Beaconnet

Testing Document

Initial design

6-May, 2023

Group D1 - CIBAY

Lam Kin Ho 1155158095

Lin Sze Po 1155176854

Cheng Ka Chi 1155175548

Chiu Wing Tung Crystal 1155174499

Wong Hoi Yin 1155176263

Dept. of Computer Science and Engineering of The Chinese University of Hong Kong

Contents

1 Test Plan **1**

2 Test Report **2**

3 Test Cases **3**

3.1 Case - Health check . . . 3

3.1.1 Purpose . . . 3

3.1.2 Input . . . 3

3.1.3 Expected Outputs & Pass/Fail Criteria . . . 3

3.2 Case - Login[UI interface] . . . 4

3.2.1 Purpose . . . 4

3.2.2 Input . . . 4

3.2.3 Expected Outputs & Pass/Fail Criteria . . . 4

3.3 Case - Send Bits [UI interface] . . . 6

3.3.1 Purpose . . . 6

3.3.2 Input . . . 6

3.3.3 Expected Outputs & Pass/Fail Criteria . . . 6

3.4 Case - ReBit [UI interface] . . . 7

3.4.1 Purpose . . . 7

3.4.2 Input . . . 7

3.4.3 Expected Outputs & Pass/Fail Criteria . . . 7

3.5 Case - Comment Bit [UI interface] . . . 8

3.5.1 Purpose . . . 8

3.5.2 Input . . . 8

3.5.3 Expected Outputs & Pass/Fail Criteria . . . 8

3.6 Case - User Search . . . 9

3.6.1 Input . . . 9

3.6.2 Expected Outputs & Pass/Fail Criteria . . . 9

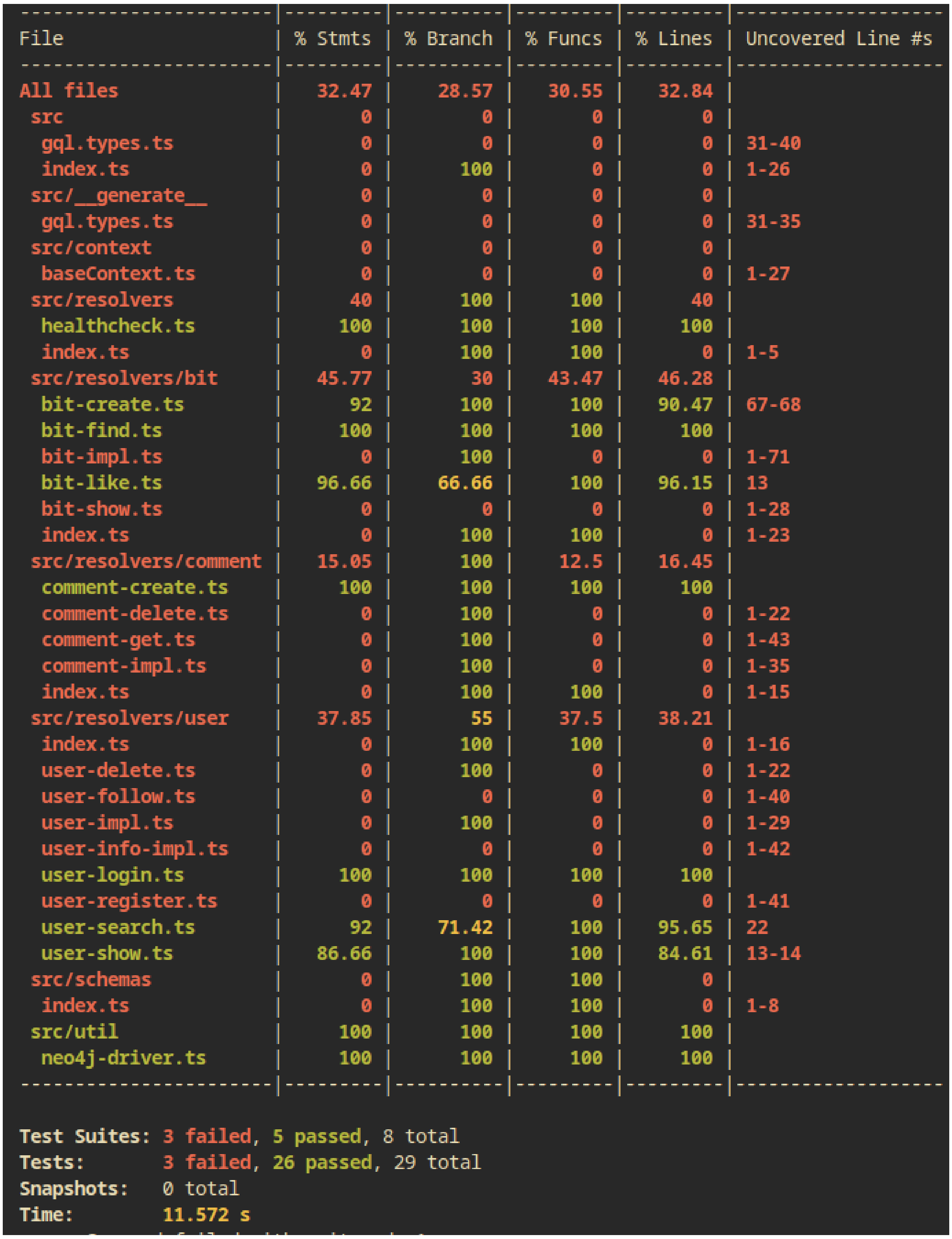
1 Test Plan

We are going to use Jest(https://jestjs.io/) to evaluate our backend and import another database driver separately from the main server. And generate the code coverage report for the testing. The testing environment is separate from the development and production environment. We plan to set some unit tests for each module(Resolver for Graphql) and integration tests to simulate the whole usage process in our program. Such testing progress should also integrate into our CI pipeline, which is conducted automatically before committing/PR to prevent breaking the origin functionality.

After such testing phases are finished, there is a UAT environment to test our product before actually going to the production. Such UAT environment can let us have real-user to actually test our product while also having a more stable production environment for most users.

For the front-end part, we plan to introduce Storybook(https://storybook.js.org/) to test the UI component in our future roadmap.

2 Test Report



```
------------------------|---------|----------|---------|---------|-------------------
File                    | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
------------------------|---------|----------|---------|---------|-------------------
All files               |   32.47 |    28.57 |   30.55 |   32.84 |
 src                    |       0 |        0 |       0 |       0 |
  gql.types.ts          |       0 |        0 |       0 |       0 | 31-40
