REPORT ON INDUSTRIAL TRAINING
CSE-350

Company Name: Bangladesh Computer Council (BCC)

Submitted By:
1. Anika Siddiqui Mayesha – 201814013
2. Sabrina Afreen Haque – 201814020
3. Anika Tahsin -201814038

Table of contents:

1. Introduction:

Bangladesh Computer Council (BCC) is a statutory and autonomous body under the ICT Division, Ministry of Science and Information and Communication Technology, Government of Bangladesh for encouraging and providing support for ICT related activities in Bangladesh. We went for our industrial attachment in BCC at SQTC center with a motive to gain an idea of corporate office life. Throughout these 3 weeks of training, we were provided the opportunity to work under the current software testing team of SQTC center. We gained hands on experience about Software Testing Life Cycle, Testing Types, Testing Levels and executing manual testing on a system. We were given 2 testing projects to implement our testing knowledge and further learn about the ins and outs of actual software testing.

2. Respective supervisor in Company:

Supervisor	Designation
Mohammad Saiful Alam Khan	Test Manager, Project Director
Progga Labony	Assistant Programmer and Tester
Saad Ahmed Sadi	Assistant Programmer and Tester

3. Topics Overview:

Throughout the 3 weeks of industrial training we were taught about how software is tested professionally. The SQTC department worked as an independent testing unit and worked with the developer team to make a better product for their clients. Though it was not possible to experience a lot of different varieties of testing in just 3 weeks, we did learn some basic skills such as:

i. Software Testing Life Cycle, Testing Types, Testing Levels and Testing Process according to TMMi certification by SQTC department.
ii. We were provided with an SRS document of our projects to analyze the user requirement and develop a bi-directional traceability with our test cases.

iii. After Requirement Analysis, we started test planning, test implementation and test execution.
iv. The bulk of our work is concentrated in functional testing, manual testing & Black Box testing
v. Our testing projects included: 1. Temple Based Child & Mass Literacy Program Software for Hindu Religious Welfare Trust and 2. Emergency Operation Dashboard For Ministry of Disaster Management and Relief Overall.
vi. We gained Performance Testing experience using tools like HP LoadRunner and Apache Jmeter.
vii. We went through a final evaluation which consisted of an MCQ test on syllabus of ISTQB – Certified Tester Foundation Level (CTFL) .

4. Topics Analysis:

Software testing is a way to assess the quality of the software and to reduce the risk of software failure in operation.

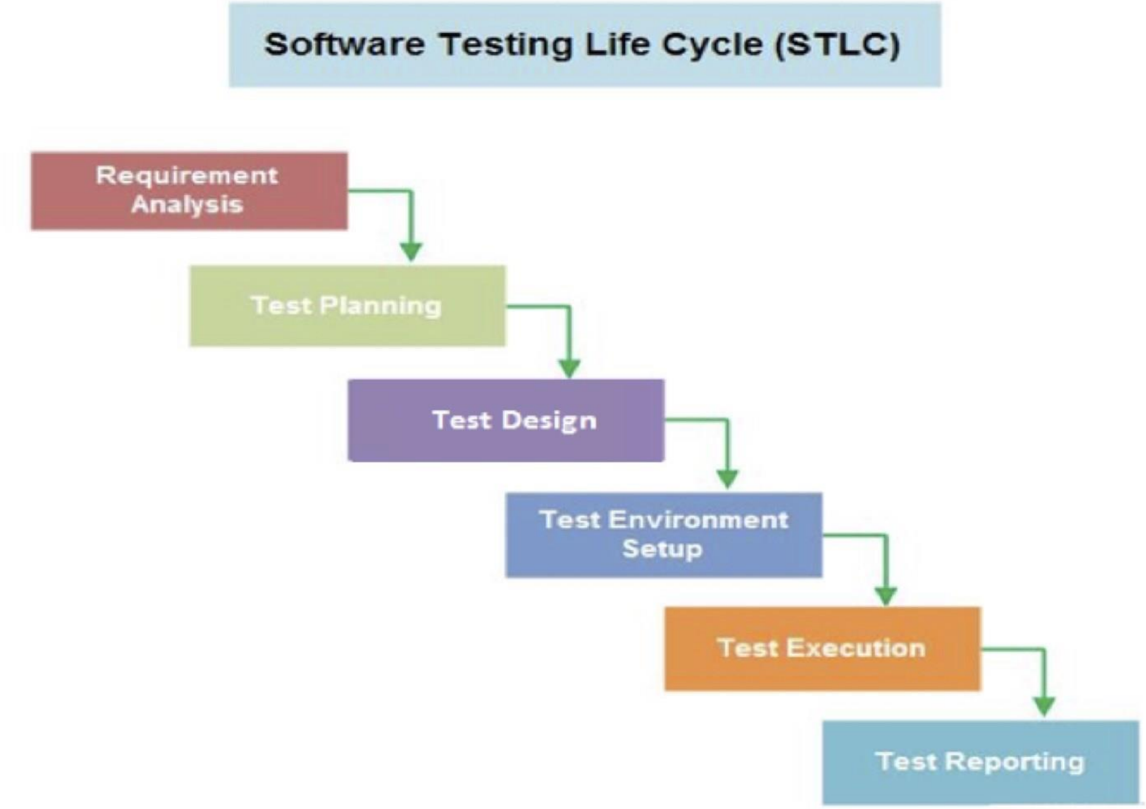
Objectives of testing:

i. To prevent defects by evaluate work products such as requirements, user stories, design and code.
ii. To verify whether all specified requirements have been fulfilled.
iii. To build confidence in the level of quality of the test object.
iv. To find defects and failures thus reduce the level of risk of inadequate software quality**.**

Necessity of Testing:

Rigorous testing of components and systems, and their associated documentation, can help reduce the risk of failures occurring during operation. When defects are detected and subsequently fixed, this contributes to the quality of the components or systems. In addition, software testing may also be required to meet contractual or legal requirements or industry-specific standards**.**

Software Development Life Cycle (SDLC):



Testing Process followed in SQTC Centre:

Planning and Control:

i. SRS Review
ii. Knowledge Transfer

Analysis and Design:

i. Test Case Design
ii. Develop Traceability Matrix

Implementation and Execution:

i. Test Case Execution

ii. Defect Management

Evaluating exit criteria and reporting:

i. Test Summary Report

Test Closure activities.

Testing Levels:

Test levels are groups of test activities that are organized and managed together. Each test level is an instance of the test process.

i. Component testing
ii. Integration testing
iii. System testing
iv. Acceptance testing

Component Testing:

Component testing (also known as unit or module testing) focuses on components that are separately testable. This testing is usually performed by the developer who wrote the code, but it at least requires access to the code being tested. Developers may alternate component development with finding and fixing defects.

Integration testing:

Integration testing focuses on interactions between components or systems. This testing is performed after component testing, and is generally automated. In iterative and incremental development, component integration tests are usually part of the continuous integration process.

Component integration tests and system integration tests should concentrate on the integration itself. For example, if integrating module A with module B, tests should focus on the communication between the modules, not the functionality of the individual modules, as that should have been covered during component testing.

System Testing:

System testing focuses on the behavior and capabilities of a whole system or product, often considering the end-to-end tasks the system can perform and the non-functional behaviors it exhibits while performing those tasks**.**

System testing often produces information that is used by stakeholders to make release decisions. System testing may also satisfy legal or regulatory requirements or standards. System testing is typically carried out by independent testers who rely heavily on specifications.

Acceptance Testing:

Acceptance testing, like system testing, typically focuses on the behavior and capabilities of a whole system or product. Objectives of acceptance testing include:

i. Establishing confidence in the quality of the system as a whole
ii. Validating that the system is complete and will work as expected
iii. Verifying that functional and non-functional behaviors of the system are as specified

Common forms of acceptance testing include the following:

i. **User Acceptance Testing (UAT):** User acceptance testing of the system is typically focused on validating the fitness for use of the system by intended users in a real or simulated operational environment.

ii. **Operational Acceptance Testing (OAT):** The acceptance testing of the system by operations or systems administration staff is usually performed in a (simulated) production environment.

iii. **Contractual and Regulatory Acceptance Testing:** Contractual acceptance testing is performed against a contract's acceptance criteria for producing custom-developed software.
Regulatory acceptance testing is performed against any regulations that must be adhered to, such as government, legal, or safety regulations.

iv. **Alpha and Beta Testing:** Alpha testing is performed at the developing organization's site, not by the development team, but by potential or existing customers, and/or operators or an independent test team.

Beta testing is performed by potential or existing customers, and/or operators at their own locations. Beta testing may come after alpha testing, or may occur without any preceding alpha testing having occurred.

