Testing Report

for

CZ3003 Hydra-Defence

Version 1.0 approved

Prepared by Mohamed Shafiq Bin Peer Mohamed
Heather Chew
Yeong Wei Xian

Team Hydra

13/4/2021

Contents

Revision History

Name	Date	Reason For Changes	Version
Mohamed Shafiq	13/04/2021	Initial Creation	1.0

1. Introduction

1.1 Testing Strategies Adopted

Below underlined are the testing strategies adopted and the reasons we chose to adopt these specific methods.

1.1.1 Black Box Testing

We conducted Black Box Testing as it helps to find the gaps in functionality, usability and other features of our system. This gives us an overview of our software performance and its output. In the event if there is a case failure, we can immediately rectify the problem. It helps to reduce the risk of software failures at the user's end.

1.1.2 White Box Testing

We conducted White Box Testing to ensure that every path through the System Under Test has been identified and tested.

1.1.3 Unit Testing

We have performed unit testing to validate that each unit of the software code performs as expected. With unit testing, we can easily refactor or re-write our code if needed.

1.1.4 Load Testing

We conducted load testing once the server was ready and deployed to ensure that we understood, analyzed, and fixed errors, bugs, and bottlenecks that could occur when the server experienced real-world traffic.

This also enabled us to determine whether our server was capable of living up to the non-functional requirements set out by the client. In this case, with NTU being the client, it was determined that the server would likely have to service about 150 students regularly throughout a semester.

While it was informally known that load was likely to be distributed as opposed to sustained, how the load would likely be distributed was unknown. As a result, we decided to test the system against the assumption that traffic would be sustained and recorded our results accordingly. Refer to the Load Test Report (**page 4 for now**) for more information on our methodology and results.

1.2 Testing Techniques Adopted

Functional Testing Techniques
Black and White Box Testing
Unit Testing
Integration Testing

Non-functional Testing Techniques
Load & Stress Testing

1.3 Testing Tools Adopted

1.3.1 Unity Test Framework (Unit Testing Tool)

Unit testing is an automated testing framework that ensures that functionalities are not broken when changes are made to the code. This helps to save time as the developers are not required to explore every part of the game when changes are made to ensure that there are no errors. This is especially true when there are numerous scenes and scripts.

1.3.2 Postman (API Endpoint Testing Tool)

Postman is an interactive and automatic **tool** for verifying the APIs of your project. **Postman** is a Google Chrome app for interacting with HTTP APIs. It presents you with a friendly GUI for constructing requests and reading responses. It works on the backend, and makes sure that each API is working as intended.

Source: https://www.axelerant.com/resources/team-blog/api-testing-with-postman

Using Postman allowed our back-end development team to easily test any new APIs that were created and store simple smoke tests in specific folders to allow for easy functionality verification before each deployment.

Postman also allowed the front-end development team to easily check the expected output of an API call and verify if their application behaved according to expectations. This way, they could easily determine if the problem lied at the game-level or the back-end level.

1.3.3 Locust (Automatic Performance Testing Tool)

Locust is an open-source load-testing tool written in **Python**. It lets you write tests against your web application which mimic your user's behavior, and then run the tests at scale to help find bottlenecks or other performance issues.

Source: https://www.promptworks.com/blog/load-testing-with-locust/

Locust was used as an alternative to JMeter Apache as it allowed for slightly more granular control while giving aesthetically pleasing and user-friendly results.

2. Black Box Results

2.1 Login Test For Teacher and Student

Test Case ID	Input			Description	Functions to be tested	Expected Output	Actual Output	Pass/Fail
	Username	Password	Teacher/Student					
Student								
LoginS01	Empty Field	abc	Student	Login with empty username field	LoginButton.onClick.AddListener()	"Username or Password field is empty!"	"Username or Password field is empty!"	Pass
LoginS02	James	Empty Field	Student	Login with empty password field	LoginButton.onClick.AddListener()	"Username or Password field is empty!"	"Username or Password field is empty!"	Pass
LoginS03	Olivia01	Wrong Password	Student	Login with wrong password	LoginCheck(string URL)	"Wrong Username or Password!"	"Wrong Username or Password!"	Pass
LoginS04	Wrong Username	kgiur5522	Student	Login with wrong username	LoginCheck(string URL)	"Wrong Username or Password!"	"Wrong Username or Password!"	Pass
LoginS05	SHAFIQ002	pass	Student	Login with correct user name and password	LoginCheck(string URL)	Load Scene Main Menu	Load Scene Main Menu	Pass
Teacher								
LoginT01	Empty Field	123edf	Teacher	Login with empty username field	LoginButton.onClick.AddListener()	"Username or Password field is empty!"	"Username or Password field is empty!"	Pass
LoginT02	Dav23	Empty Field	Teacher	Login with empty password field	LoginButton.onClick.AddListener()	"Username or Password field is empty!"	"Username or Password field is empty!"	Pass
LoginT03	Mike11	Wrong Password	Teacher	Login with wrong password	LoginCheck(string URL)	"Wrong Username or Password!"	"Wrong Username or Password!"	Pass
LoginT04	Wrong Username	kgiur5522	Teacher	Login with wrong username	LoginCheck(string URL)	"Wrong Username or Password!"	"Wrong Username or Password!"	Pass
LoginT05	SHAFIQ002	pass	Teacher	Login with correct user name and password	LoginCheck(string URL)	Load Scene Teacher Menu	Load Scene Teacher Menu	Pass