  index.ts              |       0 |      100 |       0 |       0 | 1-26
 src/__generate__       |       0 |        0 |       0 |       0 |
  gql.types.ts          |       0 |        0 |       0 |       0 | 31-35
 src/context            |       0 |        0 |       0 |       0 |
  baseContext.ts        |       0 |        0 |       0 |       0 | 1-27
 src/resolvers          |      40 |      100 |     100 |      40 |
  healthcheck.ts        |     100 |      100 |     100 |     100 |
  index.ts              |       0 |      100 |     100 |       0 | 1-5
 src/resolvers/bit      |   45.77 |       30 |   43.47 |   46.28 |
  bit-create.ts         |      92 |      100 |     100 |   90.47 | 67-68
  bit-find.ts           |     100 |      100 |     100 |     100 |
  bit-impl.ts           |       0 |      100 |       0 |       0 | 1-71
  bit-like.ts           |   96.66 |    66.66 |     100 |   96.15 | 13
  bit-show.ts           |       0 |        0 |       0 |       0 | 1-28
  index.ts              |       0 |      100 |     100 |       0 | 1-23
 src/resolvers/comment  |   15.05 |      100 |    12.5 |   16.45 |
  comment-create.ts     |     100 |      100 |     100 |     100 |
  comment-delete.ts     |       0 |      100 |       0 |       0 | 1-22
  comment-get.ts        |       0 |      100 |       0 |       0 | 1-43
  comment-impl.ts       |       0 |      100 |       0 |       0 | 1-35
  index.ts              |       0 |      100 |     100 |       0 | 1-15
 src/resolvers/user     |   37.85 |       55 |    37.5 |   38.21 |
  index.ts              |       0 |      100 |     100 |       0 | 1-16
  user-delete.ts        |       0 |      100 |       0 |       0 | 1-22
  user-follow.ts        |       0 |        0 |       0 |       0 | 1-40
  user-impl.ts          |       0 |      100 |       0 |       0 | 1-29
  user-info-impl.ts     |       0 |        0 |       0 |       0 | 1-42
  user-login.ts         |     100 |      100 |     100 |     100 |
  user-register.ts      |       0 |        0 |       0 |       0 | 1-41
  user-search.ts        |      92 |    71.42 |     100 |   95.65 | 22
  user-show.ts          |   86.66 |      100 |     100 |   84.61 | 13-14
 src/schemas            |       0 |      100 |     100 |       0 |
  index.ts              |       0 |      100 |     100 |       0 | 1-8
 src/util               |     100 |      100 |     100 |     100 |
  neo4j-driver.ts       |     100 |      100 |     100 |     100 |
------------------------|---------|----------|---------|---------|-------------------

Test Suites: 3 failed, 5 passed, 8 total
Tests:       3 failed, 26 passed, 29 total
Snapshots:   0 total
Time:        11.572 s
```

