AI Impact Assessment System - Backend Strategy Document

Version: 1.0
Date: December 31, 2025
Author: Architecture Team

Table of Contents

---

1. Executive Summary

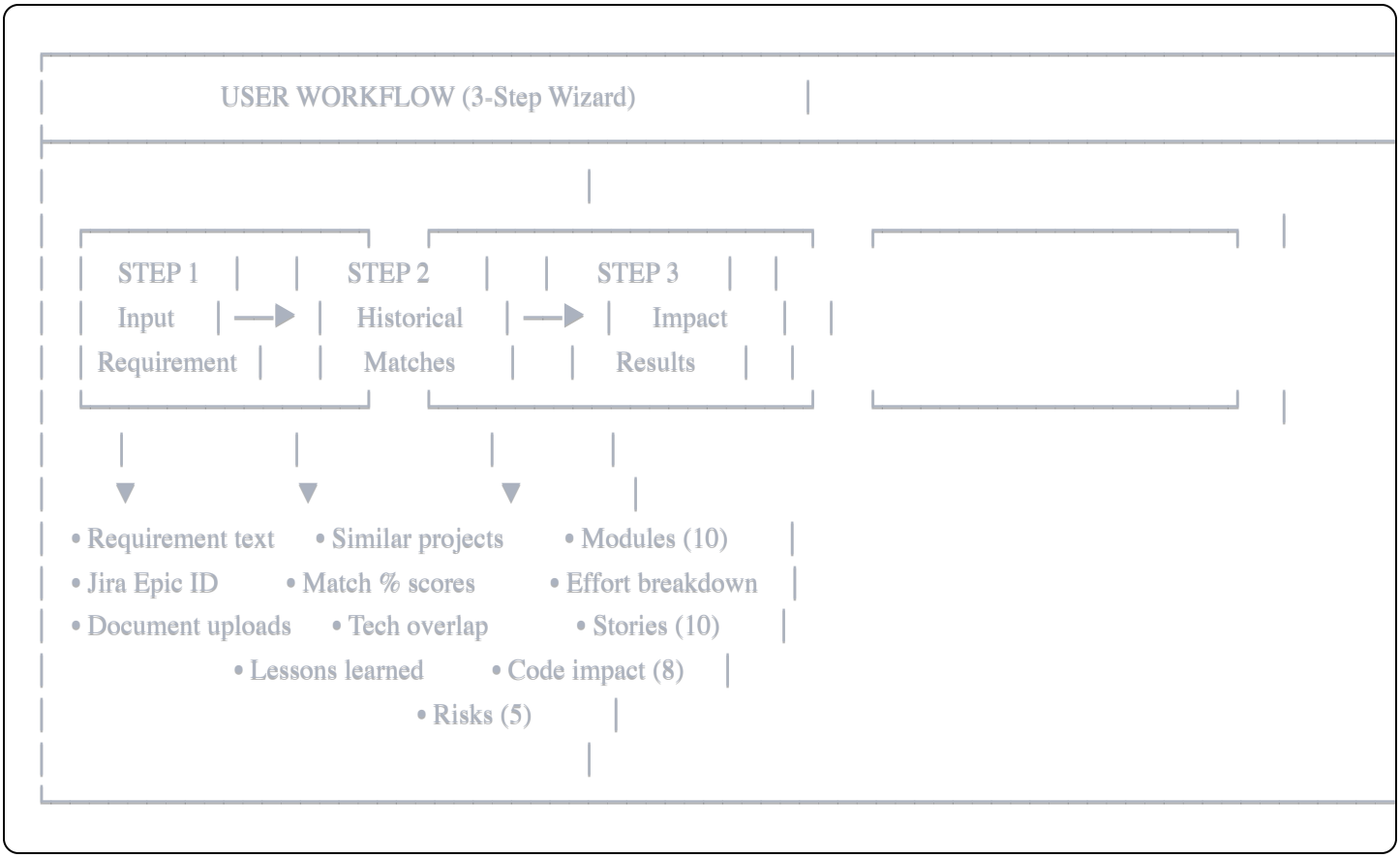
1.1 Purpose

This document outlines the backend strategy for an AI-powered Impact Assessment System that analyzes incoming project requirements and identifies impacted functional/technical modules to enable effective project planning and effort estimation.

1.2 Key Design Decisions

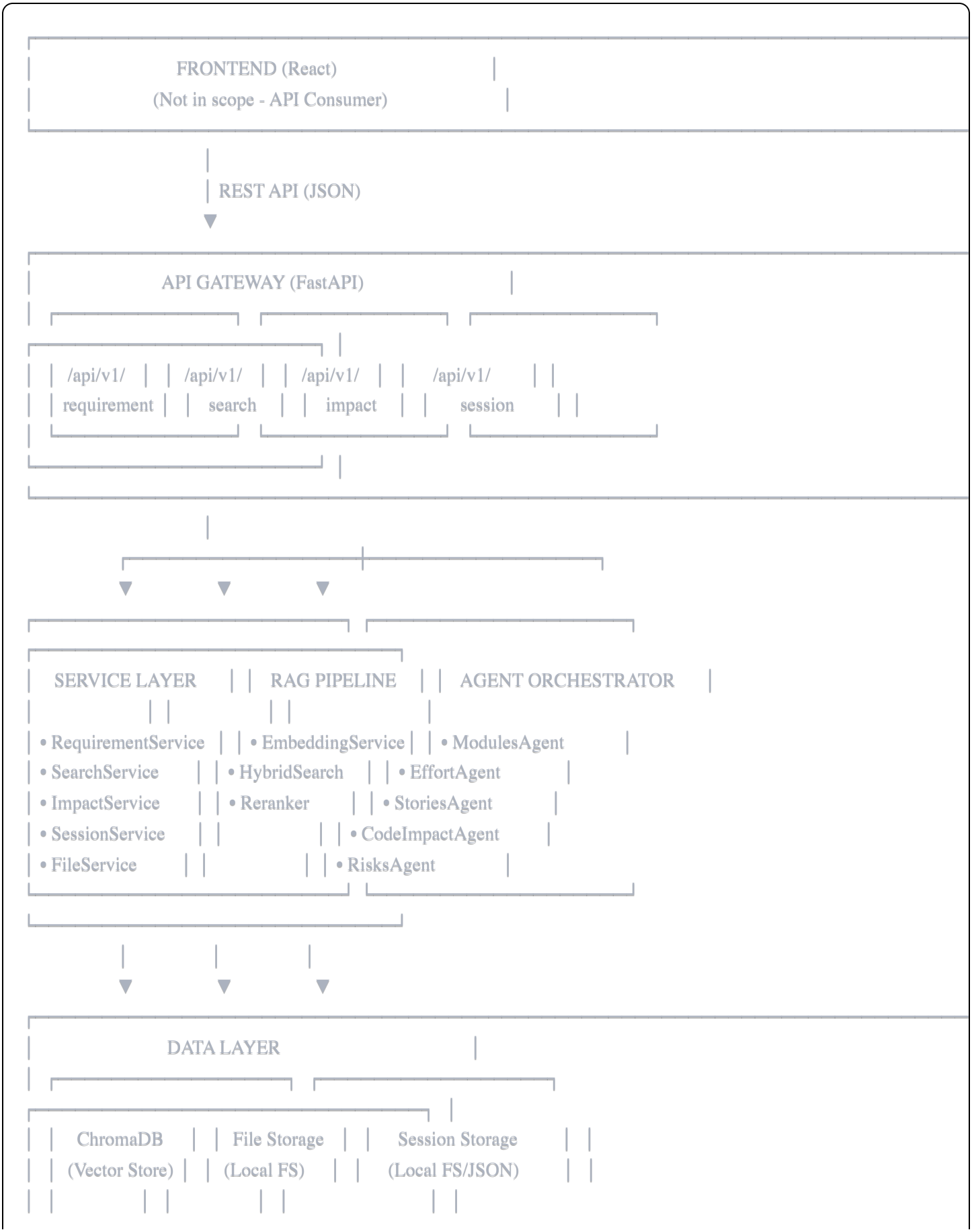
Decision	Choice	Rationale
Backend Framework	FastAPI	Modern, async, auto-generated OpenAPI docs
Vector Database	ChromaDB	Lightweight, local-first, good for POC scale
LLM Runtime	Ollama (local)	Privacy, no API costs, offline capability
LLM Model	phi3:mini	Lightweight, good structured output
Embedding Model	all-minilm	Very lightweight, sufficient for POC
Search Strategy	Hybrid (configurable)	Semantic + keyword matching

