



**CHANAKYA
UNIVERSITY**

Topic: AI-powered fashion trend forecasting

submitted by

Charan G – CU23MCA008A

Deeksha Y V – CU23MCA0011A

Siri Paladi – CU23MCA0045A

Lavanya K R – CU23MCA0023A

(Submitted as part of minor project course Summative Assessment-2024)

Under the guidance of

Dr. Setturu Bharath

Course: MCA

Minor Project

III Semester 2024 – 2025

School of Engineering

Certificate

This is to certify that the Minor project work entitled “AI-powered fashion trend forecasting” submitted to the School of Engineering, Chanakya University in partial fulfilment of the requirements of the degree of Batchelor of Computer Applications in the academic year 2024-2025 is a record of the original work done by Charan G, Deeksha Y V, Siri Paladi, Lavanya K R under my supervision and guidance and that this Minor project work has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or similar title to any candidate of any University.

Place:

Date:

Signature of Students:

- 1.Charan G
- 2.Deeksha Y V
- 3.Siri Paladi
- 4.Lavanya K R

Signature of the Guide

Contents

1. Abstract	4
2. Introduction	4-5
3. Literature Review	5-8
4. methods	9-12
5. Results and discussion	13-18
6. conclusion	18
7. Acknowledgement	19
8. code	19-51
9. Data source	52-53
10. Reference	53-55

AI-powered fashion trend forecasting

Abstract

Artificial Intelligence (AI) is transforming the fashion industry by driving innovation, enhancing consumer experiences, and promoting sustainability. This study explores AI's applications in consumer behaviour analysis, personalized recommendations, and sustainable practices, highlighting its role in reshaping design processes and addressing ethical challenges. AI technologies such as Generative Adversarial Networks (GANs), augmented reality (AR), and blockchain enable efficient supply chain management, creative ideation, and cultural preservation. However, challenges such as algorithmic bias, data privacy, and over-reliance on automation must be addressed to ensure ethical AI implementation. The findings underline the importance of balancing technological advancements with sustainability and human creativity to ensure the fashion industry's long-term growth and environmental responsibility.

Introduction

The fashion industry is undergoing a profound transformation driven by the integration of Artificial Intelligence (AI) technologies. From reshaping consumer interactions to streamlining design processes, AI has emerged as a powerful tool for enhancing efficiency, creativity, and sustainability. The global demand for personalized shopping experiences, ethical practices, and innovative designs has accelerated the adoption of AI, making it an essential component of modern fashion.

AI's applications in the fashion industry are diverse, ranging from analysing consumer behaviour and predicting trends to supporting circular fashion models. Tools like Generative Adversarial Networks (GANs) and augmented reality (AR) have not only redefined creative processes but also contributed to reducing waste and optimizing supply chains. However, with these advancements come challenges such as ethical considerations, algorithmic biases, and data privacy concerns, necessitating responsible implementation of AI.

This paper explores the multifaceted role of AI in the fashion industry, focusing on its impact on consumer behaviour, design innovation, and sustainability. It also addresses the challenges and opportunities associated with AI adoption, highlighting the importance of ethical frameworks and collaborative approaches to maximize its potential. By examining these aspects, the study provides insights into how AI can drive a more sustainable and inclusive future for the fashion industry.

Literature Review

AI in Consumer Behaviour and Personalization

Artificial Intelligence (AI) has become a transformative force in the fashion industry, particularly in influencing consumer behaviour and enhancing personalized experiences. Yeo et al. 2022 investigated how AI-powered tools, particularly on platforms like Instagram, shape consumer purchase decisions. Their study highlighted that factors such as electronic word-of-mouth (eWOM), emotional value, and perceived quality significantly influence consumer trust and engagement. AI enhances these factors by analysing browsing patterns and tailoring recommendations to meet individual preferences [1]. This personalized approach makes consumers feel more connected to brands, resulting in higher engagement and conversion rates.

Guo et al. 2023 expanded on AI's influence by focusing on how technologies such as Generative Adversarial Networks (GANs) and diffusion models enhance fashion detection, synthesis, and recommendation. These tools enable brands to predict consumer desires accurately, offering personalized recommendations based on a combination of past purchases, style preferences, and social media interactions. Virtual try-ons and augmented reality (AR) applications further enhance the consumer experience by providing lifelike previews of products, reducing uncertainty and improving satisfaction during online shopping [2].

Csanák 2020 emphasized AI's ability to process large datasets, including point-of-sale data, geographic information, and social media trends, to identify and predict fashion trends. These insights empower brands to make informed decisions, ensuring that inventory aligns with demand and reduces instances of overproduction. This predictive capability contributes to a more streamlined shopping experience for consumers while fostering sustainability [3].

The WearagAI Project 2023 showcased how AI impacts second-hand clothing markets, a growing segment of the sustainable fashion industry. AI algorithms assess environmental benefits such as water usage reduction and lower carbon emissions, making it easier for eco-conscious consumers to understand the value of purchasing pre-owned items. These tools also enable accurate pricing of garments, ensuring a balance between affordability and profitability in second-hand markets [4].

AI-Driven Design Innovation

AI has revolutionized the creative processes of fashion design, enabling designers to create innovative products while reducing the time and effort required for manual iterations. Guo et al. 2023 explored the role of Generative Adversarial Networks (GANs) and diffusion models in fashion design, highlighting their ability to generate realistic designs by synthesizing new patterns, styles, and textures based on existing data. These technologies have made it possible to visualize and test creative ideas rapidly, fostering a more dynamic and iterative design process [5].

Lee and Suh 2024 proposed integrating AI tools like ChatGPT and Midjourney into the fashion design workflow. These tools allow designers to input textual descriptions or sketches, which are then converted into high-quality visuals. This multimodal approach streamlines the collaboration between designers and stakeholders, enabling faster feedback loops and improved creative outcomes [6]. By reducing the need for repetitive manual tasks, designers can focus on exploring more innovative and experimental ideas [7].

The Cultural AI Archives 2023 highlighted AI's role in preserving cultural heritage by digitizing traditional motifs, patterns, and crafting techniques. This not only safeguards these valuable traditions for future generations but also ensures their integration into modern fashion. By leveraging AI, designers can reinterpret cultural elements in contemporary ways, promoting diversity and inclusivity in their creations [8].

AI in Sustainability and Ethical Practices

Sustainability has become a critical focus in the fashion industry, with AI playing a pivotal role in driving eco-friendly practices across the supply chain. Rathore 2016 discussed how AI supports circular fashion practices like recycling and upcycling, optimizing the use of resources while minimizing waste. These

advancements align with consumer expectations for ethical and environmentally responsible products [9].

WearagAIn 2023 further demonstrated the use of AI in second-hand clothing platforms, where algorithms predict demand, optimize inventory, and enhance the lifecycle of garments. These tools ensure that fewer products go to waste while supporting the growing consumer demand for sustainable options [10].

IoT and blockchain technologies are also instrumental in improving supply chain transparency and traceability. According to IoT and Blockchain Applications 2023, AI-driven systems enable real-time tracking of production processes, ensuring that every step, from sourcing materials to delivery, adheres to ethical standards [11]. These technologies enhance trust among consumers, as brands can demonstrate their commitment to sustainability and accountability.

Despite its benefits, regions like Tunisia face challenges in implementing AI due to limited infrastructure and insufficient technical expertise. The Tunisian Textile Report 2023 highlighted the need for targeted training programs and financial incentives to bridge this gap. With proper support, the Tunisian textile industry could leverage AI to enhance competitiveness and meet global standards [12].

AI in Education and Training

AI's impact is not limited to production and retail but extends to education, where it equips students with the skills needed for a technology-driven industry. Lee and Suh 2024 introduced the Technological Pedagogical Content Knowledge (TPACK) framework, emphasizing the integration of AI tools like ChatGPT and Midjourney into fashion design education. These tools enable students to quickly iterate on design concepts, visualize their ideas, and receive feedback, fostering creativity and efficiency in learning [13].

Additionally, the Cultural AI Archives 2023 discussed how AI helps preserve traditional design knowledge by digitizing patterns and techniques. This ensures that students learn the importance of cultural heritage while incorporating these elements into modern, sustainable designs [13]. By combining traditional skills with AI-driven tools, educational programs can prepare students for the evolving demands of the fashion industry.

Challenges and Ethical Concerns

Despite its numerous advantages, AI presents ethical challenges that the fashion industry must address. Rathore 2017 highlighted concerns such as data privacy, algorithmic bias, and job displacement. Without proper oversight, AI systems may reinforce biases present in their training data, leading to unfair outcomes in areas like hiring or product recommendations. Furthermore, over-reliance on AI risks sidelining human creativity, raising concerns about the future of traditional craftsmanship [14].

The Hum (AI)n Concept 2024 emphasizes the need for a collaborative approach where AI complements human creativity rather than replacing it. By establishing ethical frameworks and transparency in AI use, the fashion industry can ensure fairness, inclusivity, and trust among stakeholders [15].