2.2 Registration Test

Test Case ID	Inputs/Actions			Description	Functions to be tested	Expected Output	Actual Output	Pass/Fail
Reg01	User click "Create New Account"			User wants to create a new account	RegisterButton.onClick()	Load Register Scene	Load Register Scene	Pass
	Username	Password	Confirm Password					
Reg02	zack001	Empty Field	Empty Field	Registering with empty password field	RegisterAccountButton.onClick()	"Username or Password field is empty!"	"Username or Password field is empty!"	Pass
Reg03	Empty Field	leow10	leow10	Registering with empty username field	RegisterAccountButton.onClick()	"Username or Password field is empty!"	"Username or Password field is empty!"	Pass
Reg04	Existing Username	12abc98	12abc98	Registering with an existing username	RegisterAccountButton.onClick()	"This username had already been used"	"Username already exist!"	Pass
Reg05	jonas22	70n45	J0nas	Different input for password and confirm password	RegisterAccountButton.onClick()	"Password and Confirm Password are different!"	"Password and Confirm Password are different!"	Pass
Reg06	pew00	zxcv	zxcv	Registering with new username & password	RegisterAccountButton.onClick()	Avatar Selection Page	Avatar Selection Page	Pass
Reg07	User click on Water Avatar			Select choice of Avatar	public class AvatarSelect : MonoBehaviour {}	Load Login Scene	Load Login Scene	Pass

2.3 Student Mode Test

2.3.1 Story Mode

Test Case ID	Actions	Description	Functions to be tested	Expected Output	Actual Output	Pass/Fail
Story Mode						
Story01	User selected Story Mode on Menu page	User wants to play in Story Mode	StoryModeButton.onClick.AddListener (()	The World scene will be displayed to the User	The World scene will be displayed to the User	Pass
Story02	User select a 'Software engineering basics and requirement engineering' World	User wants to play 'Software engineering basics and requirement engineering' World	public class WorldSelect : MonoBehaviour{}	Section scene will be displayed to the User	Section scene will be displayed to the User	Pass
Story03	User select 'Software Engineering basics' Section	User wants to play 'Software Engineering basics' Section	public class SectionSelect : MonoBehaviour{}	Level Scene will be displayed to the User	Level Scene will be displayed to the User	Pass
Story04	User select Level 1	User wants to play level 1 and quiz will be started	public class LevelSelect : MonoBehaviour{} & public class QuestionGenerator : MonoBehaviour{}	Quiz scene will be displayed	Quiz scene will be displayed	Pass
Story05	User select an answer and click 'Next Question'	User answered the first question correctly. and Click 'Next Question'	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Green colour, add 1 points to user's score & display next quiz question	Displayed Green colour, add 1 points to user's score & display next quiz question	Pass
Story06	User select an answer and click 'Next Question'	User answered the second question correctly	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Green colour, add 1 points to user's score & display next quiz question	Displayed Green colour, add 1 points to user's score & display next quiz question	Pass
Story07	User select an answer and click 'Next Question'	User answered the third question wrongly	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Red colour & display next quiz question	Displayed Red colour & display next quiz question	Pass
Story08	User select an answer and click 'Next'	User answered the last question wrongly	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Red colour & display Quiz Ended Scene	Displayed Red colour & display Quiz Ended Scene	Pass
Story09	User will see the score for his/her quiz	Display the score of the Quiz user had completed.	public class QuizEnded : MonoBehaviour{}	Display quiz ended with "Your Score: 2"	Displayed quiz ended with "Your Score: 2"	Pass
Story10	User click "Proceed to game"	User plays the wants to start playing the game	public class GameManager : MonoBehavior{}	MainScene will be displayed	MainScene is displayed	Pass
Story11	User click on $100 weapon and place it on the map.	User wants to buy a weapon and place it on map.	public class Shop : MonoBehaviour{}	User money will be deduct by $100 and a weapon is placed on the map.	User money will be deduct by $100 and a weapon is placed on the map.	Pass
Story12	User clicks "Extra Questions"	User wants to answer more questions to get more money and buy weapon.	public void StartQuestions()	Load Scene Quiz	Loaded Scene Quiz	Pass
Story13	User select an answer and click 'Next Question'	User answered the pop up quiz correctly	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Green colour, add 1 points to user's score & display next quiz question	Display Green colour, add 1 points to user's score & display next quiz question	Pass
Story14	User click 'Next Question'	User wants to go back to game	public void OnSelectProceed()	MainScene will be displayed	MainScene is displayed	Pass
Story15	User lives finishes and game end	User finish playing the game	public class GameOver : MonoBehavior{}	Game End scene will be displayed	Game End scene is displayed	Pass
Story16	User click 'Menu'	User wants to return to Main Menu	public void Menu()	Display Scene 'Main Menu'	Displayed Scene 'Main Menu'	Pass