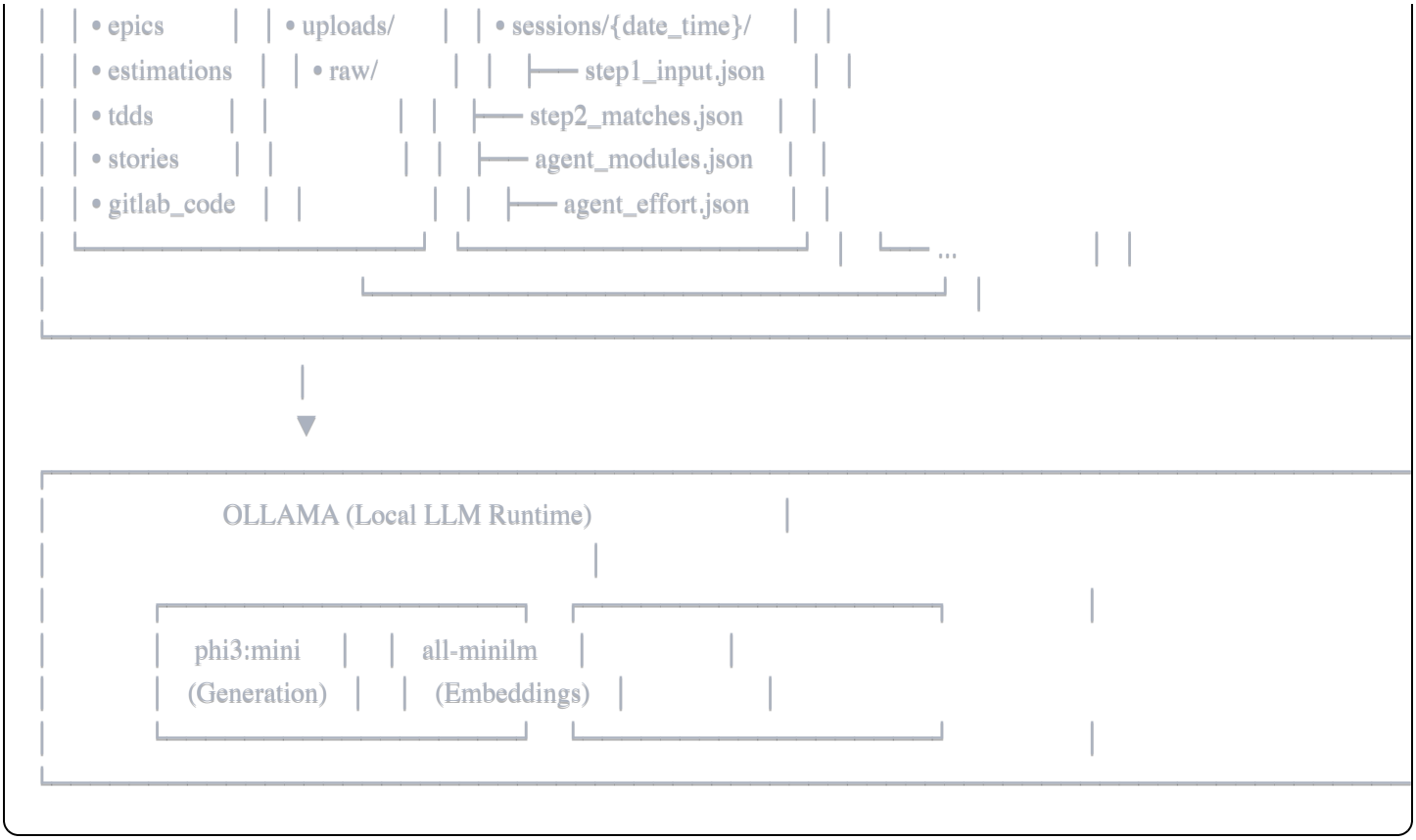
1.3 Core Workflow



2. System Architecture Overview

2.1 High-Level Architecture





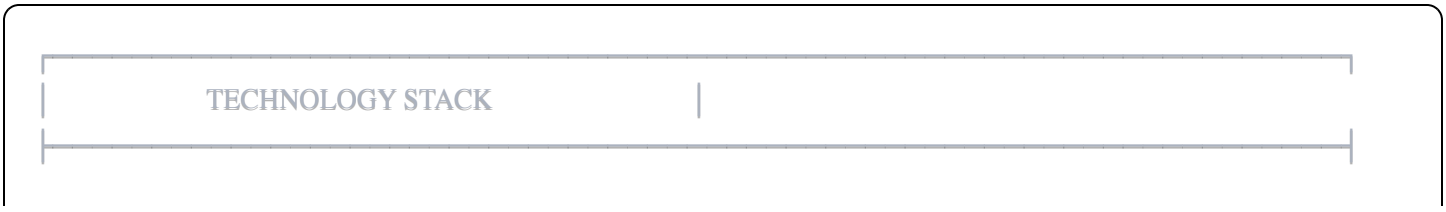
2.2 Component Responsibilities

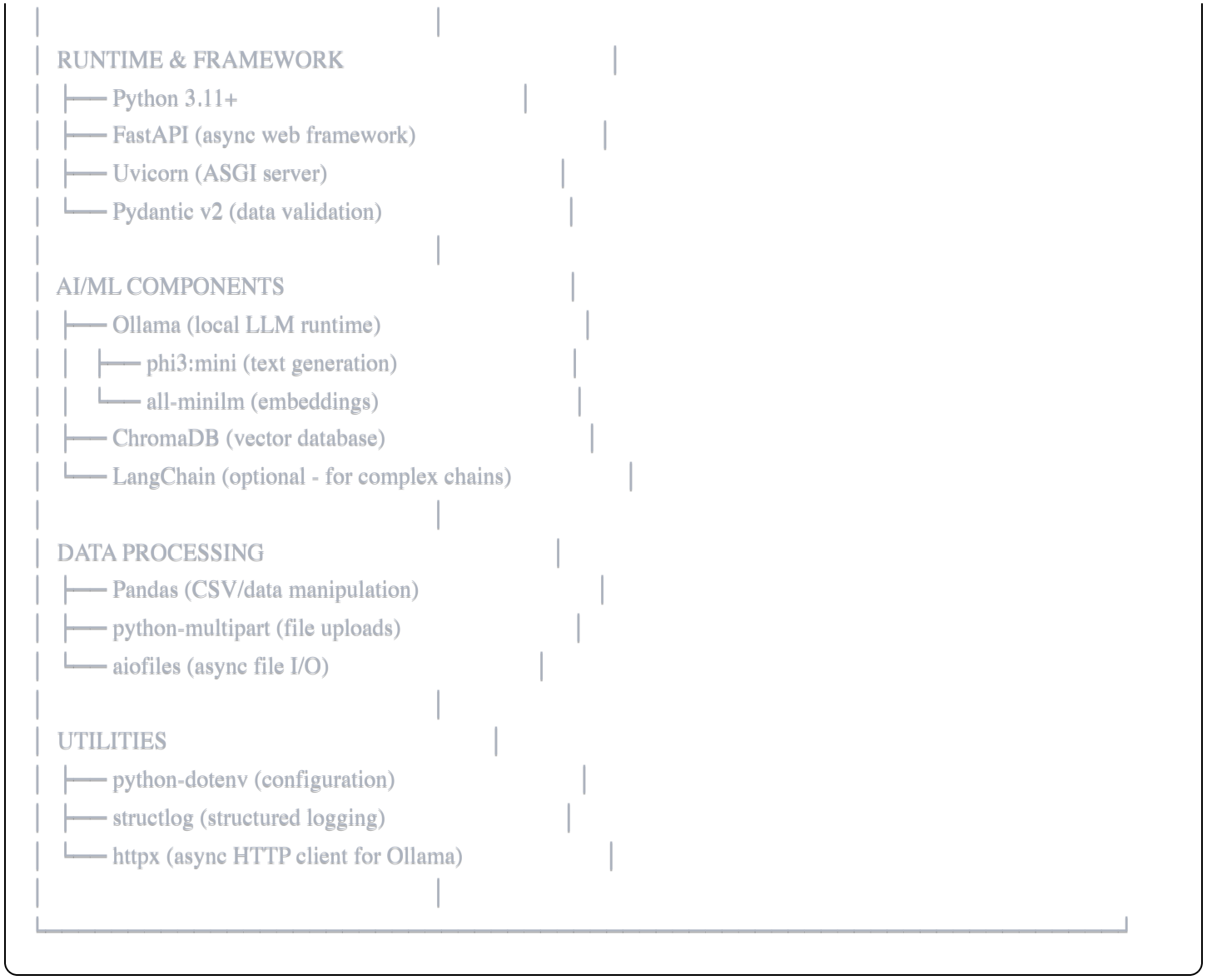
Component	Responsibility
API Gateway	Request routing, validation, error handling, CORS
Service Layer	Business logic, orchestration, data transformation
RAG Pipeline	Embedding generation, hybrid search, context retrieval
Agent Orchestrator	Multi-step LLM calls, prompt management, response parsing
Data Layer	Vector storage, file management, session persistence

---

3. Technology Stack

3.1 Core Technologies





3.2 Version Requirements

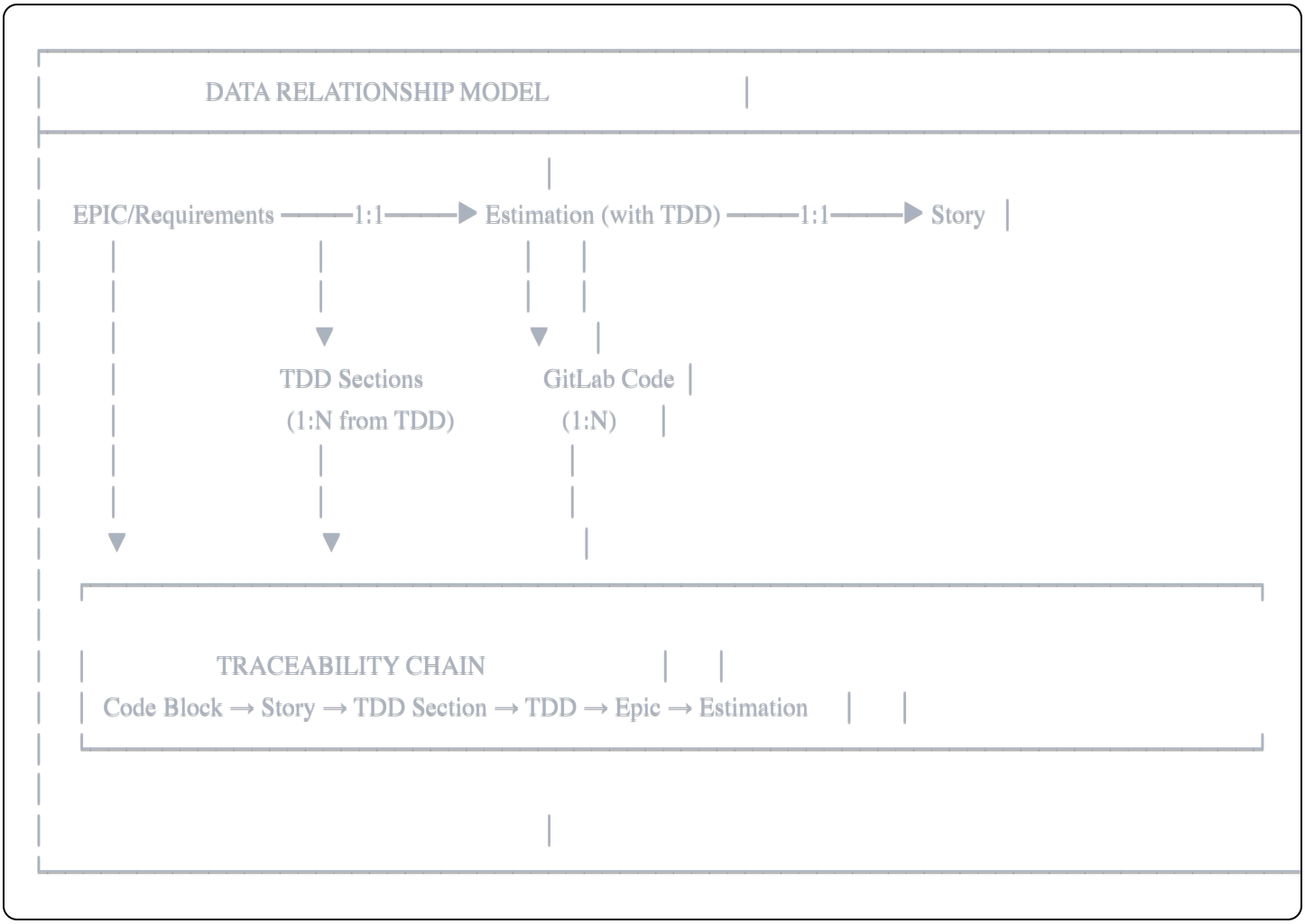
Package	Version	Purpose
python	≥3.11	Runtime
fastapi	≥0.109.0	Web framework
chromadb	≥0.4.22	Vector database
ollama	≥0.1.0	LLM client
pandas	≥2.1.0	Data processing
pydantic	≥2.5.0	Validation

