NANYANG TECHNOLOGICAL UNIVERSITY

CZ3003 - Software System Analysis & Design

Lab 3 Deliverables

Project Name: Food Wars

Group Name: Team 1

Lab group: TDDP3

Date of Submission: 17 October 2021

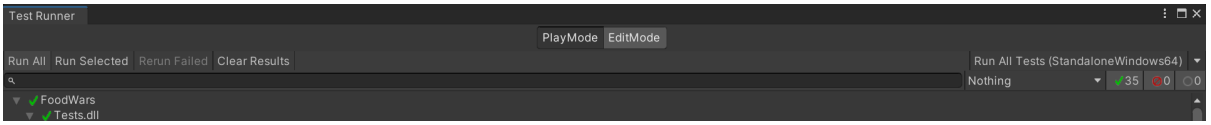
Group Member	Matric No.
David Tay Ang Peng	U1910603L
Grace Ong Yong Han	U1721575H
Jordon Kho Junyang	U1920297F
Lim Wei Rong	U1921791D
Ryan Tan Yu Xiang	U1922774F
Joy Cheng Zhaoyi	U1922716L
Tang Hoong Jing	U1721417E
Guo Wan Yao	U1822530E
Ng Wee Hau, Zaphyr	U1822044D
Lee Kai Jie, John	U1921862J
Chio Ting Kiat	U1720465K

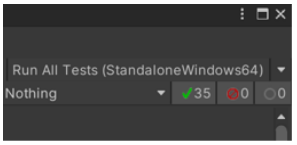
Table Of Content

Front-End Testing

Summary

For our testing, we have used the automated testing package, “UnityEngine.TestTools”, to conduct the tests on our various components. A total of 33 test cases were conducted for all the various components and all of them have passed. Two of the test cases were complementary test suite cases which run when the Unity Test Tools run.





We will go into the details for each component’s testing in the document below.

Sample Script

The following is a sample snippet from our testing scripts which is testing for whether the leaderboard is arranged in descending order by elo:

```
[UnityTest]
// Comment: Leaderboard check that all 5 ranks in rows are truthy
public IEnumerator leaderboard_content_check_correct_all_rows_have_rank()
{
    SceneManager.LoadScene("LeaderboardScene");
    yield return new WaitForSeconds(3);

    Text rankOneRank = GameObject.Find("Row1").GetComponent<RowUi>().rank;
    Text rankTwoRank = GameObject.Find("Row2").GetComponent<RowUi>().rank;
    Text rankThreeRank = GameObject.Find("Row3").GetComponent<RowUi>().rank;
    Text rankFourRank = GameObject.Find("Row4").GetComponent<RowUi>().rank;
    Text rankFiveRank = GameObject.Find("Row5").GetComponent<RowUi>().rank;

    Assert.IsTrue(rankOneRank);
    Assert.IsTrue(rankTwoRank);
    Assert.IsTrue(rankThreeRank);
    Assert.IsTrue(rankFourRank);
    Assert.IsTrue(rankFiveRank);
}
```

