Experiment No.: 08
Title: Preparing Software Test Document (STD)

Batch: B2 Roll No.: 16010421059

Aim

To prepare Software Test Document (STD).

Resources Needed

Internet Explorer, LaTeX Editor

1 Introduction

1.1 System Overview

The QuirkIQ project is an e-learning website aimed at providing educational resources for 11th and 12th-grade science students. It includes features such as interactive lessons, quizzes, virtual labs, and a virtual internship platform. The version to be tested is QuirkIQ v2.0.

1.2 Test Approach

The testing approach involves a combination of manual and automated testing techniques. Major groups of features including content delivery, user authentication, virtual internship platform, multi-language support, and quiz functionality will be thoroughly tested. Testing activities include functional testing, usability testing, performance testing, and security testing. Test tools such as Selenium for automated UI testing and JMeter for performance testing will be utilized. Constraints include tight deadlines for feature releases.

2 Test Plan

2.1 Scope, Approach, Resources, and Schedule

The testing activities will cover all features of the QuirkIQ platform. The approach includes both manual and automated testing techniques. Resources include a team of testers and access to testing tools. The schedule will follow the development timeline of the project.

2.2 Features to be Tested

Table 1: Features to be Tested

Feature	**Test Case(s)**
Content Delivery	TC-001
User Authentication	TC-002
Virtual Internship Platform	TC-003
Multi-language Support	TC-004
Quiz Functionality	TC-005

2.3 Features not to be Tested

All features of the QuirkIQ platform will be tested.

2.4 Testing Tools and Environment

The testing team will consist of 3 testers. Testing tools include Selenium for UI testing and JMeter for performance testing. The test environment requires access to the QuirkIQ website and testing servers.

3 Test Cases

3.1 Test Case 1: Content Delivery (TC-001)

3.1.1 Purpose

To verify the loading and navigation of educational content on the QuirkIQ platform.

3.1.2 Inputs

Click on lesson links.

3.1.3 Expected Outputs & Pass/Fail Criteria

Verify content loads correctly; Ensure navigation works smoothly.

3.1.4 Test Procedure

Navigate through lesson links; Record observations.

3.2 Test Case 2: User Authentication (TC-002)

3.2.1 Purpose

This test case aims to verify the functionality of the user authentication system in the QuirkIQ platform.

3.2.2 Inputs

- Valid username and password combination
- Invalid username and password combination

3.2.3 Expected Outputs & Pass/Fail Criteria

- If a valid username and password combination is provided, the user should be granted access to the platform.
- If an invalid username and password combination is provided, the user should be denied access, and an error message should be displayed.

3.2.4 Test Procedure

1. Enter a valid username and password combination.
2. Verify that access to the platform is granted.
3. Enter an invalid username and password combination.
4. Verify that access to the platform is denied, and an appropriate error message is displayed.

3.3 Test Case 3: Course Enrollment (TC-003)

3.3.1 Purpose

This test case aims to verify the functionality of the course enrollment feature in the QuirkIQ platform.

3.3.2 Inputs

- Valid course code
- Invalid course code

3.3.3 Expected Outputs & Pass/Fail Criteria

- If a valid course code is provided, the user should be successfully enrolled in the course.
- If an invalid course code is provided, the user should not be enrolled in the course, and an error message should be displayed.

3.3.4 Test Procedure

1. Enter a valid course code.
2. Verify that the user is successfully enrolled in the course.
3. Enter an invalid course code.
4. Verify that the user is not enrolled in the course, and an appropriate error message is displayed.

3.4 Test Case 4: Quiz Functionality (TC-004)

3.4.1 Purpose

This test case aims to verify the functionality of the quiz feature in the QuirkIQ platform.

3.4.2 Inputs

- Quiz questions
- User responses

3.4.3 Expected Outputs & Pass/Fail Criteria

- If the user provides correct responses to the quiz questions, they should receive a passing grade.
- If the user provides incorrect responses to the quiz questions, they should receive a failing grade.

3.4.4 Test Procedure

1. Access a quiz within the platform.
2. Answer each quiz question.
3. Submit the quiz.
4. Verify that the user receives a passing or failing grade based on their responses.

3.5 Test Case 5: Payment Processing (TC-005)

3.5.1 Purpose

This test case aims to verify the functionality of the payment processing system in the QuirkIQ platform.

3.5.2 Inputs

- Valid payment information (credit card number, expiration date, CVV)
- Invalid payment information

3.5.3 Expected Outputs & Pass/Fail Criteria

- If valid payment information is provided, the payment should be processed successfully, and the user should gain access to premium features.
- If invalid payment information is provided, the payment should not be processed, and an error message should be displayed.

3.5.4 Test Procedure

1. Enter valid payment information.
2. Verify that the payment is processed successfully, and the user gains access to premium features.
3. Enter invalid payment information.
4. Verify that the payment is not processed, and an appropriate error message is displayed.

4 Test Logs

During the execution of tests, it's essential to maintain detailed logs of what occurred. Test logs serve as a record of the testing process, documenting the steps taken, any issues encountered, and the outcomes of each test case. These logs provide valuable insights into the effectiveness of the testing process, helping to identify patterns, trends, and areas for improvement. Test logs should include:

- Test Case ID: Each test case should be uniquely identified to track its progress and outcomes.
- Description of Test Steps: Detailed descriptions of the steps taken during test execution, including inputs provided and actions performed.
- Observations and Results: Records of any observations made during testing, such as unexpected behavior or errors encountered.
- Date and Time: Timestamps indicating when each test was executed.
- Tester Information: Information about the tester responsible for executing the test, including their name or identifier.
- Incident Reporting: If any issues or anomalies occur during testing, they should be documented in an incident report within the test log, providing details of the incident, its impact, and any actions taken to address it.