4. Data Architecture

4.1 Source Data Files

```
data/
└── raw/
    ├── epics.csv            # Master project initiatives
    ├── estimations.csv      # Dev/QA effort with TDD embedded
    ├── tdds.csv             # Technical Design Documents
    ├── stories_tasks.csv    # Jira-style work items
    └── gitlab_code.json     # Code references
```

4.2 Data Relationships



4.3 ChromaDB Collections Schema

Collection: `epics`

Document: epic_title + epic_description (concatenated)
Metadata:
- epic_id (string)
- epic_status (string)
- epic_priority (string)
- epic_owner (string)
- epic_team (string)
- epic_start_date (string)
- epic_target_date (string)

Collection: `estimations`

Document: task_description + tdd_description (concatenated)
Metadata:
- estimation_id (string)
- epic_id (string)
- dev_hours (float)
- qa_hours (float)
- total_hours (float)
- complexity (string)
- risk_level (string)
- tdd_technologies (list → JSON string)
- tdd_dependencies (list → JSON string)

Collection: `tdds`

Document: tdd_section_name + tdd_section_content_summary (concatenated)
Metadata:
- tdd_id (string)
- epic_id (string)
- tdd_section_id (string)
- tdd_title (string)
- tdd_version (string)
- tdd_status (string)
- tdd_technologies (list → JSON string)
- tdd_dependencies (list → JSON string)

Collection: `stories`

Document: story_title + acceptance_criteria (concatenated)
Metadata:
- story_id (string)
- epic_id (string)
- tdd_id (string)
- tdd_section_id (string)
- story_type (string)
- story_status (string)
- story_points (int)
- story_assignee (string)
- labels (list → JSON string)

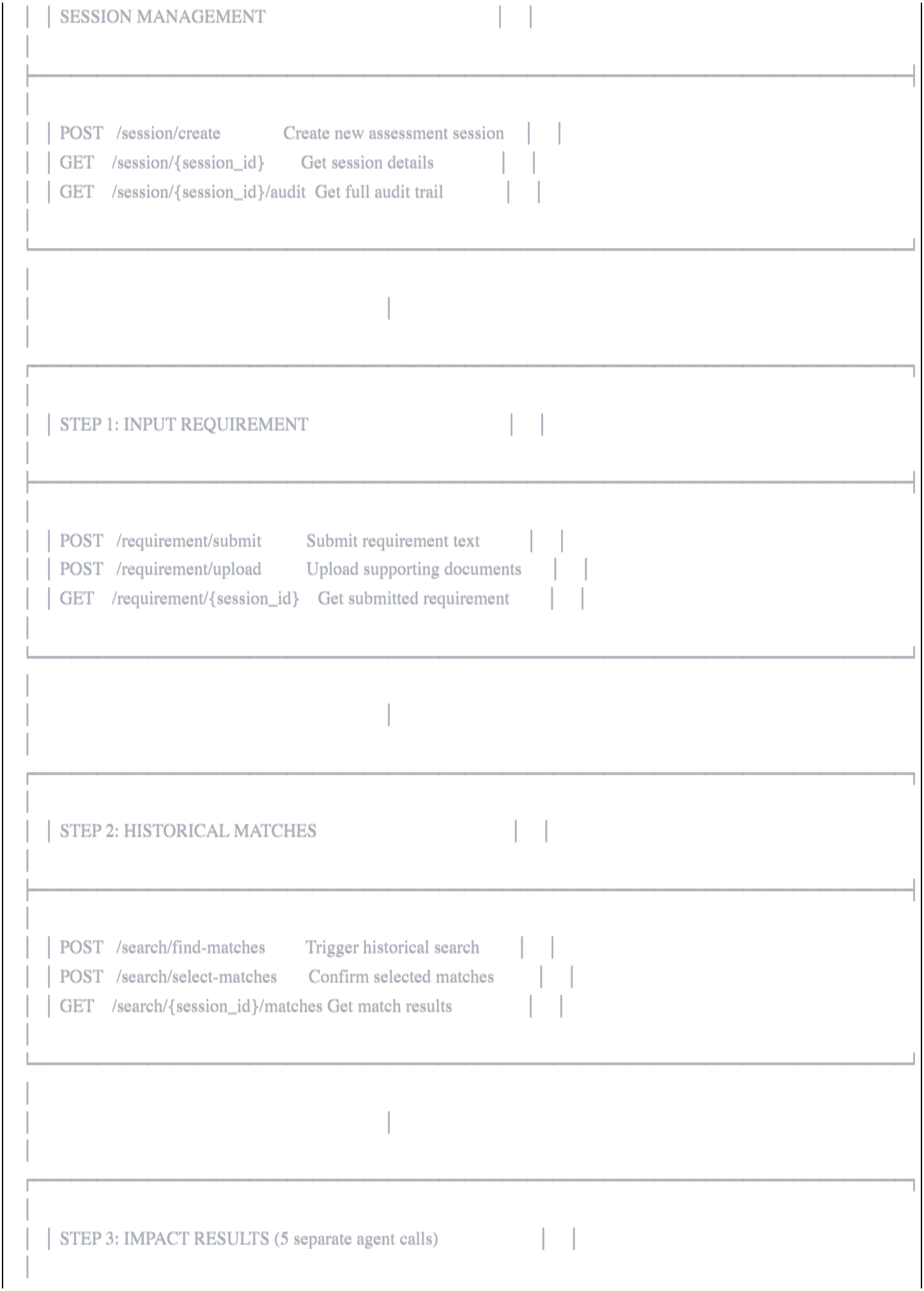
Collection: `gitlab_code`

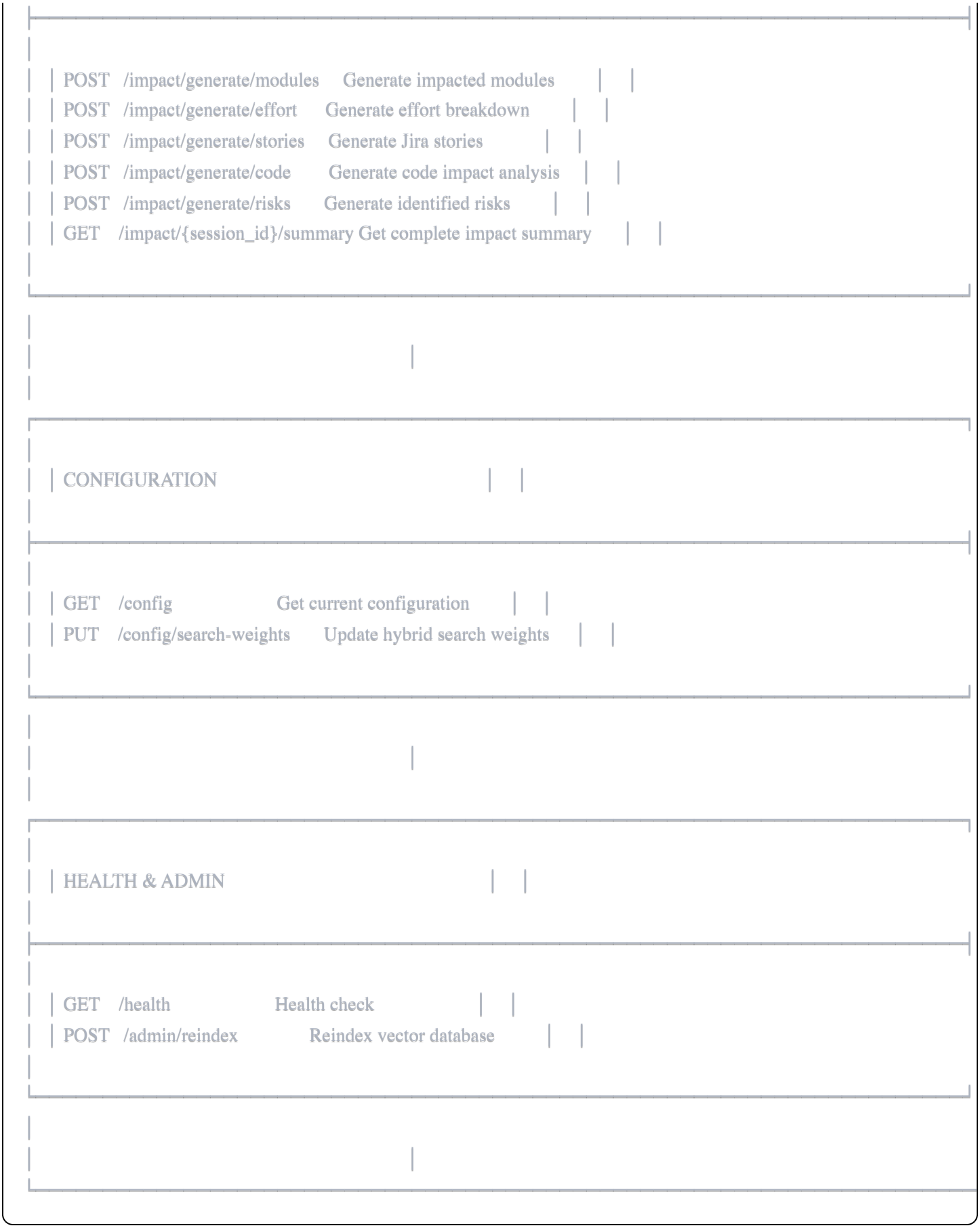
Document: code_block_description + functions_defined (concatenated)
Metadata:
- code_id (string)
- story_id (string)
- epic_id (string)
- tdd_section_id (string)
- gitlab_repo (string)
- gitlab_file_path (string)
- code_language (string)
- classes_defined (list → JSON string)
- functions_defined (list → JSON string)

