

MEKELLE UNIVERSITY COLLEGE EIT\_M

SCHOOL OF COMPUTING

ASSESSMENT Test planning documentation

NAME Tewelde Gebre

ID ugr/178430/12

Test Planning Document for Student Registration System

1. Introduction

Overview of the Project:

The Student Registration System is designed to facilitate the registration process for students enrolling in various courses. It includes features for user registration, course management, payment processing, and reporting.

Testing Goals:

To ensure the system is reliable, user-friendly, and meets all functional and non-functional requirements before its release.

Stakeholders and Roles:

Project Manager: Oversees the project and coordinates between teams.

Product Owner: Defines product vision and requirements.

Test Manager: Responsible for planning and executing the testing process.

Test Engineers: Execute test cases and report defects.

Developers: Address defects reported during testing.

2. Scope of Testing

Features to be Tested:

User registration and authentication

Course search, enrollment, and management

Payment processing (tuition fees)

User role functionalities (student, administrator, instructor)

Reporting capabilities (enrollment statistics, financial reports)

Features Not to be Tested:

Third-party integrations (external payment gateways)

Non-functional aspects not specified (e.g., specific hardware configurations)

Certainly! Here's a more detailed specification for the test objectives in the \*\*Test Planning Document\*\*:

3. Test Objectives (Specification)

The test objectives specify the key goals that the testing process aims to achieve in order to ensure the software meets its requirements and is of high quality. These objectives are focused on verifying both functional and non-functional aspects of the Student Registration System.

3.1 Functional Validation

- \*\*ObjectiveEnsure that the software meets all functional requirements as defined in the project documentation.

- \*\*User Registration and Authentication\*\*: Validate that users can successfully register and log in with valid credentials. Ensure that invalid credentials are rejected, and appropriate error messages are shown.

- \*\*Course Management\*\*: Verify that students can search for courses, enroll in them, and view course details. Ensure that administrators can add, update, or remove courses.

- \*\*Payment Processing\*\*: Confirm that the payment system correctly handles tuition fee transactions, including verifying payment success/failure, correct fee calculation, and handling of payment errors.

- \*\*Role-based Access Control\*\*: Ensure that the system correctly handles different user roles (student, instructor, administrator), enforcing permissions and restricting access to appropriate functionalities.

- \*\*Reporting\*\*: Test that enrollment statistics, financial reports, and other relevant data are generated accurately and can be accessed by the authorized users.

### 3.2 \*\*Performance Testing\*\*

- \*\*Objective\*\*: Validate the system's performance under expected usage conditions to ensure scalability and responsiveness.

- \*\*Load Testing\*\*: Simulate high numbers of concurrent users (students registering and paying) to ensure the system can handle the expected traffic without degradation in performance.

- \*\*Stress Testing\*\*: Assess the system’s performance when pushed beyond expected user limits to identify potential points of failure and the system’s ability to recover.

- \*\*Response Time\*\*: Ensure that the system performs operations (such as course search, registration, and payment) within acceptable time limits.

### 3.3 \*\*Security Testing\*\*

- \*\*Objective\*\*: Ensure that sensitive data, such as personal student information and payment details, is securely protected.

- \*\*Data Protection\*\*: Validate that all personal data entered into the system is encrypted and stored securely.

- \*\*Transaction Security\*\*: Ensure that payment transactions are processed securely, with appropriate encryption and validation mechanisms in place to prevent fraud or data leakage.

- \*\*Authentication & Authorization\*\*: Verify that user credentials are handled securely, including correct password hashing, protection against brute-force attacks, and role-based access control.

### 3.4 \*\*Usability and User Experience Testing\*\*

- \*\*Objective\*\*: Confirm that the system is intuitive, user-friendly, and meets the expectations of its intended users.

- \*\*Ease of Use\*\*: Ensure the user interface is intuitive, with clear navigation and minimal training required for users to accomplish their goals (e.g., course registration, payment processing).

- \*\*Error Handling and Feedback\*\*: Validate that the system provides helpful, clear error messages and feedback to users during interactions, such as invalid inputs or system failures.

- \*\*Mobile and Cross-Browser Compatibility\*\*: Ensure the system works correctly on a variety of devices (e.g., mobile phones, tablets) and across popular web browsers.

### 3.5 \*\*Regression Testing\*\*

- \*\*Objective\*\*: Ensure that new changes and bug fixes do not introduce new defects or impact existing functionality.

