**Experiment No. 10: Develop Test Cases for Testing Mini Project**

### **Aim:**

To develop test cases for the mini-project, ensuring comprehensive testing of all functionalities and verifying adherence to project requirements.

### **Theory:**

Testing is a critical phase in software development, aimed at identifying and rectifying errors in the system to ensure it performs as intended. Test cases are documented scenarios that validate the functionality of the software by specifying inputs, execution steps, and expected outcomes.

**Types of Testing:**

1. **Unit Testing:** Testing individual components or modules.
2. **Integration Testing:** Verifying the interactions between different modules.
3. **System Testing:** Ensuring the complete system functions correctly.
4. **Acceptance Testing:** Validating the software against user requirements.

**Key Components of a Test Case:**

1. **Test Case ID:** A unique identifier for the test case.
2. **Test Objective:** The purpose of the test case.
3. **Preconditions:** Conditions that must be met before executing the test.
4. **Test Steps:** Detailed steps to execute the test.
5. **Expected Results:** The desired outcome of the test.
6. **Actual Results:** The outcome observed during execution.
7. **Status:** Indicates whether the test passed or failed.

**Common Testing Tools:**

* Selenium (for automated testing)
* JUnit (for unit testing in Java)
* Postman (for API testing)
* TestRail (for test case management)

### **Learning Objectives:**

* To learn how to write test cases systematically.
* To validate the mini-project's functionality and identify potential issues.
* To document testing results effectively.

### **Learning Outcomes:**

At the end of this experiment, students will be able to:

1. Develop comprehensive test cases for a software project.
2. Perform functional and non-functional testing of their mini-project.
3. Analyze testing results to improve software quality.

### **Course Outcomes (COs):**

* **CO6:** Test and validate software systems to ensure quality and functionality.

### **Cognitive Levels of Attainment as per Bloom’s Taxonomy:**

* **L3 (Apply):** Develop test cases for specific project functionalities.
* **L4 (Analyze):** Evaluate testing results to identify issues and improvements.

### **Programme Outcome (POs):**

* **PO3: Design/Development of Solutions:** Develop robust test cases for project validation.
* **PO5: Modern Tool Usage:** Utilize tools and techniques for testing.
* **PO8: Ethics:** Maintain integrity and accuracy in testing practices.
* **PO10: Project Management and Finance:** Efficiently plan and execute the testing process within project timelines.

### **Programme Specific Outcome (PSO):**

* **PSO1:** Apply testing techniques to verify the correctness of software solutions.
* **PSO3:** Test and validate software solutions to meet societal and user requirements.

### **Result & Discussion:**

* **Result:** Test cases were successfully created and executed, ensuring the mini-project met the desired requirements.
* **Discussion:**
  + Test results highlighted areas for improvement.
  + Issues identified during testing were documented for resolution.

### **Conclusion:**

Through this experiment, students learned how to design, document, and execute test cases systematically. They also gained practical experience in identifying and resolving software issues, which is essential for delivering quality software solutions.

### **Steps for the Experiment**

#### ****1. Analyze the Mini-Project Requirements:****

1. Review the software requirements specification (SRS) to understand functional and non-functional requirements.
2. Identify key modules and features to be tested.

#### ****2. Develop Test Cases:****

1. Write test cases for each module using a standard test case template.
2. Ensure test cases cover all functionalities, including edge cases and error conditions.
3. Categorize test cases into types, such as functional, integration, or system tests.

#### ****3. Execute Test Cases:****

1. Follow the test steps and document the actual results.
2. Compare actual results with expected outcomes to determine the status (Pass/Fail).

#### ****4. Document Test Results:****

1. Record the outcomes of all test cases in a test case report.
2. Highlight failed test cases for further analysis and debugging.
   * 1. **Authentication & Authorization**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Preconditions** | **Test Steps** | **Expected Results** | **Actual Results** | **Status** |
| TC001 | Validate login functionality | User credentials exist | Enter valid username and password | Login successful, navigate to dashboard | Same as expected | Pass |
| TC002 | Test invalid login | User credentials exist | Enter invalid username and password | Display error message | Same as expected | Pass |
| TC003 | Test edge case for input fields | N/A | Leave fields blank and submit | Display validation error | Login page does not proceed | Pass |
| TC004 | Check JWT cookie set post-login | Valid credentials | Login and inspect cookie | JWT HTTP-only cookie is set | Same as expected | Pass |
| TC005 | Access control without token | Logged out state | Try to access dashboard directly | Redirect to login | Same as expected | Pass |

* + 1. **Group & Expense Management**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Preconditions** | **Test Steps** | **Expected Results** | **Actual Results** | **Status** |
| TC010 | Create group | User logged in | Click "Create Group", add name & members | Group created and visible | Same as expected | Pass |
| TC011 | Add member to existing group | Group exists | Open group settings, add member | Member added to group | Same as expected | Pass |
| TC012 | Expense creation under group | Group exists | Add new event inside the group | Event visible to group members | Same as expected | Pass |

* + 1. **Expense Splitting**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Preconditions** | **Test Steps** | **Expected Results** | **Actual Results** | **Status** |
| TC013 | Add equal split expense | Group exists | Create expense with equal split option | Amount split evenly among members | Same as expected | Pass |
| TC014 | Add unequal split expense | Group exists | Input custom amounts per person | Split reflects custom amounts | Same as expected | Pass |
| TC015 | Test multiple payers | Group exists | Add expense with 2+ payers | System balances shares accordingly | Same as expected | Pass |

* + 1. **Debt Management**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Preconditions** | **Test Steps** | **Expected Results** | **Actual Results** | **Status** |
| TC016 | Calculate total owed amount | Debts exist | Go to "You Owe" section | Total correctly shown | Same as expected | Pass |
| TC017 | Record manual settlement | Debt exists | Settle up manually | Debt removed/adjusted | Same as expected | Pass |

* + 1. **Data Export**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Preconditions** | **Test Steps** | **Expected Results** | **Actual Results** | **Status** |
| TC026 | Export PDF of group expenses | Group expenses exist | Click "Export PDF" | Download triggers with correct report | Same as expected | Pass |
| TC027 | PDF formatting correctness | N/A | Open exported PDF | Neat, readable layout | Same as expected | Pass |

### **Tools and Technologies:**

* **Testing Tools:** Selenium, JUnit, TestNG, Postman
* **Languages Used for Testing:** Python, Java, or JavaScript