Opulence Mainframe Deep Research Agent Architecture

1. Simple System Overview

The Opulence system has been enhanced to take legacy mainframe code, data and file structure and makes it understandable program logic in a structured manner, data flow within the subsystem using legacy mainframe code and data storage from vsam files, to determine if the usage processing for files and programs, field usage and duplication of fields and files and determine obsolete or duplicated data structures.

- **Orchestration**: A Coordinator Agent manages the workflow across various specialized research agents
- **Output**: Provides lineage maps showing how customer data flows, business logic summaries explaining trading rules, comprehensive documentation, and an interactive chat interface for asking questions

Example Scenario: Understanding how a customer's security purchase order flows through 50+ COBOL programs, what validation rules apply, and how it updates the portfolio database.

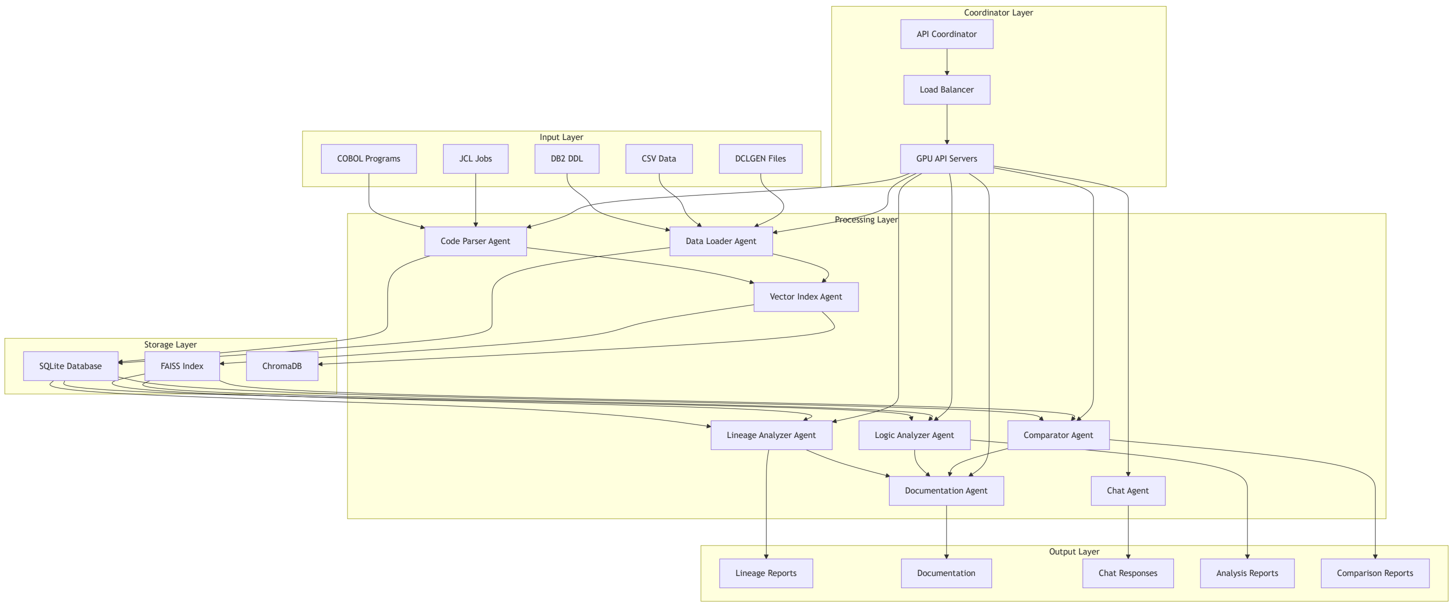
2. Core Components

Component	Function	Value
Code Parser	Converts COBOL/JCL into structured AST	Enables structured understanding of 40-year-old logic
Data Loader	Loads DB2 tables and sample transaction files	Adds real-world context from actual customer trades
Vector Index Agent	Embeds and indexes all elements in FAISS	Powers fast semantic search: "find all margin calculation logic"
Lineage Agent	Tracks fields across jobs and programs	Critical for compliance: trace customer ID through entire system

Component	Function	Value
Logic Analyzer Agent	Extracts business logic and conditional rules	Automates discovery of trading rules and validation logic
Comparator Agent	Compares similar files and identifies patterns	Finds duplicate logic, unused fields, and optimization opportunities
Documentation Agent	Summarizes components and logic	Generates readable docs explaining arcane settlement processes
Chat Agent	Interfaces with user to answer questions	"How does stop-loss order processing work?" gets instant answers
Coordinator Agent	Orchestrates flow and agent sequencing	Ensures systematic analysis of interconnected trading systems
GPU LLM API	CodeLLaMA exposed via API for summarization	Core intelligence for understanding legacy financial code

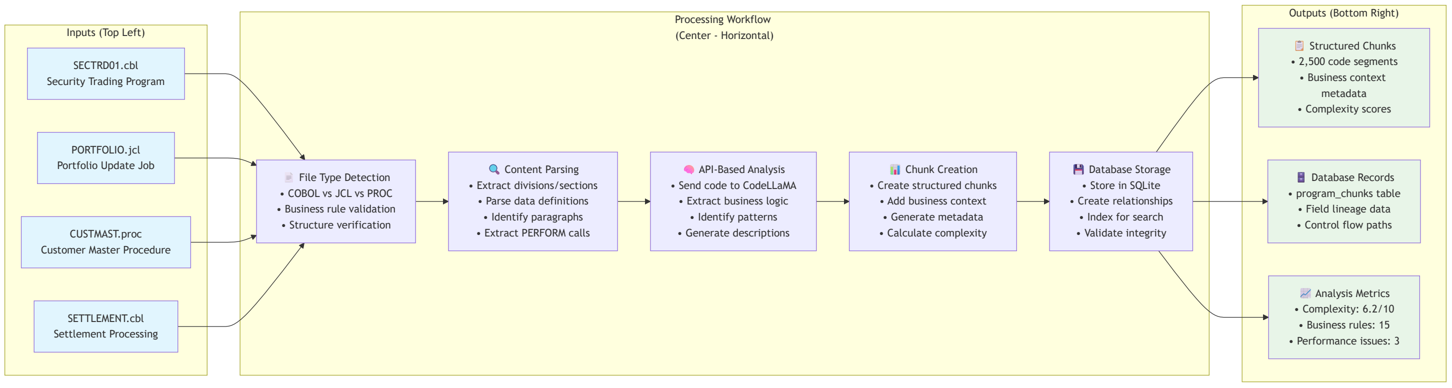
3. System Flow and Individual Agent Workflows

