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 CICD Action 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

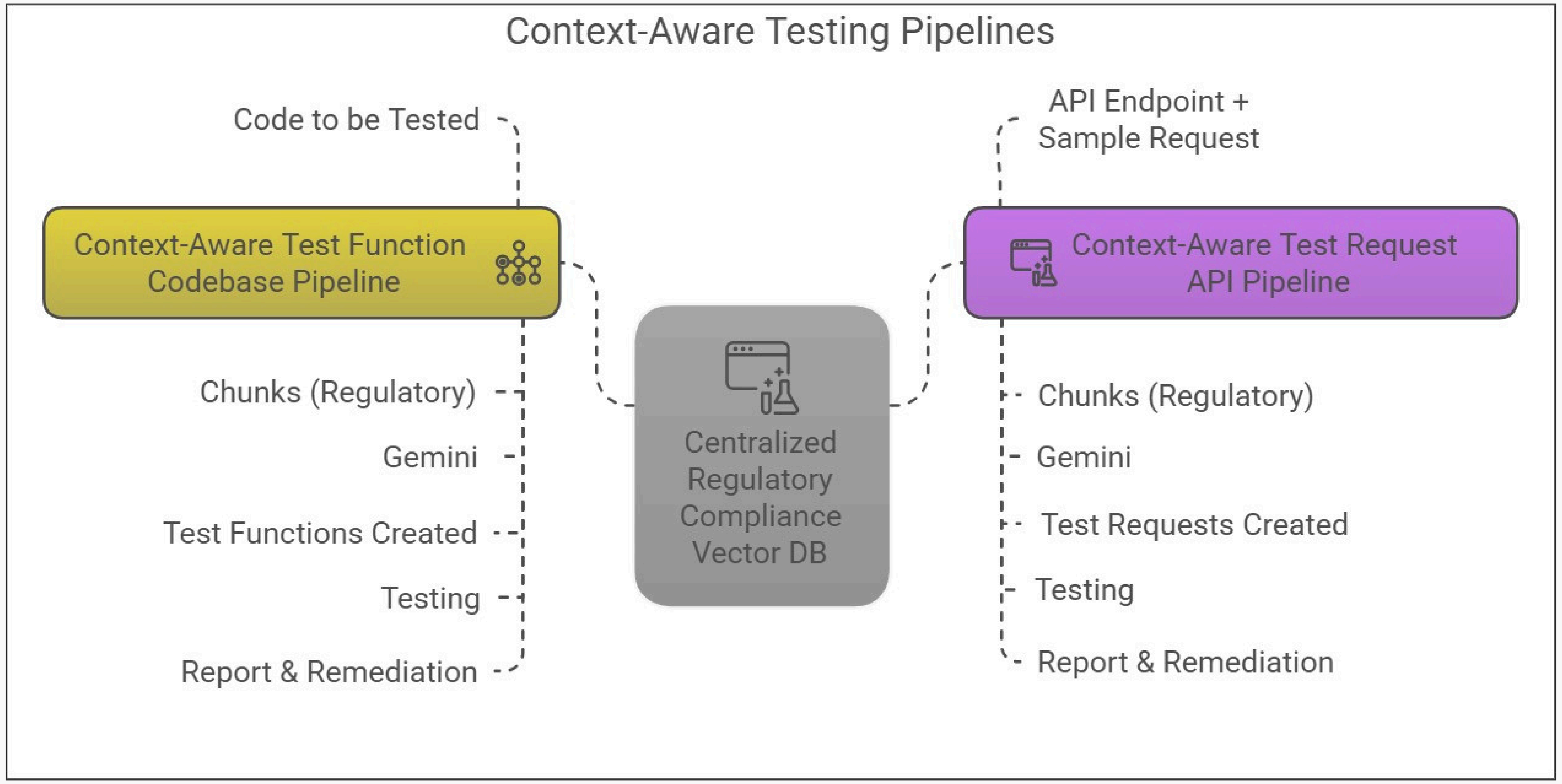
CreditEngineDefender

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 avaiable in Github Actions for seamless integration with our new Enterprise Pipeline

CONTEXT-AWARE TESTING PIPELINE


Context-Aware Testing Pipelines
Code to be Tested
Context-Aware Test Function Codebase Pipeline
Chunks (Regulatory)
Gemini
Test Functions Created
Testing
Report & Remediation
Centralized Regulatory Compliance Vector DB
API Endpoint + Sample Request
Context-Aware Test Request API Pipeline
Chunks (Regulatory)
Gemini
Test Requests Created
Testing
Report & Remediation

