littlepay

Test Strategy for Littlepay Trip Processor

1. Introduction

The **Littlepay Trip Processor** is designed to calculate and generate trip reports based on tap-on and tap-off events recorded in a transit system. This document outlines the test strategy to ensure the correctness, reliability, and performance of the system.

2. Objectives

- Verify that the system correctly processes trips based on tap-on and tap-off events.
- Ensure accurate fare calculation based on predefined fare rules.
- Validate handling different scenarios, including completed, incomplete, and canceled trips.
- Ensure system reliability and performance under load.
- Verify integration with external systems and data formats (CSV input/output).

3. Scope

In Scope

- Functional Testing
- Integration Testing
- Performance Testing
- Security Testing (Basic level for input validation)
- Automation Testing using Playwright

Out of Scope

- UI Testing (since it's a backend processing system)
- End-to-End Testing with live transit data

4. Test Approach

4.1 Test Data

The test will use valid and invalid input data to validate various scenarios. Sample test data includes:

- Valid trips (tap-on and tap-off pairs at different stops)

- Incomplete trips (only tap-on, missing tap-off)
- Canceled trips (tap-on and tap-off at the same stop)
- Erroneous input data (missing values, incorrect formats)

4.2 Test Scenarios

The below provided Functional Test Cases are identified based on the exercise and details are provided in details in Test cases Doc.

ID	Scenario	Expected Outcome
TC01	Process a completed trip (tap-on and tap-off at different stops)	Trip is marked as **COMPLETED**, correct fare is calculated
TC02	Process an incomplete trip (missing tap-off)	Trip is marked as **INCOMPLETE**, maximum fare is charged
TC03	Process a canceled trip (tap-on and tap-off at the same stop)	Trip is marked as **CANCELLED**, zero fare is charged
TC04	Process multiple trips from different users	All trips are correctly categorized and calculated
TC05	Process trips with missing/invalid input values	System handles errors gracefully and logs them
TC06	Data Validation checks for all the fields in the output	Every field should be be in correspondence in the expected format
TC07	Validate the total trip count	Total trip count should be in accordance with the expected number of trips
TC08	Cost of each trip based on the PAN	Validate cost of each trip fare based on the output generated from the application

Performance Testing [Out of Scope]

- Load Test: Process a large number of transactions to check performance.
- Stress Test: Test the system under peak loads to observe failure thresholds.

Security Testing

- Validate against malicious input (e.g., SQL injection, script injection in CSV files).
- Ensure PAN (card numbers) are not stored or displayed insecurely.

4.3 Automation Strategy

- Use **Playwright** for API and data validation testing.
- Automate test cases for verifying trip status counts (`COMPLETED`, `INCOMPLETE`, `CANCELLED`).
- Automate CSV file upload and response validation.

5. Test Execution Plan

- **Unit Testing:** Developers execute unit tests as part of CI/CD.
- **Integration Testing:** Validate the interaction between different modules.
- **Automated Testing:** Run Playwright tests after every code change.
- **Manual Testing:** Edge cases and performance testing will be executed manually.
- **Regression Testing:** Automated test suite runs before releases.

6. Defect Management

- Bugs are tracked in the project's issue-tracking tool.
- Bugs are categorized based on severity (Critical, High, Medium, Low).
- Fixes are prioritized and retested before deployment.

7. Tools & Technologies

- **Programming Language:** Javascript/Typescript
- **Testing Framework:** Playwright
- **Version Control:** Git
- **CI/CD:** GitHub Actions
- **Performance Testing:** JMeter

8. Risks & Mitigation

Risk	Mitigation
Large data volume affecting performance	Implement efficient data handling and optimize queries
Missing or inconsistent tap events	Ensure proper logging and fallback calculations
Security concerns with PAN storage	Mask or hash PAN data, follow security best practices

9. Conclusion

This test strategy provides a structured approach to ensuring the accuracy, performance, and reliability of the **Littlepay Trip Processor**. Automated testing with Playwright and performance validation will help maintain a robust transit fare processing system.