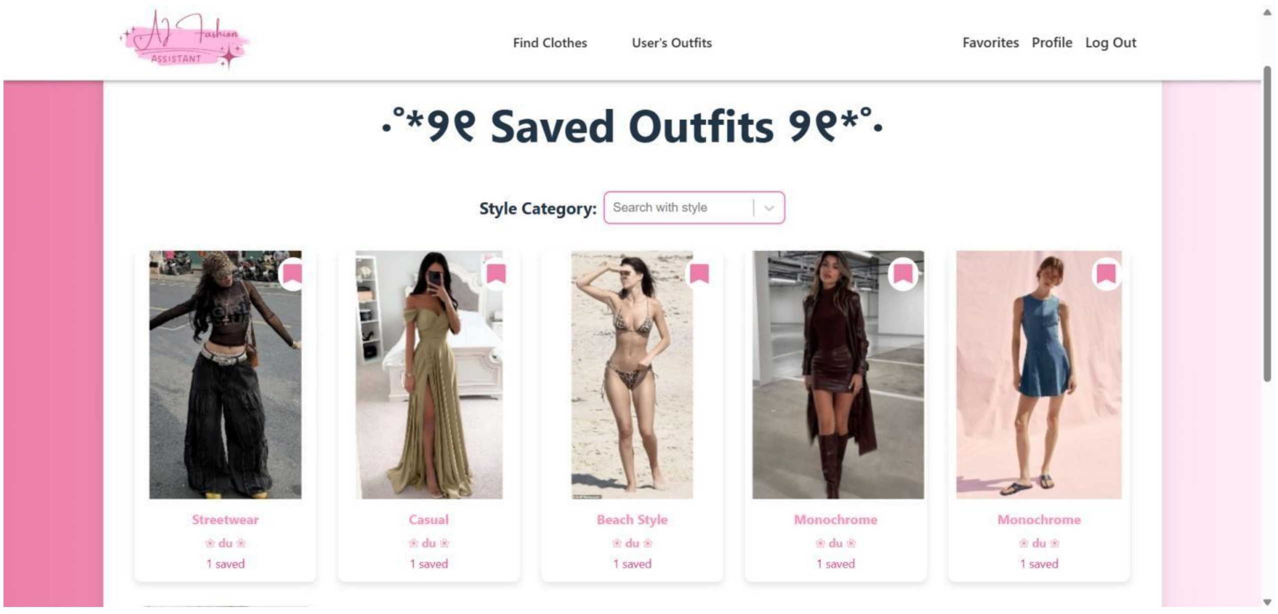
6. Upload Outfit Page:



In this page user can upload outfit with selecting style category and uploading an image or taking picture. If the image pass the NSFW and MediaPipe Pose ai's things it can be uploaded. So the uploading part is taking a long time because NSFW downloaded from another website but for that in every 8 second it shows a different write to keep the hype of the user. Here is the code piece of it;

```
const uploadMessages = [

  "Uploading... Please wait, it can took 30-60 seconds.",

  "Sorry for the delay... Your outfit just got a standing ovation from the code.",

  "It takes long cause you're a real diva girlll !",
  "Calling a real model agency cause this look can't be wasted!",

  "Sorry for waiting girl... the computer crashed because of your beauty!",

  "I'm writing your name for this year's Met Gala. ",

  "Girl... the internet's shaking. Your style just broke the algorithm.",

  "Too much slay detected. Initiating fashion emergency protocols.",

  "Hold on queen, even the servers are gossiping about your iconic look.",

  "Your outfit just made the AI question its own fashion sense ...",

  "This look deserves its own documentary. Stay tuned babe! ",

  "Stitching your look into digital fashion history... please wait. ",

  "Uploading this outfit to the Hall of Slay. Sparkles included.",

  "Warning: Viewers may faint due to extreme glam. "

];


export default function UploadingLoader({ isUploading }) {

  const [messageIndex, setMessageIndex] = useState(0);


  useEffect(() => {

    if (!isUploading) return;

    let firstTime = true;

    const interval = setInterval(() => {

      if (firstTime) {

        setMessageIndex(0);

        firstTime = false;

      } else {

        const randomIndex = Math.floor(Math.random() * (uploadMessages.length - 1)) + 1;

        setMessageIndex(randomIndex);

      }

    }, 8000);

    return () => clearInterval(interval);

  }, [isUploading]);

  if (!isUploading) return null;

  return (

    <div className="upload-loading">

      <div className="spinner"></div>

      <p>{uploadMessages[messageIndex]}</p>

    </div>

  );

}
```

