Test Plan for Payment Application

1. Test Plan ID:

TP-2025-Payment-App

2. Introduction:

This test plan outlines the testing approach for the Payment Processing Module of the application, focusing on functionality,
performance, security, and compliance with PCI DSS standards.

3. Objectives:

- Validate that users can perform transactions (payments, refunds, and transfers) successfully.
- Ensure secure handling of sensitive data like credit card details.
- Verify the application performs well under high transaction loads.

4. Scope:

In-Scope:

- User login and account management.
- Adding, updating, and deleting payment methods.
- Processing payments, refunds, and balance transfers.
- Integration with third-party payment gateways (e.g., Stripe, PayPal).
- Compliance with PCI DSS for secure data handling.

Out-of-Scope:

- Non-payment-related features (e.g., user profile updates).
- Backend infrastructure monitoring.

5. Test Approach:

- Manual Testing: For exploratory and usability testing.
- Automation Testing: For regression and repetitive functional tests.
- Performance Testing: To simulate high transaction loads.
- Security Testing: To validate encryption, authorization, and data security.

6. Test Deliverables:

- Test cases and scenarios (functional, non-functional).
- Test execution reports.
- Defect reports with severity and priority.
- Final test summary report.

7. Test Environment:

Configuration:

- Environment: QA, Staging, Production-like.
- Devices: Windows, macOS, Android, iOS.
- Browsers: Chrome, Firefox, Safari, Edge.
- Database: MySQL 8.0.

Tools:

- Selenium (Automation).
- Postman (API Testing).
- JMeter (Performance).
- OWASP ZAP (Security).

8. Test Cases:

Functional:

1. Add a new credit card as a payment method.

2. Make a payment with valid card details.

3. Verify error messages for invalid card details (e.g., expired card).

Boundary:

1. Test the maximum transaction amount allowed.

2. Validate character limits for the cardholder name.

Negative:

1. Process a payment with insufficient funds.

2. Simulate a failed payment gateway response.

Performance:

1. Process 1,000 transactions in under 2 minutes.

2. Simulate 10,000 simultaneous users making payments.

Security:

1. Validate sensitive data is encrypted during transmission.

2. Test for SQL injection vulnerabilities.

9. Test Schedule:

Activity	Start Date	End Date
Test Case Preparation	Jan 15, 2025	Jan 20, 2025
Functional Testing	Jan 21, 2025	Jan 28, 2025
Performance Testing	Jan 29, 2025	Feb 2, 2025

Security Testing	Feb 3, 2025	Feb 6, 2025
Final Reporting	Feb 7, 2025	Feb 8, 2025

10. Risks:

- Delayed API Integration: Mitigation: Use mock APIs for testing.
- High Defect Volume: Mitigation: Prioritize testing critical workflows.

11. Entry and Exit Criteria:

Entry Criteria:

- Requirements are signed off.
- Test environment is set up.
- All dependencies (APIs, databases) are functional.

Exit Criteria:

- All critical and high-severity defects are resolved.
- 95% of test cases are executed, with 90% passing.

12. Team Roles:

- QA Lead: Oversees testing strategy and reporting.
- Test Engineers: Write and execute test cases.
- Automation Engineer: Develops regression scripts.
- Performance Tester: Conducts load testing.