- \*\*Feature Validation\*\*: After each round of code changes, validate that core system features (e.g., registration, payment processing) continue to work as expected.

- \*\*Automated Regression Suite\*\*: Implement automated test scripts for key functionalities (e.g., user login, course enrollment) to quickly check for regressions after updates.

### 3.6 \*\*Compliance and Legal Testing\*\*

- \*\*Objective\*\*: Verify that the system complies with relevant regulations and legal requirements.

- \*\*Data Privacy\*\*: Confirm that the system meets the applicable data privacy laws (e.g., GDPR) and handles student data in accordance with regulations.

- \*\*Payment Compliance\*\*: Ensure that the payment system adheres to legal standards for processing payments (e.g., PCI-DSS compliance).

3.7 Defect Detection and Quality Assurance

- Objective: Identify and resolve defects early in the development lifecycle to improve the software quality and reduce risk.

- Defect Logging and Tracking:Ensure that all defects are properly logged and tracked using the test management tools (e.g., JIRA) and that critical defects are addressed promptly.

- Early Defect Detection: Focus on detecting defects during unit and integration testing phases, so they can be resolved before they become harder to fix in later testing stages.

These objectives help ensure that the system is fully functional, secure, high-performance, and user-friendly, while also adhering to necessary compliance standards.4. Test Strategy

Levels of Testing:

Unit Testing: Conducted by developers for individual components.

Integration Testing: Testing interactions between modules (e.g., user registration and payment processing).

System Testing: End-to-end testing of the entire application.

User Acceptance Testing (UAT): Validation of the system by end-users to confirm it meets their needs.

Testing Types:

Manual Testing: For exploratory and usability testing.

Automated Testing: For regression tests using tools like Selenium.

Load Testing: To assess system performance under high user loads.

Test Environment Requirements:

Staging environment that mirrors production settings.

Access to necessary databases and servers.

5. Test Deliverables

Test Cases/Scripts: Comprehensive test cases covering all functionalities.

Test Data: Data sets required for testing (e.g., student profiles, course listings).

Test Reports and Defect Logs: Documentation of test results and any identified defects.

6. Entry and Exit Criteria

Entry Criteria:

Development is complete, and the application is stable.

Test environment is set up and accessible.

All necessary test data is prepared.

Exit Criteria:

All critical and high-severity defects have been resolved or documented with a mitigation plan.

All planned test cases have been executed with documented results.

Stakeholder approval obtained for release.

7. Resources

Team Members:

Test Manager

Test Engineers/Analysts

Developers (for defect resolution)

Tools:

Test Management Tool: JIRA for tracking test cases and defects.

Automation Tool: Selenium for automated testing.

API Testing Tool: Postman for testing backend services.

Hardware/Software Requirements:

Access to a staging server with a database setup similar to production.

Necessary licenses for testing tools.

8. Schedule

Task Start Date End Date Duration

Test Planning YYYY-MM-DD YYYY-MM-DD X days

Test Case Design YYYY-MM-DD YYYY-MM-DD X days

Test Environment Setup YYYY-MM-DD YYYY-MM-DD X days

Test Execution YYYY-MM-DD YYYY-MM-DD X days

Defect Reporting/Resolution YYYY-MM-DD YYYY-MM-DD X days

Final Reporting YYYY-MM-DD YYYY-MM-DD X days

9. Risk Management

Identified Risks:

Delays in development may push back testing timelines.

Changes in requirements could lead to scope creep.

Potential security vulnerabilities in payment processing.

Mitigation Strategies:

Regular communication with development teams to monitor progress.

Early involvement of stakeholders to clarify requirements.

Conduct security assessments before release.

10. Approval

The following individuals will approve the test plan and deliverables:

Project Manager

Product Owner

Key Stakeholders

**Test Case 1: Valid Registration**

**Test Case ID**: TC001  
**Description**: Verify that a student can successfully register with valid inputs.  
**Preconditions**: The system is online, and the student has all required information (name, age, email, course selection, etc.).  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name and last name.
* Enter a valid email address.
* Enter a valid phone number.
* Select a course from the available list.
* Enter valid date of birth and address.
* Click the "Register" button.  
  **Expected Result**: Registration is successful, and a confirmation message is displayed. The student is added to the system.  
  **Postconditions**: The student data is stored in the database.