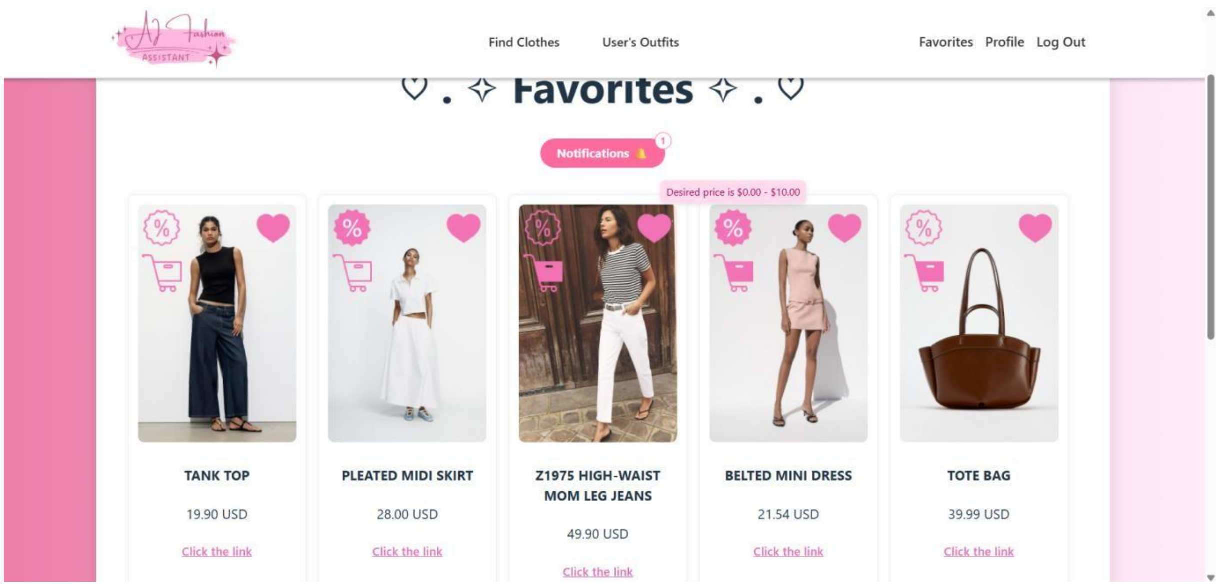
7. Saved Outfits Page:



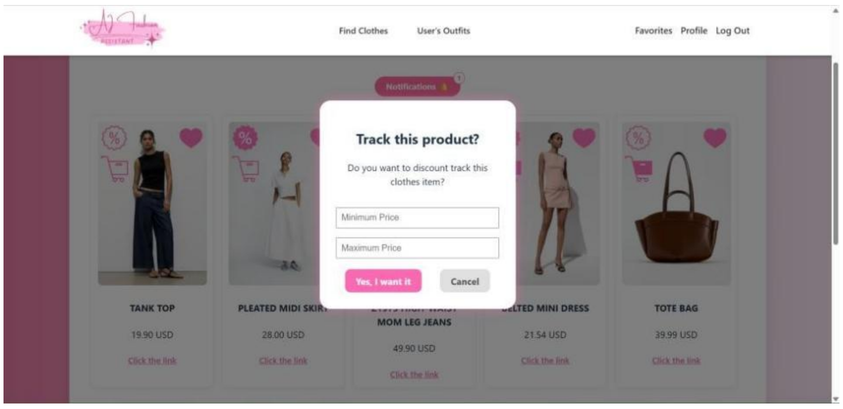
In here users can see their saved outfits of others or their outfits. Here is the backend code for getting saved outfits and their informations;

