

**SOFTWARE BUGS MANAGEMENT
(ISO/IEC/IEEE 29119 STANDARD) SYSTEM**

By
CHIN JUN KANG

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONS)
Faculty of Information and Communication Technology
(Kampar Campus)

JAN 2020

REPORT STATUS DECLARATION FORM

Title: SOFTWARE BUGS MANAGEMENT
(ISO/IEC/IEEE 29119 STANDARD) SYSTEM

Academic Session: JAN 2020

I CHIN JUN KANG
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

No. 32, Jalan Besar,
35500 Bidor,
Perak

Tan Teik Boon
Supervisor's name

Date: 24 April 2020

Date: 24 April 2020

**SOFTWARE BUGS MANAGEMENT
(ISO/IEC/IEEE 29119 STANDARD) SYSTEM**

By
CHIN JUN KANG

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONS)
Faculty of Information and Communication Technology
(Kampar Campus)

JAN 2020

DECLARATION OF ORIGINALITY

DECLARATION OF ORIGINALITY

I declare that this report entitled “**SOFTWARE BUGS MANAGEMENT**

(ISO/IEC/IEEE 29119 STANDARD) SYSTEM” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other awards.

Signature :  _____

Name : CHIN JUN KANG _____

Date : 24/04/2020 _____

ACKNOWLEDGEMENTS

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ts. Tan Teik Boon who has given me this bright opportunity to engage in a Software Bug Management project. It is my first step to establish a career in the software development field. A million thanks to you.

Furthermore, I would like to thank Ts. Yeck Yin Ping who introduce Mr Tan to me so that I can have a chance to work with Mr Tan. Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

ABSTRACT

ABSTRACT

Bug management is a crucial process in the software development process. Low-quality bug management could result in low-quality deliverable, resources wasted and profit lost. However, bug management is not an easy process, software developers often have a hard time managing software bugs. This is because a development team involves not only technical software developer but also non-technical business unit. It can be hard to communicate between the development team as the members have a different level of technical knowledge. To solve this problem, this project proposes a bug management system as a solution. Although there are many existing bug management systems, most of them do not provide a standardized bug management process. The proposed system provides a standardized bug management process following the *ISO/IEC/IEEE 29119* Software Testing Standard. The proposed system also aims to improve the understanding of the software bugs and improve the communication between the development team. With the help of the proposed system, the software developer can focus on the software developing and software development team including business unit can communicate more effectively.

TABLE OF CONTENTS

TITLE PAGE	i
TITLE PAGE	i
DECLARATION OF ORIGINALITY	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	xi
LIST OF ABBREVIATIONS	xii
 CHAPTER 1: INTRODUCTION	 1
1.1. Problem Statement	1
1.2. Background and Motivation	2
1.3. Project Scope	3
1.4. Project Objectives	3
1.5. Impact, Significance and Contribution	4
 CHAPTER 2: LITERATURE REVIEW	 5
2.1. Bugs	5
2.1.1 Bug Report	5
2.1.2 Bug life cycle	6
2.2. ISO/IEC/IEEE 29119 Software Testing Standard	7
2.2.1 ISO/IEC/IEEE 29119-1:2013 – Concepts & Definitions	7
2.2.2 ISO/IEC/IEEE 29119-2:2013 – Test Processes	7
2.2.3 ISO/IEC/IEEE 29119-3:2013 – Test Documentation	10
2.2.4 ISO/IEC/IEEE 29119-4:2015 – Test Techniques	11
2.2.5 ISO/IEC/IEEE 29119-5:2016 – Keyword Driven Testing	13
2.3. Review of Existing Works	14
2.3.1 Airbrake	14
2.3.2 Backlog	16
2.3.3 Zoho BugTracker	19
2.3.4 Comparison of Key Features	21

CHAPTER 3: PROPOSED METHOD	23
3.1. Design Specifications	23
3.1.1 Methodologies	23
3.1.2 General Work Procedures	23
3.1.4 Technologies and Tools Involved.	25
3.1.5 System Performance Definition	26
3.1.6 Verification Plan	26
3.2. System Design	28
3.2.1 System Flowchart	28
3.2.2 Use Case Diagram	29
3.2.3 Use Cases Description	30
3.2.4 Activity Diagrams	37
CHAPTER 4: ISO/IEC/IEEE 29119 STANDARD SPECIFICATION	44
4.1. General	44
4.1.1 Metrics	44
4.1.2 Classification	44
4.2. Form Design Specification	45
4.2.1 Test Case	45
4.2.2 Bug / Test Incident	46
CHAPTER 5: IMPLEMENTATION AND TESTING	49
5.1. Database Set up	49
5.2. Authentication	52
5.3. React Native Components	53
5.3.1 Community built libraries and components	53
5.3.2 Self-built components	54
5.4. User Interface Design	58
5.4.1 Mobile Platform	58
5.4.2 Web Platform	69
5.5. Use Case Testing	76
5.6. Implementation Issues and Challenges	78
CHAPTER 6: CONCLUSION	80
6.1. Project Review	80

TABLE OF CONTENTS

6.2. Novelties and Contribution	80
6.3. Future Work	81
BIBLIOGRAPHY	82
POSTER	84
PLAGIARISIM CHECK RESULT	85
CHECKLIST FOR FYP2 THESIS SUBMISSION	87

LIST OF FIGURES

Figure 2.1.1: The defect life cycle (Graham et al. 2008)	6
Figure 2.2.1: Test Management Process (International Organization for Standardization 2013)	8
Figure 2.2.2: Test Planning Process (International Organization for Standardization 2013)	8
Figure 2.2.3: Test Monitoring and Control Process (International Organization for Standardization 2013)	9
Figure 2.2.4: Dynamic Test Process (International Organization for Standardization 2013)	9
Figure 2.2.5: Test Design Techniques (International Organization for Standardization 2013)	12
Figure 2.3.1: Logo of Airbrake	14
Figure 2.3.2: Real-time errors capture and tracing features ('Airbrake' n.d.)	14
Figure 2.3.3: Real-time deployment monitoring and quality analysis features ('Airbrake' n.d.)	15
Figure 2.3.4 Logo of Backlog	16
Figure 2.3.5: Project management features ('Backlog' n.d.)	17
Figure 2.3.6: Bugs tracking and team collaboration features ('Backlog' n.d.)	17
Figure 2.3.7: Task management and version control features ('Backlog' n.d.)	17
Figure 2.3.8: Logo of Zoho BugTracker	19
Figure 2.3.9: Bug management and Custom form field features ('Zoho BugTracker' 2010)	19
Figure 2.3.10: Forum and working timesheet features ('Zoho BugTracker' 2010)	20
Figure 3.1.1: The flow of the development process	23
Figure 3.2.1: System Flowchart	28
Figure 3.2.2: Use Case Diagram	29
Figure 3.2.3: Activity diagram of the Login use case	37
Figure 3.2.4: Activity diagram of the Sign up use case	38
Figure 3.2.5: Activity diagram of the View feed list use case	39
Figure 3.2.6: Activity diagram of the View dashboard use case	39
Figure 3.2.7: Activity diagram of the View team use case	40
Figure 3.2.8: Activity diagram of the Create team use case	40

Figure 3.2.9: Activity diagram of the Add project use case	41
Figure 3.2.10: Activity diagram of the Add test case use case	42
Figure 5.1.1: Database structure of profiles	49
Figure 5.1.2: Database structure of teams	49
Figure 5.1.3: Database structure of projects	50
Figure 5.1.4: Database structure for feeds	50
Figure 5.1.5: Database structure of requirements	51
Figure 5.1.6: Database structure for test cases	51
Figure 5.1.7: Database structure for bugs	52
Figure 5.3.1: Screenshot of PasswordField.js	54
Figure 5.3.2: Screenshot of ProgressBar.js	55
Figure 5.3.3: Screenshot of Alert.js	56
Figure 5.3.4: Screenshot of style for Alert.js	57
Figure 5.4.1: The splash screen	58
Figure 5.4.2: The Login screen	59
Figure 5.4.3: The Signup screen	59
Figure 5.4.4: System alert when fields are invalid	59
Figure 5.4.5: The Feeds screen	60
Figure 5.4.6: Add function in the Feeds screen	60
Figure 5.4.7: Search function in the Feeds screen	60
Figure 5.4.8: The Drawer Navigation	61
Figure 5.4.9: The Bottom Bar Navigation	61
Figure 5.4.10: The Profile screen	62
Figure 5.4.11: The Team screen	62
Figure 5.4.12: Edit user list function in the Team screen	62
Figure 5.4.13: The Dashboard Screen - 1	63
Figure 5.4.14: The Dashboard Screen - 2	63
Figure 5.4.15: The Add Project Screen	64
Figure 5.4.16: The Requirement Screen	64
Figure 5.4.17: The Add Test Case Screen - 1	64
Figure 5.4.18: The Test Case Screen - 2	64
Figure 5.4.19: The Add Project Screen	65
Figure 5.4.20: The Requirement Screen	65
Figure 5.4.21: The Project List Screen	66

LSIT OF FIGURES

Figure 5.4.22: The Project Home Screen	66
Figure 5.4.23: The Project Requirement Screen	67
Figure 5.4.24: The Project Test Case Screen	67
Figure 5.4.25: The Project Bug Screen	67
Figure 5.4.26: The Requirement Screen	68
Figure 5.4.27: The Test Case Screen	68
Figure 5.4.28: The Bug Screen	68
Figure 5.4.29: Screenshot of Login Page	69
Figure 5.4.30: Screenshot of Signup Page	69
Figure 5.4.31: The Home Page	70
Figure 5.4.32: The Profile Page	70
Figure 5.4.33: Screenshot of Project Home Page	71
Figure 5.4.34: The Team Page	71
Figure 5.4.35: The Dashboard Page	72
Figure 5.4.36: The Add Project Page	72
Figure 5.4.37: The Add Requirement Page	73
Figure 5.4.38: The Add Test Case Page	73
Figure 5.4.39: The Add Bug Page - 1	74
Figure 5.4.40: The Add Bug Page - 2	74
Figure 5.4.41: The Requirement Page	75
Figure 5.4.42: The Test Case Page	75
Figure 5.4.43: The Bug Page	76

LIST OF TABLES

Table 2.3.1: Key features comparison table	21
Table 3.1.1: Laptop specifications	25
Table 3.1.2: Mobile device specifications	25
Table 3.2.1: Use case description of the Login use case	30
Table 3.2.2: Use case description of the Sign up use case	31
Table 3.2.3: Use case description of the View feed use case	31
Table 3.2.4: Use case description of the View dashboard use case	32
Table 3.2.5: Use case description of the View team use case	32
Table 3.2.6: Use case description of the Create team use case	33
Table 3.2.7: Use case description of the Add project use case	34
Table 3.2.8: Use case description of the Add requirement use case	34
Table 3.2.9: Use case description of the Add test case use case	35
Table 3.2.10: Use case description of the Add bugs use case	35
Table 3.2.11: Use case description of the View project detail use case	36
Table 3.2.12: Activity diagram of the Add requirement use case	41
Table 3.2.13: Activity diagram of the Add bug use case	42
Table 3.2.14: Activity diagram of the View project detail use case	43
Table 4.1.1: ISO/IEC/IEEE 29119 standard general metrics	44
Table 4.1.2: Classification used in ISO/IEC/IEEE 29119 standards	44
Table 4.2.1: Form Fields Description for Test Case	46
Table 4.2.2: Form Template for Test Case	46
Table 4.2.3: Form Fields Description for bug	47
Table 4.2.4: Form Template for Bugs	48
Table 5.3.1: Libraries and components used and the repositories	53

LIST OF ABBREVIATIONS

<i>NIST</i>	National Institute of Standards and Technology
<i>U.S.</i>	United States
<i>ISO</i>	International Organization for Standardization
<i>IEC</i>	International Electrotechnical Commission
<i>IEEE</i>	Institute of Electrical and Electronics Engineers
<i>IP</i>	Internet Protocol
<i>OS</i>	Operating System
<i>UML</i>	Unified Modeling Language
<i>UI</i>	User Interface
<i>etc.</i>	et cetera
<i>Anon.</i>	Anonymous
<i>n.d.</i>	No date
<i>pp.</i>	pages

CHAPTER 1: INTRODUCTION

1.1. Problem Statement

Poor bugs management could post problems in every software development lifecycle. First is the **misunderstanding of bugs**, not all bugs are caused by coding errors, misunderstanding can be caused by miscommunication within the development team, unclear requirements, misunderstanding of requirements and poor documentation of bugs. Sometimes issues are reported technically (as bugs), but a development team consisting of non-technical members such as customer, stakeholders, organization, etc. It could be hard for non-technical members to involve in the software development process this can cause the requirement and issue to become unclear.

Moreover, fixing bugs can become not effective and time-consuming if **the developers do not have a standardized testing process**. As mentioned by (Ahonen et al. 2004) one of the challenges in testing defects is that it is hard to ensure that all members in the development team follow good practices. During a conference, (Kajko-Mattsson & Bjornsson 2007) have also reported that organizations do not document testing processes properly. These testing-related problems can delay the time a bug is detected during the software development process and result in late in solving the bugs.

Last but not least, it will be an **ineffective use of intellectual** if a bug is not documented well (Ahonen et al. 2004). A development team may consist of experts from different domain/fields, an issue found by a developer may from a different field, he/she may use a long time to solve the problem. In a larger development team, there will be different members to do the testing which may not have coding skill to resolve the issue. The member who found the bug has to determine the skills needed to solve the problem and passes the bug to the members who have the required skill or informs every member in the team to find out who has the ability to solve the problem. If the bug is documented well, other developers who are expert in the domain can view the documentation and resolve the problem effectively.

1.2. Background and Motivation

Software bugs are always a challenge in the software development process. Software bugs are errors occur in software including coding error, unexpected software behaviour, misunderstood user requirement etc. These software bugs can prolong the project development process and delay the delivery of the final product. These problems arise because bugs are not well-documented and most of the time it is very difficult to document bugs (Singh 2013) thus developers often use Bug Management System to ease the bugs documentation process.

A bug management system is a tool for developers to report bugs, sort and filter bugs, assign bugs to individuals and tracking the bugs to resolution. A bug management system provides a standard for developers to document and distribute bugs so developers can focus on developing and not documenting. A bug management system eases the process of bug managing so the developers can focus on developing the software and thus reduce development duration and increase product quality.

Bug management is the process of documenting, keep tracking and eventually fixing the software bugs during the software development process. Bug management involves different techniques to manage software bugs such as bug report, bug life cycle monitoring, etc.

In this project, the *ISO/IEC/IEEE 29119* Software Testing Standard is used to provide a set of standardized formats for software testing and bug management processes. Bug documentation with standardized format help the development team to understand the documentation easier and hence result in solving the bugs better. A software testing standard ensures the developers in the software development team to deliver quality products.

1.3. Project Scope

The scope of the project including the following three:

- To develop a system with bug management functionalities.
- To develop the bug management system in both the web and mobile platforms.
- To develop a centralized database to maintain the bugs for the bug management system.

1.4. Project Objectives

The main objective of this project is to build a system that is able to assist developers, stakeholders, and customers in understanding the program development process. The proposed system follows the *ISO/IEC/IEEE 29119* Software Testing Standard to provide better bugs management to the members involved in the software development process. The objective can be broken down into sub-objectives:

- To offer bug tracking and management functionalities to the users such as bug report, keep track of bug's status, assign bugs to users, etc.
- To keep track the progress of bug management.
- To standardize the bug tracking and management process by using software testing standard.

1.5. Impact, Significance and Contribution

According to the report from the National Institute of Standards and Technology (NIST) (Strate & Laplante 2013), the U.S. economy loses \$60 billion a year because of software defects and fixing these defects earlier could save \$22 billion a year for the U.S. economy. The statistic shows how crucial bug management is in the software development process. It is the development team's job to test, keep track and resolve the bugs but it is not easy to deal with bugs. If the development team cannot resolve bugs effectively, the system will become buggy which highly decreases the usability of the system and even cause the system cannot meet the user requirement and result in failure. This project aims to create a tool for the development teams to help them in producing quality products.

Although there are many existing bug management systems, most of the systems are not standardized. This can cause solving bugs becomes ineffective and time-consuming. This project provides a standard based on the *ISO/IEC/IEEE 29119* Software Testing Standard so the developers can be more consistent at work and solve the bugs effectively. Another limitation for the existing works is that most systems only concern about the developer aspect and disregard for the business aspect of the development team. This project considers the business aspect of the development team and provides easy to understand non-technical information for the progress of the developments.

CHAPTER 2: LITERATURE REVIEW

2.1. Bugs

A bug is a failure, flaw, error or abnormal behaviour in software that causes the software to behave differently from what is expected. A bug can be a logical error, incorrect implementation of code, design error, syntax error or unfulfillment of user requirements.

2.1.1 Bug Report

Bug reports are the primary method users and testers of a system to communicate an issue to the developers. When a bug is detected, a bug report should be generated and send to the developers for resolving. The content of a bug report is important as it helps developers to understand the issue, find and fix the bugs. After comparing several sources by (*Bug* n.d., *Defect Management Process in Software Testing (Bug Report Template)* n.d., Davies & Roper 2014, *What is Defect or bugs or faults in software testing?* 2012), the following parameters are found to be important to be included in a bug report to assist the developers in resolving an issue.

- Bug ID – A unique identifier for each bug
- Bug Description – Detailed description of the bug including expected behaviour and observed behaviour.
- Build Information – The system and the release version of the system in which the bug was detected.
- Environment – Description of the environment the bug was found including operating system, browser system and more.
- Steps to Reproduce – The detailed steps for the developers to reproduce the same issue in different machines.
- Date Raised – Date when the bug is reported.
- Reported by – Name or ID of the tester who detected and reported the bug.
- Status – The current status of the bug, could be New, Assigned, Open, etc.
- Assigned to / Fixed by – Name or ID of the developer who assigned to solve the issue or fixed the bug.
- Date Closed – Date when the issue is closed.

- Severity – Level of the impact of the bug on the system.
- Priority – Level of urgency in resolving the bug.

2.1.2 Bug life cycle

In software development, the bug life cycle describes the statuses a bug goes through during its lifetime. It differs between organizations and projects, some may prefer a simpler life cycle like presented by (Graham et al. 2008, D'Ambros et al. 2007) while others may adopt a more extensive life cycle.

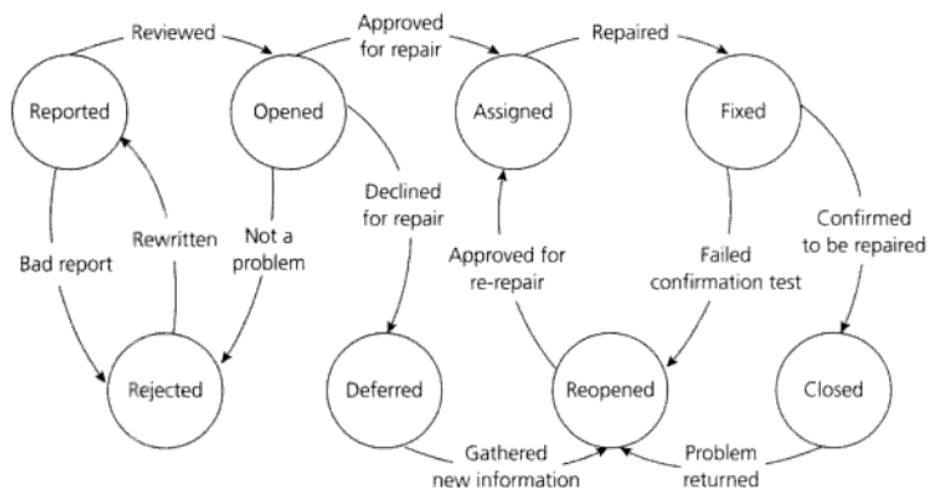


Figure 2.1.1: The defect life cycle (Graham et al. 2008)

- Reported – When a bug is first detected and reported, it is in the Reported state.
- Opened – After the bug is reviewed, it is in the Opened state.
- Assigned – After the bug is approved by the development team lead tester, the bug is assigned to the developers.
- Fixed – Once the developers have repaired the bug, it is in the Fixed state and will go through a confirmation test.
- Reopened – If the bug is checked not fixed after the confirmation test, it is reopened and repeat until it passes the confirmation test.
- Closed – The final state of the bug, after it is fixed and passes the confirmation test it is closed to indicate that the bug is repaired and do not exist in the program anymore.

- Rejected – The developers can reject the bug by a few reasons, the issue is not reproducible, duplicated issue, it is not a bug, etc..
- Deferred – Deferred state means the bug is expected to be fixed in the next releases for the reason that the bug has a low severity and low priority.

2.2. ISO/IEC/IEEE 29119 Software Testing Standard

ISO/IEC/IEEE 29119 is a software testing standard developed by ISO Software Testing Group of the ISO/IEC JTC1/SC7 Software and Systems Engineering committee starting from 2007. The ISO/IEC/IEEE 29119 consists of a series of 5 international standards for systems and software engineering testing. Each release of the ISO/IEC/IEEE 29119 focus on an aspect of software testing such as concepts and definitions, test processes, test documentation, test techniques and keyword-driven testing.

2.2.1 ISO/IEC/IEEE 29119-1:2013 – Concepts & Definitions

Part 1 of the ISO/IEC/IEEE 29119 standard release in 2013 and covers the concepts and definitions of software testing. It provides practical examples of the application of each concept and introduces vocabulary used throughout the ISO/IEC/IEEE 29119 series to help the reader in understanding the concepts presented in the ISO/IEC/IEEE 29119 series.

2.2.2 ISO/IEC/IEEE 29119-2:2013 – Test Processes

Part 2 describes more details about test processes with a generic software testing model that can be used in any software development. The series presents a multi-layer process model for governing, managing and implementing software testing. The diagrams provided by (International Organization for Standardization 2013) below describe the model in detail.

