SESSION 2021/2022, SEMESTER 2

SECJ 2203: SOFTWARE ENGINEERING

ALTERNATIVE ASSESSMENT:

SOFTWARE TESTING DOCUMENT

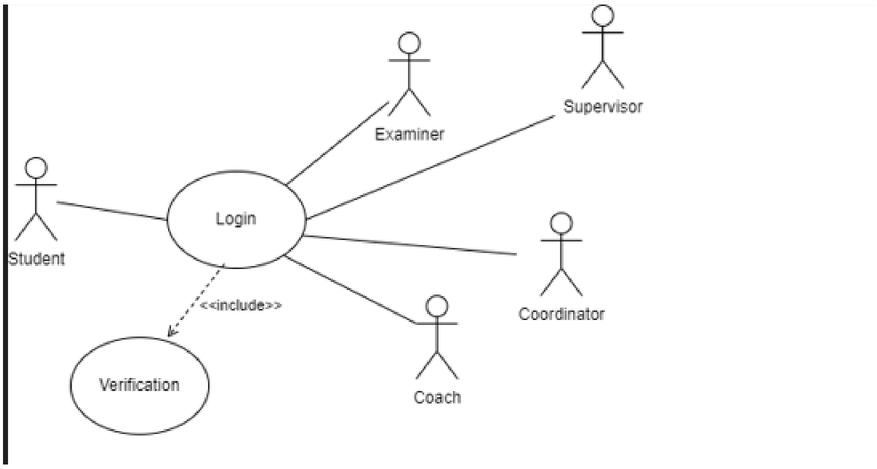
PROJECT TITLE: Inferno 2u2i Final Year Project with Industry (FYP-I) Management System

Name	**IMAN EHSAN BIN HASSAN**
Matric No.	**A20EC0048**
Year / Programme	**2 SECBH**
Section	**01**
Lecturer Name	**PUAN NOR HAWANIAH ZAKARIA**

Table of Contents

Section A: Requirements-based Testing

A1 Functional Requirements



UC001 : Use Case <Login>

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	TR ID	Test Requirements
UC001 <Login>	TR***001***	Validate that the student enters the studentID
	TR***002***	Validate that the student enters the password
	TR***003***	Validate that the student to click the login button
	TR***004***	Validate that the student to request the verification
	TR***005***	Validate that the student to have access to the system

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR*001*	**TC**$_{\textbf{TR}\textit{001}}$**_01**	studentID :imanehsan	studentID are accepted
	TC$_{\textbf{TR}\textit{001}}$**_02**	studentID: imanehsan1	studentID are incorrect
	TC$_{\textbf{TR}\textit{001}}$**_03**	studentID : (blank)	System asked the user to enter it again
TR*002*	**TC**$_{\textbf{TR}\textit{002}}$**_01**	Password: 2408iman	Password are accepted

	TCTR*002*_02	studentID: imanehsan Password: 240801	Password are incorrect and not match with the studentID
	TCTR*002*_03	Password : (blank)	System asked the user to enter it again
TR*004*	**TCTR*004*_01**	User does not click the button when the system tell to do so	Verification cannot be proceed
	TCTR*004*_02	User does not entered anything	The system pop out a message saying there are some error
	TCTR*004*_03	User does not entered anything	Verification is failed

A2 Non-Functional Requirements

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	**TR ID**	**Test Requirements**
Security	TR*001*	Access control to the users
	TR*002*	Authentication to make sure it is the right user
	TR*003*	Controlling the virus that entered the system

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	**Case No.**	**Data Entered**	**Expected Result**
TR*001*	**TCTR*001*_01**	studentID :imanehsan password: 2408iman	Access is accepted and user will entered the main menu of the system
	TCTR*001*_02	studentID :imanehsan1 password: 2408iman	Access is denied and the user are asked to re-enter the

			studentID and password again
	TC$_{TR001}$_03	studentID:imanehsan password: (blank)	Error has occurred and the system will bring the user to the login menu

A3 Summary

The level of testing is Component Testing . This is because through unit testing , we can actually analysing the login system and see whether the system can function well or not. This is easier as it is the smallest part and we can focus more on the login part more easily. We can actually find out the mistakes before proceed to other steps.

