



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

CSI3007

ADVANCED PYTHON PROGRAMMING

LAB ASSESMENT - 22

WEB APP DEVELOPMENT IN PYTHON USING FLASK

SUBMITTED BY:

- NAME: G REENA SRI
- REG NO: 22MID0009

SUBMITTED TO:

- SHARMILA BANU K

SKIN CARE PREDICTION AND RECOMMENDATION SYSTEM USING SBERT AND FM MODELS.

MAJOR OBJECTIVE:

The main objective of this project is to develop an intelligent Skin Care Prediction and Recommendation System that leverages SBERT (Sentence-Bidirectional Encoder Representations from Transformers) and Factorization Models to provide personalized cosmetic product suggestions based on individual skin profiles. The system aims to analyse the user's input data — including skin type, concerns, and preferences — and compare it with an existing cosmetic dataset using cosine similarity to measure the closeness between user features and product attributes. By identifying the most similar products in terms of composition and effectiveness, the model recommends the most suitable skincare products for the user. Additionally, the incorporation of factorization models enhances recommendation accuracy by capturing latent relationships and hidden factors influencing user-product interactions. Overall, the objective is to create a data-driven, adaptive, and user-centric skincare recommendation system that improves the decision-making process for users seeking personalized cosmetic solutions.

KEY TAKEAWAYS:

- The project focuses on developing a Skin Care Prediction and Recommendation System.
- It utilizes SBERT (Sentence-Bidirectional Encoder Representations from Transformers) and Factorization Models.
- Cosine similarity is used to determine the closeness between user input and existing cosmetic product data.
- The system recommends the most suitable skincare products based on user-specific attributes like skin type and concerns.
- Factorization models are employed to uncover hidden patterns and improve recommendation accuracy.
- The goal is to build a personalized, data-driven, and adaptive system for skincare product recommendations.
- The system aims to enhance user experience and assist in informed decision-making for selecting cosmetic products.

CODE:

main.py

```
import os
from fastapi import FastAPI
from fastapi.staticfiles import StaticFiles
from fastapi.responses import FileResponse
from fastapi.middleware.cors import CORSMiddleware
from backend.routes import router as api_router

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

app.include_router(api_router, prefix="/api")

BASE_DIR = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))
FRONTEND_DIR = os.path.join(BASE_DIR, "frontend")
STATIC_DIR = os.path.join(FRONTEND_DIR, "static")
TEMPLATES_DIR = os.path.join(FRONTEND_DIR, "templates")

app.mount("/static", StaticFiles(directory=STATIC_DIR), name="static")

@app.get("/")
def index():
    return FileResponse(os.path.join(TEMPLATES_DIR, "index.html"))
```

routes.py

```
from fastapi import APIRouter
from backend.services.recommender import recommend_products
from backend.schemas.request_models import RecommendRequest

router = APIRouter()

@router.post("/recommend")
def get_recommendations(req: RecommendRequest):
    results = recommend_products(req)
    return {"recommendations": results}
```

