FITFINDER — Project Summary

- One-line: Flask backend + static frontend for outfit generation and try-on.
- Optional Hugging Face integration; Pillow demo fallback.

Architecture

- Flask app serving APIs under /api/* and static/ frontend.
- Image generation: HF Inference API (if token) or Pillow demo images.
- Try-on: simple Pillow compositing; uploads saved to uploads/.

Key Endpoints

- GET /api/health — status + HF configured flag.
- POST /api/generate-outfit — JSON: scene, style, gender, custom_prompt.
- POST /api/tryon — multipart: person_image + cloth_image.
- POST /api/contact — stores submissions to contacts.json.

Deployment & Runtime

- Procfile: gunicorn -w 2 -b 0.0.0.0:$PORT app:app (Railway/Heroku).
- Set HF_API_TOKEN in environment to enable real AI generation.
- Generated images saved to generated_outfits/ (consider S3 for prod).

Security & Next Steps

- Do NOT commit API tokens; use platform env vars or secrets.
- Next: persistent storage, DB for metadata, better try-on model, auth for admin.

Where to find things

- app.py, app_clean.py — backend source.
- PROJECT_REPORT.md / PROJECT_REPORT.pdf — full documentation and annotated app.py.
- Generated files: generated_outfits/, uploads/.