**Test Case 2: Duplicate Email Registration**

**Test Case ID**: TC002  
**Description**: Verify that the system does not allow duplicate email registration.  
**Preconditions**: The system should already have a registered student with the same email.  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name and last name.
* Enter an email address that is already registered in the system.
* Enter a valid phone number.
* Select a course.
* Click the "Register" button.  
  **Expected Result**: The system should display an error message saying, "Email already exists, please use a different email."  
  **Postconditions**: No new student data is added to the system.

**Test Case 3: Invalid Email Format**

**Test Case ID**: TC003  
**Description**: Verify that the system rejects an invalid email format.  
**Preconditions**: The system is online.  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name and last name.
* Enter an invalid email format (e.g., "student@com").
* Enter a valid phone number.
* Select a course.
* Click the "Register" button.  
  **Expected Result**: The system should display an error message saying, "Invalid email format."  
  **Postconditions**: Registration does not proceed.

**Test Case 4: Empty Required Fields**

**Test Case ID**: TC004  
**Description**: Verify that the system prompts an error when required fields are left empty.  
**Preconditions**: The system is online, and fields such as name, email, phone number, and course selection are required.  
**Test Steps**:

* Open the student registration page.
* Leave the first name and email fields empty.
* Fill in other details (phone number, course, etc.).
* Click the "Register" button.  
  **Expected Result**: The system should highlight the empty required fields and display an error message saying, "This field is required."  
  **Postconditions**: Registration is not completed until the missing information is provided.

**Test Case 5: Invalid Age (Underage)**

**Test Case ID**: TC005  
**Description**: Verify that the system rejects students who are underage for the course.  
**Preconditions**: The system has an age restriction (e.g., students must be at least 18 years old).  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name and last name.
* Enter an email address and valid phone number.
* Enter an age below the required minimum (e.g., 16).
* Select a course.
* Click the "Register" button.  
  **Expected Result**: The system should display an error message saying, "You must be at least 18 years old to register."  
  **Postconditions**: Registration is not completed.

**Test Case 6: Successful Registration with Course Selection**

**Test Case ID**: TC006  
**Description**: Verify that a student can register and select multiple courses if the system allows.  
**Preconditions**: The system allows students to select more than one course.  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name, last name, email, and phone number.
* Select multiple courses from the available list.
* Enter the date of birth and address.
* Click the "Register" button.  
  **Expected Result**: Registration is successful, and the student is enrolled in all selected courses. A confirmation message is displayed.  
  **Postconditions**: The student is registered for all selected courses.

**Test Case 7: Password Security Check (If Password is Required)**

**Test Case ID**: TC007  
**Description**: Verify that the system enforces password security standards.  
**Preconditions**: The system requires a password for registration.  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name, last name, email, and phone number.
* Enter a password that does not meet the security requirements (e.g., "password123").
* Enter a date of birth and address.
* Click the "Register" button.  
  **Expected Result**: The system should display an error message saying, "Password must contain at least one uppercase letter, one number, and be at least 8 characters long."  
  **Postconditions**: Registration does not proceed until a valid password is entered.

**Test Case 8: Edge Case for Maximum Name Length**

**Test Case ID**: TC008  
**Description**: Verify that the system accepts names up to the maximum allowed length.  
**Preconditions**: The system has a name length limit (e.g., 100 characters).  
**Test Steps**:

* Open the student registration page.
* Enter a first name and last name that together are 100 characters long.
* Enter a valid email, phone number, and course selection.
* Click the "Register" button.  
  **Expected Result**: Registration is successful if the name length is within the allowed limit.  
  **Postconditions**: The student is registered.

**Test Case 9: Invalid Phone Number Format**

**Test Case ID**: TC009  
**Description**: Verify that the system rejects invalid phone number formats.  
**Preconditions**: The system expects a phone number in a specific format (e.g., XXX-XXX-XXXX).  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name, last name, email address, and course selection.
* Enter an invalid phone number (e.g., "1234567").
* Click the "Register" button.  
  **Expected Result**: The system should display an error message saying, "Invalid phone number format."  
  **Postconditions**: Registration is not completed.

**Steps**:

* Open the student registration page.
* Enter a valid first name and last name.
* Enter an email address and valid phone number.
* Enter an age below the required minimum (e.g., 16).
* Select a course.
* Click the "Register" button.  
  **Expected Result**: The system should display an error message saying, "You must be at least 18 years old to register."  
  **Postconditions**: Registration is not completed.