Test Cases

Test#	Component	Test Name	Expected Result	Status
1	Leaderboard	leaderboard_rows_check_arranged_by_descending_elo	Users are arranged by highest score at first to lowest score at fifth	Passed
2	Leaderboard	view_season_rewards_button_on_click_success	View season rewards button can be clicked successfully	Passed
3	Leaderboard	whatsapp_logo_check_truthy	There is a whatsapp share icon	Passed
4	Leaderboard	telegram_logo_check_truthy	There is a telegram share icon	Passed
5	Leaderboard	leaderboard_header_my_rank_check_correct_text	Leaderboard should have a 'my rank' header	Passed
6	Leaderboard	leaderboard_header_my_name_check_correct_text	Leaderboard should have a 'my name' header	Passed
7	Leaderboard	leaderboard_header_my_elo_check_correct_text	Leaderboard should have a 'my elo' header	Passed
8	Leaderboard	leaderboard_content_check_correct_five_rows	Leaderboard should have 5 rows	Passed
9	Leaderboard	leaderboard_content_check_correct_all_rows_have_name	Leaderboard should have a name that is non-empty in each row	Passed
10	Leaderboard	leaderboard_content_check_correct_all_rows_have_elo	Leaderboard should have an elo that is non-empty in each row	Passed
11	Leaderboard	leaderboard_content_check_correct_all_rows_have_rank	Leaderboard should have a rank that is non-empty in each row	Passed
12	Register	all_ui_components_check_truthy	Presence of the username, email, password and level selection fields	Passed
13	Register	username_input_on_text_entered_changed	Username field reflects what the user has input correctly	Passed
14	Register	email_input_on_text_entered_changed	Email field reflects what the user has input correctly	Passed
15	Register	password_input_on_text_entered_changed	Password field reflects what the user has input correctly	Passed
16	Register	no_input_on_register_return_warning	System prompts user for username when they click login without entering any input	Passed
17	Register	no_username_on_register_return_warning	System prompts user for username when they click login without entering the username	Passed
18	Register	no_password_on_register_return_warning	System prompts user for password when they click login without entering the password but have entered username	Passed
19	Register	no_email_on_register_return_warning	System prompts user for email when they click register without entering the email but have entered username and password	Passed
20	Register	correct_information_register_success	When a valid set of username, email and password is entered and the user clicks register, the user is successfully registered	Passed
21	Register	duplicate_email_register_returns_warning	System returns a warning when user tries to register with an email already in the database	Passed
22	Login	Valid_Input_LoginButton_Onclick_Warning_Text	When the user provides valid username and password and clicks login, they are successfully logged in	Passed
23	Login	MissingEmail_Input_LoginButton_Onclick_Warning_Text	System prompts user for email when user clicks login without entering an email	Passed
24	Login	MissingPassword_LoginButton_Onclick_Warning_Text	System prompts user for password when user clicks login without entering a password but have entered an email	Passed
25	Login	WrongPassword_LoginButton_Onclick_Warning_Text	System prompts user about wrong password when user clicks login with an invalid set of email and password	Passed
26	Login	InvalidEmail_LoginButton_Onclick_Warning_Text	System prompts user about invalid email when they click login with an email not in our database	Passed
27	Login	UserNotFound_LoginButton_Onclick_Warning_Text	System prompts user about invalid username when they click login with an username not in our database	Passed

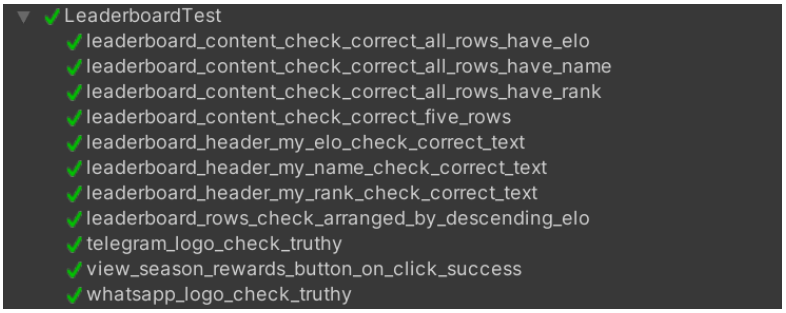
28	Single	skip_question_on_click_increase_correct_question	Test the functionality of skip question power-up. When the user clicks on this button, the number of correct questions count increases by one.	Passed
29	Single	replay_on_click_reset_points	Test the functionality of replay. When the user clicks on the replay button, the points are reset to zero.	Passed
30	Single	answer_on_click_increase_questions_seen	Test the functionality of answer buttons. When the user clicks on the answer button, the count of questions seen increases by one.	Passed
31	Inventory	display_items_on_start_only_display_owned_items	The number of displayed inventory slots matches the number of items the user owns	Passed
32	Shop	display_powerups_on_start_display_all_powerups	The number of displayed powerups matches the number of powerups in the database	Passed
33	Shop	activate_accessory_panel_on_click_load_accessory_page	The accessory page loads when 'Next' button is clicked	Passed