2.3.2 PVP

PVP Mode						
PVP01	User click 'PVP'	User wants to play against another player in PVP mode	PVPButton.onClick.AddListener()	Load Scene 'PVP Select'	Loaded Scene 'PVP Select'	Pass
PVP02	User click 'Create Room'	User creates a room	OnSelectCreateRoom()	Create Room scene will be displayed to the User	Create room scene is displayed to the user	Pass
PVP03	User select World 1, Section 1 and Level 1 and click 'Create Game'	User selects the topic that he wants to play in	public class WorldSelect : MonoBehavior{} public class SectionSelect : MonoBehavior{} public class LevelSelect : MonoBehavior{} public class PvpGetAccessCode : MonoBehavior{}	Display random Access Code "5706"	Displayed random Access Code "5706"	Pass
PVP04	User selects 'Back to Menu'	User goes back to the menu for PVP	public class OnSelectMenu()	Load Scene 'PVP Select', displays PVP menu	Loaded Scene 'PVP Select', displays PVP menu	Pass
PVP05	User click 'Join Room'	User wants to join the room Created	public void OnSelectJoinRoom()	Load Scene 'PVP Join Room', displays PVP waiting room	Loaded Scene 'PVP Join Room', displays PVP waiting room	Pass
PVP06	User enter Access Code "5706" and click 'Join Room'	User wants to go into the room he created	public class PVPJoinRoom : MonoBehavior{}	Load Scene 'Quiz'	Loaded Scene 'Quiz'	Pass
PVP07	User select an answer and click 'Next Question'	User answered the first question correctly. and Click 'Next Question'	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Green colour, add 1 points to user's score & display next quiz question	Displayed Green colour, add 1 points to user's score & display next quiz question	Pass
PVP08	User select an answer and click 'Next Question'	User answered the second question correctly	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Green colour, add 1 points to user's score & display next quiz question	Displayed Green colour, add 1 points to user's score & display next quiz question	Pass
PVP09	User select an answer and click 'Next Question'	User answered the third question wrongly	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Red colour & display next quiz question	Displayed Red colour & display next quiz question	Pass
PVP10	User select an answer and click 'Next'	User answered the last question wrongly	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Red colour & display Quiz Ended Scene	Displayed Red colour & display Quiz Ended Scene	Pass
PVP11	User will see the score for his/her quiz	Display the score of the Quiz user had completed.	public class QuizEnded : MonoBehaviour{}	Display quiz ended with "Your Score: 2"	Displayed quiz ended with "Your Score: 2"	Pass
PVP12	User click "Proceed to game"	User plays the wants to start playing the game	public class GameManager : MonoBehavior{}	MainScene will be displayed	MainScene is displayed	Pass
PVP13	User click on $100 weapon and place it on the map.	User wants to buy a weapon and place it on map.	public class Shop : MonoBehaviour{}	User money will be deduct by $100 and a weapon is placed on the map.	User money will is deducted by $100 and a weapon is placed on the map.	Pass
PVP14	User clicks "Extra Questions"	User wants to answer more questions to get more money and buy weapon.	public void StartQuestions()	Load Scene Quiz	Load Scene Quiz	Pass
PVP15	User select an answer and click 'Next Question'	User answered the pop up quiz correctly	bool CheckAnswer(int selectedAnswer) & OnNextQuestion()	Display Green colour, add 1 points to user's score & display next quiz question	Display Green colour, add 1 points to user's score & display next quiz question	Pass
PVP16	User click 'Next Question'	User wants to go back to game	public void OnSelectProceed()	MainScene will be displayed	MainScene is displayed	Pass
PVP17	User lives finishes and game end	User finish playing the game	IEnumerator EndPvp(string APIUrl){}	Load Scene 'PVP Game Ended', which displays "The winner is witeow"	Display 'PVP Game Ended', which displays "The winner is witeow"	Pass
PVP18	User click 'Main Menu'	User wants to return to Main Menu	public void OnSelectMainMenu()	Load Scene 'Main Menu'	Load Scene 'Main Menu'	Pass

