AI-Powered Auto Damage Estimator - Project Plan

Overview

This project aims to build an AI-powered web application that detects car damages from uploaded photos and generates a transparent repair cost estimate. The MVP will focus on consumer and small business use (mechanic shops, towing companies, rental fleets), emphasizing transparency, explainability, and affordability.

Features

Feature	Description
Upload	Users can upload multiple car photos (front, sides, rear).
Detection	YOLOv8 model detects 10–12 key parts and labels damage type (dent, scrape, crack, missing, intact).
Severity	Damage severity bucketed into minor, moderate, severe using rules.
Cost Engine	CSV-driven rules: labor hours × labor rate, parts costs (new/used ranges), paint/materials adders.
Report	Line-item estimate with totals (min/likely/max). Export as PDF.
Explainability	Transparent mapping from detection → severity → cost rules. Optional GPT summary for human-readable report.
User Input	Users set labor rate, toggle OEM/Used parts, and edit severity for accuracy.

Dataset Information

- Kaggle Datasets: Car Damage Detection, Car Parts Segmentation (free, CC licenses).
- Roboflow Universe: Vehicle damage and parts datasets in YOLOv8 format (free tier).
- Custom Labels: 200–300 images labeled by team for missing parts and severity cases.
- Data Format: YOLOv8-ready (images/ and labels/ folders with data.yaml config).

Step-by-Step Plan

Step	Action
1	Define scope (parts, damage states, severity rules). Create cost_rules.csv baseline with labor/parts costs from public sources (DFW averages, RockAuto/eBay ranges).
2	Assemble datasets from Kaggle/Roboflow. Label 200–300 custom images. Export in YOLOv8 format.
3	Train YOLOv8n (Colab/Kaggle GPU). Evaluate on hold-out set (mAP > 0.6).
4	Build FastAPI backend: /upload, /infer, /estimate, /report endpoints.
5	Develop Streamlit UI: upload → detection gallery → cost estimate table → PDF export.
6	Integrate severity scoring + cost engine (CSV-driven).
7	Add user edits (severity dropdown, OEM/Used toggle).
8	Add VIN decode (NHTSA API) and optional GPT summary.

9	Testing: run 10–20 cars through system, compare estimates to manual baselines.
10	Finalize MVP for demo by November with working web app + example reports.

Timeline to November

- September (Weeks 1–2): Scope, cost_rules.csv, dataset collection & labeling.
- October (Weeks 3–6): Train YOLOv8, build backend APIs, Streamlit UI skeleton.
- Late October (Weeks 7–8): Cost engine integration, severity rules, editing features.
- Early November (Week 9): Add VIN decode, GPT summary, polish UI.
- Mid-November (Week 10): Testing, prepare demo with sample reports.

Estimated MVP Costs

The MVP is designed to run almost entirely on free tiers: - **Model Training**: Kaggle/Google Colab GPUs (free tier, sufficient for YOLOv8 training).
- **Datasets**: Kaggle + Roboflow public datasets (free).
- **Backend Hosting**: Render or Railway free tiers (approx. 750–1000 compute hours/month).
- **Frontend**: Streamlit Cloud free tier (supports ~100 users/month).
- **Database/Storage**: Supabase free tier (up to 500 MB storage, 50k monthly requests).
- **VIN Decode**: NHTSA API (free, unlimited).
- **PDF Generation**: ReportLab (free, open-source).
- **AI Summary (optional)**: OpenAI gpt-4o-mini (~$0.0003 per request). For 200 reports/month, cost is ~$0.06–0.10 total. **Total Estimated MVP Cost**: $0/month if GPT summaries disabled, <$1/month if enabled.
Reports Supported: 100–200 per month comfortably within free tier limits.