---

5. API Design

5.1 API Endpoints Overview


SESSION MANAGEMENT
POST /session/create Create new assessment session
GET /session/{session_id} Get session details
GET /session/{session_id}/audit Get full audit trail
STEP 1: INPUT REQUIREMENT
POST /requirement/submit Submit requirement text
POST /requirement/upload Upload supporting documents
GET /requirement/{session_id} Get submitted requirement
STEP 2: HISTORICAL MATCHES
POST /search/find-matches Trigger historical search
POST /search/select-matches Confirm selected matches
GET /search/{session_id}/matches Get match results
STEP 3: IMPACT RESULTS (5 separate agent calls)



5.2 Detailed API Specifications

5.2.1 Session Management

POST /api/v1/session/create

```json
// Request
{
  "user_id": "optional-user-identifier"
}

// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "created_at": "2025-12-31T14:30:52Z",
  "status": "created",
  "audit_path": "sessions/2025-12-31/sess_20251231_143052_abc123/"
}
```

5.2.2 Step 1: Input Requirement

POST /api/v1/requirement/submit

```json
// Request
{
  "session_id": "sess_20251231_143052_abc123",
  "requirement_description": "Implement OAuth 2.0 authentication with Azure AD B2C for member portal login with MFA su
  "jira_epic_id": "EPIC-1234"  // optional
}

// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "requirement_id": "req_001",
  "status": "submitted",
  "character_count": 95,
  "extracted_keywords": ["OAuth 2.0", "Azure AD B2C", "authentication", "MFA", "member portal"]
}
```

POST /api/v1/requirement/upload

```json
```

```
// Request: multipart/form-data
// - session_id: string
// - files: File[] (.doc, .pdf, .png, .jpg)

// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "uploaded_files": [
    {
      "file_id": "file_001",
      "filename": "requirements_spec.pdf",
      "size_bytes": 245000,
      "mime_type": "application/pdf",
      "stored_path": "uploads/sess_20251231_143052_abc123/requirements_spec.pdf"
    }
  ]
}
```

5.2.3 Step 2: Historical Matches

POST /api/v1/search/find-matches

json

```
// Request
{
  "session_id": "sess_20251231_143052_abc123",
  "search_config": {
    "semantic_weight": 0.7,      // configurable
    "keyword_weight": 0.3,       // configurable
    "max_results": 10
  }
}

// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "matches": [
    {
      "match_id": "match_001",
      "epic_id": "EPIC-005",
      "epic_name": "Enterprise SSO Implementation",
      "description": "Implemented SAML 2.0 based SSO across all internal applications with MFA support",
      "match_score": 0.94,
      "score_breakdown": {
        "semantic_similarity": 0.92,
        "keyword_match": 0.88
      },
      "technologies": ["OAuth 2.0", "SAML", "Azure AD", "React", "Node.js"],
      "team_size": 5,
      "actual_hours": 480,
      "estimated_hours": 520,
      "variance": "-8%",
      "lessons_learned": "Early integration testing with identity provider saved significant debugging time"
    },
    // ... more matches
  ],
  "search_metadata": {
    "total_candidates_searched": 100,
    "search_duration_ms": 245,
    "weights_used": {
      "semantic": 0.7,
      "keyword": 0.3
    }
  }
}
```

POST /api/v1/search/select-matches

json

```
// Request
{
  "session_id": "sess_20251231_143052_abc123",
  "selected_match_ids": ["match_001", "match_002", "match_004"]
}

// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "selected_matches": 3,
  "status": "matches_confirmed",
  "ready_for_impact_analysis": true
}
```

5.2.4 Step 3: Impact Results (5 Agent Calls)

POST /api/v1/impact/generate/modules

json

```
// Request
{
  "session_id": "sess_20251231_143052_abc123"
}

// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "agent": "modules",
  "generated_at": "2025-12-31T14:35:22Z",
  "modules": {
    "functional": [
      {"name": "Member Login Flow", "impact": "HIGH", "reason": "Core authentication changes"},
      {"name": "Session Management", "impact": "HIGH", "reason": "Token-based session handling"},
      {"name": "Password Reset", "impact": "MEDIUM", "reason": "Integration with Azure AD B2C"},
      {"name": "Remember Me Feature", "impact": "LOW", "reason": "Minor token storage updates"}
    ],
    "technical": [
      {"name": "Auth Service", "impact": "HIGH", "reason": "New OAuth flow implementation"},
      {"name": "API Gateway", "impact": "HIGH", "reason": "Token validation middleware"},
      {"name": "Database Schema", "impact": "MEDIUM", "reason": "User token storage tables"},
      {"name": "Logging Service", "impact": "LOW", "reason": "Auth event logging"}
    ]
  },
  "total_modules": 10,
  "audit_file": "sessions/2025-12-31/sess_20251231_143052_abc123/agent_modules.json"
}
```

POST /api/v1/impact/generate/effort

```
json
```

```
// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "agent": "effort",
  "effort_breakdown": {
    "development": {
      "hours": 476,
      "breakdown": [
        {"module": "Auth Service", "hours": 160},
        {"module": "API Gateway", "hours": 120},
        // ...
      ]
    },
    "qa_testing": {
      "hours": 120,
      "breakdown": [...]
    },
    "design": {
      "hours": 40,
      "breakdown": [...]
    },
    "total_hours": 596
  },
  "story_points": 89,
  "historical_comparison": {
    "average": 420,
    "min": 280,
    "max": 520
  },
  "confidence": 0.85,
  "confidence_reasoning": "Based on 2 similar historical projects with 87%+ match"
}
```

POST /api/v1/impact/generate/stories

```
json
```

```
// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "agent": "stories",
  "stories": [
    {
      "story_id": "generated_001",
      "title": "Implement Azure AD B2C Integration",
      "type": "Story",
      "story_points": 13,
      "priority": "HIGH",
      "acceptance_criteria": [
        "User can authenticate via Azure AD B2C",
        "MFA is enforced for all logins",
        "Error handling for authentication failures"
      ],
      "linked_modules": ["Auth Service", "Member Login Flow"]
    },
    // ... more stories
  ],
  "total_stories": 10,
  "total_story_points": 89
}
```

POST /api/v1/impact/generate/code

```
json
```

```
// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "agent": "code_impact",
  "impacted_files": [
    {
      "file_path": "src/auth/AuthContext.tsx",
      "repository": "member-portal-frontend",
      "language": "TypeScript",
      "change_type": "modified",
      "reason": "Update auth context for OAuth flow"
    },
    {
      "file_path": "src/pages/Login.tsx",
      "repository": "member-portal-frontend",
      "language": "TypeScript",
      "change_type": "modified",
      "reason": "Integrate Azure AD B2C login button"
    },
    {
      "file_path": "src/auth/OAuthCallback.tsx",
      "repository": "member-portal-frontend",
      "language": "TypeScript",
      "change_type": "new",
      "reason": "Handle OAuth callback and token exchange"
    },
    // ... more files
  ],
  "total_files": 8,
  "summary": {
    "new_files": 2,
    "modified_files": 6,
    "repositories_affected": ["member-portal-frontend", "auth-service"]
  }
}
```

POST /api/v1/impact/generate/risks

```
json
```

```
// Response
{
  "session_id": "sess_20251231_143052_abc123",
  "agent": "risks",
  "risks": [
    {
      "risk_id": "risk_001",
      "title": "Azure AD B2C Service Dependency",
      "severity": "HIGH",
      "description": "Authentication flow depends on Azure AD B2C availability. Service outage would prevent all logins.",
      "mitigation": "Implement fallback authentication mechanism and proper error handling"
    },
    {
      "risk_id": "risk_002",
      "title": "Token Migration Complexity",
      "severity": "MEDIUM",
      "description": "Existing users with active sessions need seamless migration to new token-based system.",
      "mitigation": "Implement grace period with dual-auth support during migration"
    },
    // ... more risks
  ],
  "total_risks": 5,
  "high_risks": 2,
  "medium_risks": 2,
  "low_risks": 1
}
```

GET /api/v1/impact/{session_id}/summary

json

```
// Response - Aggregated from all agent outputs
{
  "session_id": "sess_20251231_143052_abc123",
  "summary": {
    "total_effort_hours": 596,
    "story_points": 89,
    "modules_impacted": 10,
    "high_risks": 2
  },
  "estimation_confidence": {
    "score": 0.85,
    "based_on": "2 similar historical projects"
  },
  "modules": {...},
  "effort": {...},
  "stories": {...},
  "code_impact": {...},
  "risks": {...},
  "generated_at": "2025-12-31T14:40:15Z"
}
```