5 Test Results

Test results provide a summary of the outcomes of each test case executed during testing. They document whether the test case passed or failed, along with any relevant observations or findings. Test results serve as a critical component of the testing process, providing stakeholders with insights into the quality and readiness of the software being tested. Key elements of test results include:

- Test Case ID: Each test case should be referenced to link the results back to the specific test scenario.
- Execution Date and Time: Timestamps indicating when each test case was executed.
- Outcome: Indicates whether the test case passed, failed, or produced inconclusive results.
- Observations: Any additional observations or findings made during the execution of the test case, such as unexpected behavior or errors encountered.
- Error Messages: If a test case fails, any error messages generated should be documented to aid in troubleshooting and resolution.
- Location of Output: If the test case produces output, such as on-screen messages or reports, the location of this output should be recorded for reference.

Table 2: Features to be Tested

Feature	Test Case(s)
Content Delivery	TC-001
User Authentication	TC-002
Virtual Internship Platform	TC-003
Multi-language Support	TC-004
Quiz Functionality	TC-005

5.1 Features not to be Tested

All features of the QuirkIQ platform will be tested.

5.2 Testing Tools and Environment

The testing team will consist of 3 testers. Testing tools include Selenium for UI testing and JMeter for performance testing. The test environment requires access to the QuirkIQ website and testing servers.

6 Test Cases

Table 3: Test Case Table

Test Case ID	Feature	Description	Inputs	Expected Outputs & Pass/Fail Criteria	Test Procedure
TC-001	Content Delivery	Verify content loading and navigation	Click on lesson links	Verify content loads correctly; Ensure navigation works smoothly	Navigate through lesson links; Record observations
TC-002	User Authentication	Test user login and registration functionality	Enter valid/invalid credentials	Verify successful login/register; Ensure error messages display correctly	Attempt login/register with different credentials; Record results
TC-003	Virtual Internship Platform	Test virtual internship platform functionality	Access internship tasks	Complete assigned tasks; Verify task completion status	Navigate through internship tasks; Record completion status
TC-004	Multi-language Support	Test language selection and content translation	Select different language options	Verify content displays in selected language; Ensure translations are accurate	Change language settings; Verify content language
TC-005	Quiz Functionality	Test quiz creation and submission	Create and submit quiz	Verify quiz creation; Ensure accurate submission results	Create and submit quiz; Record submission outcome

7 Questions

7.1 Differentiate between verification and validation.

Verification: It is the process of evaluating a system or component to determine whether it meets specified requirements. In verification, the focus is on ensuring that the software is built correctly according to its design specifications. It involves activities such as reviews, inspections, and walkthroughs to check for consistency, completeness, and correctness at various stages of the development process. Verification answers the question, ”Are we building the product right?”

Validation: It is the process of evaluating a system or component during or at the end of the development process to determine whether it meets customer requirements and expectations. In validation, the focus is on ensuring that the right product is being built to solve the customer’s problem or fulfill their needs. It involves activities such as testing and user feedback to assess whether the software meets its intended purpose and delivers the expected value. Validation answers the question, ”Are we building the right product?”

In summary, verification checks whether the software is built correctly according to specifications, while validation checks whether the software meets the customer's needs and expectations.

7.2 List down all OO software testing strategies.

Object-Oriented Software Testing Strategies:

1. Class-Based Testing: This strategy focuses on testing individual classes or objects within the software. Test cases are designed to cover various scenarios for each class, including boundary conditions, error handling, and interactions with other classes.
2. Inheritance-Based Testing: In object-oriented systems, inheritance allows classes to inherit properties and behavior from parent classes. Inheritance-based testing aims to test the behavior of subclasses inherited from their parent classes, ensuring that they maintain the intended functionality while adding new features.
3. Polymorphism-Based Testing: Polymorphism allows objects to be treated as instances of their parent classes, enabling dynamic method binding and runtime flexibility. Polymorphism-based testing verifies that objects can behave correctly in different contexts and that method calls are dispatched appropriately based on object types.
4. State-Based Testing: Object-oriented systems often have complex states that objects can transition between during their lifecycle. State-based testing focuses on testing the behavior of objects as they transition between different states, ensuring that state transitions occur correctly and that objects maintain consistency in their behavior.
5. Interaction-Based Testing: This strategy involves testing the interactions between objects or components within the software. It aims to verify that objects collaborate and communicate correctly with each other to achieve the desired functionality. Interaction-based testing often involves techniques such as mock objects and test doubles to isolate and control dependencies during testing.
6. Use-Case-Based Testing: Use cases represent typical interactions between users and the system, describing how users interact with the software to accomplish specific tasks. Use-case-based testing focuses on testing the software's functionality from the user's perspective, ensuring that it meets user requirements and delivers value in real-world scenarios.

8 Outcomes

- CO4: Demonstrate test case design.

9 Conclusion

In this experiment, I have highlighted the crucial distinctions between verification and validation, emphasizing their roles in ensuring software quality. Additionally, I've outlined various object-oriented software testing strategies, demonstrating the importance of tailored approaches to address the complexities of object-oriented systems and enhance overall testing effectiveness.

10 References

1. Roger S. Pressman, *Software Engineering: A practitioners Approach*, 7th Edition, McGraw Hill, 2010.
2. Ian Somerville, *Software Engineering*, 9th edition, Addison Wesley, 2011.
3. `http://vlabs`