Testing Types:

Functional Testing:

Functional Testing is defined as a type of testing which verifies that each function of the software application operates in conformance with the requirement specification.

This testing involves checking of User Interface, APIs, Database, client/ server applications and functionality of the Application Under Test.

Types:

The following types of functional testing are done in SQTC Centre:

i. Manual Testing
ii. Automation Testing

Standards followed for Functional Testing:

i. Test Maturity Model Integration (TMMi)
ii. ISO

iii. ISTQB

Tools Used for Functional Testing:

i. Selenium - Popular Open Source Functional Testing Tool

ii. QTP/UFT - Very user-friendly Functional Test tool by HP/Microfocus

iii. JUnit- Used mainly for Java applications and this can be used in Unit and System Testing

iv. soapUI - This is an open source functional testing tool, mainly used for Web service testing. It supports multiple protocols such as HTTP, SOAP, and JDBC.

v. Watir - This is a functional testing tool for web applications. It supports tests executed at web browser and uses Ruby scripting language.

Non-Functional Testing:

Non-functional testing is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters, which are never addressed, by functional testing.

An excellent example of non-functional test would be to check how many people could simultaneously login into software.

Non-functional Testing Parameters:

Security –> Security Testing

Reliability-> Reliability Testing

Survivability -> Recovery Testing

Availability -> Stability Testing

Usability -> Usability Testing

Scalability-> Scalability Testing

Interoperability-> Interoperability Testing

Efficiency

Flexibility

Portability

Reusability

Manual Testing:

Manual Testing is a type of Software Testing where Testers manually execute test cases without using any automation tools.

Manual testing is the most primitive of all testing types and helps find bugs in the software system.

Any new application must be manually tested before its testing can be automated. Manual testing requires more effort, but is necessary to check automation feasibility.

Goals of Manual Testing:

i. The key concept of Manual Testing is to ensure that the application is error free and it is working in conformance to the specified functional requirements.
ii. Test Suites or cases are designed during the testing phase and should have 100% test coverage.

Automation Testing:

Test automation is the use of independent software (automation tool) to assist the execution of tests on Application under test. While assisting in tests, the automation tools performed the desired test steps and then compares the actual outcomes to predicted outcomes.

Below are the common criteria used to identify the automation tool for a project:

i. Project Requirements
ii. Budget
iii. Skilled Resources
iv. Support Types For Application
v. Tool Maintenance
vi. Ease Of Use.

Black Box Testing:

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the tester does not know the internal structure /design/ implementation of the item being tested.

Levels Applicable To:

i. Integration Testing
ii. System Testing
iii. Acceptance Testing
iv. The higher the level, and hence the bigger and more complex the box, the more black-box testing method comes into use.

Example:

A tester, without knowledge of the internal structures of a website, tests the web pages by using a browser; providing inputs (clicks, keystrokes) and verifying the outputs against the expected outcome.

White Box Testing:

White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or structural testing) is a software testing method in which the tester knows the internal structure /design/ implementation of the item being tested.

Programming and implementation knowledge is essential. White box testing beyond the user interface and into the nitty-gritty of a system.

Levels Applicable To:

i. Unit Testing: For testing paths within a unit.
ii. Integration Testing: For testing paths between units.
iii. System Testing: For testing paths between subsystem.

However, it is mainly applicable to Unit testing.

Example:

A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) and illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

Gray Box Testing

Gray Box Testing is a technique to test the software product or application with partial knowledge of the internal structure of an application.

Gray Box Testing is a software testing method, which a combination of both Black Box and White Box testing.

In Software Engineering, Gray Box Testing gives the ability to test both sides of an application , presentation layer as well as the code part.

Level applicable to:

i. Integration Testing

Regression Testing:

When any modification or changes are done to the application or even when any small change is done to the code then it can bring unexpected issues. Along with the new changes it becomes very important to test whether the existing functionality is intact or not.

Regression testing is re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change.

Performance Testing:

Performed to determine how a system performs in terms of responsiveness and stability under particular workload.

The following types of performance testing are done in SQTC Centre:

i. Load Testing: Load Testing is performed to test the performance of the system or software application under extreme load.

ii. Stress Testing: Stress Testing is performed to test the robustness of the system or software application under extreme load.