**Test Case 6: Successful Registration with Course Selection**

**Test Case ID**: TC006  
**Description**: Verify that a student can register and select multiple courses if the system allows.  
**Preconditions**: The system allows students to select more than one course.  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name, last name, email, and phone number.
* Select multiple courses from the available list.
* Enter the date of birth and address.
* Click the "Register" button.  
  **Expected Result**: Registration is successful, and the student is enrolled in all selected courses. A confirmation message is displayed.  
  **Postconditions**: The student is registered for all selected courses.

**Test Case 7: Password Security Check (If Password is Required)**

**Test Case ID**: TC007  
**Description**: Verify that the system enforces password security standards.  
**Preconditions**: The system requires a password for registration.  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name, last name, email, and phone number.
* Enter a password that does not meet the security requirements (e.g., "password123").
* Enter a date of birth and address.
* Click the "Register" button.  
  **Expected Result**: The system should display an error message saying, "Password must contain at least one uppercase letter, one number, and be at least 8 characters long."  
  **Postconditions**: Registration does not proceed until a valid password is entered.

**Test Case 8: Edge Case for Maximum Name Length**

**Test Case ID**: TC008  
**Description**: Verify that the system accepts names up to the maximum allowed length.  
**Preconditions**: The system has a name length limit (e.g., 100 characters).  
**Test Steps**:

* Open the student registration page.
* Enter a first name and last name that together are 100 characters long.
* Enter a valid email, phone number, and course selection.
* Click the "Register" button.  
  **Expected Result**: Registration is successful if the name length is within the allowed limit.  
  **Postconditions**: The student is registered.

**Test Case 9: Invalid Phone Number Format**

**Test Case ID**: TC009  
**Description**: Verify that the system rejects invalid phone number formats.  
**Preconditions**: The system expects a phone number in a specific format (e.g., XXX-XXX-XXXX).  
**Test Steps**:

* Open the student registration page.
* Enter a valid first name, last name, email address, and course selection.
* Enter an invalid phone number (e.g., "1234567").
* Click the "Register" button.  
  **Expected Result**: The system should display an error message saying, "Invalid phone number format."**Postconditions**: Registration is no  
  t completed.

1.2 Test Control for Student Registration System

1.2.1 Objectives

• Ensure Testing Stays on Track: Monitor the progress of testing activities to ensure they are proceeding as planned according to the test schedule.

• Adapt to Changes in Scope, Schedule, or Resources: Be flexible in adjusting the test plan and resources in response to any changes in project requirements or timelines.

• Provide Visibility to Stakeholders: Keep stakeholders informed about the testing status, progress, and any issues that arise throughout the testing process.

1.2.2 Activities

1. Monitoring

• Track Test Execution Progress: Use a test management tool to monitor the execution of test cases and ensure that testing milestones are being met.

• Monitor Defect Metrics: Regularly analyze defect metrics such as defect density (number of defects per module), severity (high, medium, low), and the status of reported defects.

• Check Adherence to the Test Schedule: Review the test schedule regularly to ensure that testing activities are on track and that any deviations are addressed promptly.

2. Reporting

• Regular Status Reports: Provide daily or weekly status reports to stakeholders detailing the progress of testing activities.

• Summarize Key Metrics: Include key metrics such as test coverage (percentage of requirements tested), test pass/fail rate, and defect trends in the reports.

• Highlight Risks and Issues: Identify and communicate any risks or issues encountered during testing, along with proposed mitigation strategies.

3. Issue Management

• Identify Bottlenecks and Resolve Conflicts: Actively monitor for any bottlenecks in the testing process (e.g., delays in defect resolution) and work with relevant teams to resolve conflicts.

• Reallocate Resources if Needed: If certain areas of testing are falling behind, assess resource allocation and make adjustments as necessary to ensure timely completion.

4. Change Management

• Adapt Test Plans: Make necessary adjustments to the test plan in response to changes in requirements or project scope, ensuring all stakeholders are informed of these changes.

• Ensure Proper Version Control: Maintain version control for all test cases and artifacts to track changes over time and ensure that everyone is working with the most current information.

5. Defect Tracking

• Ensure Timely Resolution of Reported Defects: Work closely with development teams to ensure that defects are addressed promptly and that there is clear communication regarding their status.

• Prioritize Defects Based on Severity and Impact: Establish criteria for prioritizing defects, focusing on high-severity issues that could impact user experience or system functionality.

