Key Components of a Test Plan Document:

1. **Introduction**
    - Purpose of the document
    - Scope of testing
    - Testing objectives
2. **Test Items**
    - Features to be tested
    - Features not to be tested
3. **Test Environment**
    - Hardware, software, and tools required for testing
    - Test data requirements
4. **Test Approach**
    - Testing levels (unit, integration, system, acceptance)
    - Testing types (functional, performance, security, etc.)
    - Manual vs. automated testing
5. **Entry and Exit Criteria**
    - Conditions required to start and stop testing
6. **Test Deliverables**
    - Test cases, test scripts, test data
    - Test summary reports
7. **Responsibilities**
    - Roles and responsibilities of team members
8. **Schedule**
    - Testing timelines
    - Milestones
9. **Risk Management**
    - Identified risks and mitigation strategies
10. **Approval**
    - Sign-off from stakeholders

Related Documents:

Other documents that may complement or support the test plan include:

- **Requirements Specification Document (RSD)**: Defines the functional and non-functional requirements of the system.
- **Test Case Specification Document**: Lists individual test cases, their steps, expected results, and actual results.
- **Test Strategy Document**: A high-level document outlines the testing approach for the entire project.
- **Traceability Matrix**: Maps requirements to test cases to ensure coverage.

Test Plan for Microservice Application: Student Registration with PayPal Integration

1. Introduction

This test plan outlines the strategy, scope, objectives, and testing procedures for a microservice-based Student Registration system built using Spring Boot, with a PayPal payment integration. The goal is to ensure the system functions as expected, providing smooth user registration and payment services while maintaining data security.

---

2. Scope of Testing

In-Scope:

- Student Registration (CRUD Operations).
- PayPal Payment Integration:
    - Payment creation.
    - Payment execution.
    - Payment cancellation and rollback scenarios.
- Communication between microservices.
- Security of sensitive data (e.g., payment details).
- Performance testing under various loads.

Out-of-Scope:

- PayPal API behavior (handled by PayPal itself).
- UI/Frontend testing (if the focus is backend).

---

3. Test Objectives (Improved):

1. Validate the Student Registration CRUD Operations:

    - Ensure that the system correctly handles student data across all CRUD operations (Create, Read, Update, Delete) and ensures data integrity in both normal and edge cases (e.g., duplicate data, invalid input).
    - Verify that data validation and error handling for the student registration process are robust and user-friendly.
2. Test the PayPal Payment Process (End-to-End):
    - Validate that the PayPal payment gateway integration works seamlessly from the initiation of the payment to the completion, including handling various transaction scenarios like successful payment, payment failure, and cancellations.
    - Ensure the system properly communicates with PayPal's APIs, handles payment statuses correctly, and provides accurate feedback to users.
3. Verify Integration Between Registration and Payment Services:
    - Ensure that the registration and payment services are tightly integrated, allowing for smooth data flow between the two systems (e.g., successful registration triggering the payment process and vice versa).
    - Test for possible failure scenarios, including payment failure or registration issues, and confirm that the system can rollback or recover appropriately without leaving inconsistent data.
4. Ensure Secure Handling of Sensitive Data:
    - Verify that sensitive information, such as student data and payment details, is securely transmitted and stored, in compliance with security best practices (e.g., encryption, tokenization, PCI-DSS compliance).
    - Test for potential security vulnerabilities, including SQL injection, cross-site scripting (XSS), and other common attack vectors.
5. Confirm System Reliability Under Normal and Peak Conditions:
    - Conduct performance testing to ensure the system can handle both normal and peak traffic volumes, particularly during registration periods (e.g., during a new term or special promotion).
    - Test the system's scalability by simulating different loads and identifying bottlenecks, ensuring the application remains responsive and functional under stress.
6. Ensure User Experience and Transaction Transparency:
    - Validate that the entire student registration and payment process provides clear, intuitive, and seamless user interactions.
    - Test for clear feedback messages, proper redirections, and appropriate error messages during both successful and failed transactions to ensure a positive user experience.