2.3.3 Leader board

Leaderboard						
Leaderboard01	User clicks on 'Leaderboard'	User wants to view Leaderboard	public class LeaderboardGenerator : MonoBehavior{}	Leaderboard scene will be loaded and displayed	Leaderboard scene is loaded and displayed	Pass
Leaderboard02	User clicks on 'Main Menu'	User wants to go back to Main Menu	public void OnSelectMainMenu()	Main Menu scene will be loaded and displayed	Main Menu scene is be loaded and displayed	Pass

2.3.4 Assignment

Leaderboard						
Leaderboard01	User clicks on 'Leaderboard'	User wants to view Leaderboard	public class LeaderboardGenerator : MonoBehavior{}	Leaderboard scene will be loaded and displayed	Leaderboard scene is loaded and displayed	Pass
Leaderboard02	User clicks on 'Main Menu'	User wants to go back to Main Menu	public void OnSelectMainMenu()	Main Menu scene will be loaded and displayed	Main Menu scene is be loaded and displayed	Pass

2.4 Teacher Mode Test

2.4.1 View Reports and Analytics

Test Case ID	Actions	Description	Functions to be tested	Expected Output	Actual Output	Pass/Fail
Teacher01	Teacher click "Quiz Report"	Teacher wants to view the quiz report.	QuizReportButton.onClick.AddListener()	Load Scene Quiz Report Generation	Load Scene Quiz Report Generation	Pass
Teacher02	Teacher click "Back"	Teacher wants to go back to the Menu.	BackButon.onClick.AddListener()	Load Scene Teacher Menu	Load Scene Teacher Menu	Pass
Teacher03	Teacher click "Assignment Report"	Teacher wants to view the assignment report.	ReportGenerationButton.onClick.AddListener()	Load Scene Assignment Report	Load Scene Assignment Report	Pass
Teacher04	Teacher select Access Code "21" from the drop-down box	Teacher wants to view a particular assignment report with the access code 21.	public void ChangeAccessCode()	Display correct fields	Display correct fields	Pass
Teacher05	Teacher click "Back"	Teacher wants to go back to the Menu.	BackButon.onClick.AddListener()	Load Scene Teacher Menu	Load Scene Teacher Menu	Pass
Teacher06	Teacher click "Logout"	Teacher wants to logout	LogoutButton.onClick.AddListener()	Load Scene Login Scene	Load Scene Login Scene	Pass

2.4.2 Login to Assignment System

Test Case ID	Input		Description	Functions to be tested	Expected Output	Actual Output	Pass/Fail
	Username	Password					
Login01	admin	EMPTY FIELD	No input password.	LoginCheck()	"Username or Password field is empty!"	"Username or Password field is empty!"	Pass
Login02	EMPTY FIELD	1234	No input username.	LoginCheck()	"Username or Password field is empty!"	"Username or Password field is empty!"	Pass
Login03	admin	1234	Wrong password.	IEnumerator LoginAPI(string URL)	"Wrong Username or Password!"	"Wrong Username or Password!"	Pass
Login04	adm1n	pass	Wrong username.	IEnumerator LoginAPI(string URL)	"Wrong Username or Password!"	"Wrong Username or Password!"	Pass
Login05	admin	pass	Correct username and password.	IEnumerator LoginAPI(string URL)	Load Scene AssignmentSettings	Load Scene AssignmentSettings	Pass

