

**CONTEXT AWARE  
TESTING FOR  
FINANCIAL  
ECOSYSTEMS**



## **TEAM CTRL-ALT-ELITE TEAM MEMBERS**

- **ATIRAJ KUMAR**
- **JYOTHIKAMALESH S**
- **KAARTHIK SHANKAR**
- **AYUSH SINGH**
- **SAURAV KUMAR**

# SOLUTIONS

## FINSURE

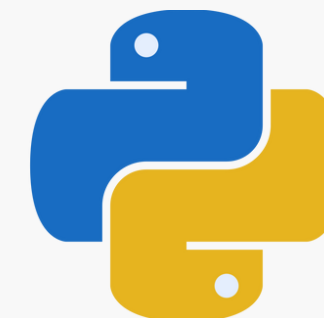
A **regulation RAG** and **context-aware testing** solution with **agentic AI**

*Features:*

- **Regulations Knowledge Database and RAG Pipeline**
- **Context aware Test Function Generator and Executor Pipeline**
- **Context-aware Test API Requests Generator and Executor Pipeline (API agnostic)**
- **AI Agents for UI Automation**



VSCode Plugin



Python package



CLI Tool



Github Workflow Generator



Regulatory Compliance VectorDB(Elasticsearch)



### PaymentAPI Defender

GenAI helps create scenarios to test the robustness of our partner Payments systems, with explanations of failure that can be understood by all stakeholders



### CodebaseDefender

An intelligent agentic system for testing your code - armed with knowledge of all compliance requirements, as soon as they become available



### Agentic-UI Tester & PromptDefender

Automatic natural language testing of UI components  
Automatic checking of Chatbots for data security and data loss prevention



### CreditEngineDefender

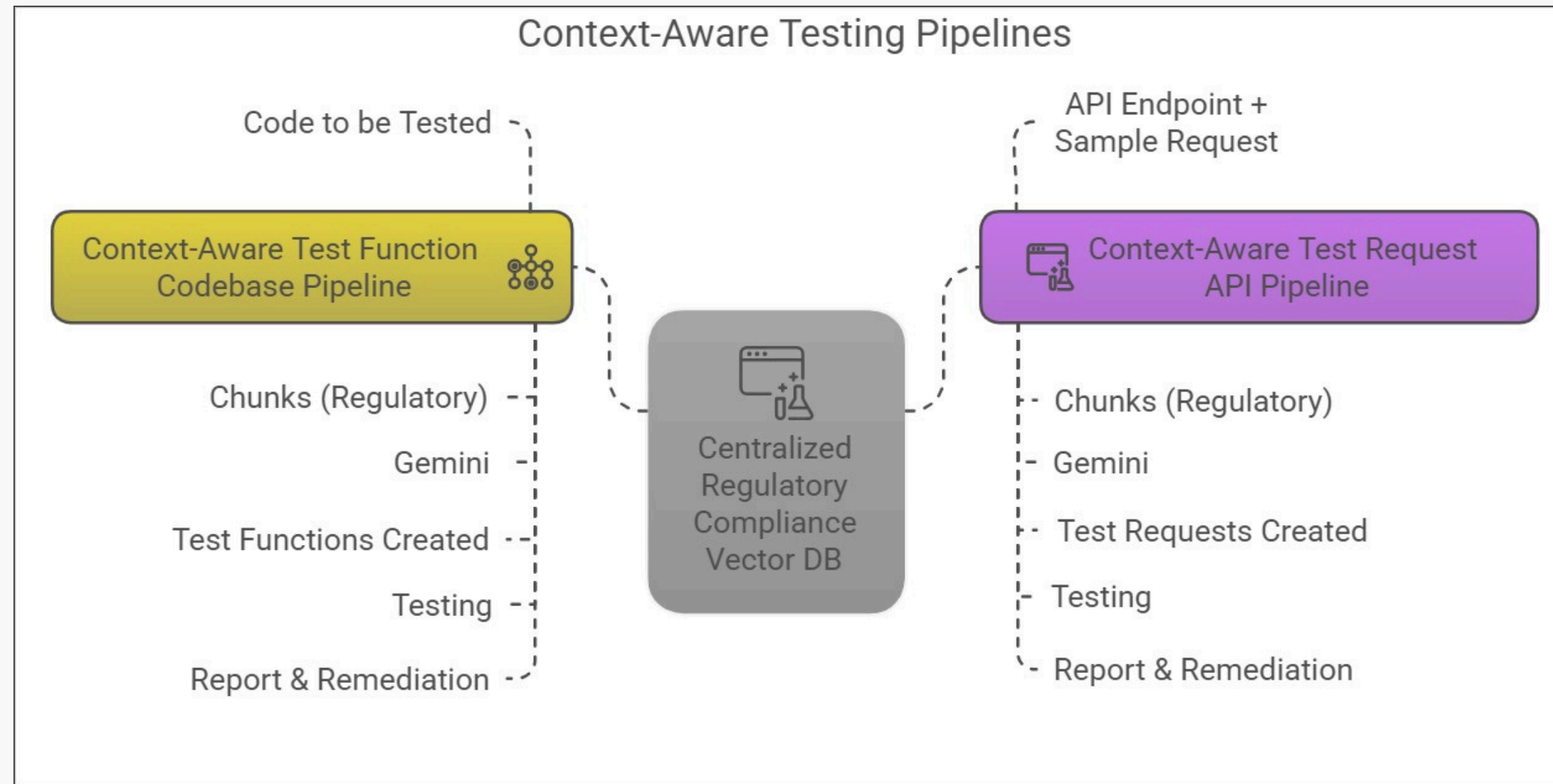
GenAI helps simulate different personas to automatically test Loan and Credit risk assessment models for reliability and fairness