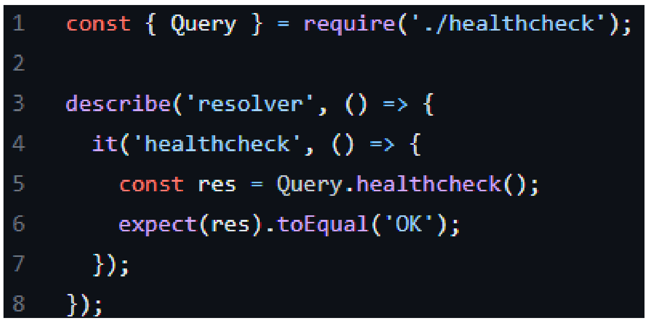
3 Test Cases

3.1 Case - Health check

3.1.1 Purpose

The main purpose of this testing is to check the health of the Beaconnect.

3.1.2 Input



```
const { Query } = require('./healthcheck');

describe('resolver', () => {
  it('healthcheck', () => {
    const res = Query.healthcheck();
    expect(res).toEqual('OK');
  });
});
```

3.1.3 Expected Outputs & Pass/Fail Criteria

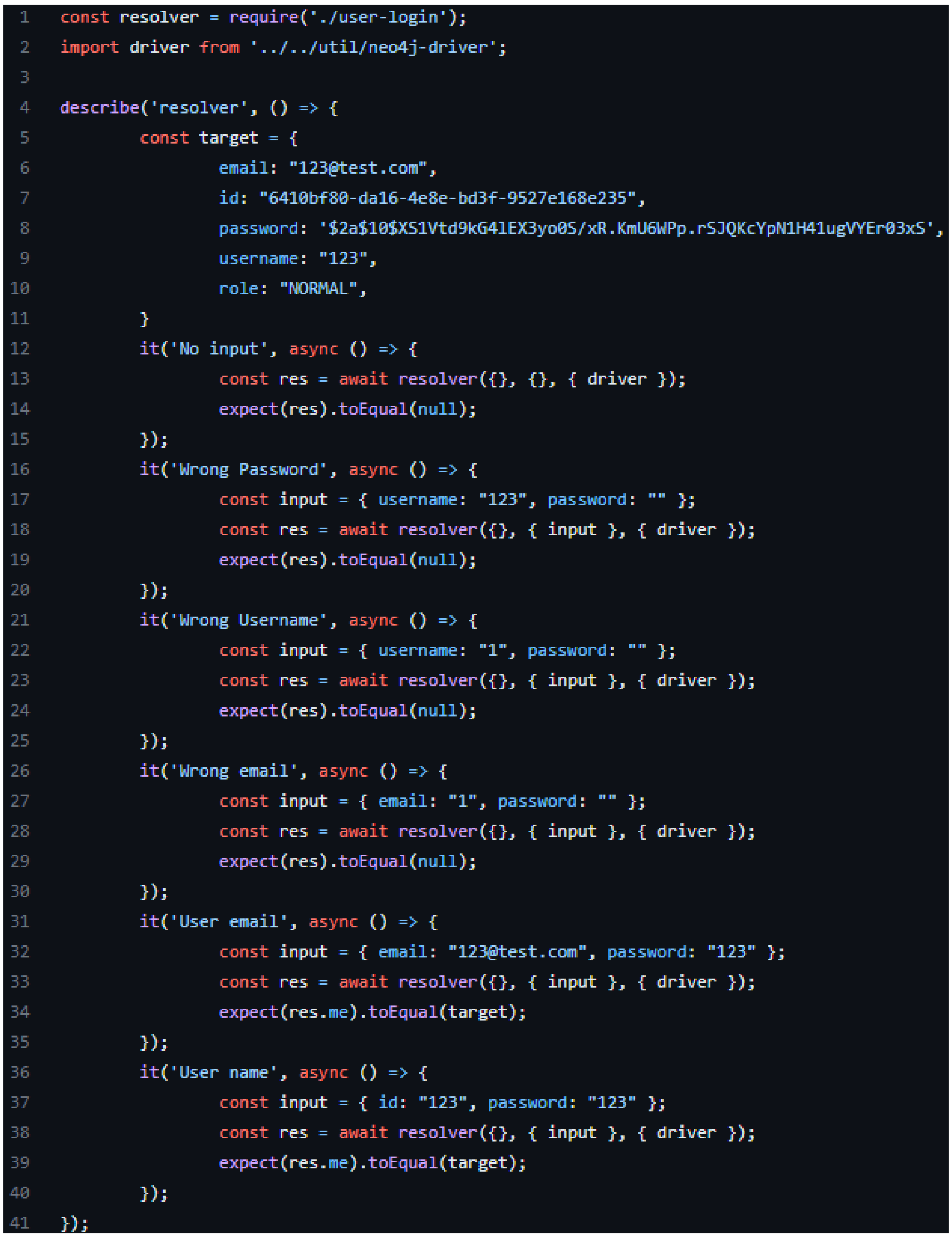
The expected output is ”OK” as it use to check the return function of healthcheck.
The passing criteria should be return a string of ”OK”.
The fail criteria would be return other string expect ”OK” and no return.

3.2 Case - Login[UI interface]

3.2.1 Purpose

The main purpose of this testing is to check the login system of Beaconnect, such that it should check whether the email and the password have been used. Also, check whether the email and the password follow the required format. Second, check using Username to login.

3.2.2 Input



```
const resolver = require('./user-login');
import driver from '../../util/neo4j-driver';

describe('resolver', () => {
        const target = {
                email: "123@test.com",
                id: "6410bf80-da16-4e8e-bd3f-9527e168e235",
                password: '$2a$10$XS1Vtd9kG41EX3yo0S/xR.KmU6WPp.rSJQKcYpN1H41ugVYEr03xS',
                username: "123",
                role: "NORMAL",
        }
        it('No input', async () => {
                const res = await resolver({}, {}, { driver });
                expect(res).toEqual(null);
        });
        it('Wrong Password', async () => {
                const input = { username: "123", password: "" };
                const res = await resolver({}, { input }, { driver });
                expect(res).toEqual(null);
        });
        it('Wrong Username', async () => {
                const input = { username: "1", password: "" };
                const res = await resolver({}, { input }, { driver });
                expect(res).toEqual(null);
        });
        it('Wrong email', async () => {
                const input = { email: "1", password: "" };
                const res = await resolver({}, { input }, { driver });
                expect(res).toEqual(null);
        });
        it('User email', async () => {
                const input = { email: "123@test.com", password: "123" };
                const res = await resolver({}, { input }, { driver });
                expect(res.me).toEqual(target);
        });
        it('User name', async () => {
                const input = { id: "123", password: "123" };
                const res = await resolver({}, { input }, { driver });
                expect(res.me).toEqual(target);
        });
});
```

3.2.3 Expected Outputs & Pass/Fail Criteria

Test case 1 : [No input]
The expected output is ”fill out this field” for this test case as not input for this test case, user should fill

up the input field. Therefore the passing criteria is unable to login and show a test of ”fill out this field”.