5.2.5 Configuration

PUT /api/v1/config/search-weights

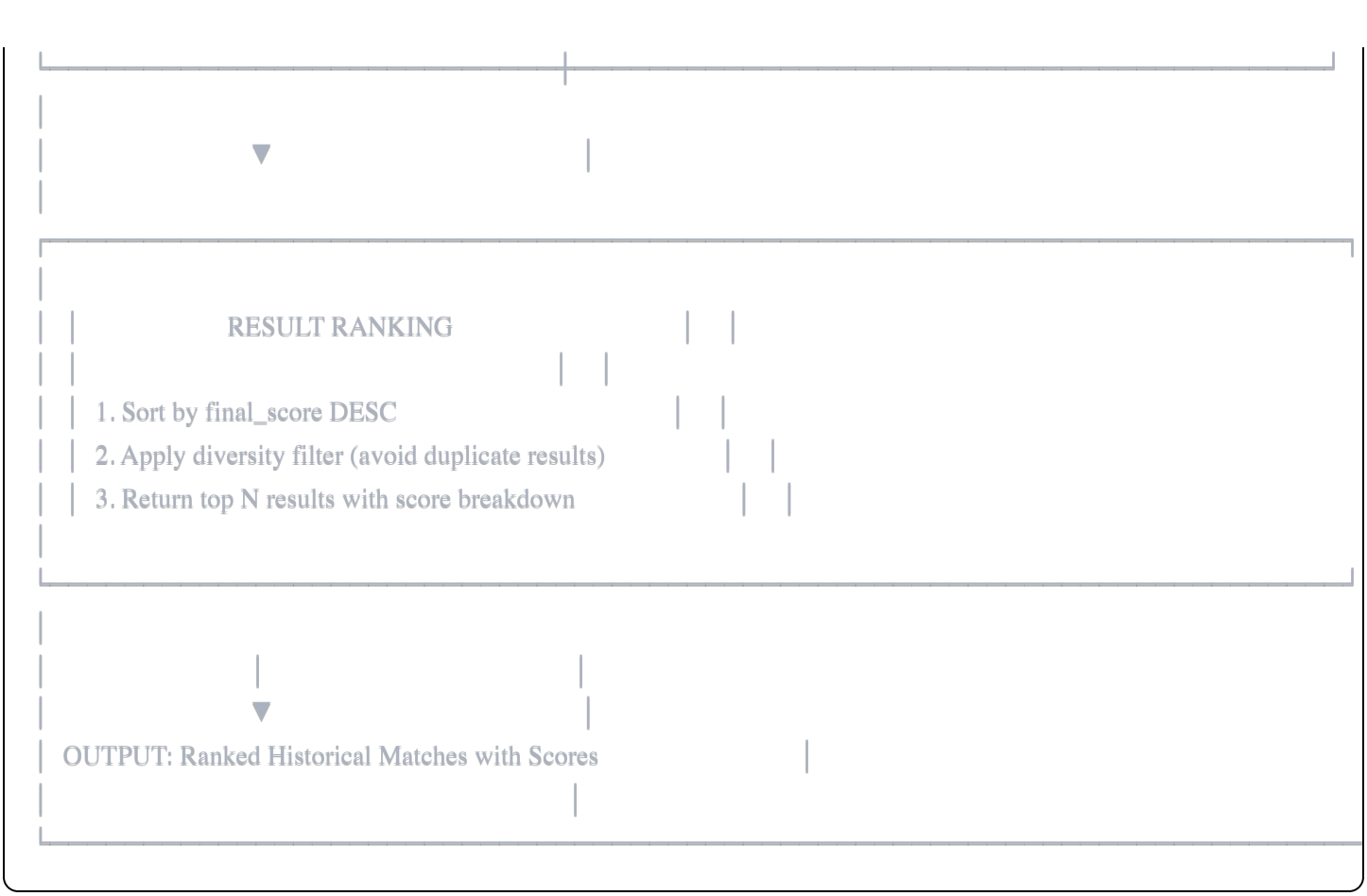
json

```
// Request
{
  "semantic_weight": 0.6,
  "keyword_weight": 0.4
}

// Response
{
  "status": "updated",
  "weights": {
    "semantic_weight": 0.6,
    "keyword_weight": 0.4
  },
  "validation": "Weights sum to 1.0 ✓"
}
```

6. RAG Pipeline Design

6.1 Hybrid Search Architecture

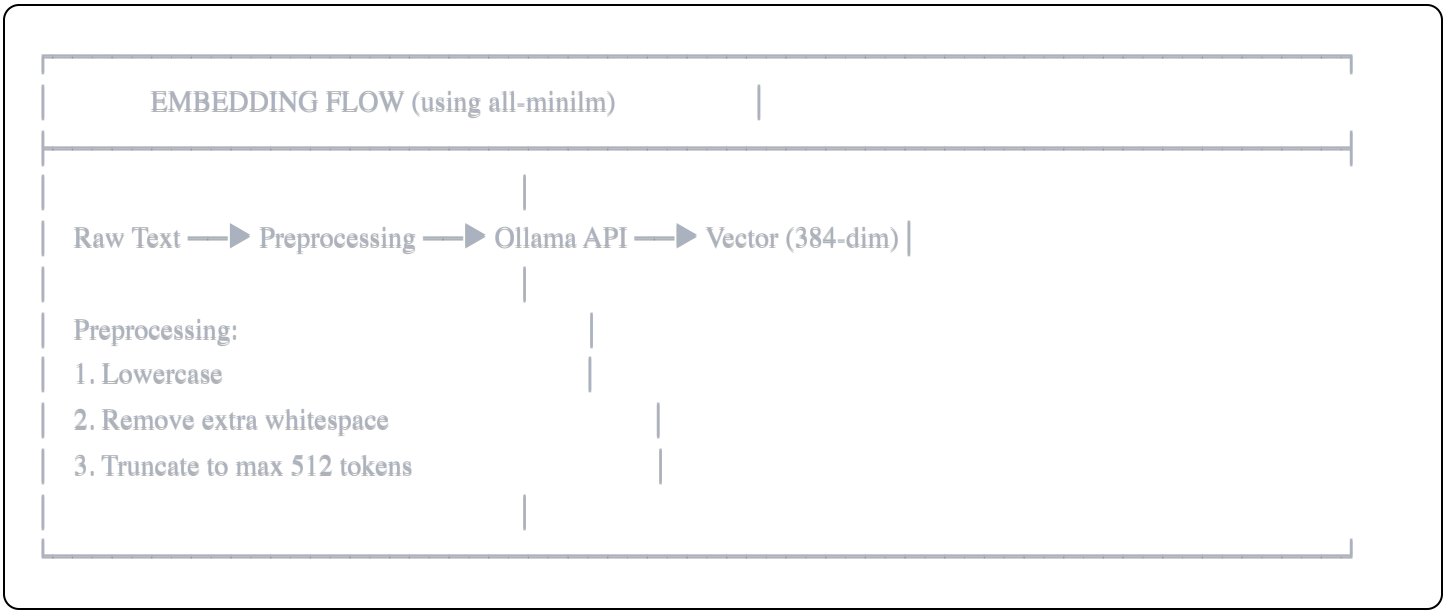


6.2 Embedding Strategy

Document Chunking & Embedding:

Collection	Fields to Embed	Chunking Strategy
epics	epic_title + epic_description	Single chunk per epic
estimations	task_description + tdd_description	Single chunk per estimation
tdds	tdd_section_name + tdd_section_content_summary	Single chunk per TDD section
stories	story_title + acceptance_criteria	Single chunk per story
gitlab_code	code_block_description + functions_defined	Single chunk per code block

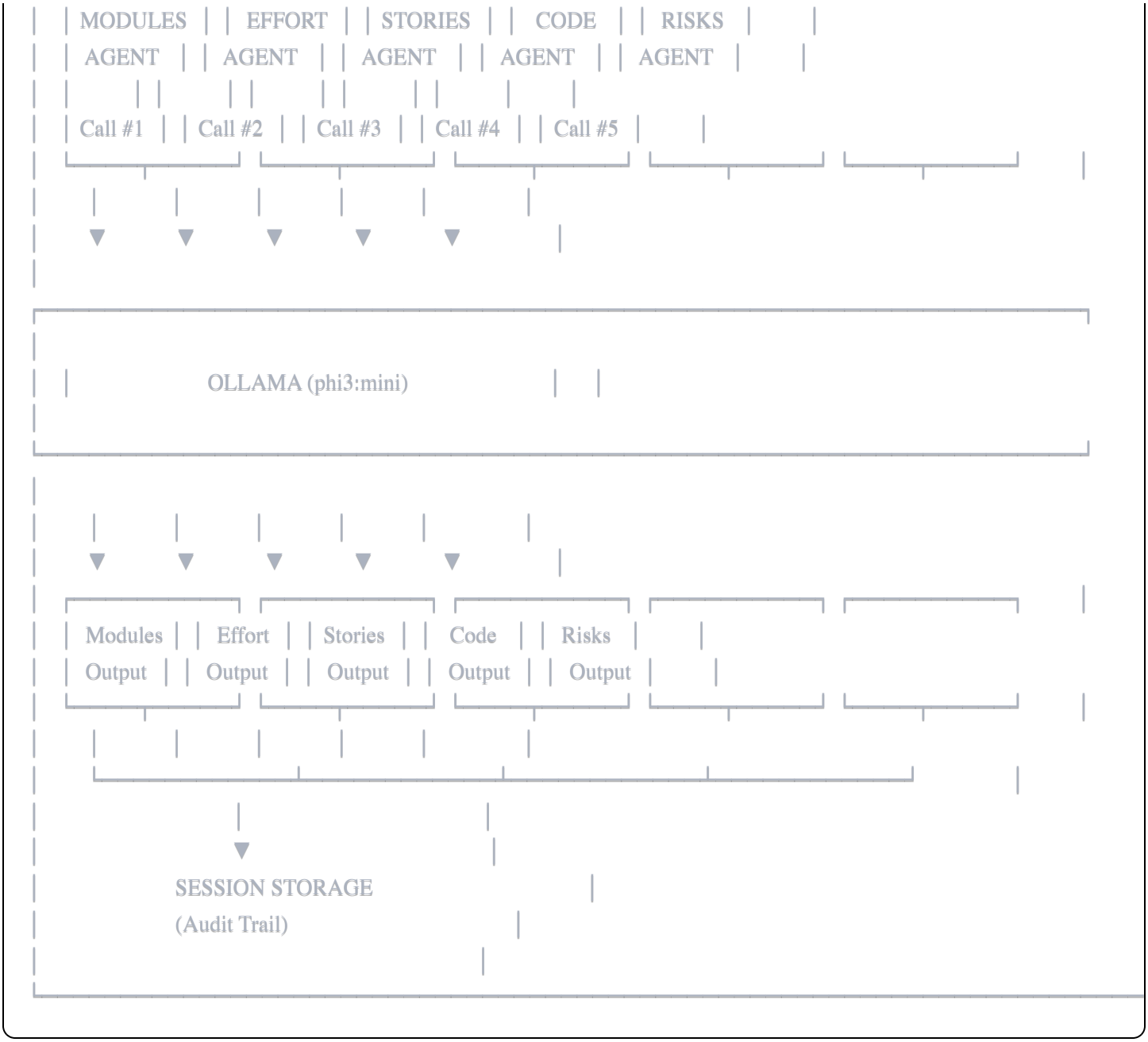
Embedding Generation:



---

7. Multi-Agent LLM Orchestration

7.1 Agent Architecture



7.2 Agent Specifications

Agent 1: Modules Agent

Purpose: Identify impacted functional and technical modules

Input Context:

- Requirement description
- Selected historical matches

Prompt Template:

```
You are an expert software architect analyzing project requirements.

REQUIREMENT:
{requirement_description}

SIMILAR HISTORICAL PROJECTS:
{formatted_historical_matches}

Based on this requirement and historical data, identify the impacted modules.

OUTPUT FORMAT (JSON):
{
  "functional_modules": [
    {"name": "string", "impact": "HIGH|MEDIUM|LOW", "reason": "string"}
  ],
  "technical_modules": [
    {"name": "string", "impact": "HIGH|MEDIUM|LOW", "reason": "string"}
  ]
}

Provide exactly 10 modules total (mix of functional and technical).
```

Output Schema:

```
json

{
  "functional_modules": [
    {"name": "string", "impact": "enum", "reason": "string"}
  ],
  "technical_modules": [
    {"name": "string", "impact": "enum", "reason": "string"}
  ]
}
```

Agent 2: Effort Agent

Purpose: Generate effort breakdown by activity type

Input Context:

- Requirement description
- Selected historical matches (with actual/estimated hours)
- Generated modules (from Agent 1)

Prompt Template:

```
You are a project estimation expert.

REQUIREMENT:
{requirement_description}

HISTORICAL EFFORT DATA:
{historical_effort_data}

Provide effort estimates based on historical patterns.

OUTPUT FORMAT (JSON):
{
  "development_hours": number,
  "qa_testing_hours": number,
  "design_hours": number,
  "total_hours": number,
  "story_points": number,
  "confidence_score": 0.0-1.0,
  "confidence_reasoning": "string"
}
```

Agent 3: Stories Agent

Purpose: Generate Jira stories with acceptance criteria

Input Context:

- Requirement description
- Generated modules (from Agent 1)
- Historical stories patterns

Prompt Template:

```
You are a product owner creating user stories.

REQUIREMENT:
{requirement_description}

IMPACTED MODULES:
{modules}

Generate user stories for implementation.

OUTPUT FORMAT (JSON):
{
  "stories": [
    {
      "title": "string",
      "type": "Story|Task",
      "story_points": number,
      "priority": "HIGH|MEDIUM|LOW",
      "acceptance_criteria": ["string"],
      "linked_modules": ["string"]
    }
  ]
}

Generate exactly 10 stories.
```

Agent 4: Code Impact Agent

Purpose: Identify files and repositories that will be impacted

Input Context:

- Requirement description
- Generated modules (from Agent 1)
- Historical code patterns from similar projects

Prompt Template:

```
You are a senior developer analyzing code impact.

REQUIREMENT:
{requirement_description}

IMPACTED MODULES:
{modules}

HISTORICAL CODE PATTERNS:
{historical_code}

Identify files and repositories that will be affected.

OUTPUT FORMAT (JSON):
{
  "impacted_files": [
    {
      "file_path": "string",
      "repository": "string",
      "language": "string",
      "change_type": "new|modified",
      "reason": "string"
    }
  ]
}

Generate 8-12 impacted files.
```

Agent 5: Risks Agent

Purpose: Identify potential risks and mitigation strategies

Input Context:

- Requirement description
- Generated modules (from Agent 1)
- Historical lessons learned

Prompt Template:

```
You are a risk analyst reviewing project requirements.

REQUIREMENT:
{requirement_description}

IMPACTED MODULES:
{modules}

HISTORICAL LESSONS LEARNED:
{lessons_learned}

Identify potential risks and mitigation strategies.

OUTPUT FORMAT (JSON):
{
  "risks": [
    {
      "title": "string",
      "severity": "HIGH|MEDIUM|LOW",
      "description": "string",
      "mitigation": "string"
    }
  ]
}

Generate 5 risks with varying severity levels.
```

7.3 LLM Configuration

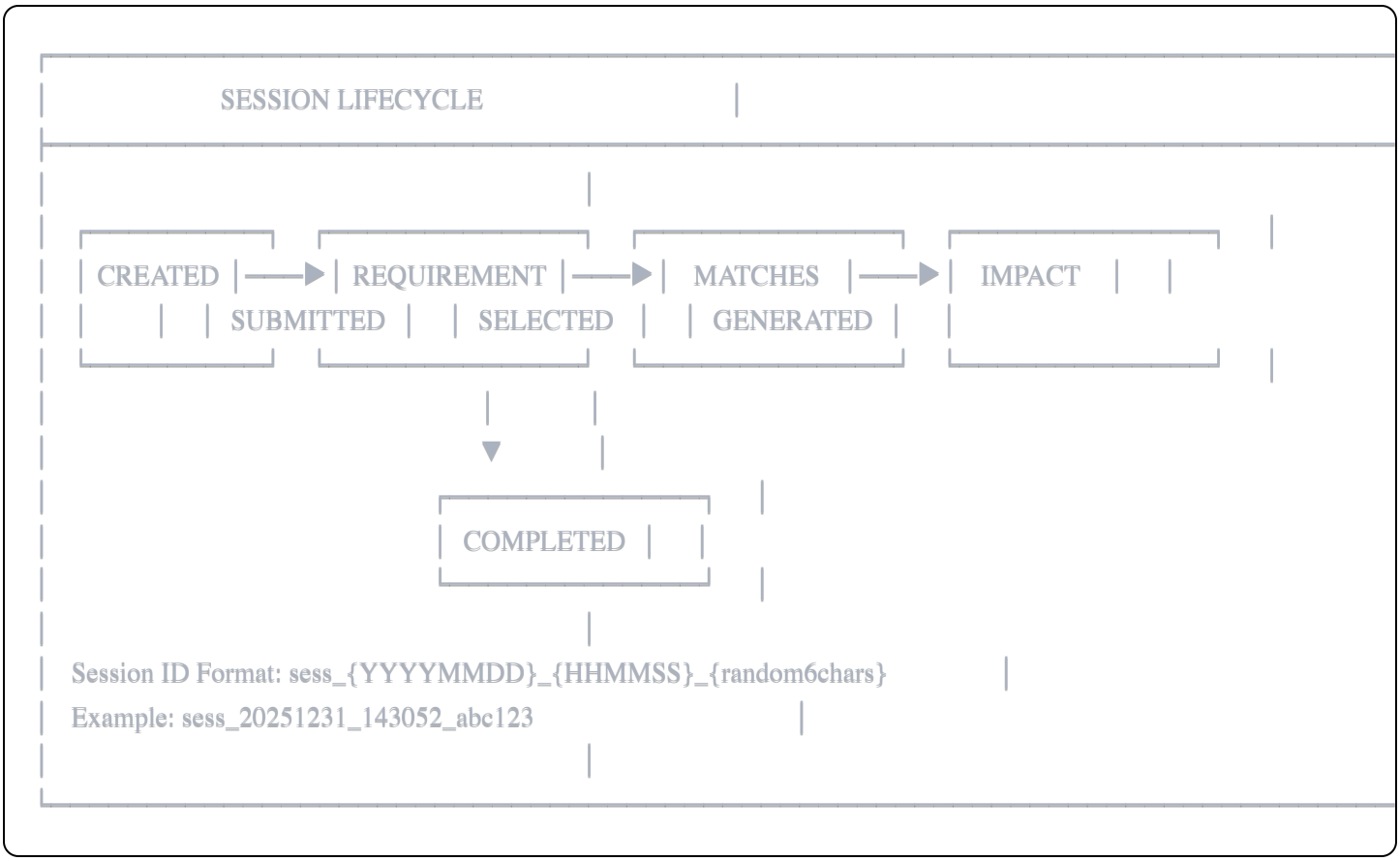
```
┌──────────────────────────────────────────────────────────────────────────────┐
│                        OLLAMA CONFIGURATION                                  │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  Model: phi3:mini                                                            │
│                                                                              │
│  Parameters:                                                                 │
│  ├── temperature: 0.3  (lower for more consistent output)                    │
│  ├── top_p: 0.9                                                              │
│  ├── max_tokens: 2048                                                        │
│  └── format: json (for structured output)                                    │
│                                                                              │
│  System Message:                                                             │
```

```
"You are a helpful AI assistant that always responds in
 valid JSON format. Follow the output schema exactly."
```