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
Student	*studentID*	Valid - More than 0 and must contain characters [a-z] between 10-15 characters Invalid - less than 0 and must contain characters[a-z] between 10-15 characters invalid - More than 0 and must contain characters[a-z] and numbers [0-9] between 10-15 characters
	studentEmail	Valid - Must contain @graduate.utm.my Invalid – does not contain @graduate.utm.my
	studentPassword	Valid - Must contain alphabetic and number with length between 3-12 Invalid - contain alphabetic[a-z] with length between 3-12 Invalid - contain number [0-9] with length between 3-12
	studentName	Valid - Cannot contain special characters and only can contain alphabectec number Invalid – Cannot contain special characters and only can contain alphabectec number Invalid – Only contain special characters

Object class	**Attributes**	**Equivalence Partition and Input Range**
Supervisor	*svID*	Valid - More than 0 and must contain characters [a-z] between 10-15 characters Invalid - less than 0 and must contain characters[a-z] between 10-15 characters invalid - More than 0 and must contain characters[a-z] and numbers [0-9] between 10-15 characters
	svEmail	Valid - Must contain @graduate.utm.my Invalid – does not contain @graduate.utm.my
	svPassword	Valid - Must contain alphabetic and number with length between 3-12 Invalid - contain alphabetic[a-z] with length between 3-12 Invalid - contain number [0-9] with length between 3-12
	svName	Valid - Cannot contain special characters and only can contain alphabectec number Invalid – Cannot contain special characters and only can contain alphabectec number Invalid – Only contain special characters

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: Student

Method name: EnterUserNamePassword()

Case No.	**Equivalence Class**	**Pass/Fail?**	**Representative (BVA)**	**Expected Result**
TC*001*	studentID >0 & character [a-z] & 10< studentID < 15	Pass	imanehsan	Access accepted
TC002	studentID >0 & not character [a-z] & 10< studentID < 15	Fail	imanehsan12134	Access Denied
TC003	studentID <0 & character [a-z] & 10< studentID < 15	Fail	(blank)	Access Denied

Object name: Supervisor

Method name: EnterUserNamePassword()

Case No.	**Equivalence Class**	**Pass/Fail?**	**Representative (BVA)**	**Expected Result**
TC*004*	svID >0 & character [a-z] & 10< svID < 15	Pass	shahrilamiris	Access accepted
TC005	svID >0 & not character [a-z] & 10< svID < 15	Fail	shahrilamiris12134	Access Denied
TC006	svID <0 & character [a-z] & 10< svID < 15	Fail	(blank)	Access Denied

B2 Summary

Which level of testing that black-box strategy is appropriate to be executed? Give your opinion and justify your answer.

In my opinion, the level of testing for black box strategy is system testing. This is because through system testing , we can check whether the system will allow the user to enter the system although the username or password is incorrect . This is to make sure the system is working properly by examining it to see whether we got any defects on the system. We can also test the whole system for the End to End testing