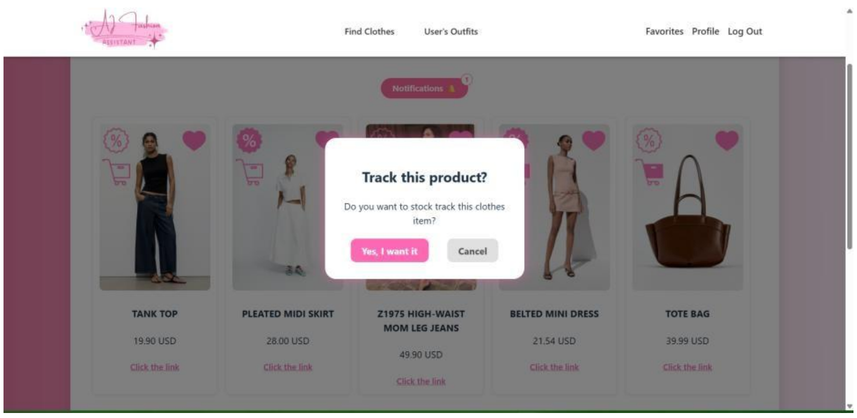
```
router.get("/saved-outfits", verifyToken, async (req, res) =>

  { try {

    const userId = req.user.id;


    const [savedOutfits] = await pool.query(

      `SELECT

         uo.id AS outfit_id,

         uo.image_url,

         uo.created_at,

         os.style_name,

         u.username,

         (

           SELECT COUNT(*)

           FROM user_saved_outfits

           WHERE outfit_id = uo.id AND is_saved = 1

         ) AS saved_count

       FROM user_saved_outfits uso

       JOIN user_outfits uo ON uso.outfit_id = uo.id

       LEFT JOIN outfit_styles os ON uo.id = os.outfit_id

       LEFT JOIN users u ON uo.user_id = u.id
      WHERE uso.user_id = ? AND uso.is_saved = 1
      ORDER BY uso.created_at DESC`,
     [userId]
    );
    res.status(200).json(savedOutfits);
  } catch (error) {
    console.error("Error fetching saved outfits:", error);
    res.status(500).json({ error: "Internal server error." });
  }
});
```

8. Favorites Page:



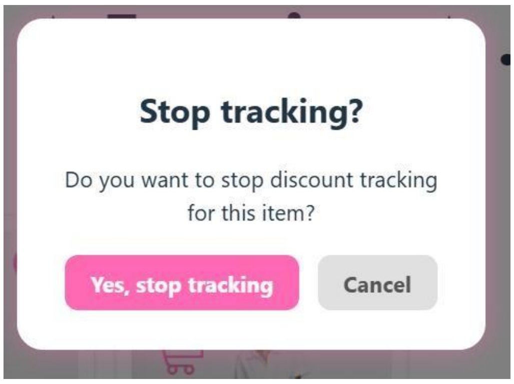
In this page user can see their favorited products. They can start stock or discount trackings with clicking the buttons. Also when they comes to a clicked discount icon they can see what is the range price they choose for this tracking. When you want to stop a tracking you should fill or just click the yes,stop tracking button in the modals. If they want to stop the tracking they can click the clicked icons again. You can see the modals for starting trackings ans stopping tracking in the below;





Track this product?

Do you want to stock track this clothes item?



Stop tracking?

Do you want to stop discount tracking for this item?

Here is the backend code of trackings ;