### Plug and Play Workflows for Enterprise Pipeline

All solutions, immediately available in Github Actions for seamless integration with our new Enterprise Pipeline

# CONTEXT-AWARE TESTING PIPELINE



## Regulations Knowledge Database and RAG Pipeline

*Tech stack: Elastic search*

## Context aware Test Generator and Executor Pipeline Codebase-defender

(SOX, KYC, AML focused test functions)

*Tech stack: Java and Gradle*

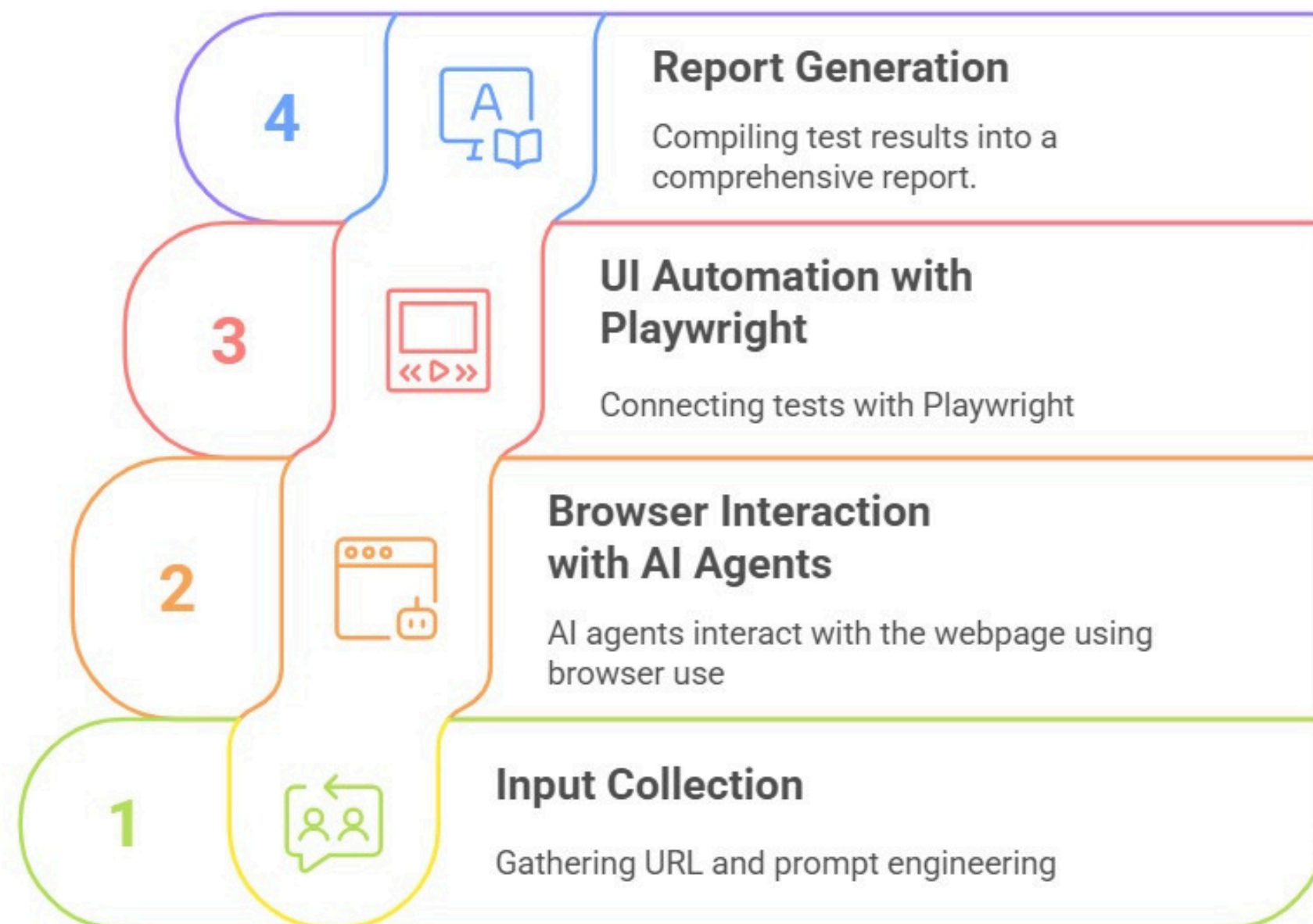
## Context-aware Test Request API: Generator and Executor Pipeline (API agnostic)

