# MEKELLE UNIVERSITY

# DEPARTMENT OF SOFTWAR ENGINEERING

# TEST PLAN FOR STUDENTS REGISTRATION SYSTEM

# By;

Daniel kindeya g/HER

Ugr/179057/12

**Test Plan for Laravel Student Registration System**

## 1. Introduction

This test plan describes the comprehensive approach to testing the Laravel-based Student Registration System. The system allows students to register, browse courses, enroll in courses, and securely pay via Stripe payment gateway. The backend uses an SQLite database to manage user and transaction data efficiently.

Stakeholders:  
- Development Team: Responsible for implementing the system.  
- Testing Team: Ensures the functionality and reliability of the application.  
- Project Manager: Oversees the project and ensures timelines are met.  
- End Users: Students who interact with the system.

## 2. Scope of Testing

The testing focuses on verifying the following functionalities:  
- Student registration and login.  
- Browsing the course catalog and viewing course details.  
- Enrolling in one or more courses.  
- Processing payments using the Stripe payment gateway.  
- Recording enrollment and payment transactions.

Out of scope:  
- Multi-role support (only the student role is implemented).  
- Integration with other payment gateways.

## 3. Objectives

**Ensure Core Functionalities Work as Intended:**

* Verify that the student registration process captures and stores user data accurately, ensuring successful creation of accounts.
* Confirm that the login mechanism, including authentication and authorization, functions securely and reliably.
* Validate the course browsing and enrollment processes, ensuring students can view detailed course information and select courses for registration.
* Test payment processing through the Stripe gateway to ensure secure, efficient, and error-free transactions.

**Security Validation:**

* Ensure sensitive data, such as passwords and payment information, is encrypted during storage and transmission.
* Validate proper implementation of user roles, ensuring restricted access to critical functions.

**Usability and User Experience:**

* Evaluate the system's user interface to ensure an intuitive and user-friendly experience for students.
* Assess the clarity and accessibility of error messages, notifications, and guidance during the user journey.

**Data Integrity and Accuracy:**

* Confirm that user and transaction records are accurately stored and retrieved from the SQLite database.
* Validate database constraints, such as unique identifiers for users and proper relationships between data entities, ensuring consistency and reliability.

**System Performance and Scalability:**

* Measure response times for critical operations, such as registration, login, and payment processing, ensuring the system performs efficiently under expected workloads.
* Test the system’s ability to handle increased user loads without degradation in performance or functionality.

**Error Handling and Recovery:**

* Verify that the system handles unexpected inputs or actions gracefully, displaying clear error messages and preventing crashes.
* Test fallback mechanisms to ensure smooth recovery in case of system failures or errors, minimizing disruption to the user experience.

**Stakeholder Confidence:**

* Deliver a thoroughly tested system that meets the expectations of stakeholders, including developers, testers, and end-users.
* Provide evidence-based assurance through defect logs, test coverage reports, and detailed summaries of testing outcomes.

## **4. Test Strategy**

**Levels of Testing:**  
- Unit Testing: Test individual components such as registration and payment modules.  
- Integration Testing: Validate interactions between modules, especially Stripe integration.  
- System Testing: Test the system end-to-end.  
- Acceptance Testing: Verify that the system meets business and functional requirements.

**Testing Types:**  
- Functional Testing: Verify all functional aspects of the system.  
- Security Testing: Ensure payment data is processed securely.  
- Performance Testing: Assess response times for key operations.  
- Usability Testing: Ensure the system is intuitive and user-friendly.

## **5. Test Cases**

### **1. User Registration and Authentication**

#### ****Test Case 1.1: Successful Registration****

* **Description:** Test if a student can successfully register with valid details.
* **Steps:**
  1. Navigate to the registration page.
  2. Enter valid name, email, and password.
  3. Submit the registration form.
* **Expected Result:** Student account is created, and a confirmation message is displayed.

#### ****Test Case 1.2: Email Validation****

* **Description:** Ensure email addresses are validated during registration.
* **Steps:**
  1. Enter invalid email formats (e.g., "user@domain").
  2. Enter email starts with number (“[122@gmail.com](mailto:122@gmail.com)”)
  3. Submit the registration form.
* **Expected Result:** An error message is displayed: "Invalid email address."

#### ****Test Case 1.3: Login with Valid Credentials****

* **Description:** Test login functionality with registered email and password.
* **Steps:**
  1. Navigate to the login page.
  2. Enter valid email and password starts with alphabet
  3. Submit the login form.
* **Expected Result:** Login is successful, and the student is redirected to the dashboard.

