Test Plan for Student Registration System with Payment Module

Table of Contents

1. Introduction

The purpose of this test plan is to ensure the functionality, reliability, and performance of the Student Registration System, including its Payment Module. This document defines the testing strategy, scope, and approach to validate that the system meets the specified requirements.

2. Scope

The scope of testing includes the following modules:

- **Student Registration Module:** Registration of students with personal, academic, and login details.
- **Payment Module:** Processing payments via multiple methods (credit/debit card, net banking, UPI, etc.) and generating invoices.

- **Integration:** Ensuring seamless integration between the registration and payment modules.

Out of scope: Testing of external payment gateway services.

3. Objectives

- Validate that the student registration functionality works as expected.
- Verify the accuracy and security of payment processing.
- Ensure smooth integration between the modules.
- Confirm that the system meets usability, performance, and security requirements.

4. Test Items

- Registration Form
- Student Profile Management
- Payment Gateway Integration
- Payment Confirmation and Invoicing
- Error Handling and Validation
- Reports and Logs

5. Test Environment

Hardware Requirements:

- Server: 16 GB RAM, Quad-core processor, 500 GB storage.
- Client: 8 GB RAM, Dual-core processor, 20 GB free storage.

Software Requirements:

- OS: Windows/Linux/MacOS
- Browser: Chrome, Firefox, Safari, Edge

- Database: MySQL/PostgreSQL
- Payment Gateway: Sandbox environment for testing (e.g., Stripe, PayPal)

6. Test Scenarios

Student Registration Module

1. Validate mandatory field checks.
2. Test form submission with valid and invalid data.
3. Verify account creation and confirmation email.
4. Check student profile updates.
5. Ensure duplicate registrations are prevented.

Payment Module

1. Validate multiple payment options (credit card, debit card, UPI, etc.).
2. Test payment processing with valid and invalid details.
3. Verify refund and cancellation scenarios.
4. Ensure payment confirmation and receipt generation.
5. Test error handling for failed transactions.

Integration Testing

1. Verify student registration triggers payment process.
2. Ensure seamless transition from registration to payment.
3. Validate post-payment updates to student profiles.

Security Testing

1. Test SQL injection vulnerabilities.
2. Ensure sensitive data is encrypted (e.g., passwords, card details).
3. Verify session management and timeout features.
4. Validate compliance with data protection regulations (e.g., GDPR, PCI DSS).

Performance Testing

1. Test response time for registration and payment.
2. Verify system performance under load (e.g., 1000 concurrent users).
3. Identify bottlenecks during peak usage.

7. Test Cases

Sample Test Cases

ID	Test Case	Steps	Expected Result
TC001	Validate mandatory fields in registration form	1. Open registration form.2. Leave all fields blank.3. Submit.	Error messages displayed for all mandatory fields.
TC002	Verify successful student registration	1. Fill form with valid data.2. Submit form.	Account created; confirmation email sent.
TC003	Test payment with valid credit card	1. Select credit card option.2. Enter valid details.3. Submit.	Payment successful; receipt generated.
TC004	Test payment with invalid card details	1. Select credit card option.2. Enter invalid details.3. Submit.	Error message displayed; payment not processed.
TC005	Verify refund for canceled payment	1. Process payment.2. Request refund.3. Confirm refund status.	Refund processed; status updated.
TC0	Test SQL injection vulnerability in registration form	1. Enter SQL commands in input fields.2. Submit form.	Input sanitized; error message displayed.

06			

8. Tools and Resources

- **Automation Tools:** Selenium, JMeter
- **Bug Tracking:** JIRA, Bugzilla
- **Test Management:** TestRail, Zephyr
- **Database Tools:** MySQL Workbench, pgAdmin
- **Payment Gateway Sandboxes:** PayPal Sandbox, Stripe Test Environment

9. Schedule

- **Test Planning:** 1 week
- **Test Case Design:** 2 weeks
- **Test Execution:** 3 weeks
- **Bug Fixing and Retesting:** 2 weeks
- **Final Verification:** 1 week

10. Risks and Assumptions

Risks:

- Delays in payment gateway sandbox setup.
- Unavailability of testing resources.

Assumptions:

- All requirements are finalized and approved.
- Test environment is fully configured before execution.

Approval

Prepared By: [KIROS GEBREMEDHN]

Approved By: [INST MESSELE]

Date: [08/05/2017 E.C]

ID UGR/ 170100/12