Future Trends and Opportunities

Looking ahead, AI is poised to drive innovation and sustainability in fashion. Kathuria and Chaudhary 2024 suggested fostering collaborations between policymakers, educators, and industry leaders to create a balanced ecosystem for responsible AI use. They emphasized the importance of energy-efficient AI systems and transparent practices to address ethical concerns while maximizing AI's benefits [16].

AI Sustainability Reports 2023 predicted that predictive analytics and smart production systems will continue to optimize supply chains, reducing waste and aligning production with consumer demand. These technologies will play a key role in promoting slow fashion by supporting conscious consumerism and minimizing overproduction [17].

IoT Tools for Smart Clothing 2023 explored how IoT-enabled systems enhance production efficiency by offering real-time insights into material usage and manufacturing processes. Combined with blockchain for traceability, these tools will further align fashion practices with sustainability goals [18].

GAN Technologies in Design 2023 highlighted AI's role in promoting slow fashion by forecasting demand and reducing overproduction. This supports conscious consumerism while aligning with sustainability goals [19].

Finally, the Global Fashion Trends Report 2023 concluded that combining AI with technologies like augmented reality (AR) and Internet of Things (IoT) will revolutionize the fashion industry by enhancing consumer experiences and achieving sustainability [20].

Method

This project uses a methodology to develop an AI-powered e-commerce platform which unites recommendation services with safe payment processing. The development framework for backend functions relies on Django alongside HTML CSS and JavaScript which compose the frontend component delivering a smooth user experience. An integrated AI model within a camera function analyses user-submitted images through which it recommends clothing items matching face shape and body types. The web platform provides users with product browsing followed by cart function and secure payment through Paytm. The platform maintains data through an SQLite database system to save user credentials together with product information along with transaction records. User authentication when combined with data encryption and secure API integration functions as security measures to protect the database. Through extensive testing the system goes through unit and integration tests to verify its future as well as improve user interaction. Customers benefit from AI recommendation tools combined with straightforward navigation and transaction security when they use the smart efficient shopping platform.

Flow chart

The e-commerce project begins when the user visits the website. As an additional feature, the system integrates an AI-powered recommendation system with a camera function. The user can upload their image, which the AI analyses to suggest clothing items that best suit them. Based on these recommendations, the user can either choose a suggested product or browse the catalog manually.

Once a product is selected, the user adds it to the cart and proceeds to checkout. At this stage, authentication is required, prompting the user to either log in or sign up. After successful authentication, the checkout process begins, where the system verifies essential details such as the user's name, email, address, selected items, and total amount. These details are stored in the database for order processing.

Next, the system directs the user to the payment gateway (such as Paytm) for transaction completion. If the payment is successful, the system confirms the order, updates the database, and sends a confirmation message to the user. If the payment fails, the user is notified and given the option to retry. Once the order is confirmed, users can access their profile to view their order history and manage personal details. The process concludes with the successful placement of the order, marking the end of the transaction flow.

This structured approach ensures smooth AI-based recommendations, user-friendly shopping, secure authentication, seamless payment integration, and efficient order management.

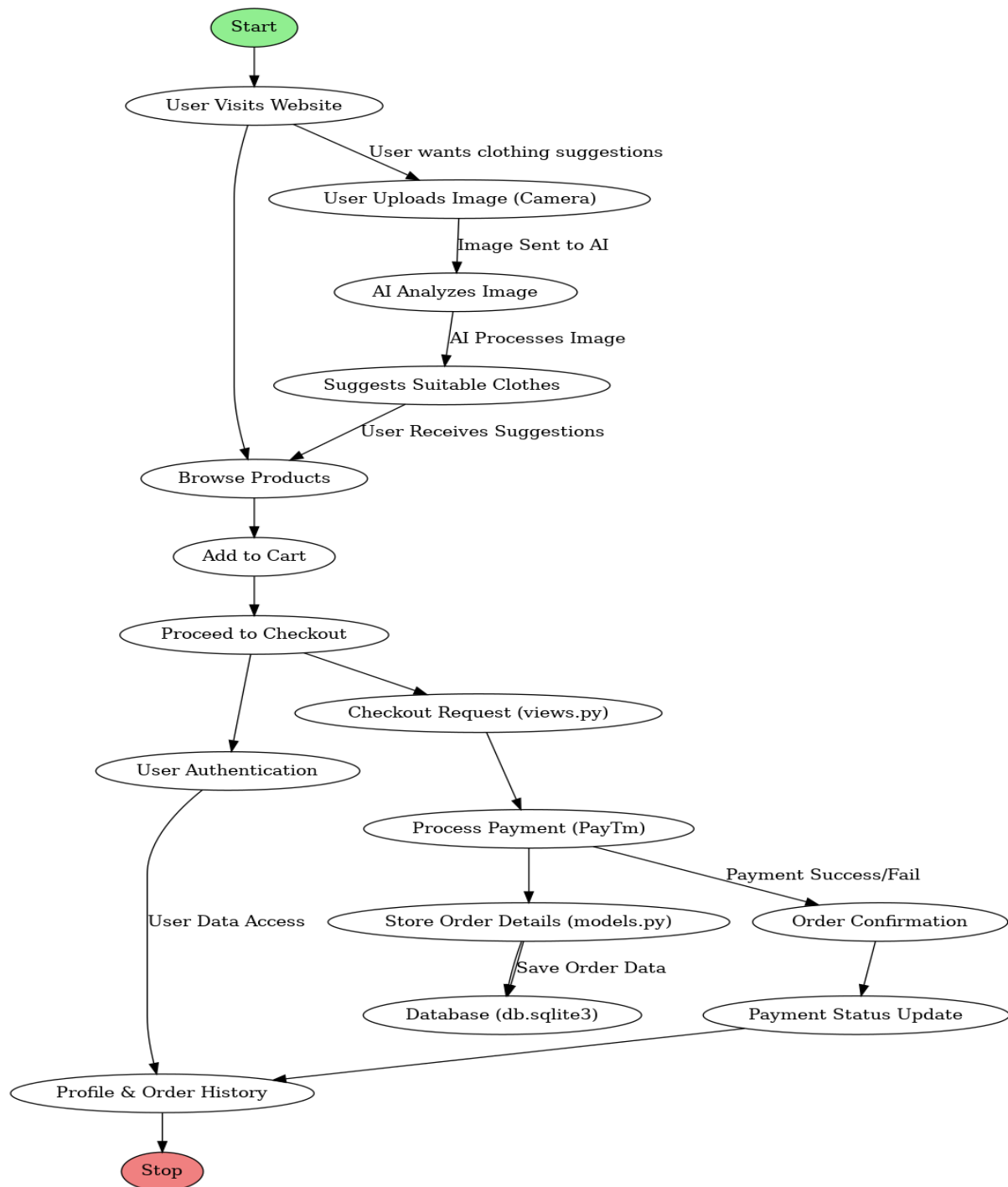


Fig1: flow chart

Tools

The project employs the use of a combination of advanced technologies to ensure efficient performance and a seamless user experience. Django is the main backend framework used for authentication, database interactions, and API integrations. SQLite is used to store user and product data for lightweight and efficient database management. The frontend is built using HTML, CSS, and JavaScript for a responsive and interactive interface. The AI model, which is powered by machine learning and OpenCV, analyses the user images to suggest appropriate clothing. Paytm is integrated for secure online transactions, and Git and GitHub are used for version control and collaborative development. All these tools collectively enhance the functionality, security, and scalability of the e-commerce platform.

1. Django (Python Framework)

Django is a high-level Python web framework that streamlines the development of complex web applications. It uses the Model-View-Template (MVT) architecture so that the entire backend is structured and scalable. Here, Django assists the development team in managing user authentication, product handling, and integrating APIs. It also assists with database queries, templates to render, and security by protecting against SQL injection and XSS attacks.

2. SQLite Database

SQLite is a small relational database management system used for storing data about users, listings of products, orders, and payment transactions. SQLite is great for small- to medium-sized applications because it is serverless, making it perfect for this project. This database is hooked up with Django's ORM that allows efficient handling of data with Python code and does not write complex SQL queries.

3. HTML, CSS, and JavaScript

The frontend of the e-commerce website is developed by using standard web technologies as listed below:

HTML (Hypertext Markup Language): It structures web pages and gives the content structure.

CSS (Cascading Style Sheets): It styles web pages to provide a good experience for users with a beautiful look and responsive design.

JavaScript: It adds interactivity to the platform, making it handle events such as adding items to the cart, filters, and other dynamic updates that do not need to reload the page.

4. AI Model (Machine Learning)

The project includes an AI-powered recommendation system that suggests suitable clothing based on user-uploaded images. The AI model analyses face shape, skin tone, and body proportions using machine learning algorithms and deep learning techniques. The model is trained on a dataset of various body types and fashion styles, helping users make personalized clothing choices.

5. OpenCV (Computer Vision Library)

OpenCV, or Open-Source Computer Vision, is used for image processing and real-time camera capture. As soon as the user uploads the photo, OpenCV detects facial features and body structure, which are then processed by the AI model for clothing recommendations. It also helps in basic image preprocessing like resizing, filtering, and background removal.

6. Paytm Payment Gateway

For safe online transactions, Paytm is integrated with the platform. Here, one can pay via credit/debit cards, UPI, net banking, or digital wallets. This payment gateway makes sure the details of a transaction are encrypted and there will not be any breach of data and unauthorized access.