test_verify.py

```
from backend.services.recommender import recommend_products
from backend.schemas.request_models import RecommendRequest

rec = recommend_products()
req = RecommendRequest(
    skin_type="dry",
    concerns=["hydration", "barrier repair"],
    product_type="moisturizer",
    budget_min=300,
    budget_max=1300,
    avoid_ingredients=["fragrance"],
    prefer_ingredients=["ceramide", "hyaluronic acid"],
    top_k=5,
    user_id=None
)
items = rec.recommend(req)
print("Top results (name, price, score):")
for it in items:
    print(it["name"], it["price"], round(it["score"],4))
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Skincare Recommender</title>
  <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
  <div class="container">
    <h1>Find Your Skincare Match</h1>
    <p class="subtitle">No ingredient knowledge needed – answer a few basics
and we'll recommend the best fits.</p>

    <form id="rec-form">
      <div class="grid">
        <!-- 1. Product Goal / Type -->
        <label>
          <span>What type of product are you looking for?</span>
          <select name="product_type">
            <option value="">Any</option>
            <option>Cleanser</option>
            <option>Moisturizer</option>
```

```
        <option>Serum</option>
        <option>Sunscreen</option>
        <option>Toner</option>
        <option>Face Mask</option>
        <option>Eye Cream</option>
        <option>Other</option>
    </select>
</label>

<!-- 2. Skin Type -->
<label>
    <span>What is your skin type?</span>
    <select name="skin_type">
        <option value="">Not sure</option>
        <option>Dry</option>
        <option>Oily</option>
        <option>Normal</option>
        <option>Combination</option>
        <option>Sensitive</option>
    </select>
</label>

<!-- Concerns -->
<label>
    <span>Any skin concerns?</span>
    <select name="concerns" multiple size="5">
        <option>Acne / Breakouts</option>
        <option>Dry patches</option>
        <option>Oiliness / Shine</option>
        <option>Redness / Sensitivity</option>
        <option>Dullness / Uneven tone</option>
    </select>
    <small>Tip: Hold Ctrl/Cmd to select multiple.</small>
</label>

<!-- 3. Budget -->
<label>
    <span>What is your budget?</span>
    <select name="budget">
        <option value="">Any</option>
        <option>Less than $10</option>
        <option>$10 - $30</option>
        <option>$30 - $50</option>
        <option>$50+</option>
    </select>
</label>

<!-- 4. Brand Preference -->
```

```
<label>
  <span>Preferred brand?</span>
  <input name="brand" placeholder="Type brand or leave blank for 'No
preference'"/>
</label>

<!-- 5. Ingredient Preferences / Allergies -->
<label>
  <span>Ingredient preferences / allergies</span>
  <select name="ingredient_prefs" multiple size="4">
    <option>Fragrance-free</option>
    <option>Paraben-free</option>
    <option>Vegan / Natural</option>
    <option>No preference</option>
  </select>
</label>

<!-- 6. Specific Product (optional) -->
<label>
  <span>Specific product name (optional)</span>
  <input name="product_name" placeholder="e.g., 'Niacinamide Serum'"/>
</label>

<!-- 7. Priority / Focus -->
<label>
  <span>What matters most?</span>
  <select name="priority">
    <option value="">No specific priority</option>
    <option>Hydration / Moisturization</option>
    <option>Anti-aging / Wrinkle reduction</option>
    <option>Oil control / Matte finish</option>
    <option>Brightening / Even skin tone</option>
    <option>Sensitive skin friendly</option>
  </select>
</label>

<!-- Top K -->
<label>
  <span>How many results?</span>
  <select name="top_k">
    <option>5</option>
    <option selected>10</option>
  </select>
</label>
</div>

<button type="submit">Get Recommendations</button>
</form>
```

```
<div id="results" class="cards"></div>
</div>

<script src="/static/app.js"></script>
</body>
</html>
```

styles.css

```
body {
  background-image: url('images.jpg');
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  background-attachment: fixed;
  color: #e0e0ff;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

header {
  background: rgba(30, 30, 47, 0.8);
  backdrop-filter: blur(10px);
  border-radius: 20px;
  padding: 2rem;
  margin-bottom: 2rem;
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.7);
}

h1 {
  background: linear-gradient(45deg, #a855f7, #7c3aed);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  background-clip: text;
  text-shadow: 0 2px 4px rgba(0, 0, 0, 0.5);
}

.bg-white.rounded-lg.shadow-lg {
  background: rgba(30, 30, 47, 0.85);
  backdrop-filter: blur(10px);
  border: 1px solid rgba(168, 85, 247, 0.3);
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.7);
  color: #dcdcff;
}
```

```
select, input {
  background-color: #2a2a3d;
  color: #e0e0ff;
  border-radius: 10px;
  border: 1px solid #5a4a9a;
  transition: all 0.3s ease;
}

select:focus, input:focus {
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(168, 85, 247, 0.6);
  outline: none;
}

button[type="submit"] {
  background: linear-gradient(45deg, #a855f7, #7c3aed);
  border: none;
  border-radius: 25px;
  font-weight: 600;
  letter-spacing: 0.5px;
  transition: all 0.3s ease;
  color: white;
}

button[type="submit"]:hover {
  background: linear-gradient(45deg, #9333ea, #6d28d9);
  transform: translateY(-2px);
  box-shadow: 0 6px 20px rgba(168, 85, 247, 0.7);
}

.bg-gradient-to-r.from-purple-50.to-pink-50 {
  background: linear-gradient(135deg, #2a2a3d 0%, #3a2a3d 100%);
  border: 1px solid rgba(168, 85, 247, 0.5);
  border-radius: 15px;
  transition: all 0.3s ease;
  color: #dcdcff;
}

.bg-gradient-to-r.from-purple-50.to-pink-50:hover {
  transform: translateY(-5px);
  box-shadow: 0 10px 25px rgba(168, 85, 247, 0.8);
}

h3 {
  color: #cbb4f9;
  font-weight: 700;
}
```



```

@media (max-width: 768px) {
  .container {
    padding: 1rem;
  }

  header {
    padding: 1.5rem;
  }

  h1 {
    font-size: 2rem;
  }
}

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

.container {
  animation: fadeIn 0.8s ease-out;
}

::-webkit-scrollbar {
  width: 8px;
}

::-webkit-scrollbar-track {
  background: #1e1e2f;
}

::-webkit-scrollbar-thumb {
  background: linear-gradient(45deg, #a855f7, #7c3aed);
  border-radius: 4px;
}

::-webkit-scrollbar-thumb:hover {
  background: linear-gradient(45deg, #9333ea, #6d28d9);
}

