Car Lease Contract Review and Negotiation AI Assistant

AI/LLM-Based System for Automotive Contract Analysis

1. Executive Summary

The Car Lease Contract Review and Negotiation AI Assistant is an intelligent system designed to empower consumers in understanding and negotiating automotive lease and loan contracts. Leveraging state-of-the-art Large Language Models (LLMs) and advanced document processing techniques, the system provides comprehensive contract analysis, fairness scoring, and actionable insights.

1.1 Key Capabilities

- Automated extraction of 50+ contract parameters including financial terms, conditions, and obligations
- AI-powered fairness analysis with transparent scoring methodology across 5 key dimensions
- Real-time red flag detection for potentially unfavorable contract terms
- VIN-based vehicle verification and history lookup
- Market price benchmarking and comparison tools
- Context-aware AI chatbot for on-the-spot contract questions
- User authentication — login, sign-up, and password recovery flows
- Interactive web interface for seamless user experience

2. Project Overview

2.1 Problem Statement

Car lease and loan contracts are complex legal documents filled with technical jargon, hidden fees, and conditions that average consumers struggle to understand. This information asymmetry between dealers/financial institutions and consumers often leads to unfavorable terms, unexpected costs, and buyer's remorse.

2.2 Solution Approach

Our solution employs artificial intelligence to democratize contract understanding. By combining optical character recognition (OCR), natural language processing (NLP), and Google's Gemini LLM, we transform opaque legal documents into clear, structured data. The Flutter frontend delivers this experience on any device, while the integrated chatbot lets users ask follow-up questions in natural language.

2.3 Target Users

- Individual consumers leasing or financing vehicles
- Small business fleet managers
- Financial advisors assisting clients with vehicle purchases
- Consumer advocacy organizations
- Legal professionals reviewing automotive contracts

3. System Architecture

The system follows a modular, layered architecture designed for scalability, maintainability, and extensibility. The architecture consists of six main layers:

3.1 Flutter Presentation Layer

The primary user-facing interface is a Flutter application (CAR_LEASE_INTELLIGENCE_APP) written in Dart. It targets web browsers as the primary platform while also supporting iOS, macOS, and Linux.

- Dark-themed, gradient-based UI with branded styling
- PDF upload with animated progress feedback
- Fully selectable text (SelectableText widgets) across all data
- Embedded chatbot widget accessible at all times
- Responsive layout for desktop, tablet, and mobile

3.2 Authentication Layer

A complete user-identity system protects access to the platform:

- Sign-up page — collects name, email, and password; posts to the auth router
- Login page — credential validation with inline error feedback
- Forgot-password page — initiates password-recovery via the backend
- Session-based access control enforced at the router level (auth.py)
- User-service layer (users.py) for credential and profile management

3.3 API Layer

Built on FastAPI, the API layer provides RESTful endpoints for all system operations. The API architecture includes:

- FastAPI framework for high-performance async operations
- CORS middleware for secure cross-origin requests
- Comprehensive error handling and validation
- OpenAPI/Swagger documentation at /docs endpoint
- Health check endpoints for system monitoring
- File upload handling with type validation

3.4 Business Logic Layer

The core processing layer implements the intelligent contract analysis pipeline:

- Document text extraction (extract_text.py) - Handles PDF parsing with OCR fallback
- LLM contract processing (fineline.py) - Orchestrates Gemini API for structured extraction
- Fairness scoring engine (fairness_score.py) - Implements multi-dimensional contract evaluation
- Red flag detection (red_flags.py) - Pattern matching for problematic contract terms

- Utility functions (utils.py) - Common data validation and processing helpers

3.5 AI/LLM Layer

The AI layer leverages Google's Gemini Flash model for intelligent document understanding:

- Google Generative AI (Gemini Flash) for rapid, cost-effective processing
- Structured prompt engineering for consistent JSON output
- Error handling and retry logic for API resilience
- Safe JSON parsing with fallback mechanisms
- Context window management for large documents

