SESSION 2021/2022, SEMESTER 2

SECJ 2203: SOFTWARE ENGINEERING

ALTERNATIVE ASSESSMENT:

SOFTWARE TESTING DOCUMENT

PROJECT TITLE: Inferno 2u2i Final Year Project With Industry (FYP-I) Management System

Name	**Shahril Bin Saiful Bahri**
Matric No.	**A20EC0144**
Year / Programme	**2 SECBH**
Section	**01**
Lecturer Name	**Puan Nor Hawaniah Zakaria**

Table of Contents

Section A: Requirements-based Testing

A1 Functional Requirements

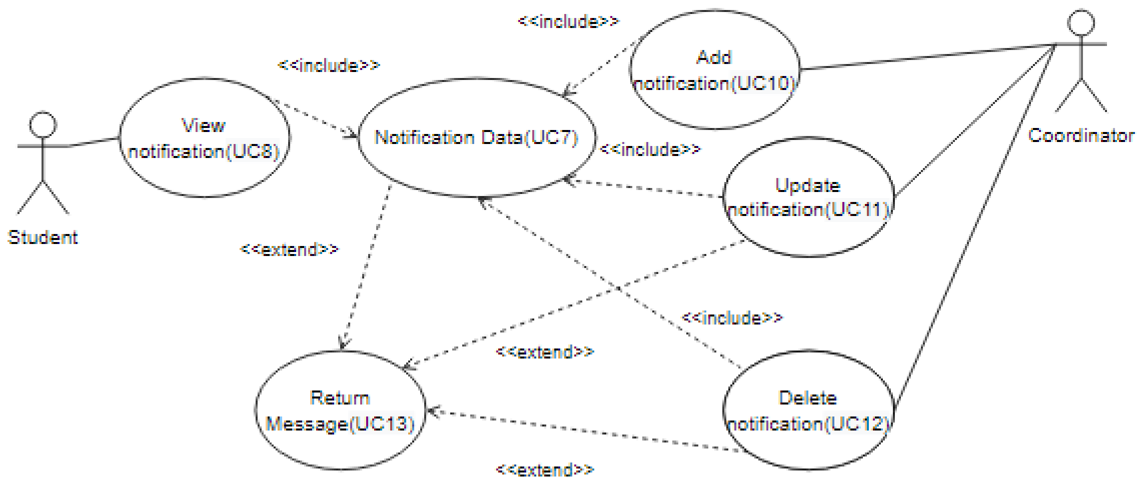


Diagram 1: Use Case Notification

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	TR ID	Test Requirements
UC <09> <Add Notification>	TR*001*	Validate that Coordinator able press Notifications
	TR*002*	Validate that Coordinator able to choose Add Notification
	TR*003*	Validate that Coordinator able select user (to send), enter title and content of Notification
	TR*004*	Validate that Coordinator able to Add Notification
	TR*005*	Validate that Student able to choose Add Notification

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR002	**TCTR002_01**	User click Add Notification	Display "Add Notification Tab"

	TC$_{TR002}$_02	User click Add Notification	No output display
	TC$_{TR002}$_03	User didn't click Add Notification	No output display
TR$_{003}$	**TC$_{TR003}$_01**	User selects user to send	Display "user name" on Add Notification Tab
	TC$_{TR003}$_02	User enter title and content of Notification	Display the data from user input "Title <user input> Content <user input>"
	TC$_{TR003}$_03	User enter title and content of Notification	Unable to display user input
TR$_{004}$	**TC$_{TR004}$_01**	User click Add Notification after successfully input (user, title and content)	Display "Notification is added"
	TC$_{TR004}$_02	User click Add Notification but missing (user/title/content)	Display "This area is missing, please enter the input"
	TC$_{TR004}$_03	User didn't click Add Notification	No output display

A2 Non-Functional Requirements

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	**TR ID**	**Test Requirements**
Performance of System	TR$_{001}$	System should be able to take 1.0 second after clicking anything on the system GUI
	TR$_{002}$	System should be able to handle more than 1000 students (including staff) and at least 200 users at one time during peak hours
	TR$_{003}$	Sytem should have the capacity of having 10 contents in a page with scrolling involved.

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	**Case No.**	**Data Entered**	**Expected Result**
TR*001*	**TC$_{\text{TR}\textit{001}}$_01**	Coordinator clicks on Add Notification	Display "Add Notification Tab" after 1.0 seconds.
	TC$_{\text{TR}\textit{002}}$_02	Coordinator clicks on Delete Notification	Display "Notification List" after 1.0 seconds.
	TC$_{\text{TR}\textit{003}}$_03	Coordinator clicks on Delete Notification	Display "Notification List" after 1.0 seconds.

A3 Summary

In my opinion, the best level of testing is the unit testing is because this type of testing is performed at the earliest stages of development process. This is an advantage as we are able to detect any errors in the early stages of the software and by doing so it minimize the software development risk as well time and money in changing the full completed software in the future. In conclusion, I think unit testing is the best level of testing to implement in any software engineering project.

