\*\*Project\*\*: \*Black Styles – A culturally intelligent, AI-powered fashion assistant for men\*

\*\*Stack\*\*: React (Vite) frontend, Flask backend, OpenCV + TensorFlow for AI, Firebase/MongoDB for storage

\*\*Day 1 Progress\*\*:

\* Set up full project environment (frontend + backend)

\* Built and tested frontend photo upload component

\* Scaffolded Flask backend with image route structure

\* Resolved initial rendering bug and connected Axios to API stub

\*\*Next Steps\*\*:

\* Build `/api/image/upload` route to handle file uploads

\* Extract color palette from image with OpenCV + KMeans

\* Return top 3–5 hex colors to the frontend for display

\*\*Project\*\*: Black Styles – Culturally intelligent fashion + grooming app

\*\*Progress Day 2\*\*:

- Built and tested full backend route for image uploads

- Integrated OpenCV + scikit-learn for real-time color palette extraction

- Connected frontend to backend and validated working response pipeline

\*\*Next\*\*:

- Visually display palette on frontend

- Style with Tailwind or similar

- Begin body metric detection (MediaPipe)

# 🎨 Black Styles — Day 3 Backend Development Summary

## 🛠️ Completed Features

- Flask `/api/image/upload` route

- File type validation for secure uploads

- Request logging with Flask middleware

- Center auto-cropping with OpenCV

- Graceful error handling for image processing

## 🧪 Tested and Verified

- Image files processed → ✅ palette returned

- Bad/corrupt files → ✅ handled cleanly

- Terminal logs confirm error tracking and request metadata

## 📁 Files Updated

- `routes.py` → added upload logic

- `utils.py` → added helper for cropping + palette

- `\_\_init\_\_.py` → initialized app and routes

- `run.py` → Flask bootstrapping

## 🔄 Branch

- `feat/day3-palette-ui-backend-enhancements`

---

🟢 Backend functionality complete. Ready for Day 4 frontend UI enhancements.

# Black Styles — Day 4 Frontend Development Summary

## Features Implemented

- Added `ColorSwatches` component to render extracted color palette visually.

- Integrated palette into `UploadForm`, showing top 3–5 hex colors with swatches.

- Implemented loading state to indicate upload in progress.

- Added error UI for failed uploads or missing files.

## Stack

- Frontend: React (Vite)

- Communication: Axios POST to Flask backend

- Visualization: Dynamic swatches with hex labels

## Testing

- Verified swatches appear with correct hex values after successful upload.

- Loading indicator displays during request.

- Error messages surface with meaningful backend feedback.

## Branch

- `feat/day4-palette-ui-frontend`

## Next Goals

- Clipboard copy for hex codes.

- Persistent recent palettes.

- Styling polish (Tailwind integration).