1.2.3 Tools for Test Control

• Test Management Tool:

• JIRA or TestRail for managing test cases, tracking execution status, and reporting progress.

• Defect Tracking Tool:

• JIRA or Bugzilla for logging defects, monitoring their status, and managing resolutions.

• Dashboarding Tool:

• Power BI or Tableau for visualizing key metrics and providing stakeholders with an overview of testing progress through interactive dashboards.

1.2.4 Deliverables

• Test Plan Document: A comprehensive guide detailing the testing strategy, objectives, scope, resources, schedule, and methodologies specific to the Student Registration System.

• Test Metrics Reports: Regularly generated reports providing quantitative measures of testing progress, including metrics such as test execution status, defect density, and pass/fail rates.

• Final Test Report: A summary document at the end of the testing phase that outlines all testing activities conducted, outcomes achieved, defect resolution status, and recommendations for future releases or improvements.

Certainly! Below is a detailed section on Testing Analysis and Design for the Student Registration System, structured according to your provided outline.

▎2 Testing Analysis and Design

▎2.1 Test Analysis

Test analysis is the process of reviewing and understanding the requirements and design documents to identify test conditions and objectives specific to the Student Registration System.

▎2.1.1 Objectives

• Identify What to Test (Test Conditions): Determine the specific functionalities and features of the Student Registration System that need to be tested.

• Ensure Complete Test Coverage of the Requirements: Verify that all functional and non-functional requirements are covered by the test cases.

• Establish Traceability Between Requirements and Test Cases: Create a mapping between the requirements of the Student Registration System and the corresponding test cases to ensure all requirements are validated.

2.1.2 Activities

1. Requirement Analysis

• Review Business and Technical Requirements: Analyze documents such as the Software Requirements Specification (SRS) and Business Requirements Specification (BRS) for the Student Registration System.

• Identify Ambiguities, Inconsistencies, or Missing Details in Requirements: Document any unclear or conflicting requirements related to student registration processes, such as enrollment procedures, user roles, and system notifications.

• Classify Requirements as Functional, Non-Functional, or Technical: Distinguish between functional requirements (e.g., user registration, course selection), non-functional requirements (e.g., performance, security), and technical requirements (e.g., integration with third-party systems).

2. Test Basis Identification

• Use Artifacts Such As:

• Requirements Specifications (SRS, BRS): Gather information from the SRS detailing user registration workflows and system features.

• Use Cases and User Stories: Identify use cases that describe user interactions with the registration system (e.g., student registration, course enrollment).

• Architecture and Design Documents: Review architectural diagrams that outline how different components of the system interact.

• Risk Assessments: Analyze potential risks associated with critical functionalities like data privacy during registration.

3. Derive Test Conditions

• Identify High-Level Test Scenarios and Conditions: Develop scenarios such as "Student registers for a course," "Admin approves student registration," and "System handles duplicate registrations."

• Map Test Conditions to Requirements to Ensure Coverage: Create a traceability matrix linking each identified test condition to its corresponding requirement.

4. Prioritization

• Rank Test Conditions Based on Risk, Importance, and Complexity: Prioritize test conditions based on factors like the criticality of the feature (e.g., registration process), likelihood of failure, and complexity of implementation.

5. Entry Criteria

• Requirements Are Finalized and Approved: Ensure that all requirements for the Student Registration System have been reviewed and approved by stakeholders.

• Test Environment Setup Has Started (If Required): Confirm that the necessary test environment is being prepared for executing test cases.

2.2 Test Design

Test design is the process of creating detailed test cases, test data, and test scripts based on the test conditions derived during analysis for the Student Registration System.

2.2.1 Objectives

• Create Test Cases to Validate Identified Conditions: Develop specific test cases that will validate each of the identified test conditions.

• Ensure Test Cases Are Detailed, Reproducible, and Traceable: Write test cases that include clear steps, expected results, and links to requirements for traceability.

• Define Test Data Requirements for Execution: Specify what data will be needed to execute the test cases effectively (e.g., sample student data).

2.2.2 Activities

1. Test Case Development

• Create detailed test cases for each identified test condition, including:

• Test Case ID: Unique identifier for each test case.

• Test Description: A brief description of what the test case will validate.

• Preconditions: Any setup required before executing the test case (e.g., user must be logged in).

