StyleMate – ML-Powered Color & Style Recommender

Link: [GitHub Link]

1. Business Problem

People often struggle to choose suitable fashion styles based on their natural features. StyleMate addresses this by offering AI-driven color and style recommendations based on skin, hair, and eye color.

2. Tech Stack Used

- Frontend: HTML, CSS, JavaScript
- Backend: Flask (Python)
- ML Tools: Scikit-learn, media pipe ,Pandas, Numpy
- Model: RandomForestClassifier

3. Key Features / Code Highlights

- Extracted LAB values from uploaded photos
- Trained Random Forest classifier (93.75% accuracy)
- Flask backend for prediction and routing
- Recommendation engine for colors, clothes, makeup and grooming tips

4. Code Snippets

1. **Code that extracts LAB values from uploaded images.**

```python
import cv2
import numpy as np
import mediapipe as mp
import os

Tabnine | Edit | Test | Explain | Document
def get_average_color(image, center, size=5):
    x, y = center
    h, w, _ = image.shape
    half = size // 2
    x_start = max(x - half, 0)
    x_end = min(x + half + 1, w)
    y_start = max(y - half, 0)
    y_end = min(y + half + 1, h)
    region = image[y_start:y_end, x_start:x_end]
    return np.mean(region.reshape(-1, 3), axis=0)
```

2. **ML model training using Random Forest**

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import joblib
import numpy as np

# Load dataset
df = pd.read_csv('filled_season_color_dataset.csv')

# Use correct column names (case-sensitive)
X = df[['Hair_L', 'Hair_A', 'Hair_B', 'Skin_L', 'Skin_A', 'Skin_B', 'Eye_L', 'Eye_A', 'Eye_B']]
y = df['Season']

# Train/Test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

3. **Flask route handling the prediction request**

```python
@app.route('/')
def index():
    return render_template("index.html")

Tabnine | Edit | Test | Explain | Document
def rgb_to_lab(color):
    rgb_color = np.uint8([[color]])
    lab_color = cv2.cvtColor(rgb_color, cv2.COLOR_RGB2LAB)[0][0]
    return lab_color
Tabnine | Edit | Test | Explain | Document
def lab_to_rgb(lab_color):
    # Convert standard LAB to OpenCV scale
    L = lab_color[0] * 255 / 100
    a = lab_color[1] + 128
    b = lab_color[2] + 128

    lab_scaled = np.uint8([[[L, a, b]]])
    rgb = cv2.cvtColor(lab_scaled, cv2.COLOR_LAB2RGB)[0][0]
    return [int(x) for x in rgb]
```

5. Project Explanation

Style Mate is a web application that analyzes a user's photo to extract skin, hair, and eye color using OpenCV and LAB color space.
It predicts the user's seasonal color palette using a Random Forest model.
Based on the prediction, it recommends suitable outfit colors, fabrics, and makeup tones.
The frontend is clean and responsive, while Flask handles backend logic and model inference.
It promotes inclusive styling with a personalized, AI-powered approach.

Wanderla – Real Estate Web Application

Link: GitHub Link

1. Business Problem

Real estate users face challenges in finding, listing, and managing properties with poor user interfaces and lack of real-time location mapping. Wanderla solves this with a full-stack listing and exploration platform.

2. Tech Stack Used

- Frontend: HTML, CSS, Bootstrap, JavaScript
- Backend: Node.js, Express.js
- Database: MongoDB
- Cloud Services: Cloudinary, Mapbox

3. Key Features / Code Highlights

- Property listing creation and image upload via Cloudinary
- Map integration using Mapbox
- Secure user authentication
- Filtering, reviews, and property detail pages