*** Passed means that the result of running the test is consistent with the expected result**

Detailed Test Descriptions – Leaderboard Testing

For the leaderboard component, the following tests were conducted:

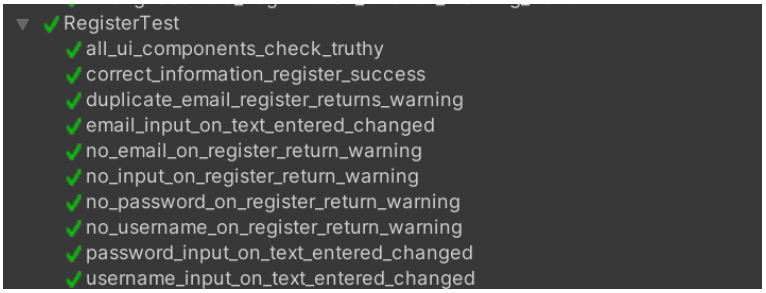
- There were five rows of data in the leaderboard
- The users in the leaderboard were arranged in descending order by their elo
- All five rows of data were correct
- The table headers were all correct
- The sharing buttons for WhatsApp and Telegram were both present
- View Season Rewards Button was functioning on click



Detailed Test Descriptions – Register Testing

For the registration component, the following tests were conducted:

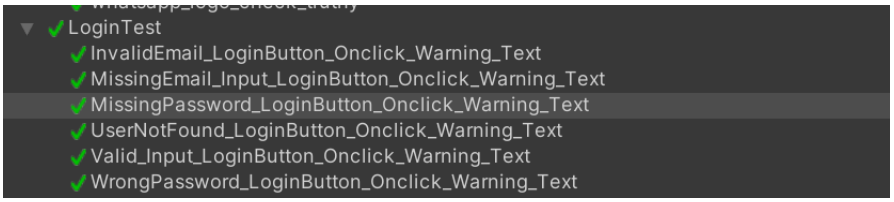
- All UI components, meaning the input boxes, dropdown and buttons, were functioning
- Missing information, be it username, email or password, prompted the user to enter them and registration fails
- The input boxes were containing the correct information after the information is entered
- Usage of an email already in the database would cause registration to fail due to duplicate emails
- Registration is successful if all information are provided and valid



Detailed Test Descriptions – Login Testing

For the login component, the following tests were conducted:

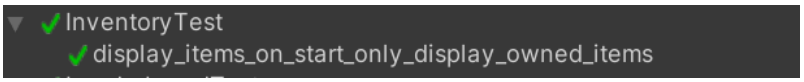
- Missing information would cause login to fail and a warning message would appear to prompt the user for the missing information
- If the user entered is not in the database, a warning message is shown to inform that the user credentials does not exist
- If the wrong password is entered meaning it does not match the database's for the user, a warning message is shown to inform that the user credentials entered is invalid
- Login is successful if valid credentials are entered



Detailed Test Descriptions – Inventory Testing

For the inventory component, the following tests were conducted:

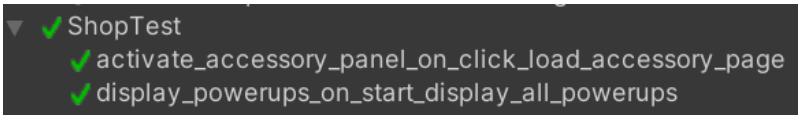
- Only items that are owned by the users are displayed on the inventory screen



Detailed Test Descriptions – Shop Test

For the shop component, the following test were conducted:

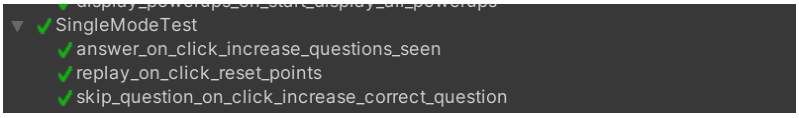
- The powerups were displayed in the shop
- The accessory page is displayed when the user clicks on the tab to access the accessory page