```

app.py

```

from flask import Flask, render_template, request
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

```

```

from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer

app = Flask(__name__)
df = pd.read_csv("C:/Users/gokul/OneDrive/Desktop/cosmetics.csv")

skin_cols = ["Combination", "Dry", "Normal", "Oily", "Sensitive"]
df = df[df[skin_cols].sum(axis=1) > 0].reset_index(drop=True)

def build_text(row):
    parts = []
    if pd.notna(row["Label"]): parts.append(str(row["Label"]))
    if pd.notna(row["Brand"]): parts.append(str(row["Brand"]))
    if pd.notna(row["Name"]): parts.append(str(row["Name"]))
    if pd.notna(row["Ingredients"]): parts.append(str(row["Ingredients"]))
    return " || ".join(parts)

df["text"] = df.apply(build_text, axis=1)
texts = df["text"].tolist()
model = SentenceTransformer("all-MiniLM-L6-v2")
embeddings = model.encode(
    texts, batch_size=64, convert_to_numpy=True, normalize_embeddings=True
)
extra_cols = ["Price", "Rank"]
numX = df[extra_cols].fillna(df[extra_cols].median()).to_numpy(dtype=float)
scaler = StandardScaler().fit(numX)
numX = scaler.transform(numX)
X = np.hstack([embeddings, numX])
@app.route("/", methods=["GET", "POST"])
def index():
    recommendations = []
    if request.method == "POST":
        skin_type = request.form["skin_type"]
        skin_concern = request.form["skin_concern"]
        product_type = request.form["product_type"]
        avoid_ingredients = request.form["avoid_ingredients"]
        budget_range = request.form["budget_range"]
        product_format = request.form["product_format"]
        user_query = (
            f"Skin type: {skin_type}. "
            f"Concern: {skin_concern}. "
            f"Product: {product_type}. "
            f"Avoid ingredients: {avoid_ingredients}. "
            f"Budget: {budget_range}. "
            f"Format: {product_format}."
        )
        query_emb = model.encode(
            [user_query], convert_to_numpy=True, normalize_embeddings=True

```

```

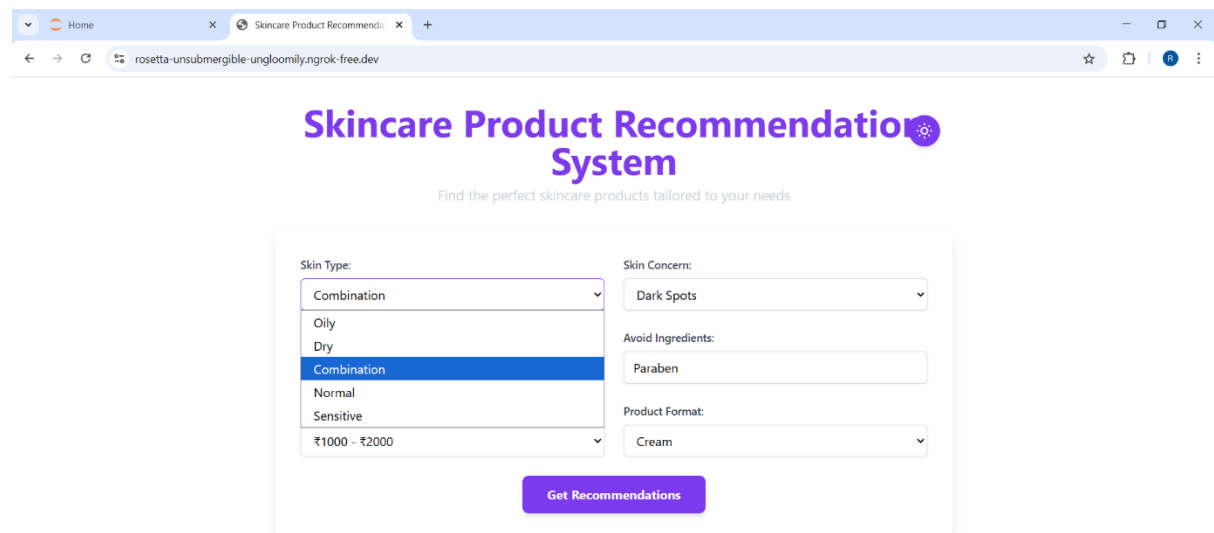
    )
    sims = cosine_similarity(query_emb, embeddings)[0]
    top_idx = np.argsort(-sims)[:5]
    for i in top_idx:
        recommendations.append({
            "Product": df.iloc[i]["Name"],
            "Brand": df.iloc[i]["Brand"],
            "Similarity": round(sims[i], 3)
        })

    return render_template("index.html", recommendations=recommendations)

if __name__ == "__main__":
    app.run(debug=True)