2.4.3 Create Assignment

2.4.3.1 Table 1

CreateAss12	Click "Import Question"	User wants to import question from database for question 3.	ImportQuestionButton.onClick.AddListener(DisplayImportDatabaseDropdownpanel)	Display Import Database Dropdown Panel	Display Import Database Dropdown Panel	Pass
CreateAss13	Select "Software engineering basics and requirement engineering" for topic, "Requirement Elicitation Techniques" for Sub Topic and "1" for Difficulty. Click "Find Question"	User choose the topic and difficulty level for question 3.	GenerateSecondRequest() public void DisplayImportedQuestionPreviewPanel()	Load imported questions and their description from database	Load imported questions and their description from database	Pass
CreateAss14	Click "Question 2"	User choose question in database for assignment question 3.	public void PreviewQuestions(List<QuestionTemplate> questionBank)	Display the particular question and description from database	Display the particular question and description from database	Pass
CreateAss15	Click "Save as question"	User save the question to the assignment question 3.	public void saveAsQuestion()	Return to Scene MainAssignment	Return to Scene MainAssignment	Pass
CreateAss16	Click "Create Assignment"	User fills up all the field and wants to create assignment.	createAssignmentButton.onClick.AddListener(CreateAssignment);	Load Scene SocialMedia	Load Scene SocialMedia	Pass
CreateAss17	Click "Log in with Facebook"	User wants to post the assignment on Facebook.	FBLoginButton.onClick.AddListener(FBLogin) FBPostButton.onClick.AddListener(FBShare)	Open Facebook Tab	Open Facebook Tab	Pass
CreateAss18	Click Twitter icon.	User wants to post the assignment on Twitter.	twitterButton.onClick.AddListener(TwitterTweet)	Open Twitter Tab	Open Twitter Tab	Pass
CreateAss19	Click "Main Menu"	User wants to return to Main Menu.	MainMenu.OnClick.AddListener(BackToUI	Load Scene AssignmentSettings	Load Scene AssignmentSettings	Pass
CreateAss20	Input "1" for Number of Questions, "55" for Access Code & Click Continue	User sets the configuration to add on questions to a previously existing assignment	ContinueButton.onClick.AddListener(CreateQuestion)	Load Scene MainAssignment	Load Scene MainAssignment	Pass
CreateAss21	Click "Import Question"	User wants to import question from database	ImportQuestionButton.onClick.AddListener(DisplayImportDatabaseDropdownpanel)	Display Import Database Dropdown Panel	Display Import Database Dropdown Panel	Pass
CreateAss22	Select "Software engineering basics and requirement engineering" for topic, "Requirement Elicitation Techniques" for Sub Topic and "1" for Difficulty. Click "Find Question"	User choose the topic and difficulty level for questio.	GenerateFirstRequest() public void DisplayImportedQuestionPreviewPanel()	Load imported questions and their description from database	Load imported questions and their description from database	Pass
CreateAss23	Click "Question 1"	User select first question from database.	public void PreviewQuestions(List<QuestionTemplate> questionBank)	Display the particular question and description from database	Display the particular question and description from database	Pass
CreateAss24	Click "Create Assignment"	User fills up all the field and wants to create assignment with more question	createAssignmentButton.onClick.AddListener(CreateAssignment);	An error message is displayed as there are duplicates of question from the previously posted assignment	An error message is displayed as there are duplicates of question from the previously posted assignment	Pass
CreateAss25	Click "Logout"	User wants to logout.	LogoutButton.onClick.AddListener(LogoutFunction)	Load Scene TeacherLogin	Load Scene Teacher Login	Pass

