# Multi Agent Credit Risk Automation System

*Technical Architecture & Workflow Documentation*

## Table of Contents

## 1.System Overview

This document outlines the technical architecture of a proof-of-concept credit risk system designed for retail lending and embedded finance organizations. The solution leverages a multi-agent AI pipeline to automate the underwriting process for Buy Now, Pay Later (BNPL) loans, delivering a 50% decrease in processing time.

## 2.Use Case Summary

### 2.1 Business Context
Client Profile: Retail lending, embedded finance organizations offering Buy Now, Pay Later (BNPL) loans
Challenge: Manual underwriting processes dependent on analyst review and static credit bureau reports
Market Pressure: Rising demand for faster, fairer, and more explainable credit decisions
Time-to-Market: Current underwriting decisions take hours to days, slowing loan approvals and impacting customer experience

### 2.2 Problem Statement
● Manual Underwriting: Reliance on human review leads to delays and inconsistencies
● Limited Automation: Legacy systems lack intelligent decisioning and real-time risk assessment
● Fragmented Data: Applicant data scattered across credit bureaus, internal systems, and public records
● Regulatory Pressure: Need for explainable, auditable credit decisions to meet compliance requirements

### 2.3 Solution Value Proposition
Multi-Agent Orchestration: AI-powered pipeline with specialized agents for data intake, feature engineering, risk scoring, and decision explanation
Hybrid AI Approach: Combination of LLM reasoning and ML models for accurate, explainable underwriting decisions
Near Real-Time Processing: Automated risk assessment reduces underwriting time from hours to seconds
Auditability and Compliance: Structured trace of agent outputs and model explanations for regulatory reporting

## 3. Key Performance Indicators
- **Latency:** ~16 s (end-to-end, single profile)
- **Balanced Accuracy:** 0.729
- **Precision:** 0.380
- **Recall:** 0.775
- **F1-Score:** 0.510

## 4. Solution Overview

### 4.1 Core Technology Stack
**Frontend:** Streamlit UI for interactive applicant intake and results visualization
**Authentication**: Environment-based secure API key management for agent orchestration
**Orchestration Layer:** Python-based DAG pipeline coordinating agents with clear execution flow
**AI/ML Services:** Lyzr API for LLM reasoning, scikit-learn models for credit risk scoring with SHAP explainability

**Data Processing**: Pandas and NumPy for feature engineering and derived metric computation
**Storage:** Local model artifacts and configuration files
**Monitoring:** Verbose logging with structured JSON traces for auditability and debugging

### 4.2 Business Capabilities

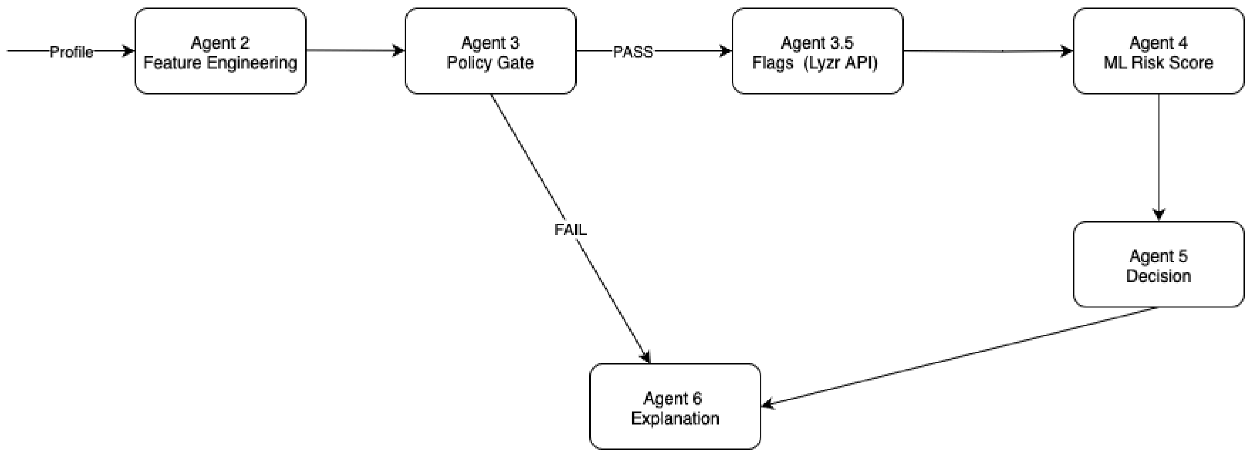
● Applicant Intake: Manual data entry in Streamlit simulating credit bureau and internal applicant data
● Feature Engineering: Dynamic calculation of derived credit metrics
● Risk Flagging: Configurable red/green flag detection (with RAG retrieval)
● ML-Driven Scoring: Probability of default prediction with SHAP-based explainability
● Decision Automation: Policy-based approval, review, or rejection with full traceability for audits
● Visual Reporting: Streamlit UI displaying applicant profile, risk factors, and final decision in real time