Standards followed for Performance Testing

i. TMMi
ii. ISO

Tools Used for Performance Testing

i. HP LoadRunner
ii. Apache Jmeter
iii. Perfmon

Test Case Designing:

Test Case: Test cases are nothing but steps to validate the functionality of a feature. It is a set of test inputs, execution conditions, and expected results developed for a particular objective.

<u>A test case must be:</u>

i. Clear of purpose
ii. Well Organized
iii. Reviewable
iv. Maintainable

v. Complete
vi. Re-usable
vii. Useful to other testers

Some Terminologies:

Error:

i. Human mistake that causes fault
ii. Refers to difference between Actual Output and Expected output
iii. For example, a developer may misunderstand a requirement, or type a variable name incorrectly – leads to an Error. It is the one which is generated because of wrong login, loop or due to syntax**.**

Fault:

i. Discrepancy in code that causes a failure
ii. Fault is when any functionality is not working correctly
iii. A fault is introduced into the software as the result of an error

Defect:

i. A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g., an incorrect statement or data definition. A defect, if encountered during execution, may cause a failure of the component or system

Bug:

i. Defect accepted by development team is called Bug.

Failure:

i. Failure is an inability of the system or components to perform its required function
ii. Failure occurs when fault is executed.

Severity:

i. It is the extent to which the defect can affect the software. In other words it defines the impact that a given defect has on the system.

For example:

If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare.

Priority:

i. Priority defines the order in which we should resolve a defect.

For example: If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

5. Projects/Tasks Assigned:

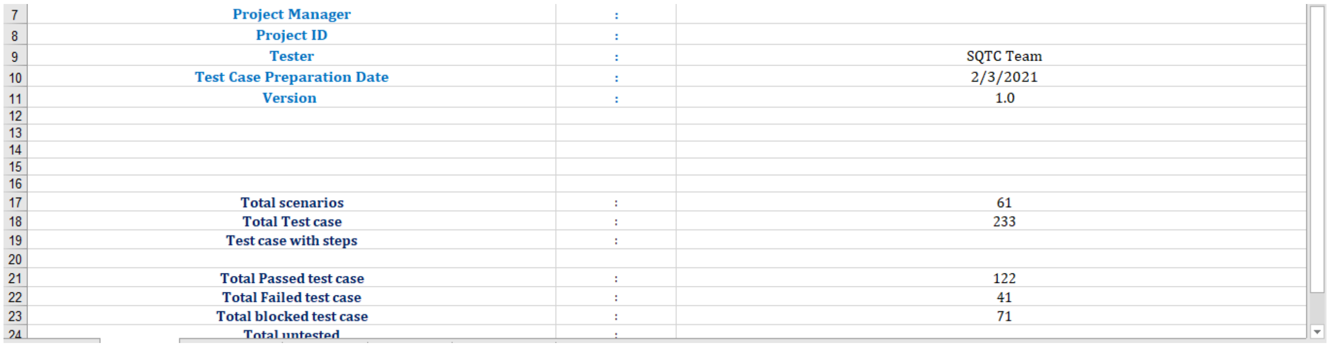
We were assigned two projects: 1. Temple Based Child & Mass Literacy Program Software for Hindu Religious Welfare Trust and 2. Emergency Operation Dashboard For Ministry of Disaster Management and Relief Overall.

We were given the SRS document to analysis the user requirements of the project. After getting the Software program, we started planning tests, identifying test scenarios , writing test cases and executing test cases.

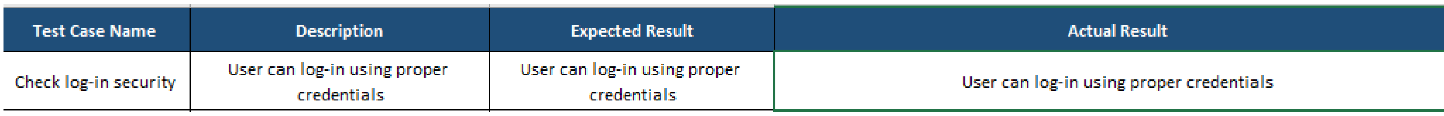
Temple Based Child & Mass Literacy Program Software was assigned at first for testing but this software crashes during test execution. As developer was not able to fix the bug, we started working on Emergency Operation Dashboard.