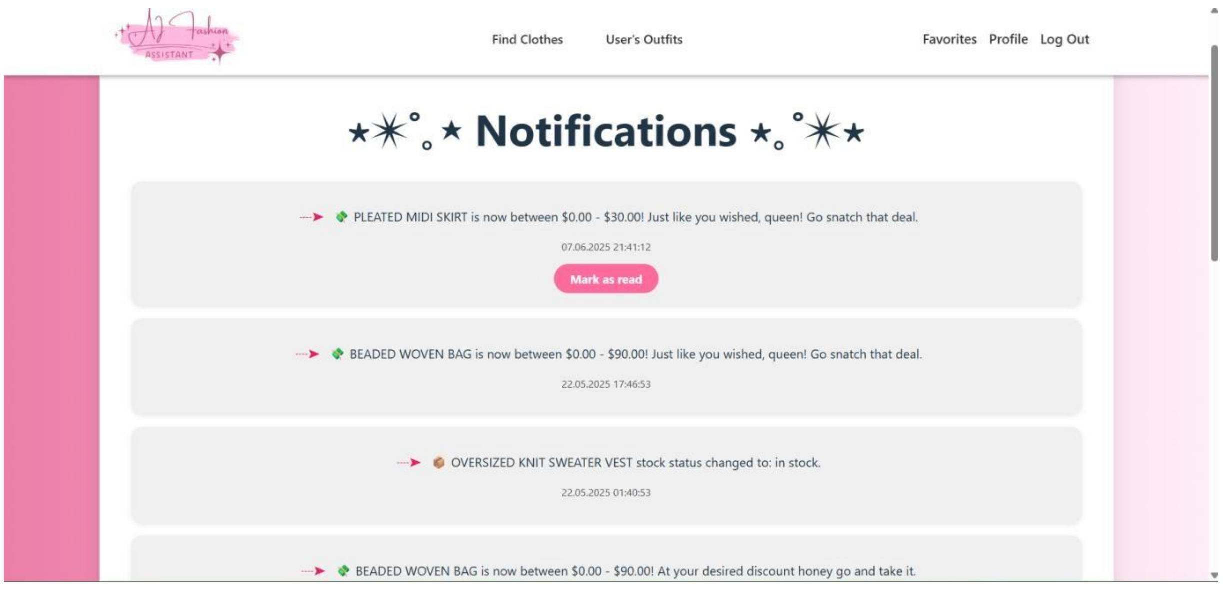
```
router.post("/price-tracking", verifyToken, async (req, res) =>
 { const userId = req.user.id;
 const { product_id, min_price, max_price } = req.body;


 try {
   await pool.query(
     "INSERT INTO price_tracking (user_id, product_id, min_price, max_price) VALUES (?, ?, ?, ?)",
     [userId, product_id, min_price, max_price]
   );
   res.json({ success: true });
 } catch (err) {
   res.status(500).json({ error: "Server error" });
 }
});
```

```
router.post("/stock-tracking", verifyToken, async (req, res) =>
 { const userId = req.user.id;
 const { product_id, availability } = req.body;
```

```
  const validStocks = ["in_stock", "out_of_stock", "low_on_stock"];

  if (!validStocks.includes(availability)) {

    return res.status(400).json({ error: "Invalid or missing availability value." });

  }
```

```
  try {

    await pool.query(

      "INSERT INTO stock_tracking (user_id, product_id, last_known_stock) VALUES (?, ?, ?)",

      [userId, product_id, availability]

    );

    res.json({ success: true });

  } catch (err) {

    res.status(500).json({ error: "Server error" });

  }

});
```

9. Notifications Page



★✳˚｡★ Notifications ★｡˚✳★

PLEATED MIDI SKIRT is now between $0.00 - $30.00! Just like you wished, queen! Go snatch that deal.
07.06.2025 21:41:12

BEADED WOVEN BAG is now between $0.00 - $90.00! Just like you wished, queen! Go snatch that deal.
22.05.2025 17:46:53

OVERSIZED KNIT SWEATER VEST stock status changed to: in stock.
22.05.2025 01:40:53

BEADED WOVEN BAG is now between $0.00 - $90.00! At your desired discount honey go and take it.

In this page when the tracking done users will hear a voice. This means their tracking for one product is happened. Then they can see the notification and they should be mark as read or the count of unreaded notifications will be showing in the Favorites page's notification button's above.

Here is the all the routes of notification page:

```
router.get("/notifications", verifyToken, async (req, res) => {

  const userId = req.user.id;
  const page = parseInt(req.query.page) || 1;

  const limit = 10;

  const offset = (page - 1) * limit;


  const [rows] = await pool.query(

    `SELECT * FROM notifications

     WHERE user_id = ?

     ORDER BY created_at DESC

     LIMIT ? OFFSET ?`,

    [userId, limit, offset]

  );

  const [[{ count }]] = await pool.query(

    `SELECT COUNT(*) AS count FROM notifications WHERE user_id = ?`,

    [userId]

  );

  res.json({ data: rows, total: count, page, totalPages: Math.ceil(count / limit) });

});

router.post("/notifications/:id/read", verifyToken, async (req, res) =>

  { const { id } = req.params;

  await pool.query(

    "UPDATE notifications SET is_read = TRUE WHERE id = ?",

    [id]

  );

  res.json({ success: true });

});

router.get("/notifications/unread-count", verifyToken, async (req, res) =>

  { const userId = req.user.id;

  const [rows] = await pool.query(

    `SELECT COUNT(*) as count FROM notifications WHERE user_id = ? AND is_read = FALSE`,

    [userId]

  );

  res.json({ count: rows[0].count });

});
```