4. Code Snippets

1. **Express route for posting properties**

```
const express = require("express");
const router = express.Router();
const wrapAsync = require("../utils/wrapAsync.js");
const Listing = require("../models/listing.js");
const {isLoggedIn, isOwner, validateListing} = require("../middleware.js");
const listingController = require("../controllers/listing.js");
const multer  = require('multer');  227.8k (gzipped: 43.6k)
const {storage} = require("../cloudConfig.js");
const upload = multer({storage});

  //Create Route
router.route("/")
.get( wrapAsync(listingController.index))
.post(
  isLoggedIn,
  upload.single('listing[image]'),
  validateListing ,       You, 9 months ago • Add .gitattributes to enforce consistent line e…
  wrapAsync(listingController.createListing),
);
```

2. **Cloudinary image upload logic**

```
const cloudinary = require('cloudinary').v2;  Calculating...
const { CloudinaryStorage } = require('multer-storage-cloudinary');
      You, 9 months ago • Add .gitattributes to enforce consistent l
cloudinary.config({
    cloud_name :process.env.CLOUD_NAME,
    api_key : process.env.CLOUD_API_KEY,
    api_secret : process.env.CLOUD_API_SECRET
});

const storage = new CloudinaryStorage({
    cloudinary: cloudinary,
    params: {
      folder: 'wanderlust_DEV',
      allowedFormats: ["png" , "jpg" , "jpeg" , "pdf"],
    },
  });

  module.exports ={
    cloudinary,
    storage,
  };
```

3. **Map box location embedding in listings**

```
module.exports.createListing = async(req,res , next)=>{
   let response = await geocodingClient
   .forwardGeocode({
     query: req.body.listing.location ,
     limit: 1,
   })
     .send();       You, 9 months ago • Add .gitattributes to enforc
   let url = req.file.path;
   let filename = req.file.filename;
 const newlisting = new Listing(req.body.listing);
   newlisting.owner = req.user._id;
   newlisting.image = {url,filename};
   newlisting.geometry = response.body.features[0].geometry;
 let savedListing =  await newlisting.save();
 console.log(savedListing);
  req.flash("success" , "New Listing Created!");
   res.redirect("/listings");
 };
```

5. Project Explanation

Wanderla is a full-stack property listing platform built using the MERN stack.
Users can browse, post, and review real estate listings with images and location tagging.
Cloudinary handles image uploads while Map box powers interactive property maps.
The system features secure authentication.

Innova – Stock Trading Platform

Link: Github Link

1. Business Problem

Beginners and casual users often find traditional trading platforms complex and slow. Innova solves this a user-friendly stock trading simulation platform and clean UX for seamless trading experiences.

2. Tech Stack Used

- Frontend: React.js
- Backend: Node.js, Express.js
- Database: MongoDB
- Auth: JWT, bcrypt

3. Key Features / Code Highlights

- JWT-based authentication system.
- MongoDB models for users and trades.
- React UI for charts and trading interface.

4. Code Snippets

1. **JWT-based protected route logic**

```
app.get('/allHoldings', async(req,res)=>{
    let allHoldings = await HoldingsModel.find({});
    res.json(allHoldings);
});

Tabnine | Edit | Test | Explain | Document
app.get('/allPositions', async(req,res)=>{
    let allPositions = await PositionsModel.find({});
    res.json(allPositions);
});

Tabnine | Edit | Test | Explain | Document
app.post('/newOrder', async(req,res)=>{
     let newOrder = new OrdersModel({
        name: req.body.name,
        qty: req.body.qty,
        price: req.body.price,
        mode: req.body.mode,
     });
```

2. **MongoDB schema for users and trades**

```
const {Schema} = require("mongoose");

const HoldingsSchema = new Schema({
    name: String,
    qty: Number,
    avg: Number ,
    price: Number,
    net: String,
    day: String,

});

module.exports = {HoldingsSchema};
```

3. **React components for stock chart & trade form**

4. Project Explanation

Innova is a MERN-based stock trading simulation platform built for learning and practice.
It allows users to sign up securely, simulate buy trades, and manage a virtual portfolio.
Predefined or mock stock data is used to ensure consistency and simplicity.
The platform emphasizes clean UI, low-latency execution, and user-friendly design.
It’s ideal for students and beginners looking to explore trading concepts in a safe environment.