Overall System Architecture Flow



4. Individual Agent Workflows

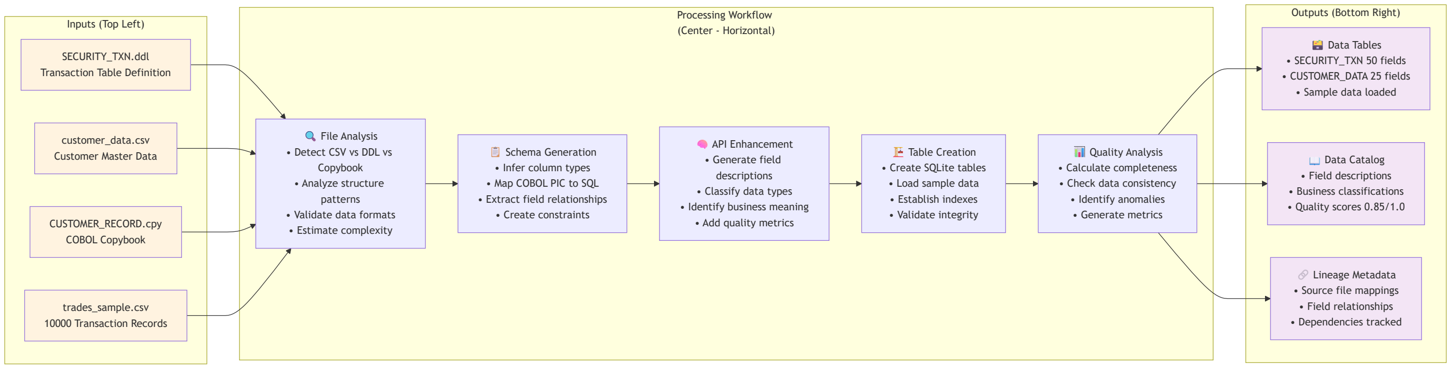
4.1 Code Parser Agent Flow



Sample Output:

```
{
  "program_name": "SECTRD01.cbl",
  "total_chunks": 156,
  "complexity_score": 6.2,
  "business_rules_found": 15,
  "performance_issues": 3,
  "key_sections": [
    {
      "section": "VALIDATE-ORDER",
      "line_start": 245,
      "line_end": 387,
      "complexity": 8.1,
      "business_logic": "Validates customer orders against credit limits and risk parameters"
    }
  ]
}
```

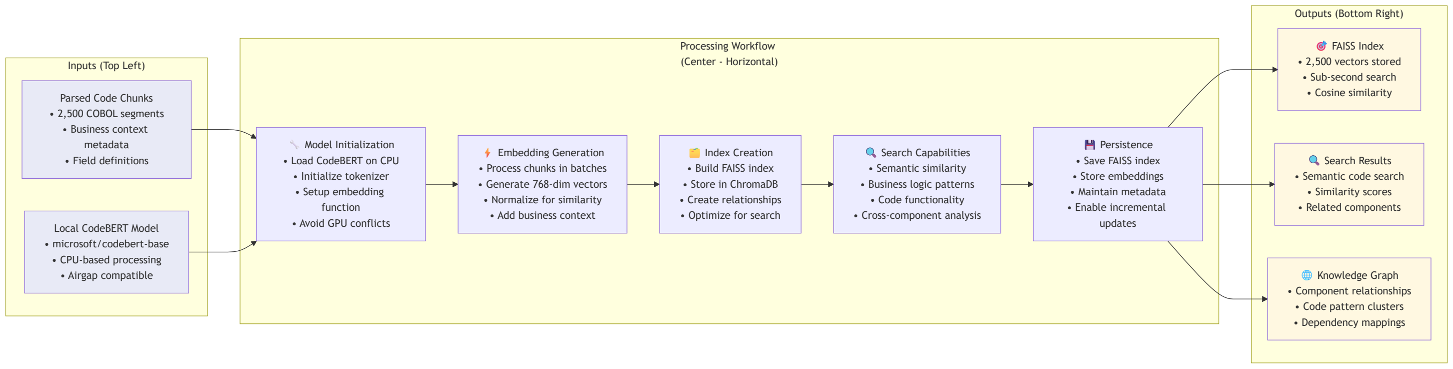
4.2 Data Loader Agent Flow



Sample Output:

```
{
  "table_name": "SECURITY_TXN",
  "total_fields": 50,
  "data_quality_score": 0.85,
  "loaded_records": 10000,
  "field_classifications": {
    "CUSTOMER_ID": {
      "type": "CHAR(10)",
      "description": "Primary customer identifier",
      "business_category": "Customer Reference",
      "completeness": 1.0
    },
    "TRADE_AMOUNT": {
      "type": "DECIMAL(15,2)",
      "description": "Total trade value in USD",
      "business_category": "Financial",
      "completeness": 0.98
    }
  }
}
```

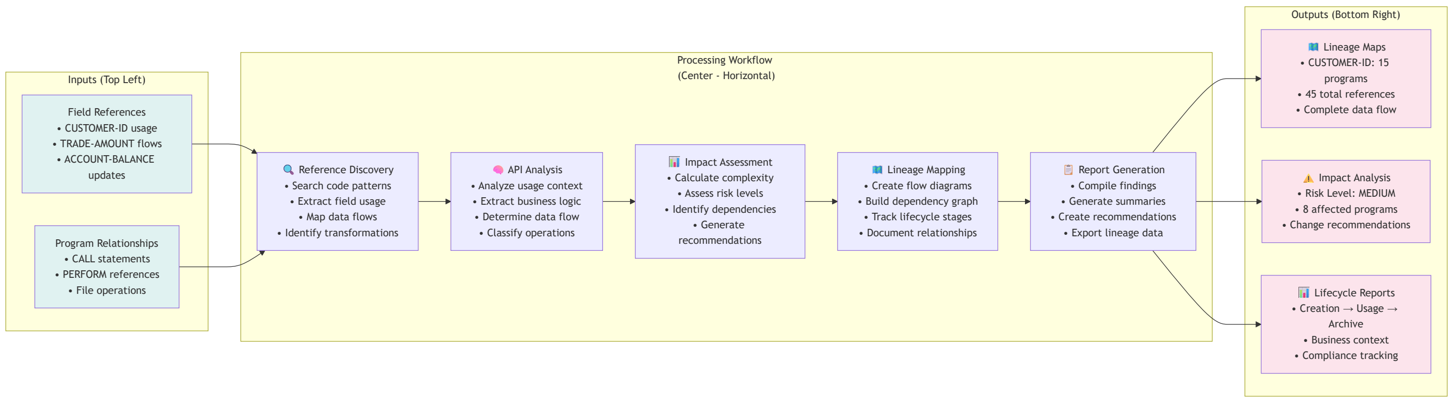
4.3 Vector Index Agent Flow



Sample Output:

```
{
  "index_stats": {
    "total_vectors": 2500,
    "embedding_dimension": 768,
    "index_size_mb": 45.2,
    "search_time_ms": 23
  },
  "search_results": [
    {
      "query": "customer credit validation",
      "matches": [
        {
          "chunk_id": "SECTRD01_245_387",
          "similarity_score": 0.94,
          "content": "VALIDATE-CUSTOMER-CREDIT section",
          "program": "SECTRD01.cbl"
        }
      ]
    }
  ]
}
```