7. VS Code (Visual Studio Code)

The main code editor for the project's development is VS Code. This editor has multiple extensions for Python, Django, and JavaScript that make debugging and code management much easier. An integrated terminal helps to run commands for Django, database migrations, and version control without a hitch.

8. API

API keys like gamin slash and hugging face are used to communicate. API is used to communicate the frontend and backend. In this project, APIs are used for user authentication, order placement, AI-based recommendations, and payment processing. The data is exchanged between the frontend and backend in JSON (JavaScript Object Notation) format to ensure fast and efficient data transmission. API keys like gamin slash and hugging face are used to communicate.

These tools work together to help the AI-driven e-commerce site run smoothly and deliver a smooth shopping experience to the user with recommendations and safe transactions.

Results and Discussion

The AI Fashion Trends platform successfully integrates artificial intelligence to analyse clothing trends, generate fashion recommendations, and enhance user engagement through an interactive web interface. The system's multi-functional approach—combining image capture, AI-driven analysis, and generative image models—demonstrates significant potential in transforming how users interact with fashion technology.

Webcam-Based Image Capture and Analysis

One of the major functionalities of this platform is the ability to capture real-time images using a webcam. The system was tested under various lighting conditions, and the results showed that image quality played a significant role in the accuracy of AI-based clothing analysis. In well-lit environments, the model successfully identified clothing patterns, colours, and general styles, whereas images taken in dim or overexposed conditions resulted in lower accuracy. This highlights the importance of preprocessing techniques such as brightness correction and noise reduction to improve analysis reliability.

Additionally, the system efficiently processed the captured images and sent them for analysis without noticeable lag. The ease of access to the webcam and the ability to instantly capture and analyse an image enhanced user engagement. However, challenges were observed in accurately differentiating between overlapping clothing layers and complex textures, indicating potential improvements in fine-tuning the AI model to handle such nuances.

Gemini Image Analysis and Fashion Recommendations

The Gemini Image Analysis component played a crucial role in analysing uploaded images and generating appropriate fashion recommendations. The results indicated that the model successfully categorized clothing styles and suggested relevant outfit choices based on user input. The AI-generated recommendations for traditional and international clothing were particularly well-received, as they provided diverse styling options catering to different fashion preferences.

However, it was noted that the model sometimes struggled with distinguishing between subtle variations in fabric patterns and intricate embroidery details, which can be critical in high-fashion or traditional attire. This suggests that further refinement using an expanded dataset, particularly one that includes a diverse range of fashion styles from different cultures, could enhance the system's recommendation accuracy.

Another key observation was that predefined prompts significantly improved recommendation accuracy. By allowing users to specify whether they wanted traditional or international clothing suggestions, the model generated more contextually appropriate results. This user-driven customization feature made the system more intuitive and adaptable to individual preferences.

AI-Generated Clothing Images

The integration of AI-generated fashion images provided a novel way for users to visualize clothing based on textual descriptions. During testing, the generative model produced highly detailed images for common clothing types such as T-shirts, jeans, and sarees. The ability to select a clothing type dynamically updated the text prompt, ensuring more accurate image generation.

While the generated images were generally accurate, challenges arose when attempting to depict complex traditional outfits with intricate embroidery, layering, or culturally specific designs. In some cases, the AI-generated images deviated slightly from traditional expectations, suggesting that additional training on region-specific fashion trends could enhance precision. Additionally, incorporating user feedback mechanisms—such as allowing users to rate or refine generated images—could improve overall satisfaction and usability.

User Experience and Interface Design

The platform's user interface was designed with a modern and visually appealing aesthetic, incorporating smooth animations, a gradient background, and interactive elements. Users responded positively to the clean layout, intuitive form inputs, and well-structured recommendation sections. The inclusion of real-time visual feedback (such as displaying captured images and generated clothing recommendations) enhanced user engagement.

One of the most appreciated features was the dynamic text input adjustment based on clothing type selection, which streamlined the process of generating AI-driven fashion visuals. Additionally, the recommendations section, which provided users with product details such as price, brand, size, and color, proved effective in simulating a real-world shopping experience.

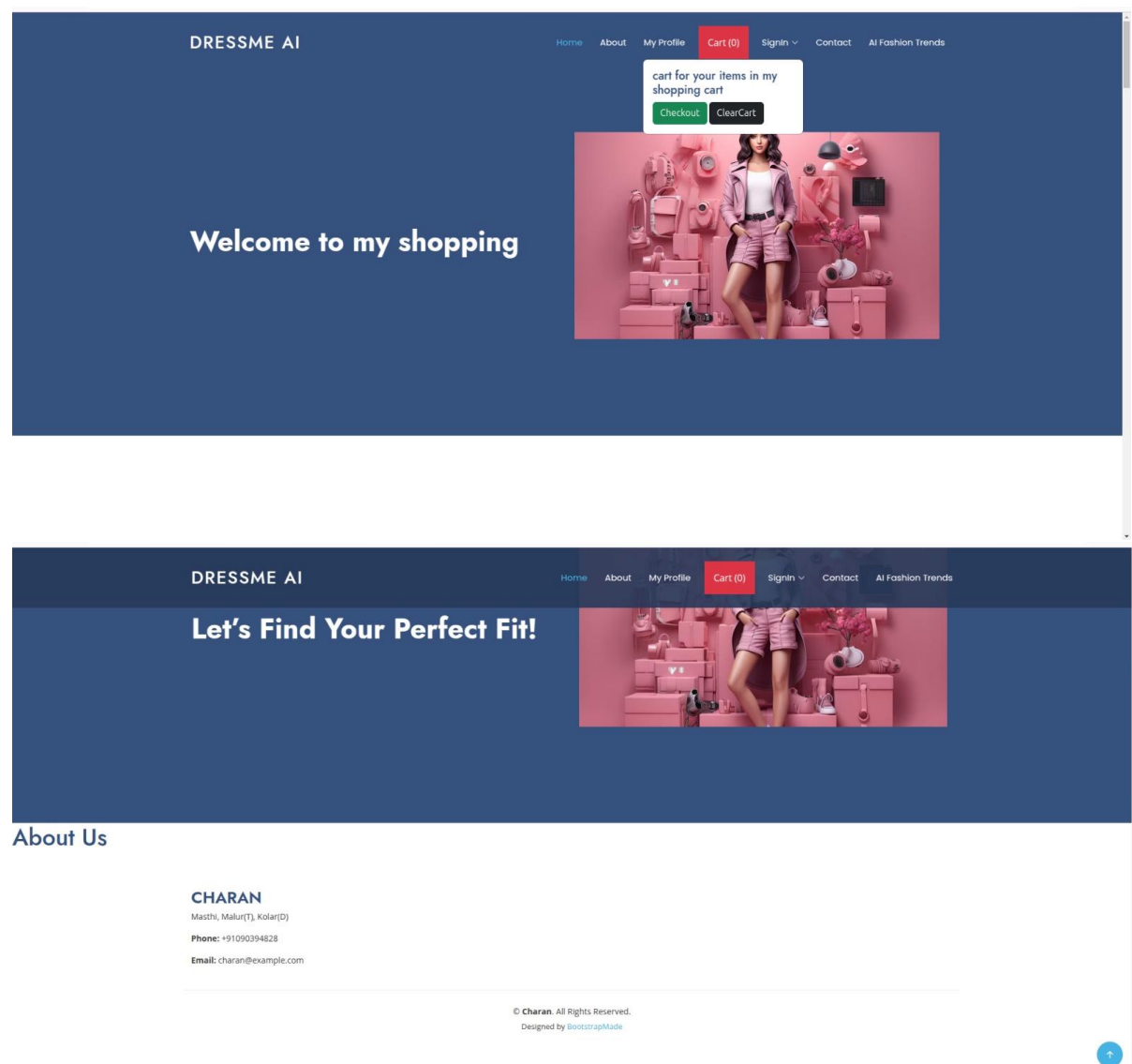
Challenges and Areas for Improvement

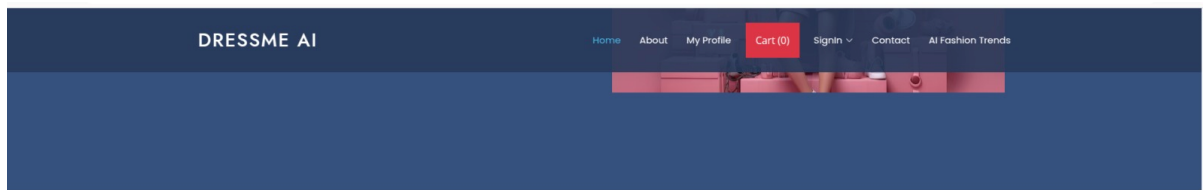
Despite the system's success, a few challenges were identified that could be addressed in future iterations:

1. Improving AI Model Accuracy – The fashion recognition model could benefit from additional training on a more diverse dataset to better distinguish subtle variations in styles, patterns, and cultural influences.