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

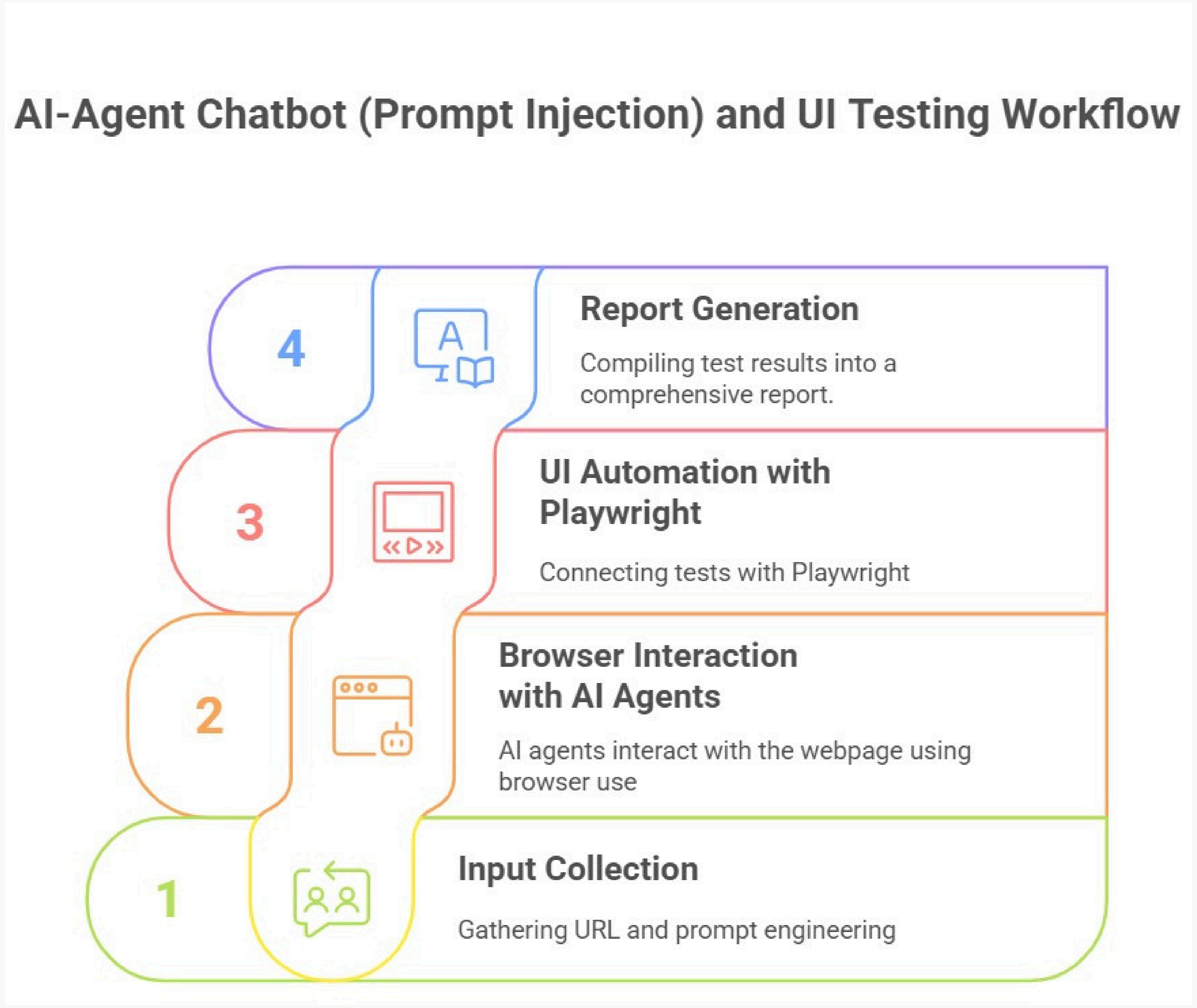
Context-aware Test Request API: Generator and Executor Pipeline (API agnostic)

- ***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
4
Report Generation
Compiling test results into a comprehensive report.
3
UI Automation with Playwright
Connecting tests with Playwright
2
Browser Interaction with AI Agents
AI agents interact with the webpage using browser use
1
Input Collection
Gathering URL and prompt engineering