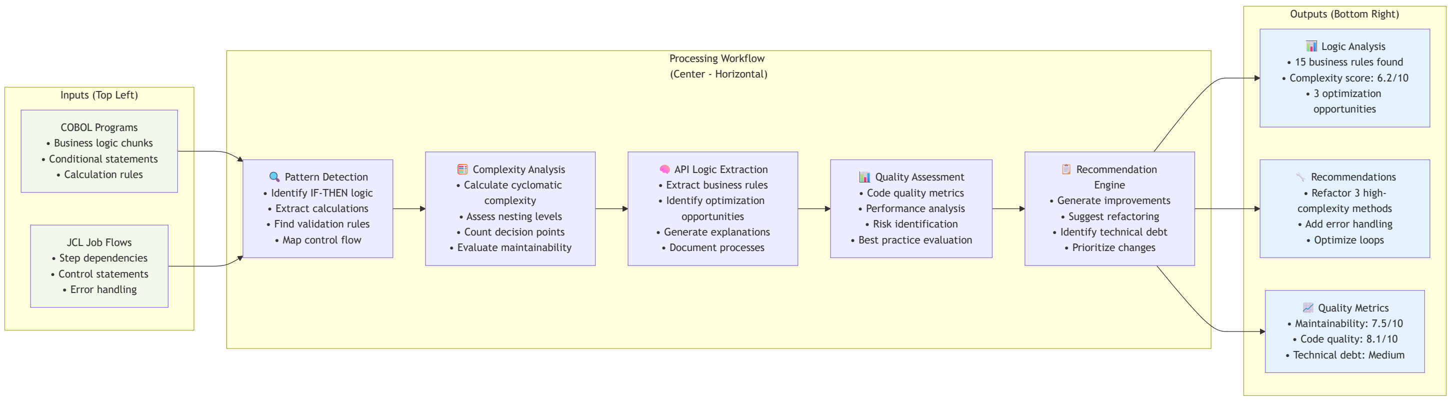
4.4 Lineage Analyzer Agent Flow



Sample Output:

```json
{
  "field_name": "CUSTOMER_ID",
  "total_references": 45,
  "programs_affected": 15,
  "lineage_flow": [
    {
      "program": "CUSTMAST.cbl",
      "operation": "CREATE",
      "line_number": 156,
      "context": "Initial customer registration"
    },
    {
      "program": "SECTRD01.cbl",
      "operation": "READ",
      "line_number": 245,
      "context": "Order validation lookup"
    },
    {
      "program": "PORTFOLIO.cbl",
      "operation": "UPDATE",
      "line_number": 389,
      "context": "Portfolio balance update"
    }
  ],
  "impact_analysis": {
    "risk_level": "MEDIUM",
    "change_complexity": 7.2,
    "affected_business_processes": 8
  }
}
```

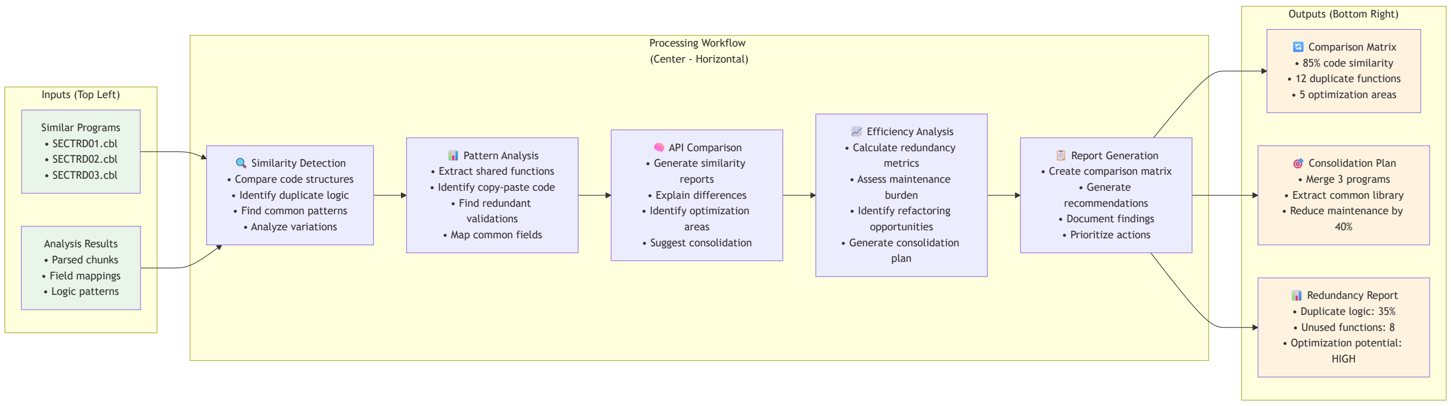
4.5 Logic Analyzer Agent Flow


Inputs (Top Left)
COBOL Programs • Business logic chunks • Conditional statements • Calculation rules
JCL Job Flows • Step dependencies • Control statements • Error handling
Processing Workflow (Center - Horizontal)
🔍 Pattern Detection • Identify IF-THEN logic • Extract calculations • Find validation rules • Map control flow
🧮 Complexity Analysis • Calculate cyclomatic complexity • Assess nesting levels • Count decision points • Evaluate maintainability
🧠 API Logic Extraction • Extract business rules • Identify optimization opportunities • Generate explanations • Document processes
📊 Quality Assessment • Code quality metrics • Performance analysis • Risk identification • Best practice evaluation
📋 Recommendation Engine • Generate improvements • Suggest refactoring • Identify technical debt • Prioritize changes
Outputs (Bottom Right)
📊 Logic Analysis • 15 business rules found • Complexity score: 6.2/10 • 3 optimization opportunities
🔧 Recommendations • Refactor 3 high-complexity methods • Add error handling • Optimize loops
📈 Quality Metrics • Maintainability: 7.5/10 • Code quality: 8.1/10 • Technical debt: Medium

Sample Output:

```
{
  "program_name": "SECTRD01.cbl",
  "business_rules_extracted": 15,
  "complexity_metrics": {
    "cyclomatic_complexity": 6.2,
    "nesting_levels": 4,
    "decision_points": 23,
    "maintainability_score": 7.5
  },
  "extracted_rules": [
    {
      "rule_id": "LARGE_ORDER_CHECK",
      "condition": "IF TRADE-AMOUNT > 250000",
      "action": "PERFORM MANUAL-APPROVAL-PROCESS",
      "business_context": "Orders over $250K require manual approval"
    }
  ],
  "recommendations": [
    {
      "type": "REFACTOR",
      "priority": "HIGH",
      "description": "Break down VALIDATE-ORDER section - too complex"
    }
  ]
}
```