2. Handling Real-Time Constraints – While the platform performed efficiently, real-time image processing could be further optimized to reduce latency, particularly for large image files or high-resolution video inputs.
3. Enhancing Customization – Allowing users to provide feedback on recommendations and AI-generated images could improve personalization and refine model accuracy.
4. Better Handling of Complex Outfits – The AI model should be enhanced to recognize layered clothing, accessories, and fabric textures more effectively.
5. Integration with E-Commerce – Linking AI-generated recommendations with real-time e-commerce data could improve user experience by allowing direct purchasing options.

Screenshots





My Profile

Order ID	Name	Products	Amount Paid	Payment Status	Address	Phone Number	Delivery Status	Delivered	Date
----------	------	----------	-------------	----------------	---------	--------------	-----------------	-----------	------

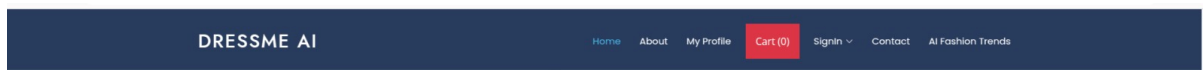
CHARAN

Masthi, Malur(T), Kolar(D)

Phone: +91090394828

Email: charan@example.com

© Charan. All Rights Reserved.
Designed by [BootstrapMade](#)



Contact Us

No valid orders found. X

No valid orders found. X

No valid orders found. X

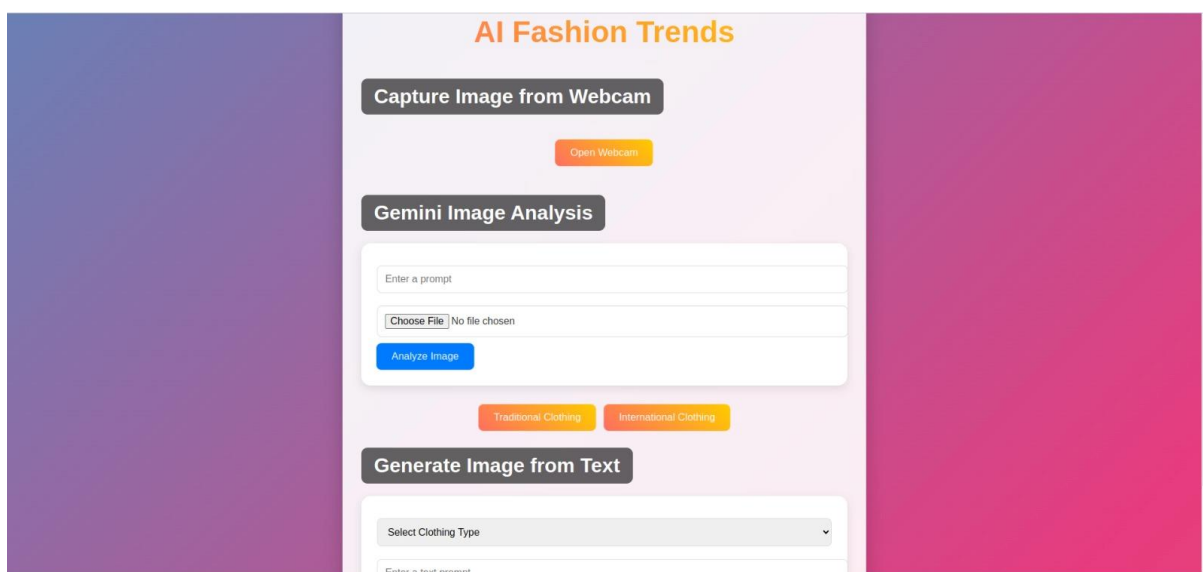
Full Name

Email

How Can I Help You?

Phone Number

CHARAN



Open Webcam

Gemini Image Analysis

Enter a prompt

Choose File | No file chosen

Analyze Image

Traditional Clothing

International Clothing

Generate Image from Text


Select Clothing Type

Enter a text prompt

Generate Image

Response:

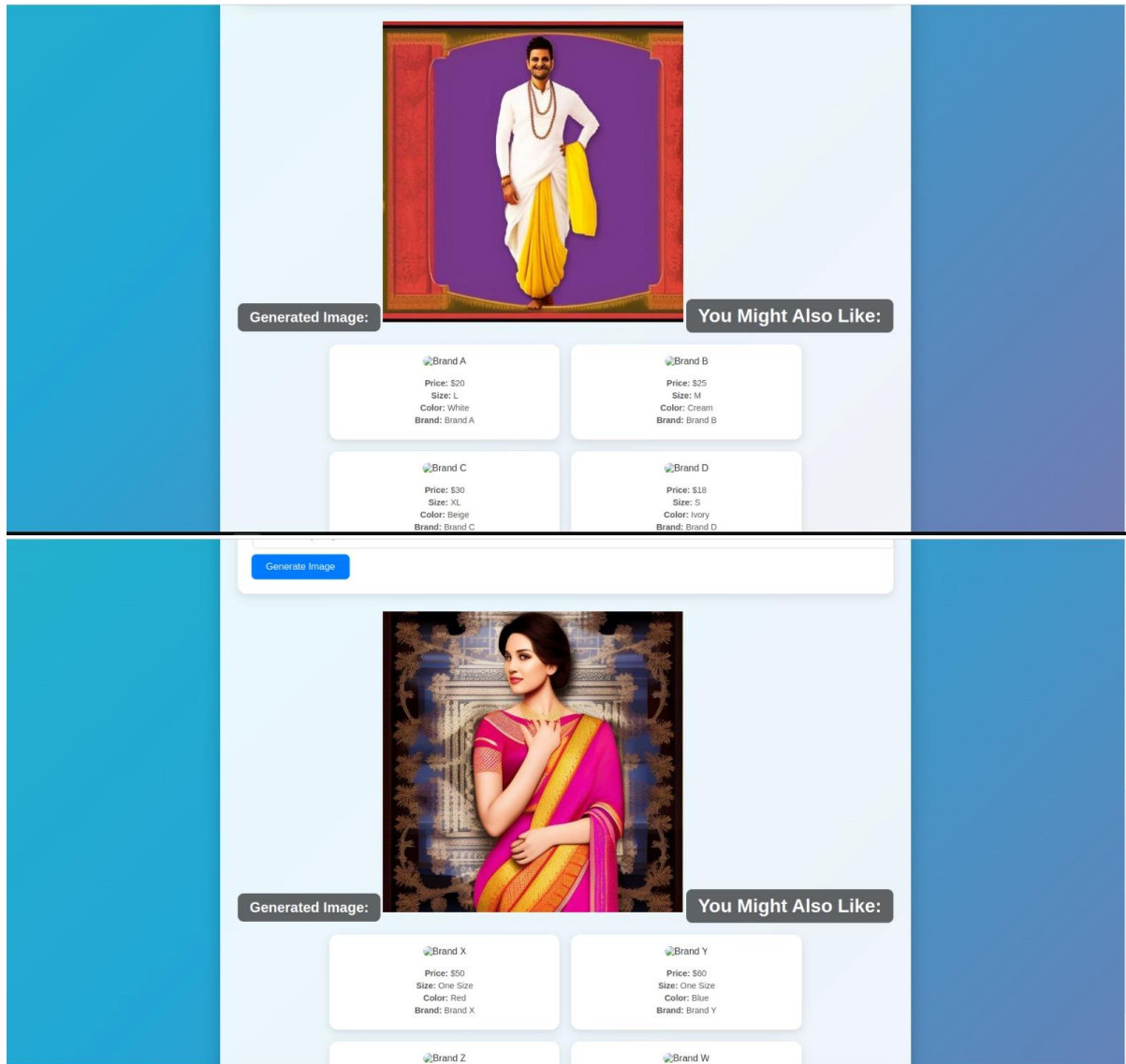
Analysis Result: Here's a caption for the image: ****Option 1 (Simple):**** > A headshot of a young man. ****Option 2 (More descriptive):**** > A portrait of a man with dark hair, wearing a dark patterned shirt. He appears to be in an office setting. ****Option 3 (If you know his name):**** > [Name], looking sharp in his office. Choose the option that best suits your needs. If you have any more information about the image (context, etc.), please share it and I can help craft a more specific caption.



Generate Image from Text

Dhoti

Generate an image of a stylish dhoti



Conclusion

Artificial Intelligence is reshaping the fashion industry by enabling unprecedented levels of innovation, personalization, and sustainability. AI-powered tools like GANs, AR, and blockchain have revolutionized how brands interact with consumers, design products, and manage supply chains. These technologies enhance efficiency while promoting eco-friendly practices, making AI a key driver of progress in the industry.

Despite its benefits, the adoption of AI presents significant challenges, including data privacy concerns, algorithmic bias, and the potential displacement of human creativity. Addressing these issues requires ethical AI implementation and transparent practices to build trust and foster inclusivity. Furthermore, balancing

the integration of AI with traditional craftsmanship and human ingenuity is crucial to preserving the industry's cultural and creative essence.

The future of AI in fashion lies in its ability to harmonize technological advancements with environmental and social responsibilities. By fostering collaborations among stakeholders, investing in education, and prioritizing ethical standards, the fashion industry can harness AI's transformative potential to achieve sustainable growth and innovation. Ultimately, AI offers an opportunity to redefine the fashion landscape, creating a more inclusive, efficient, and environmentally conscious industry.