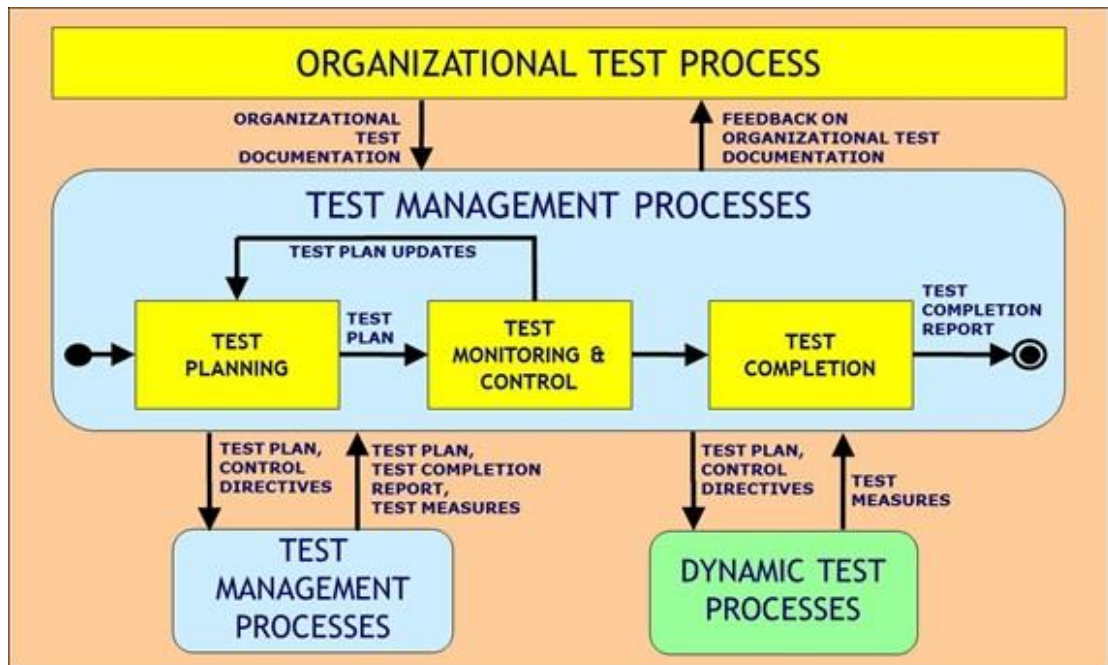


Figure 2.2.1: Test Management Process (International Organization for Standardization 2013)

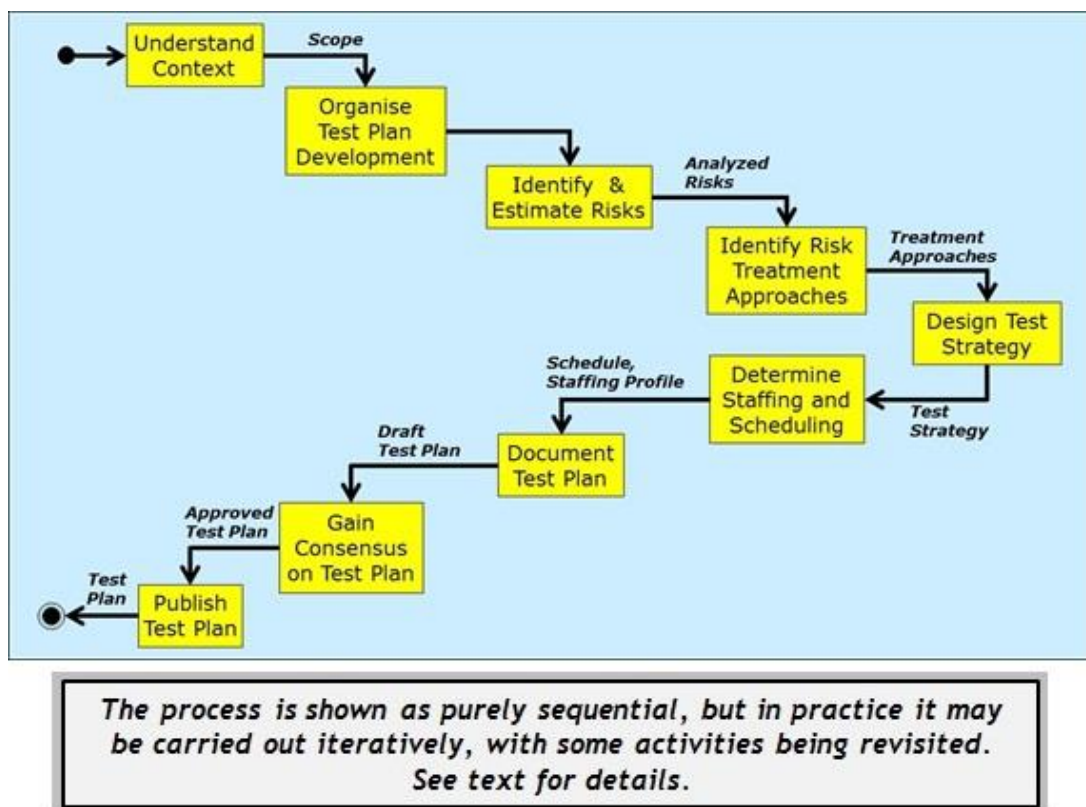


Figure 2.2.2: Test Planning Process (International Organization for Standardization 2013)

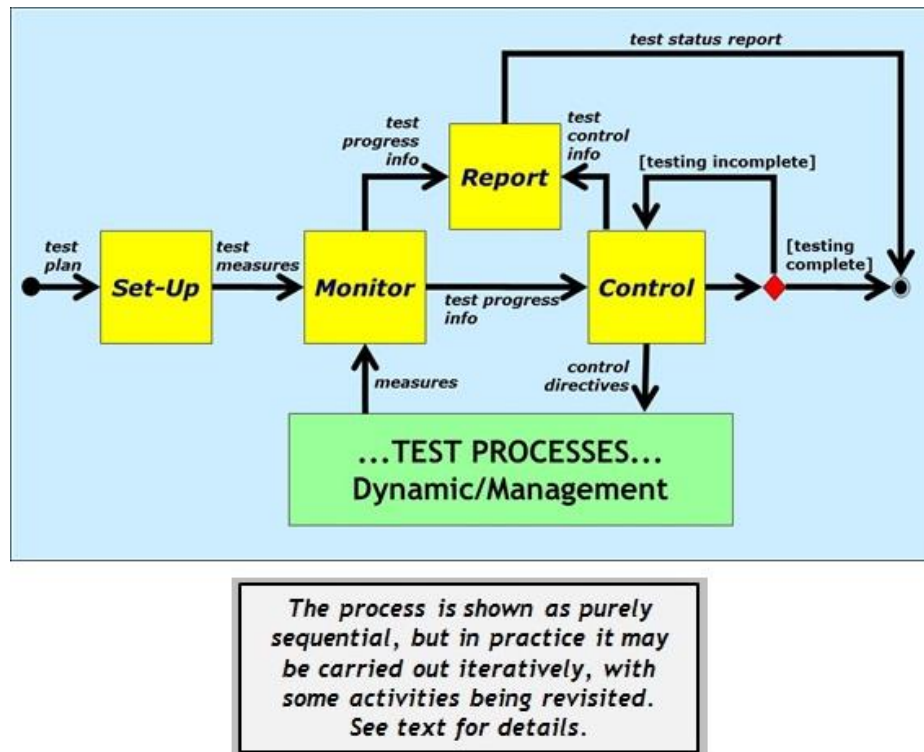


Figure 2.2.3: Test Monitoring and Control Process (International Organization for Standardization 2013)

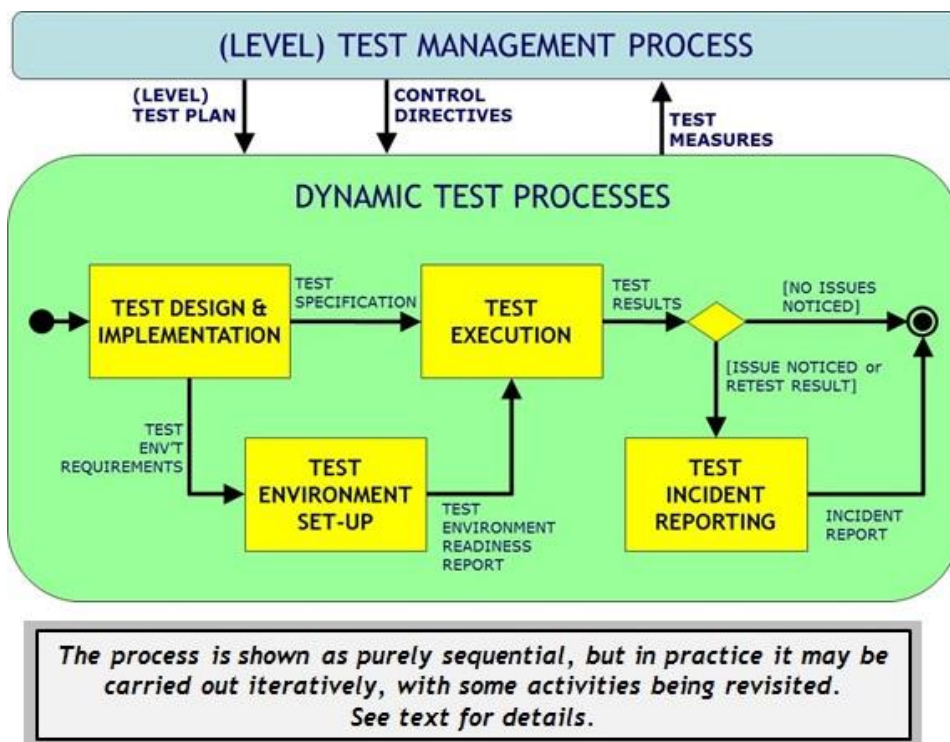


Figure 2.2.4: Dynamic Test Process (International Organization for Standardization 2013)

2.2.3 ISO/IEC/IEEE 29119-3:2013 – Test Documentation

Part 3 introduces the software test documentation with standard templates. All templates follow the three test process levels defined in ISO/IEC/IEEE 29119-2 and can be customized for any organization and software development process. The test documentation standard presented is built on top of the IEEE 829 Test Documentation standard.

The documents defined in this part are as follows (International Organization for Standardization 2013):

- **Organizational Test Process Documentation:**
 - Test Policy
 - Organizational Test Strategy
- **Test Management Process Documentation:**
 - Test Plan (including a Test Strategy)
 - Test Status Report
 - Test Completion Report
- **Dynamic Test Process Documentation:**
 - Test Design Specification
 - Test Case Specification
 - Test Procedure Specification
 - Test Data Requirements
 - Test Data Readiness Report
 - Test Environment Requirements
 - Test Environment Readiness Report
 - Actual Results
 - Test Result
 - Test Execution Log
 - Test Incident Report

2.2.4 ISO/IEC/IEEE 29119-4:2015 – Test Techniques

Part 4 of the standard covers software test design techniques based on the BS-7925-2 Component Testing standard. It defines different types of testing and testing techniques that can be applied as well as provides detailed examples of the implementation of each technique.

Following are the testing type discussed in the standard(International Organization for Standardization 2013):

- Accessibility Testing
- Backup/Recovery Testing
- Compatibility Testing
- Conversion Testing
- Disaster Recovery Testing
- Functional Testing
- Installability Testing
- Interoperability Testing
- Localization Testing
- Maintainability Testing
- Performance-Related Testing (e.g. Load Testing, Stress Testing and Endurance Testing)
- Portability Testing
- Procedure Testing
- Reliability Testing
- Security Testing
- Stability Testing
- Usability Testing

The diagram below shows the test design techniques that are discussed in the standard (International Organization for Standardization 2013):

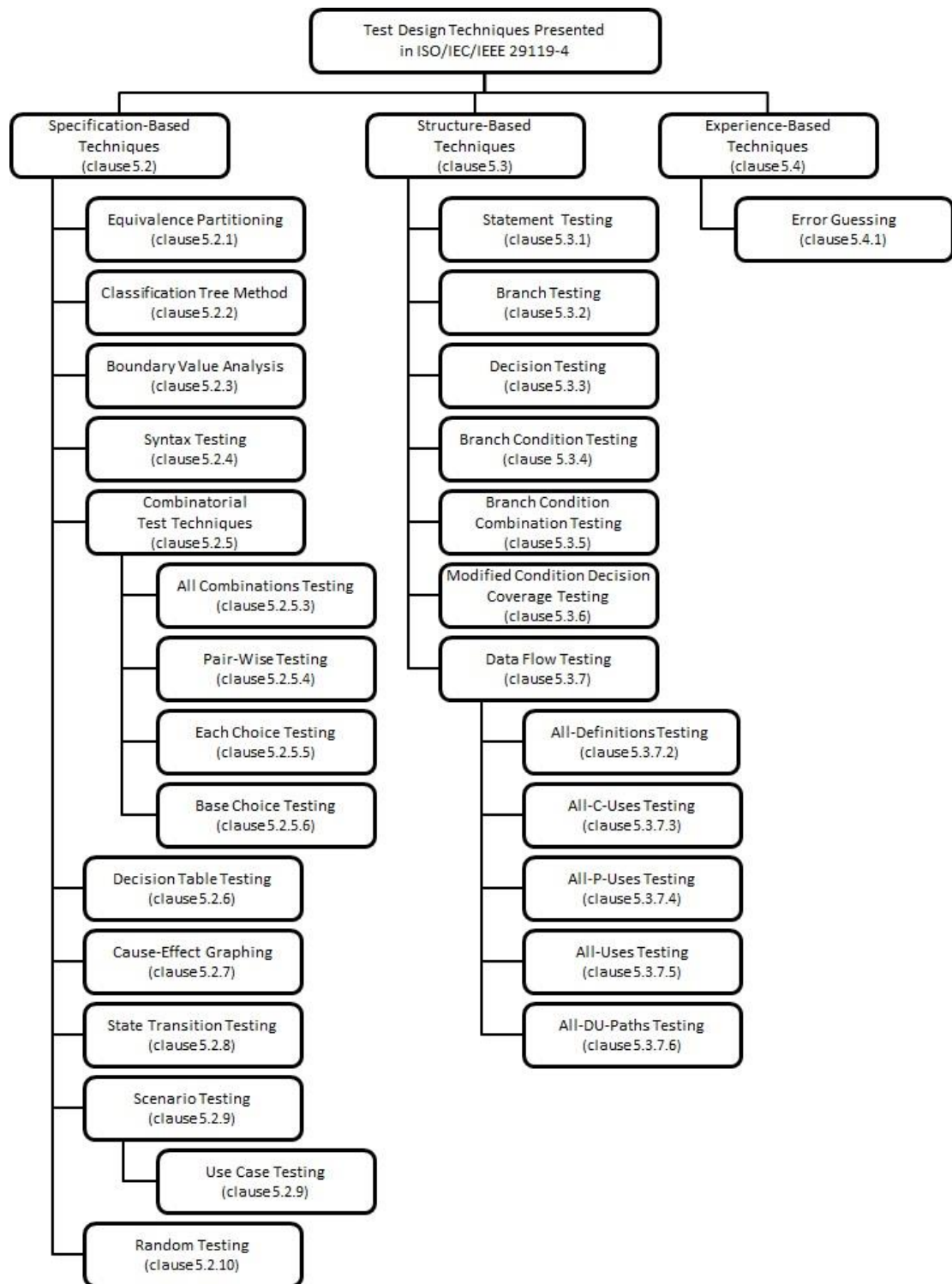


Figure 2.2.5: Test Design Techniques (International Organization for Standardization 2013)

2.2.5 ISO/IEC/IEEE 29119-5:2016 – Keyword Driven Testing

Part 5 of the standard introduce Keyword-Driven Testing, an approach of software testing that using predefined keywords in describing test cases. The standard covers the introduction in Keyword-Driven Testing, application of Keyword-Driven Testing, frameworks, data interchange, roles and tasks in Keyword-Driven Testing and some basic Keywords. The standard also discussed on benefits and issues of Keyword-Driven Testing.

2.3. Review of Existing Works

There are many similar systems exist in the market, 3 of them are selected for review.

2.3.1 Airbrake



Figure 2.3.1: Logo of Airbrake

Airbrake is a cloud-based monitoring tool for software development teams to manage and monitor deployed projects. Airbrake is an API that automatically captures uncaught errors in deployed and ongoing projects and reports to development teams directly. Currently, there are many well-known companies using Airbrake including Twitch, TED, Zenefits, Salesforce, SoundCloud, Adobe and more.

Key Features:

- Capture real-time errors automatically from deployed projects
- Real-time issues tracking and monitoring
- Errors code tracing
- Real-time deploy monitoring
- Deployment quality analysis and insight
- User experience monitoring
- Security features to protect users data.

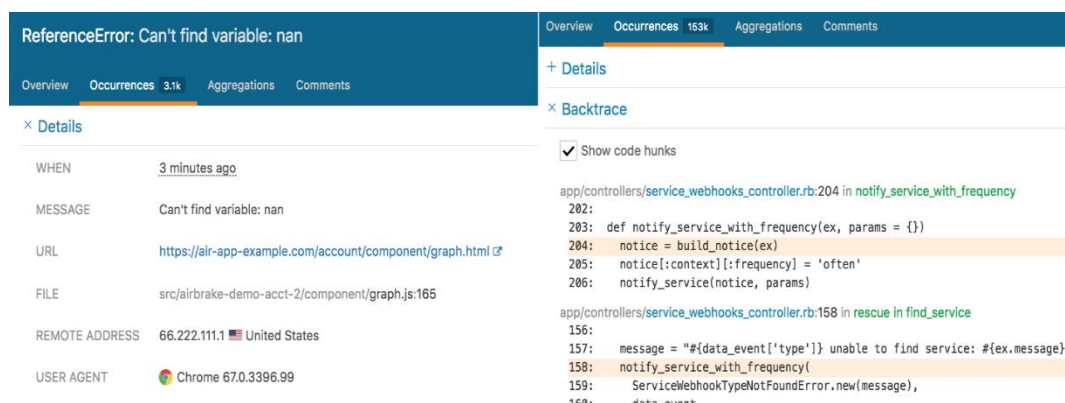


Figure 2.3.2: Real-time errors capture and tracing features ('Airbrake' n.d.)

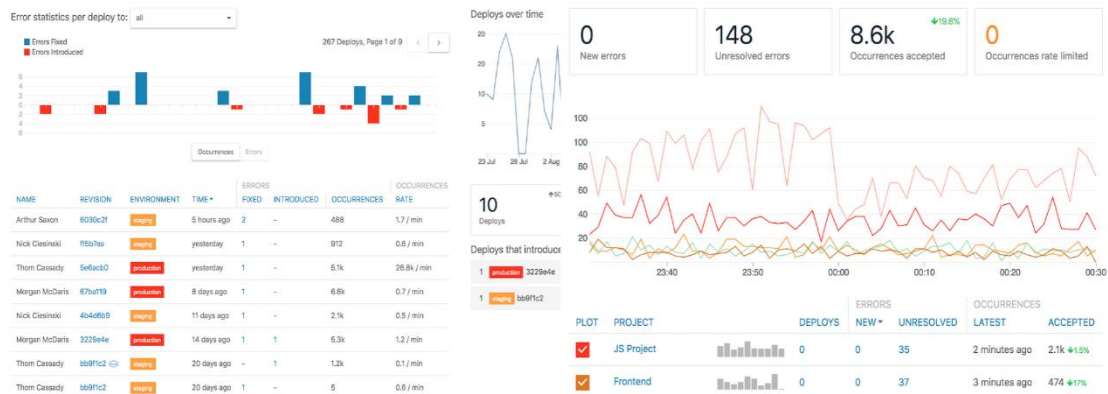


Figure 2.3.3: Real-time deployment monitoring and quality analysis features ('Airbrake' n.d.)

Strength:

- Errors are captured automatically in real-time as they arise on client sides.
- Provides real-time deployment quality monitoring and analysis.
- Notification for new issues.
- Support for a vast amount of programming languages.
- Support for numbers of third-party integrations.
- Easy to install and integrate it into software projects.
- Highly secured data protection.
- API is available for developers.
- Accessible anywhere anytime as a web-based service.
- Mobile-friendly web design.

Weakness:

- Complex user interface.
- Lack of team communication functionality.
- Lack of support for different platforms.
- Users are unable to attach additional files on an error.
- Lack of test case management features.
- Lack of requirement management features.
- Lack of traceability of bugs.

2.3.2 Backlog



Figure 2.3.4 Logo of Backlog

Backlog is an all-in-one software development assisting tool that includes functions such as project management, bug tracking, task management, version control and more. In Backlog, developers can communicate effectively throughout the software development process. Currently, companies that using Backlog include Omron, SoftBank Robotics, Adobe, TransferWise and Vanilla Air.

Key Features:

- Push notifications for new issues.
- Bugs tracking and managing.
- Extensive task managing.
- Version control support with integration with Git.
- Centralize repository for each project.
- Project scheduling with Gantt charts.
- Access control for IP addresses and roles.
- Team collaboration using comments and attachments.
- Project wiki page.
- Customizable forms.
- Jira and Redmine importer.

CHAPTER 2: LITERATURE REVIEW

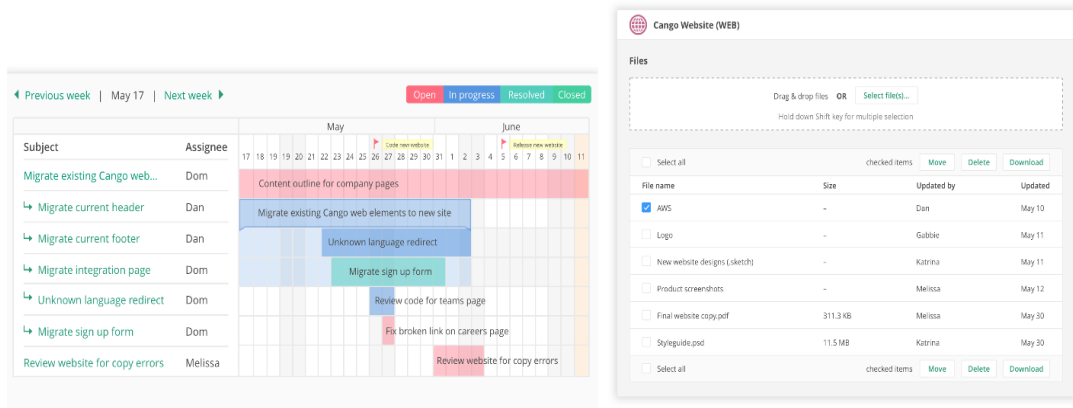


Figure 2.3.5: Project management features ('Backlog' n.d.)

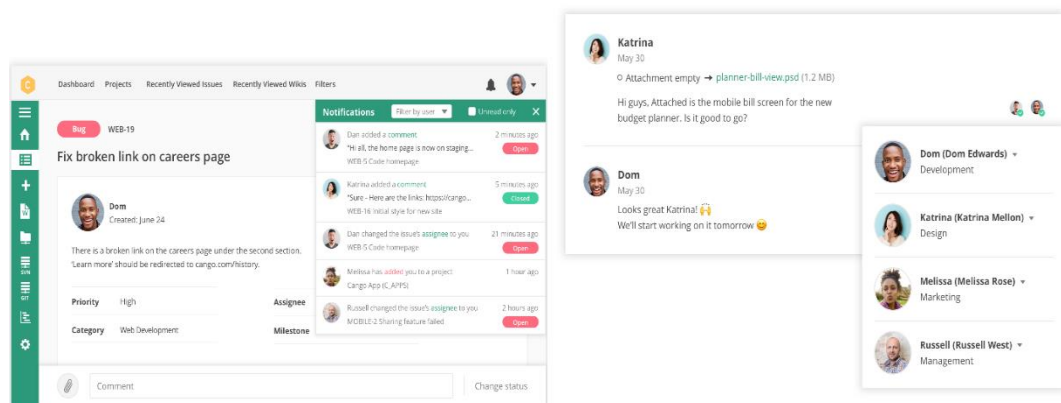


Figure 2.3.6: Bugs tracking and team collaboration features ('Backlog' n.d.)

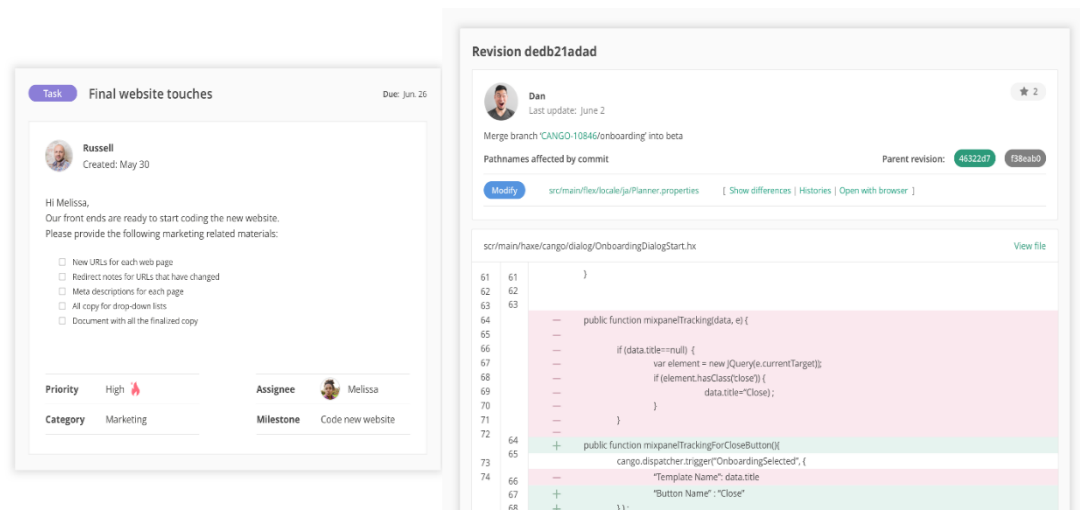


Figure 2.3.7: Task management and version control features ('Backlog' n.d.)

Strength:

- Create functionalities in team collaboration.
- Wiki page to help newly added developers to understand the project effectively.
- Supports for numbers of third-party integrations.
- Push notification for new issues.
- Simple user interface design.
- API is available for developers.
- Customizable forms to suit different projects and organization.
- Display project planning and progress using Gantt charts.
- Centralized repository for every project.
- Support for multiple platforms.
- Accessible anywhere anytime as a web-based service.
- Self-hosting is available for data protection.
- Version control is available.

Weakness:

- Lack of dashboard or reports for bug management.
- Status options for issues are insufficient and not customizable.
- Priority options for issues are insufficient and not customizable.
- Poorly implemented version control function.
- Notification functionalities are poorly implemented.
- Lack of test case management features.
- Lack of requirement management features.
- Lack of traceability of bugs.

2.3.3 Zoho BugTracker



Figure 2.3.8: Logo of Zoho BugTracker

Zoho BugTracker is a cloud-based bug tracking system that helps development teams in delivering issue free products. Zoho BugTracker provides simple bug management for the development team to track and fix bugs quickly and easily.

Key Features:

- Bug tracking and management.
- Milestone management.
- Integration with other Zoho Cloud products.
- Personalize interfaces.
- Timesheet for work hour record.
- Include chat and forums for better communications.

