



NANYANG
TECHNOLOGICAL
UNIVERSITY

CZ3003 - Software System Analysis & Design

Lab 3 Deliverables

Project Name: Food Wars

Group Name: Team 1

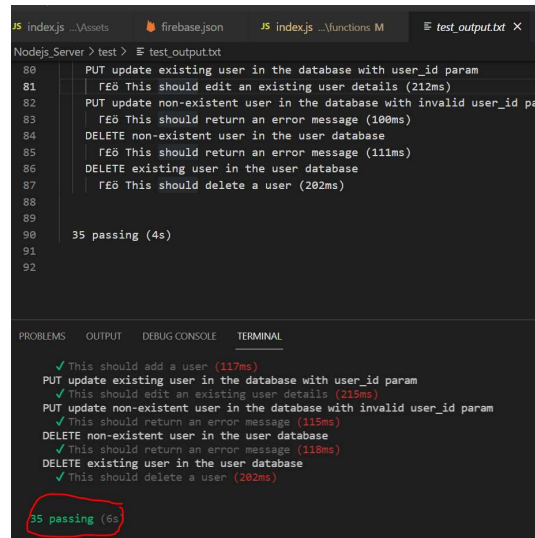
Lab group: TDDP3

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

Testing Summary

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



```
index.js ...Assets  firebase.json  JS index.js ...Functions M  test_output.txt X
Nodejs_Server > test > test_output.txt
80 PUT update existing user in the database with user_id param
81   ✓ This should edit an existing user details (212ms)
82 PUT update non-existent user in the database with invalid user_id param
83   ✓ This should return an error message (100ms)
84 DELETE non-existent user in the user database
85   ✓ This should return an error message (111ms)
86 DELETE existing user in the user database
87   ✓ This should delete a user (202ms)
88
89
90 35 passing (4s)
91
92
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
✓ This should add a user (117ms)
✓ This should edit an existing user details (212ms)
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 (110ms)
DELETE existing user in the user database
✓ This should delete a user (202ms)
35 passing (6s)
```

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 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();
      })
  })
})
```

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)
```

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)
```

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 (191ms)
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 get all questions for the specified primaryLevel
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 (163ms)
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 (171ms)
```

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)
```

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)
```

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

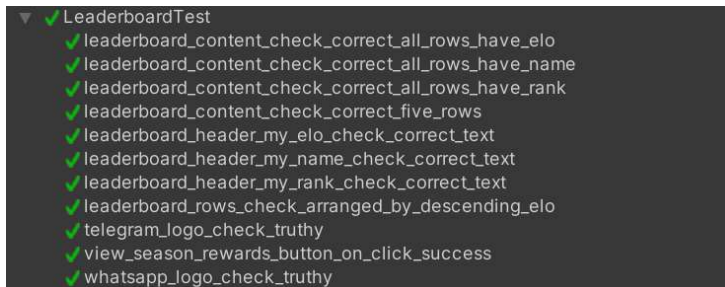
29	User	GET user with a valid user_id param	User object with the corresponding id is returned	Passed
30	User	GET user with a invalid user_id param	An error message regarding invalid user_id is returned	Passed
31	User	POST new user into the database	A new user object is added into the database	Passed
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**

Leaderboard Testing

For the leaderboard component, the following tests were conducted:

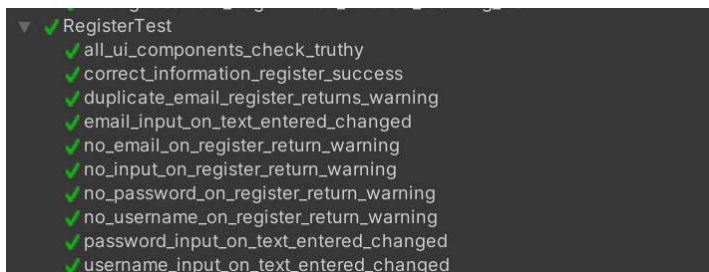
- There were five rows of data in the leaderboard
- The users in the leaderboard were arranged in descending order by their elo
- All five rows of data were correct
- The table headers were all correct
- The sharing buttons for WhatsApp and Telegram were both present
- View Season Rewards Button was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



Register Testing

For the registration component, the following tests were conducted:

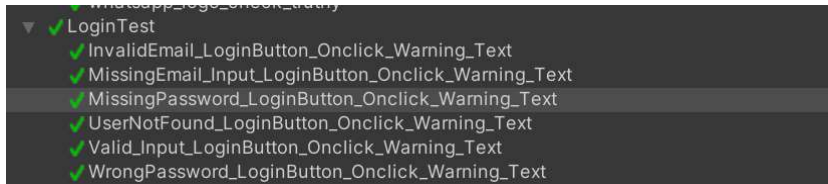
- All UI components, meaning the input boxes, dropdown and buttons, were functioning
- Missing information, be it username, email or password, prompted the user to enter them and registration fails
- The input boxes were containing the correct information after the information is entered
- Usage of an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



Login Testing

For the login component, the following tests were conducted:

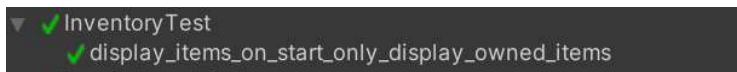
- Missing information would cause login to fail and a warning message would appear to prompt the user for the missing information
- If the user entered is not in the database, a warning message is shown to inform that the user credentials does not exist
- If the wrong password is entered meaning it does 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



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



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



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