- **PaymentAPI-defender**
  - Swift messaging standard compliance, FED, CHIP based requests)
- **Creditengine-defender**
  - (ML model test requests)

*Tech stack: PayPal Sandbox API and Python*

# AI AGENTS FOR UI AUTOMATION

## AI-Agent Chatbot (Prompt Injection) and UI Testing Workflow



- **Prompt-defender**

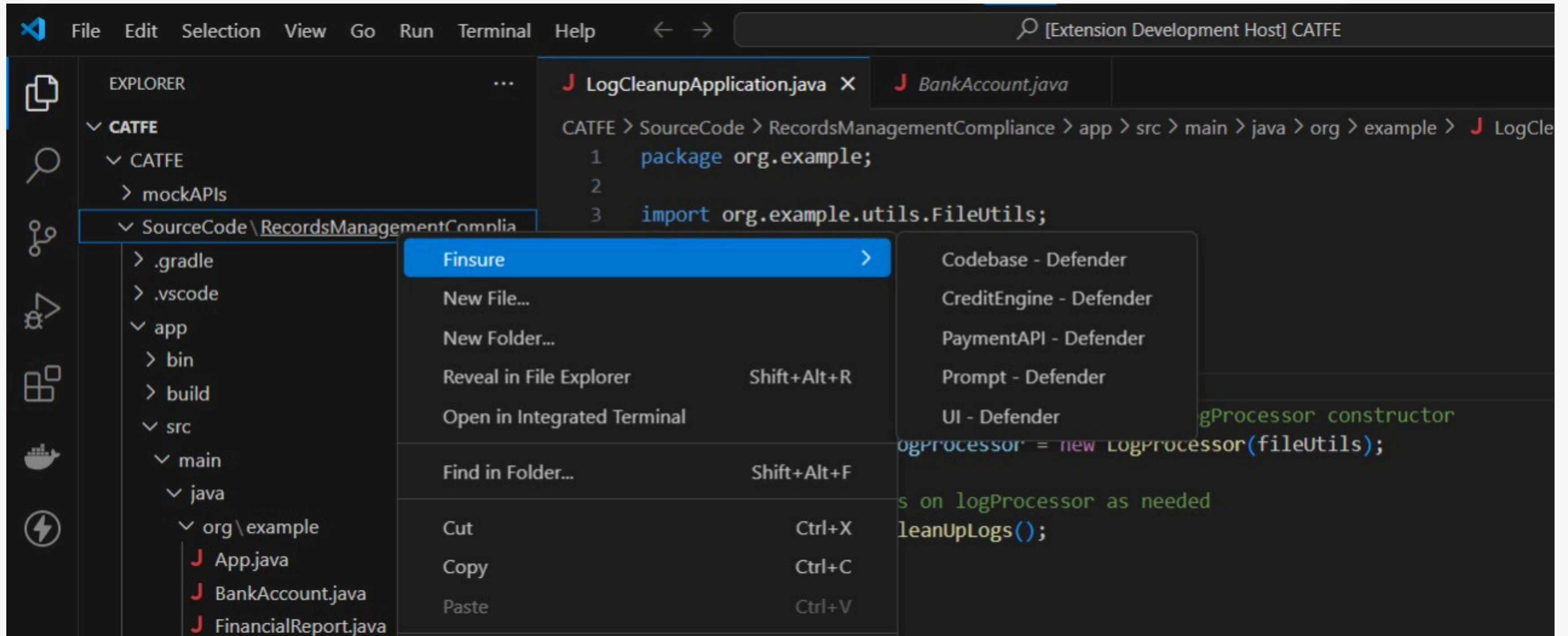
- UI test agent that generates finance environment aware prompt injections and creates test reports

