IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

| Name: R T Athukorala |
| --- |
| Student Reference Number: 10820936 |

| Module Code: PUSL 2020 | Module Name: Software Development tools and Practices |
| --- | --- |
| Coursework Title: Group Assigment ( Group 43 ) | |
| Deadline Date: 23 May 2023 | Member of staff responsible for coursework:<br>Dr. Rasika Ranaweera \| Ms. Pavithra Subhashini |
| Programme: BSc (Hons) Computer Science | |

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

1. 10820936 – R T Athukorala
2. 10820935 – D G L P Geethanjana
3. 10820945 – R M K M Rathnayake
4. 10819554 – K D N I Wijenayake
5. 10820282 – C S Weerasuriya
6. 10820941 – N A R Anandage

***We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.***

Signed on behalf of the group: 

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed :

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software……

**Overall mark _____%  Assessors Initials _____  Date_____**

*Please delete as appropriate Sci/ps/d:/students/cwkfrontcover/2013/14

# Table of contents

## Contents

# Introduction

The Colombo Municipal Council (CMC) has embarked on a significant project to optimize garbage collection by launching a web application that allows Green Task Force (GTF) volunteers, Green Captains, and Web Masters to interact and collaborate seamlessly. Developed using PHP and MySQL, the application aims to facilitate the efficient reporting of thrown-away garbage, streamline the process of validating and approving incidents, and enable garbage collection staff to easily locate and prioritize garbage collection spots based on their importance.

This report aims to present a comprehensive overview of the testing strategies employed throughout the development process to ensure the application meets its functional requirements and delivers a high-quality user experience. The report will detail the test cases designed for each user type, such as GTF members, Green Captains, and Web Masters, to validate their respective functionalities and user interactions. Additionally, the report will describe the structure and role of mock objects used during testing, which simulate complex components of the application, enabling more focused and efficient testing of individual modules.

The report will also explain the unit and integration tests performed, highlighting the importance of testing individual components in isolation and then testing their interaction within the entire system. This approach ensures that each component functions correctly and that the overall application works as expected. Furthermore, the report will outline the functional test plans, which encompass various user scenarios and workflows, to verify that the application meets the needs of its intended users and performs as intended in real-world situations.

A critical analysis of the test strategy will be presented, explaining the rationale behind the chosen testing methods and their effectiveness in identifying and addressing potential issues in the application. This analysis will provide insights into the strengths and weaknesses of the testing process, guiding future testing efforts and improvements. Finally, the tools and techniques utilized during the testing process will be discussed, including the use of automated testing frameworks, code coverage tools, and performance monitoring tools, to ensure a thorough and efficient testing process.

## 1. Test Cases Design

| Test Case ID | User Type | Functionality Tested | Test Case Description | Test Steps | Expected Result | Actual Result | Pass/Fail |
|---|---|---|---|---|---|---|---|
| TC1 | GTF Member | Registration and Sign-up | Register a new GTF Member with valid details | Enter valid details | Successful registration | Successful registration | Pass |
| TC2 | GTF Member | Registration and Sign-up | Attempt to register a new GTF Member with invalid details (e.g. empty fields, incorrect email format) | Enter invalid details | Registration fails with error messages | Registration fails with error messages | Pass |
| TC3 | GTF Member | Login | Test login with invalid credentials (GTF Member) | Enter invalid details | Error message is displayed. | Error message is displayed. | Pass |
| TC4 | GTF Member | Report Garbage Incident | Submit a new garbage incident with valid data | Go to “add a incident” page and enter valid details | Incident reported successfully | Incident reported successfully | Pass |
| TC5 | GTF Member | Report Garbage Incident | Submit a new garbage incident with invalid data | Go to “add a incident” page and enter invalid details | Incident reporting fails with an error message | Incident reporting fails with an error message | Pass |
| TC6 | Green Captain | View Incident List and Map | View the list and map of reported incidents | Go to Dashboard and view the list | Successfully view the list and map | Successfully view the list and map | Pass |
| TC7 | Green Captain | Approve/Reject Incident | Approve a reported garbage incident | Go to | Incident status updated to approved | Incident status updated to approved | Pass |
| TC8 | Green Captain | Approve/Reject Incident | Reject a reported garbage incident | Go to “manage_incidents” and update the status | Incident status updated to rejected | Incident status updated to rejected | Pass |
| TC9 | Green Captain | Set Importance Flag | Set an importance flag for an approved incident | Go to “manage_incidents” and update the status | Importance flag set successfully | Importance flag set successfully | Pass |
| TC10 | Admin | Create Accounts | Create a new account for a | Go to Admin dashboard and create new account | Account created successfully | Account created successfully | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | garbage collection staff | | | | |
| TC11 | Admin | Create Accounts | Create a new account for a garbage collection staff | Go to admin dashboard | Account created successfully | Account created successfully | Pass |
| TC11 | Admin | Post Article | Post a new article related to garbage collection | Go to admin dashboard | Article posted successfully | Article posted successfully | Pass |
| TC12 | Admin | View article | Test view news articles (General Public) | Go to home page and view articles | News articles are displayed. | News articles are displayed. | Pass |