- ***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

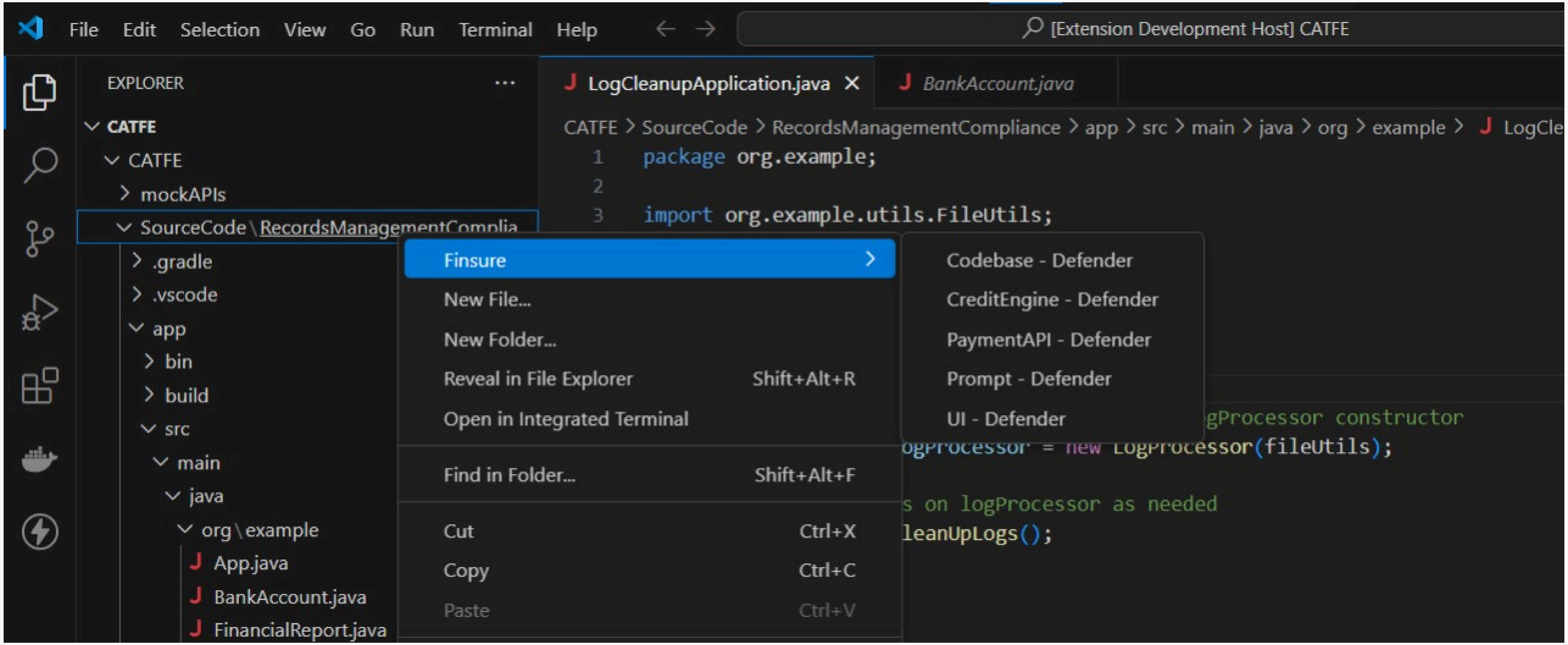
VECTORDB FOR FACT-GROUNDED REGULATION KNOWLEDGE

Data Ingestion Pipeline capable of ingesting webpages and pdfs as and when they become available

Ingested with the following data:

- Federal Reserve Website publications:
  - https://www.federalreserve.gov/publications/2021-ar-payment-system-and-reserve-bank-oversight.htm
  - https://www.federalreserve.gov/aboutthefed/fedexplained/payment-systems.htm
  - https://www.federalreserve.gov/paymentsystems/files/psr_policy.pdf
  - https://www.federalreserve.gov/paymentsystems/psr_about.htm
  - https://www.federalreserve.gov/paymentsystems/pfs_frpaysys.htm
- The Clearing House: CHIPS Rules and Administrative Procedures Effective April 1 2025
- ISO20022: Survival Guide
- ISO15022: Specification
- Federal Reserve Policy on Payment System Risk
- ISO20022 for dummies
- SWIFT MT Standards Publication November 2022
- Anti Money Laundering
  - https://iclg.com/practice-areas/anti-money-laundering-laws-and-regulations/usa
- Sarbanes-Oxley
  - https://www.upguard.com/blog/sox-compliance