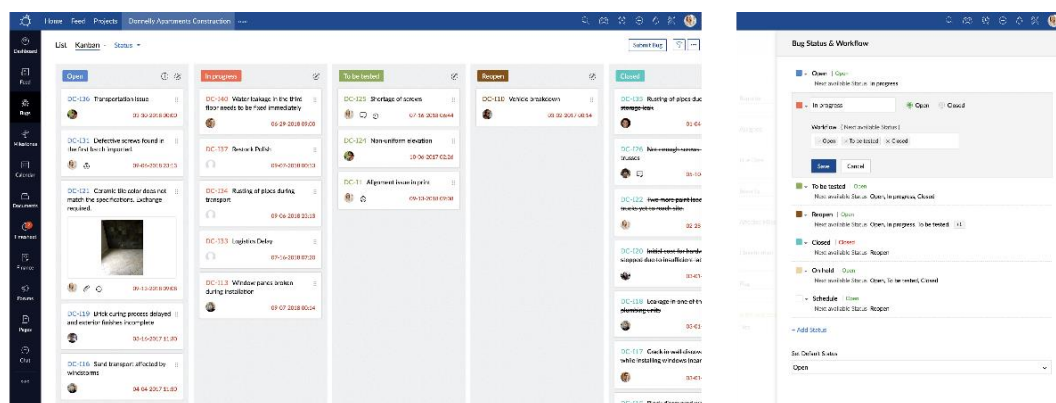


Figure 2.3.9: Bug management and Custom form field features ('Zoho BugTracker' 2010)

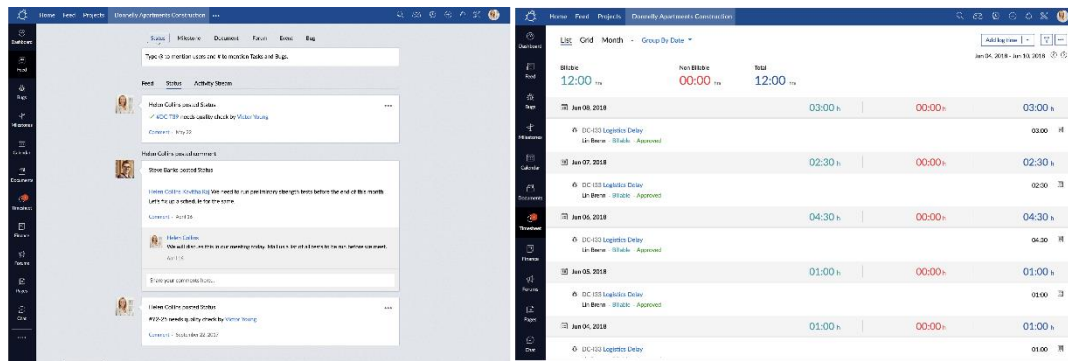


Figure 2.3.10: Forum and working timesheet features ('Zoho BugTracker' 2010)

Strength:

- Allow integration with other Zoho products.
- Working timesheet recording feature.
- Project milestone management feature.
- Customizable interface.
- Customizable forms to suit any projects and organizations.
- Forum and chat feature.
- Accessible across the web and mobile platform

Weakness:

- Lack of dashboard or reports for bug management.
- Lack of test case management features.
- Lack of requirement management features.
- Lack of traceability of bugs.

2.3.4 Comparison of Key Features

	Airbrake	Backlog	Zoho BugTracker	Proposed System
Project Management				
Group Bugs by Projects	✓	✓	✓	✓
Requirements Management				✓
Track overall progress	✓	✓	✓	✓
Gantt Chart		✓		
Burndown chart		✓		
Agile Board				
Dashboard for project	✓	✓		
Bug Management				
Automatic capture error in real-time	✓			
Automatic include environment information	✓			
Record bugs	✓	✓	✓	✓
Show user feeds	✓	✓	✓	✓
Backtraces & Breadcrumbs for errors	✓			
Navigate to code which error occur	✓			
Template for forms	✓	✓	✓	✓
Custom fields for forms		✓	✓	
Filters	✓	✓	✓	✓
Notification for new issues	✓	✓	✓	✓
Customize notifications	✓			
Dashboard for bug management	✓		✓	✓
Add issues via email		✓		
Test Case Management				
Create test case				✓
Assign test case to testers				✓
Link requirements, test cases and bugs				✓
Progress Tracking				
Progress tracking (bug density, test progress ...)				✓
Dashboard for progress				✓
Platform				
Web application	✓	✓	✓	✓
iOS app		✓	✓	✓
Android app		✓	✓	✓
Mobile friendly web sites	✓			

Table 2.3.1: Key features comparison table

It is clear that each of the systems reviewed focus in different domain and included different functionality. Airbrake is focusing on deployment management and capture error in real-time, Backlog focus in project management and task management and Zoho BugTracker focus in bug management and team communication while the proposed system will focus on bug management with ISO/IEC/IEEE 29119 standards.

Although each system focusses in different domains, there are similar features included in all 3 of the systems and it shows that these are good features in a bug management system and relevant features should be included in the proposed system. The proposed system also includes some features that can enhance team collaboration to improve usability as well as effectiveness in resolving issues. It is also noticed that there are limited progress tracking and bug tracing features in those systems.

Progress tracking is an important feature in a bug management system because it monitors the software development progress and makes sure the bugs are fixed in time and not delaying the software release. Progress tracking features such as bug density and test progress allows users to keep track which module is heavily bugged and assign more resource to the module to ensure the product delivered in time.

Bug tracing features ensure the developing system meets the requirements and goals as well as ensures the bugs are resolved in the right manner. Bug tracing allows users to link bugs to test cases and requirement, this gives developers more understanding on what is the actual problem and how to fix the bugs.

Both progress tracking and bug tracing are particularly important to the business unit in the development team. As discussed in Chapter 1, a development team consists not only technical members but also business unit such as stakeholders, customers, etc. and they do not understand Software Bugs which are reported in technical terms. Progress tracking and bug tracing features help the business unit to understand is the system developed following the goals set and how is the progress.

For the platform selection, the proposed system will be supporting web application, iOS and Android OS. A web-based application is convenient for users as it can be accessed in anywhere anytime. As for mobile platforms, it is better to implement an app so the users can get push notifications when new issues added which web application is unable to.

CHAPTER 3: PROPOSED METHOD

3.1. Design Specifications

3.1.1 Methodologies

The development model used to develop the proposed system is the Extreme Prototyping Model in which prototypes are built, evaluated and refined throughout the whole development process.

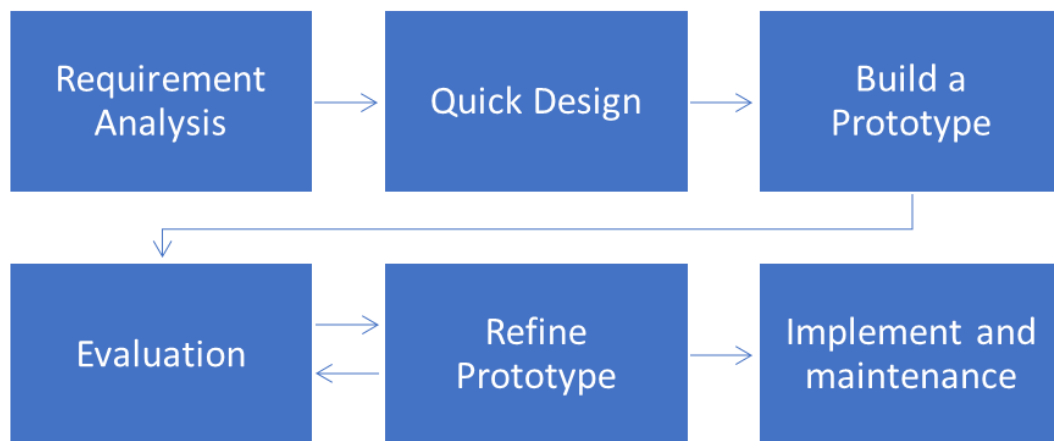


Figure 3.1.1: The flow of the development process

Figure 4.1-1 shows the flow of the development process for the proposed system. Note that the evaluation phase and refining prototype phase form a cycle, prototypes are evaluated and refined until the prototype meets all the user requirements. Using Prototype Model ensures the proposed system meets all the user requirements and can be deployed in time with high usability.

3.1.2 General Work Procedures

1. Requirement Analysis

The development process starts with analysing the requirement for the proposed system. The requirements are gathered by doing research on existing works in the bugs management area and review of existing bug management products in the market then useful features are analysed and selected as user requirements.

2. Quick Design

The proposed system is designed based on the user requirements gathered from the first phase using system UML diagrams such as use-case diagram, activity diagram and sequence diagram.

3. Build a Prototype

In this phase, a basic front-end prototype with minimal functionalities and UI design is built based on the design created in the second phase.

4. Evaluation

After the first prototype is created, it is presented to the user for evaluation. The feedback and comments from the user are gathered and analysed for refining the user requirements.

5. Refining Prototype

The prototype is then refined based on the user's feedback and comments gathered from the evaluation phase.

Then the evaluation phase and refining prototype phase is iterated until all the user requirements are met. Once the user satisfied with the prototype, actual functionalities such as data processing are implemented and integrated into the final prototype as the final system.

6. Implementation

After the final system is implemented, it is then thoroughly tested and deployed as the final deliverable after it passed all tests. This ensures the final deliverable of the proposed system meets all the user requirements with high usability.

3.1.4 Technologies and Tools Involved.

Hardware Tools

■ Laptop

Hardware	Specification
Operating system	Windows 10 Home Single Language
Processor	Intel® Core™ i7-4710HQ @ 2.50GHz
Ram	8.00 GB
Graphics	NVIDIA GeForce GTX 860M

Table 3.1.1: Laptop specifications

■ Mobile Device

Hardware	Specification
Operating system	Android OS v6.0
Processor	Hisilicon Kirin 935 @ 2.2GHz octa-core
Ram	3.00 GB

*Table 3.1.2: Mobile device specifications*Software Tools

■ Firebase Realtime Database

Firebase Realtime Database is a cloud-based NoSQL database developed by Google. It acts as a centralised database for application on different platforms to update and retrieve data from the database in real-time.

■ React Native

React Native is an open-source framework created by Facebook. It allows developers to develop native Android, iOS and Web application without needing to repeating the development for each platform. The proposed system can be developed with high accessibility for different platforms using React Native framework.

■ Visual Studio Code

Visual Studio Code is a code editor developed by Microsoft that allows users to build and debug modern applications. It also includes features and extensions that ease the development process for developers.

■ BrowserStack

BrowserStack is a web application that provides interactive cross-platform testing experience to users. BrowserStack provides live streaming of thousands of devices in different operating systems and sizes with native behaviour. BrowserStack is going to be the testing tool for this project to ensure it is compatible with different operating systems and different sizes of devices.

3.1.5 System Performance Definition

The performance of the final deliverable shall define by 3 main aspects:

- Usability and Accessibility
 - Ensure users can achieve their objectives in the least steps without extra training needed.
 - Able to be accessed from different platforms.
- Bug Management Functionalities
 - Users shall able to report bugs.
 - Users shall able to keep track of bugs status.
 - Test manager shall able to assign bugs to users.
- Progress Tracking
 - Users shall able to keep track of the progress of the bug management process.

3.1.6 Verification Plan

The verification plan for this project is defined in 3 phases:

➤ Coding Phase

During development for prototypes, the prototype is verified in parallel with coding to ensure the prototype is working as expected.

➤ Evaluation Phase

During the evaluation phase of each prototype, the prototype is presented to the user to get feedback and comments for refinement.

➤ Testing Phase

After the implementation of the final system, the system is tested thoroughly to ensure the final deliverable meeting all the requirement and behave as expected.

3.2. System Design

3.2.1 System Flowchart

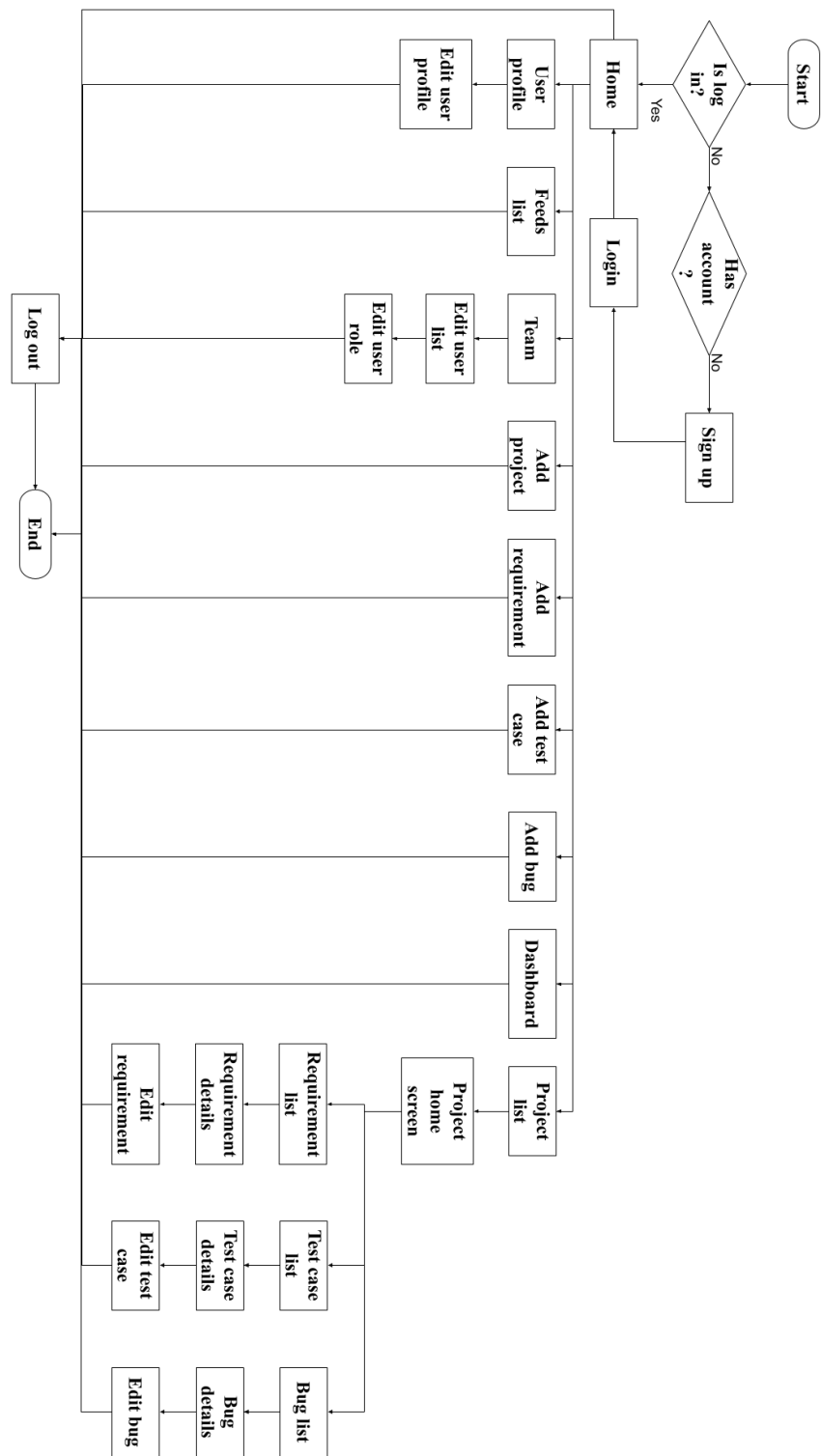


Figure 3.2.1: System Flowchart

3.2.2 Use Case Diagram

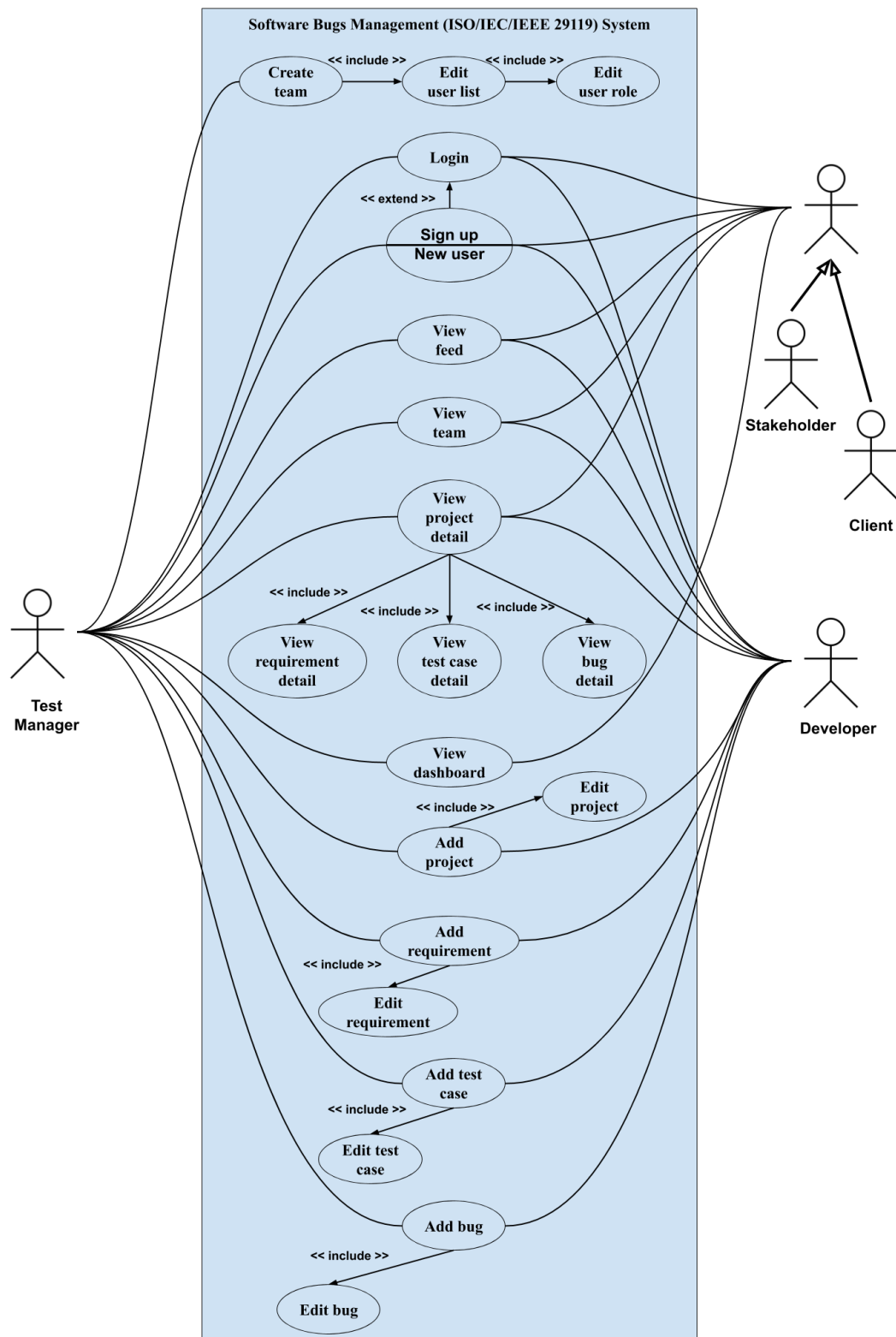


Figure 3.2.2: Use Case Diagram

3.2.3 Use Cases Description

Login

Use Case ID	UC001	
Feature	Login	
Purpose	To allow user to login with an account	
Actor	Test manager, tester/developer, business unit	
Trigger	The application is started up	
Precondition	User is not logged in.	
Main Flow	Step	Action
	1	User enter email and password.
	2	User press the “Login” button.
	3	System authenticate user’s account.
	4	System redirects user to the home screen.
Alternative Flow – 1a. Invalid email or password	Step	Action
	1a	User enters invalid email or password.
	2	System displays error message.

Table 3.2.1: Use case description of the Login use case

Sign up

Use Case ID	UC002	
Feature	Sign up	
Purpose	To allow user to create a new account	
Actor	Test manager, tester/developer, business unit	
Trigger	User press the “Sign up” button	
Precondition	User is not logged in; user has no account.	
Main Flow	Step	Action
	1	User enters email and password.
	2	User confirms password.
	3	System creates a new account.
	4	System log user in.
	5	System redirects user to the home screen.
Alternative Flow – 2a. Unable to confirm password	Step	Action
	2a	User enters different password during password confirmation.
	3	System displays error message.

*Table 3.2.2: Use case description of the Sign up use case*View feed

Use Case ID	UC003	
Feature	View feed	
Purpose	To allow user to view recent feeds	
Actor	Test manager, tester/developer, business unit	
Trigger	User enters feeds screen.	
Precondition	User is logged in.	
Main Flow	Step	Action
	1	System shows a list of most recent feeds.
	2	User press on feed.
	3	System displays feed details.

Table 3.2.3: Use case description of the View feed use case

View dashboard

Use Case ID	UC004	
Feature	View dashboard	
Purpose	To allow user to view chart and information about the progress of bug management	
Actor	Test manager, tester/developer, business unit	
Trigger	User enters the dashboard screen.	
Precondition	User is logged in.	
Main Flow	Step	Action
	1	User filter information option.
	2	System display relevant information and chart.

*Table 3.2.4: Use case description of the View dashboard use case*View team

Use Case ID	UC005	
Feature	View team	
Purpose	To allow user to view users in the team	
Actor	Test manager, tester/developer, business unit	
Trigger	User enters team screen.	
Precondition	User is logged in.	
Main Flow	Step	Action
	1	System displays user list.

Table 3.2.5: Use case description of the View team use case

Create team

Use Case ID	UC006	
Feature	Create team	
Purpose	To allow user to add users to the team and edit the user roles.	
Actor	Test manager	
Trigger	User enters team screen.	
Precondition	User is logged in. User role is test manage.	
Main Flow	Step	Action
	1	User creates a team.
	2	User adds other users to the team.
	3	User edits the user roles.
	4	System update feed list.
	5	System notifies related users.
Alternative Flow – 2a User adds a user which is already in the team	Step	Action
	2a	User adds a user which is already in the team.
	3	System display error message.
Alternative Flow – 2b User remove the last member in the team	Step	Action
	2a	User removes the last member in the team
	3	System displays error message.
Alternative Flow – 2c User removes the last admin in the team	Step	Action
	2a	User removes the last admin in the team
	3	System displays error message.

Table 3.2.6: Use case description of the Create team use case

Add project

Use Case ID	UC007	
Feature	Add project	
Purpose	To allow user to create project.	
Actor	Test manager, tester/developer	
Trigger	User enter add project screen.	
Precondition	User is logged in. User role is test manager or developer.	
Main Flow	Step	Action
	1	User enter project information.
	2	System saves the project information in the database.
	3	System update feed list.
	4	System notifies related users.
Alternative Flow – 1a User did not enter mandatory fields	Step	Action
	1a	User did not enter mandatory fields.
	2	System display error message.

*Table 3.2.7: Use case description of the Add project use case*Add requirement

Use Case ID	UC008	
Feature	Add requirement	
Purpose	To allow user to add requirement.	
Actor	Test manager, tester/developer	
Trigger	User enter add requirement screen.	
Precondition	User is logged in. User role is test manager or developer.	
Main Flow	Step	Action
	1	User enter requirement information.
	2	System save the requirement information in database.
	3	System update feed list.
	4	System notifies related users.
Alternative Flow – 1a User did not enter mandatory fields	Step	Action
	1a	User did not enter mandatory fields.
	2	System display error message.

Table 3.2.8: Use case description of the Add requirement use case

Add test case

Use Case ID	UC009	
Feature	Add test case	
Purpose	To allow user to add test case.	
Actor	Test manager, tester/developer	
Trigger	User enter add requirement screen.	
Precondition	User is logged in. User role is test manager or developer.	
Main Flow	Step	Action
	1	User enters test case information.
	2	System saves the test case information in database.
	3	System update feed list.
	4	System notifies related users.
Alternative Flow – 1a User did not enter mandatory fields	Step	Action
	1a	User did not enter mandatory fields.
	2	System display error message.

*Table 3.2.9: Use case description of the Add test case use case*Add bugs

Use Case ID	UC010	
Feature	Add bugs	
Purpose	To allow user to report bugs.	
Actor	Test manager, tester/developer	
Trigger	User enter add bug screen.	
Precondition	User is logged in. User role is test manager or developer.	
Main Flow	Step	Action
	1	User enters bug information.
	2	The system saves the bug information in the database.
	3	System update feed list.
	4	System notifies related users.
Alternative Flow – 1a User did not enter mandatory fields	Step	Action
	1a	User did not enter mandatory fields.
	2	System display error message.