3.6 Data Processing Layer

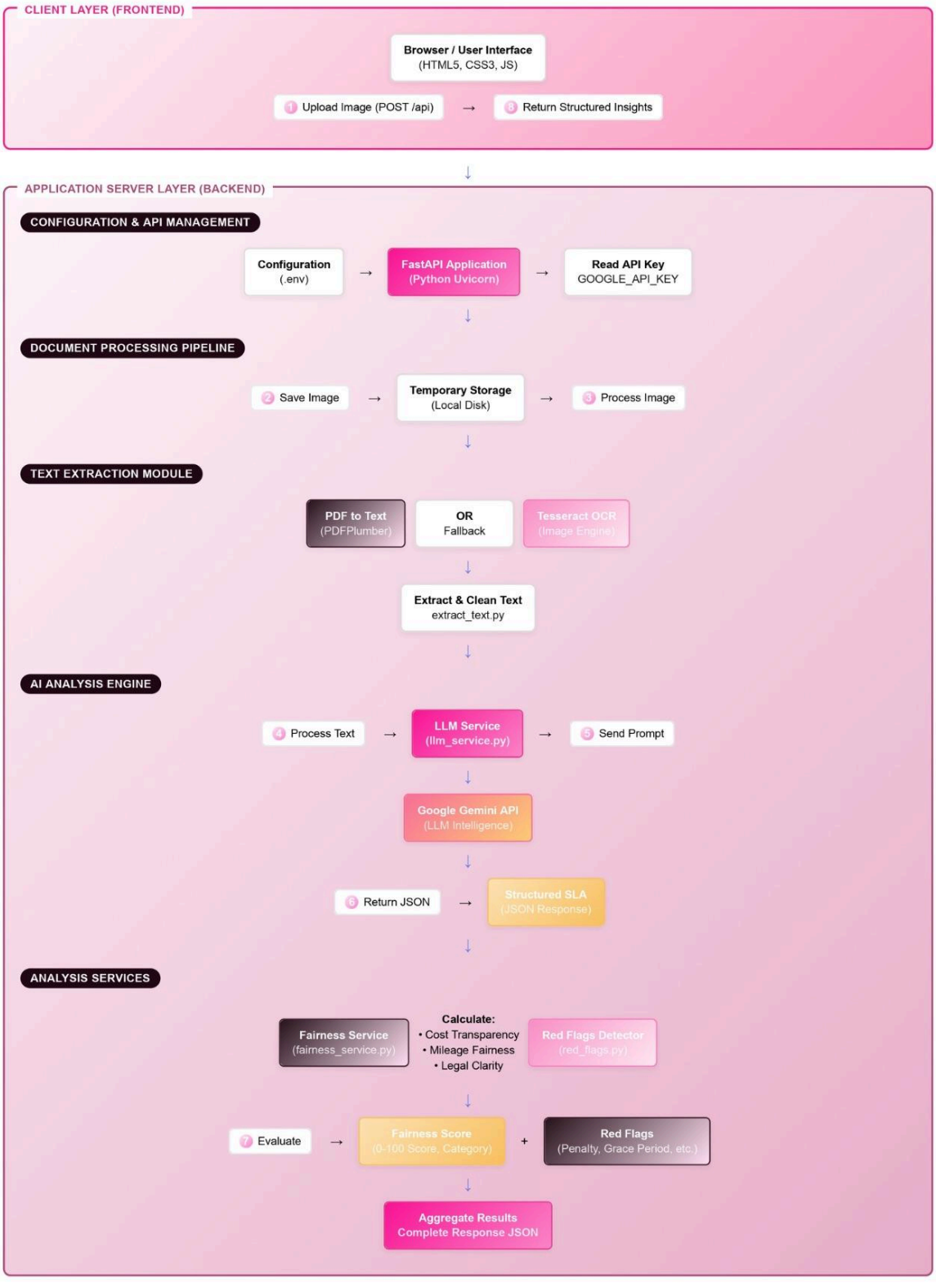
The foundation layer handles document ingestion and text extraction:

- PDFPlumber for native PDF text extraction
- PDF2Image + Pytesseract for OCR processing of scanned documents
- Intelligent fallback mechanism (native extraction → OCR if needed)
- Text preprocessing and noise reduction
- Temporary file management with automatic cleanup