Acknowledgement

we would like to express my deepest gratitude to Dr. Setturu Bharath, whose guidance, patience, and insightful feedback have been invaluable throughout this research. Their expertise and encouragement played a crucial role in shaping the direction of this study.

we also extend my appreciation to Chanakya university and school of engineering for providing the necessary resources and a conducive environment for conducting this research.

we sincerely grateful to my colleagues and research team, Charan G, Deeksha Y V, Siri Paladi, Lavanya K R, for their constructive discussions, technical assistance, and support throughout the research process. Their valuable input and cooperation greatly enriched this study.

Finally, we would like to thank my family and friends for their unwavering support, patience, and motivation during this journey. Their encouragement has been a source of strength, and we truly appreciate their belief in us.

Code

1.django: views.py

```
import os
import io
import time
import base64
import json
import requests
from math import ceil
from datetime import datetime
from PIL import Image
from django.shortcuts import render, redirect
from django.http import JsonResponse
from django.conf import settings
from django.core.files.storage import default_storage
from django.core.files.base import ContentFile
from django.views.decorators.csrf import csrf_exempt
from django.contrib import messages
from dotenv import load_dotenv
from typing import Dict, List
import google.generativeai as genai
from PayTm import Checksum
from ecommerceapp.models import Contact, Product, OrderUpdate, Orders
from ecommerceapp import keys

# Load environment variables
load_dotenv()

# Configure Google Generative AI
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
```

```

# Hugging Face API settings
HF_API_KEY = os.getenv("HF_API_KEY")
API_URL = "https://api-inference.huggingface.co/models/stabilityai/stable-diffusion-2-1-base"
headers = {"Authorization": f"Bearer {HF_API_KEY}"}

# Ensure directories exist for image storage
CAPTURED_IMAGES_DIR = os.path.join(settings.MEDIA_ROOT, "captured_images")
GENERATED_IMAGES_DIR = os.path.join(settings.MEDIA_ROOT, "generated_images")
os.makedirs(CAPTURED_IMAGES_DIR, exist_ok=True)
os.makedirs(GENERATED_IMAGES_DIR, exist_ok=True)

# Recommendations dataset with correct image paths
RECOMMENDATIONS: Dict[str, List[Dict[str, str]]] = {
    "Saree": [
        {"image": "ecommerce/media/Saree3.jpg", "price": "$50", "size": "One Size", "color": "Red",
         "brand": "Brand X"},
        {"image": "ecommerce/media/Saree2.webp", "price": "$60", "size": "One Size", "color": "Blue",
         "brand": "Brand Y"},
        {"image": "ecommerce/media/Saree1.jpg", "price": "$70", "size": "One Size", "color": "Green",
         "brand": "Brand Z"},
        {"image": "/home/princedj/Downloads/Saree4.jpg", "price": "$80", "size": "One Size", "color":
         "Yellow", "brand": "Brand W"}
    ],
    "Dhoti": [
        {"image": "images/Dhoti2.jpg", "price": "$20", "size": "L", "color": "White", "brand": "Brand
A"},
        {"image": "images/Dhoti1.jpg", "price": "$25", "size": "M", "color": "Cream", "brand": "Brand
B"},
        {"image": "images/Dhoti4.webp", "price": "$30", "size": "XL", "color": "Beige", "brand": "Brand
C"},
        {"image": "images/Dhoti5.webp", "price": "$18", "size": "S", "color": "Ivory", "brand": "Brand
D"}
    ],
}

```

```
# Home Page View
```

```
def index(request):
```

```
    allProds = []
```

```
    catprods = Product.objects.values('category', 'id')
```

```
    cats = {item['category'] for item in catprods}
```

```
    for cat in cats:
```

```
        prod = Product.objects.filter(category=cat)
```

```
        n = len(prod)
```

```
        nSlides = n // 4 + ceil((n / 4) - (n // 4))
```

```
        allProds.append([prod, range(1, nSlides), nSlides])
```

```
    params = {'allProds': allProds}
```

```
    return render(request, "index.html", params)
```

```
# Contact View
```

```
def contact(request):
```

```
    if request.method == "POST":
```

```
        name = request.POST.get("name")
```

```
        email = request.POST.get("email")
```

```
        desc = request.POST.get("desc")
```

```
        pnumber = request.POST.get("pnumber")
```

```
        myquery = Contact(name=name, email=email, desc=desc, phonenumber=pnumber)
```

```
        myquery.save()
```

```
        messages.info(request, "We will get back to you soon.")
```

```
        return render(request, "contact.html")
```

```
    return render(request, "contact.html")
```

```
# About View
```

```
def about(request):
```

```
    return render(request, "about.html")
```

```
# Checkout View
```

```
def checkout(request):
```

```
    if request.method == "POST":
```

```
        items_json = request.POST.get('itemsJson', '')
```

```
        name = request.POST.get('name', '')
```

```
        amount = request.POST.get('amt')
```

```
        email = request.POST.get('email', '')
```

```
        address1 = request.POST.get('address1', '')
```

```
        address2 = request.POST.get('address2', '')
```

```
        city = request.POST.get('city', '')
```

```
        state = request.POST.get('state', '')
```

```
        zip_code = request.POST.get('zip_code', '')
```

```
        phone = request.POST.get('phone', '')
```

```
        Order = Orders(items_json=items_json, name=name, amount=amount, email=email,  
address1=address1, address2=address2, city=city, state=state, zip_code=zip_code, phone=phone)
```

```
        Order.save()
```

```
        update = OrderUpdate(order_id=Order.order_id, update_desc="The order has been placed")
```

```
        update.save()
```

```
# PAYMENT INTEGRATION
```

```
id = Order.order_id
```

```
oid = str(id) + "ShopyCart"
```

```
param_dict = {
```

```
    'MID': keys.MID,
```

```
    'ORDER_ID': oid,
```

```
    'TXN_AMOUNT': str(amount),
```

```
    'CUST_ID': email,
```

```
    'INDUSTRY_TYPE_ID': 'Retail',
```

```
    'WEBSITE': 'WEBSTAGING',
```

```
    'CHANNEL_ID': 'WEB',
```

```
    'CALLBACK_URL': 'http://127.0.0.1:8000/handlerequest/',
```

```
}
```

```
param_dict['CHECKSUMHASH'] = Checksum.generate_checksum(param_dict, keys.MK)
```

```

        return render(request, 'paytm.html', {'param_dict': param_dict})

    return render(request, 'checkout.html')

# Handle Payment Request
@csrf_exempt
def handlerequest(request):
    form = request.POST
    response_dict = {}
    for i in form.keys():
        response_dict[i] = form[i]
        if i == 'CHECKSUMHASH':
            checksum = form[i]

    verify = Checksum.verify_checksum(response_dict, keys.MK, checksum)
    if verify:
        if response_dict['RESPCODE'] == '01':
            print('Order successful')
            a = response_dict['ORDERID']
            b = response_dict['TXNAMOUNT']
            rid = a.replace("ShopyCart", "")

            if rid.isdigit():
                filter2 = Orders.objects.filter(order_id=int(rid))
                for post1 in filter2:
                    post1.oid = a
                    post1.amountpaid = b
                    post1.paymentstatus = "PAID"
                    post1.save()
            else:
                print("Invalid order ID format")
        else:

```



```

        print('Order was not successful because ' + response_dict['RESPMSG'])
    return render(request, 'paymentstatus.html', {'response': response_dict})

# Profile View
def profile(request):
    currentuser = request.user.username
    items = Orders.objects.filter(email=currentuser)
    rid = None
    status = []

    for i in items:
        if i.oid: # Ensure oid is not None or empty
            rid = i.oid.replace("ShopyCart", "") # Remove "ShopyCart" from oid
            if not rid.isdigit(): # Check if rid is a valid integer
                rid = None
                continue # Skip to the next item if rid is invalid
            rid = int(rid) # Convert rid to an integer
            break # Use the first valid rid found

    if rid is not None: # Only query OrderUpdate if rid is valid
        status = OrderUpdate.objects.filter(order_id=rid)
    else:
        messages.warning(request, "No valid orders found.")

    context = {"items": items, "status": status}
    return render(request, "profile.html", context)

# AI Fashion Trends View
def ai_fashion_trends(request):
    if request.method == "POST":
        action = request.POST.get("action")

```

```

if action == "gemini_analysis":
    input_text = request.POST.get("input_text", "")
    uploaded_file = request.FILES.get("uploaded_file")

    if uploaded_file:
        # Save the uploaded file to media directory
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        file_name = f"uploaded_{timestamp}.jpg"
        file_path = os.path.join(CAPTURED_IMAGES_DIR, file_name)

        with open(file_path, "wb") as f:
            for chunk in uploaded_file.chunks():
                f.write(chunk)

        # Generate file URL
        file_url = f"{settings.MEDIA_URL}captured_images/{file_name}"

        # Process image with Gemini AI
        image = Image.open(file_path)
        response = get_gemini_response(input_text, image)

        return render(request, "ai_fashion_trends.html", {"response": response, "image_url":
file_url})
    else:
        return render(request, "ai_fashion_trends.html", {"error": "No image uploaded."})

elif action == "capture_image":
    return capture_image(request) # Call the capture image function directly

elif action == "generate_image":
    input_text = request.POST.get("input_text", "")
    if input_text:
        result = generate_image_from_text(input_text)