## 5. System Architecture Overview

### 5.1 Major Layers

Frontend & End-User Layer: Streamlit UI for manual applicant data entry and real-time results visualization
Backend Orchestration Layer: Python DAG pipeline coordinating Agents 2–6 with clear dependencies
AI/ML Reasoning Layer: Lyzr API for reasoning (Agent 3.5), scikit-learn models for risk scoring with SHAP explainability (Agent 4)
Knowledge Retrieval Layer: RAG-backed document retrieval in Agent 3.5 for red/green flag generation

### 5.2 Architecture Diagram



## 6. Detailed Technical Workflow

### 6.1 Data Processing & Intake (Agent 1)

Entry: Manual data entry in Streamlit form simulating credit bureau and personal data

Validate Input: Throw an error if any of the following required fields are missing:

- `identity`
- `credit_bureau_data`
- `location`
- `family`
- `socioeconomic`
- `application_context`

### 6.2 Feature Engineering (Agent 2)

Derive set of core credit metrics and append to profile JSON

- **DTI** = `total_monthly_debt_payments / (annual_income / 12)`
- **PTI** = `(purchase_amount / term_months) / (annual_income / 12)`
- **HCR** = `monthly_housing_cost / (annual_income / 12)`
- **ResidualMonthlyIncome** = `(annual_income / 12) - total_monthly_debt_payments - monthly_housing_cost`
- **CreditVelocity** = `total_open_accounts / credit_history_years`

### 6.3 Policy Gate & Hard-No Rules (Agent 3)

Policy checks applied upfront to short-circuit unqualified profiles.
Current thresholds:

- `age < 18`
- `credit_score < 520 or > 900`
- `residual_monthly_income < 0`
- `derogatory_marks_any = true`
- `employment_duration_years < 0 or > (age - 13)`
- `credit_utilization` outside `[0,1]`

**Output:** `status` (PASS / REJECT) + reason if rejected. If rejected → jump directly to Agent 6 for explanation.

### 6.4 Risk Flagging & Prior Risk (Agent 3.5)

Agent 3.5 is responsible for identifying nuanced risk indicators. It combines deterministic, rule-based checks with a RAG-backed Lyzr call for contextual, LLM-driven inference.

**Field Extraction**

- Reads applicant profile data
- Extracts numeric values such as `DTI`, `PTI`, `HCR`, `CreditVelocity`, `ResidualMonthlyIncome`, `total_balance`, `total_credit_limit`, `employment_duration_years`, and `credit_history_years`.

**Deterministic Flag Generation**

Runs local consistency and affordability checks (priority + severity assigned):

- Utilization Consistency — mismatch between reported vs. computed utilization.

- Balance Exceeds Limit — over-limit credit utilization.
- History vs Age — credit history longer than feasible age range.
- High PTI / DTI — PTI > 0.15 or DTI > 0.43 flagged as high risk.
- Employment vs Income Mismatch — unemployed with positive income.
- Dependents with $0 Expense — data quality issue.
- Housing Cost Stress — elevated (≥30%) or severe (≥50%) HCR.
- Negative Residual Income — housing + debt > monthly income.
- Credit Velocity > 2 — excessive new account opening rate.
- Residence vs Cost Mismatch — rent/mortgage/own reported but $0 housing cost.

