DevLens AI – Complete Technical Documentation

1. Project Overview

DevLens AI is an AI-powered Developer Intelligence Engine designed to analyze a candidate's GitHub and LeetCode profiles and generate a structured technical evaluation. The system extracts coding metrics, computes engineered scores, visualizes performance, and uses a locally deployed Large Language Model (LLM) for contextual evaluation. The architecture follows a modular backend + modern frontend structure. Backend: FastAPI + MongoDB Atlas Frontend: React (Vite) + TailwindCSS + Recharts AI Layer: Local LLM (phi3 via Ollama)

2. System Architecture

The system consists of four primary layers: 1. Data Extraction Layer 2. Feature Engineering Layer 3. Scoring Engine Layer 4. AI Interpretation Layer 5. Visualization Layer Data flows from GitHub and LeetCode APIs → Feature extraction → Score computation → Final readiness classification → AI interpretation → Dashboard visualization.

3. Backend Components (FastAPI)

Main Backend File: main.py Endpoints: - /github/{username} - /leetcode/{username} - /engineering-score/{username} - /dsa-score/{username} - /collaboration-score/{username} - /consistency-score/{github}/{leetcode} - /devlens-evaluate/{github}/{leetcode} - /chat/{github}/{leetcode} Each endpoint follows REST principles and processes structured JSON responses.

4. Feature Engineering Layer

Engineering Features: - Language Entropy: Measures diversity of programming languages using Shannon entropy. - Project Depth: Based on average repository size. - Documentation Score: Ratio of repositories with descriptions. - Popularity Score: Based on stars and forks. - Activity Score: Based on recent pushes. DSA Features: - Hard Problem Weighting - Total Solved Volume - Acceptance Rate Normalization - Ranking Score (inverse scaled) Collaboration Features: - Followers count normalization - Fork ratio - Issue participation - Repository scale indicator Consistency Features: - Activity stability - Repo spread over time - Submission intensity - Acceptance stability

5. Scoring Engine Logic

Scores are calculated using weighted linear combinations. Engineering Score = 0.25 * Language Entropy + 0.20 * Project Depth + 0.20 * Documentation + 0.20 * Popularity + 0.15 * Activity DSA Score = Weighted Hard Score + Volume Score + Acceptance Score + Ranking Score Final Readiness Score = Weighted sum of: Engineering + DSA + Consistency + Collaboration Category Classification: < 45 → Early Stage 45-65 → Developing 65-80 → Industry Ready > 80 → Highly Competitive

6. AI Interpretation Layer

The AI layer uses a locally deployed phi3:mini model via Ollama. Process: 1. Evaluation scores are structured into JSON. 2. A prompt template is constructed. 3. The LLM generates contextual interpretation: - Strengths - Weaknesses - Hiring readiness estimate - Improvement roadmap This enables natural language reasoning on structured metrics.

7. Frontend Dashboard

Frontend is built using React (Vite) and TailwindCSS. Dashboard Features: - Score Cards - Radar Chart (multi-metric visualization) - LeetCode Line Chart - Skills Section (auto-detected languages) - Repository List - Floating AI Chat Button Charts are implemented using Recharts library.

8. Database Layer (MongoDB Atlas)

MongoDB stores evaluation history. Document Structure: { github_username, leetcode_username, engineering_score, dsa_score, consistency_score, collaboration_score, final_score, category, created_at } The _id field is excluded when returning responses to avoid serialization errors.

9. Engineering Challenges & Solutions

1. CORS Error: Solved by adding CORSMiddleware in FastAPI. 2. ObjectId Serialization Error: Resolved by excluding _id field from MongoDB responses. 3. Tailwind PostCSS Error: Installed correct plugin configuration. 4. Data Normalization: All scores scaled to 0-100 for comparability. 5. Modular Architecture: Separated services into: - github_service.py - leetcode_service.py - scoring_engine.py - llm_service.py

10. Conclusion

DevLens AI transforms raw coding activity into structured intelligence. It integrates API data extraction, feature engineering, rule-based scoring, LLM interpretation, and interactive visualization. While currently rule-based, the system can be upgraded into a full machine learning hiring predictor by incorporating labeled hiring data. This project demonstrates applied AI, backend engineering, data analytics, system architecture, and product-level thinking.