System Architecture: OCR & Gemini Intelligent Contract Analysis

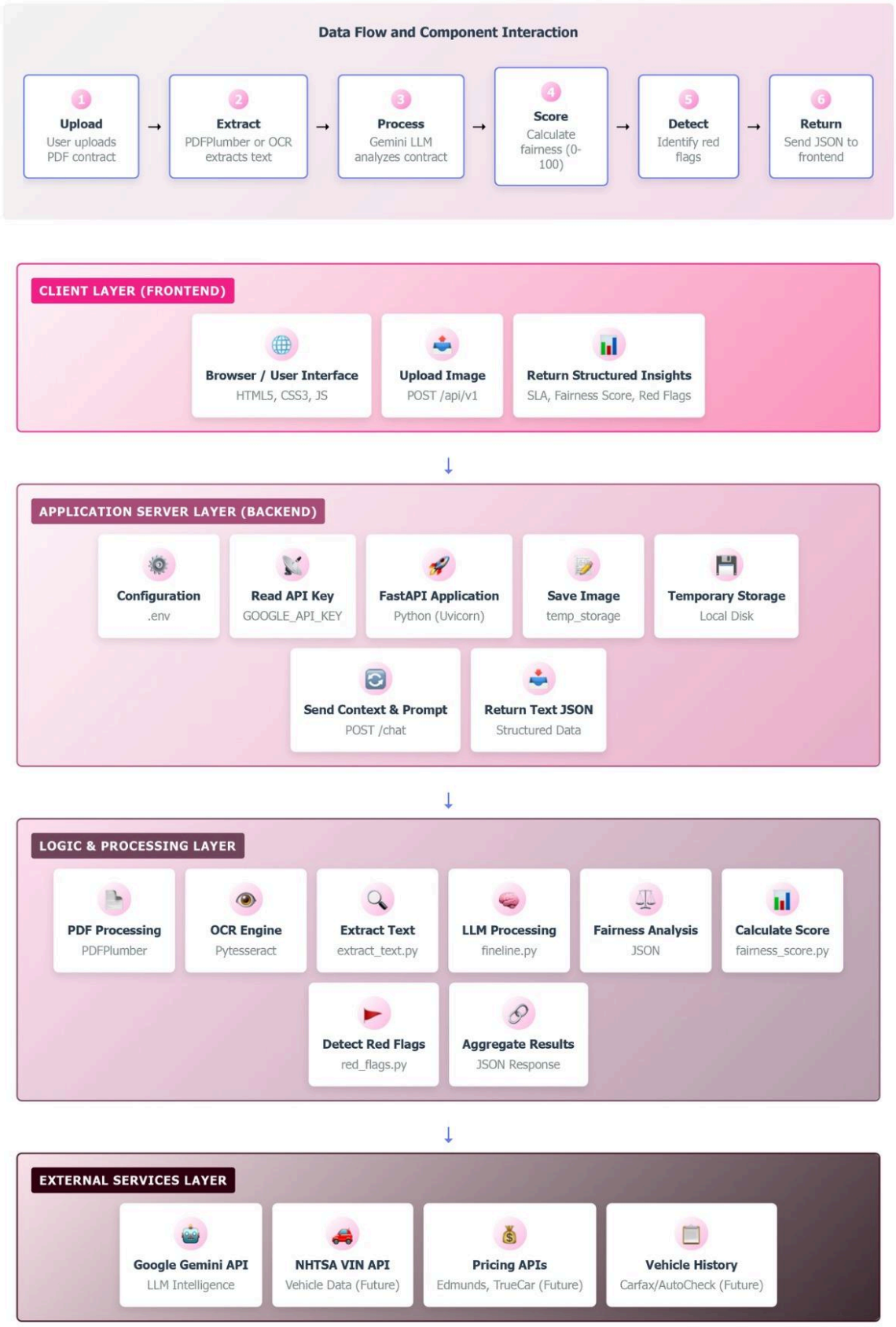
Automated Contract SLA Extraction and Fairness Analysis System Architecture

Figure 1: System Architecture - Data Flow and Component Interaction



System Architecture: AI-Powered Car Lease Contract Analysis

Automated Contract SLA Extraction and Fairness Analysis System



4. System Processing Flow.

The system processes lease contracts through a sophisticated six-stage pipeline:

Stage 1: Document Upload

The user uploads PDF via web interface. The system validates file type and size, generates a unique identifier, and stores temporarily for processing.