---

8. Session & Audit Management

8.1 Session Lifecycle



8.2 Audit Folder Structure

```
sessions/
└── 2025-12-31/
    └── sess_20251231_143052_abc123/
        │
        ├── session_metadata.json       # Session info and status
        │
        ├── step1_input/
        │   ├── requirement.json        # Submitted requirement text
        │   ├── extracted_keywords.json # AI-extracted keywords
        │   └── uploads/                # Uploaded documents
```

```
│       ├── requirements_spec.pdf
│       └── architecture_diagram.png
│
├── step2_search/
│   ├── search_request.json      # Search configuration used
│   ├── all_matches.json         # All returned matches
│   └── selected_matches.json    # User-selected matches
│
├── step3_agents/
│   ├── agent_modules.json       # Modules agent output
│   │   ├── input_prompt.txt     # Full prompt sent
│   │   ├── raw_response.txt     # Raw LLM response
│   │   └── parsed_output.json   # Parsed structured output
│   │
│   ├── agent_effort.json        # Effort agent output
│   │   └── ...
│   │
│   ├── agent_stories.json       # Stories agent output
│   │   └── ...
│   │
│   ├── agent_code.json          # Code impact agent output
│   │   └── ...
│   │
│   └── agent_risks.json         # Risks agent output
│       └── ...
│
└── final_summary.json           # Aggregated final output
```

8.3 Session Metadata Schema

json

```
{
  "session_id": "sess_20251231_143052_abc123",
  "created_at": "2025-12-31T14:30:52Z",
  "updated_at": "2025-12-31T14:45:30Z",
  "status": "completed",
  "user_id": "optional-user-id",
  "steps_completed": {
    "step1_requirement": true,
    "step2_matches": true,
    "step3_modules": true,
    "step3_effort": true,
    "step3_stories": true,
    "step3_code": true,
    "step3_risks": true
  },
  "timing": {
    "step1_duration_ms": 1200,
    "step2_search_duration_ms": 2450,
    "step3_modules_duration_ms": 3200,
    "step3_effort_duration_ms": 2800,
    "step3_stories_duration_ms": 4100,
    "step3_code_duration_ms": 3500,
    "step3_risks_duration_ms": 2900,
    "total_duration_ms": 20150
  },
  "configuration_used": {
    "search_weights": {
      "semantic": 0.7,
      "keyword": 0.3
    },
    "llm_model": "phi3:mini",
    "embedding_model": "all-minilm"
  }
}
```


---

9. Configuration Management

9.1 Configuration Schema

yaml

```yaml
# config/settings.yaml

app:
  name: "AI Impact Assessment System"
  version: "1.0.0"
  environment: "development"  # development | production

server:
  host: "0.0.0.0"
  port: 8000
  cors_origins:
    - "http://localhost:3000"
    - "http://localhost:5173"

ollama:
  base_url: "http://localhost:11434"
  generation_model: "phi3:mini"
  embedding_model: "all-minilm"
  timeout_seconds: 120
  generation_params:
    temperature: 0.3
    top_p: 0.9
    max_tokens: 2048

chromadb:
  persist_directory: "./data/chroma"
  collection_prefix: "impact_assessment"
  embedding_dimension: 384

search:
  default_weights:
    semantic: 0.70
    keyword: 0.30
  max_results: 10
  min_score_threshold: 0.3

data:
  raw_data_path: "./data/raw"
  uploads_path: "./data/uploads"
  sessions_path: "./data/sessions"

logging:
  level: "INFO"
```

```
  format: "json"
  file_path: "./logs/app.log"
```

9.2 Environment Variables

```
bash

# .env file

# Application
APP_ENV=development
DEBUG=true

# Ollama
OLLAMA_BASE_URL=http://localhost:11434
OLLAMA_GEN_MODEL=phi3:mini
OLLAMA_EMBED_MODEL=all-minilm

# ChromaDB
CHROMA_PERSIST_DIR=./data/chroma

# Search Weights (defaults, configurable via API)
SEARCH_WEIGHT_SEMANTIC=0.70
SEARCH_WEIGHT_KEYWORD=0.30

# Paths
DATA_RAW_PATH=./data/raw
DATA_UPLOADS_PATH=./data/uploads
DATA_SESSIONS_PATH=./data/sessions
```

---

10. Folder Structure

```
ai-impact-assessment/
│
├── app/
│   ├── __init__.py
│   ├── main.py                 # FastAPI application entry point
│   │
│   ├── api/
│   │   ├── __init__.py
```

```
│   │   ├── routes/
│   │   │   ├── __init__.py
│   │   │   ├── session.py          # Session management endpoints
│   │   │   ├── requirement.py      # Step 1 endpoints
│   │   │   ├── search.py           # Step 2 endpoints
│   │   │   ├── impact.py           # Step 3 endpoints
│   │   │   ├── config.py           # Configuration endpoints
│   │   │   └── health.py           # Health check endpoints
│   │   │
│   │   └── dependencies.py         # FastAPI dependencies
│   │
│   ├── core/
│   │   ├── __init__.py
│   │   ├── config.py               # Configuration management
│   │   └── logging.py              # Logging configuration
│   │
│   ├── models/
│   │   ├── __init__.py
│   │   ├── request.py              # Pydantic request models
│   │   ├── response.py             # Pydantic response models
│   │   └── domain.py               # Domain models
│   │
│   ├── services/
│   │   ├── __init__.py
│   │   ├── session_service.py      # Session management logic
│   │   ├── requirement_service.py  # Requirement processing
│   │   ├── search_service.py       # Hybrid search logic
│   │   ├── impact_service.py       # Impact analysis orchestration
│   │   └── file_service.py         # File upload handling
│   │
│   ├── agents/
│   │   ├── __init__.py
│   │   ├── base_agent.py           # Base agent class
│   │   ├── modules_agent.py        # Modules identification agent
│   │   ├── effort_agent.py         # Effort estimation agent
│   │   ├── stories_agent.py        # Story generation agent
│   │   ├── code_agent.py           # Code impact agent
│   │   ├── risks_agent.py          # Risk identification agent
│   │   └── orchestrator.py         # Agent orchestration
│   │
│   ├── rag/
│   │   ├── __init__.py
│   │   ├── embeddings.py           # Embedding generation
│   │   ├── vector_store.py         # ChromaDB operations
```

```
│   │   └── hybrid_search.py        # Hybrid search implementation
│   │
│   └── utils/
│       ├── __init__.py
│       ├── audit.py                # Audit logging utilities
│       └── helpers.py              # General utilities
│
├── data/
│   ├── raw/                        # Source CSV/JSON files
│   │   ├── epics.csv
│   │   ├── estimations.csv
│   │   ├── tdds.csv
│   │   ├── stories_tasks.csv
│   │   └── gitlab_code.json
│   │
│   ├── chroma/                     # ChromaDB persistence
│   │
│   ├── uploads/                    # Uploaded documents
│   │
│   └── sessions/                   # Session audit trails
│       └── {date}/
│           └── {session_id}/
│
├── config/
│   └── settings.yaml               # Main configuration
│
├── scripts/
│   ├── init_vector_db.py           # Initialize ChromaDB with data
│   ├── test_ollama.py              # Test Ollama connection
│   └── reindex.py                  # Reindex vector database
│
├── tests/
│   ├── __init__.py
│   ├── test_api/
│   ├── test_services/
│   ├── test_agents/
│   └── test_rag/
│
├── logs/                           # Application logs
│
├── docs/                           # Documentation
│   └── DATA_SCHEMA_README.md
│
├── .env                            # Environment variables
```

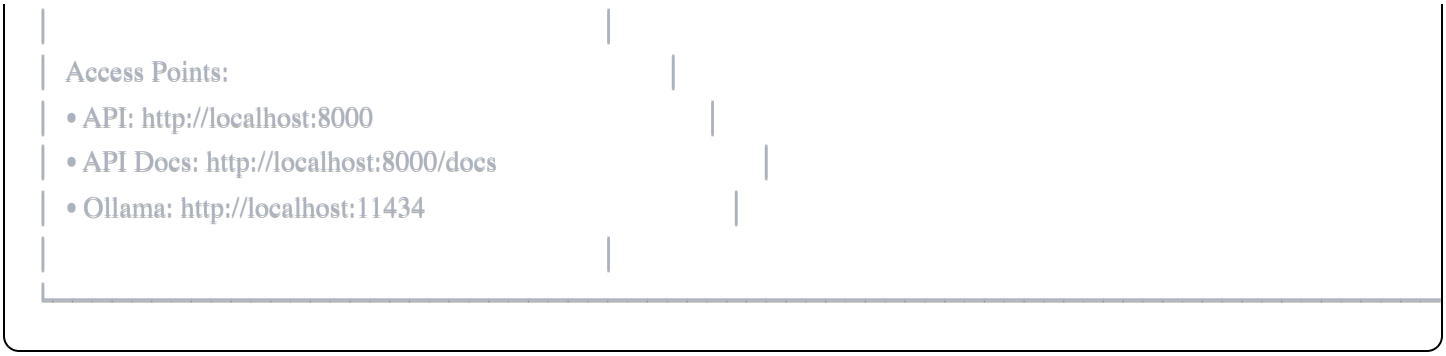
```
├── .env.example              # Example environment file
├── requirements.txt          # Python dependencies
├── pyproject.toml            # Project metadata
├── Makefile                  # Common commands
└── README.md                 # Project README
```