---

4. Test Items

Modules to Test:

1. **Student Registration Microservice**
    - Create, Read, Update, and Delete (CRUD) operations for student data.
2. **Payment Service**
    - Payment creation, execution, and cancellation through PayPal.
3. **Integration Testing**
    - Registration + Payment end-to-end flow.
    - Error handling for payment failures.

---

5. Test Environment

- **Backend Framework**: Spring Boot
- **Database**: MySQL or PostgreSQL
- **Payment Gateway**: PayPal Sandbox
- **Tools**: Postman, JUnit, Mockito, RestAssured
- **Environment**: Local Development Server or Dockerized Setup

---

6. Test Approach

Testing Types:

1. **Unit Testing**: Validate individual components of the registration and payment microservices.
    - Tools: JUnit, Mockito.
2. **Integration Testing**: Test communication between registration and payment services and PayPal API.
    - Tools: RestAssured.
3. **System Testing**: Verify the complete functionality of the system.
4. **Performance Testing**: Measure API performance under different loads.
    - Tools: JMeter.
5. **Security Testing**: Ensure sensitive data (like payment credentials) is encrypted.

---

7. Entry and Exit Criteria

Entry Criteria:

- All services are deployed locally.
- Sample test data is seeded in the database.
- PayPal Sandbox credentials are configured.

Exit Criteria:

- All planned test cases are executed.
- All critical defects are resolved.
- The system meets performance benchmarks.

8.1 Student Registration Service:

Test Case ID: TC-001

- **Test Scenario:** Create Student
- **Pre-Condition:**
    - The user is authenticated (if required).
    - The student registration page is available.
- **Steps:**
    - A user submits valid student registration data (e.g., name, email, course, etc.) via the registration form.
    - The system processes this information and creates a new student entry in the database.
- **Expected Result:** The student is successfully added to the system, and their details are saved in the database.
- **Post-Condition:**
    - The student's details are stored in the database.
    - The student can be retrieved via their unique ID.

Test Case ID: TC-002

- **Test Scenario:** Update Student
- **Pre-Condition:**
    - The student exists in the system.
    - The user is authenticated and authorized to modify the student's information.
- **Steps:**
    - A user retrieves the student's profile by providing the student's unique ID.
    - The user updates their student data (e.g., changing course, updating contact details).
    - The system processes the update and saves the modified student data.
- **Expected Result:** The student's data is successfully updated in the system, and the changes are reflected in the database.
- **Post-Condition:**

    - The updated student data is saved in the database.
    - The student's profile has been updated with new information.

Test Case ID: TC-003

- **Test Scenario:** Delete Student
- **Pre-Condition:**
    - The student exists in the system.
    - The user is authenticated and authorized to delete the student's profile.
- **Steps:**
    - A user selects the student's profile they wish to remove from the system.
    - The user confirms the action to delete the student from the database.
    - The system deletes the student's data from the database.
- **Expected Result:** The student is successfully removed from the system, and their details are deleted from the database.
- **Post-Condition:**
    - The student is no longer available in the system.
    - The student's details are removed from the database.

Test Case ID: TC-004

- **Test Scenario:** Retrieve Student
- **Pre-Condition:**
    - The student exists in the system.
    - The user has access to view student data.
- **Steps:**
    - Users request to view a student's details by entering their unique student ID.
    - The system fetches and displays the student's details from the database.
- **Expected Result:** The correct student information is retrieved and displayed accurately.
- **Post-Condition:**
    - The system successfully returns the student's details.

8.2 Payment Service:

Test Case ID: TC-101

- **Test Scenario:** Create Payment
- **Pre-Condition:**
    - The student has been registered.
    - The user is authenticated and authorized to make payments.
- **Steps:**
    - A user initiates the payment process by clicking the "Pay" button.
    - The system redirects the user to PayPal and generates a payment link.