Stage 2: Text Extraction

PDFPlumber attempts native text extraction. If unsuccessful (scanned document), PDF2Image converts pages to images, and Pytesseract performs OCR. Raw text is validated for content presence.

Stage 3: Text Preprocessing

Raw text undergoes cleaning: email addresses removed, long numeric sequences filtered, whitespace normalized. First 3,500 characters prioritized for LLM processing.

Stage 4: LLM Extraction

Gemini Flash processes cleaned text with structured prompts containing 50+ field definitions across 9 major sections. Model returns JSON-formatted contract data with "Information not available" for missing fields.

Stage 5: VIN Verification

If a VIN is found, the NHTSA vPIC API decodes it. Returned attributes (make, model, year, body class, plant country) enrich the contract data.

Stage 5: Analysis & Scoring

Extracted data flows through two parallel analyzers: Fairness Scorer evaluates a contract across 5 dimensions (Cost Transparency, Mileage Fairness, Termination Clarity, Residual Logic, Legal Clarity) generating 0-100 score. Red Flag Detector identifies potentially problematic terms using pattern matching.

Stage 6: Response Generation

System aggregates all analysis results, injects fairness metrics and red flags into structured response, cleans up temporary files, and returns JSON to frontend for visualization.

5. Flutter Frontend — Detailed Overview

The Flutter application (CAR_LEASE_INTELLIGENCE_APP) is the primary user interface, written entirely in Dart with a clean separation of concerns.

5.1 File Structure

Frontend File Structure

```
CAR_LEASE_INTELLIGENCE_APP/
├── lib/
│   ├── models/              # Data models (lease, user, VIN)
│   ├── routing/             # Navigation & route definitions
│   ├── screens/             # Full-screen page containers
│   ├── services/            # API communication layer
│   └── widgets/
│       ├── chat_bot.dart       # Chatbot overlay widget
│       ├── home_page.dart      # Analysis results screen
│       ├── login_page.dart     # Login screen
│       ├── signup_page.dart    # Registration screen
│       ├── forgot_password_page.dart
│       └── main.dart           # App entry point & theme
├── test/
│       └── widget_test.dart
├── web/
│       ├── assets/
│       │       └── pink_logo.png      #Web Icon
│       ├── index.html       #Web App Entry Point
│       └── manifest.json      #Progressive Web App (PWA) Config
└── ios/  macos/  linux/        # Platform bindings
```

5.2 Authentication Screens

- login_page.dart — Email / password form, validation, error feedback, links to sign-up and forgot-password.
- signup_page.dart — Name, email, password registration; posts credentials to the auth backend router.
- forgot_password_page.dart — Sends a password-reset request to the backend.

5.3 Home Page & Results Display

- Each extracted section renders in its own dark card with a coloured section header.
- Fields returning "Information not available" are hidden by default.

- Fairness sub-scores display as colour-coded progress bars: green (>= 80 %), gold (>= 60 %), orange (>= 40 %), red (< 40 %).
- Red flags are listed as Flag 1, Flag 2, etc.
- All data text uses SelectableText widgets for copy-paste access.

5.4 Chatbot Widget

- chat_bot.dart — a floating widget backed by the chat_bot.py router.
- Receives the full extracted contract as context; answers questions via Gemini Flash.
- Maintains conversation history within the session.
- Accessible from any screen during results viewing.

6. AI Chatbot

The chatbot is a context-aware conversational assistant that helps users understand their lease contract without leaving the application.

6.1 Architecture

- Backend — chat_bot.py: a FastAPI router that receives the user's question together with the full lease_details JSON.
- LLM — Gemini Flash: prompted with the contract data and the question to produce a focused answer.
- Frontend — chat_bot.dart: maintains local conversation history, sends requests, and renders a bubble-style chat UI.

6.2 Capabilities & Limitations

- Answers questions about any field in the analysed contract.
- Explains fairness scores and red flags in plain language.
- Provides guidance on what missing information means for the lessee.
- History is session-scoped — does not persist across page reloads.
- Can only answer questions relating to the currently loaded contract.

7. VIN Lookup Integration

VIN decoding is handled by the NHTSA vPIC API — a free, officially maintained U.S. government service.