```

```

if result:
    generated_image, file_name = result
    # Generate the URL for the generated image
    image_url = f'{settings.MEDIA_URL}generated_images/{file_name}'

    # Get recommendations based on input text
    suggested_items = []
    for category, items in RECOMMENDATIONS.items():
        if category.lower() in input_text.lower(): # Case-insensitive match
            suggested_items.extend(items)

    return render(request, "ai_fashion_trends.html", {
        "generated_image_url": image_url,
        "suggested_items": suggested_items,
        "input_text": input_text
    })
else:
    return render(request, "ai_fashion_trends.html", {"error": "Failed to generate image."})

return render(request, "ai_fashion_trends.html")

# Function to get Gemini response
def get_gemini_response(input_text, image):
    try:
        model = genai.GenerativeModel("gemini-1.5-flash")
        response = model.generate_content([input_text, image] if input_text else [image])
        return response.text if response else "No response from AI."
    except Exception as e:
        return f'An error occurred: {str(e)}'

def generate_image_from_text(input_text):
    payload = {"inputs": input_text, "options": {"wait_for_model": True}}

```

```

max_retries = 5 # Increased retries
backoff_time = 5 # Initial backoff time in seconds

for attempt in range(1, max_retries + 1):
    try:
        response = requests.post(API_URL, headers=headers, json=payload, timeout=120) # Increased
        timeout to 120s
        print(f'Attempt {attempt}: Status {response.status_code}')

        if response.status_code == 200:
            content_type = response.headers.get('Content-Type', '')
            if content_type.startswith('image/'):
                image = Image.open(io.BytesIO(response.content))
                timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
                file_name = f'{timestamp}_generated_image.jpg'
                image_path = os.path.join(GENERATED_IMAGES_DIR, file_name)
                image.save(image_path)
                return image, file_name
            else:
                error_message = response.json()
                print(f'API Response (JSON): {error_message}')
                return None

        elif response.status_code in [503, 504]: # Handle both 503 and 504
            estimated_time = response.json().get("estimated_time", backoff_time)
            print(f'Server busy. Retrying after {estimated_time + backoff_time} seconds...')
            time.sleep(estimated_time + backoff_time)
            backoff_time = min(backoff_time * 2, 60) # Exponential backoff, capped at 60s
        else:
            print(f'API Error: {response.status_code}, {response.text}')
            return None

    except requests.exceptions.Timeout:

```

```

        print(f'Request timed out on attempt {attempt}. Retrying...")
        time.sleep(backoff_time)
        backoff_time = min(backoff_time * 2, 60) # Cap backoff time

    except Exception as e:
        print(f'Unexpected error: {str(e)}")
        return None

return None


# Handle image capture
@csrf_exempt
def capture_image(request):
    """Handles image upload from the frontend and saves the image properly."""
    if request.method == "POST" and request.body:
        try:
            # Get base64 image data from the request
            data = request.body.decode('utf-8')
            header, encoded_image = data.split(",", 1)

            # Decode the image
            image_data = base64.b64decode(encoded_image)

            # Create a unique filename with timestamp
            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
            file_name = f'image_{timestamp}.jpg"
            relative_path = os.path.join("captured_images", file_name)

            # Save the image to the appropriate directory

```

```

        saved_path = default_storage.save(relative_path, ContentFile(image_data))

        # Get full URL path for the image
        full_url = f'{settings.MEDIA_URL}{saved_path}'

        # Return the response with the file URL
        return JsonResponse({"message": "Image successfully saved", "file_url": full_url})
    except Exception as e:
        return JsonResponse({"error": f'Failed to process image: {str(e)}'}, status=400)

return JsonResponse({"error": "Invalid request"}, status=400)

```

2.Html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AI Fashion Trends</title>
    <style>
        /* General Styles */
        body {
            font-family: 'Poppins', sans-serif;
            margin: 0;
            padding: 0;
            background: linear-gradient(-45deg, #ee7752, #e73c7e, #23a6d5, #23d5ab);
            background-size: 400% 400%;
            animation: gradientBackground 15s ease infinite;
            color: #333;
            min-height: 100vh;
            display: flex;
            justify-content: center;

```

```
    align-items: center;
}

@keyframes gradientBackground {
    0% { background-position: 0% 50%; }
    50% { background-position: 100% 50%; }
    100% { background-position: 0% 50%; }
}

h1, h2, h3 {
    text-align: center;
    margin-bottom: 20px;
}

h1 {
    font-size: 3rem;
    background: linear-gradient(45deg, #ff6f61, #ffcc00);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
    animation: fadeIn 1.5s ease-in-out;
}

h2 {
    font-size: 2rem;
    color: #fff;
    background: rgba(0, 0, 0, 0.6);
    padding: 10px 20px;
    border-radius: 10px;
    display: inline-block;
    animation: slideIn 1s ease-in-out;
}
```

```

h3 {
    font-size: 1.5rem;
    color: #fff;
    background: rgba(0, 0, 0, 0.6);
    padding: 10px 20px;
    border-radius: 10px;
    display: inline-block;
}

.container {
    max-width: 1200px;
    margin: 20px;
    padding: 30px;
    background: rgba(255, 255, 255, 0.9);
    border-radius: 20px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.2);
    animation: fadeInUp 1.5s ease-in-out;
}

/* Form Styles */
form {
    background: #fff;
    padding: 25px;
    border-radius: 15px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
    margin-bottom: 30px;
    animation: fadeInUp 1s ease-in-out;
}

input[type="text"], input[type="file"], select {
    width: 100%;
    padding: 12px;

```



```

margin: 10px 0;
border: 1px solid #ddd;
border-radius: 8px;
font-size: 1rem;
transition: border-color 0.3s ease, box-shadow 0.3s ease;
}

input[type="text"]:focus, input[type="file"]:focus, select:focus {
    border-color: #007bff;
    box-shadow: 0 0 8px rgba(0, 123, 255, 0.3);
}

button, input[type="submit"] {
    padding: 12px 24px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-size: 1rem;
    transition: background-color 0.3s ease, transform 0.3s ease;
}

button:hover, input[type="submit"]:hover {
    background-color: #0056b3;
    transform: translateY(-2px);
}

/* Webcam Section */
#webcam {
    width: 100%;
    max-width: 640px;

```

```

border-radius: 15px;
box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
margin: 20px 0;
display: none;
}

#canvas {
  display: none;
}

#capture-result {
  text-align: center;
  margin: 20px 0;
}

/* Recommendations Section */
.recommendations {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  justify-content: center;
}

.recommendation-item {
  background: #fff;
  padding: 20px;
  border-radius: 15px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
  width: calc(33.333% - 20px);
  text-align: center;
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}

```

```
.recommendation-item:hover {  
    transform: translateY(-10px);  
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.2);  
}
```

```
.recommendation-item img {  
    width: 100%;  
    max-height: 200px;  
    object-fit: cover;  
    border-radius: 10px;  
    margin-bottom: 15px;  
    transition: transform 0.3s ease;  
}
```

```
.recommendation-item:hover img {  
    transform: scale(1.05);  
}
```

```
.recommendation-item p {  
    margin: 5px 0;  
    font-size: 0.9rem;  
    color: #555;  
}
```

```
/* Error Messages */
```

```
.error {  
    color: red;  
    text-align: center;  
    margin: 10px 0;  
    font-weight: bold;  
}
```

```
/* Animations */

@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

@keyframes slideIn {
  from { transform: translateY(-20px); opacity: 0; }
  to { transform: translateY(0); opacity: 1; }
}

@keyframes fadeInUp {
  from { transform: translateY(20px); opacity: 0; }
  to { transform: translateY(0); opacity: 1; }
}

/* Responsive Layout */

@media (max-width: 768px) {
  .recommendation-item {
    width: calc(50% - 20px);
  }
}

@media (max-width: 480px) {
  .recommendation-item {
    width: 100%;
  }

  h1 {
    font-size: 2.5rem;
  }
}
```

```

        h2 {
            font-size: 1.8rem;
        }
    }
</style>
</head>
<body>
    <div class="container">
        <h1>AI Fashion Trends</h1>

        <!-- Capture Image from Webcam Section -->
        <h2>Capture Image from Webcam</h2>
        <button id="open-webcam-btn">Open Webcam</button>
        <button id="close-webcam-btn" style="display:none;">Close Webcam</button>
        <button id="capture-face-btn" style="display:none;">Capture Image</button>

        <div id="capture-result"></div>

        <video id="webcam" width="640" height="480" style="display:none; max-width: 100%;"
        autoplay></video>
        <canvas id="canvas" style="display:none;"></canvas>

        <!-- Gemini Image Analysis Section -->
        <h2>Gemini Image Analysis</h2>
        <form method="post" enctype="multipart/form-data">
            {% csrf_token %}
            <input type="hidden" name="action" value="gemini_analysis">
            <input type="text" id="input_text" name="input_text" placeholder="Enter a prompt" required>
            <input type="file" name="uploaded_file" accept="image/*" required>
            <input type="submit" value="Analyze Image">
        </form>
    </div>
</body>
</html>

```