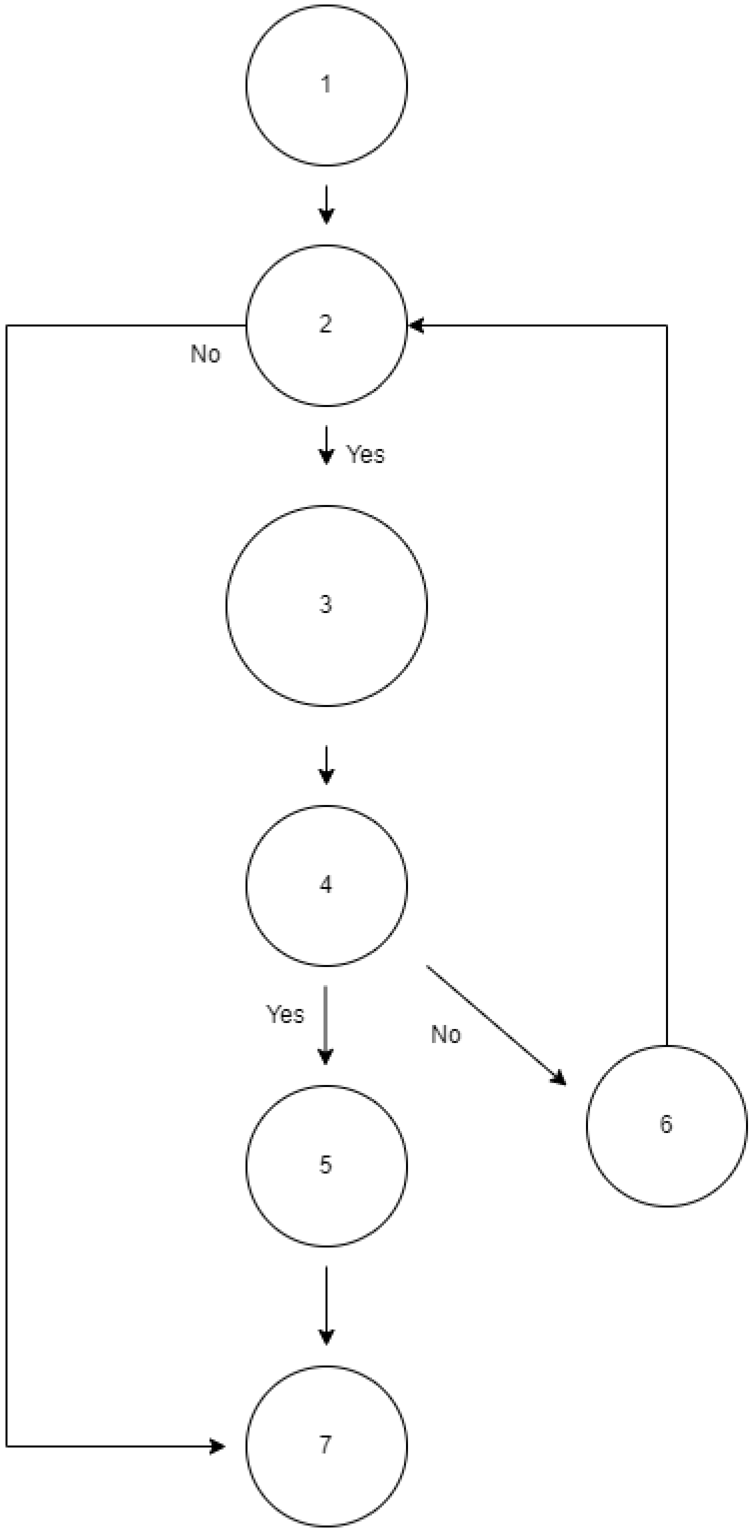
Section C: White-box Testing

C1 Methods Class

Table 7. Methods Class

Entity Name	Student
Method Name	EnterUsernamePassword()
Input	studentID and password
Output	Login successful / Verification accepted
Algorithm	1. Start 2. Get studentID and password 3. If (studentID and password = in the system) 4. Verification in progress 5. Access is accepted 6. The user will be access the main menu of the system 7. Else 8. studentID and password != in the system 9. Access denied 10. System will take to the login menu again 11. End

C1.1 Flow Graph



C1.2 Cyclomatic Complexity

The first formula that I use is e-n+2p= v(G) where e= edge, n=nodes and p= connected components where

V(G) = cyclomatic complexity

E= 8 ,N= 7, P=1

8-7+2(1)= 3

The seond formula is

V(G)= π +1 where π is predicate nodes , it means that the nodes produce two outcomes.

V(G) = 2+1 = 3

The third formula Is the number of regions. This means that the region is surrounded by nodes

V(G) = 3

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	PASS	studentID : imanehsan Password: 2408iman	True
TC*002*	1-2-3-4-6-2-3-4-5-7	PASS	studentID : imanehsan Password: 2408iman	True
TC*003*	1-2-3-4-6-2-7	FAIL	studentID : imanehsan Password: 240801	False

C2 Summary

In my opinion, white box testing strategy is a component testing. This is because white box focus on the algorithm and flow of the system. So by doing that, we need to focus on the coding of the sysem itself and if we got any error, we can detect it and make an amendment quickly and effectively. We also can checks the predifened inputs agfainst any expected and desired outputs that we want. Basically we focus on the flow of the inputs and outpurs by maintainance of the software.

White box can be defined as the internal perspective of the system and how do we see the algorithm of the system. So that it is easier to detetct the bug and improved it.