VSCODE PLUGIN

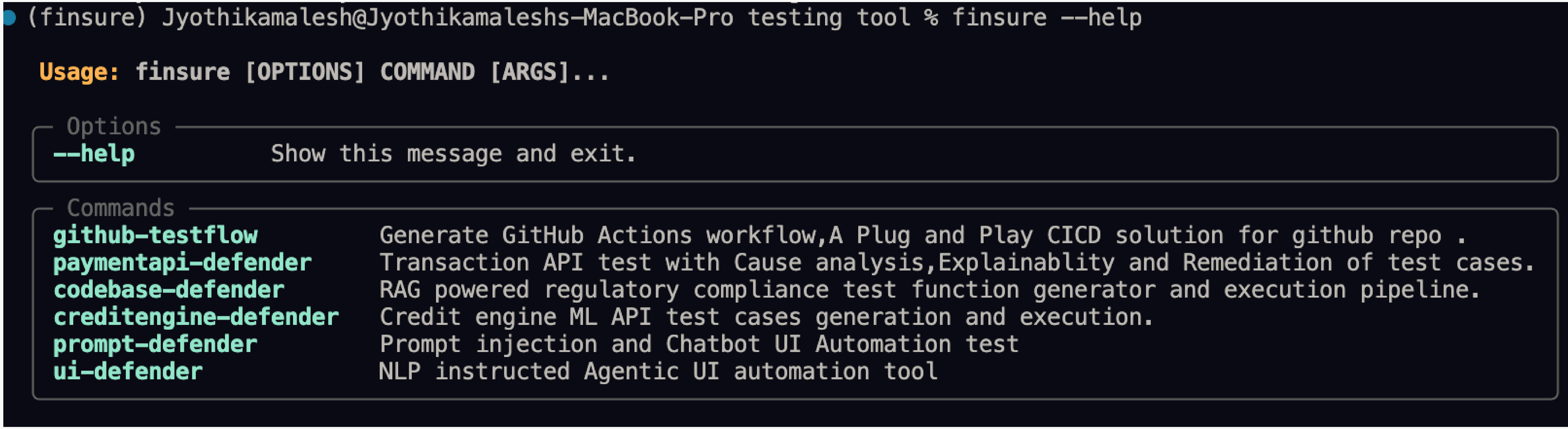


```
package org.example;

import org.example.utils.FileUtils;
```

```
gProcessor constructor
ogProcessor = new LogProcessor(fileUtils);

s on logProcessor as needed
leanUpLogs();
```