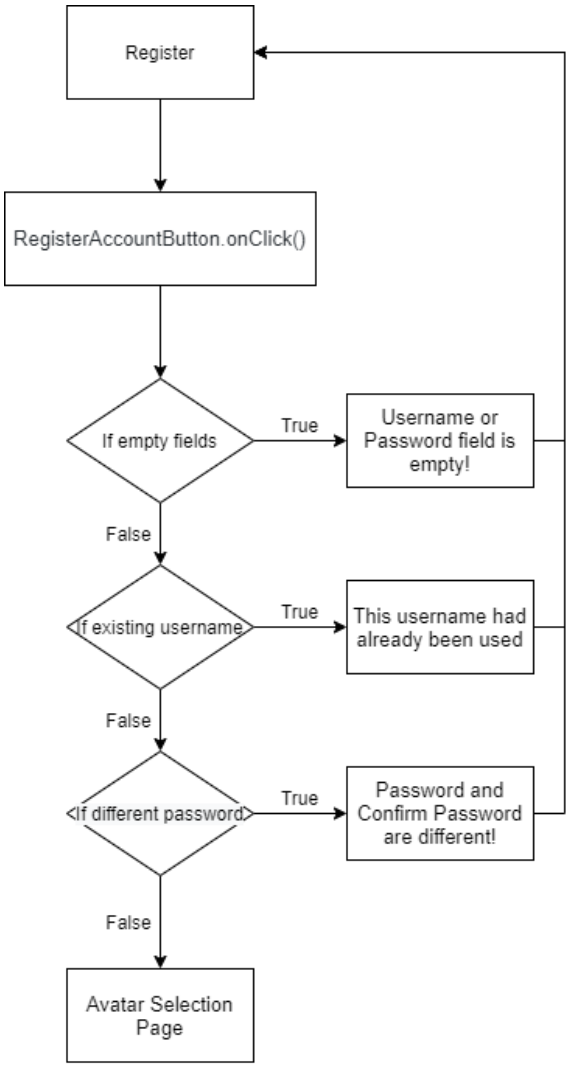
2.4.3.2 Table 2

Test Case ID	Actions	Description	Functions to be tested	Expected Output	Actual Output	Pass/Fail
CreateAss01	Input "3" for Number of Questions only & Click Continue	User only input number of questions without access code and wants to create assignment.	ContinueButton.onClick.AddListener (CreateQuestion)	"Number of Questions or Access Code field needs to be numeric only!	"Number of Questions or Access Code field needs to be numeric only!	Pass
CreateAss02	Input "55" for Access Code only & Click Continue	User only input access code without the number of questions and wants to create assignment.	ContinueButton.onClick.AddListener (CreateQuestion)	"Number of Questions or Access Code field needs to be numeric only!	"Number of Questions or Access Code field needs to be numeric only!	Pass
CreateAss03	Input "3" for Number of Questions, "55" for Access Code & Click Continue	User set the configurations for a new assignment with number of questions and access code.	ContinueButton.onClick.AddListener (CreateQuestion)	Load Scene MainAssignment	Load Scene MainAssignment	Pass
CreateAss04	Click "Create Assignment	User wants to create an assignment straight away without settings the question.	public string QuestionString(int questionNo){}	Display Error Message.	Display Error Message.	Pass
CreateAss05	Click "Question 1"	User wants to configure question 1.	public void MainAssignmentPanelSetup(){}	Display field for Question Title, Answer and Option 1 to Option 4	Display field for Question Title, Answer and Option 1 to Option 4	Pass
CreateAss06	Input "Which course does CZ2006 represents?" into Question Title. Input "Software Engineering" into Answer. Input "Advance Software Engineering" into Option 1. Input "Software System Architecture Design" into Option 2. Input "Algorithm" into Option 3. Input "Software Engineering" into Option 4. click "Question 2"	User manually type in the question and answer for first question.	public void MainAssignmentPanelSetup(){}	Display field for Question Title, Answer and Option 1 to Option 4	Display field for Question Title, Answer and Option 1 to Option 4	Pass
CreateAss07	Click "Import Question"	User wants to import question from database for question 2.	ImportQuestionButton.onClick.AddListener(DisplayImportDatabaseDropdownpanel)	Display Import Database Dropdown Panel	Display Import Database Dropdown Panel	Pass
CreateAss08	Select "Software engineering basics and requirement engineering" for topic, "Requirement Elicitation Techniques" for Sub Topic and "1" for Difficulty. Click "Find Question"	User choose the topic and difficulty level for question 2.	GenerateFirstRequest() public void DisplayImportedQuestionPreviewPanel()	Load imported questions and their description from database	Load imported questions and their description from database	Pass
CreateAss09	Click "Question 1"	User select first question from database.	public void PreviewQuestions(List<QuestionTemplate> questionBank)	Display the particular question and description from database	Display the particular question and description from database	Pass
CreateAss10	Click "Save as question"	User save the question to the assignment question 2.	public void saveAsQuestion()	Return to Scene MainAssignment	Return to Scene MainAssignment	Pass
CreateAss11	Click "Question 3"	User wants to configure question 3.	public void MainAssignmentPanelSetup(){}	Display field for Question Title, Answer and Option 1 to Option 4	Display field for Question Title, Answer and Option 1 to Option 4	Pass

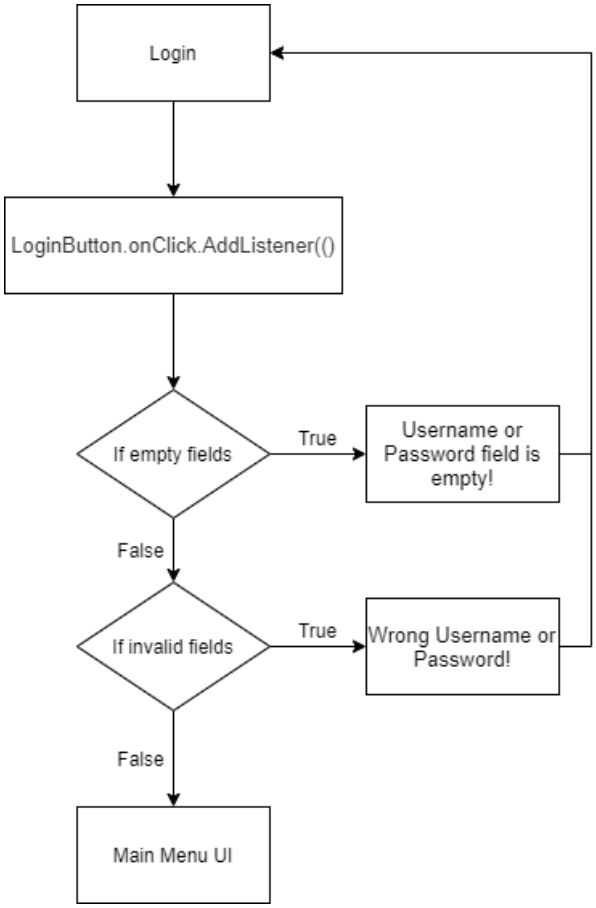
3. White Box Report

White Box Testing enables us to find flaws in the logic of the system. We are able to efficiently realise the parts that have errors or problems. There must be a limitation set for user inputs, such that only those inputs that we are looking for are accepted, while other irrelevant inputs will be rejected accordingly. Any empty or invalid field entered will result in an error message to prompt the user for relevant inputs.