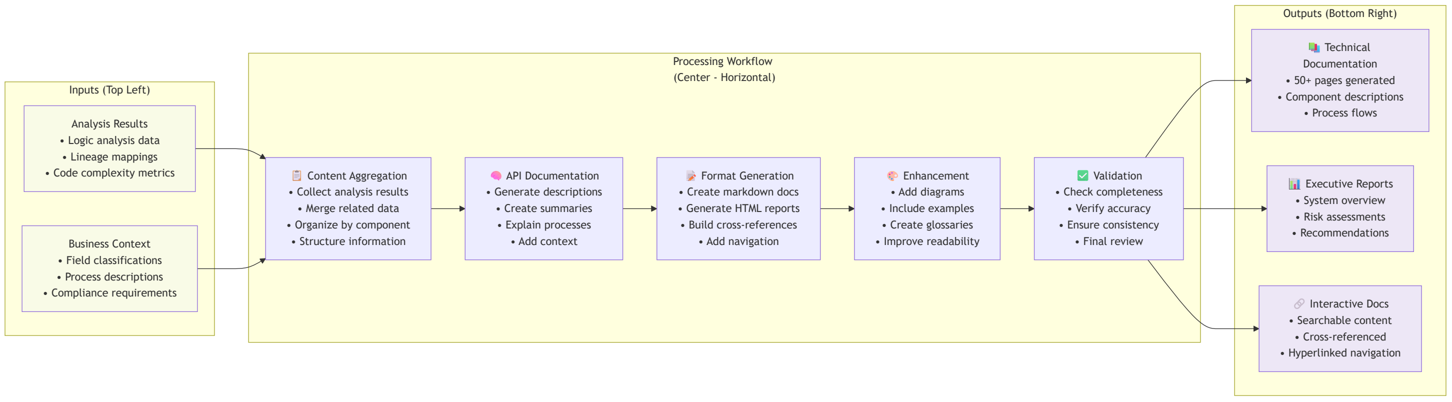
4.6 Comparator Agent Flow


Inputs (Top Left)
Similar Programs • SECTRD01.cbl • SECTRD02.cbl • SECTRD03.cbl
Analysis Results • Parsed chunks • Field mappings • Logic patterns
Processing Workflow (Center - Horizontal)
Similarity Detection • Compare code structures • Identify duplicate logic • Find common patterns • Analyze variations
Pattern Analysis • Extract shared functions • Identify copy-paste code • Find redundant validations • Map common fields
API Comparison • Generate similarity reports • Explain differences • Identify optimization areas • Suggest consolidation
Efficiency Analysis • Calculate redundancy metrics • Assess maintenance burden • Identify refactoring opportunities • Generate consolidation plan
Report Generation • Create comparison matrix • Generate recommendations • Document findings • Prioritize actions
Outputs (Bottom Right)
Comparison Matrix • 85% code similarity • 12 duplicate functions • 5 optimization areas
Consolidation Plan • Merge 3 programs • Extract common library • Reduce maintenance by 40%
Redundancy Report • Duplicate logic: 35% • Unused functions: 8 • Optimization potential: HIGH

Sample Output:

```
{
  "comparison_summary": {
    "files_compared": ["SECTRD01.cbl", "SECTRD02.cbl", "SECTRD03.cbl"],
    "similarity_score": 0.85,
    "duplicate_functions": 12,
    "redundant_code_percentage": 35
  },
  "duplicate_patterns": [
    {
      "pattern": "VALIDATE-CUSTOMER-CREDIT",
      "occurrences": 3,
      "files": ["SECTRD01.cbl", "SECTRD02.cbl", "SECTRD03.cbl"],
      "consolidation_opportunity": "HIGH"
    }
  ],
  "optimization_recommendations": [
    {
      "type": "EXTRACT_COMMON_LIBRARY",
      "description": "Create shared validation library",
      "estimated_effort": "3 weeks",
      "maintenance_reduction": "40%"
    }
  ]
}
```

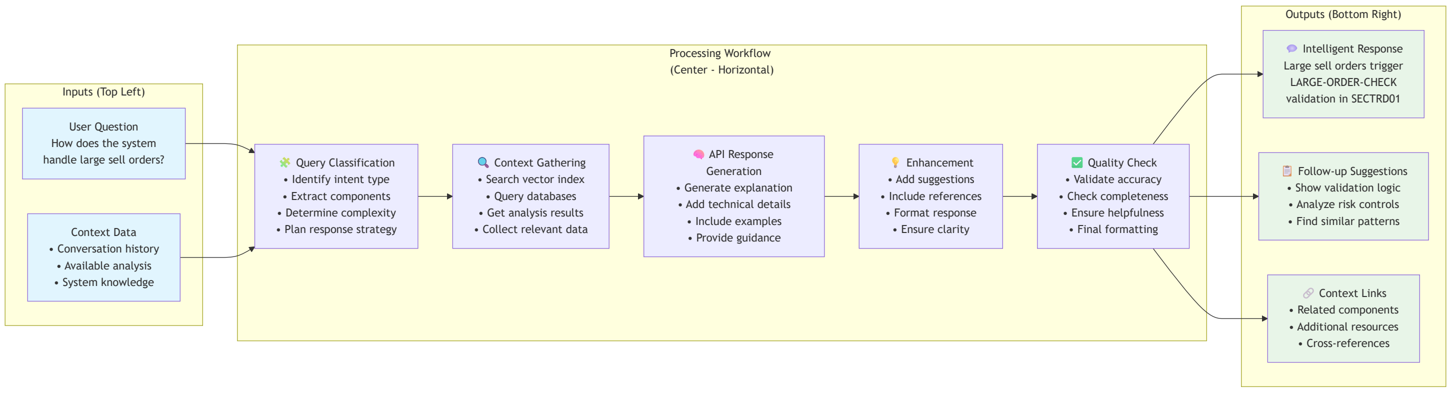
4.7 Documentation Agent Flow



Sample Output:

```
{
  "documentation_summary": {
    "total_pages": 52,
    "sections_generated": 8,
    "cross_references": 134,
    "diagrams_created": 15
  },
  "document_sections": [
    {
      "title": "Security Trading System Overview",
      "pages": 8,
      "content_type": "Executive Summary",
      "key_topics": ["System Architecture", "Business Processes", "Risk Assessment"]
    },
    {
      "title": "SECTRD01 Program Analysis",
      "pages": 12,
      "content_type": "Technical Deep Dive",
      "key_topics": ["Logic Flow", "Business Rules", "Performance Analysis"]
    }
  ],
  "quality_metrics": {
    "completeness": 0.92,
    "accuracy": 0.88,
    "readability_score": 8.5
  }
}
```