VSCode Plugin user interface

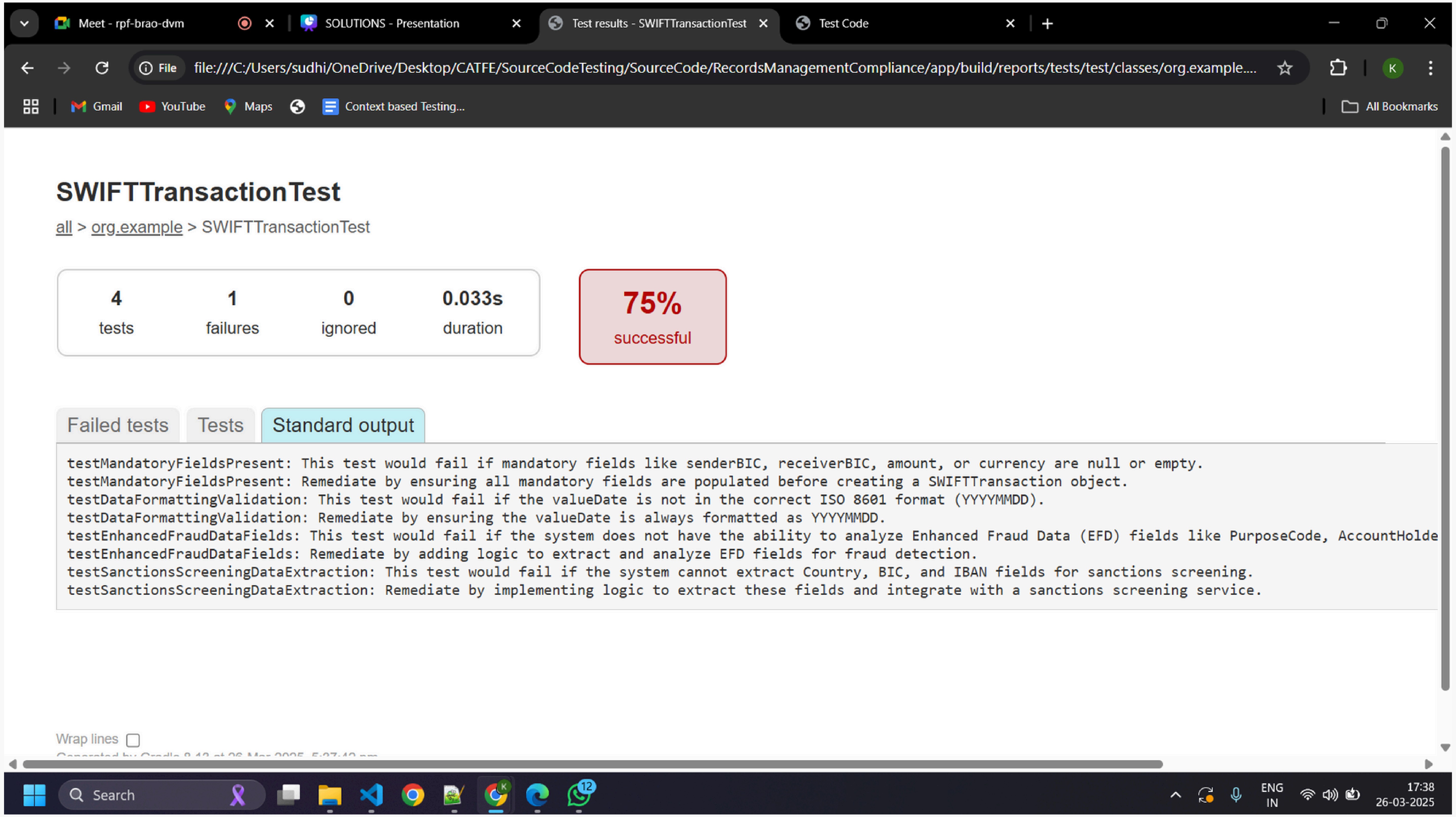
COMMAND LINE INTERFACE



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

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

╭─ Options ──────────────────────────────────────────────────────────────────────────────────────────╮
│ --help          Show this message and exit.                                                        │
╰────────────────────────────────────────────────────────────────────────────────────────────────────╯
╭─ Commands ─────────────────────────────────────────────────────────────────────────────────────────╮
│ github-testflow          Generate GitHub Actions workflow,A Plug and Play CICD solution for github repo .
│ paymentapi-defender      Transaction API test with Cause analysis,Explainablity and Remediation of test cases.
│ codebase-defender        RAG powered regulatory compliance test function generator and execution pipeline.