Detailed Test Descriptions – Single Mode Test

For the single player mode component, the following tests were conducted:

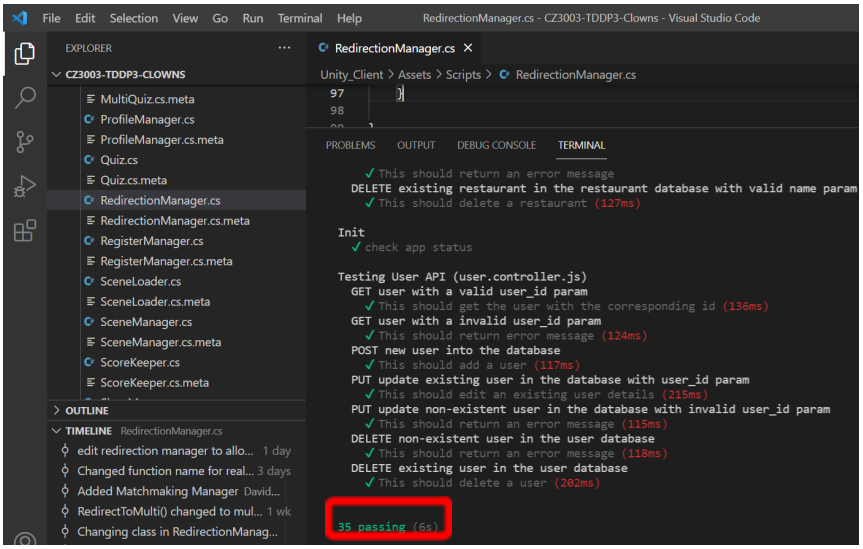
- When a question is answered, the questions seen count increases as another question is shown
- When replay is selected, the points reset to zero
- When skip question is selected, the number of correct questions increases by one



Back-End Testing

Testing Summary

For our Blackbox testing, our team has adopted the mocha and chai package. A total of 35 test cases spanning all our database collections were conducted for all the various components and all of them have passed.



Sample Script

The following is a sample snippet from our testing scripts which is testing for whether the character object retrieved from the server has the necessary properties:

```
//Test GET
describe('GET character with id', () => {
    it("This should get a character", (done) => {
        chai.request(server)
        .get("/character")
        .query({id: trial_id})
        .end((err, res) => {
            expect(res).to.have.status(200);
            expect(res).to.be.json;
            expect(res.body).to.have.property('characterName');
            expect(res.body).to.have.property('characterDescription');
            expect(res.body).to.have.property('characterSprite');
            expect(res.body).to.have.property('characterID').eq(trial_id);
        done();
        })
    })
})
```