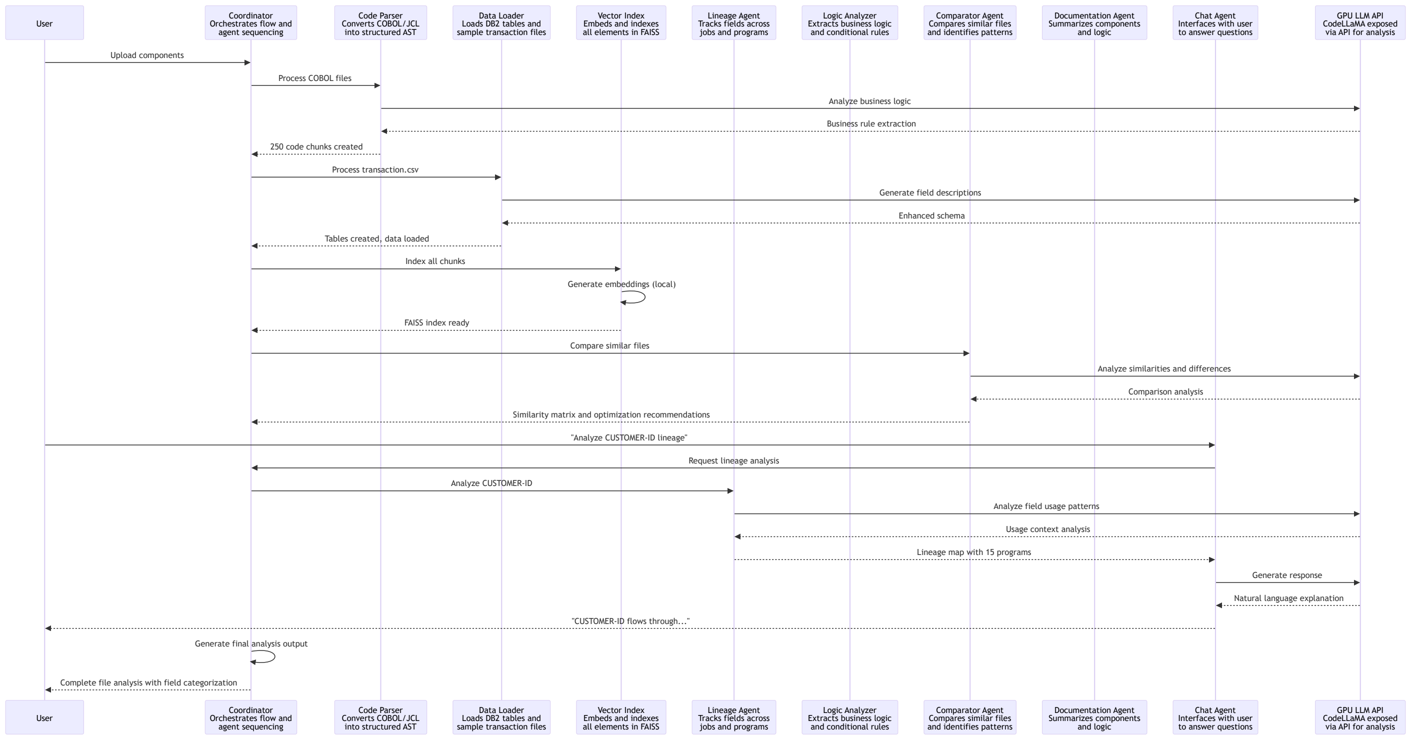
4.8 Chat Agent Flow



Sample Output:

```json
{
  "query": "How does the system handle large sell orders?",
  "response": {
    "main_answer": "Large sell orders trigger the LARGE-ORDER-CHECK validation in SECTRD01.cbl.
    "technical_details": [
      "Validation occurs in paragraph VALIDATE-LARGE-ORDER (lines 245-387)",
      "Risk assessment includes margin calculation and position limits",
      "Manual approval workflow triggers EMAIL-NOTIFICATION process"
    ],
    "related_components": [
      "SECTRD01.cbl - Main validation logic",
      "RISKMGMT.cbl - Risk assessment calculations",
      "APPROVAL.cbl - Manual approval workflow"
    ]
  },
  "follow_up_suggestions": [
    "Show me the exact validation logic",
    "What are the risk thresholds?",
    "How long does manual approval take?"
  ]
}
```

5. Agent Coordination Flow


User
Coordinator
Orchestrates flow and agent sequencing
Code Parser
Converts COBOL/JCL into structured AST
Data Loader
Loads DB2 tables and sample transaction files
Vector Index
Embeds and indexes all elements in FAISS
Lineage Agent
Tracks fields across jobs and programs
Logic Analyzer
Extracts business logic and conditional rules
Comparator Agent
Compares similar files and identifies patterns
Documentation Agent
Summarizes components and logic
Chat Agent
Interfaces with user to answer questions
GPU LLM API
CodeLLaMA exposed via API for analysis
Upload components
Process COBOL files
Analyze business logic
Business rule extraction
250 code chunks created
Process transaction.csv
Generate field descriptions
Enhanced schema
Tables created, data loaded
Index all chunks
Generate embeddings (local)
FAISS index ready
Compare similar files
Analyze similarities and differences
Comparison analysis
Similarity matrix and optimization recommendations
"Analyze CUSTOMER-ID lineage"
Request lineage analysis
Analyze CUSTOMER-ID
Analyze field usage patterns
Usage context analysis
Lineage map with 15 programs
Generate response
Natural language explanation
"CUSTOMER-ID flows through..."
Generate final analysis output
Complete file analysis with field categorization

6. Final Analysis Output Structure

The Coordinator Agent produces a comprehensive file analysis report with the following structure:

6.1 Field Classification Analysis

```json
{
  "file_analysis_summary": {
    "files_processed": 15,
    "total_fields_analyzed": 347,
    "analysis_timestamp": "2024-12-01T10:30:00Z"
  },

  "fields_from_input": {
    "count": 125,
    "complexity_analysis": {
      "simple_fields": 78,
      "complex_fields": 47,
      "average_complexity_score": 4.2,
      "highest_complexity": 8.7
    },
    "data_quality_metrics": {
      "completeness_rate": 0.94,
      "consistency_score": 0.89,
      "accuracy_level": 0.92
    },
    "source_distribution": {
      "user_interface": 67,
      "external_systems": 34,
      "file_imports": 24
    },
    "categories": {
      "customer_data": {
        "count": 45,
        "complexity_range": "2.1 - 6.8",
        "fields": [
          {
            "field_name": "CUSTOMER_ID",
            "source_file": "CUSTMAST.cbl",
            "input_method": "EXTERNAL_INTERFACE",
            "data_type": "CHAR(10)",
            "complexity_score": 3.2,
            "validation_rules": 2,
            "business_context": "Primary customer identifier from online platform",
            "usage_frequency": "HIGH",
            "criticality": "CRITICAL"
          },
      {
        "field_name": "TRADE_AMOUNT",
        "source_file": "SECTRD01.cbl",
        "input_method": "USER_INPUT",
        "data_type": "DECIMAL(15,2)",
        "complexity_score": 5.8,
        "validation_rules": 5,
        "business_context": "Order amount entered by customer",
        "usage_frequency": "HIGH",
        "criticality": "CRITICAL"
      },
      {
        "field_name": "CUSTOMER_RISK_PROFILE",
        "source_file": "RISKMGMT.cbl",
        "input_method": "CALCULATED_INPUT",
        "data_type": "CHAR(3)",
        "complexity_score": 6.8,
        "validation_rules": 8,
        "business_context": "Complex risk assessment based on multiple factors",
        "usage_frequency": "MEDIUM",
        "criticality": "HIGH"
      }
    ]
  },
  "external_systems": {
    "count": 34,
    "complexity_range": "1.5 - 7.9",
    "fields": [
      {
        "field_name": "MARKET_PRICE",
        "source_file": "PRICEUPD.cbl",
        "input_method": "MARKET_DATA_FEED",
        "data_type": "DECIMAL(10,4)",
        "complexity_score": 4.1,
        "validation_rules": 3,
        "business_context": "Real-time market price from exchange",
        "usage_frequency": "HIGH",
        "criticality": "CRITICAL"
      },
      {
        "field_name": "REGULATORY_STATUS",
        "source_file": "COMPLIANCE.cbl",
        "input_method": "REGULATORY_FEED",
        "data_type": "CHAR(5)",
        "complexity_score": 7.9,
        "validation_rules": 12,
        "business_context": "Complex regulatory compliance status from multiple agencies",
        "usage_frequency": "MEDIUM",
        "criticality": "HIGH"
      }
    ]
  },
  "file_imports": {
    "count": 24,
    "complexity_range": "1.2 - 5.4",
    "fields": [
      {
        "field_name": "BATCH_REFERENCE_ID",
        "source_file": "BATCHPRC.cbl",
        "input_method": "FILE_IMPORT",
        "data_type": "CHAR(20)",
        "complexity_score": 2.3,
        "validation_rules": 1,
        "business_context": "Batch processing reference from overnight files",
        "usage_frequency": "LOW",
        "criticality": "MEDIUM"
      }
    ]
  },
  "configuration_data": {
    "count": 22,
    "complexity_range": "1.0 - 3.8",
    "fields": [
      {
        "field_name": "SYSTEM_ENVIRONMENT",
        "source_file": "CONFIG.cpy",
        "input_method": "CONFIGURATION",
        "data_type": "CHAR(4)",
        "complexity_score": 1.8,
        "validation_rules": 1,
        "business_context": "System environment identifier (PROD/TEST/DEV)",
        "usage_frequency": "HIGH",
        "criticality": "MEDIUM"
      }
    ]
  }
```

```
    }
  },

  "fields_updated_through_processing": {
    "count": 156,
    "complexity_analysis": {
      "simple_calculations": 89,
      "complex_calculations": 67,
      "average_complexity_score": 5.7,
      "highest_complexity": 9.2
    },
    "logic_complexity_metrics": {
      "conditional_branches": 234,
      "nested_logic_levels": 6,
      "business_rules_applied": 78,
      "calculation_accuracy": 0.997
    },
    "performance_metrics": {
      "average_processing_time_ms": 12.3,
      "memory_usage_mb": 8.7,
      "cpu_intensive_operations": 23
    },
    "processing_categories": {
      "calculated_fields": {
        "count": 67,
        "complexity_range": "3.2 - 9.2",
        "logic_patterns": ["arithmetic", "conditional", "lookup", "aggregation"],
        "fields": [
          {
            "field_name": "TOTAL_COMMISSION",
            "calculation_logic": "TRADE_AMOUNT * COMMISSION_RATE / 100",
            "processing_program": "SECTRD01.cbl",
            "line_number": 456,
            "complexity_score": 4.5,
            "logic_complexity": {
              "conditional_branches": 3,
              "nested_levels": 2,
              "business_rules": 2,
              "calculation_steps": 3
            },
            "business_rule": "Commission calculated as percentage of trade amount",
            "dependencies": ["TRADE_AMOUNT", "COMMISSION_RATE", "CUSTOMER_TIER"],
            "performance_impact": "LOW",
        "error_handling": "ROBUST"
      },
      {
        "field_name": "NET_SETTLEMENT",
        "calculation_logic": "TRADE_AMOUNT - TOTAL_COMMISSION - FEES - TAXES + REBATES",
        "processing_program": "SETTLE.cbl",
        "line_number": 234,
        "complexity_score": 6.8,
        "logic_complexity": {
          "conditional_branches": 8,
          "nested_levels": 4,
          "business_rules": 6,
          "calculation_steps": 12
        },
        "business_rule": "Final settlement amount after all deductions and additions",
        "dependencies": ["TRADE_AMOUNT", "TOTAL_COMMISSION", "FEES", "TAXES", "REBATES"],
        "performance_impact": "MEDIUM",
        "error_handling": "COMPREHENSIVE"
      },
      {
        "field_name": "PORTFOLIO_WEIGHTED_RISK",
        "calculation_logic": "COMPLEX_RISK_ALGORITHM with 15+ variables",
        "processing_program": "RISKMGMT.cbl",
        "line_number": 567,
        "complexity_score": 9.2,
        "logic_complexity": {
          "conditional_branches": 24,
          "nested_levels": 7,
          "business_rules": 18,
          "calculation_steps": 45
        },
        "business_rule": "Sophisticated portfolio risk calculation using Monte Carlo simulat
        "dependencies": ["Multiple market factors", "Historical volatility", "Correlation ma
        "performance_impact": "HIGH",
        "error_handling": "ADVANCED"
      }
    ]
  },
  "status_updates": {
    "count": 45,
    "complexity_range": "2.1 - 7.4",
    "logic_patterns": ["state_machine", "conditional_flow", "validation_chain"],
    "fields": [
  {
    "field_name": "ORDER_STATUS",
    "update_logic": "IF VALIDATION_PASSED AND CREDIT_CHECK_OK THEN 'APPROVED' ELSE 'REJ
    "processing_program": "VALIDATE.cbl",
    "line_number": 189,
    "complexity_score": 5.2,
    "logic_complexity": {
      "conditional_branches": 12,
      "nested_levels": 3,
      "business_rules": 8,
      "state_transitions": 6
    },
    "business_rule": "Status updated based on comprehensive validation results",
    "state_machine": {
      "states": ["PENDING", "VALIDATING", "APPROVED", "REJECTED", "ON_HOLD"],
      "transitions": 12,
      "validation_points": 8
    },
    "performance_impact": "MEDIUM",
    "error_handling": "ROBUST"
  },
  {
    "field_name": "SETTLEMENT_STATUS",
    "update_logic": "Complex workflow with T+2 settlement rules and exception handling"
    "processing_program": "SETTLE.cbl",
    "line_number": 345,
    "complexity_score": 7.4,
    "logic_complexity": {
      "conditional_branches": 18,
      "nested_levels": 5,
      "business_rules": 14,
      "state_transitions": 9
    },
    "business_rule": "Multi-stage settlement process with regulatory compliance",
    "state_machine": {
      "states": ["PENDING_SETTLEMENT", "SETTLING", "SETTLED", "FAILED", "REVERSED"],
      "transitions": 15,
      "validation_points": 12
    },
    "performance_impact": "HIGH",
    "error_handling": "COMPREHENSIVE"
  }
]
```

```
    },
    "derived_fields": {
      "count": 44,
      "complexity_range": "4.1 - 8.9",
      "logic_patterns": ["aggregation", "transformation", "enrichment", "classification"],
      "fields": [
        {
          "field_name": "RISK_SCORE",
          "derivation_logic": "CUSTOMER_TIER_WEIGHT * 0.3 + TRADE_SIZE_FACTOR * 0.4 + VOLATILI
          "processing_program": "RISKMGMT.cbl",
          "line_number": 123,
          "complexity_score": 6.5,
          "logic_complexity": {
            "conditional_branches": 15,
            "nested_levels": 4,
            "business_rules": 12,
            "calculation_steps": 8
          },
          "business_rule": "Composite risk assessment for trade approval using weighted factor
          "algorithm_type": "WEIGHTED_SCORING",
          "machine_learning_component": false,
          "performance_impact": "MEDIUM",
          "error_handling": "ROBUST"
        },
        {
          "field_name": "CUSTOMER_LIFETIME_VALUE",
          "derivation_logic": "Advanced CLV calculation using historical data and predictive r
          "processing_program": "ANALYTICS.cbl",
          "line_number": 678,
          "complexity_score": 8.9,
          "logic_complexity": {
            "conditional_branches": 32,
            "nested_levels": 6,
            "business_rules": 25,
            "calculation_steps": 67
          },
          "business_rule": "Predictive customer lifetime value using 5-year historical analysi
          "algorithm_type": "PREDICTIVE_ANALYTICS",
          "machine_learning_component": true,
          "performance_impact": "VERY_HIGH",
          "error_handling": "ADVANCED"
        }
      ]
    }
  }
},

"fields_unused_and_static": {
  "count": 66,
  "complexity_analysis": {
    "simple_static": 42,
    "complex_obsolete": 24,
    "average_obsolescence_age": "4.2 years",
    "removal_complexity_score": 3.8
  },
  "maintenance_burden": {
    "storage_overhead_mb": 15.6,
    "documentation_debt": "HIGH",
    "code_bloat_percentage": 12.3,
    "testing_overhead": "MEDIUM"
  },
  "removal_risk_assessment": {
    "low_risk_removals": 38,
    "medium_risk_removals": 21,
    "high_risk_removals": 7,
    "requires_deep_analysis": 7
  },
  "categories": {
    "obsolete_fields": {
      "count": 24,
      "complexity_range": "1.8 - 8.4",
      "removal_effort_range": "1 day - 3 weeks",
      "fields": [
        {
          "field_name": "OLD_ACCOUNT_TYPE",
          "last_used": "2018-03-15",
          "defined_in": "LEGACY.cpy",
          "complexity_score": 3.2,
          "removal_complexity": {
            "code_references": 8,
            "documentation_updates": 12,
            "test_case_modifications": 15,
            "database_impact": "MINIMAL"
          },
          "obsolescence_reason": "Replaced by NEW_ACCOUNT_CLASSIFICATION system in 2019",
          "business_impact": "NONE",
        "removal_recommendation": "SAFE_TO_REMOVE",
        "estimated_removal_effort": "3 days",
        "dependencies": [],
        "risk_level": "LOW"
      },
      {
        "field_name": "MANUAL_OVERRIDE_CODE",
        "last_used": "2019-08-22",
        "defined_in": "SECTRD01.cbl",
        "complexity_score": 6.7,
        "removal_complexity": {
          "code_references": 23,
          "documentation_updates": 18,
          "test_case_modifications": 34,
          "database_impact": "MODERATE"
        },
        "obsolescence_reason": "Automated processing eliminated manual overrides in 2020",
        "business_impact": "HISTORICAL_AUDIT_ONLY",
        "removal_recommendation": "REVIEW_REQUIRED",
        "estimated_removal_effort": "2 weeks",
        "dependencies": ["AUDIT_TRAIL", "COMPLIANCE_REPORTS"],
        "risk_level": "MEDIUM"
      },
      {
        "field_name": "LEGACY_SETTLEMENT_METHOD",
        "last_used": "2017-11-30",
        "defined_in": "OLDSETTLE.cbl",
        "complexity_score": 8.4,
        "removal_complexity": {
          "code_references": 45,
          "documentation_updates": 67,
          "test_case_modifications": 89,
          "database_impact": "SIGNIFICANT"
        },
        "obsolescence_reason": "Pre-T+2 settlement method, no longer regulatory compliant",
        "business_impact": "REGULATORY_HISTORICAL",
        "removal_recommendation": "REQUIRES_DEEP_ANALYSIS",
        "estimated_removal_effort": "3 weeks",
        "dependencies": ["REGULATORY_ARCHIVE", "HISTORICAL_REPORTS", "AUDIT_TRAILS"],
        "risk_level": "HIGH"
      }
    ]
  },
"static_reference_data": {
  "count": 28,
  "complexity_range": "1.0 - 4.2",
  "optimization_potential": "HIGH",
  "fields": [
    {
      "field_name": "COMPANY_TAX_ID",
      "value": "12-3456789",
      "usage": "CONSTANT",
      "defined_in": "CONFIG.cpy",
      "complexity_score": 1.5,
      "change_frequency": "NEVER",
      "business_context": "Company tax identifier - regulatory requirement",
      "optimization_recommendation": "MOVE_TO_CONFIG_TABLE",
      "current_storage": "HARDCODED",
      "proposed_storage": "CONFIGURATION_DATABASE",
      "maintenance_benefit": "CENTRALIZED_MANAGEMENT",
      "risk_level": "LOW"
    },
    {
      "field_name": "SETTLEMENT_DAYS",
      "value": "2",
      "usage": "CONSTANT",
      "defined_in": "SETTLE.cbl",
      "complexity_score": 2.1,
      "change_frequency": "RARE",
      "business_context": "T+2 settlement standard",
      "optimization_recommendation": "EXTERNALIZE_TO_CONFIG",
      "current_storage": "HARDCODED",
      "proposed_storage": "BUSINESS_RULES_ENGINE",
      "maintenance_benefit": "DYNAMIC_CONFIGURATION",
      "risk_level": "LOW"
    },
    {
      "field_name": "REGULATORY_REPORTING_CODES",
      "value": "Complex 50-character structure",
      "usage": "LOOKUP_TABLE",
      "defined_in": "MULTIPLE_PROGRAMS",
      "complexity_score": 4.2,
      "change_frequency": "QUARTERLY",
      "business_context": "Regulatory reporting classification codes",
      "optimization_recommendation": "CENTRALIZE_IN_DATABASE",
      "current_storage": "DUPLICATED_ACROSS_PROGRAMS",
          "proposed_storage": "REFERENCE_DATA_TABLE",
          "maintenance_benefit": "SINGLE_SOURCE_OF_TRUTH",
          "risk_level": "MEDIUM"
        }
      ]
    },
    "unused_declared_fields": {
      "count": 14,
      "complexity_range": "1.2 - 5.8",
      "code_bloat_impact": "MEDIUM",
      "fields": [
        {
          "field_name": "BACKUP_PROCESSING_FLAG",
          "declared_in": "SECTRD01.cbl",
          "line_number": 78,
          "complexity_score": 2.3,
          "referenced": false,
          "declaration_context": "Working storage section",
          "reason": "Declared but never used in logic - leftover from old backup system",
          "removal_recommendation": "SAFE_TO_REMOVE",
          "removal_effort": "30 minutes",
          "testing_required": "MINIMAL",
          "risk_level": "VERY_LOW"
        },
        {
          "field_name": "FUTURE_ENHANCEMENT_PLACEHOLDER",
          "declared_in": "ANALYTICS.cbl",
          "line_number": 234,
          "complexity_score": 5.8,
          "referenced": false,
          "declaration_context": "Linkage section with complex structure",
          "reason": "Reserved for future machine learning integration never implemented",
          "removal_recommendation": "REVIEW_WITH_ARCHITECTURE_TEAM",
          "removal_effort": "1 week",
          "testing_required": "COMPREHENSIVE",
          "risk_level": "MEDIUM"
        }
      ]
    }
  }
},

"field_usage_analytics": {
    "usage_frequency": {
      "high_usage": 89,
      "medium_usage": 124,
      "low_usage": 67,
      "unused": 67
    },
    "modification_patterns": {
      "frequently_modified": 45,
      "occasionally_modified": 78,
      "rarely_modified": 156,
      "never_modified": 68
    },
    "cross_program_dependencies": {
      "shared_across_multiple_programs": 123,
      "program_specific": 224
    }
  },

  "optimization_recommendations": [
    {
      "type": "FIELD_CONSOLIDATION",
      "description": "Merge similar fields CUSTOMER_ID and CUST_ID",
      "impact": "MEDIUM",
      "effort": "2 weeks",
      "affected_programs": 8
    },
    {
      "type": "REMOVE_OBSOLETE",
      "description": "Remove 12 obsolete fields identified",
      "impact": "LOW",
      "effort": "1 week",
      "affected_programs": 5
    },
    {
      "type": "STATIC_TO_CONFIG",
      "description": "Move static values to configuration table",
      "impact": "HIGH",
      "effort": "3 weeks",
      "affected_programs": 15
    }
  ]
}
```