│ creditengine-defender    Credit engine ML API test cases generation and execution.
│ prompt-defender          Prompt injection and Chatbot UI Automation test
│ ui-defender              NLP instructed Agentic UI automation tool
╰────────────────────────────────────────────────────────────────────────────────────────────────────╯
```

Command line interface help screen showcasing our solutions

CODEDEFENDER


Meet - rpf-brao-dvm
SOLUTIONS - Presentation
Test results - SWIFTTransactionTest
Test Code
file:///C:/Users/sudhi/OneDrive/Desktop/CATFE/SourceCodeTesting/SourceCode/RecordsManagementCompliance/app/build/reports/tests/test/classes/org.example....
Gmail
YouTube
Maps
Context based Testing...
All Bookmarks
Search
ENG IN
17:38 26-03-2025

SWIFTTransactionTest

all > org.example > SWIFTTransactionTest

4	1	0	0.033s
tests	failures	ignored	duration

75% successful

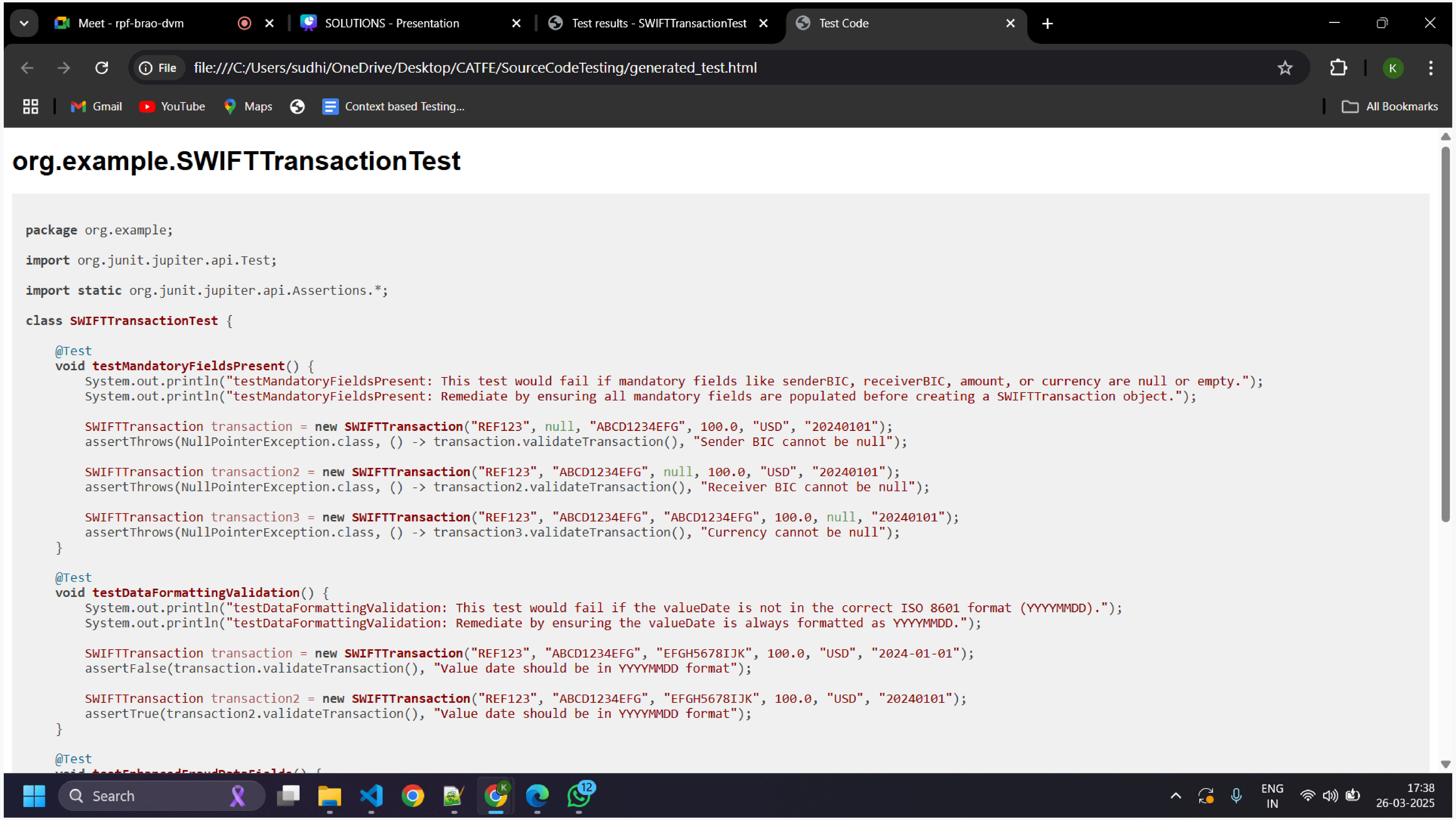
Failed tests | Tests | Standard output