Property	*Detail*
Endpoint	*https://vpic.nhtsa.dot.gov/api/vehicles/decodevin/{vin}?format=json*
Method	*GET*
Auth	*None — publicly available*
Backend service	*vin_lookup.py (app/services/)*
Backend router	*vin.py (app/routers/)*

7.1 Returned Fields

- Manufacturer Name, Make, Model, Model Year, Vehicle Type, Body Class, Plant Country
- Full raw NHTSA response retained for extensibility

8. Data Extraction Schema

The system extracts structured information across nine comprehensive categories, covering all critical aspects of automotive lease contracts:

1. Vehicle Consultancy/Dealer Info

- Vehicle Consultancy Name
- Contact Details
- Dealer/Lessor Name
- Location

2. Vehicle Identification & Basic Details

- Car Name/Model
- Variant
- Vehicle Identification Number (VIN)
- Registered Number
- VAT Number
- Manufacturer
- Manufacturer OTR (On-The-Road Price)
- P11D (Official List Price - UK)

3. Vehicle Specifications

- Body Type
- Car Color
- Fuel Type
- Transmission
- CO_2 Emissions
- Vehicle Depreciation Rate per Month
- Mileage and Tenure Limit

4. Lease Terms/Agreement Details

- Agreement Duration/Term Period
- Rental Period (Start/End Dates)
- Expiry Date
- Termination Date
- Early Termination Fee
- Mileage Allowance
- Maintenance Responsibility

- Insurance Management Requirements
- Other Terms and Conditions/Disclaimer

5. Payment Details

- Upfront/Signing Payments:
- - Lease Signing Payment
- - Capitalized Cost Reduction
- - Net Trade-In Allowance
- - Down Payment
- - First Monthly Payment
- - Refundable Security Deposit
- - Amount to be Paid in Cash
- - Title Fees
- - Registration Fees
- - Processing Fee
- Recurring Payments:
- - Monthly Payments
- - Monthly Sales/Use Tax
- - Other Charges
- - Total Monthly Payment
- - Total of Payments
- - Amortized Amount over Lease

6. Taxes & Additional Fees

- VAT/Sales Tax
- Other Fees and Taxes

7. Residual & End-of-Lease Details

- Residual Value
- Vehicle Depreciation Considered
- Options at Lease End

8. Fairness Analysis (Auto-Generated)

- Overall Fairness Score (0-100)
- Risk Category (Fair/Needs Review/High Risk)
- Cost Transparency Score
- Mileage Fairness Score
- Termination Clarity Score

- Residual Logic Score
- Legal Clarity Score

9. Red Flags (Auto-Detected)

- List of identified problematic terms
- Missing critical information warnings
- Unusual or unfavorable conditions

9. Fairness Scoring Methodology

The fairness scoring system evaluates contracts across five critical dimensions, providing a transparent, objective assessment of contract quality. The scoring algorithm starts at 100 points and applies deductions based on missing or problematic information. Every category exposes both its earned score and its maximum (e.g. 25 / 35), rendered as a colour-coded progress bar in the Flutter UI.

9.1 Scoring Breakdown

Category	*Max*	*What is checked*
Cost Transparency	*35*	*Monthly payment (15), total cost (10), initial payment (10)*
Mileage Fairness	*25*	*Lease duration (10), mileage allowance (10), excess charge (5)*
Termination Clarity	*20*	*Early termination fee (10), termination date (10)*
Residual Logic	*20*	*Vehicle make/model (5), year (5), residual value (10)*
Legal Clarity	*15*	*Maintenance (5), insurance (5), VAT / tax (5)*
End-of-Lease Bonus	*+5*	*Residual value (+3), purchase option (+2) — bonus only*

9.2 Risk Categories

Score	*Category*	*Risk Level*
85 – 100	*Excellent*	*Low Risk*
70 – 84	*Good*	*Low to Moderate Risk*
55 – 69	*Fair*	*Moderate Risk*
40 – 54	*Needs Review*	*High Risk*
0 – 39	*Poor*	*Very High Risk*

9.3 Output Structure

compute_fairness_score() returns: fairness_score (int), category, risk_level, breakdown (dict mapping each category to {"score": X, "max": Y}), warnings (list of deduction reasons), max_possible_score, and a plain-English interpretation string.