Table 3.2.10: Use case description of the Add bugs use case

View project detail

Use Case ID	UC011	
Feature	View project detail	
Purpose	To allow the user to view project detail.	
Actor	Test manager, tester/developer, business unit	
Trigger	User enter project home screen.	
Precondition	User is logged in.	
Main Flow	Step	Action
	1	System display project information.
	2	System display requirement, test case and bug list.

Table 3.2.11: Use case description of the View project detail use case

3.2.4 Activity Diagrams

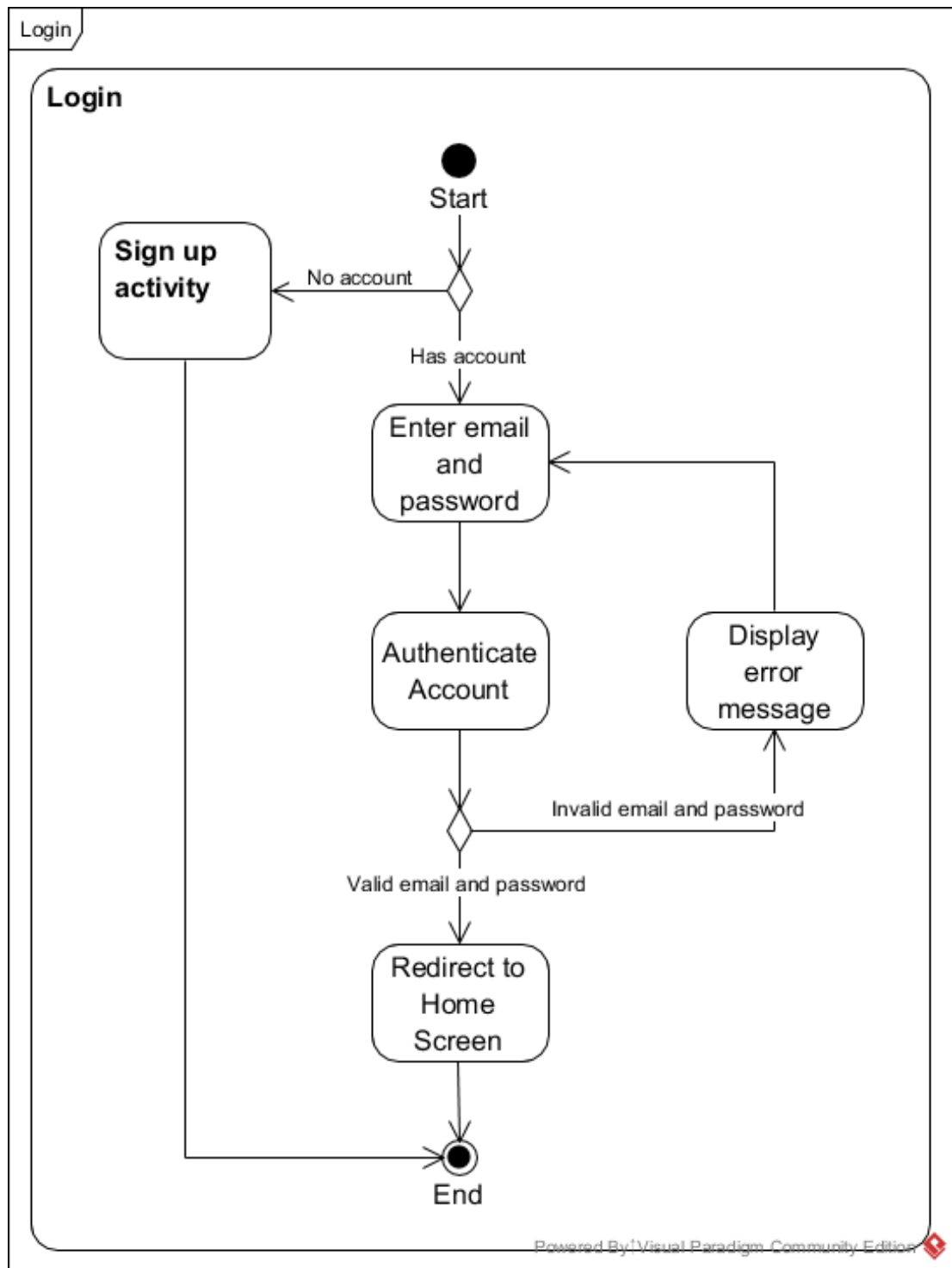


Figure 3.2.3: Activity diagram of the Login use case

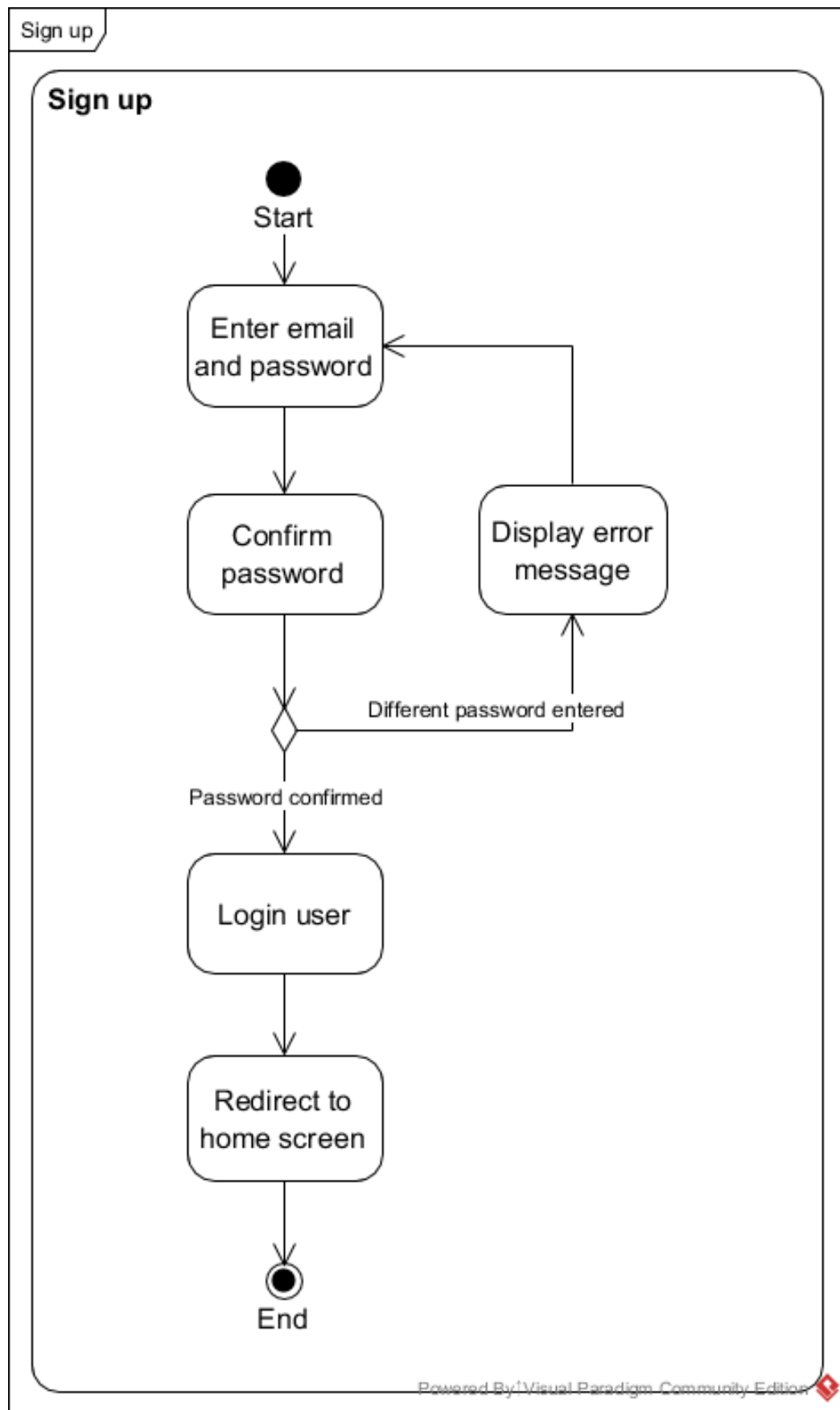


Figure 3.2.4: Activity diagram of the Sign up use case

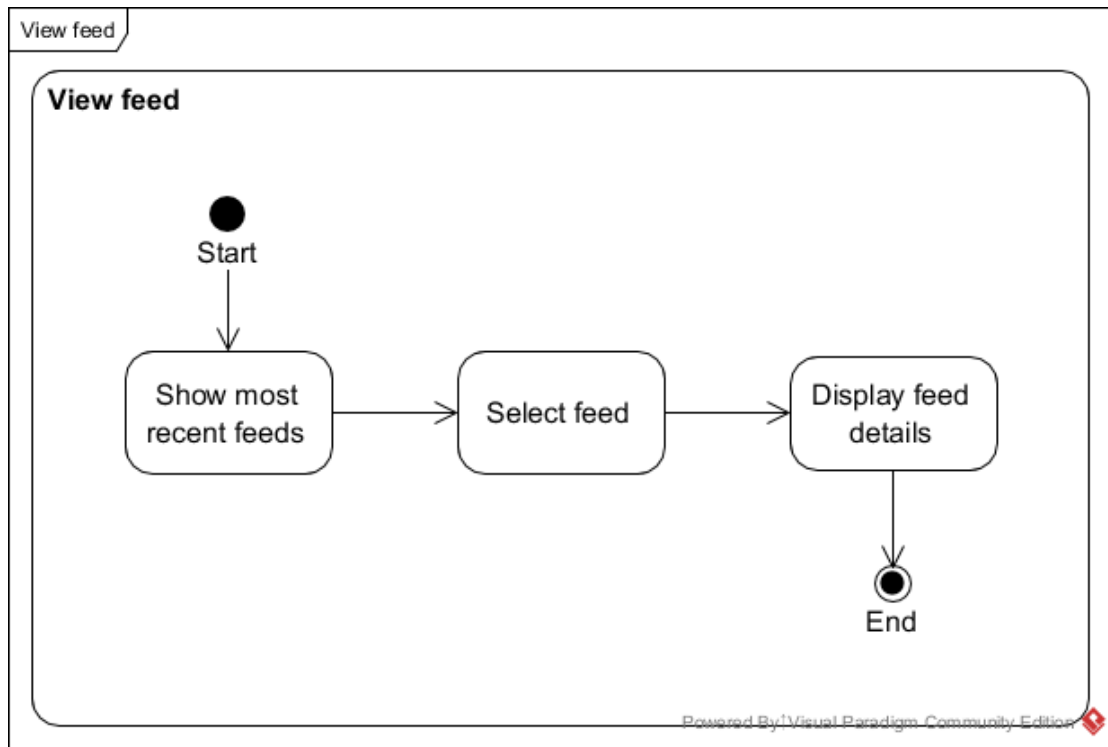


Figure 3.2.5: Activity diagram of the View feed list use case

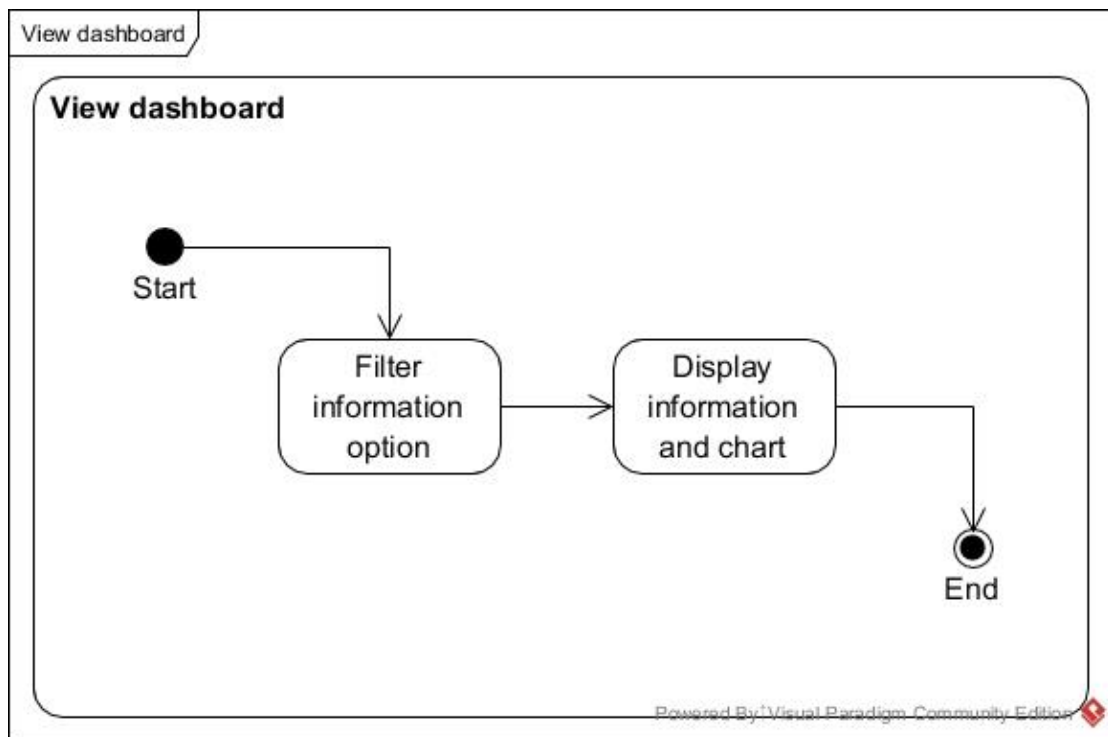


Figure 3.2.6: Activity diagram of the View dashboard use case

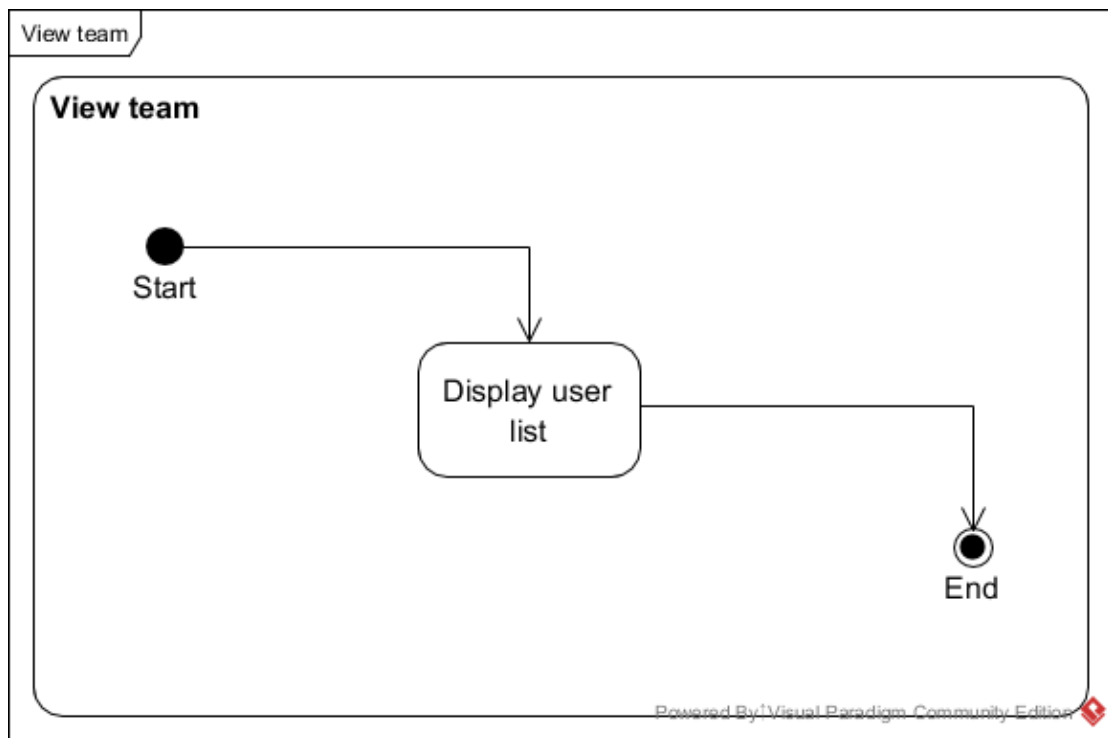


Figure 3.2.7: Activity diagram of the View team use case

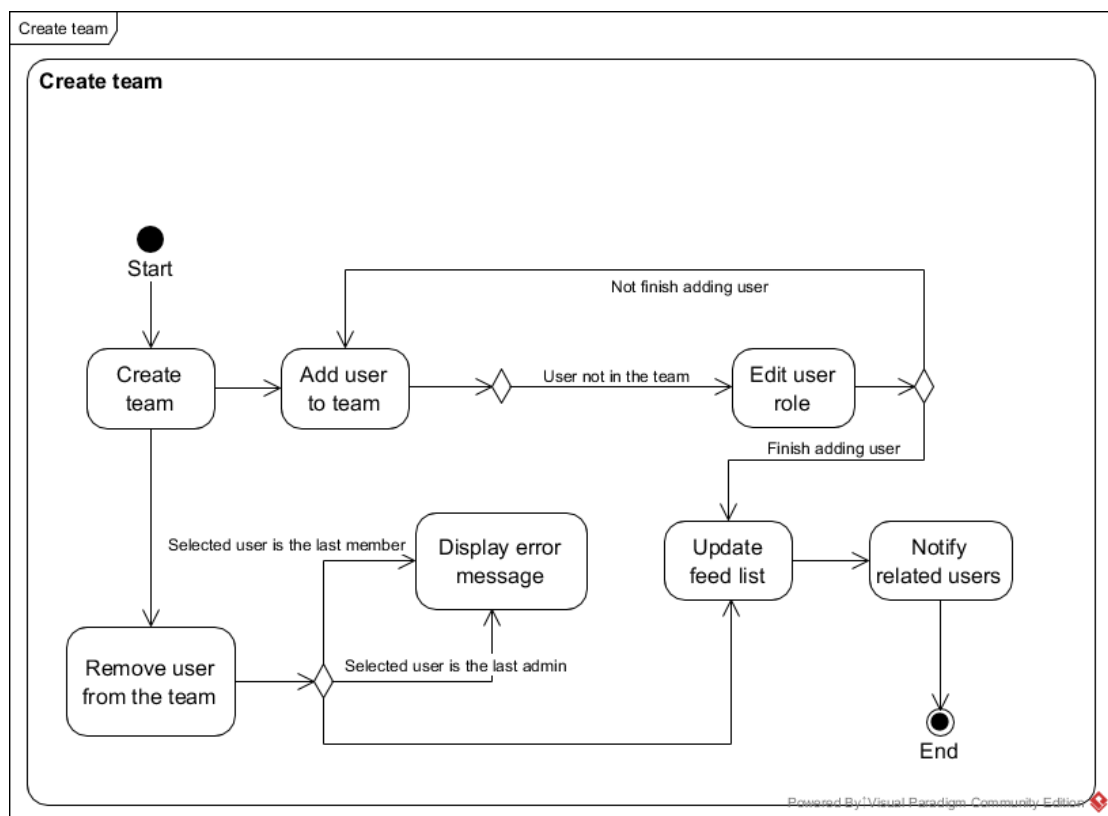


Figure 3.2.8: Activity diagram of the Create team use case

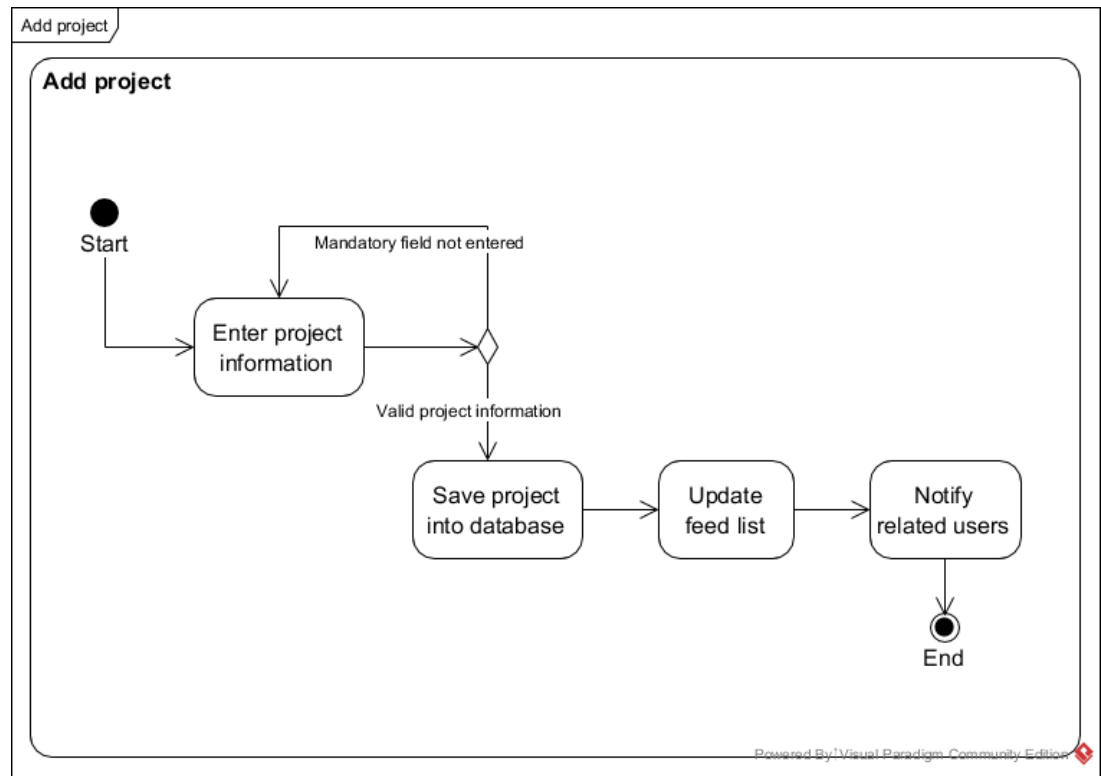


Figure 3.2.9: Activity diagram of the Add project use case

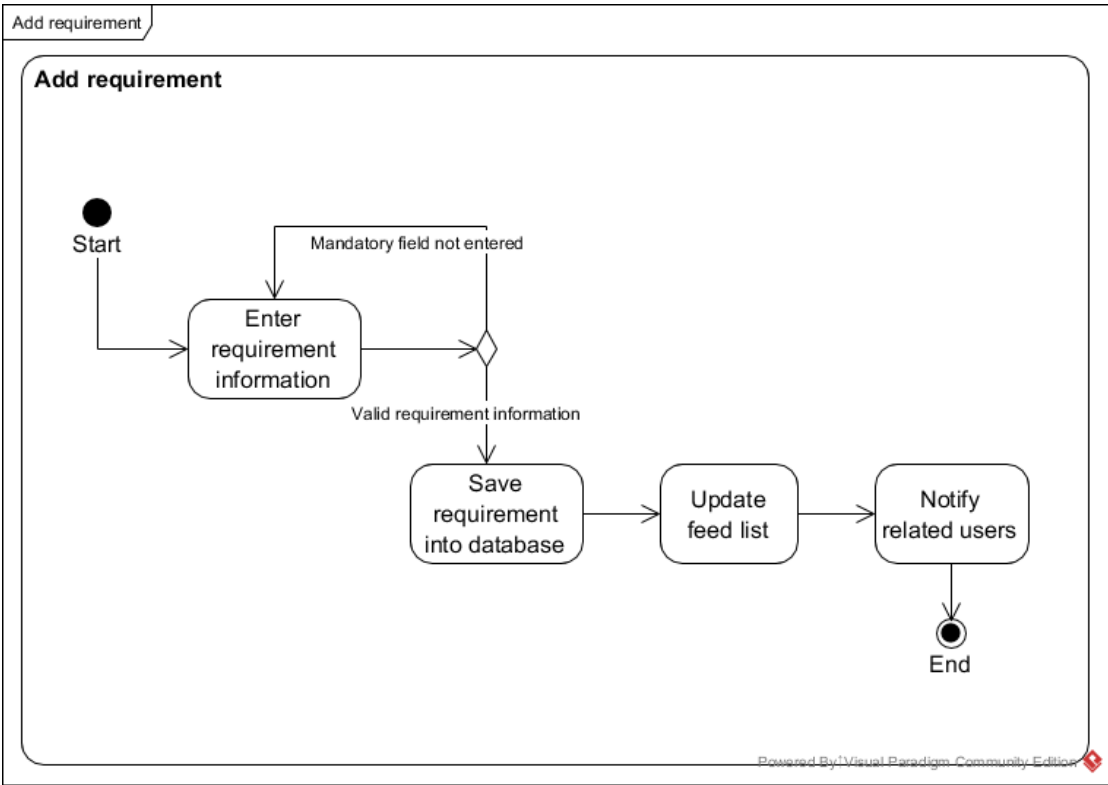


Table 3.2.12: Activity diagram of the Add requirement use case

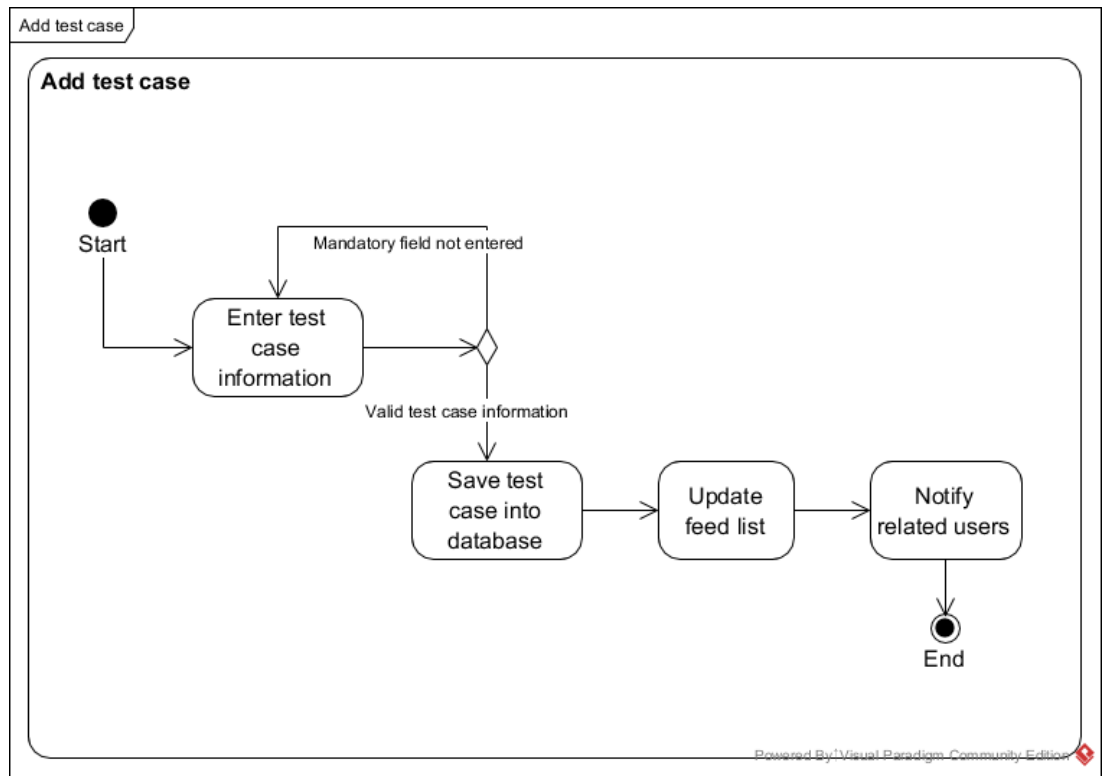


Figure 3.2.10: Activity diagram of the Add test case use case

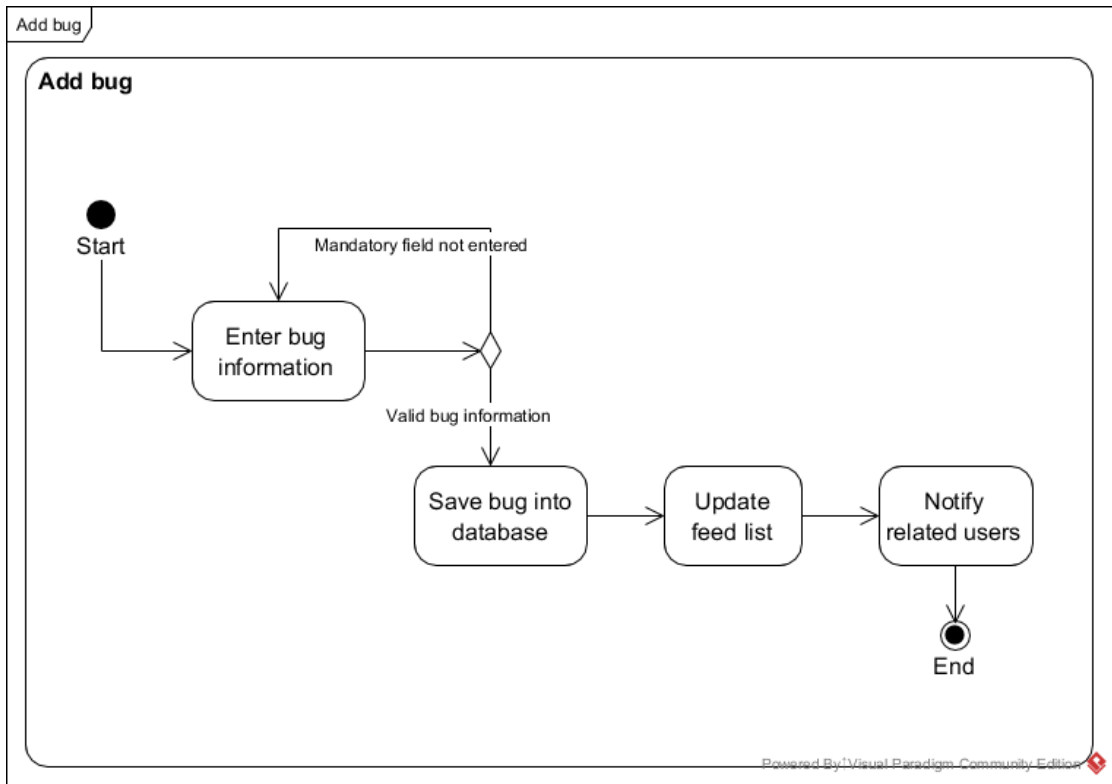


Table 3.2.13: Activity diagram of the Add bug use case

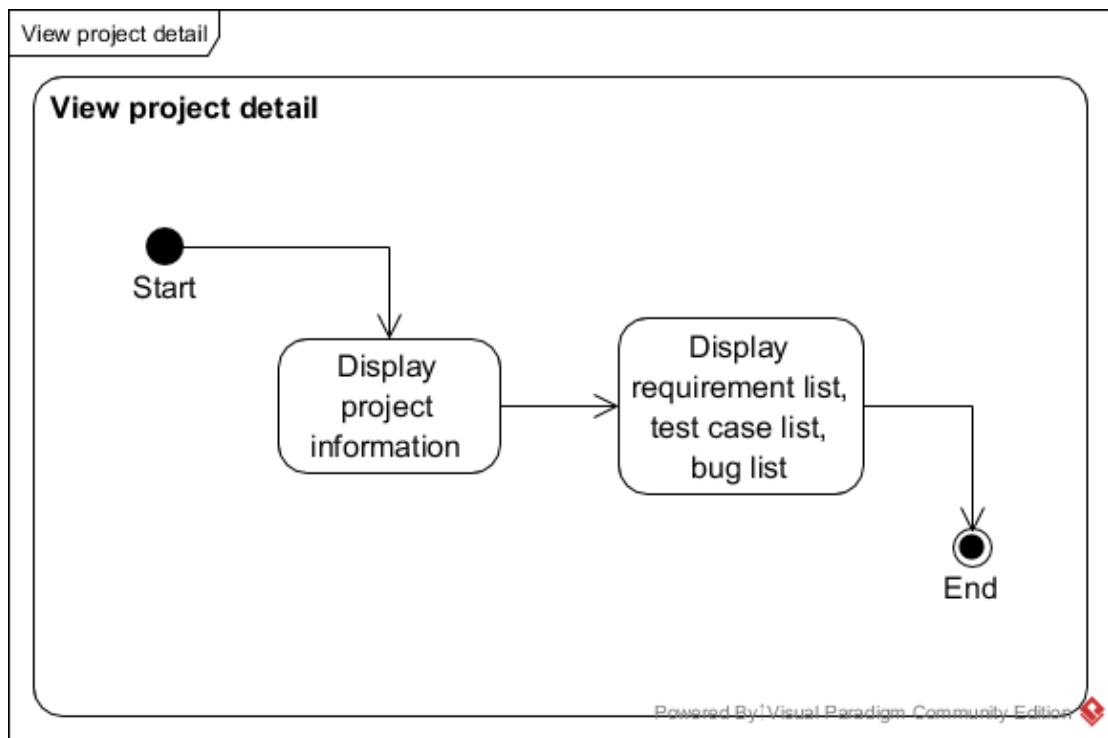


Table 3.2.14: Activity diagram of the View project detail use case

CHAPTER 4: ISO/IEC/IEEE 29119 STANDARD SPECIFICATION

4.1. General

This section shows the definition and description of different metrics and classification as described in the 29119 standard.

4.1.1 Metrics

Metric	Metric Description
Percentage of test cases run.	Metric to monitor the total number of test cases executed.
Percentage of test cases passed over the total of test cases executed.	Metric to monitor the total number of test cases executed and passed.
Percentage of bugs in each status.	Metric to monitor the total number of bugs in each status.
Bug density	Metric to monitor the total number of bugs identified per requirement.

Table 4.1.1: ISO/IEC/IEEE 29119 standard general metrics

The table above shows the metrics used in the *ISO/IEC/IEEE 29119* standard to measure the progress and quality of the system. These metrics help to track down which part of the project needs extra attention and increase the effectiveness of the software development process.

4.1.2 Classification

Impact	Description
Mission Critical	The application will not function or system fails.
Major	Severe problems but possible to workaround.
Minor	Does not impact the functionality or usability of the process is not according to requirements/design specifications.
Priority	Description
Immediate	Must be fixed as soon as possible.
Delayed	The system is unstable but the incident must be fixed prior to the next level of test or shipment.
Deferred	The defect can be left in if necessary due to time or costs

Table 4.1.2: Classification used in ISO/IEC/IEEE 29119 standards

4.2. Form Design Specification

In this section, the forms design specification as stated in the *ISO/IEC/IEEE 29119* standard is shown. To prevent confusion, some of the fields are renamed as some words are interchangeable according to the *ISO/IEC/IEEE 29119* standard. The *ISO/IEC/IEEE 29119* standards discussed the comprehensive software testing standard including requirement management, test management and bug management, therefore some fields do not apply to the proposed system as this project focus on bug management only.

4.2.1 Test Case

Form Fields

Fields	Field Description	Renamed to	Included /Left Out
Test Case ID	Unique identifier for the test case.		Included
Related Feature ID	The ID of the related requirement.	Related Requirement ID	Included
Objective	The objective of the test case.		Included
Covered Test Coverage Items	One or more IDs of Test Coverages that are covered in the test case.		Left Out
Input	Actual input value, type of input or the action that executed in the test case.		Included
Expected Result	The expected outputs and performance of the system.		Included
Special Procedural Requirements	Special procedures that are required to execute the test case.		Included

Intercase Dependency	One or more IDs of other test cases that must be executed prior to this test case.		Included
-------------------------	--	--	----------

*Table 4.2.1: Form Fields Description for Test Case*Form Template with Examples

Test Case ID	TC-04-004		
Related Requirement ID	F004		
Objective	ID Verification (8 digits)		
Input	Expected Result	Special Procedural Requirements	Intercase Dependency