```
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, AccountHolde
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.
```

Wrap lines ☐

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

CODEDEFENDER

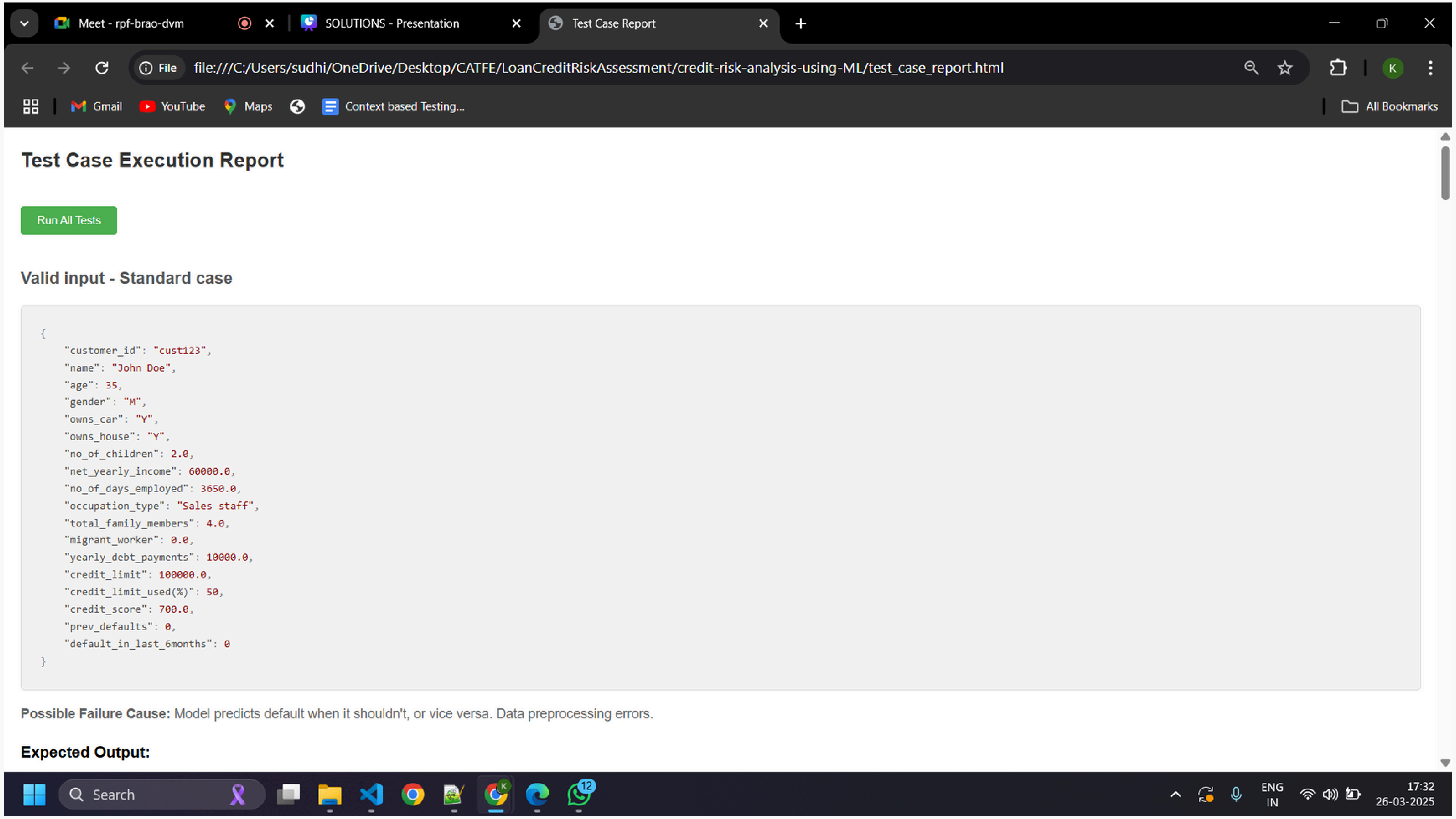


org.example.SWIFTTransactionTest