```

POSSIBLE OPTIONS / CATEGORIES:



Skincare Product Recommendation System

Find the perfect skincare products tailored to your needs

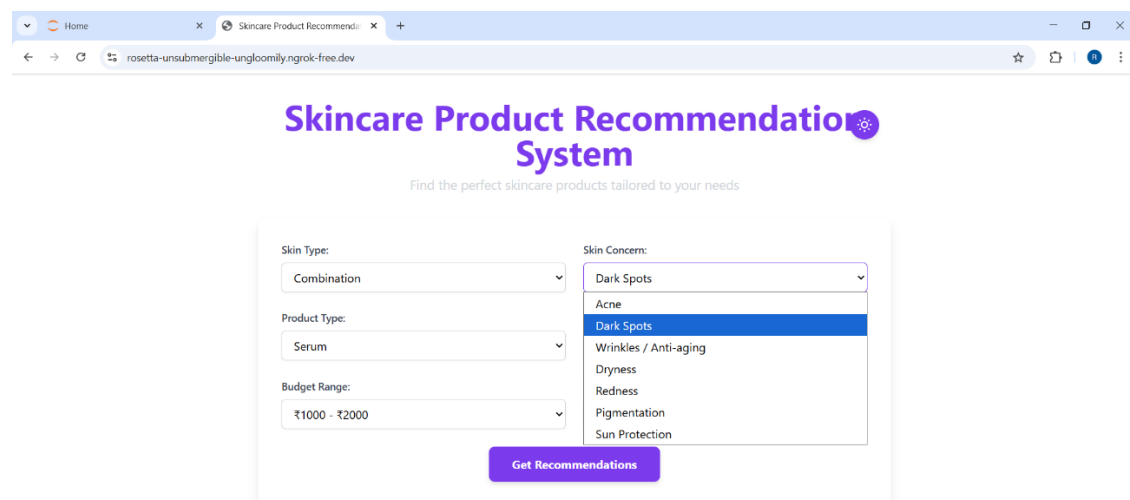
Skin Type:
 Combination
 Oily
 Dry
 Combination
 Normal
 Sensitive
 ₹1000 - ₹2000

Skin Concern:
 Dark Spots

Avoid Ingredients:
 Paraben

Product Format:
 Cream

Get Recommendations



Skincare Product Recommendation System

Find the perfect skincare products tailored to your needs

Skin Type:
 Combination

Product Type:
 Serum

Budget Range:
 ₹1000 - ₹2000

Skin Concern:
 Dark Spots
 Acne
 Dark Spots
 Wrinkles / Anti-aging
 Dryness
 Redness
 Pigmentation
 Sun Protection

Get Recommendations

Skincare Product Recommendation System

Find the perfect skincare products tailored to your needs

Skin Type:

Skin Concern:

Product Type:

Avoid Ingredients:

Product Format:

Recommendations

Skincare Product Recommendation System

Find the perfect skincare products tailored to your needs

Skin Type:

Skin Concern:

Product Type:

Avoid Ingredients:

Budget Range:

Product Format:

Recommendations

Skincare Product Recommendation System

Find the perfect skincare products tailored to your needs

Skin Type:

Skin Concern:

Product Type:

Avoid Ingredients:

Budget Range:

Product Format:

Get Recommendation

OUTPUT SCREENSHOT

USER INPUT 1:

127.0.0.1:5000

Gmail Bard Exam Dashboard lab - JupyterLab shor.ipynb - Colabo... Perplexity Claude DeepSeek - Into the... Google Gemini Alchemy University SmartED Innovation... All Bookmarks

Skincare Product Recommendation System

Find the perfect skincare products tailored to your needs

Skin Type:

Skin Concern:

Product Type:

Avoid Ingredients:

Budget Range:

Product Format:

Get Recommendations

RECOMMENDATION:

127.0.0.1:5000

Gmail Bard Exam Dashboard lab - JupyterLab shor.ipynb - Colabo... Perplexity Claude DeepSeek - Into the... Google Gemini Alchemy University SmartED Innovation... All Bookmarks

Budget Range:

Product Format:

Get Recommendations

Top 5 Recommended Products

BB Tinted Treatment 12-Hour Primer Broad Spectrum SPF 30 Sunscreen Brand: TARTE Similarity: 0.623	Lingerie de Peau BB Cream Brand: GUERLAIN Similarity: 0.565	Ultra Facial Moisturizer SPF 30 Brand: KIEHL'S SINCE 1851 Similarity: 0.558
Your Skin But Better™ CC+™ Cream with SPF 50+ Brand: IT COSMETICS Similarity: 0.542	BB Cream SPF 35 Brand: BOBBI BROWN Similarity: 0.539	

USER INPUT 2:

127.0.0.1:5000

GmailBardExam Dashboardlab - JupyterLabshor.ipynb - Colabo...PerplexityClaudeDeepSeek - Into the...Google GeminiAlchemy UniversitySmartED Innovation...All Bookmarks

Skincare Product Recommendation System

Find the perfect skincare products tailored to your needs

Skin Type:

Sensitive

Skin Concern:

Dryness

Product Type:

Serum

Avoid Ingredients:

alcohol

Budget Range:

₹500 - ₹1000

Product Format:

Gel

Get Recommendations

RECOMMENDATION:

127.0.0.1:5000

GmailBardExam Dashboardlab - JupyterLabshor.ipynb - Colabo...PerplexityClaudeDeepSeek - Into the...Google GeminiAlchemy UniversitySmartED Innovation...All Bookmarks

Budget Range:

₹500 - ₹1000

Product Format:

Cream

Get Recommendations

Top 5 Recommended Products

Perfect Canvas Skin Finishing Serum

Brand: REN CLEAN SKINCARE

Similarity: 0.536

BB Tinted Treatment 12-Hour Primer Broad Spectrum SPF 30 Sunscreen

Brand: TARTE

Similarity: 0.52

Ready Steady Glow Daily AHA Tonic

Brand: REN CLEAN SKINCARE

Similarity: 0.517

Benefiance WrinkleResist24 Night Cream

Brand: SHISEIDO

Similarity: 0.507

Lingerie de Peau BB Cream

Brand: GUERLAIN

Similarity: 0.505

USER INPUT 3:

127.0.0.1:5000

GmailBardExam Dashboardlab - JupyterLabshor.ipynb - Colabo...PerplexityClaudeDeepSeek - Into the...Google GeminiAlchemy UniversitySmartED Innovation...All Bookmarks

Skincare Product Recommendation System

Find the perfect skincare products tailored to your needs

Skin Type:

Combination

Skin Concern:

Wrinkles / Anti-aging

Product Type:

Toner

Avoid Ingredients:

alcohol

Budget Range:

₹500 - ₹1000

Product Format:

Foam

Get Recommendations

RECOMMENDATION:

127.0.0.1:5000

GmailBardExam Dashboardlab - JupyterLabshor.ipynb - Colabo...PerplexityClaudeDeepSeek - Into the...Google GeminiAlchemy UniversitySmartED Innovation...All Bookmarks

Moisturizer

e.g., Alcohol, Paraben

Budget Range:

₹500 - ₹1000

Product Format:

Cream

Get Recommendations

Top 5 Recommended Products

Essential Power Skin Toner for Combination to Oily Skin

Brand: LANEIGE

Similarity: 0.567

BB Tinted Treatment 12-Hour Primer Broad Spectrum SPF 30 Sunscreen

Brand: TARTE

Similarity: 0.549

Lingerie de Peau BB Cream

Brand: GUERLAIN

Similarity: 0.536

Ultra Facial Moisturizer SPF 30

Brand: KIEHL'S SINCE 1851

Similarity: 0.502

Ultra Facial Toner

Brand: KIEHL'S SINCE 1851

Similarity: 0.501

The screenshot shows a VS Code editor with a project named 'SKINCAREWEB'. The Explorer sidebar on the left shows the file structure: templates, images.jpg, index.html, style.css, and app.py. The app.py file is open in the editor, showing Python code for a Flask web application. The code imports Flask, pandas, numpy, sklearn, and sentence transformers. It reads a CSV file 'cosmetics.csv' and defines a route 'build_text' that processes data from the CSV. The terminal at the bottom shows the command 'python app.py' being executed, and the output displays a series of HTTP requests and responses, indicating the application is running successfully.

```
1 from flask import Flask, render_template, request
2 import pandas as pd
3 import numpy as np
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.metrics.pairwise import cosine_similarity
6 from sentence_transformers import SentenceTransformer
7
8 app = Flask(__name__)
9 df = pd.read_csv("C:/Users/gokul/OneDrive/Desktop/cosmetics.csv")
10
11 skin_cols = ["Combination", "Dry", "Normal", "Oily", "Sensitive"]
12 df = df[df[skin_cols].sum(axis=1) > 0].reset_index(drop=True)
13
14 def build_text(row):
15     parts = []
16     if pd.notna(row["Label"]): parts.append(str(row["Label"]))
17     if pd.notna(row["Brand"]): parts.append(str(row["Brand"]))
18     if pd.notna(row["Name"]): parts.append(str(row["Name"]))
19     if pd.notna(row["Ingredients"]): parts.append(str(row["Ingredients"]))
20     return " | ".join(parts)
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS F:\skincareweb> python app.py

* Debugger PIN: 432-316-652

127.0.0.1 - - [06/Nov/2025 23:08:55] "GET / HTTP/1.1" 200 -

127.0.0.1 - - [06/Nov/2025 23:08:55] "GET /style.css HTTP/1.1" 404 -

127.0.0.1 - - [06/Nov/2025 23:09:47] "POST / HTTP/1.1" 200 -

127.0.0.1 - - [06/Nov/2025 23:09:47] "GET /style.css HTTP/1.1" 404 -

127.0.0.1 - - [06/Nov/2025 23:10:30] "POST / HTTP/1.1" 200 -

127.0.0.1 - - [06/Nov/2025 23:10:30] "GET /style.css HTTP/1.1" 404 -

127.0.0.1 - - [06/Nov/2025 23:11:09] "POST / HTTP/1.1" 200 -

127.0.0.1 - - [06/Nov/2025 23:11:09] "GET /style.css HTTP/1.1" 404 -

127.0.0.1 - - [06/Nov/2025 23:12:06] "POST / HTTP/1.1" 200 -

127.0.0.1 - - [06/Nov/2025 23:12:06] "GET /style.css HTTP/1.1" 404 -

DEPLOYMENT IN NGROK SERVER

The screenshot shows the same VS Code editor with the 'SKINCAREWEB' project. The app.py file is still open. The terminal at the bottom shows the command 'python app.py' being executed, and the output displays a series of HTTP requests and responses, indicating the application is running successfully. The terminal also shows the Ngrok session status, including the account name 'Gokulesh (Plan: Free)', the version '3.24.0-msix', the region 'India (in)', the latency '25ms', the web interface 'http://127.0.0.1:4040', and the forwarding URL 'https://rosetta-unsubmergible-ungloomily.ngrok-free.dev -> http://localhost:5000'.

```
1 from flask import Flask, render_template, request
2 import pandas as pd
3 import numpy as np
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.metrics.pairwise import cosine_similarity
6 from sentence_transformers import SentenceTransformer
7
8 app = Flask(__name__)
9 df = pd.read_csv("C:/Users/gokul/OneDrive/Desktop/cosmetics.csv")
10
11 skin_cols = ["Combination", "Dry", "Normal", "Oily", "Sensitive"]
12 df = df[df[skin_cols].sum(axis=1) > 0].reset_index(drop=True)
13
14 def build_text(row):
15     parts = []
16     if pd.notna(row["Label"]): parts.append(str(row["Label"]))
17     if pd.notna(row["Brand"]): parts.append(str(row["Brand"]))
18     if pd.notna(row["Name"]): parts.append(str(row["Name"]))
19     if pd.notna(row["Ingredients"]): parts.append(str(row["Ingredients"]))
20     return " | ".join(parts)
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Decouple policy and sensitive data with vaults: <https://ngrok.com/r/secrets>