12012378	System display message “No student id found”.	Import of student data	None

Table 4.2.2: Form Template for Test Case

4.2.2 Bug / Test Incident

Form Fields

Fields	Field Description	Renamed to	Included /Left Out
Test Incident Number	Unique identifier for the bug.	Bug ID	Included
Summary	A description of the bug.	Bug Description	Included
Date and Time Incident	The date and time the bug is reported	Reported Time	Included
Context	The project name the bug found in including the release version	Project	Included
Test Procedure	The specification of test cases.		Left Out
Procedure to reproduce the incident	The sequence of actions which cause the bug.		Included

Test Environment	The setup of hardware and software when executing the test case or when the bug was observed.		Included
Attempt to repeat	Times the test case tested to produce the same result.		Included
Tester's Name	The name of the tester who tested the test case or bug.	Tester	Included
Observer's Name	The name of the tester who observed the bug.	Reported by	Included
Status of Incident	The current status of the bug.	Status	Included
Impact	The level of impact of the bug on the system.		Included
Priority	The level of urgency in resolving the bug.		Included
Approvals	The name of the person that confirmed the bug is fixed and resolved.	Approved by	Included

Table 4.2.3: Form Fields Description for bug

Form Template with Examples

Bug ID:	TIR-1.1
Bug Description:	
The user had inserted correct input student ID to search for the student information before he/she wants to delete the record. However, the student information does not show up although is a correct ID.	
Reported Time:	19/04/2017
Project:	Library Management System (LMS_1.1)
Procedure to reproduce the incident:	1. Key in a valid ID and it exists in the text file. 2. student info didn't show up due to the wrong format in the file
Test Environment:	Aspire F 15 Acer Intel Core i7-7500 U 2.7GHz
Attempt to repeat	The procedure is done 3 times continuously after incident 1st discovered. The same result emerged
Tester	Tester 4
Reported by:	Tester 3
Status	
<input checked="" type="checkbox"/> Open <input type="checkbox"/> Assigned for Resolution <input type="checkbox"/> Retested with the fix confirmed <input type="checkbox"/> Approved for Resolution <input type="checkbox"/> Fixed	
Impact	
<input checked="" type="checkbox"/> Mission Critical: Application will not function or system fails <input type="checkbox"/> Major: Severe problems but possible to workaround <input type="checkbox"/> Minor: Does not impact the functionality or usability of the process is not according to requirements/design specifications.	
Priority	
<input checked="" type="checkbox"/> Immediate: Must be fixed as soon as possible <input type="checkbox"/> Delayed: System is unstable but the incident must be fixed before the next level of test or shipment <input type="checkbox"/> Deferred: Defect can be left in if necessary due to time or costs	
Approved By	Tester 2

Table 4.2.4: Form Template for Bugs

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1. Database Set up

The Firebase Realtime Database is used in this project as the centralized database. Firebase Realtime Database is a NoSQL cloud database developed by Google, the backend is managed by Google and developer interact with the database using API without worrying the backend implementation. Data is stored as JSON format in Firebase which is better in representing an object compare to the traditional relational database.

In this project, the database is organized so that data is grouped by team or project so that it is easier to filter and fetch the specific data.

Structure of profiles

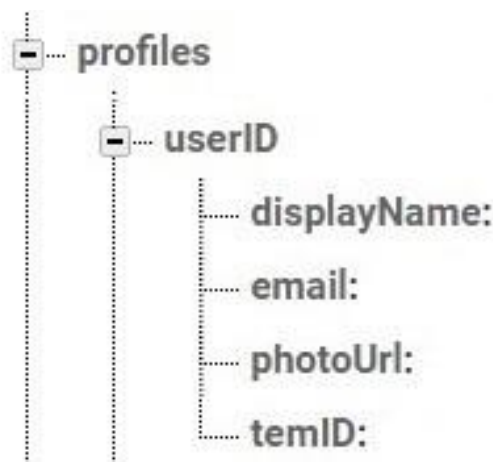


Figure 5.1.1: Database structure of profiles

Structure for teams

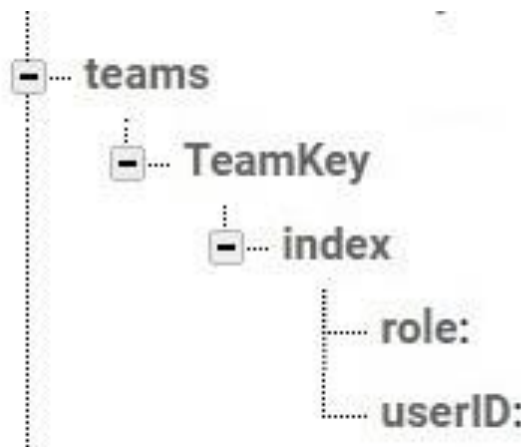


Figure 5.1.2: Database structure of teams

Structure for projects

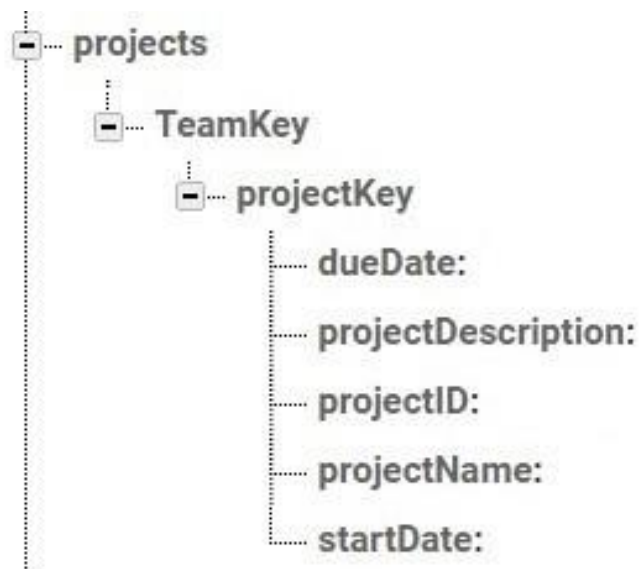


Figure 5.1.3: Database structure of projects

Structure for feeds

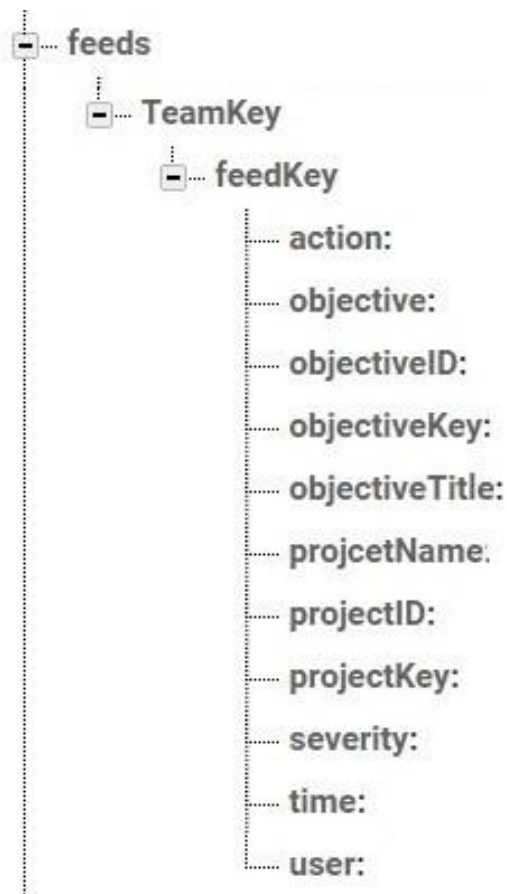


Figure 5.1.4: Database structure for feeds

Structure for requirements

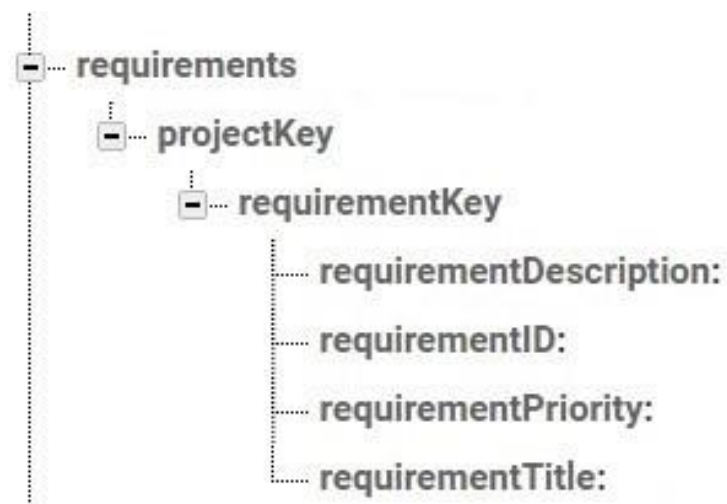


Figure 5.1.5: Database structure of requirements

Structure for test cases

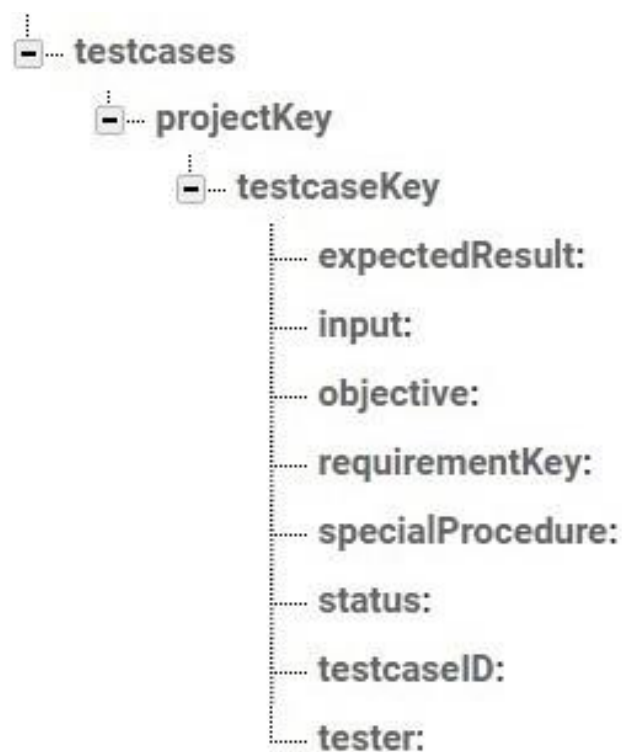
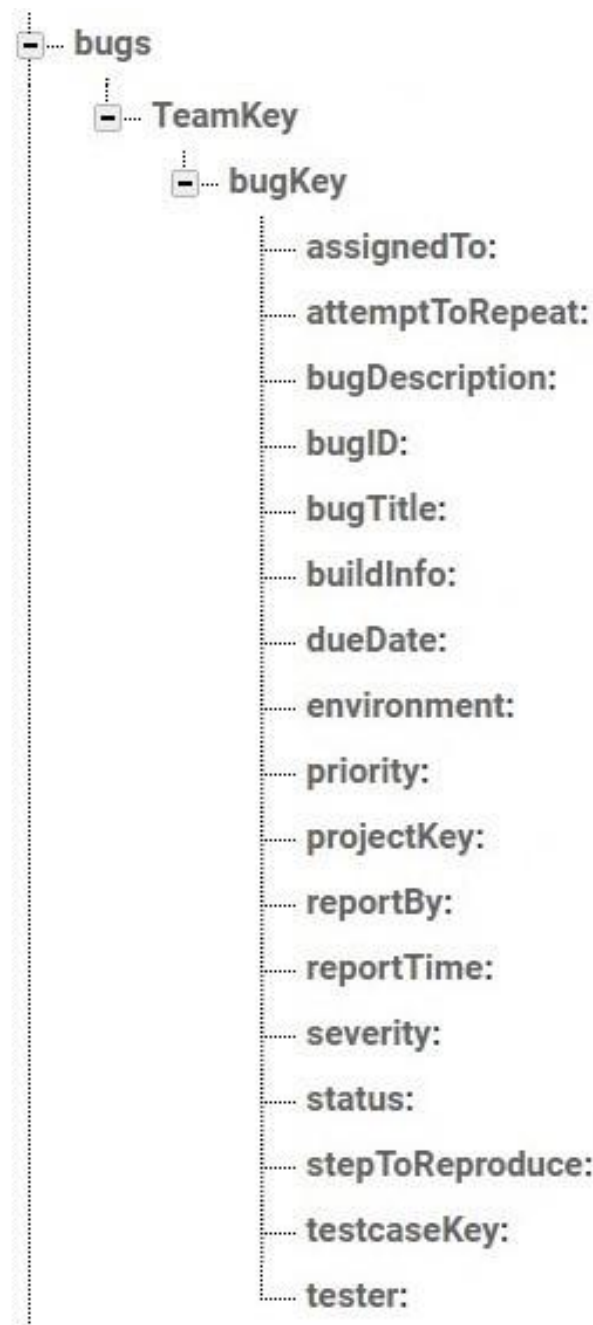


Figure 5.1.6: Database structure for test cases

Structure for bugs*Figure 5.1.7: Database structure for bugs***5.2. Authentication**

The authentication service used in this project is the Firebase Authentication service by Google. All the backends are managed by Google and developers only need to communicate with the server through SDKs and libraries. Firebase Authentication provides different types of authentication methods including passwords, phone numbers, link with Google, Facebook and Twitter, etc.