```java
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 testEnhancedFraudDataFields() {
```

Screenshot of the summary of the automatically generated source code 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

Test Case Execution Report

Run All Tests

Valid input - Standard case

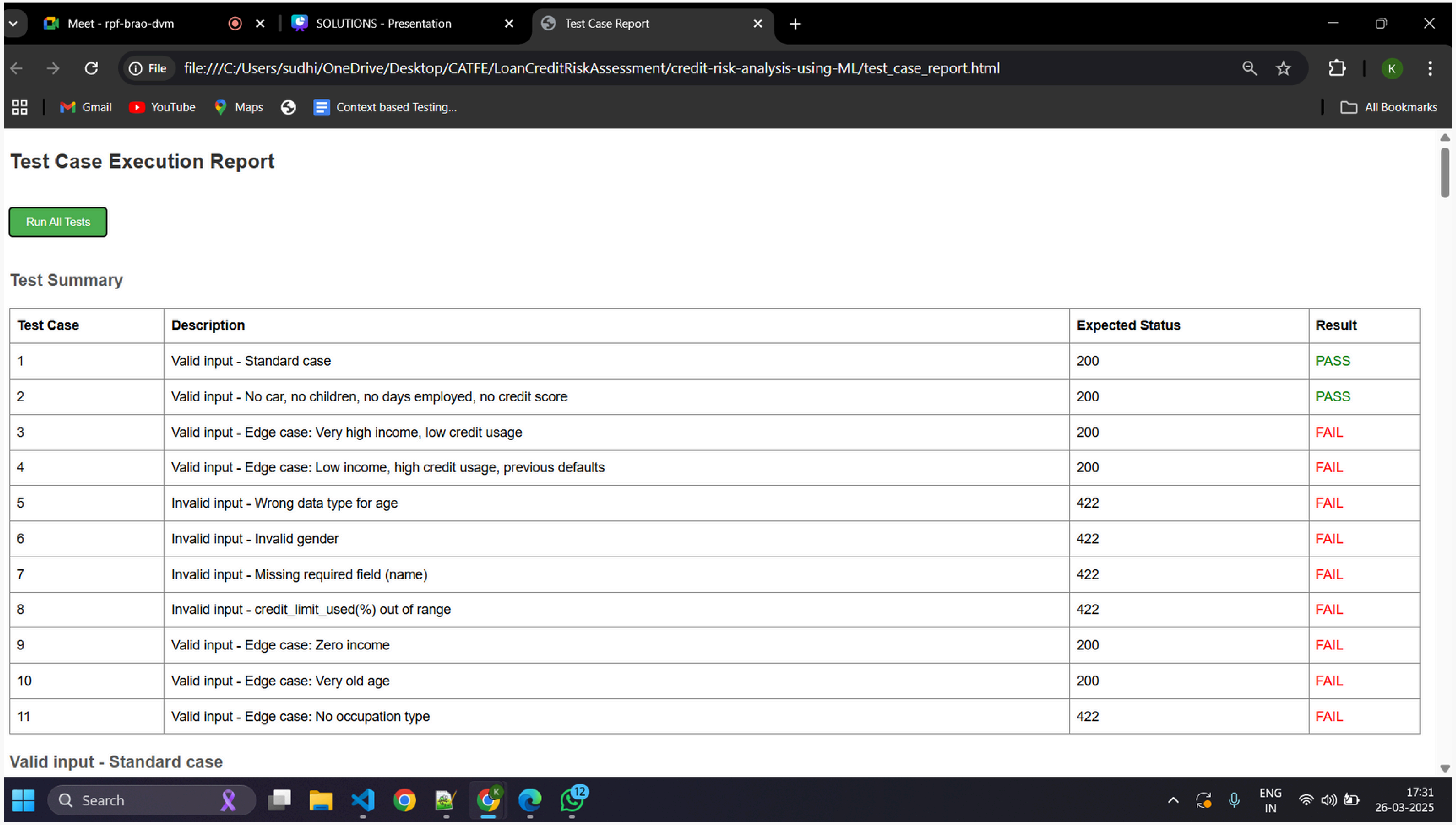
```
{
    "customer_id": "cust123",
    "name": "John Doe",
    "age": 35,
    "gender": "M",
    "owns_car": "Y",
    "owns_house": "Y",
    "no_of_children": 2.0,
    "net_yearly_income": 60000.0,
    "no_of_days_employed": 3650.0,
    "occupation_type": "Sales staff",
    "total_family_members": 4.0,
    "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



Test Case Execution Report

Run All Tests

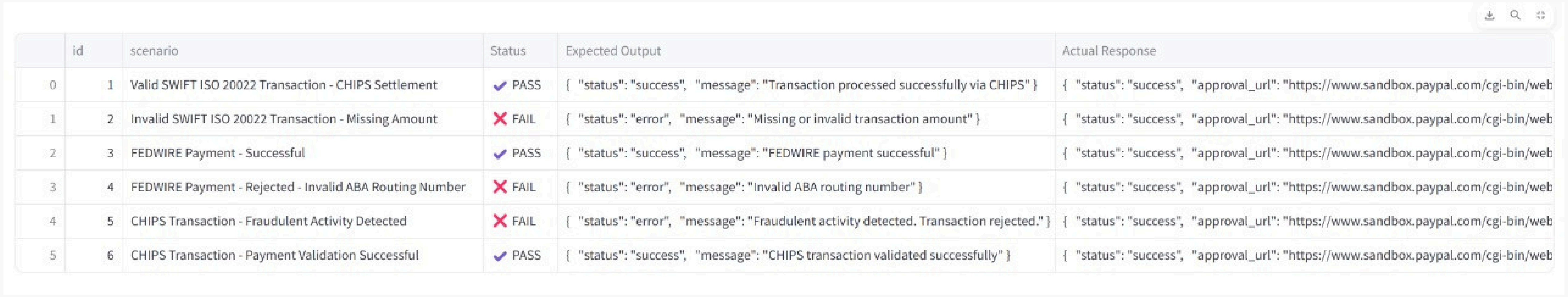
Test Summary

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

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