• Test Steps: Step-by-step instructions on how to execute the test case.

• Expected Results: The anticipated outcome after executing the test steps.

2. Test Data Preparation

• Define what data is required for testing various scenarios, including:

• Valid student information for successful registrations.

• Invalid data scenarios (e.g., missing fields, incorrect formats) to test validation rules.

• Edge cases (e.g., maximum allowable characters in fields).

3. Review and Approval

• Conduct reviews of the test cases with stakeholders (e.g., project managers, developers) to ensure accuracy and completeness.

• Obtain approval on the finalized test cases before proceeding to execution.

4. Traceability Matrix Creation

• Develop a traceability matrix that maps each test case back to its respective requirement to ensure all requirements are covered.

5. Test Script Development (if applicable)

• For automated tests, create scripts based on the detailed test cases using appropriate testing tools (e.g., Selenium for web-based tests).

2.2 Test Design

2.2.2 Activities

1. Test Case Design

• Write Detailed Test Cases Based on Test Scenarios:

• Develop test cases for each identified test scenario related to the Student Registration System.th valid data.

• Preconditions: User must be logged into the system.

• Test Steps:

1. Navigate to the student registration page.

2. Enter valid student details (name, email, password).

3. Click the "Register" button.

• Expected Result: Registration successful message displayed, and user redirected to the dashboard.

• Priority: High

2. Test Data Design

• Identify Data Inputs Required to Validate Test Cases:

• Determine the types of data needed for testing various scenarios (e.g., valid student names, emails, course selections).

• Create Valid, Invalid, Boundary, and Edge-Case Data:

• Valid Data: Correctly formatted student information.

• Invalid Data: Incorrect email formats, missing required fields.

• Boundary Data: Maximum character limits for name and email fields

• Edge Cases: Extremely long names or emails to test system limits.

• Use Tools or Scripts to Generate Large Datasets (if needed):

• Utilize data generation tools or scripts (e.g., Faker library) to create a large volume of test data for performance testing.

3. Test Environment Setup

• Define Hardware, Software, and Network Requirements:

• Specify the necessary hardware (e.g., server specifications), software (e.g., operating system, database), and network configurations required for testing

• Ensure the Environment Replicates the Production Setup (for System Testing):

• Set up a testing environment that closely mirrors the production environment to ensure accurate testing results.

4. Test Automation Design (if applicable)

• Identify Test Cases Suitable for Automation:

• Select repetitive and high-impact test cases for automation, such as user registration and login processes.

• Select Tools (e.g., Selenium, JUnit, TestNG, etc.):

• Choose appropriate testing tools based on the technology stack of the Student Registration System (e.g., Selenium for web-based tests).

• Write Reusable and Maintainable Test Scripts:

• Develop automation scripts that can be reused across multiple test cases and are easy to maintain.

5. Traceability Matrix

• Create a Requirements Traceability Matrix (RTM) to Map Test Cases to Requirements:

• Develop an RTM that links each test case back to its corresponding requirement in the SRS to ensure comprehensive coverage.

6. Entry Criteria

• Test Conditions Are Approved:

• Confirm that all test scenarios and conditions have been reviewed and approved by stakeholders.

• Test Design Templates and Standards Are Ready:

• Ensure that all templates for test cases and documentation standards are finalized and accessible.

2.2.3 Deliverables

1. Test Scenarios

• High-level conditions derived from requirements, such as:

• Student registration process.

• Course enrollment verification.

• Admin approval workflow.

2. Test Cases

• Detailed steps and expected results for each scenario, including:

• Test Case ID

• Description

• Preconditions

• Test Steps

• Expected Result

• Priority

3. Test Data

• Valid and invalid inputs, boundary cases, etc., organized in a format that can be easily referenced during testing.

4. Requirements Traceability Matrix (RTM)

• A document ensuring all requirements are covered by corresponding test cases, facilitating easier validation of requirements.

2.2.4 Tools for Analysis and Design

• Test Management:

• Tools such as Jira or TestRail can be used to manage test cases, track progress, and report on testing activities.

• Automation:

• Selenium for web application testing.

• JUnit or TestNG for unit testing Java applications.

• Traceability Matrix:

• Excel or dedicated tools like RequirementsHub or Helix RM can be utilized to create and maintain the Requirements Traceability Matrix.

This structured approach ensures that all aspects of testing for the Student Registration System are thoroughly planned, documented, and executed effectively.