Session Status online

Account Gokulesh (Plan: Free)

Update update available (version 3.32.0, Ctrl-U to update)

Version 3.24.0-msix

Region India (in)

Latency 25ms

Web Interface http://127.0.0.1:4040

Forwarding https://rosetta-unsubmergible-ungloomily.ngrok-free.dev -> http://localhost:5000

The screenshot shows the same VS Code editor with the 'SKINCAREWEB' project. The app.py file is still open. The terminal at the bottom shows the command 'python app.py' being executed, and the output displays a series of HTTP requests and responses, indicating the application is running successfully. The terminal also shows the Ngrok session status, including the account name 'Gokulesh (Plan: Free)', the version '3.24.0-msix', the region 'India (in)', the latency '31ms', the web interface 'http://127.0.0.1:4040', and the forwarding URL 'https://rosetta-unsubmergible-ungloomily.ngrok-free.dev -> http://localhost:5000'.

```
1 from flask import Flask, render_template, request
2 import pandas as pd
3 import numpy as np
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.metrics.pairwise import cosine_similarity
6 from sentence_transformers import SentenceTransformer
7
8 app = Flask(__name__)
9 df = pd.read_csv("C:/Users/gokul/OneDrive/Desktop/cosmetics.csv")
10
11 skin_cols = ["Combination", "Dry", "Normal", "Oily", "Sensitive"]
12 df = df[df[skin_cols].sum(axis=1) > 0].reset_index(drop=True)
13
14 def build_text(row):
15     parts = []
16     if pd.notna(row["Label"]): parts.append(str(row["Label"]))
17     if pd.notna(row["Brand"]): parts.append(str(row["Brand"]))
18     if pd.notna(row["Name"]): parts.append(str(row["Name"]))
19     if pd.notna(row["Ingredients"]): parts.append(str(row["Ingredients"]))
20     return " | ".join(parts)
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Decouple policy and sensitive data with vaults: <https://ngrok.com/r/secrets>

Session Status online

Account Gokulesh (Plan: Free)

Update update available (version 3.32.0, Ctrl-U to update)

Version 3.24.0-msix

Region India (in)

Latency 31ms

Web Interface http://127.0.0.1:4040

Forwarding https://rosetta-unsubmergible-ungloomily.ngrok-free.dev -> http://localhost:5000

AFTER DEPLOYMENT:

The screenshot shows a web browser with two tabs: 'Home' and 'Skincare Product Recommendation System'. The address bar shows the URL 'rosetta-unsubmergible-ungloomily.ngrok-free.dev'. The page title is 'Skincare Product Recommendation System' with a gear icon. Below the title is the subtitle 'Find the perfect skincare products tailored to your needs'. The main form has six input fields: 'Skin Type' (Oily), 'Skin Concern' (Acne), 'Product Type' (Sunscreen), 'Avoid Ingredients' (e.g., Alcohol, Paraben), 'Budget Range' (₹500 - ₹1000), and 'Product Format' (Cream). A purple 'Get Recommendations' button is at the bottom of the form. Below the form is a section titled 'Top 5 Recommended Products' with three placeholder cards.

The screenshot shows the same web browser with the 'Skincare Product Recommendation System' tab. The 'Budget Range' is set to '₹500 - ₹1000' and 'Product Format' is set to 'Cream'. The 'Get Recommendations' button is visible. Below the form is a section titled 'Top 5 Recommended Products' with five product cards:

- Lingerie de Peau BB Cream**
Brand: GUERLAIN
Similarity: 0.591
- BB Tinted Treatment 12-Hour Primer Broad Spectrum SPF 30 Sunscreen**
Brand: TARTÉ
Similarity: 0.584
- One Essential Eye Serum**
Brand: DIOR
Similarity: 0.565
- One Essential Skin Boosting Super Serum**
Brand: DIOR
Similarity: 0.562
- BB Cream SPF 35**
Brand: BOBBI BROWN
Similarity: 0.56

WEBSITE LINK:

<https://rosetta-unsubmergible-ungloomily.ngrok-free.dev/>