Flags are created as structured dictionaries:

```
{
   "flag_type": "red",
     "name": "High PTI",
     "reasoning": "PTI 0.20 > allowable 0.15",
     "severity": "high"
   }
```

Flags are sorted by `(priority, severity)`, deduplicated, and capped to 5 per profile.

**Lyzr Augmentation (RAG)**

Agent 3.5 invokes the Lyzr agent API to infer additional red/green flags grounded in domain knowledge.

**Knowledge Retrieval:**

- The knowledge base is preprocessed into semantically searchable embeddings and stored in a vector index
- At runtime, the system performs a top-k semantic search using the applicant context
- Merges retrieved context with deterministic flags before passing it to **GPT-4o-mini** for reasoning

**Payload Construction:**

- Includes the applicant `profile`, `initial_red_flags` from local checks, and the `remaining_slots` available (ensuring the total does not exceed 5).

**Response Handling:**

- Post the payload to the Lyzr API and robustly parse the response.
- Normalizes the output into a consistent format for downstream processing.
- Flags are converted into a numeric prior risk score using a deterministic, configurable points table based on flag type (red/green) and severity (low/medium/high). The process is handled by `score_flags_v0()`, which outputs:
    - total_score – aggregated risk score
    - counted – number of flags contributing to the score
    - ignored – number of flags beyond the cap (default 5)
    - breakdown – per-flag contribution details

**Final Output:**

Returns combined result for downstream use:

```
{
  "flags": [...],
  "prior_risk_score":
}
```
`flags` → included in the decision trace (Agent 6 explanations).

`prior_risk_score` → passed as a feature to Agent 4's ML model

### 6.5 Risk Scoring (Agent 4)

Agent 4 is the ML-driven risk scorer responsible for estimating the probability of default (PD) and producing interpretable feature-level explanations for each decision.

1. **Input & Feature Alignment**
    - Accepts the profile JSON output from Agent 2 and `prior_risk_score` from Agent 3.5
    - Flattens nested fields (`credit_bureau_data`, `socioeconomic`, etc.) into a tabular row
    - Ensures all expected numeric and categorical columns are present, imputing missing values as `NaN`
2. **Model Training**
    - **Dataset:** John Hull's Lending Club dataset with inferred loan status (0/1)
    - **Split:** 60/20/20 stratified Train/Validation/Test split.
    - **Model:** `HistGradientBoostingClassifier` with tuned hyperparameters
    - **Calibration:** Isotonic regression on validation probabilities to ensure well-calibrated PD estimates.
    - **Threshold Selection:** Optimal decision threshold $\tau$ chosen to maximize Balanced Accuracy on the validation set.
    - **Metrics Recorded:** Balanced Accuracy, Precision, Recall, F1, and Confusion Matrix.
    - **Artifacts Saved:**
        - `agent4_risk_model.joblib` – Calibrated classifier
        - `agent4_meta.joblib` – Metadata
        - `agent4_metrics.json` – Performance metrics for reproducibility
        - `agent4_confusion_matrix.csv` – Test set confusion matrix
3. **Inference (Runtime)**
    - Loads the saved model and metadata.
    - Predicts **PD** (float, 0–1) for the applicant profile.
    - Converts PD → `risk_score` (0–100) for downstream decisioning (Agent 5).
4. **Explainability (SHAP)**
    - Uses **Kernel SHAP** to compute local feature attributions:
    - Produces top-k (default 3) factors sorted by absolute SHAP value.
    - Each factor includes:
      ```
      {
        "feature": "DTI",
        "value": 0.48,
      ```

```
  "direction": "+",
  "abs_contribution": 0.06,
  "contribution": 0.06
}
```

`"direction": "+"` means the feature increased predicted risk.
`"direction": "-"` means the feature decreased predicted risk.