## 2. Unit and Integration Tests

Unit and integration tests are essential components of the software testing process. Unit tests focus on individual components or functions, while integration tests evaluate how different parts of the system work together. In this report, we will discuss both unit and integration tests implemented for the CMC garbage collection web application.

Unit Tests

Unit tests are designed to verify the correctness of individual functions, classes, or components within the application. For the CMC garbage collection web application, unit tests can be written for the following functionalities:

- User authentication (GTF Members, Green Captain, and Web Master)
- GTF Member registration
- Incident reporting, updating, and deletion
- Incident approval and rejection by Green Captain
- Setting flags for incident importance
- Account creation for Green Captain and collection staff by Web Master
- Article posting by Web Master

Unit testing frameworks like PHPUnit (for PHP) or similar tools can be used to write and execute these tests. Mock objects can be utilized to isolate the component being tested from other parts of the application, ensuring that the tests are focused solely on the component's functionality.

Integration Tests

Integration tests are designed to verify that different components of the application work together correctly. For the CMC garbage collection web application, integration tests can be written for the following scenarios:

- GTF Member reporting an incident and Green Captain approving or rejecting it
- Green Captain setting flags for incident importance and the changes being reflected in the system
- Admin creating accounts for Green Captain and collection staff, and their successful login
- Admin posting articles and their appearance on the public-facing side of the application

Integration tests can be written using testing frameworks and tools like Codeception (for PHP) or similar tools. These tests should be executed after the successful completion of unit tests to ensure that the individual components are working correctly before testing their integration.

To run both unit and integration tests, you can set up a testing environment, configure the testing framework, and execute the tests using a command-line interface or an integrated development environment (IDE). It is essential to perform these tests regularly during the development process to identify and fix any issues that may arise.

## 3. Functional Test Plans

Functional testing is a crucial part of the software testing process that ensures that the application's features and functionalities meet the specified requirements. A functional test plan outlines the testing activities, test cases, and testing strategies to verify that the application works as intended. For the CMC garbage collection web application, the functional test plan can be divided into the following sections:

Test Objectives

The primary objective of the functional test plan is to ensure that the application satisfies the functional requirements specified in the project scenario. These include:

- User authentication and registration for GTF Members, Green Captain, and Web Master
- Incident reporting, updating, and deletion by GTF Members
- Incident approval, rejection, and flag setting by Green Captain
- Account creation for Green Captain and collection staff by Admin
- Article posting by Admin.

Test Approach

To achieve the test objectives, the functional testing approach will consist of the following activities:

- Test case design: Create a comprehensive list of test cases that cover all the functional requirements of the application.
- Test environment setup: Configure the testing environment, including software, hardware, and network configurations.
- Test execution: Execute the designed test cases using manual or automated testing tools.
- Test result analysis: Analyze the test results and identify any defects or areas that need improvement.