Firebase Authentication service is secure and time saving by providing useful features to developers. The guide for setting up of the authentication service can be found in the Firebase documentation <<https://firebase.google.com/docs/auth>>

5.3. React Native Components

React Native is used as the development framework for this project. One of the advantages of using React Native is that React Native has many official libraries that can help developers to reduce wasting time on repetitive works that have done by others. React Native also has a very large and healthy community which shares the components that they built. In this project, there are some libraries and components used to speed up the development process.

5.3.1 Community built libraries and components

Following are the library and component used, installation and usage guide can be found in their repositories.

Library/Component	Repository
React Native Chart Kit	https://github.com/indiespirit/react-native-chart-kit
React Native Dropdown Menu	https://github.com/WheelerLee/react-native-dropdown-menu#readme
React Native Elements	https://react-native-elements.github.io/react-native-elements/
React Native Paper	https://callstack.github.io/react-native-paper/
React Native Simple Toast	https://github.com/vonovak/react-native-simple-toast#readme
React Navigation V4.0	https://reactnavigation.org/
React Redux	https://react-redux.js.org/
React Router Dom	https://reacttraining.com/react-router/
Redux	https://redux.js.org/
Redux Persist	https://github.com/rt2zz/redux-persist
Redux Thunk	https://github.com/reduxjs/redux-thunk
RN Multi Progress Bar	https://github.com/abaktiar/rn-multi-progress-bar
Victory Native	https://formidable.com/open-source/victory/docs

Table 5.3.1: Libraries and components used and the repositories

5.3.2 Self-built components

By the time this project is in the development stage, React Native for web is in beta version, so many features only support mobile platform and not web platform same as community-built components and libraries. Because of that, some components are built to use in the web platform.

Password Field



```

PasswordField.js - Notepad
File Edit Format View Help
import React, { useState, useEffect } from "react";
import { View, TextInput, StyleSheet, TouchableOpacity } from "react-native";
import { MaterialCommunityIcons } from "@expo/vector-icons";

import Colors from "../../common/constants/Colors";

const PasswordField = props => {
  const [isHidden, setIsHidden] = useState(true);

  return (
    <View style={styles.inputContainer}>
      <TextInput
        {...props}
        style={styles.textInput}
        placeholder={props.placeholder}
        secureTextEntry={isHidden}
        autoComplete={false}
      />
      <TouchableOpacity onPress={() => setIsHidden(prevState => !prevState)}>
        <MaterialCommunityIcons
          name={isHidden ? "eye" : "eye-off"}
          color={Colors.light}
          size={30}
          style={styles.inputLabel}
        />
      </TouchableOpacity>
    </View>
  );
};

const styles = StyleSheet.create({
  inputContainer: {
    flex: 1,
    flexDirection: "row"
  },
  inputLabel: {
    marginLeft: 10
  },
  textInput: {
    flex: 1,
    fontFamily: "roboto-regular",
    color: Colors.light
  }
});

export default PasswordField;

```

Figure 5.3.1: Screenshot of PasswordField.js

The React Native password field does not have an option for the user to show the entered password. To increase the usability of the system, a custom password field

component is built which allow users to show the password by pressing the eye icon located beside the text input. This can easily be achieved by using the React Native state.

Progress Bar



```

ProgressBar.js - Notepad
File Edit Format View Help
import React from "react";
import { View, Text, StyleSheet } from "react-native";

const ProgressBar = (props) => {
  return (
    <View style={styles.progressBar}>
      {props.data.map((item, index, array) => {
        return (
          <View
            style={[
              styles.progress,
              {
                backgroundColor: item.color,
                flex: item.progress,
                borderTopLeftRadius: index === 0 ? 5 : 0,
                borderBottomLeftRadius: index === 0 ? 5 : 0,
                borderTopRightRadius: index === array.length - 1 ? 5 : 0,
                borderBottomRightRadius: index === array.length - 1 ? 5 : 0,
              },
            ]}
            key={index}
          ></View>
        );
      })}
    </View>
  );
};

const styles = StyleSheet.create({
  progressBar: {
    flexDirection: "row",
    flex: 1,
    justifyContent: "center",
    alignItems: "center",
    width: "100%",
    height: 10,
  },
  progress: {
    height: 10,
  },
});

export default ProgressBar;

```

Figure 5.3.2: Screenshot of ProgressBar.js

For the progress bar, the community-built RN Multi Progress Bar is used for the mobile platform. However, it does not support web platform so a progress bar for the web platform is created. It replicates the look of the RN Multi Progress Bar which allow multiple progress show in the same bar.

Alert


```
Alert.js - Notepad
File Edit Format View Help
import React from "react";
import {
  View,
  Text,
  StyleSheet,
  Button,
  TouchableWithoutFeedback,
} from "react-native";

import Card from "../Card";

import Colors from "../../common/constants/Colors";

const Alert = (props) => {
  const { buttonArray } = props;

  return (
    <View style={styles.modal}>
      <View style={styles.modalContent}>
        <Text style={styles.title}>{props.title}</Text>
        <Text style={styles.content}>{props.content}</Text>
        <View style={styles.buttonContainer}>
          {buttonArray.map((item, index) => {
            return (
              <View style={styles.btn} key={item.text}>
                <Button
                  title={item.text}
                  color={item.color ? item.color : Colors.primaryColor}
                  onPress={() => {
                    props.show(false);
                    if (item.onPress) {
                      item.onPress();
                    }
                  }}
                />
              </View>
            );
          })}
        </View>
      </View>
    </View>
  );
};

const styles = StyleSheet.create({
  modal: {
    position: "absolute",
    zIndex: 100,
    width: "100vw",
    height: "100vh",
    backgroundColor: "rgba(0,0,0,0.4)",
    alignItems: "center",
    padding: 20,
  },
});
```

Ln 1, Col 1 100%

Figure 5.3.3: Screenshot of Alert.js



```

    </View>
  );
  })}
</View>
</View>
</View>
);
};

const styles = StyleSheet.create({
  modal: {
    position: "absolute",
    zIndex: 100,
    width: "100vw",
    height: "100vh",
    backgroundColor: "rgba(0,0,0,0.4)",
    alignItems: "center",
    paddingTop: 200,
  },
  modalContent: {
    width: 350,
    minHeight: 150,
    backgroundColor: "#fff",
    padding: 20,
    elevation: 8,
    shadowColor: "black",
    shadowOpacity: 0.26,
    shadowOffset: { width: 2, height: 2 },
    shadowRadius: 8,
    zIndex: 101,
  },
  title: {
    fontSize: 20,
  },
  content: {
    flex: 1,
    fontSize: 16,
    marginTop: 10,
    marginBottom: 20,
  },
  buttonContainer: {
    flexDirection: "row",
    alignItems: "flex-end",
    justifyContent: "flex-end",
  },
  btn: {
    marginHorizontal: 5,
  },
});

export default Alert;

```

Figure 5.3.4: Screenshot of style for Alert.js

React Native has an Alert component for the mobile platform but it does not support the web platform. Usually, developers use the native javascript alert function but it is too general and cannot be customized for different usage and function. For this reason, a custom Alert is created which replicates the native Alert component for consistency across mobile and web platforms.

5.4. User Interface Design

Both mobile and web platform of this project have different user interface design to increase the usability of the system across different platform. For the mobile platform, as the screen is smaller, each screen focus on less information and the navigation of the App is more native mobile design. For the web platform, as the screen is bigger, more information is shown in each screen and more functions can be achieved within one page so it is more effective in using the system.

5.4.1 Mobile Platform

Splash Screen



Figure 5.4.1: The splash screen

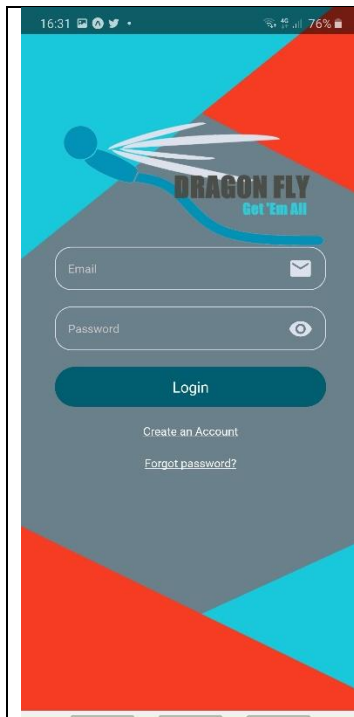
Authentication Screen

Figure 5.4.2: The Login screen

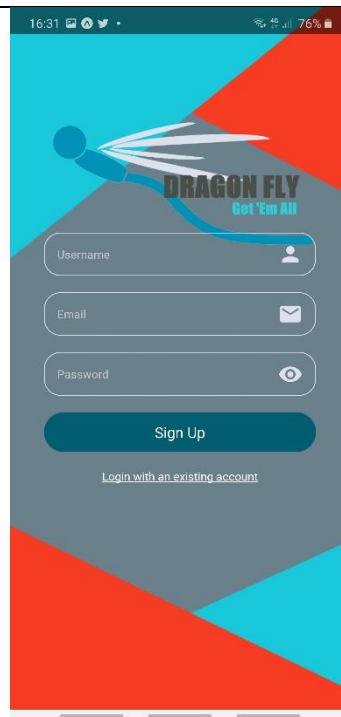


Figure 5.4.3: The Signup screen

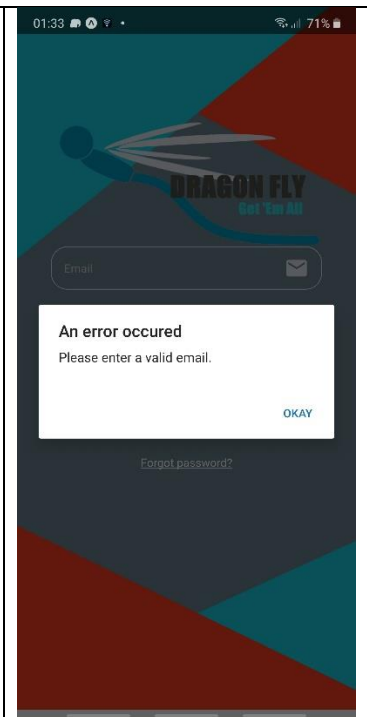


Figure 5.4.4: System alert when fields are invalid

The Splash screen is the first screen users will see after launching the App, it serves as the loading page when the App prepares for the start-up. Then is the authentication screens, users can log in or create an account if they are not signed up before. An alert message will show if fields entered are invalid. There is also a “Forgot password?” option for users to reset the password if they forgot, an email will send to the user for reset password.

Feeds Screen

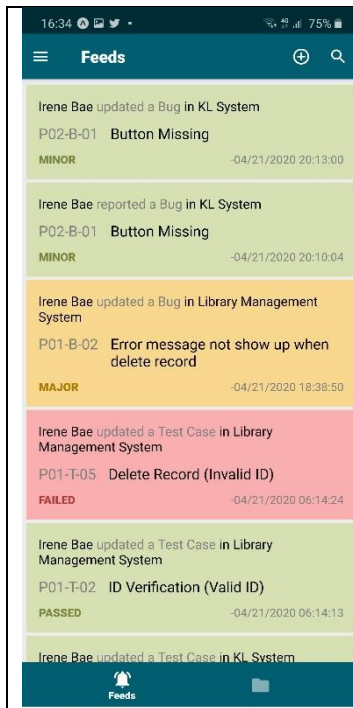


Figure 5.4.5: The Feeds screen

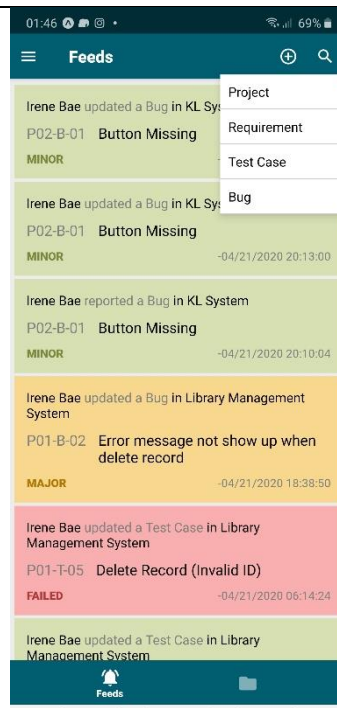


Figure 5.4.6: Add function in the Feeds screen

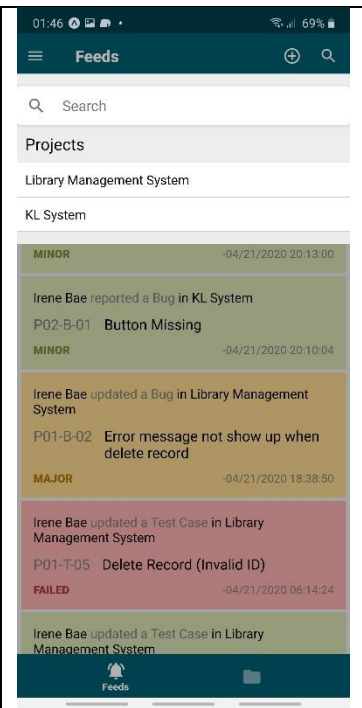
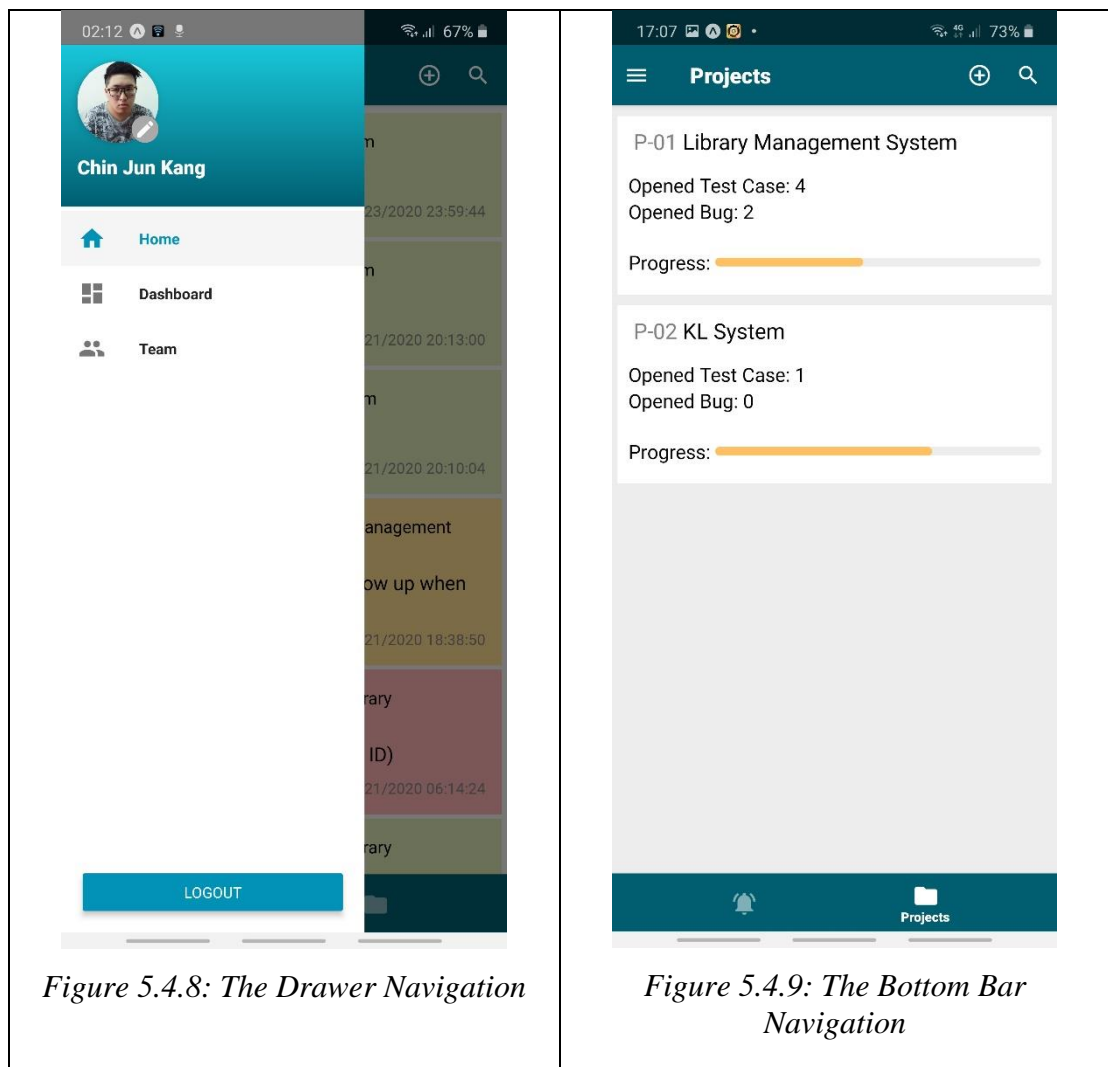


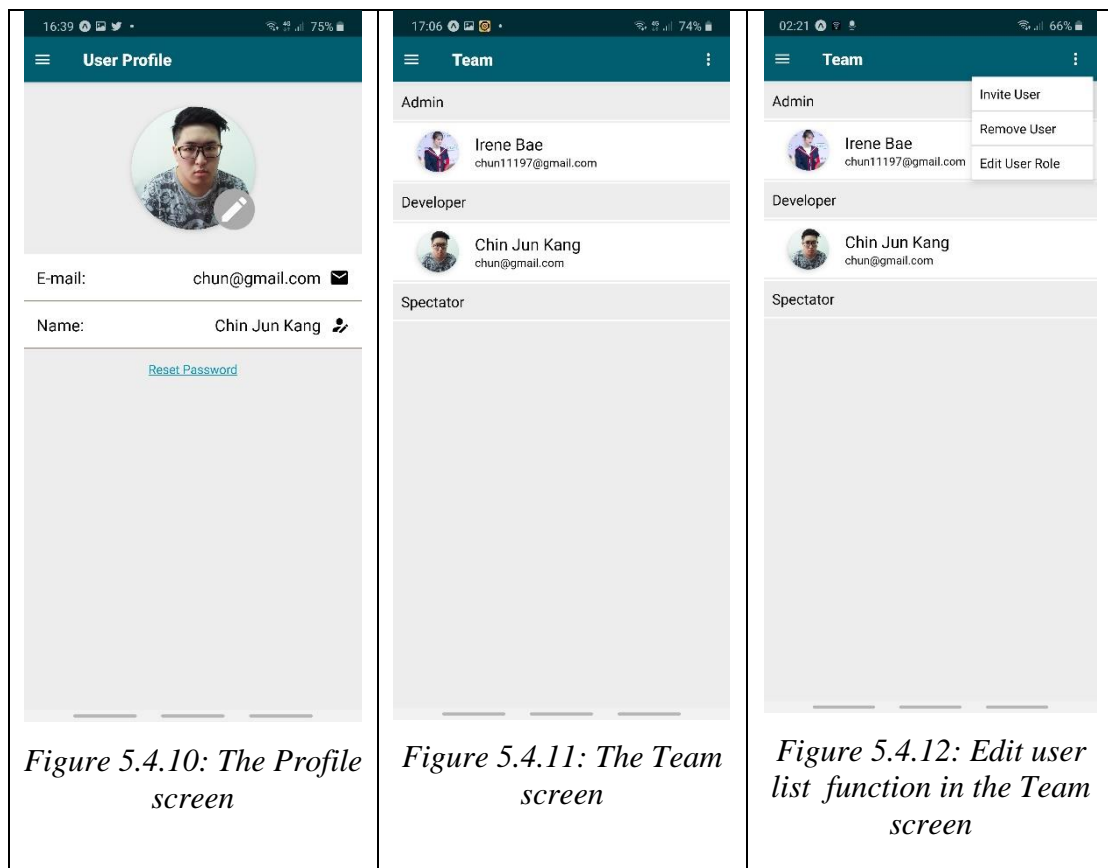
Figure 5.4.7: Search function in the Feeds screen

After logging in, the first screen the user will see is the Feeds screen which shows the most recent feeds or update of the team including new and updates action. When the user press on the feeds, the App will navigate to the according screen shows the details of the feed. There are 2 functions on the top right corner, add and search functions. Add function allows users to add project, requirement, test case or bug while the search function allows users to filter the feeds list with project or query.

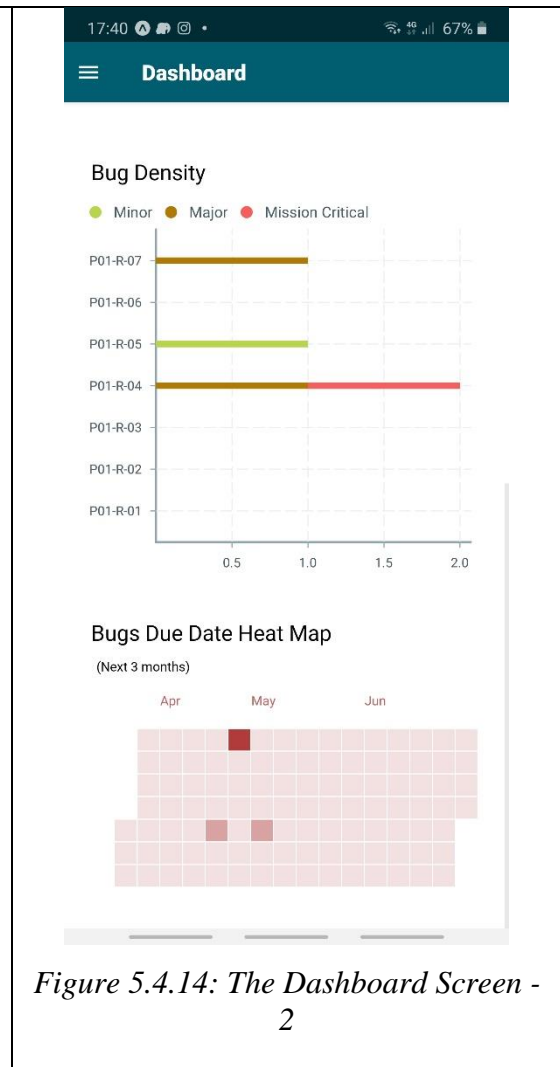
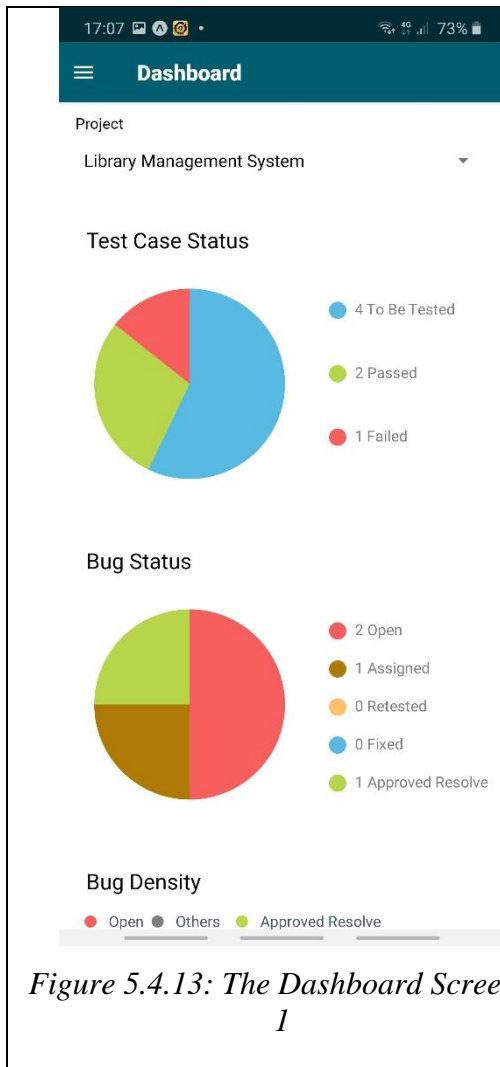
Navigation



There is 2 main navigation in the App, Drawer Navigation and Bottom Bar Navigation. The Drawer Navigation can be accessed by pressing the menu button in the top left corner and navigates through Home Screen, Profile Screen, Dashboard Screen, Team Screen and Logout function. The Bottom Bar Navigation is located in the bottom and navigates through Feeds Screen and Project List Screen.

Profile Screen and Team Screen

In the Profile Screen, users can edit the profile picture, display name and reset the password. In the Team Screen admin can edit the user list by pressing the top right corner button. The edit function includes invite user, remove user and edit user role. The system will display an alert message for any invalid actions.

Dashboard Screen

In the Dashboard Screen, users can choose the project and the relevant charts will show. The charts in the Dashboard Screen include Test Case Status, Bug Status, Bug Density, and Bugs Due Date Heat Map.

