MEKELLE UNVIERSITY



**EITM**

# **SCHOOL OF COMPUTING**

## **DEPARTMENT OF SOFTWARE ENGINEERING**

**TEST PLAN OF**

**STUDENT REGISTRATION SYSTEM**

#### Name: Semere Herruy

* ID: ugr/177912/12

Submission date: 08/02/2017 E.c

Submitted to : Inst. Mesele

### **TESTING ANALYSIS AND DESIGN**

#### ****2.1 TEST ANALYSIS****

##### **2.1.1 OBJECTIVES:**

The objectives of the test analysis phase are to:

1. Identify testable requirements from the SRS document.
2. Define the scope and boundaries of testing based on system components.
3. Establish test conditions and criteria for success.
4. Ensure all functional and non-functional requirements are covered in the test scenarios.

##### **2.1.2 ACTIVITIES:**

1. **Requirement Analysis:**
   * Analyze functional requirements, such as course registration, payment integration, and session management.
   * Assess non-functional requirements like performance, scalability, and security.
2. **Risk Assessment:**
   * Identify high-risk areas, such as payment security and session reliability.
   * Prioritize testing for critical components, including API Gateway and Payment Service.
3. **Defining Test Objectives:**

* Validate user scenarios like:
* Student login/future consideration/ and account creation
* Registration process (validating user input such as name, email, phone number)
* Course registration and payment process
* Chapa gateway interaction and successful payment.
* Test session handling during high-traffic scenarios to avoid system downtime.

1. **Mapping Requirements to Test Cases:**
   * Create traceability between system requirements and corresponding test cases.

#### ****2.2 TEST DESIGN****

##### **2.2.1 OBJECTIVES:**

The objectives of the test design phase are to:

1. Develop detailed test cases and scripts based on identified test conditions.
2. Design test data to cover normal, edge, and error scenarios.
3. Prepare the testing environment and establish necessary configurations.

##### **2.2.2 ACTIVITIES:**

1. **Test Case Development:**

* Create test cases for each system feature, ensuring comprehensive coverage:
* **Student Registration**: Testing the input fields for name, email, and phone number. Include validation for format, non-emptiness, and boundary cases.
* **Course Registration**: Validate course details (name, description, amount).
* **Payment Processing**: Test valid and invalid payment scenarios, including interactions with the Chapa gateway.
* Include test cases for edge cases (e.g., invalid inputs like a missing email, phone number format issues, and payment failures).

1. **Test Data Design:**

* Develop datasets for valid, invalid, and boundary conditions:
* **Valid Data**:
  + - Valid student credentials (e.g., correct name, email, and phone number format)
    - Correct course details (e.g., course name, description, and amount)
    - Valid payment information (e.g., card number, CVV, and expiry date)
* **Invalid Data**:
  + Missing required fields (e.g., missing email or payment details)
  + Invalid email or phone format
  + Payment gateway failures
* **Boundary Conditions**:
* Maximum number of course registrations
* Maximum session duration

1. **Test Environment Setup:**
   * Configure a **staging environment** to simulate production, including the **API Gateway**, **Payment Service**, **Session Store**, and the **Database**.
   * Integrate tools and libraries required for testing.
2. **Review and Validation:**
   * Peer-review test cases to ensure completeness and accuracy.
   * Validate test cases against requirements for traceability.

**2.2.3 DELIVERABLES:**

1. **Test Case Documentation**:
   * Detailed test case descriptions, including:
     + **Test Case ID**
     + **Test Case Description**
     + **Test Steps**
     + **Expected Results**
     + **Preconditions** (e.g., logged-in student)
     + **Postconditions**
     + **Priority** (Critical/High/Medium/Low)
2. **Test Data**:
   * Sample datasets for valid, invalid, and boundary scenarios.
3. **Test Scripts**:
   * Automated test scripts for functional and performance testing.
4. **Test Environment Report**:
   * Documentation of the testing environment setup and configurations.

##### **2.2.4 TOOLS FOR ANALYSIS AND DESIGN:**

1. **Test Case Management:**
   * JIRA or TestRail for organizing test cases and mapping them to requirements.
2. **Automation Tools:**
   * Selenium for functional testing.
   * JMeter for performance and load testing.
3. **Database Tools:**
   * MySQL Workbench for managing and preparing test data.
4. **Version Control:**
   * GitHub for storing and versioning test scripts and related files.