    - The user is directed to the PayPal payment page to complete the transaction.
- **Expected Result:** A valid payment link is generated, and the user is able to proceed to PayPal for payment.
- **Post-Condition:**
    - A payment link is created and is ready for the user to complete the transaction.

Test Case ID: TC-102

- **Test Scenario:** Execute Payment
- **Pre-Condition:**
    - The user has initiated the payment process.
    - The user is on the PayPal payment page and has provided valid payment details.
- **Steps:**
    - The user submits payment credentials and authorizes the payment on PayPal.
    - The system verifies PayPal's payment status.
    - The system marks the payment as successful if PayPal confirms.
- **Expected Result:** The payment is completed, verified, and marked as successful in the system.
- **Post-Condition:**
    - The payment is marked as completed in the system.
    - The transaction is logged in the payment history.

Test Case ID: TC-103

- **Test Scenario:** Cancel Payment
- **Pre-Condition:**
    - The user has initiated the payment process and is using PayPal.
    - The user decides to cancel the payment before completion.
- **Steps:**
    - The user decides to cancel the payment process via PayPal.
    - The system receives a cancellation notification from PayPal.
    - The system updates the payment status to "Cancelled."
- **Expected Result:** The payment was canceled successfully, and the status was updated in the system.
- **Post-Condition:**
    - The payment status is updated to "Cancelled."
    - The transaction is logged as canceled.

8.3 Integration Testing:

Test Case ID: TC-201

- **Test Scenario:** Register Student and Pay

- **Pre-Condition:**
    - The student registration form is available.
    - The user has entered valid student data.
- **Steps:**
    - A user completes the student registration process by providing all necessary information.
    - Once registration is completed, the user is redirected to the payment gateway.
    - The user initiates the payment through PayPal.
    - After payment completion, the system confirms registration and payment success.
- **Expected Result:** The student is successfully registered, and the payment is successfully processed.
- **Post-Condition:**
    - The student's registration is confirmed.
    - The payment status is marked as successful.

Test Case ID: TC-202

- **Test Scenario:** Payment Failure Rollback
- **Pre-Condition:**
    - The student registration is initiated.
    - The system is connected to PayPal for payment processing.
- **Steps:**
    - A user starts the registration process.
    - The user is redirected to PayPal for payment, but an error occurs, causing the payment to fail.
    - The system detects the payment failure and rolls back the student registration.
- **Expected Result:** The registration process is canceled, and no student data is saved in the system due to the failed payment.
- **Post-Condition:**
    - No student data is saved.
    - The system logs the failure and ensures no incomplete registration records exist.

Test Case ID: TC-203

- **Test Scenario:** Invalid PayPal Response
- **Pre-Condition:**
    - The user is registered and ready to make a payment.
    - The system is connected to PayPal and is waiting for a valid response.
- **Steps:**
    - PayPal sends an invalid or incorrect response (e.g., failure to confirm payment status).
    - The system detects the invalid response and displays an error message to the user.

- **Expected Result:** An error message is displayed, and the user is informed that there was an issue with processing the payment.
- **Post-Condition:**
    - No payment is processed.
    - The system logs the error and ensures the payment status is correctly marked as failed.

9. Test Deliverables

- Test Cases Document.
- Test Data and Scripts.
- Test Execution Report.
- Defect Reports (if any).

---

10. Responsibilities

Role	Responsibility
Test Lead	Oversees test execution.
Developer	Fixes issues and retests.
Tester	Writes and executes test cases.

---

11. Schedule

Activity	Start Date	End Date
Test Planning	YYYY-MM-DD	YYYY-MM-DD
Test Case Development	YYYY-MM-DD	YYYY-MM-DD
Test Execution	YYYY-MM-DD	YYYY-MM-DD
Bug Fixing & Retesting	YYYY-MM-DD	YYYY-MM-DD

---

12. Risks and Mitigation

Risk	Mitigation
PayPal API downtime	Use PayPal sandbox environment.
Sensitive data exposure	Encrypt all sensitive data.
Integration failures	Test thoroughly with mock APIs.