Test Schedule

The test schedule outlines the timeline for the execution of the functional test plan. This includes the start and end dates for each testing activity, such as test case design, test environment setup, test execution, and test result analysis. The test schedule should be aligned with the overall project timeline to ensure that functional testing is completed on time.

Test Deliverables

The test deliverables include all the artifacts produced during the functional testing process. These may include:

- Test plan document
- Test case specifications
- Test scripts (for automated testing)
- Test data and results
- Defect reports and logs

## 4. Test and validation metrics

Test and validation metrics are essential for evaluating the effectiveness of the testing process and measuring the quality of the software product. Here are some important test and validation metrics that can be used to evaluate the CMC garbage collection web application:

Code Coverage

Code coverage is a metric that measures the percentage of the application's source code that is executed during testing. It helps determine how thoroughly the code has been tested and identify any untested areas of the code. For the CMC garbage collection web application, aim for a high code coverage percentage (ideally 80% or more) to ensure the majority of the codebase is tested.

Test Case Coverage

Test case coverage measures the percentage of functional requirements and user scenarios covered by the test cases. This metric helps ensure that all the critical functionalities and user scenarios are tested. For the CMC garbage collection web application, strive for a high test case coverage percentage (ideally 90% or more) to guarantee that all important functionalities and user scenarios are tested.

Defect Density

Defect density is the number of defects identified during testing divided by the size of the software (typically measured in lines of code or function points). This metric helps assess the overall quality of the software and identify any areas with high defect rates. For the CMC garbage collection web application, maintain a low defect density to ensure a high-quality software product.

Defect Removal Efficiency

Defect removal efficiency (DRE) is the percentage of defects identified and fixed during the testing process compared to the total number of defects found (including those found after release). A high DRE indicates that the testing process is effective in identifying and fixing defects before the software is released. For the CMC garbage collection web application, aim for a high DRE (ideally 90% or more) to ensure the majority of defects are identified and fixed during testing.

Test Execution Rate

Test execution rate is the number of test cases executed divided by the total number of test cases planned. This metric helps assess the efficiency of the testing process and ensure that all planned test cases are executed. For the CMC garbage collection web application, maintain a high test execution rate (ideally 100%) to guarantee that all planned test cases are executed and any potential issues are identified.

| Name | Index number | Workload Metrics |
|---|---|---|
| DGLP Geethanjana | 10820935 | Introduction, Test Objectives, Test Approach |
| RT Athukorala | 10820936 | Test Cases Design, Unit Tests, Test Case Coverage |
| RMKM Rathnayake | 10820945 | Test Cases Design, Test Schedule, Defect Density |
| KDNI Wijenayake | 10819554 | Test Cases Design, Test Deliverables, Defect Removal Efficiency |
| CS Weerasuriya | 10820282 | Test Cases Design, Integration Tests, Introduction |
| NAR Anandage | 10820941 | Test Cases Design, Code Coverage, Test Execution Rate |

# 5. Reference

## References


Anon., n.d. *How To Write Test Cases: The Ultimate Guide With Examples.* [Online]
Available at: https://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/

Brownlee, J., n.d. *What is the Difference Between Test and Validation Datasets?.* [Online]
Available at: https://machinelearningmastery.com/difference-test-validation-datasets/

Hamilton, T., n.d. *Software Testing Techniques with Test Case Design Examples.* [Online]
Available at: https://www.guru99.com/software-testing-techniques.html

Shevchenko, A., n.d. *Functional Test Plans: The Right Way!.* [Online]
Available at: https://www.developer.com/guides/functional-test-plans-the-right-way/

tutorialspoint, n.d. *Test Case Design Technique.* [Online]
Available at:
https://www.tutorialspoint.com/software_testing_dictionary/test_case_design_technique.htm