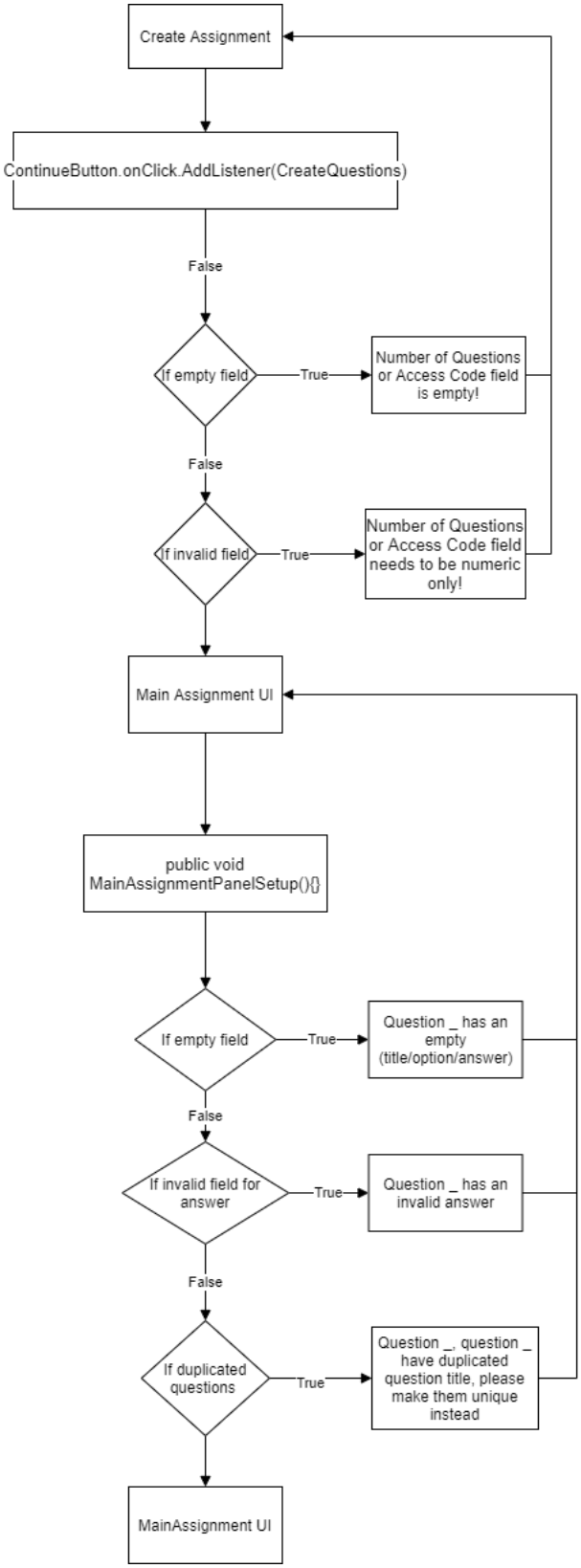
3.1 Account Creation



3.2 Account Login



3.3 Assignment Creation



4. Unity Unit Testing

The development team had decided to handle most of the logic in the backend, as within Unity, there are many interdependencies within scenes and scripts. This allowed us to minimise the potential errors we would have to face when developing the game in Unity.
Therefore, we have implemented unit tests to ensure that core functionalities are not broken when the program is updated and when the different subsystems are integrated together.

All of our classes in the unity program are inherited from the Monobehaviour class. The Monobehaviour class does not allow for instantiation and thus we were limited in our capabilities.

One possible workaround to this issue is to create an interface for each class and subsequently mock the interfaces using NSubstitute (https://technicallyshane.com/2019/10/29/unit-testing-unity.html). Since our other test methods have sufficiently tested our program, we have decided not to pursue this method.

Subsystem	Number of tests	Pass rate
Quiz	2	100%
Leaderboard	1	100%
Game	2	100%

5. Load Test

5.1 User Capacity Analysis

The User Capacity Analysis calculates how many users the application can support based on the configured performance goals.