Section B: Black-box Testing

B1 Object Class

B1.1 Equivalence Partitioning and Boundary Value Analysis

Table 5. Equivalence Partition and Input Range

Object class	Attributes	Equivalence Partition and Input Range
Coordinator	coordinatorID	1. Valid : coordinatorID must be characters [a-z] & characters [0-9] with length between 1-10 2. Invalid : coordinatorID must be characters [a-z] & characters [0-9] with length more than 10 3. Invalid : coordinatorID must be characters [a-z] & characters [0-9] with length less than 1
	coordinatorName	1. Valid : coordinatorName must be characters [a-z] with length between 10-50 2. Invalid : coordinatorName must be characters [a-z] with length more 50 3. Invalid : coordinatorName must be characters [a-z] with length less 10
	coordinatorEmail	1. Valid : coordinatorEmail must be characters [a-z] with length between 10-20 & “@example.com” 2. Invalid : coordinatorEmail must be characters [a-z] with length more 20 & “@example.com”

		3. Invalid : coordinatorEmail must be characters [a-z] with length less 10 & "@example.com"
	coordinatorPassword	1. Valid : coordinatorPassword must be characters [a-z] with length between 10-20 2. Invalid : coordinatorPassword must be characters [a-z] with length more 20 3. Invalid : coordinatorPassword must be characters [a-z] with length less 10
	coordinatorAddress	1. Valid : coordinatorPassword must be characters [a-z] with length between 10-20 2. Invalid : coordinatorPassword must be characters [a-z] with length more 20 3. Invalid : coordinatorPassword must be characters [a-z] with length less 10
	coordinatorAge	1. Valid : coordinatorAge must be integer more than 0 2. Invalid : coordinatorAge must be integer less than 0
	coordinatorDateOfBirth	1. Valid : coordinatorDateOfBirth must be date that is less than the currentDate 2. Invalid : coordinatorDateOfBirth must be date that is same with the currentDate 3. Invalid : coordinatorDateOfBirth must be date that is same more the currentDate
	coordinatorGender	1. Valid : coordinatorGender must be character [M/F]

		2. Invalid : coordinatorGender must be character other than [M/F]
	coordinatorPhoneNo	1. Valid : coordinatorPhoneNo must be characters [0-9] and between 1 and 15 2. Invalid : coordinatorPhoneNo must be characters [0-9] and more than 15 3. Invalid : coordinatorPhoneNo must be characters [0-9] and less than 1

Object class	Attributes	Equivalence Partition and Input Range
Notification Database	notiId	1. Valid : notiId must be characters [a-z] & characters [0-9] with the length between 1-10 2. Invalid : notiId must be characters [a-z] & characters [0-9] with the length more than 10 3. Invalid : notiId must be characters [a-z] & characters [0-9] with the length less than 1
	notiTitle	1. Valid : notiTitle must be characters [a-z] with the length between 5-50 2. Invalid : notiTitle must be characters [a-z] with the length more than 50 3. Invalid : notiTitle must be characters [a-z] with the length less than 5
	notiContent	1. Valid : notiContent must be characters [a-z] with the length between 5- 100 2. Invalid : notiContent must be characters [a-z] with the length more than 100 3. Invalid : notiContent must be characters [a-z] with the length less than 5
	notiTime	1. Valid : notiTime must be time which equals to the currentTime

		2. Invalid : notiTime must be time which more than the currentTime 3. Invalid : notiTime must be time which less than the currentTime
	notiDate	1. Valid: notiDate must be date which equals to the currentDate 2. Invalid : notiDate must be date which more than the currentDate 3. Invalid : notiDate must be date which less than the currentDate
	notiStatus	1. Valid : notiStatus must be characters [a-z] with the length between 1-20 2. Invalid : notiStatus must be characters [a-z] with the length more than 20 3. Invalid : notiStatus must be characters [a-z] with the length less than 1

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: Coordinator

Method name: AddData

Case No.	**Equivalence Class**	**Pass /Fail ?**	**Representative (BVA)**	**Expected Result**
TC*001*	notiContent is character [a-z] with length between 5-100	Pass	Hello everyone I love you	Data is added to Notification
TC*002*	notiContent is character [a-z] with length more than 100	Fail	**	Invalid data
TC003	notiContent is character [a-z] with length less than 5	Fail	Hai	Invalid length

Object name: NotificationDatabase

Method name: AddData

Case No.	**Equivalence Class**	**Pass /Fail ?**	**Representative (BVA)**	**Expected Result**
TC*001*	notiContent is character [a-z] with length between 5-100	Pass	Hello everyone I love you	Data is added to Notification
TC*002*	notiContent is character [a-z] with length more than 100	Fail	**	Invalid data
TC003	notiContent is character [a-z] with length less than 5	Fail	Hai	Invalid length

B2 Summary

In my opinion, this level of black box testing is to check whether the user entered the right input. This is because it is important to check if the data is correct or it will make an error to the system. For example we take the method AddData for both object Coordinator & NotificationDatabase and the attribute being the notiContent. If the user successfully entered the right format and length for the notiContent than they system will be able to insert the data to the table. If we use the wrong format or length, the system cannot process the data and it will be an invalid data. In conclusion, I think its very important to do black-box strategy because it can maintain the system to be working as it is supposed to do.