9.4 Colour Coding in the UI

Colour	*Percentage Range*	*Meaning*
Green	*80 – 100 %*	*Strong transparency*
Gold	*60 – 79 %*	*Good, minor gaps*

Orange	*40 – 59 %*	*Moderate concerns*
Red	*0 – 39 %*	*Significant gaps*

10. Red Flag Detection System

The red flag detector identifies potentially problematic contract terms using pattern matching and semantic analysis. Flags are triggered when:

Missing Total Cost

The contract lacks "Total of Payments" disclosure, making it impossible to calculate the true cost of the lease.

Undefined Early Termination

Early termination fees are not specified, exposing consumers to potentially unlimited penalties.

Missing Residual Value

Residual value is not disclosed, preventing assessment of end-of-lease buyout options.

Lessee Maintenance Burden

The contract explicitly states maintenance is the lessee's responsibility, potentially resulting in unexpected repair costs.

Unilateral Price Changes

Terms include language like "subject to change" allowing the lessor to modify pricing without notice.

11. Technical Implementation Details

11.1 Technology Stack

Backend Framework: FastAPI 0.104+ (Python 3.11+)

AI/LLM: Google Generative AI (Gemini Flash)

PDF Processing: PDFPlumber, PDF2Image, Pytesseract

Frontend: HTML5, CSS3, Vanilla JavaScript

API Documentation: OpenAPI/Swagger (auto-generated)

Development Server: Uvicorn ASGI server

Testing: Custom test suite with colored output

11.2 API Endpoints

Endpoint	*Description*
GET /	*Root endpoint - API information and available endpoints*
GET /health	*Global health check*
GET /lease/health	*Lease service health check*
POST /lease/extract	*Main endpoint: Upload PDF and receive analysis*
GET /vin/decode/{vin}	*Decode VIN via NHTSA*
POST /chat/ask	*Question + context → LLM answer*
POST /auth/login	*Authenticate user*
POST /auth/signup	*Register new user*
POST /auth/forgot-password	*Initiate password recovery*
GET /docs	*Interactive Swagger/OpenAPI documentation*
GET /ui	*Web interface for contract analysis*

11.3 File Structure

Project organization follows modular best practices:

Backend File Structure

```
CAR_LEASE_AI/
├── app/
│   ├── main.py                 # FastAPI entry point
│   ├── routers/
│   │   ├── auth.py            # Login / signup / forgot-password
│   │   ├── chat_bot.py        # Chatbot endpoint
│   │   ├── lease.py           # PDF upload & extraction
│   │   └── vin.py             # VIN decode endpoint
│   ├── services/
│   │   ├── users.py           # User management
│   │   └── vin_lookup.py      # NHTSA API wrapper
│   ├── templates/
│   │   └── index.html         # Legacy web UI
│   └── utils/
├── extract_text.py            # PDF text extraction
├── fineline.py                # LLM processing
├── fairness_score.py          # Redesigned scoring
├── red_flags.py               # Red flag detection
├── utils.py                   # Helpers (is_missing, etc.)
├── test.py                    # Backend test suite
└── requirements.txt
```

11.4 Error Handling

The system implements comprehensive error handling at multiple levels:

- File type validation (PDF only)
- Empty document detection
- OCR fallback for scanned documents
- LLM API error recovery with detailed logging
- JSON parsing with safe fallback mechanisms
- HTTP status codes: 200 (success), 400 (bad request), 422 (validation error), 500 (server error)
- Structured error responses with error type and details
- Automatic temporary file cleanup (finally blocks)

12. Testing & Quality Assurance

A comprehensive test suite (test.py) validates all system components and integration points:

Backend Tests (test.py)

- Root endpoint validation
- Global and service-level health checks
- Invalid-file rejection (non-PDF)
- End-to-end PDF extraction with real lease contracts
- Swagger / OpenAPI accessibility

Flutter Tests

widget_test.dart — smoke and widget-level tests; key screens render without errors.