```

<!-- New Buttons for Clothing Recommendation -->

<div style="text-align: center; margin-top: 20px;">

    <button type="button" onclick="setPrompt('Recommend a traditional clothing for this
person')">Recommend Traditional Clothing</button>

    <button type="button" onclick="setPrompt('Recommend an international clothing for this
person')">Recommend International Clothing</button>

</div>


<script>

    function setPrompt(text) {

        document.getElementById('input_text').value = text;

    }

</script>


{% if response %}

    <h3>Response:</h3>

    <p>{{ response }}</p>

{% elif error %}

    <p class="error">{{ error }}</p>

{% endif %}


<!-- Generate Image from Text Section -->

<h2>Generate Image from Text</h2>

<form method="post">

    {% csrf_token %}

    <input type="hidden" name="action" value="generate_image">

    <select id="clothing-type" onchange="updateTextPrompt()">

        <option value="">Select Clothing Type</option>

        <option value="t-shirt">T-Shirt</option>

        <option value="jeans">Jeans</option>

        <option value="saree">Saree</option>

        <option value="kurta">Kurta</option>

```

```

        <option value="dhoti">Dhoti</option>

    </select>

    <input type="text" id="text-prompt" name="input_text" placeholder="Enter a text prompt"
required>

    <input type="submit" value="Generate Image">

</form>

{% if generated_image_url %}

    <h3>Generated Image:</h3>

    {% elif error %}

        <p class="error">{{ error }}</p>

    {% endif %}

<!-- Suggested Items Section -->

{% if suggested_items %}

    <h2>You Might Also Like:</h2>

    <div class="recommendations">

        {% for item in suggested_items %}

            <div class="recommendation-item">

                <p><strong>Price:</strong> {{ item.price }}</p>

                <p><strong>Size:</strong> {{ item.size }}</p>

                <p><strong>Color:</strong> {{ item.color }}</p>

                <p><strong>Brand:</strong> {{ item.brand }}</p>

            </div>

        {% endfor %}

    </div>

{% endif %}

</div>

<script>

```

```

const video = document.getElementById("webcam");
const canvas = document.getElementById("canvas");
const openWebcamButton = document.getElementById("open-webcam-btn");
const closeWebcamButton = document.getElementById("close-webcam-btn");
const captureFaceButton = document.getElementById("capture-face-btn");
const captureResult = document.getElementById("capture-result");

let webcamStream = null;

async function startWebcam() {
  try {
    webcamStream = await navigator.mediaDevices.getUserMedia({ video: true });
    video.srcObject = webcamStream;
    video.style.display = "block";
    openWebcamButton.style.display = "none";
    closeWebcamButton.style.display = "inline";
    captureFaceButton.style.display = "inline";
  } catch (err) {
    captureResult.innerHTML = <p class="error">Could not access the webcam: ${err}</p>;
  }
}

function stopWebcam() {
  if (webcamStream) {
    webcamStream.getTracks().forEach(track => track.stop());
  }
  video.style.display = "none";
  openWebcamButton.style.display = "inline";
  closeWebcamButton.style.display = "none";
  captureFaceButton.style.display = "none";
}

```



```

async function captureImage() {
  if (video.style.display === "none") {
    captureResult.innerHTML = <p class="error">Webcam is not active. Please start the webcam
first.</p>;
    return;
  }

  canvas.width = video.videoWidth;
  canvas.height = video.videoHeight;
  const ctx = canvas.getContext("2d");
  ctx.drawImage(video, 0, 0);

  const imageData = canvas.toDataURL("image/jpeg");

  captureResult.innerHTML = <p>Capturing image... Please wait.</p>;

  const response = await fetch("/capture_image/", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-CSRFToken": document.querySelector("[name=csrfmiddlewaretoken]").value
    },
    body: JSON.stringify({ image_data: imageData })
  });

  const data = await response.json();
  if (data.file_url) {
    captureResult.innerHTML =
      `<p>Image captured successfully!</p>
      `;
  } else {
    captureResult.innerHTML = <p class="error">Failed to capture image.</p>;
  }
}

```

```

    }
}

openWebcamButton.addEventListener("click", startWebcam);
closeWebcamButton.addEventListener("click", stopWebcam);
captureFaceButton.addEventListener("click", captureImage);

// Function to update text prompt based on selected clothing type
function updateTextPrompt() {
    const clothingType = document.getElementById("clothing-type").value;
    const textPrompt = document.getElementById("text-prompt");

    if (clothingType) {
        textPrompt.value = Generate an image of a stylish ${clothingType};
    } else {
        textPrompt.value = "";
    }
}

</script>
</body>
</html>

```

3.url.py

```

from django.urls import path
from . import views
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path("", views.index, name='index'),
    path('contact/', views.contact, name='contact'),
    path('about/', views.about, name='about'),
    path('checkout/', views.checkout, name='checkout'),

```

```

    path('handlerequest/', views.handlerequest, name='handlerequest'),
    path('profile/', views.profile, name='profile'),
    path('ai_fashion_trends/', views.ai_fashion_trends, name='ai_fashion_trends'),
    path('capture_image/', views.capture_image, name='capture_image'),
]

# Serve media files during development
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

4.tests.py

```

from django.test import TestCase

# Create your tests here.

```

5.models.py

```

from django.db import models

# Create your models here.
class Contact(models.Model):
    # contact_id=models.AutoField(primary_key=True)
    name=models.CharField(max_length=50)
    email=models.EmailField()
    desc=models.TextField(max_length=500)
    phonenumber=models.IntegerField()

    def __int__(self):
        return self.id

class Product(models.Model):
    product_id = models.AutoField

```

```
product_name = models.CharField(max_length=100)
category = models.CharField(max_length=100, default="")
subcategory = models.CharField(max_length=50, default="")
price = models.IntegerField(default=0)
desc = models.CharField(max_length=300)
image = models.ImageField(upload_to='images/images')
```

```
def __str__(self):
    return self.product_name
```

```
class Orders(models.Model):
```

```
    order_id = models.AutoField(primary_key=True)
    items_json = models.CharField(max_length=5000)
    amount = models.IntegerField(default=0)
    name = models.CharField(max_length=90)
    email = models.CharField(max_length=90)
    address1 = models.CharField(max_length=200)
    address2 = models.CharField(max_length=200)
    city = models.CharField(max_length=100)
    state = models.CharField(max_length=100)
    zip_code = models.CharField(max_length=100)
    oid=models.CharField(max_length=150,blank=True)
    amountpaid=models.CharField(max_length=500,blank=True,null=True)
    paymentstatus=models.CharField(max_length=20,blank=True)
    phone = models.CharField(max_length=100,default="")
    def __str__(self):
        return self.name
```

```
class OrderUpdate(models.Model):
```

```

update_id = models.AutoField(primary_key=True)
order_id = models.IntegerField(default="")
update_desc = models.CharField(max_length=5000)
delivered=models.BooleanField(default=False)
timestamp = models.DateField(auto_now_add=True)

def __str__(self):
    return self.update_desc[0:7] + "..."

```

6.admins.py

```

from django.contrib import admin
from ecommerceapp.models import Contact,Product,Orders,OrderUpdate
# Register your models here.
admin.site.register(Contact)
admin.site.register(Product)
admin.site.register(Orders)
admin.site.register(OrderUpdate)

```

7.apps.py

```

from django.apps import AppConfig

class EcommerceappConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'ecommerceapp'

```

8.wsgi.py

```

"""

```

WSGI config for ecommerce project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

<https://docs.djangoproject.com/en/5.1/howto/deployment/wsgi/>

```
"""
```

```
import os
```

```
from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')
```

```
application = get_wsgi_application()
```

9.setting.py

```
from pathlib import Path
```

```
import os
```

```
from dotenv import load_dotenv
```

```
from django.contrib import messages
```

```
# Load environment variables from a .env file
```

```
load_dotenv()
```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.
```

```
BASE_DIR = Path(__file__).resolve().parent.parent
```

```
# Secret Key should not be hardcoded, it should be loaded from environment variables
```

```
SECRET_KEY = os.getenv('SECRET_KEY', 'your-default-secret-key')
```

```
# Security settings
```

```
DEBUG = os.getenv('DEBUG', 'True') == 'True' # Default to True in development
```

```
ALLOWED_HOSTS = os.getenv('ALLOWED_HOSTS', 'localhost,127.0.0.1').split(',')
```

```
# Application definition
```

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'ecommerceapp', # Your app name
    'authcart', # Your other app name
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'ecommerce.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            BASE_DIR / 'templates', # Include your global templates directory
        ],
        'APP_DIRS': True, # Look for templates in app-specific directories
        'OPTIONS': {
            'context_processors': [

```

```

        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
},
]

```

```
WSGI_APPLICATION = 'ecommerce.wsgi.application'
```

```
# Database settings: Using SQLite for development
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