- **UI-Defender**

- UI test Agent that can do any task with natural language instruction
- Techstack - Browseruse and Playwright*



# VSCODE PLUGIN



*VS Code Plugin user interface*

# COMMAND LINE INTERFACE

```
● (finsure) Jyothikamalesh@Jyothikamaleshs-MacBook-Pro testing tool % finsure --help
```

**Usage:** finsure [OPTIONS] COMMAND [ARGS]...

Options

**--help** Show this message and exit.

Commands

<b>github-testflow</b>	Generate GitHub Actions workflow,A Plug and Play CI/CD solution for github repo .
<b>paymentapi-defender</b>	Transaction API test with Cause analysis,Explainability and Remediation of test cases.
<b>codebase-defender</b>	RAG powered regulatory compliance test function generator and execution pipeline.
<b>creditengine-defender</b>	Credit engine ML API test cases generation and execution.
<b>prompt-defender</b>	Prompt injection and Chatbot UI Automation test
<b>ui-defender</b>	NLP instructed Agentic UI automation tool

*Command line interface help screen showcasing our solutions*

# CODEDEFENDER

The screenshot displays the CodeDefender web application interface. At the top, a browser window shows the URL: `file:///C:/Users/sudhi/OneDrive/Desktop/CATFE/SourceCodeTesting/SourceCode/RecordsManagementCompliance/app/build/reports/tests/test/classes/org.example....`. The main heading is **SWIFTTransactionTest**, with a breadcrumb path: `all > org.example > SWIFTTransactionTest`.

Summary statistics are shown in a row of four boxes:

<b>4</b> tests	<b>1</b> failures	<b>0</b> ignored	<b>0.033s</b> duration
-------------------	----------------------	---------------------	---------------------------

To the right of these boxes is a red box indicating **75% successful**.

Below the summary, there are three tabs: **Failed tests**, **Tests**, and **Standard output**. The **Standard output** tab is active, displaying the following text:

```
testMandatoryFieldsPresent: This test would fail if mandatory fields like senderBIC, receiverBIC, amount, or currency are null or empty.
testMandatoryFieldsPresent: Remediate by ensuring all mandatory fields are populated before creating a SWIFTTransaction object.
testDataFormattingValidation: This test would fail if the valueDate is not in the correct ISO 8601 format (YYYYMMDD).
testDataFormattingValidation: Remediate by ensuring the valueDate is always formatted as YYYYMMDD.
testEnhancedFraudDataFields: This test would fail if the system does not have the ability to analyze Enhanced Fraud Data (EFD) fields like PurposeCode, AccountHolder
testEnhancedFraudDataFields: Remediate by adding logic to extract and analyze EFD fields for fraud detection.
testSanctionsScreeningDataExtraction: This test would fail if the system cannot extract Country, BIC, and IBAN fields for sanctions screening.
testSanctionsScreeningDataExtraction: Remediate by implementing logic to extract these fields and integrate with a sanctions screening service.
```

At the bottom left, there is a checkbox labeled "Wrap lines" which is currently unchecked. The Windows taskbar at the bottom shows the time as 17:38 on 26-03-2025.