Estimated User Capacity : Between 100 to 150
Maximum Users Analyzed : 300

Performance Testing Range: Between 50 to 300

5.2 Preamble

	Functions Tested	**API URL Used**	**Justification for choosing this function**
1	Start-up tester	https://223.25.69.254:10002/	The Start-up tester function behaves as the ideal API call our server can service as it has next to no internal logic, which can help show us the performance ceiling we can expect with the current server configuration.
2	Retrieve Leaderboard	https://223.25.69.254:10002/get_leaderboard/username=SHAFIQ002	This function will act as a proxy for any average API Call that seeks information from the server and backend, with no data being written.
3	Update Quiz Performance	https://223.25.69.254:10002/update_performance/username=SHAFIQ002&world=1&section=1&no_of_correct=3	This function will act as a proxy for any average API Call that seeks and writes information from the server and backend, with no data being written. This API will help us identify the performance floor we can expect with the current server configuration.

While all the tests were conducted together, note that any tests conducted with users = 300 were the equivalent of a stress test. Thus, both load and stress tests were conducted during the testing phase of deploying this application.

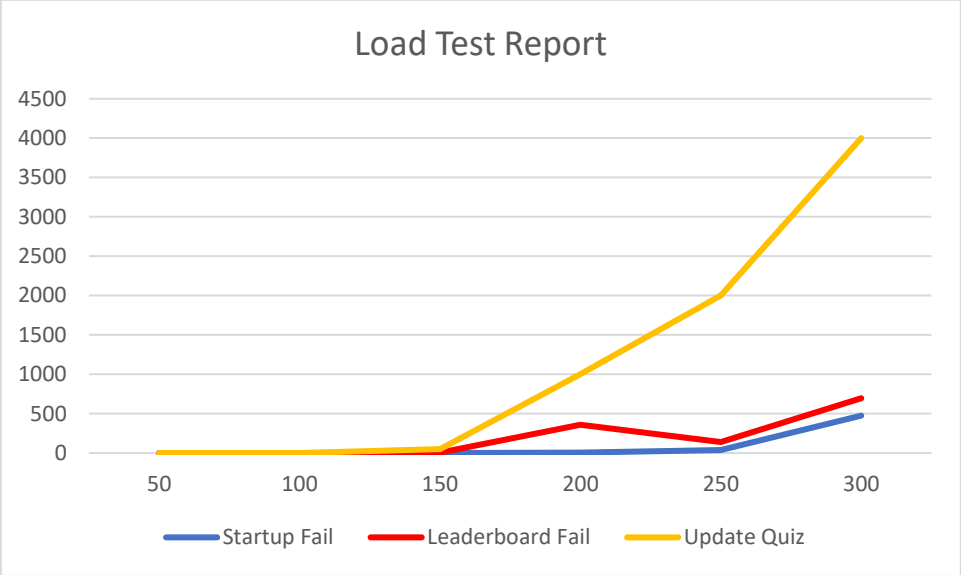
5.3 Test Results

Start-Up Tester								
Max Users	**# of Requests**	**# of Fails**	**Median**	**90%ile**	**Mean**	**Min**	**Max**	**Failures/s**
50	11949	0	950	1200	961	16	1235	0
100	38991	0	1900	1900	1532	16	2313	0
150	25121	0	3300	5200	3519	203	2313	0
200	13462	5	2500	9100	3801	31	21032	0
250	11957	36	2500	9300	4554	31	21079	0.2
300	14379	475	3600	10000	5821	125	21109	3.9
Leaderboard								
Max Users	**# of Requests**	**# of Fails**	**Median**	**90%ile**	**Mean**	**Min**	**Max**	**Failures/s**
50	12644	0	1200	1200	1159	141	1563	0
100	13296	0	2300	2400	2294	156	3328	0
150	15774	4	3000	5500	3462	15	21062	0
200	13001	360	3800	5900	4611	31	32734	0.7
250	14136	141	4100	9800	5620	31	21110	1.4
300	13102	696	5700	10000	6322	16	21094	6
Update Quiz Performance								
Max Users	**# of Requests**	**# of Fails**	**Median**	**90%ile**	**Mean**	**Min**	**Max**	**Failures/s**
50	11204	0	2000	2100	2050	2000	2110	0
100	23120	0	2000	2100	2050	2000	2110	0
150	11619	48	2000	2100	2048	2000	2125	0.1
200	5000	NULL	NULL	NULL	NULL	NUL	NUL	NULL

						L	L	
250	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
300	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

NOTE: A NULL cell refers to a test case that led to unstable server behavior that made testing inaccurate and too difficult to conduct.

5.4 Visualizations



NOTE: NULL Values are presented as exponentially increasing fault cases, instead of being omitted from visualization