Context Aware Testing System for Financial Ecosystems

1. Scope

This document covers the requirements, design and architecture for Context aware testing system.

2. Requirements

Functional Requirements

1. Generate context aware test cases for financial transactions, customer interactions, fraud detection, regulatory compliance and risk assessment
2. AI agent which can update test cases, based on system changes, reducing maintenance efforts
3. AI driven test scenario synthesis to simulate real world banking activities such as KYC validation, loan approvals, real-time fraud detection and compliance monitoring.

Non-Functional Requirements

1. Improve test efficiency
2. Improve test accuracy
3. Cost savings

3. Solution Design & Architecture

This application supports below testing scenarios:

- Generic context
- Ethical hacking
- Fraud Detection
- Loan KYC
- Financial Stock
- Chat bot based on Agentic AI
- BDD testing

The solution consists of below major components:

- Context Sources

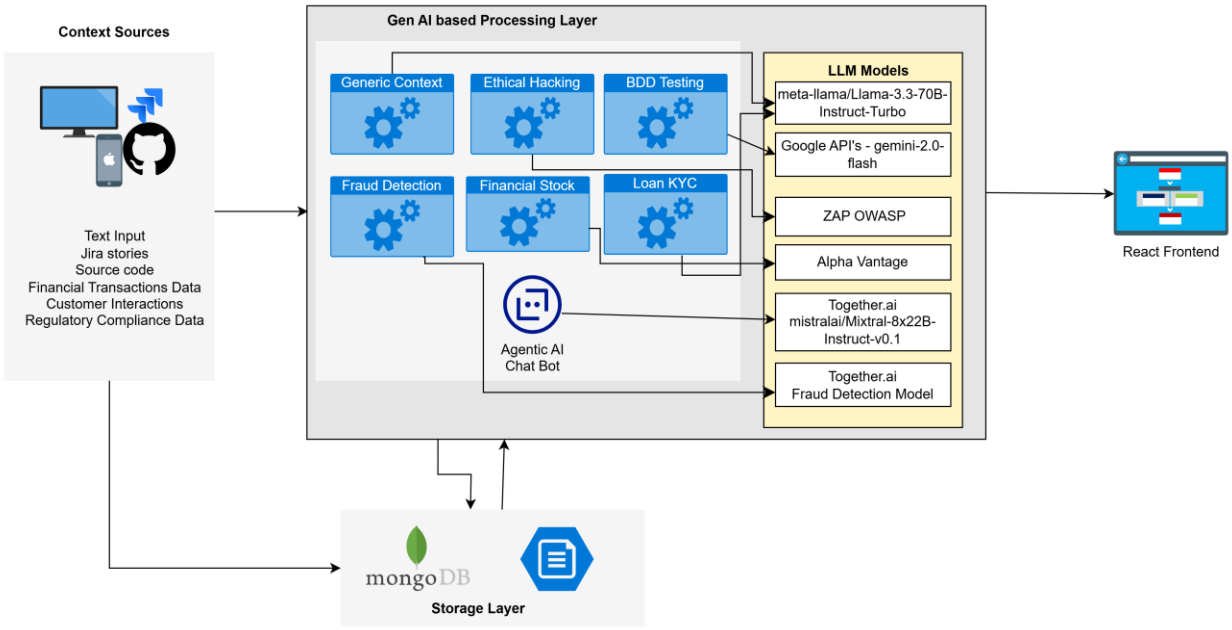
a. Text input, financial transactions, customer interactions and regulatory compliance standards are the context sources used.

- Gen AI based Processing layer

The processing layer uses context sources & LLM models to generate test cases, run test cases & generates test results. The user input will be captured via React frontend and results are also displayed.

- Storage Layer
  The storage layer takes care of storing the application usage data and storing test scenarios, results etc (currently not implemented in solution).
  a. Mongo DB
  b. Cloud file storage
- Front End
  a. Interacts with processing layer via HTTP requests.
  b. Displays chatbot UI and test reports.

Below diagram depicts the high-level conceptual architecture.



Below is the high-level technical architecture of the system:

1. **Frontend (React/Vue/Angular - Not included in the uploaded files)**
   - Interacts with backend APIs via HTTP requests.
   - Displays chatbot UI and test reports.
2. **Processing Layer (Spring Boot Microservices)**
   - **Controllers (REST APIs)**:
     - `ChatController` → AI-based chat

        - ♣ `ContextAwareController` → AI-driven test automation
        - ♣ `GreeterController` → Greeting API
    - **Services (Business Logic)**:
        - ♣ `AgentService` → AI chat + test automation
        - ♣ `OpenAIService` → Test case generation
        - ♣ `GenAIService` → Fraud detection + test automation
        - ♣ `EthicalHackService` → Security scanning
        - ♣ `TestExecutorService` → BDD test execution
3. **External Integrations**
    - **OpenAI/LLM APIs** → AI-based test case generation
    - **Alpha Vantage API** → Real-time stock data
    - **OWASP ZAP** → Security penetration testing
    - **TogetherAI API** → Fraud detection
4. **Deployment & Execution**
    - **Runs on Kubernetes** using ConfigMaps.
    - Uses **Maven + Cucumber** for test execution.
    - Generates **HTML reports for security & test execution**.