Credit Card Statement Parser — LayoutSign™

Objective:

The goal of this project is to design and implement a credit card statement PDF parser capable of extracting 5 key data points (Issuer, Card Last-4 digits, Billing Cycle, Payment Due Date, and Statement Balance) from statements of multiple credit card issuers.

System Overview:

LayoutSign™ combines deterministic regex with layout-based heuristics to robustly detect issuers and extract structured information. It uses pdfplumber for text extraction, Tesseract OCR as a fallback, and Pydantic models for structured data representation.

Key Features:

• Provider-agnostic design supporting 5 issuers (Chase, Amex, Citi, Capital One, Discover).
• Hybrid extraction using layout signatures and regex.
• Confidence scoring and evidence for every extracted field.
• CLI for batch operations and Streamlit dashboard for demos.
• Offline and privacy-safe — no external API calls.
• Easily extendable to new issuers with minimal code changes.

Architecture:

The architecture follows a modular approach with distinct components for detection, extraction, data modeling, and provider-specific logic. Each provider module encapsulates its own regex rules and layout cues, enabling plug-and-play extensibility.

Files & Structure:

• ccparser/: Core logic including extractor, models, and providers.
• samples/: Synthetic statements and generator script.
• cli.py: Command-line interface.
• app.py: Streamlit application.
• README.md: Documentation and run instructions.
• tests/: Unit tests validating parsing correctness.

Usage Instructions:

1. Create a virtual environment and install dependencies using requirements.txt.
2. Run `python cli.py samples/chase_sample.pdf --show` to view extraction.

3. Run `streamlit run app.py` to launch the web dashboard.
4. Upload PDFs and view structured outputs with confidence scores.

Uniqueness:

LayoutSign™ introduces layout fingerprinting (via simhash) to identify issuers based on structural patterns. This makes it resilient to formatting drift and suitable for production-scale document automation.

Evaluation:

The project was tested on five synthetic issuer statements and verified for accurate field extraction. All modules passed pytest-based validation tests.

Conclusion:

This project demonstrates a robust, extensible, and privacy-preserving document parsing framework. Its modular design, hybrid logic, and transparent confidence scoring make it ideal for enterprise-grade financial document automation.