---

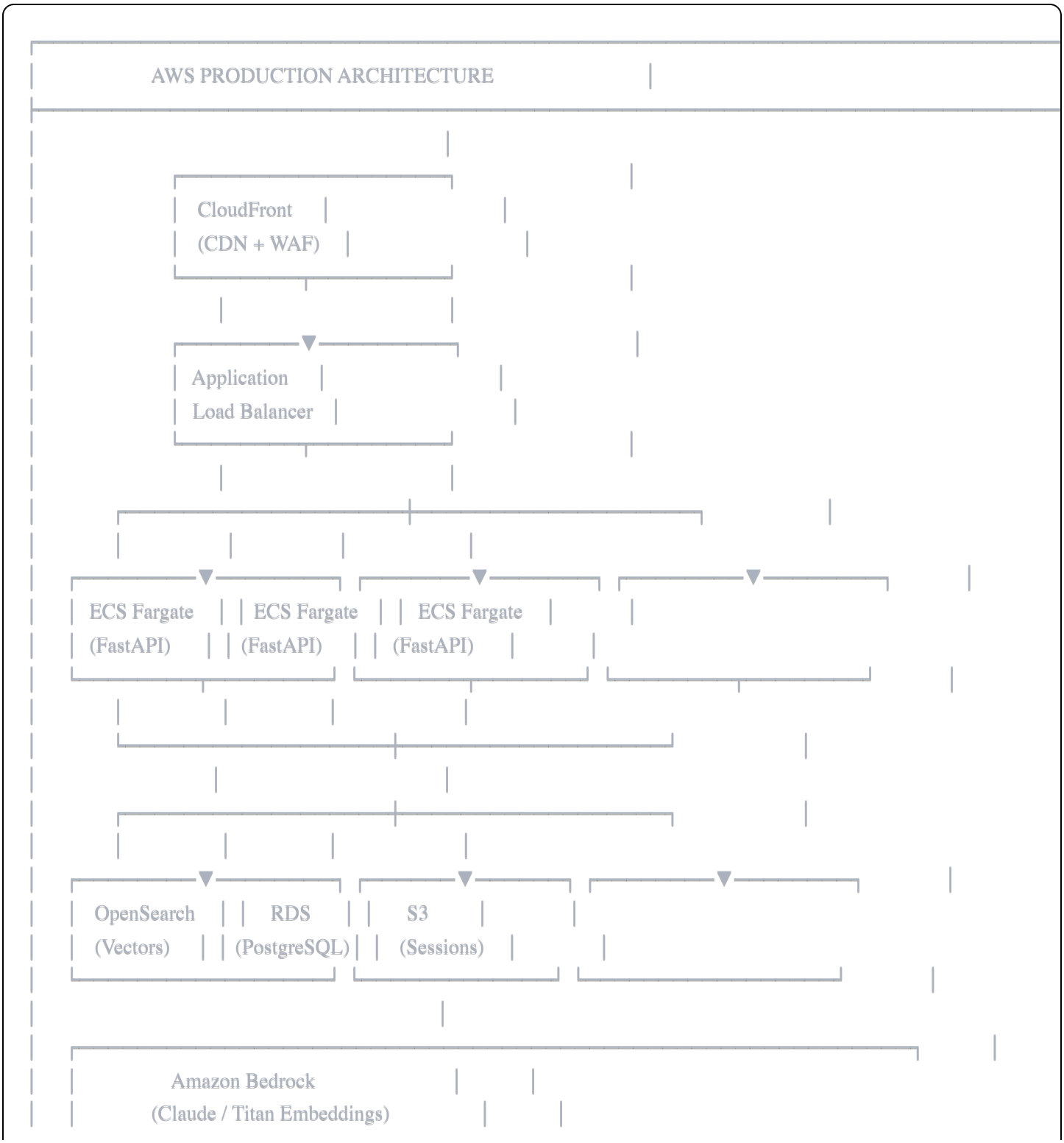
11. Deployment Strategy

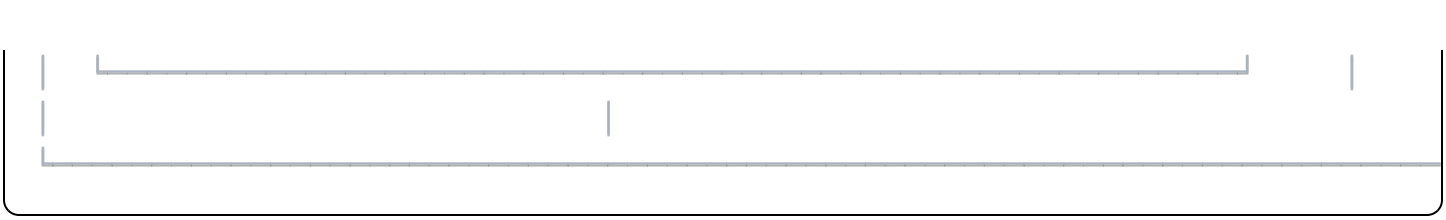
11.1 Local Development (MacBook)

```
┌─────────────────────────────────────────────────────────────────────────┐
│                    LOCAL DEVELOPMENT SETUP                              │
├─────────────────────────────────────────────────────────────────────────┤
│                                                                         │
│  Prerequisites:                                                         │
│  1. Python 3.11+ installed                                              │
│  2. Ollama installed (brew install ollama)                              │
│  3. Pull required models:                                               │
│     - ollama pull phi3:mini                                             │
│     - ollama pull all-minilm                                            │
│                                                                         │
│  Startup Sequence:                                                      │
│                                                                         │
│  ┌─────────────────────────────────────────────────────────────────┐    │
│  │                                                                 │    │
│  │  Terminal 1: Start Ollama                                       │    │
│  │  $ ollama serve                                                 │    │
│  │                                                                 │    │
│  │  Terminal 2: Start FastAPI                                      │    │
│  │  $ cd ai-impact-assessment                                      │    │
│  │  $ python -m venv venv                                          │    │
│  │  $ source venv/bin/activate                                     │    │
│  │  $ pip install -r requirements.txt                              │    │
│  │  $ python scripts/init_vector_db.py  # First time only          │    │
│  │  $ uvicorn app.main:app --reload --port 8000                    │    │
│  │                                                                 │    │
│  │  Terminal 3: Start Frontend (if applicable)                     │    │
│  │  $ cd frontend && npm run dev                                   │    │
│  │                                                                 │    │
│  └─────────────────────────────────────────────────────────────────┘    │
│                                                                         │
```



11.2 AWS Production Architecture (Future)





---

12. Future Enhancements

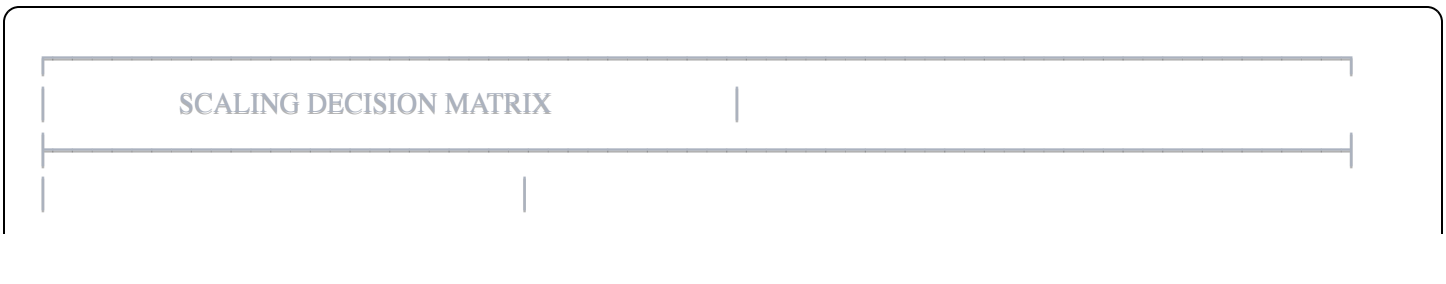
12.1 Phase 2 Enhancements

Feature	Description	Priority
Real-time Jira Integration	Create tickets directly in Jira	High
Document Content Extraction	Parse PDF/DOC content for context	High
Streaming Responses	Stream agent outputs for better UX	Medium
User Authentication	Add user management and auth	Medium
Export to PDF/Excel	Generate downloadable reports	Medium

12.2 Phase 3 Enhancements

Feature	Description	Priority
GitLab Integration	Pull real code structure	High
Feedback Loop	Learn from actual vs estimated	High
Multi-tenant Support	Support multiple organizations	Medium
A/B Testing for Prompts	Optimize agent prompts	Low
Dashboard Analytics	Historical accuracy metrics	Low

12.3 Scalability Considerations



```
| Data Volume        Current        Recommended Action          |
|-------------------------------------------------------------- |
| < 1,000 records    ChromaDB       Keep current setup          |
| 1,000-10,000       ChromaDB       Add indexing, pagination    |
| 10,000-100,000     Pinecone       Migrate to managed vector DB|
| > 100,000          OpenSearch     Enterprise vector search    |
|                                                               |
| Concurrent Users   Current        Recommended Action          |
|-------------------------------------------------------------- |
| < 10 users         Single server  Keep current setup          |
| 10-50 users        Single server  Add caching (Redis)         |
| 50-200 users       Load balanced  Add multiple instances      |
| > 200 users        ECS/K8s        Full containerized deploy   |
|                                                               |
```

Appendix A: API Error Codes

Code	HTTP Status	Description
`SESSION_NOT_FOUND`	404	Session ID does not exist
`INVALID_SESSION_STATE`	400	Operation not allowed in current session state
`REQUIREMENT_TOO_SHORT`	400	Requirement text < 20 characters
`FILE_TYPE_NOT_ALLOWED`	400	Uploaded file type not supported
`SEARCH_WEIGHTS_INVALID`	400	Search weights don't sum to 1.0
`OLLAMA_UNAVAILABLE`	503	Cannot connect to Ollama server
`LLM_TIMEOUT`	504	LLM response exceeded timeout
`VECTOR_DB_ERROR`	500	ChromaDB operation failed

Document End

This strategy document serves as the foundation for implementing the AI Impact Assessment System backend. All implementation should follow the patterns and structures defined herein.