Low Level Design Document

Introduction

This Low Level Design (LLD) document details the implementation plan for **FinScenarioMap - AI-Powered Banking Scenario Mapping Tool**. The system enables users to input or upload hypothetical banking scenarios, maps them to similar historical cases using AI, and generates recommended actions. The design leverages Python, langgraph, agentic-ai, generative-ai, and OpenAI for modular, secure, and extensible scenario analysis.

---

1. System Components

Component	Description / Responsibility
API Layer	Receives scenario input (text/upload), returns results
Scenario Parser	Parses and validates user scenarios
Scenario Mapper	Maps input to historical cases using agentic-ai/langgraph
Recommendation Engine	Generates actions using generative-ai/OpenAI
Data Store	Stores historical cases and scenario metadata
Workflow Orchestrator	Manages modular workflow via langgraph

---

2. Class/Interface Overview

Class/Interface	Responsibility	Key Methods/Attributes
`ScenarioInputHandler`	Input validation & parsing	`parse_input()` , `validate()`
`ScenarioMapper`	Mapping to historical cases	`find_similar_cases()` , `score_case()`
`RecommendationEngine`	Generate recommendations	`generate_actions()` , `summarize()`
`CaseRepository`	Data access for cases	`get_cases()` , `save_scenario()`
`WorkflowManager`	Orchestrate workflow steps	`run_workflow()` , `handle_errors()`

Relationships:

- `API Layer` → `ScenarioInputHandler` → `ScenarioMapper` → `RecommendationEngine`
- `ScenarioMapper` & `RecommendationEngine` use `CaseRepository`
- `WorkflowManager` coordinates all steps

---

3. Data Structure Overview

Model	Fields / Attributes

Scenario	`id` , `user_id` , `description` , `input_type` , `timestamp`
HistoricalCase	`case_id` , `summary` , `tags` , `outcome` , `actions`
Recommendation	`scenario_id` , `actions` , `confidence_score` , `details`

---

4. Algorithms / Logic

Scenario Mapping & Recommendation Flow:

```
def process_scenario(input_data):
    scenario = ScenarioInputHandler.parse_input(input_data)
    similar_cases = ScenarioMapper.find_similar_cases(scenario)
    recommendations = RecommendationEngine.generate_actions(scenario, similar_ca
    return recommendations
```

Mapping Logic (Pseudocode):

- Parse scenario
- Embed scenario using agentic-ai
- Retrieve top-N similar historical cases (vector similarity)
- Pass scenario + cases to generative-ai/OpenAI for recommendations

---

5. Error Handling

Error Scenario	Handling Approach
Invalid Input/Format	Return 400 error, log details
No Similar Cases Found	Return empty result, suggest alternatives
AI/Model API Failure	Retry, fallback to default response
Data Store Unavailable	Return 503 error, log and alert
Workflow Step Failure	Abort workflow, return error message

End of Document