Test Cases

Test#	Component	Test Name	Expected Result	Status
1	Character	GET character with id	A character object is returned from the database	Passed
2	Character	GET character with missing id param	An error message for missing id param	Passed
3	Character	POST new character	Adds a character to database	Passed
4	Character	DELETE character that does not exist	Nothing is deleted and an error message is returned	Passed
5	Character	DELETE existing character in the database	Delete an existing character	Passed
6	Item	GET Shop powerup with 2 valid params	Get all powerups in shop	Passed
7	Item	GET leaderboard accessories with 2 valid params	Get all accessories on the leaderboard	Passed
8	Item	GET shop powerup with missing itemType param	An error message for missing itemType parameter	Passed
9	Item	GET shop powerup with missing itemSource param	An error message for missing itemSource parameter	Passed
10	Item	POST new item into the shop	An item is added into the shop	Passed
11	Item	DELETE item that does not exist	No items are deleted and an error message is returned	Passed
12	Item	DELETE existing item in the items database	An existing item is deleted from database	Passed
13	Question	GET questionList with valid param	Returns all questions for the specified primaryLevel	Passed
14	Question	GET questionList with invalid param	An error message regarding invalid parameters is returned	Passed
15	Question	GET questionList with missing param	An error message regarding missing parameters is returned	Passed
16	Question	GET questionList with invalid param	An error message regarding invalid parameters is returned	Passed
17	Question	POST new question into the database	A question is added into database	Passed
18	Question	DELETE non-existent question in the questions database with invalid qn param	No questions are deleted and an error message regarding invalid parameters is returned	Passed
19	Question	DELETE question in the questions database with missing qn param	No questions are deleted and an error message regarding missing parameters is returned	Passed
20	Question	DELETE existing question in the questions database with valid qn param	An existing question is deleted from the database	Passed
21	Restaurant	GET Restaurant with valid param	All dishes for the specified Restaurant are returned	Passed
22	Restaurant	GET Restaurant with invalid name param	An error message regarding invalid name is returned	Passed
23	Restaurant	GET Restaurant with missing name param	An error message regarding missing name is returned	Passed
24	Restaurant	POST new restaurant dish into the database	New restaurant dish is added to the database	Passed
25	Restaurant	DELETE non-existent restaurant in the restaurant database with invalid name param	No restaurants are deleted and an error message regarding invalid name is returned	Passed
26	Restaurant	DELETE restaurant in the restaurant database with missing name param	No restaurants are deleted and an error message regarding missing name is returned	Passed
27	Restaurant	DELETE existing restaurant in the restaurant database with valid name param	A restaurant object is deleted from the database	Passed
28	Init	Check app status	The app is without error	Passed
29	User	GET user with a valid user_id param	User object with the corresponding id is returned	Passed

30	User	GET user with a invalid user_id param	An error message regarding invalid user_id is returned	Passed
31	User	POST new user into the database	A new user object is added into the database	Passed
32	User	PUT update existing user in the database with user_id param	The existing user object is updated in the database	Passed
33	User	PUT update non-existent user in the database with invalid user_id param	No user object is updated and an error message regarding invalid user_id is returned	Passed
34	User	DELETE non-existent user in the user database	No user is deleted and an error is returned	Passed
35	User	DELETE existing user in the user database	User object is deleted	Passed

*** Passed means that the result of running the test is consistent with the expected result**
**** Full report can be found in "CZ3003-TDDP3-Clowns\Nodejs_Server\test\test_output"**

We ordered and implemented the tests based on each of the CRUD (create, read, update and delete) operations required for our application from each of our database collections, and the details are described below as follows:

Detailed Test Results – Testing Character API

```
Testing Character API (character.controller.js)
  GET character with id
    ✓ This should get a character (1241ms)
  GET character with missing id param
    ✓ This should get an error message for missing id param
  POST new character
    ✓ This should add a character (78ms)
  DELETE character that does not exist
    ✓ This should not delete anything and return an error message (469ms)
  DELETE existing character in the database
    ✓ This should delete an existing character (138ms)
```

For our character database, we implemented the following tests:

1. Extracting a character from the database using a valid character id (GET)
2. Removing an existing character from the database using a valid character id (DELETE)
3. Adding a new character into the database by inputting the correct body format (POST)
4. Error test cases for each of the 2 CRUD operations above, where an invalid id is used. Our predefined error message will be displayed

Detailed Test Results – Testing Item API

```
Testing Item API (item.controller.js)
  GET Shop powerup with 2 valid params
    ✓ This should get all powerups in shop (327ms)
  GET leaderboard accessories with 2 valid params
    ✓ This should get all accessories on the leaderboard (190ms)
  GET shop powerup with missing itemType param
    ✓ This should get an error message for missing itemType param
  GET shop powerup with missing itemSource param
    ✓ This should get an error message for missing itemSource param
  POST new item into the shop
    ✓ This should add a item into the shop
  DELETE item that does not exist
    ✓ This should not delete any items and return an error message (120ms)
  DELETE existing item in the items database
    ✓ This should delete an existing item (595ms)
```

For our item database, we implemented the following tests:

1. Extracting all items from the database which fulfills the 2 query conditions “itemSource” and “itemType” (GET)
2. Adding a new item into the database by inputting the correct body format (POST)
3. Deleting an existing item from the database using a valid id (DELETE)
4. Error test cases for each of the CRUD operations above, where:
    a. test is called with missing parameters.
    b. an invalid id is used.