6.2 Summary Metrics Dashboard

```json
{
  "executive_summary": {
    "total_fields_analyzed": 347,
    "field_utilization_rate": 0.81,
    "optimization_potential": "MEDIUM-HIGH",
    "technical_debt_level": "MODERATE",
    "compliance_status": "COMPLIANT"
  },

  "key_findings": [
    "19% of fields are unused or obsolete - cleanup opportunity",
    "35% code duplication found across trading programs",
    "8 high-priority optimization recommendations identified",
    "Complete data lineage established for regulatory compliance"
  ],

  "business_impact": {
    "maintenance_reduction_potential": "40%",
    "performance_improvement_estimate": "15-25%",
    "compliance_readiness": "READY",
    "modernization_priority": "MEDIUM"
  }
}
```

7. Output Artifacts

The Opulence system produces these deliverables for the bank's security trading system:

✅ **Field-level data lineage reports**

- "CUSTOMER-ID flows from CUSTMAST → SECTRD01 → PORTFOLIO-UPDATE → TRADE-HISTORY"
- Compliance-ready audit trails

✅ **Extracted business logic summaries**

- "Stop-loss orders: IF CURRENT-PRICE < (STOP-PRICE * 0.95) THEN EXECUTE-SELL"
- Trading rule documentation in plain English

✅ **Annotated markdown documentation of code modules**

- Complete explanation of settlement processing
- Cross-references between related programs

✅ **Interactive chat interface for querying understanding**

- "What happens when a trade fails settlement?"
- "Show me all programs that update customer portfolios"