5. **Model-Derived Flags**
    - `extract_model_flags_from_shap()` maps SHAP-positive features to standardized flags (e.g., `HIGH_DTI`, `LOW_SCORE`, `SHORT_HISTORY`).
    - These flags are passed along with deterministic flags to Agent 6 for explanation generation.
6. **Output**
   Agent 4 produces a clean JSON object for downstream agents:

```
{
  "pd": 0.42,
 "risk_score": 42,
"top_factors": [
   {"feature": "DTI", "value": 0.48, "direction": "+",
"abs_contribution": 0.06},
   {"feature": "credit_score", "value": 640, "direction":
"-", "abs_contribution": 0.05}
  ]
}
```

## 6.6 Decisioning (Agent 5)

Agent 5 converts the outputs from prior agents into a final decision outcome.

- **Policy Gate Integration:** If Agent 3 returned a hard-no, Agent 5 immediately issues a DECLINE without considering model output.
- **Score Aggregation:** If policy is passed, combine the prior risk score from Agent 3.5 and the risk score from Agent 4 using a weighted linear blend (weights fixed in code but configurable).
- **Thresholding:** Compares the resulting composite score against two cutoffs:
    - **APPROVE:** Composite score below lower cutoff
    - **APPROVE WITH CONDITION:** Composite score between lower and upper cutoffs (e.g., partial approval at 25% of requested amount)
    - **DECLINE:** Composite score above upper cutoff

**Configurable Parameters:**

- Both weights and cutoffs are tunable in code, allowing re-calibration without changing the pipeline logic.

## 6.7 Decision Explanation (Agent 6)

Agent 6 consolidates all upstream signals (policy outcomes, deterministic flags, and ML-driven feature attributions) into a final, auditable decision rationale.

- **Policy Handling:** If the profile fails Agent 3's hard-no checks, Agent 6 immediately returns a policy-based decline with a single standardized code.
- **Flag Mapping:** Deterministic flags (Agent 3.5) and model-derived flags (Agent 4) are mapped to a controlled vocabulary of reason codes using lookup tables, with fallbacks for unmapped cases.
- **Prioritization:** Codes are deduplicated, sorted by business-defined priority to keep explanations concise.
- **Traceability:** Produces a structured trace object listing raw flags, mapped codes, and their source (policy vs ML).

**Output Example:**

```
{
  "basis": "flags",
  "codes": ["R001", "R006"],
  "labels": ["High DTI", "Severe Housing Cost Burden"],
  "trace": {
    "ml_flags": ["HIGH_DTI"],
    "det_flags": [{"name": "Housing cost burden (severe)"}]
  }
}
```

## 6.8 Storage Design

Dataset: CSV file with 1000-sample subset for reproducible tests
Features & Flags: Appended to profile JSON per run
Models: Stored as `agent4_risk_model.joblib` and `agent4_meta.joblib`
Traces: Stored as structured JSON (decision trace, flags, SHAP outputs) for compliance/auditability

## 7. Evaluation

The pipeline was validated on a stratified sample of labeled applications, reporting balanced accuracy, precision, recall, and F1-score. Confusion matrix and probability cutoffs were reviewed to ensure decisions aligned with risk tolerance.

## 8. Future Scope

Introduce an additional intake agent dedicated to data preprocessing and data collection. This agent will ingest data about the user from multiple sources (e.g., credit bureau reports, application forms, internal records), normalize field names and formats, and output a single standardized json profile for downstream agents. This will make the pipeline more robust and easier to integrate with real-world data feeds.

## 9. Conclusion

This proof-of-concept demonstrates an end-to-end, modular credit risk underwriting pipeline that combines feature engineering, deterministic policy checks, retrieval-augmented risk flagging, calibrated machine learning risk scoring, and a traceable explanation layer. The system produces consistent, auditable decisions and highlights which factors contributed to each outcome, ensuring transparency throughout the process.

By unifying these components into a single flow, the pipeline reduces manual review effort, and improves decision consistency. This PoC establishes a foundation for scaling credit risk automation to larger datasets, refining evaluation metrics, and optimizing performance over time.