Add Project Screen and Add Requirement Screen

Figure 5.4.15: The Add Project Screen

Figure 5.4.16: The Requirement Screen

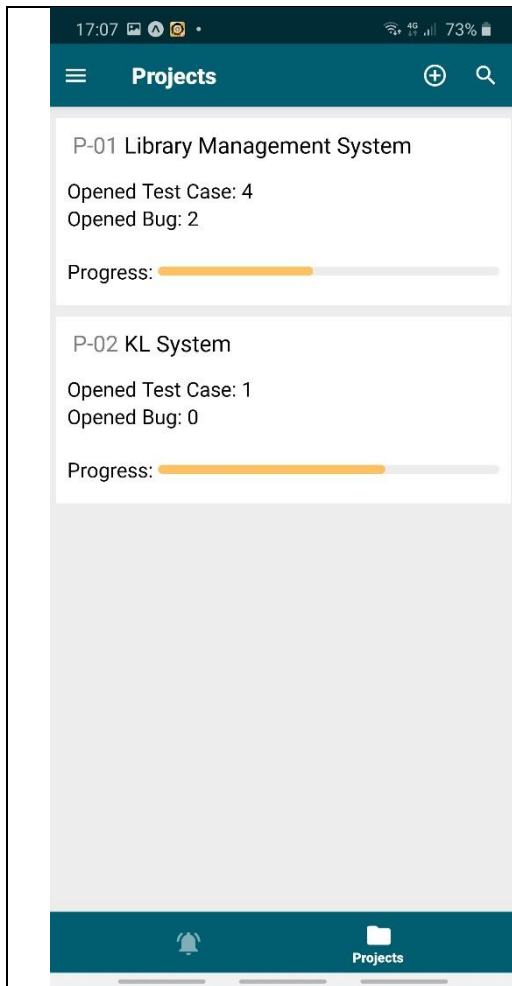
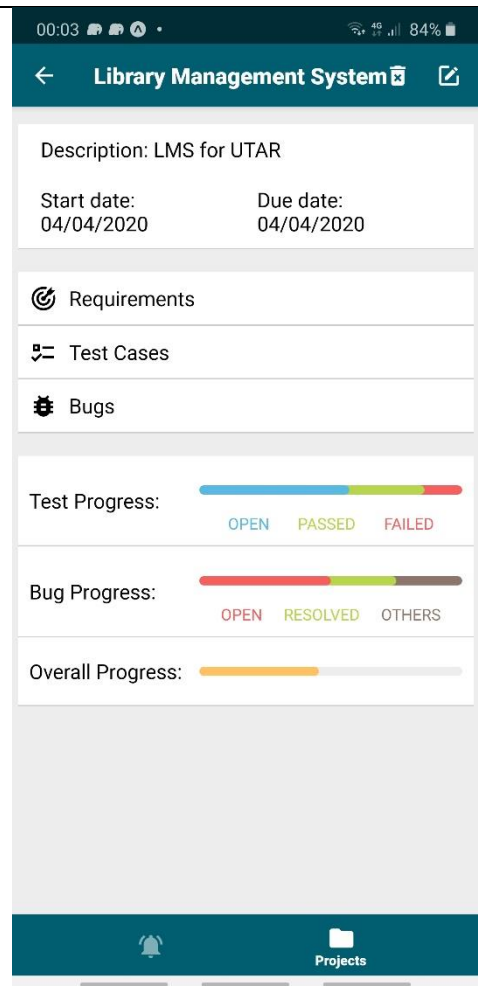
Add Test Case Screen

Figure 5.4.17: The Add Test Case Screen - 1

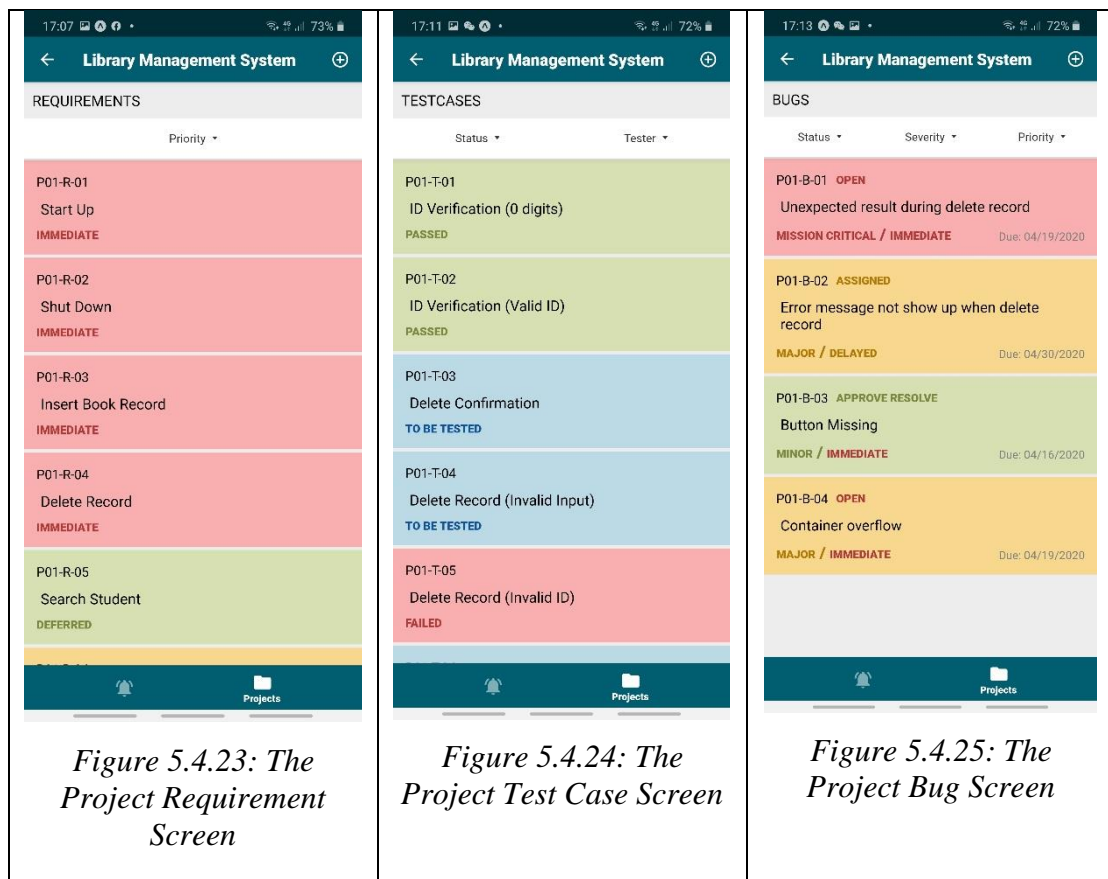
Figure 5.4.18: The Test Case Screen - 2

Add Bug Screen
Figure 5.4.19: The Add Project Screen
Figure 5.4.20: The Requirement Screen

For all form screens, an alert message will be displayed if there are mandatory fields not entered or there are invalid inputs. For the Add Bug Screen, users can attach images by pressing the paperclip button located in the top right corner. The image will then upload to the Firebase Cloud Storage upon the bug is saving.

Project List Screen and Project Home Screen*Figure 5.4.21: The Project List Screen**Figure 5.4.22: The Project Home Screen*

The Project List Screen shows the list of the project and the overall progress of each project including the number of opened test cases and the number of opened bugs. The Project Home Screen shows the information about the project and more detail progress. There are 3 options in the Project Home Screen, Requirements, Test Cases and Bugs which navigate to the list accordingly.

Project Requirement Screen & Project Test Case Screen & Project Bug Screen

The Project Requirement Screen shows the list of the requirements, the Project Test Case Screen shows the list of the test cases and the Project Bug Screen shows the list of the bugs of the selected project. Each of the list screens has options for the user to filter the list and add a new entry.

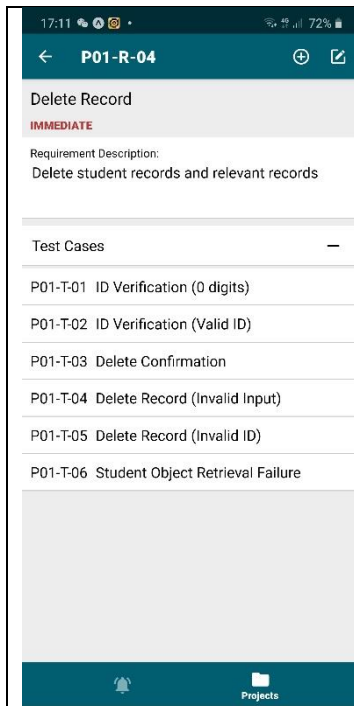
Requirement Screen & Test Case Screen & Bug Screen

Figure 5.4.26: The Requirement Screen

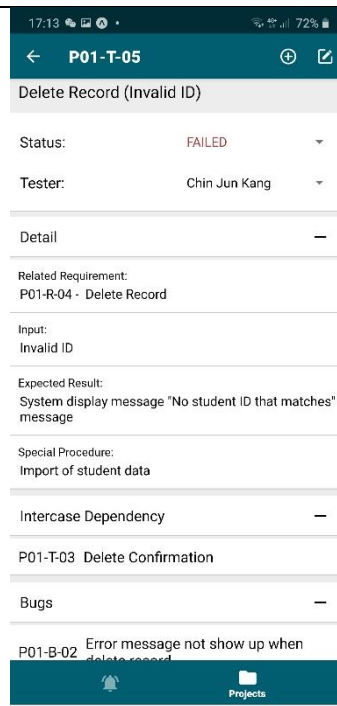


Figure 5.4.27: The Test Case Screen

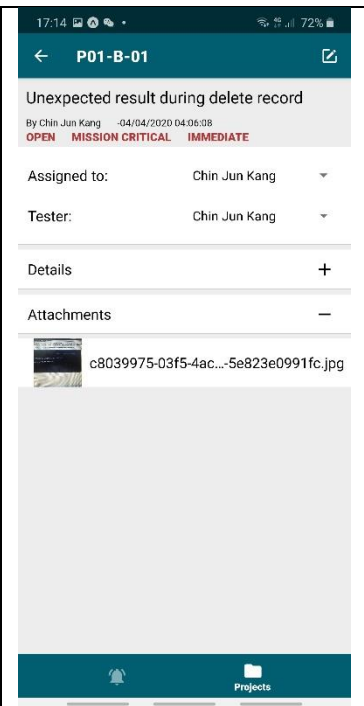


Figure 5.4.28: The Bug Screen

Each of the detail screens shows the detail of the entry and the related entries for traceability. For example, Requirement Screen has a list of the related test cases, Test Case Screen has a list of the related bugs. All of the details screen have the information group by section and each of the section can be hide so the screen is no heavily crowded and reduces the readability.

5.4.2 Web Platform

Authentication Screen

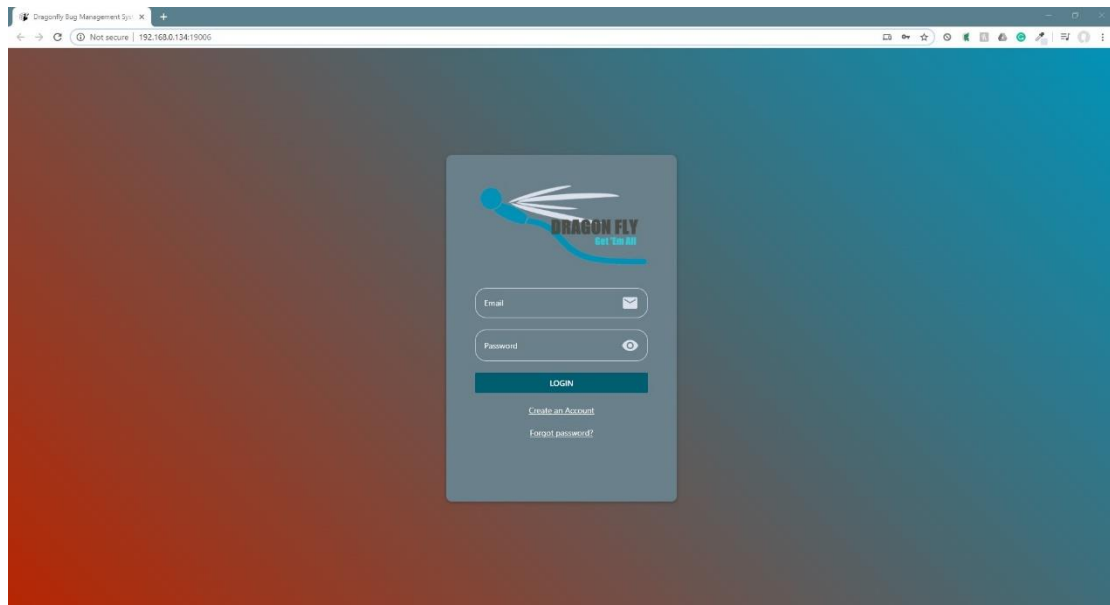


Figure 5.4.29: Screenshot of Login Page

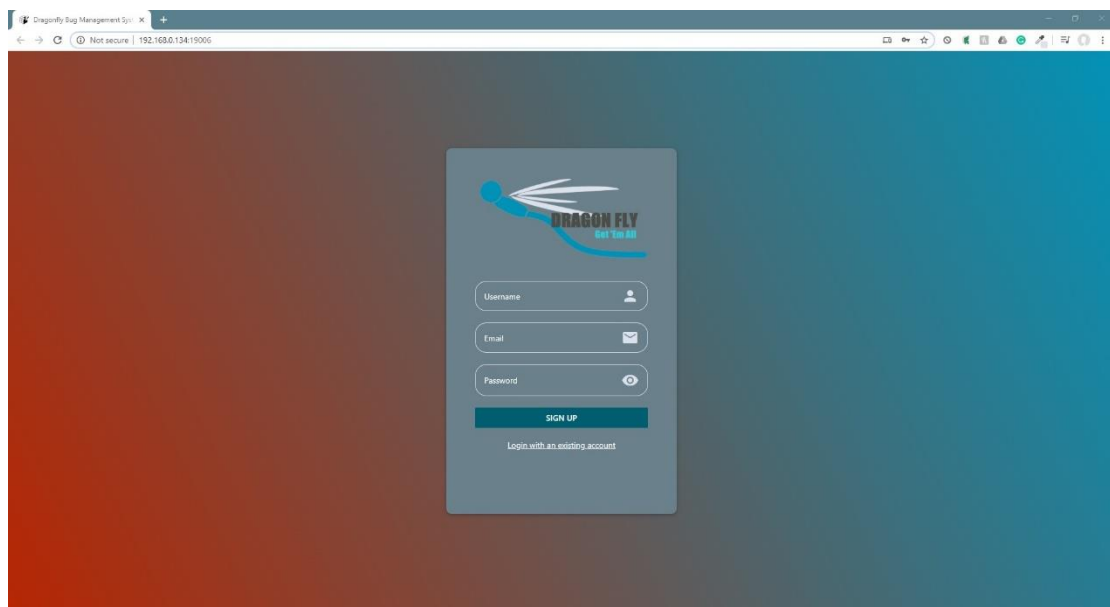


Figure 5.4.30: Screenshot of Signup Page

The authentication page of the web platform has the same design and functions as the mobile platform.

Home Page

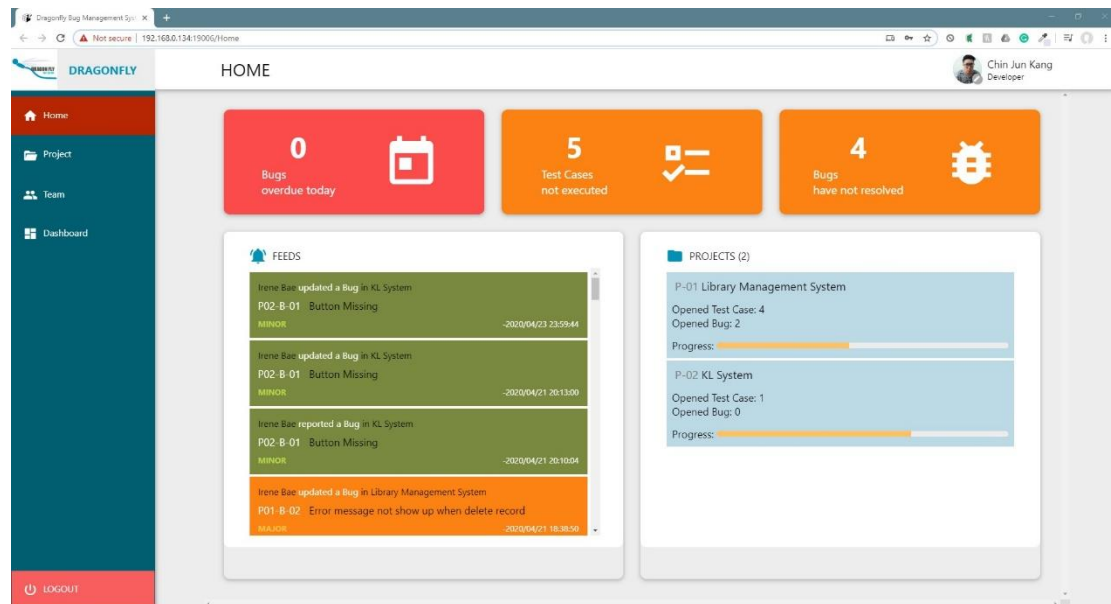


Figure 5.4.31: The Home Page

The Home Page combined the Feeds Screen and Project List Screen of the mobile platform and provides extra information on the total number of bug due today, the number of test case not executed and the number of bugs not resolved.

Profile Page

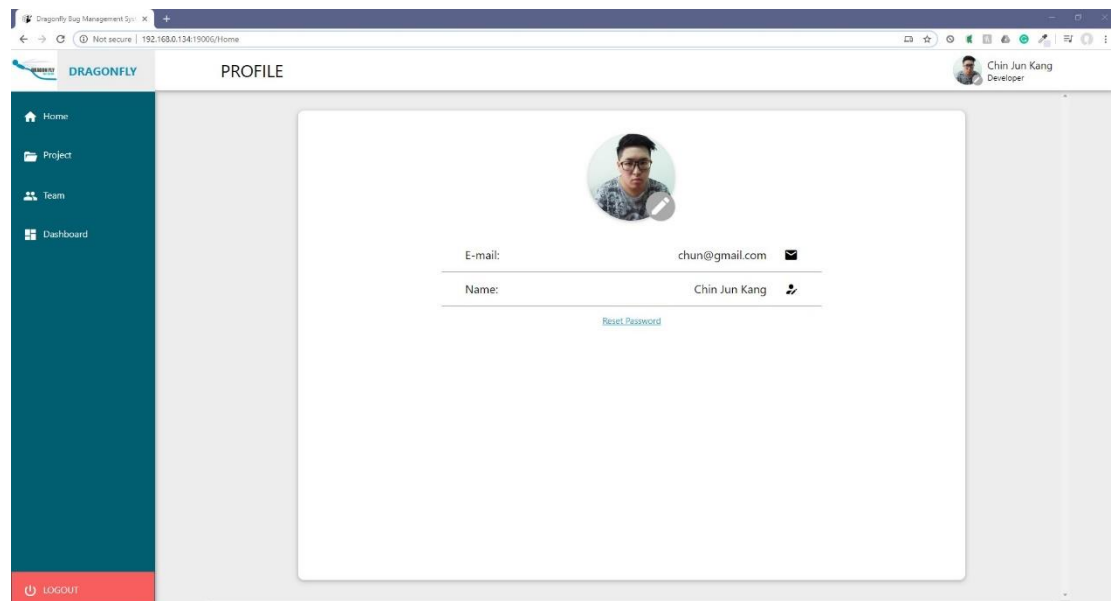


Figure 5.4.32: The Profile Page

Project Home Page

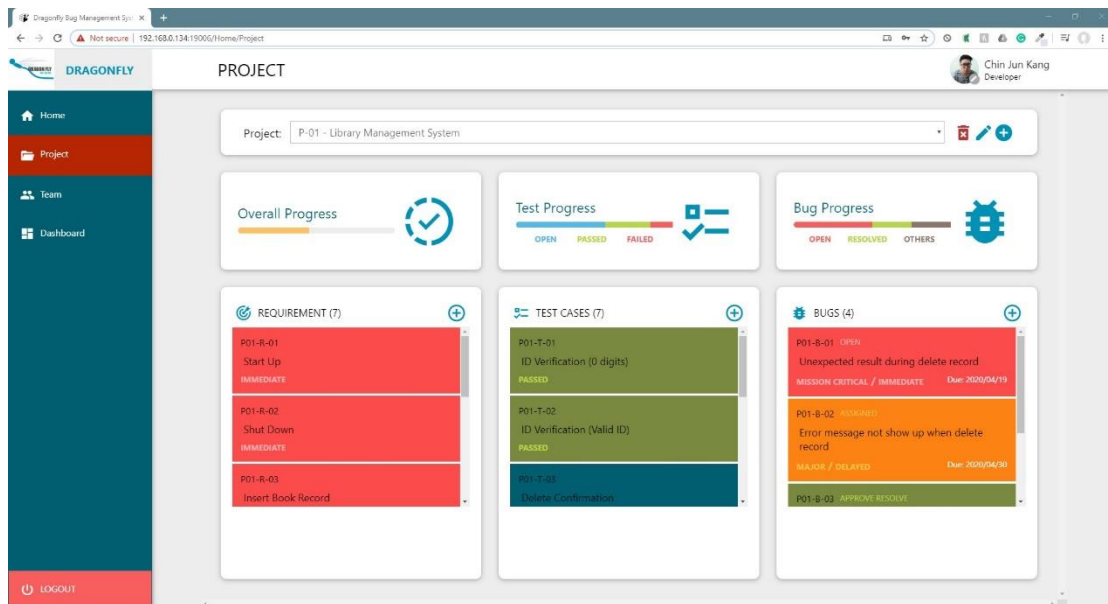


Figure 5.4.33: Screenshot of Project Home Page

The Project Home Page combined the Project Home Screen, Project Requirement Screen, Project Test Case Screen and Project Bug Screen of the mobile Platform.

Team Page

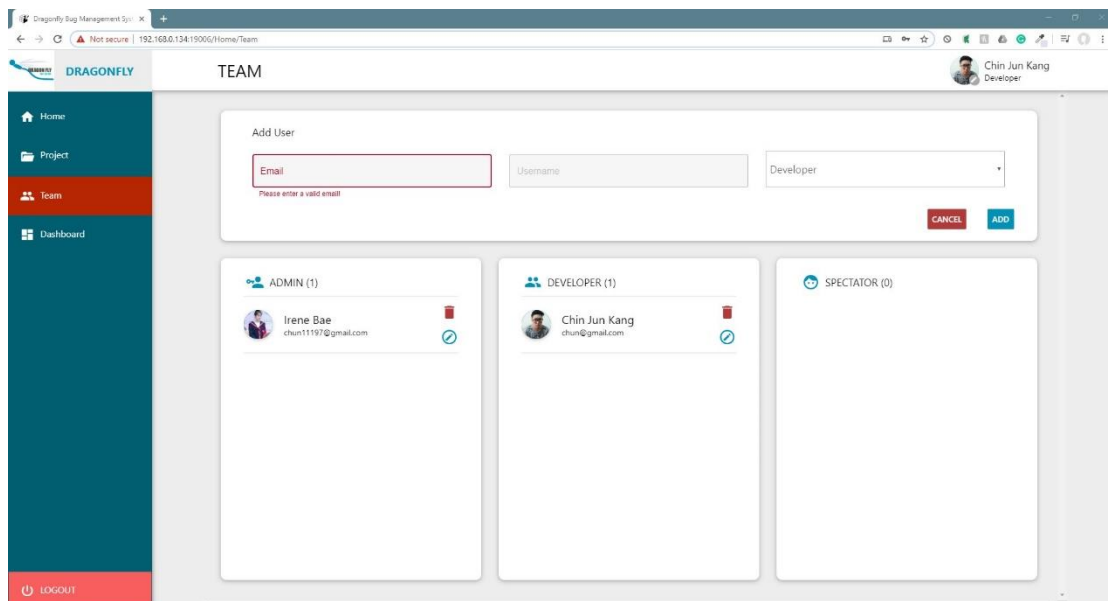


Figure 5.4.34: The Team Page

Dashboard Page

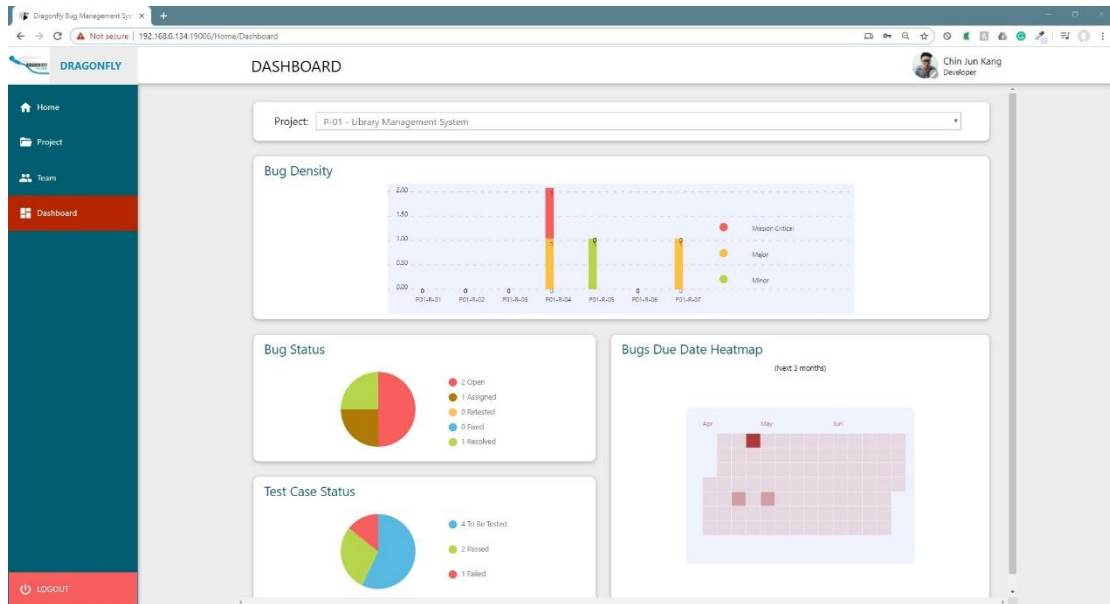


Figure 5.4.35: The Dashboard Page

With a bigger screen, all the charts are able to fit in the screen with one glance.