6. Conclusion

6.1 Benefits

a. Benefits to users :

1. When users like a product they can use discount tracking feature and write their desired price when their desired price come they will be notified so they will buy their favorite piece.

2. When a product is out of the stock users can use stock tracking feature and when the product is in the stock they will be get it before its to late.

3. Users can see products from different brands at the same time.

4. When they like something they can add it too their favorites, so instead of looking at different liked products from different brands' applications, they can see all their favorites in one place.

5. Users can upload their outfit so they can show their style. Also can look at the other people's outfits so they can get inspire.

6. When a user love an outfit they can save it so whenever they want they can see it.

7. With filters, users can quickly find the products or outfits they are looking for.

8. If a user see a clothes item somewhere else, they can use the product similarity filter and find the exact same or similiars with different types, prices or colors etc.

9. If a user want to style an clothes item, they can use outfit similarity filter and find outfits with that clothes item or something similar.

10. When a user try to use similar filter or uploading an outfit image, they can use our crop feature to remove unnecessary things. So they don't need to crop it before in their gallery or another app.

b. Benefits to me :

1. I gain experience to developing process of a big project.

2. I gain experience of full-stack web developing process.

3. I gain experience about designing process. Like selecting icons, making css.

4. I learn and experience different API's.

5. I experience that before starting to develop anything we should deeply search about it even it looks like it's easy to do it.

6. I improve my coding skills that the languages i use in this project.

7. My backend programming wasn't good i learn how to use node.js

8. I learn effectively using database queries to getting the data i want.

9. I learn to handle user authentication and taking user information using JWT.

10. I learn the process of a user's upload image like using camera or directly uploading. I learn how to handle this process like cropping feature, converting the image format as a one image format, for more efficient storage i learn about blob images etc.

11. I learn to use NSFW artificial intelligence for avoiding inappropriate image.

12. I learn about CLIP artificial intelligence. Like how it makes the embeddings of an image and how we use them.

13. I learn about embeddings like how i can extract them as json format from image datas.

14. I learn FAISS artificial intelligence. Like how it is finding the most similar embeddings when we search with an embedding of an image.

15. I learn hoe to integrate different things in the porject. L,ke frontend, backend, database, 4 different ai model.

16. This project make me more patient and investigative person which i think is the biggest benefit that i gained in this process.

6.2 Ethics

- When i collect the product information datas i didn't web scraping or etc. I just use API for taking the product informations. Because a lot of websites legally prohibited from obtaining its own data without their information. So when we want collect data we should use the APIs that the websites give us for datas (Monolith Law Office,2023).

- For avoiding bad malicious users upload inappropriate images i use NSFW artificial intelligience. AI generally used for preventing malicious users shares in the websites or applications (related Digital,2022).

- In my website we will be not using users datas to giving them manipulative suggestions for make them buying any spesific clothing company. All the product that we show to the user will be generated with users selected filters. And we will not take their datas for that purpose. Because it's a bad use of users data to managing their buying process and it will be unethical (Mehmet Etlioğlu,2024).

- My artificial intelligence that users can use for similarity filters is totally just for giving the users similar outfits they will not give manipulative results (Nazan Yeşilkaya,2022).

- In my website users datas will not sharing with third parties. Sharing or selling them to anyone without users knowledge is not ethical and it violates users privacy (Boray Soydan,2023) .

- My application will not give the outfits to the users so they will upload them and if they upload a copyrighted image they will be responsible from that. If they use it will be disrespect for the image's owner and it will be seem as unfair profit so it will be against to laws and unethical (Web Acil, 2024) .