*Screenshot of the summary of the automatically generated cause explanation and suggested remediations*



# CODEDEFENDER

```
package org.example;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class SWIFTTransactionTest {

    @Test
    void testMandatoryFieldsPresent() {
        System.out.println("testMandatoryFieldsPresent: This test would fail if mandatory fields like senderBIC, receiverBIC, amount, or currency are null or empty.");
        System.out.println("testMandatoryFieldsPresent: Remediate by ensuring all mandatory fields are populated before creating a SWIFTTransaction object.");

        SWIFTTransaction transaction = new SWIFTTransaction("REF123", null, "ABCD1234EFG", 100.0, "USD", "20240101");
        assertThrows(NullPointerException.class, () -> transaction.validateTransaction(), "Sender BIC cannot be null");

        SWIFTTransaction transaction2 = new SWIFTTransaction("REF123", "ABCD1234EFG", null, 100.0, "USD", "20240101");
        assertThrows(NullPointerException.class, () -> transaction2.validateTransaction(), "Receiver BIC cannot be null");

        SWIFTTransaction transaction3 = new SWIFTTransaction("REF123", "ABCD1234EFG", "ABCD1234EFG", 100.0, null, "20240101");
        assertThrows(NullPointerException.class, () -> transaction3.validateTransaction(), "Currency cannot be null");
    }

    @Test
    void testDataFormattingValidation() {
        System.out.println("testDataFormattingValidation: This test would fail if the valueDate is not in the correct ISO 8601 format (YYYYMMDD).");
        System.out.println("testDataFormattingValidation: Remediate by ensuring the valueDate is always formatted as YYYYMMDD.");

        SWIFTTransaction transaction = new SWIFTTransaction("REF123", "ABCD1234EFG", "EFGH5678IJK", 100.0, "USD", "2024-01-01");
        assertFalse(transaction.validateTransaction(), "Value date should be in YYYYMMDD format");

        SWIFTTransaction transaction2 = new SWIFTTransaction("REF123", "ABCD1234EFG", "EFGH5678IJK", 100.0, "USD", "20240101");
        assertTrue(transaction2.validateTransaction(), "Value date should be in YYYYMMDD format");
    }

    @Test
    void testEnhancedEmptyDataFields() {
```

*Screenshot of the summary of the automatically generated source code test cases*



# CREDITENGINE DEFENDER

The screenshot shows a web browser window with the title "Test Case Report". The address bar displays the file path: `file:///C:/Users/sudhi/OneDrive/Desktop/CATFE/LoanCreditRiskAssessment/credit-risk-analysis-using-ML/test_case_report.html`. The browser's taskbar at the bottom shows various application icons, including a search bar, a purple ribbon icon, and several web browsers. The main content area of the browser displays a "Test Case Execution Report".

## Test Case Execution Report

[Run All Tests](#)

### Valid input - Standard case

```
{
  "customer_id": "cust123",
  "name": "John Doe",
  "age": 35,
  "gender": "M",
  "owns_car": "Y",
  "owns_house": "Y",
  "no_of_children": 2.0,
  "net_yearly_income": 60000.0,
  "no_of_days_employed": 3650.0,
  "occupation_type": "Sales staff",
  "total_family_members": 4.0,
  "migrant_worker": 0.0,
  "yearly_debt_payments": 10000.0,
  "credit_limit": 100000.0,
  "credit_limit_used(%)": 50,
  "credit_score": 700.0,
  "prev_defaults": 0,
  "default_in_last_6months": 0
}
```

**Possible Failure Cause:** Model predicts default when it shouldn't, or vice versa. Data preprocessing errors.

**Expected Output:**

*Screenshot of the automatically generated Payment API test cases*

# CREDITENGINE DEFENDER

Meet - rpf-brao-dvm

SOLUTIONS - Presentation

Test Case Report

file:///C:/Users/sudhi/OneDrive/Desktop/CATFE/LoanCreditRiskAssessment/credit-risk-analysis-using-ML/test\_case\_report.html

GmailYouTubeMapsContext based Testing...