```
# For production, you can use PostgreSQL or MySQL
```

```
# DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.postgresql',
#         'NAME': os.getenv('DB_NAME'),
#         'USER': os.getenv('DB_USER'),
#         'PASSWORD': os.getenv('DB_PASSWORD'),
#         'HOST': os.getenv('DB_HOST', 'localhost'),
#         'PORT': os.getenv('DB_PORT', '5432'),
#     }
# }
```

```
# Password validation
```



```

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Localization
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True

# Email configuration
EMAIL_HOST = os.getenv('EMAIL_HOST', 'smtpout.secureserver.net')
EMAIL_HOST_USER = os.getenv('EMAIL_HOST_USER', 'your-email@example.com')
EMAIL_HOST_PASSWORD = os.getenv('EMAIL_HOST_PASSWORD', 'your-email-password')
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

# Static and media files
STATIC_URL = '/static/'
STATICFILES_DIRS = [

```

```

    os.path.join(BASE_DIR, 'static'),
]
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'ecommerce/media')


# Default primary key field type
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'


# Message tags for bootstrap
MESSAGE_TAGS = {
    messages.ERROR: 'danger',
    messages.SUCCESS: 'success',
    messages.INFO: 'info',
    messages.WARNING: 'warning',
}


# Security settings for production
if not DEBUG:
    CSRF_COOKIE_SECURE = True
    SESSION_COOKIE_SECURE = True
    SECURE_SSL_REDIRECT = True
    X_FRAME_OPTIONS = 'DENY'


# Ensure directories exist for image storage
CAPTURED_IMAGES_DIR = os.path.join(MEDIA_ROOT, 'captured_images')
GENERATED_IMAGES_DIR = os.path.join(MEDIA_ROOT, 'generated_images')


# Ensure these directories exist
os.makedirs(CAPTURED_IMAGES_DIR, exist_ok=True)
os.makedirs(GENERATED_IMAGES_DIR, exist_ok=True)

```

9. asgi.py

```
"""
```

ASGI config for ecommerce project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see

<https://docs.djangoproject.com/en/5.1/howto/deployment/asgi/>

```
"""
```

```
import os
```

```
from django.core.asgi import get_asgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')
```

```
application = get_asgi_application()
```

10. utils.py

```
from django.contrib.auth.tokens import PasswordResetTokenGenerator
```

```
import six
```

```
class TokenGenerator(PasswordResetTokenGenerator):
```

```
    def _make_hash_value(self, user, timestamp):
```

```
        return (six.text_type(user.pk)+six.
```

11. main.py

```
#!/usr/bin/env python
```

```
"""Django's command-line utility for administrative tasks."""
```

```
import os
```

```
import sys
```

```

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

```

Data sources

The project uses multiple data sources for efficient operation, AI-driven clothing recommendations, and secure transactions. The following details each data source with relevant URLs for reference.

1. User Data (Database - SQLite)

User data will be stored in an SQLite database. These will include personal details such as username, email, password encrypted, and contact information. This will be fundamental for authentication, profile management, and personalized shopping experiences.

url: `sqlite://user_data.db`

2. Product Data (E-commerce Catalog Database)

All clothing products listed on the website are held in a database of the product catalog. Details for every entry include name of the product, description, price, sizes offered, colors offered, and image link.

url: `sqlite://product_catalog.db`

3. AI Training Dataset (Fashion Dataset for Machine Learning)

In this project, Gemini AI is utilized along with an API key and merchant key to analyze a structured dataset containing various images of clothing styles, body types, and fashion trends. The dataset helps the AI system recommend clothing that best suits the user's face shape and body type. It includes labeled data from public sources like Fashion MNIST, DeepFashion, or custom-labeled databases. By leveraging Gemini AI, the model processes these images, identifies emerging trends, and provides data-driven fashion recommendations, enhancing personalization and accuracy in trend forecasting.

url: `local_storage/fashion_images/` or `sqlite://ai_training_data.db`

4. Payment Transaction Data (PayTm API)

All payment transactions are processed securely through the PayTm API. The transaction details include order ID, payment amount, transaction status (success/failure), and user ID.

url: `https://developer.paytm.com/docs/api`

5. Order History Data (Database - SQLite)

Order history data is stored in SQLite, which keeps track of past purchases, including product details, quantity, total amount, payment status, and delivery tracking.

url: `sqlite://order_history.db`

6. Real-Time User Input (Camera & Image Uploads)

Users can upload images through their device's camera or select existing photos. The AI system processes these images using OpenCV and deep learning techniques to analyze facial features and recommend suitable clothing.

url: `local_storage/user_images/` or ``https://camera.upload/api``

These data sources work together to create an intelligent and efficient e-commerce platform. User and product databases store essential information, AI-driven datasets enhance personalized recommendations, PayTm handles secure transactions, and real-time image uploads enable interactive fashion advice. By integrating these technologies, the platform ensures a smooth, secure, and engaging shopping experience.

References

1. Adelakun, J., & Adewale, O. (2023). The challenges of online shopping in Nigeria: A focus on the fashion industry. *Journal of Nigerian Retail Studies*, 15(2), 45-59.
2. Akram, S. V., Malik, P. K., Singh, R., Gehlot, A., Juyal, A., Ghafoor, K. Z., & Shrestha, S. (2022). Implementation of digitalized technologies for Fashion Industry 4.0: Opportunities and challenges. *Scientific Programming*. <https://doi.org/10.1155/2022/7523246>
3. Bolesnikov, M., Popović Stijačić, M., Keswani, A. B., & Brkljač, N. (2022). Perception of innovative usage of AI in optimizing customer purchasing experience within the sustainable fashion industry. *Sustainability*, 14(10082).
4. Chatbri, H., Jemmali, S., & Hannachi, S. (2019). Empowering the Tunisian textile industry with artificial intelligence. *Discussion article*, July 2019.
5. Cohen, E. (2016). Ethnic tourism in mainland Southeast Asia: The state of the art. *Tourism Recreation Research*, 41, 232-245. <https://doi.org/10.1080/02508281.2016.1188485>
6. Csanák, E. (2020). AI for fashion. *13th International Scientific-Professional Symposium Textile Science and Economy*, Zagreb, Croatia.
7. Group4.1 Report. (2023). AI transforms the fashion industry for circular commerce. *Sponsored by SAP, Munich University of Applied Sciences*.
8. Guo, Z., Zhu, Z., Li, Y., Cao, S., Chen, H., & Wang, G. (2023). AI-assisted fashion design: A review. *IEEE Access*.
9. Kathuria, A., & Chaudhary, J. (2024). Emerging paradigms and challenges in multidisciplinary education and research. *Selfypage Developers Pvt Ltd.*, Karnataka, India.
10. Lee, J., & Suh, S. (2024). AI technology integrated education model for empowering fashion design ideation. *Sustainability*, 16(7262), 1–27. <https://doi.org/10.3390/su16177262>
11. Mishra, K., & Rangnekar, S. (2024). HUM(AI)N fashion - A collaborative design approach to revolutionize the fashion industry. *IFFTI Annual Proceedings*, 3, 264-271.
12. O'Connell, M. J. (2024). The role of Open AI in fashion design: Transforming creativity through innovation. *School of Fashion, Seneca Polytechnic*.
13. Rathore, B. (2016). AI and the future of ethical fashion marketing: A comprehensive analysis of sustainable methods and consumer engagement. *Eduzone Journal*.

14. Rathore, B. (2016). AI and the future of ethical fashion marketing: A comprehensive analysis of sustainable methods and consumer engagement. *EDUZONE: International Peer Reviewed/Refereed Multidisciplinary Journal*, 5(2), 14-19. <https://doi.org/10.56614/eiprmj.v5i2y16.323>
15. Rathore, B. (2017). Beyond trends: Shaping the future of fashion marketing with AI, sustainability, and machine learning. *EDUZONE: International Peer Reviewed/Refereed Multidisciplinary Journal*, 6(2), 16-20. <https://doi.org/10.56614/eiprmj.v6i2y17.340>
16. Rathore, B. (2017). Cloaked in code: AI & machine learning advancements in fashion marketing. *EDUZONE: International Peer Reviewed/Refereed Multidisciplinary Journal*, 6(2), 25-28. <https://doi.org/10.56614/eiprmj.v6i2y17.342>
17. Rathore, B. (2017). Sustainable fashion marketing AI-powered solutions for effective promotions. *International Journal of New Media Studies: International Peer Reviewed Scholarly Indexed Journal*, 4(2), 70-80.
18. Rathore, B. (2019). Fashion sustainability in the AI era: Opportunities and challenges in marketing. *EDUZONE: International Peer Reviewed/Refereed Multidisciplinary Journal (EIPRMJ)*, 8(2), 1-9.
19. Shirkhani, S., Mokayed, H., Saini, R., & Chai, H. Y. (2023). Study of AI-driven fashion recommender systems. *SN Computer Science*, 4(514). <https://doi.org/10.1007/s42979-023-01932-9>
20. Yeo, S. F., Tan, C. L., Kumar, A., Tan, K. H., & Wong, J. K. (2022). Investigating the impact of AI-powered technologies on Instagrammers' purchase decisions in the digitalization era: A study of the fashion and apparel industry. *Elsevier*