Add Project Page

The screenshot shows the 'PROJECT' page for the 'Dragonfly Bug Management System'. The left sidebar contains navigation links: Home, Project (highlighted), Team, and Dashboard. The main content area displays a 'PROJECT INFORMATION' form. The form includes a 'Project Title' field with a validation message '* Please enter a valid title', 'Start Date' and 'Due Date' fields with date pickers, and a 'Project Description' text area. At the bottom right of the form are 'CANCEL' and 'ADD' buttons.

Figure 5.4.36: The Add Project Page

Add Requirement Page

The screenshot shows the 'Add Requirement' page of the Dragonfly Bug Management System. The page has a dark teal sidebar on the left with navigation links: Home, Project, Team, and Dashboard. At the bottom of the sidebar is a red 'LOGOUT' button. The main content area is titled 'REQUIREMENT' and features a white form titled 'REQUIREMENT INFORMATION'. The form contains a 'Requirement Title' text field with a red border and a red error message 'Please enter a valid title'. Below this are two dropdown menus: 'Project' (set to 'Library Management System') and 'Priority' (set to 'Immediate'). A large 'Requirement Description' text area is below these. At the bottom right of the form are two buttons: a red 'CANCEL' button and a blue 'ADD' button. The top of the browser window shows the URL '192.168.0.134:19005/Home/AddRequirement' and the user 'Chin Jun Kang Developer'.

Figure 5.4.37: The Add Requirement Page

Add Test Case Page

The screenshot shows the 'Add Test Case' page of the Dragonfly Bug Management System. The layout is similar to the previous page, with a dark teal sidebar and a main content area titled 'TEST CASE'. The form is titled 'TEST CASE INFORMATION' and includes a 'Project' dropdown menu set to 'Library Management System'. It features several text fields with red borders and error messages: 'Objective' (error: 'Objective cannot be empty'), 'Related Requirement' (set to 'none'), 'Status' (set to 'To Be Tested'), 'Tester' (set to 'none'), 'Input' (error: 'Input cannot be empty'), 'Expected Result' (error: 'Expected Result cannot be empty'), and 'Special Procedure Requirement'. At the bottom, there is a dropdown for 'Interase Dependencies (multiple)' set to 'none'. Red 'CANCEL' and blue 'ADD' buttons are at the bottom right. The browser window shows the URL '192.168.0.134:19005/Home/AddTestCase' and the user 'Chin Jun Kang Developer'.

Figure 5.4.38: The Add Test Case Page

Add Bug Page

Dragonfly Bug Management System

BUG

Chin Jun Kang Developer

BUG INFORMATION

Bug Title
Title cannot be empty

Project* Library Management System

Related Test Case none

Bug Description

Status* Open Due Date: 2020-04-24

Severity Minor Priority Immediate

Build Information

Environment

Assign to none Tester none

LOGOUT

Figure 5.4.39: The Add Bug Page - 1

Dragonfly Bug Management System

BUG

Chin Jun Kang Developer

Severity Minor Priority Immediate

Build Information

Environment

Assign to none Tester none

Step to Reproduce

Attempt to Repeat

Attach images (multiple)

1587672062206.png

CANCEL ADD

LOGOUT

Figure 5.4.40: The Add Bug Page - 2

Requirement Page

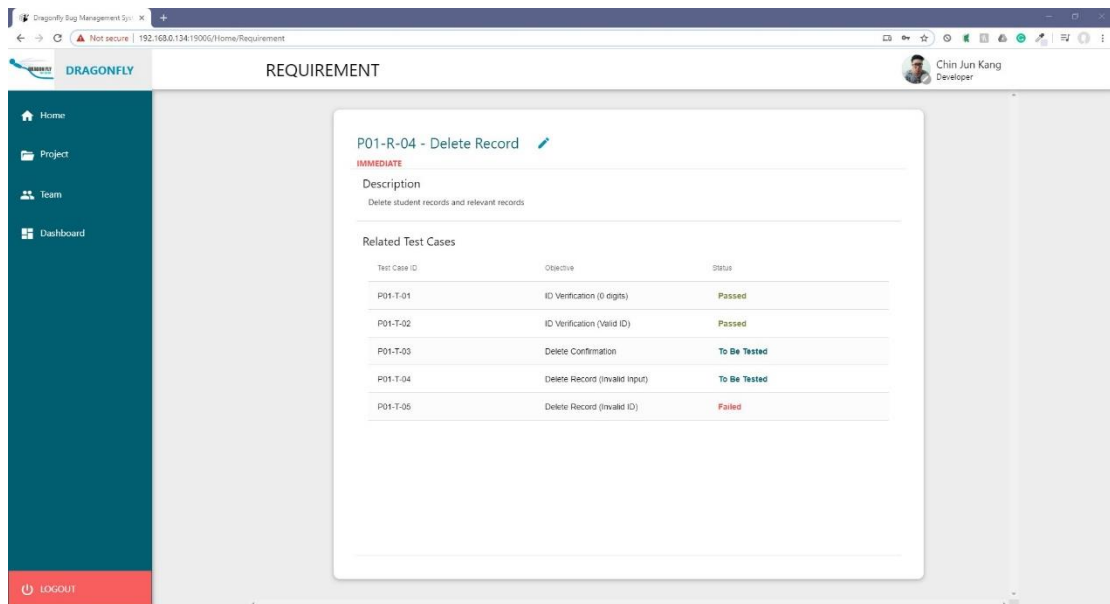


Figure 5.4.41: The Requirement Page

Test Case Page

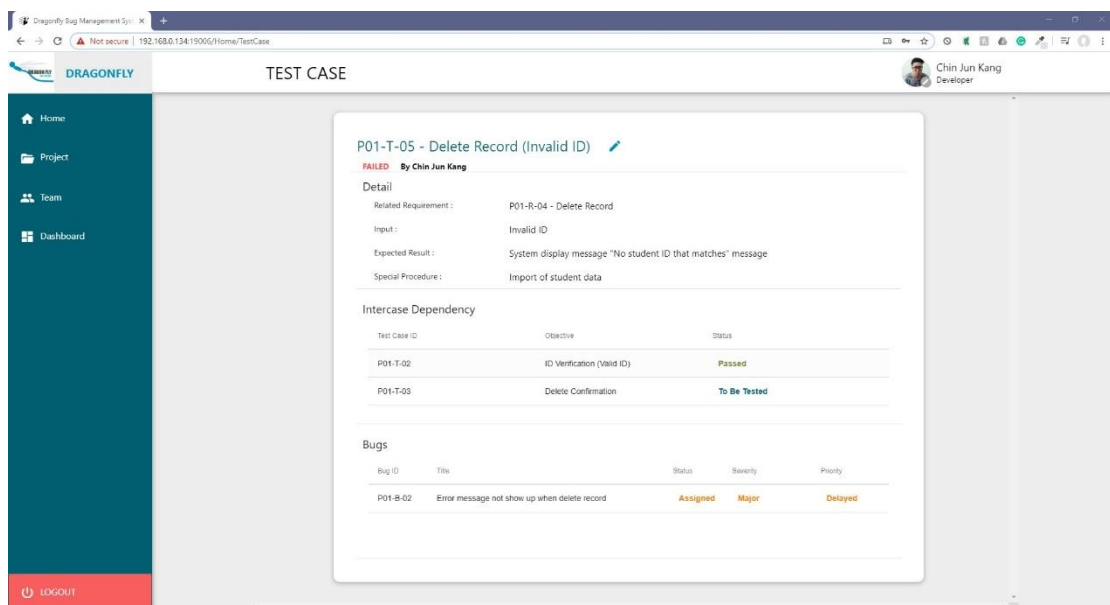


Figure 5.4.42: The Test Case Page

Bug Page

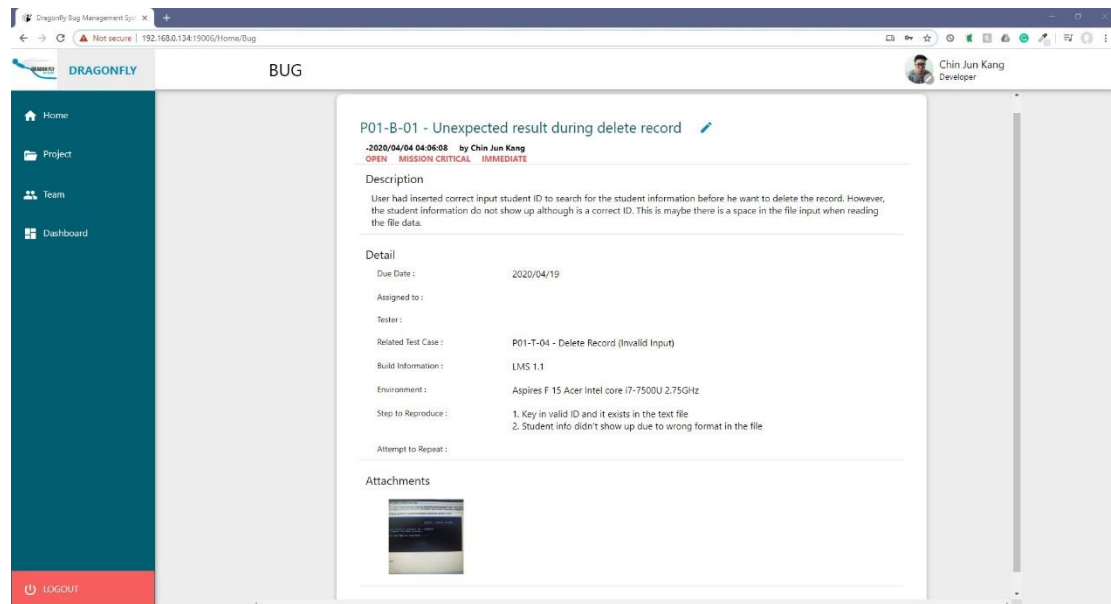


Figure 5.4.43: The Bug Page


5.5. Use Case Testing

Use Case	Input	Actual Output	Expected Output	Result
UC001 Login	Valid email and password	Successful login and navigate to Home screen	Successful login and navigate to Home screen	Passed
	Invalid email	System display error message	System display error message	Passed
	Invalid password	System display error message	System display error message	Passed
UC002 Sign up	Valid email and password	Successful login and navigate to Home screen	Successful login and navigate to Home screen	Passed
	Invalid email	System display error message	System display error message	Passed
	Invalid password	System display error message	System display error message	Passed

UC003 View feed	User press on the feed.	Navigate user to the feed details screen	Navigate user to the feed details screen	Passed
UC004 View dashboard	User selects a project.	System display information and chart for the project.	System display information and chart for the project.	Passed
UC005 View team	User navigates to team screen	System display user list.	System display user list.	Passed
UC006 Create team	User adds new user to the team.	Update and show the new user list.	Update and show the new user list.	Passed
	User removes the last member.	System display error message	System display error message	Passed
	User removes the last admin.	System display error message	System display error message	Passed
UC007 Add project	User enter valid project information	Update database, notify users and navigate to project screen.	Update database, notify users and navigate to project screen.	Passed
	User enter invalid project information.	System display error message	System display error message	Passed
UC008 Add requirement	User enter valid requirement information	Update database, notify users and navigate to requirement screen.	Update database, notify users and navigate to requirement screen.	Passed

	User enter invalid requirement information.	System display error message	System display error message	Passed
UC009 Add test case	User enter valid test case information	Update database, notify users and navigate to test case screen.	Update database, notify users and navigate to test case screen.	Passed
	User enter invalid test case information.	System display error message	System display error message	Passed
UC010 Add bug	User enter valid bug information	Update database, notify users and navigate to bug screen.	Update database, notify users and navigate to bug screen.	Passed
	User enter invalid bug information.	System display error message	System display error message	Passed
UC011 View project details	User navigates to project screen	System display project information, requirement, test case and bug list	System display project information, requirement, test case and bug list	Passed

5.6. Implementation Issues and Challenges

 Compatibility issues of APIs and native features across platforms.

To implement all the features in the user requirements, APIs and some native device features are needed. For example, to upload pictures as information in bug reports, mobile platforms have a native camera while not all laptops and desktops have a

camera. To ensure the system well-functioning across any platforms, some functions are limited to particular platforms and develop alternative approaches for the functions.

High usability design across platforms.

Building a high usability system is one of the objectives of this project. However, different platforms have different user interactions. It is a challenge to build a high usability system across different platforms while maintaining the consistency of the user interface design. For instance, the mobile platform has a smaller screen so it might be more appropriate to limit the information on the screen to focus in few sections while for devices with a bigger screen, it might be considered to provide more information on the screen.

CHAPTER 6: CONCLUSION

6.1. Project Review

Bugs management is often a challenge in software development as bugs can be easily misunderstood. Misunderstood bugs are resource-wasting as the bug might be assigned to a not familiar developer. Although there are many existing solutions, most of them are not standardized and have only limited progress tracking and traceability features. This project aims to help the software development team to manage bugs with standardization.

Throughout this project, many problems have encountered and most of them are because of cross-platform compatibility. By the time this project is developed, React Native for the web is in beta version, it is unstable and risen many problems during development. Most of the time the functions provided by the library cannot be used and a self-built version has to create work across multi-platform which consume a lot of time during the project development.

The objectives include providing bug management functionality and progress tracking functionality which follows the *ISO/IEC/IEEE 29119* Software Testing Standard to provide better bugs management. The final system built have achieved all the objectives. By using the system, users can focus on software development and not need to worry about bugs documentation and hence result in shorter development time and better product quality.

6.2. Novelties and Contribution

Compare to other existing bug management system, the system built in this project have follows the *ISO/IEC/IEEE 29119* Software Testing Standard to standardized the software testing process. The final system also provides bug traceability features which other existing bug management system does not have. Bug traceability links up the relationship between requirements, test cases and software bugs and gives a clear view of the current progress of the project and the quality of the project. This is very important for developers to ensure the project met the user requirements and for the client to monitor the project progress and quality.

6.3. Future Work

Currently, this project only focuses on bug management functions, the system can be further extended to include more software testing functions. As the *ISO/IEC/IEEE 29119* Software Testing Standard discussed thoroughly the whole software testing process, it can be used to extend this project's functions., this provides users a complete software testing process which could improve the software development quality.

Other sign in methods can also provide as the future extension of this project. Since Firebase Authentication is used in this project, it provides Google, Facebook and Twitter authentication service which could reduce the troublesome and increase the effectiveness of signing up for a new account. This could increase the pleasant of the project and hence increase the user's satisfaction.

BIBLIOGRAPHY

- Ahonen, JJ, Junttila, T, & Sakkinen, M 2004, 'Impacts of the Organizational Model on Testing: Three Industrial Cases', *Empirical Software Engineering*, vol. 9, no. 4, pp. 275–296, <<http://link.springer.com/10.1023/B:EMSE.0000039880.99096.af>>.
- 'Airbrake' n.d., viewed 7 August 2019, <<https://airbrake.io/>>.
- 'Backlog' n.d., viewed 7 August 2019, <<https://backlog.com/>>.
- Bug* n.d., *tutorialspoint*, viewed 6 August 2019, <https://www.tutorialspoint.com/software_testing_dictionary/bug>.
- D'Ambros, M, Lanza, M, & Pinzger, M 2007, "A Bug's Life" Visualizing a Bug Database', in, *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, IEEE, pp.113–120, <<http://ieeexplore.ieee.org/document/4290709/>>.
- Davies, S & Roper, M 2014, 'What's in a bug report?', in, *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14*, ACM Press, New York, New York, USA, pp.1–10, <<https://strathprints.strath.ac.uk/50083/1/esemBugReportDataCamera.pdf>>.
- Defect Management Process in Software Testing (Bug Report Template)* n.d., *Guru99*, viewed 6 August 2019, <<https://www.guru99.com/defect-management-process.html>>.
- Graham, D, Veenendaal, E, & Evans, I 2008, *Foundations of software testing : ISTQB certification*, Course Technology Cengage Learning.
- International Organization for Standardization 2013, 'Software Testing Standard', *ISO/IEC/IEEE 29119*, <<http://www.softwaretestingstandard.org>>.
- Kajko-Mattsson, M & Bjornsson, T 2007, 'Outlining Developers' Testing Process Model', in, *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)*, IEEE, pp.263–270, <<http://ieeexplore.ieee.org/document/4301088/>>.
- Singh, S 2013, 'Analysis of Bug Tracking Tools', *International Journal of Scientific &*

BIBLIOGRAPHY

Engineering Research, vol. 4, no. 7, pp. 134–140.

Strate, JD & Laplante, PA 2013, 'A Literature Review of Research in Software Defect Reporting', *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 444–454, <<http://ieeexplore.ieee.org/document/6509998/>>.

What is Defect or bugs or faults in software testing? 2012, *TRY QA*, viewed 6 August 2019, <<http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/>>.

'Zoho BugTracker' 2010, <<https://www.zoho.com/bugtracker/>>.

Software Bugs Management (ISO/IEC/IEEE 29119 Standard) System

Prepared by Chin Jun Kang

Supervised by Ts Tan Teik Boon

DESCRIPTION

Bug management is a crucial process in the software development process. Low-quality bug management could result in low-quality deliverable, resources wasted and profit lost. However, bug management is not an easy process, software developers often have a hard time managing software bugs. To solve this problem, this project proposes a bug management system that is accessible across different platforms as a solution.

MOTIVATION

Although there are many existing bug management systems, most of them do not provide a standardized bug management process. The proposed system provides a standardized bug management process following the ISO/IEC/IEEE 29119 Software Testing Standard. It can be hard to communicate between software developer and non-technical business unit in development team as they have different level of technical knowledge. With the help of the proposed system, software developer can focus on the software developing and software development team including business unit can communicate more effectively.

PROJECT OBJECTIVES

- To offer bug tracking and management functionalities to the users
- To keep track the progress of bug management
- To standardize the bug tracking and management process by using the ISO/IEC/IEEE 29119 Software Testing Standard.

PROJECT SCOPES

- To develop a system with bug management functionalities.
- To develop the bug management system in both the web and mobile platforms.
- To develop a centralized database to maintain the bugs for the bug management system.

BCS (Hons) Computer Science Faculty of Information and Communication Technology (Kampar Campus), UTAR



PLAGIARISM CHECK RESULT

Turnitin - Google Chrome
turnitin.com/newreport.asp?r=88.3335963545246&svr=498&lang=en_us&oid=1305944808&pbid=2&ft=1&ro=

preferences

turnitin
Originality Report

Processed on: 24-Apr-2020 07:00 +08
ID: 1305944808
Word Count: 8332
Submitted: 3

Software_Bugs_Mar
29119-S...
By Chin Jun Kang

Similarity Index
10%

Similarity by Source
Internet Sources: 3%
Publications: 3%
Student Papers: 10%

Document Viewer

include quoted include bibliography exclude small matches mode: show highest matches together Change mode

Poor bugs management could post problems in every software development lifecycle. First is the misunderstanding of bugs, not all bugs are caused by coding errors, misunderstanding can be caused by miscommunication within the development team, unclear requirements, misunderstanding of requirements and poor documentation of bugs. Sometimes issues are reported technically (as bugs), but a development team consisting of non-technical members such as customer, stakeholders, organization, etc. It could be hard for non-technical members to involve in the software development process this can cause the requirement and issue to become unclear. Moreover, fixing bugs can become not effective and time-consuming if the developers do not have a standardized testing process. As mentioned by (Ahonen et al. 2004) one of the challenges in testing defects is that it is hard to ensure that all members in the development team follow good practices. During a conference, (Kajko- Mattsson & Bjornsson 2007) have also reported that organizations do not document testing processes properly. These testing-related problems can delay the time a bug is detected during the software development process and result in late in solving the bugs. Last but not least, it will be an ineffective use of intellectual if a bug is not documented well (Ahonen et al. 2004). A development team may consist of experts from different domain/fields, an issue found by a developer may from a different field, he/she may use a long time to solve the problem. In a larger development team, there will be different members to do the testing which may not have coding skill to resolve the issue. The member who found the bug has to determine the skills needed to solve the problem and passes the bug to the members who have the required skill or informs every member in the team to find out who has the ability to solve the problem. If the bug is documented well, other developers who are expert in the domain can view the documentation and resolve the problem effectively. 1.1. Background and Motivation Software bugs are always a challenge

in the software development process. Software bugs are errors occur in 80

software including coding error, unexpected software behaviour, misunderstood user requirement etc. These software bugs can prolong the project development process and delay the delivery of

- 1% match (Internet from 11-Dec-2019)
https://limswiki.org/index.php/ISO/IEC_2911
- 1% match (Internet from 29-Jan-2015)
<http://softwaretestingstandard.org>
- < 1% match (Internet from 11-May-2015)
<http://blackboxtest.com>
- < 1% match (student papers from 06-Oct-2014)
[Submitted to 76830](#)
- < 1% match (publications)
[Amarmend Dashbalbar, Eun-Chul Lee, Jung-Won Lee, Byeongjeong Lee, "Describing Activities to Verify Artifacts \(Documents and Program\) in Software R&D", Journal of Internet Computing and Services, 2016](#)
- < 1% match (Internet from 22-May-2019)
<https://fex-team.github.io/blog/2019/01/fex-weekly-14/>
- < 1% match (student papers from 05-Jul-2012)
[Submitted to University of Greenwich](#)
- < 1% match (student papers from 13-Aug-2016)
[Submitted to University of Wales Institute, Cardiff](#)
- < 1% match (Internet from 30-Sep-2019)
<https://www.omg.org/spec/UTP2/2.0/PDF>
- < 1% match (publications)
["Matching Context Aware Software Testing Design Techniques to ISO/IEC/IEEE 29119", Communications in Computer and Information Science, 2015.](#)
- < 1% match (publications)
[Adam Roman, "Thinking-Driven Testing", Springer Science and Business Media LLC,](#)

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	CHIN JUN KANG
ID Number(s)	15ACB04844
Programme / Course	BACHELOR OF COMPUTER SCIENCE (HONS)
Title of Final Year Project	SOFTWARE BUGS MANAGEMENT (ISO/IEC/IEEE 29119 STANDARD) SYSTEM

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u>10</u> % Similarity by source Internet Sources: <u>3</u> % Publications: <u>3</u> % Student Papers: <u>10</u> %	
Number of individual sources listed of more than 3% similarity: <u>0</u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Name: Tan Teik Boon

Date: 24 April 2020

Signature of Co-Supervisor

Name: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN



FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	15ACB04844
Student Name	CHIN JUN KANG
Supervisor Name	Ts. TAN TEIK BOON

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Front Cover
✓	Signed Report Status Declaration Form
✓	Title Page
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <div style="text-align: center;">  </div> <p>(Signature of Student) Date: 24 April 2020</p>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <div style="text-align: center;">  </div> <p>(Signature of Supervisor) Date: 24 April 2020</p>
--	---

CHECKLIST FOR FYP2 THESIS SUBMISSION