All Bookmarks

Test Case Execution Report

Run All Tests

Test Summary

Test Case	Description	Expected Status	Result
1	Valid input - Standard case	200	PASS
2	Valid input - No car, no children, no days employed, no credit score	200	PASS
3	Valid input - Edge case: Very high income, low credit usage	200	FAIL
4	Valid input - Edge case: Low income, high credit usage, previous defaults	200	FAIL
5	Invalid input - Wrong data type for age	422	FAIL
6	Invalid input - Invalid gender	422	FAIL
7	Invalid input - Missing required field (name)	422	FAIL
8	Invalid input - credit_limit_used(%) out of range	422	FAIL
9	Valid input - Edge case: Zero income	200	FAIL
10	Valid input - Edge case: Very old age	200	FAIL
11	Valid input - Edge case: No occupation type	422	FAIL

Valid input - Standard case

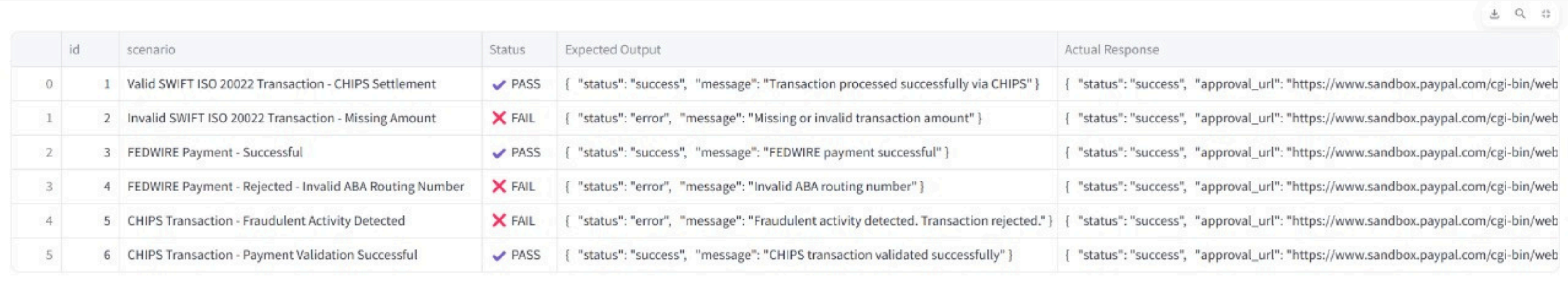
Search

ENG IN

17:31 26-03-2025

*Screenshot of the summary of the automatically generated Credit Risk Model test cases*

# PAYMENTAPI DEFENDER



	id	scenario	Status	Expected Output	Actual Response
0	1	Valid SWIFT ISO 20022 Transaction - CHIPS Settlement	✓ PASS	{ "status": "success", "message": "Transaction processed successfully via CHIPS" }	{ "status": "success", "approval_url": "https://www.sandbox.paypal.com/cgi-bin/web
1	2	Invalid SWIFT ISO 20022 Transaction - Missing Amount	✗ FAIL	{ "status": "error", "message": "Missing or invalid transaction amount" }	{ "status": "success", "approval_url": "https://www.sandbox.paypal.com/cgi-bin/web
2	3	FEDWIRE Payment - Successful	✓ PASS	{ "status": "success", "message": "FEDWIRE payment successful" }	{ "status": "success", "approval_url": "https://www.sandbox.paypal.com/cgi-bin/web
3	4	FEDWIRE Payment - Rejected - Invalid ABA Routing Number	✗ FAIL	{ "status": "error", "message": "Invalid ABA routing number" }	{ "status": "success", "approval_url": "https://www.sandbox.paypal.com/cgi-bin/web
4	5	CHIPS Transaction - Fraudulent Activity Detected	✗ FAIL	{ "status": "error", "message": "Fraudulent activity detected. Transaction rejected." }	{ "status": "success", "approval_url": "https://www.sandbox.paypal.com/cgi-bin/web
5	6	CHIPS Transaction - Payment Validation Successful	✓ PASS	{ "status": "success", "message": "CHIPS transaction validated successfully" }	{ "status": "success", "approval_url": "https://www.sandbox.paypal.com/cgi-bin/web

*Screenshot of the summary of the automatically generated Payment API test cases*

**THANK YOU**