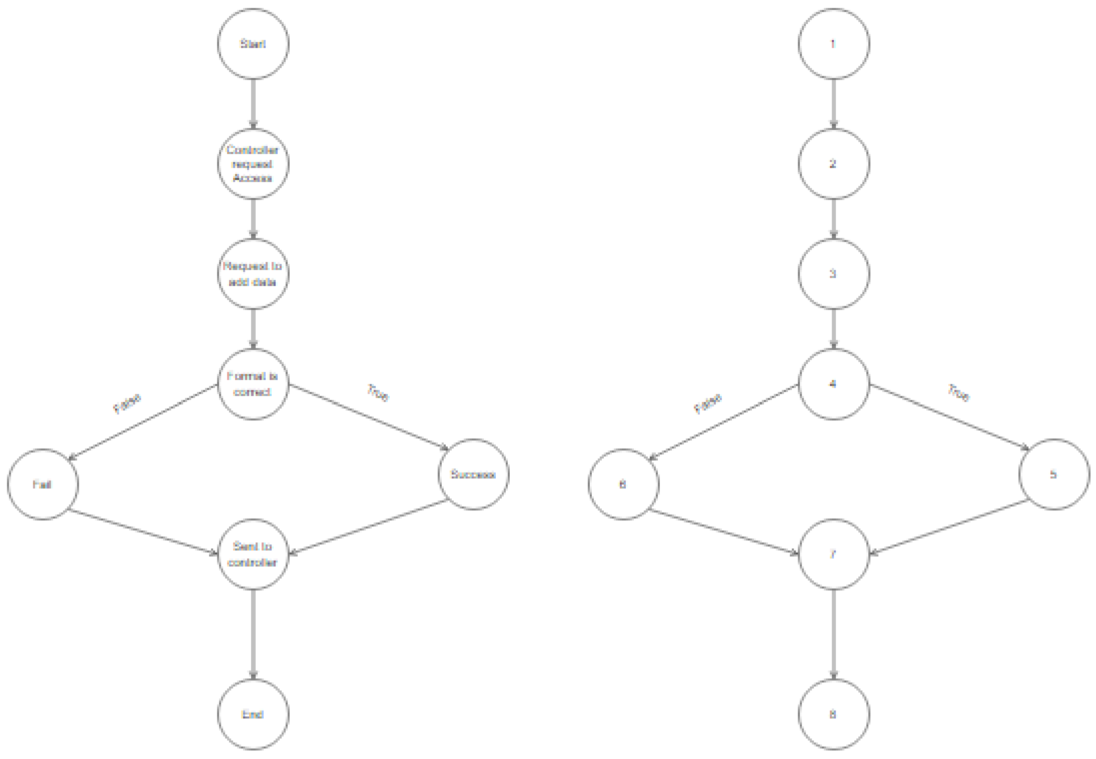
Section C: White-box Testing

C1 Methods Class

Table 7. Methods Class

Entity Name	NotificationDatabase
Method Name	addData()
Input	notiId, notiTitle, notiContent, notiTime, notiDate, notiStatus
Output	-
Algorithm	1. Start 2. Controller request access to database 3. If the controller request to add data 3.1 If format is correct 3.1.1 Data is added to database 3.2 Else 3.2.1 Failed to add data to database 4. Existed data will be sent back to controller 5. End

C1.1 Flow Graph



C1.2 Cyclomatic Complexity

Formula 1: V(G) = #Edges - #Nodes +2

= 8 – 8 + 2

= 2

Formula 2: V(G) = #Predicates Nodes +1

= 1 + 1

= 2

Formula 3: V(G) = Region

= 2

Independent Path:

1. 1-2-3-4-5-7-8
2. 1-2-3-4-6-7-8

C1.3 Test Cases

Table 8. Independent Path Based Test Cases

Case No.	Independent Path	Pass/Fail?	Data* for Test Cases	Expected Result
TC*001*	1-2-3-4-5-7-8	Pass	noti Content= ”Hello Everyone”	Data is added to Notification
TC*002*	1-2-3-4-6-7-8	Pass	noti Content= “ ”	Invalid Data

C2 Summary

In my opinion, for the best level of white-box testing is unit testing because it is performed on each unit or code as it is developing. This method helps us to find the path of the system which can either be pass or fail based on our algorithm. For example for in table 8 we can determine that if the system works. In conclusion, its important to do white-box testing to ensure that our system is working fine.