Test Output

- Colour-coded console output (green = pass, red = fail, yellow = skip)
- Field-population statistics and processing-time measurement
- JSON output file generation for manual inspection

13. Current Implementation Status

The system has successfully completed Milestones 1-3 of the original project plan, with core functionality fully operational:

13.1 Completed

- PDF upload & text extraction with OCR fallback
- LLM-based extraction of 50+ fields
- Redesigned fairness scoring with per-category breakdown
- Red flag detection
- VIN lookup via NHTSA vPIC API
- AI chatbot for contract Q&A
- Flutter cross-platform frontend
- User authentication (login, sign-up, forgot password)
- Selectable text and colour-coded progress bars
- Backend + Flutter test suites
- Swagger API documentation

13.2 Pending/Future Features

- Vehicle price estimation (Edmunds, TrueCar APIs)
- Market-price benchmarking
- Multi-contract side-by-side comparison dashboard
- Cloud storage for saved contracts
- App-store deployment (iOS / Android)

14. Deployment & Operations

14.1 Backend Setup

```
git clone <repo-url> && cd CAR_LEASE_AI
pip install -r requirements.txt
export GOOGLE_API_KEY="your-gemini-api-key"
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

14.2 Flutter Setup

```
cd CAR_LEASE_INTELLIGENCE_APP
flutter pub get
flutter run -d chrome          # web
flutter run                    # default device / emulator
```

14.3 Production Architecture

- Docker + Kubernetes for container orchestration
- Nginx reverse proxy, Gunicorn + Uvicorn workers
- PostgreSQL for contract / user storage; Redis for caching
- Flutter app hosted via Firebase Hosting or Vercel
- Prometheus + Grafana monitoring; ELK stack logging

14.4 Security

- API rate limiting, 10 MB upload cap, virus scanning
- HTTPS enforcement, CORS restricted to authorised domains
- API keys stored as environment variables
- Input sanitisation, OWASP scanning, regular dependency updates

15. Performance & Scalability

15.1 Metrics

Operation	*Latency*
Full contract processing	*10 – 30 s*
PDF text extraction	*1 – 3 s*
Gemini Flash extraction	*5 – 15 s*
Fairness scoring	*< 1 s*
Red flag detection	*< 1 s*
VIN decode (NHTSA)	*1 – 3 s*
Chatbot response	*2 – 8 s*
Max PDF size	*10 MB*
Concurrent users (single instance)	*10 – 20*

15.2 Scalability Strategy

Horizontal scaling approach for production:

- Stateless API design enables load balancing across multiple instances
- Task queue (Celery + Redis) for asynchronous processing
- Database read replicas for query scaling
- CDN for static asset delivery
- API response caching for repeated queries
- Auto-scaling based on CPU/memory metrics
- Microservices architecture for independent component scaling

15.3 Optimization Opportunities

- LLM prompt optimization to reduce token usage
- Caching of common contract patterns
- Batch processing for multiple contracts
- Model fine-tuning for faster inference
- Database query optimization and indexing
- Lazy loading of non-critical UI components

16. Conclusion

The Car Lease Contract Review and Negotiation AI Assistant represents a significant advancement in consumer empowerment for automotive transactions. By leveraging cutting-edge AI technology, the system transforms complex legal documents into actionable insights, enabling consumers to make informed decisions and negotiate better terms.

The modular backend architecture proved its value — each new feature (authentication, chatbot, VIN lookup) was added as an independent router without disrupting existing functionality. As the system evolves through market-price benchmarking, multi-contract comparison, and enterprise features, this architectural foundation ensures smooth, sustainable growth.

It has the potential to become an essential tool for millions of consumers, reducing information asymmetry in the automotive marketplace and promoting fairer, more transparent transactions.

Appendix A: Sample API Request/Response

Sample cURL request:

```
curl -X POST "http://localhost:8000/lease/extract" \
  -H "accept: application/json" \
  -H "Content-Type: multipart/form-data" \
  -F "file=@lease_contract.pdf"
```


```
{
  "success": true,
  "filename": "lease_contract.pdf",
  "lease_details": {
    ...
    "8. Fairness Analysis": {
      "Fairness Score": "72 / 100",
      "Risk Category": "Good",
      "Cost Transparency":     { "score": 25, "max": 35 },
      "Mileage Fairness":      { "score": 20, "max": 25 },
      "Termination Clarity":   { "score": 10, "max": 20 },
      "Residual Logic":        { "score": 10, "max": 20 },
      "Legal Clarity":         { "score": 7,  "max": 15 },
      "Warnings": ["Early termination fee not specified", ...]
    },
    "9. Red Flags": { "Flag 1": "...", ... }
  }
}
```