For Temple Based Child & Mass Literacy Program:

Test Scenario	Test Cases	Number of Tests Executed	Number of Passed Test Cases	Number of Failed Test Cases	Number of skipped/untested Test Cases
61	233	163	122	41	70 (Software Crashed)



7	Project Manager	:	
8	Project ID	:	
9	Tester	:	SQTC Team
10	Test Case Preparation Date	:	2/3/2021
11	Version	:	1.0
12			
13			
14			
15			
16			
17	Total scenarios	:	61
18	Total Test case	:	233
19	Test case with steps	:	
20			
21	Total Passed test case	:	122
22	Total Failed test case	:	41
23	Total blocked test case	:	71
24	Total untested	:	

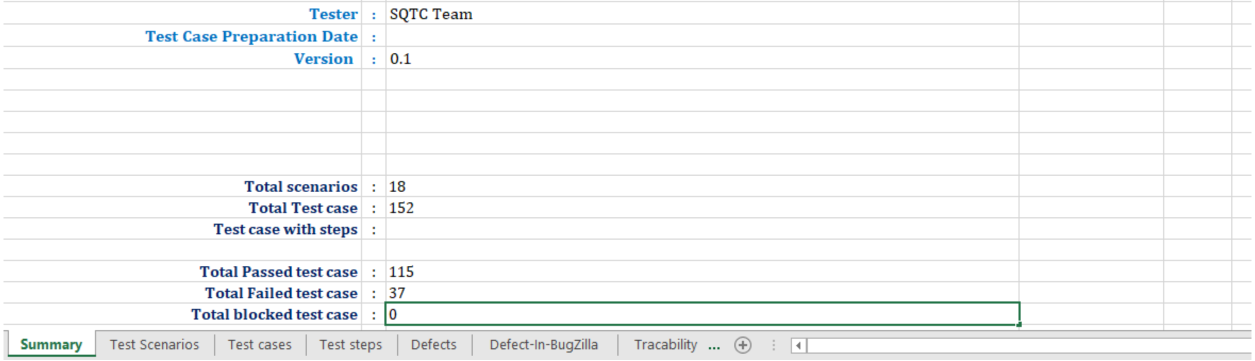


Test Case Name	Description	Expected Result	Actual Result
Check log-in security	User can log-in using proper credentials	User can log-in using proper credentials	User can log-in using proper credentials

Because of Confidentiality issue, We are not allowed to share screenshots. It is just a demo of log in page test.

For Emergency Operation Dashboard:

Test Scenario	Test Cases	Number of Tests Executed	Number of Passed Test Cases	Number of Failed Test Cases	Number of skipped/untested Test Cases
18	152	152	115	37	0



Tester	:	SQTC Team
Test Case Preparation Date	:	
Version	:	0.1
Total scenarios	:	18
Total Test case	:	152
Test case with steps	:	
Total Passed test case	:	115
Total Failed test case	:	37
Total blocked test case	:	0

6. Learning/takeaways from the company's training:

Most of our MIST courses included project development so it was an entirely new experience learning how to test a system professionally. Certain skills and conventions of testing are transferable to development or software/hardware work in general. For example, the differences between the methodologies and mindset of a tester versus a developer. The critical way of looking at our system modules and workflow helped us understand how to better develop and build up a stable system. Confirmation bias is a developers biggest weakness and working as a tester helps a developer become more aware of it and helps them better regulate their judgement with critical pessimism.

In SQTC center, we mostly implemented Functional Testing, Black Box Testing and Manual Testing on two projects. We further learned about Performance Testing using HP LoadRunner and Apache Jmeter tools. The department works under TMMi level-3 certification, which helps them, maintain an international standard for their testing processes.

Working offline and in an Office setting provided us with a very worthwhile experience of a workplace environment. We saw first-hand exactly how important effective communication and co-operation amongst colleagues is when working together. While working on our first project we encountered a defect while caused the software to crash. We personally witnessed how the testing team communicated and prioritized decisions regarding this project with the development team. Unfortunately a consequence of this incident ended up being that we couldn't personally work on Bug Report, Summary Report, Re-testing and Closure Reports. Our supervisor however gave us an in-depth look into her own projects and how the rest of the testing process concludes.

7.Conclusion:

Our industrial training provided us with a glimpse into our current industry trends and corporate workplace environment. Due to our training being offline we got a hands on experience in functionality, acceptance and performance testing on real practical projects.

Overall it was a worthwhile experience learning and growing during these 3 weeks and we remain thankful to MIST for the wonderful opportunity.