#### ****Test Case 1.4: Login with Invalid Credentials****

* **Description:** Test login with incorrect email or password.
* **Steps:**
  1. Enter invalid email or password.
  2. Enter not registered username and password.
  3. Submit the login form.
* **Expected Result:** An error message is displayed: "Invalid email or password."

### **2. Course Management**

#### ****Test Case 2.1: View Courses****

* **Description:** Ensure students can view the course catalog.
* **Steps:**
  1. Log in to the system.
  2. Navigate to the course catalog.
* **Expected Result:** A list of available courses is displayed, showing title, code, description, credit hours, and fee.

#### ****Test Case 2.2: Course Details****

* **Description:** Check if detailed course information is displayed correctly.
* **Steps:**
  1. Select a course from the catalog.
* **Expected Result:** Details of the selected course (title, code, description, credit hours, and fee) are displayed.

### **3. Enrollment Process**

#### ****Test Case 3.1: Select Courses for Enrollment****

* **Description:** Test if students can select courses for enrollment.
* **Steps:**
  1. Log in to the system.
  2. Browse the course catalog.
  3. Select one courses.
* **Expected Result:** Selected courses are added to the enrollment cart.

#### ****Test Case 3.2: Confirm Enrollment****

* **Description:** Ensure students can confirm enrollment by making a payment.
* **Steps:**
  1. Select courses.
  2. Proceed to the payment page.
  3. Confirm the payment.
* **Expected Result:** Enrollment is confirmed, and the student receives a success message.

### **4. Payment Processing**

#### ****Test Case 4.1: Successful Payment****

* **Description:** Test successful payment via Stripe.
* **Steps:**
  1. Select courses.
  2. Navigate to the payment page.
  3. Enter valid payment details.
  4. Complete the payment.
* **Expected Result:** Payment is processed successfully, and a confirmation email is sent to the student.

#### ****Test Case 4.2: Payment Failure****

* **Description:** Handle payment failure due to invalid card details or insufficient funds.
* **Steps:**
  1. Enter invalid payment details (e.g., expired card).
  2. Attempt to complete the payment.
* **Expected Result:** An error message is displayed: "Payment failed. Please try again."

#### ****Test Case 4.3: Payment Record Storage****

* **Description:** Verify that payment details are stored in the database.
* **Steps:**
  1. Complete a successful payment.
  2. Check the database for stored transaction details.
* **Expected Result:** Payment details (transaction ID, timestamp) are stored.

### **5. Enrollment History**

#### ****Test Case 5.1: View Enrollment History****

* **Description:** Test if students can view their enrollment history.
* **Steps:**
  1. Log in to the system.
  2. Navigate to the enrollment history page.
* **Expected Result:** A list of previously enrolled courses is displayed.

### **6. System Constraints**

#### ****Test Case 6.1: Database Limitations****

* **Description:** Test system behavior when the SQLite database reaches its storage limit.
* **Steps:**
  1. Simulate maximum database capacity.
  2. Attempt to register a new student or process a payment.
* **Expected Result:** The system gracefully handles the error and displays an appropriate message.

## **6. Test Deliverables**

1. Test Cases and Scripts.  
2. Test Data (including valid, invalid, and edge-case scenarios).  
3. Defect Logs and Issue Reports.  
4. Test Summary Report with coverage metrics, pass/fail rates, and identified risks.

## **7. Entry and Exit Criteria**

**Entry Criteria:**  
- All system components are developed and unit tested.  
- Test environment is set up and stable.  
- Test data is ready.

**Exit Criteria:**  
- All critical test cases pass.  
- No high-severity defects remain unresolved.  
- Final sign-off from stakeholders is obtained.

## **8. Resources**

**Team:**  
- QA Engineer: Execute test cases and log defects.  
- Developers: Fix defects.  
  
**Tools:**  
- PHPUnit for unit testing.  
- Postman for API testing.  
- Selenium or Laravel Dusk for automated UI testing.  
- SQLite database management tools.

## 9. Schedule

1. Test Plan Preparation: 2 day  
2. Unit Testing: 1 week  
3. Integration Testing: 4 days  
4. System Testing: 1 week  
5. Defect Resolution and Final Review: 1 Week

## 9. Risk Management

Potential Risks:  
- Stripe API downtime during testing.  
- Issues with SQLite database scalability under load.  
- Environment configuration discrepancies.  
  
Mitigation Strategies:  
- Use Stripe’s sandbox environment to simulate transactions.  
- Validate database configurations and perform stress testing.  
- Standardize test environment setup across teams.