    Our predefined error messages will be displayed accordingly.

Detailed Test Results – Testing Question API

```
Testing Question API (question.controller.js)
  GET questionList with valid param
    ✓ This should get all questions for the specified primaryLevel (127ms)
  GET questionList with invalid param
    ✓ This should return an error message
  GET questionList with invalid param
    ✓ This should return an error message
  GET questionList with missing param
    ✓ This should return an error message
  POST new question into the database
    ✓ This should add a question into the database
  DELETE non-existent question in the questions database with invalid qn param
    ✓ This should return an error message (167ms)
  DELETE question in the questions database with missing qn param
    ✓ This should return an error message
  DELETE existing question in the questions database with valid qn param
    ✓ This should delete an existing question (106ms)
```

For our question database, we implemented the following tests:

1. Extracting all question from the database which fulfills the query condition "primaryLevel" (GET)
2. Adding a new question into the database by inputting the correct body format (POST)
3. Deleting an existing question from the database using a valid id (DELETE)
4. Error test cases for each of the CRUD operations above, where:
    a. test is called with missing parameters.
    b. invalid parameters are used.
    c. an invalid id is used.

    Our predefined error messages will be displayed accordingly.

Detailed Test Results – Testing Restaurant API

```
Testing Restaurant API (restaurant.controller.js)
  GET Restaurant with valid param
    ✓ This should get all dishes for the specified Restaurant (205ms)
  GET Restaurant with invalid name param
    ✓ This should return an error message (186ms)
  GET Restaurant with missing name param
    ✓ This should return an error message
  POST new restaurant dish into the database
    ✓ This should add a restaurant dish into the database (184ms)
  DELETE non-existent restaurant in the restaurant database with invalid name param
    ✓ This should return an error message (114ms)
  DELETE restaurant in the restaurant database with missing name param
    ✓ This should return an error message
  DELETE existing restaurant in the restaurant database with valid name param
    ✓ This should delete a restaurant (127ms)
```

For our restaurant database, we implemented the following tests:

1. Extracting all dishes from the database which fulfills the query condition restaurant "name" (GET)
2. Adding a new restaurant dish into the database by inputting the correct body format (POST)
3. Deleting an existing restaurant from the database using a valid restaurant "name" (DELETE)
4. Error test cases for each of the CRUD operations above, where:
    d. test is called with missing parameters.
    e. invalid parameters are used.
    f. an invalid name is used.

Our predefined error messages will be displayed accordingly.

Detailed Test Results – Testing User API

```
Init
  ✓ check app status

Testing User API (user.controller.js)
  GET user with a valid user_id param
    ✓ This should get the user with the corresponding id (136ms)
  GET user with a invalid user_id param
    ✓ This should return error message (124ms)
  POST new user into the database
    ✓ This should add a user (117ms)
  PUT update existing user in the database with user_id param
    ✓ This should edit an existing user details (215ms)
  PUT update non-existent user in the database with invalid user_id param
    ✓ This should return an error message (115ms)
  DELETE non-existent user in the user database
    ✓ This should return an error message (118ms)
  DELETE existing user in the user database
    ✓ This should delete a user (202ms)
```

For our user database, we implemented the following tests:

1. Extracting the user from the database which fulfills the query condition user id (GET)
2. Adding a new user into the database by inputting the correct body format (POST)
3. Deleting an existing user from the database using a valid user id (DELETE)
4. Update records of an existing user in the database using a valid user id (PUT)
5. Error test cases for each of the CRUD operations above, where:
    g. invalid parameters are used.
    h. an invalid user id is used.

    Our predefined error messages will be displayed accordingly.