- My project's first aim is when users want to buy something they can look at the outfits that maded by from that clothes item too so they can look at the pieces the combination pieces and they can decide that i have no piece for making this clothes item to combine. With this users will be prevent unnecessary shoppings so no more fast fashion because an clothes item is trendy now. (Pınar Çınar, 2025) So my aim is at least stop some users fast fashion shopping habits for helping the nature (Şerafettin & Müzeyyen, 2024) .

Why did I choose this project?

The reason why i choose this project is i want to see different brands clothes item at the same place. So i can use some filters and take the most advantaged clothes piece and also i can favorite them and see my favorites in the same place. Also when i see a clothes item at the social media or somewhere else i can take it's picture and with similarity filter i can see the exact same or similars of that clothes item. Also users can upload outfits. So i can get inspired by others outfits i can use filters to find my style outfits. And when i like a clothes item i can use the outfit similarity filter and see the outfits that build with that clothes item. So without spending time i can combine the clothes items.

I also suffered for stock or discount of the clothes. When a brands discounts start i didn't get notification so i can just see it from social media. And it will be too late. Or i like something i didn't know the stock situation of that product so it can run out the stock or a popular clothes item is always run out very quickly so i didn't catch up sometimes. But with the tracking system i can easily catch my favorited clothes items in the desired price and without missing the stock.

6.3 Future Works

Yes i will continue developing my application after the graduation.

1. First of all i want to add more brands products to my application. So users can have more choice.

2. I want to add more types of products. Like right i have only skirt, top, etc. But i want to add accessories, outerwear etc.

3. I want to add comment section for the products so users

4. I want to add a page that give users product recommendations based on their favorited products.

5. I want to make the user's outfits page more personalized. Like the outfits will be showing as based on which outfits that user saved before.

6. I want to add a feature so when users click an outfit image, we will be giving the products from our database which make up that outfit.

7. I want to make my filters more specific like right now for all the top clothing you just select the top category because i don't have enough data. If i have enough data like (+2000 clothes item) users can select top clothing and then they can select T-shirt, body, top, shirt etc..

8. I want a Tik Tok integration for when a user love a clothes item they can see which users in Tik Tok try it on themself so it will be more easy shopping for users.

9. I want to make fine tuning for all of my artificial intelligence models for making their work more better.

10. In the future i want to bring a reward system to users when their uploaded outfit share is getting more saved a certain number.

7. References

[1] Melek Coşgun (2012). Popular Culture and Consumer Society

[2] Pınar Çınar (2025). Social Media Influencers' Influence on Fashion Shopping Behavior

[3] Joydeep Bhattacharya . Visual Search Stats: Unlocking the Future of Search Technology Retrieved from https://seosandwitch.com/visual-search-stats/?

[4] Fatih B. & İpek O. (2022) . Internet - Based Problematic Shopping Behavior: Online Shopping Addiction

[5] (2025) . Diedrot Effect Retrieved From https://en.wikipedia.org/wiki/Diderot_effect?

[6] Cyberclick (2024).The Diedrot Effect: Boosting Sales Through Consumer Behavior Insights Retrieved From

https://www.cyberclick.net/numericalblogen/the-diderot-effect-boosting-sales-through-consumer-behavior-insights?

[7] Monolith Law Office (2023). What is Scraping? Explaining the Legal Challenges of this Convenient Data Collection Method That's Gaining Attention . Retrieved from https://monolith.law/tr/general-corporate/scraping-datacollection-law

[8] Related Digital (2022). How Should Content Management Be Done ? Retrieved from https://www.relateddigital.com/tr/blog/icerik-yonetimi-nasil-yapilmali/

[9] Mehmet Etlioğlu (2024). Ethics in Digital Marketing

[10] Nazan Yeşilkaya (2022). Ethical Issues Regarding Artificial Intelligence

[11] Boray Soydan (2023). New Media and Social Media: Ethical Issues and Solution Approaches

[12] Web Acil (2024). How to Pay Attention to Copyright Issues for Images on a Website Retrieved from https://webacil.com/web-sitesindeki-gorseller-icin-telif-hakki-konularina-nasil-dikkat-edilir/

[13] Şerafettin S. & Müzeyyen P. (2024). Environmental Impacts of Fast Fashion