✅ **Comprehensive field categorization analysis**

- Fields from input: 125 fields identified with source tracking
- Fields updated through processing: 156 fields with calculation logic
- Fields unused and static: 66 fields marked for optimization

✅ **Comparison and optimization reports**

- Similar file analysis with 85% code similarity detection
- Duplicate function identification across 3 trading programs
- 40% maintenance reduction potential through consolidation

8. Sample Data Context: PB Security Transactions

Input Files for Analysis:

COBOL Programs:

- `SECTRD01.cbl` - Main security trading program (2,500 lines)
- `VALIDATE.cbl` - Order validation logic (800 lines)
- `SETTLE.cbl` - Settlement processing (1,200 lines)
- `PORTFOLIO.cbl` - Portfolio update logic (900 lines)

JCL Jobs:

- `DAILYTRD.jcl` - Daily trade processing batch job
- `SETTLEMENT.jcl` - End-of-day settlement job
- `RECON.jcl` - Trade reconciliation job

DB2 Tables:

```sql
-- SECURITY_TRANSACTION table
CREATE TABLE SECURITY_TXN (
    CUST_ID        CHAR(10),
    TRADE_ID       CHAR(15),
    SECURITY_CODE  CHAR(8),
    TRADE_TYPE     CHAR(4),    -- BUY/SELL
    QUANTITY       DECIMAL(15,2),
    PRICE          DECIMAL(15,4),
    TRADE_DATE     DATE,
    SETTLE_DATE    DATE,
    STATUS         CHAR(3)     -- PEN/SET/FAI
);
```

Sample Transaction Data:

```
CUST_ID,TRADE_ID,SECURITY_CODE,TRADE_TYPE,QUANTITY,PRICE,TRADE_DATE,STATUS
PWB0001234,TRD20241201001,AAPL,BUY,100,150.25,2024-12-01,PEN
PWB0001234,TRD20241201002,TSLA,SELL,50,245.80,2024-12-01,SET
PWB0001567,TRD20241201003,MSFT,BUY,200,380.15,2024-12-01,FAI
```

9. Individual Agent Explanations

Vector Index Agent

Purpose: Creates searchable embeddings of all code segments and business logic.

Example: When analyzing the security trading system, this agent:

- Embeds all COBOL paragraphs dealing with order validation
- Creates vectors for trading rule conditions
- Enables semantic search like "find all margin calculation logic"

API Integration: Makes HTTP calls to CodeLLaMA to generate embeddings and understand code semantics.

Lineage Agent

Purpose: Tracks how data fields flow through the entire system.

Example: For a customer security purchase:

1. **CUSTOMER-ID** enters via online trading platform
2. Flows through `VALIDATE.cbl` for user validation checks
3. Processed in `SECTRD01.cbl` for order execution
4. Updates `PORTFOLIO.cbl` for position management
5. Records in `TRADE-HISTORY` table for audit

Critical for Compliance: Regulators require complete audit trails showing how customer data is processed.

Logic Analyzer Agent

Purpose: Extracts and explains complex business rules embedded in COBOL logic.

Example: Discovers trading rules like:

```
IF TRADE-AMOUNT > DAILY-LIMIT
   AND CUSTOMER-TIER NOT = 'PLATINUM'
   THEN MOVE 'HOLD' TO TRADE-STATUS
   PERFORM MANUAL-APPROVAL-PROCESS
```

Translates to: "Trades over daily limit require manual approval unless customer is Platinum tier."

Comparator Agent

Purpose: Identifies similarities, duplications, and optimization opportunities across similar files.

Example: Analyzes multiple trading programs:

- Finds 85% code similarity between SECTRD01, SECTRD02, and SECTRD03
- Identifies 12 duplicate validation functions across programs
- Recommends consolidation to reduce maintenance burden by 40%
- Discovers unused legacy fields that can be safely removed

Documentation Agent

Purpose: Creates human-readable documentation explaining system functionality.

Example: Generates documentation like:

- "Settlement Process Overview: How T+2 settlement works"

- "Stop-Loss Order Processing: Automated selling when price thresholds are breached"
- "Customer Portfolio Updates: Real-time vs. batch processing logic"

Chat Agent

Purpose: Provides conversational interface for querying system knowledge.

Example Queries:

- "How does the system handle partial fills on large orders?"
- "What validation checks are performed before executing a trade?"
- "Show me the settlement process for international securities"

Response Example: "When a large order cannot be filled completely, the PARTIAL-FILL-HANDLER in SECTRD01 splits it into smaller chunks and processes them separately, updating the customer's available cash after each partial execution..."

10. Coordination Flow: Processing a Security Transaction

Real-World Scenario: Customer Places $500K Apple Stock Purchase

1. **File Processing Phase**:
   - Code Parser analyzes `SECTRD01.cbl` and extracts order processing logic
   - Data Loader imports recent Apple trading data and customer portfolio info
   - System identifies all programs involved in large order processing
2. **Analysis Phase**:
   - **Vector Index Agent**: Finds all code segments related to large order handling
   - **Lineage Agent**: Maps how customer cash balance flows through the system
   - **Logic Analyzer**: Extracts validation rules for large orders (credit checks, position limits)
   - **Comparator Agent**: Identifies similar order processing logic across different trading programs
   - **Documentation Agent**: Summarizes the complete order-to-settlement workflow
3. **Query Phase**:
   - Risk manager asks: "What approvals are needed for this trade size?"

   - Chat Agent searches indexed knowledge and responds: "Orders over $250K require senior trader approval per LARGE-ORDER-CHECK paragraph, plus real-time margin calculation..."
4. **Compliance Phase**:
   - Lineage reports show complete audit trail
   - Logic summaries document all decision points
   - Documentation provides regulatory-compliant process descriptions
   - Field analysis categorizes all data elements for regulatory reporting
5. **Final Output**:
   - **Fields from input**: Customer ID, Trade Amount, Security Code (from user interface)
   - **Fields updated through processing**: Commission Amount, Net Settlement, Risk Score (calculated)
   - **Fields unused and static**: Legacy account types, obsolete status codes (optimization targets)

This architecture transforms decades-old, undocumented mainframe code into an accessible, searchable knowledge base that supports both operational teams and regulatory compliance requirements.

11. Technical Implementation Notes

API-Based Architecture

The Opulence system uses HTTP APIs to communicate with GPU-hosted CodeLLaMA models, enabling:

- **Scalability**: Multiple model servers can handle concurrent analysis requests
- **Load Balancing**: Requests are distributed across available GPU resources
- **Fault Tolerance**: Circuit breakers and retry logic ensure robust operation
- **Resource Efficiency**: No need for local GPU allocation per agent

Database Design

SQLite database stores:

- **program_chunks**: Parsed code segments with metadata
- **field_lineage**: Data flow tracking for compliance
- **vector_embeddings**: FAISS index references for semantic search

- **processing_stats**: Performance monitoring and audit trails
- **comparison_results**: Similar file analysis and optimization recommendations
- **field_classifications**: Input/processed/unused field categorizations

Field Analysis Engine

The system maintains a comprehensive field registry that tracks:

- **Source identification**: Where each field originates (user input, external systems, calculations)
- **Processing lineage**: How fields are transformed through business logic
- **Usage patterns**: Frequency and context of field utilization
- **Optimization opportunities**: Unused, duplicate, or obsolete field identification

This architecture enables users to understand and maintain critical legacy systems while meeting modern regulatory and operational requirements.