Test case 2 : [Wrong password]
The expected output is showing ”Wrong Input” and the user is not able to login BeaConnect. For the passing criteria, BeaConnect should show the expected output, like ”Wrong Input”, and other case shown would be the fail criteria.

Test case 3 : [Wrong username]
The expected output is showing ”Wrong Input” and the user is not able to login BeaConnect with wrong username. For the passing criteria, BeaConnect should show the expected output, like ”Wrong Input”, and other case shown would be the fail criteria.

Test case 4 : [Wrong Email]
The expected output is showing ”Wrong Input” and the user is not able to login BeaConnect with wrong email. For the passing criteria, BeaConnect should show the expected output, like ”Wrong Input”, and other case shown would be the fail criteria.

Test case 5 : [MatchingUser email]
The expected output is login successful, user is able to login to BeaConnect with the user email, and go to the home page of BeaConnect. Therefore, the passing criteria is go to the home page of BeaConnect with login. And, the fail criteria including go to the home page without login, fail to go to the home page or unsuccess login.

Test case 5 : [User email]
The expected output is login successful, user is able to login to BeaConnect with uesr email, and go to the home page of BeaConnect. Therefore, the passing criteria is go to the home page of BeaConnect with login. And, the fail criteria including go to the home page without login, fail to go to the home page or unsuccess login.

3.3 Case - Send Bits [UI interface]

3.3.1 Purpose

The main purpose of this testing is to check the sendBits function, including function disable when user is not login, Bit's text show is complectly same as the input and send Bits with picture is available.

3.3.2 Input

Test case 1 : [User is not login]
Input : There is not input in this test case.

Test case 2 : [Sending text Bits]
Input : "Hello world"

Test case 3 : [Sending bits with picture]
Input : "This is a horse."
Picture :

3.3.3 Expected Outputs & Pass/Fail Criteria

Test case 1 : [User is not login]
The expected output is the sendBits function is disable. For the passing criteria is the sendBits function not shown in the interface to user and user cannot send Bits.

Test case 2 : [Sending text Bits]
The expected output is the Bit's text will shown up under the sendBits function. The passing criteria including the Bit text, user icon, username, user ID and time of sendBits.

Test case 3 : [Sending bits with picture]
The expected output is the Bit's text will shown up under the sendBits function, same as the Sending test Bits but with the picture. The passing criteria including the Bit text, picture, user icon, username, user ID and sendBits time.

3.4 Case - ReBit [UI interface]

3.4.1 Purpose

The main purpose of this testing is to check the ReBits function, it almost do the same function with sendBits. The ReBit's should show ReBits' tect and the ReBits' Bits.

3.4.2 Input

Test case 1 : [ReBits with text]
Input : "Test ReBits" and click the ReBits buttom

Test case 2 : [ReBits without text]
Input : No text input and click the ReBits buttom

3.4.3 Expected Outputs & Pass/Fail Criteria

Test case 1 : [ReBits with text]
The expected output should show the ReBits' bits and the ReBits' text, including basic user information, and the ReBit's bit text and the Bit's text. Therefore the passing criteria should be showing correct user informaion with ReBits' user and the ReBits' Bits user, also with the ReBits' text and ReBits' Bits' text correctly. Other case occurs would be the fail criteria, such as user information is missing and Bits' text is missing.

Test case 2 : [ReBits without text]
The expected output would be fail ReBits, as there is not input for the ReBits. Therefore fail ReBits will be the passing criteria and success ReBits will be the fail criteria.

3.5 Case - Comment Bit [UI interface]

3.5.1 Purpose

The main purpose of this testing is to check the Comment Bit function, including showing comment under the Bit with text.

3.5.2 Input

Test case 1 : [Comment with text]
Input : ”Testing comment”

Test case 2 : [Comment without text]
Input : no input

3.5.3 Expected Outputs & Pass/Fail Criteria

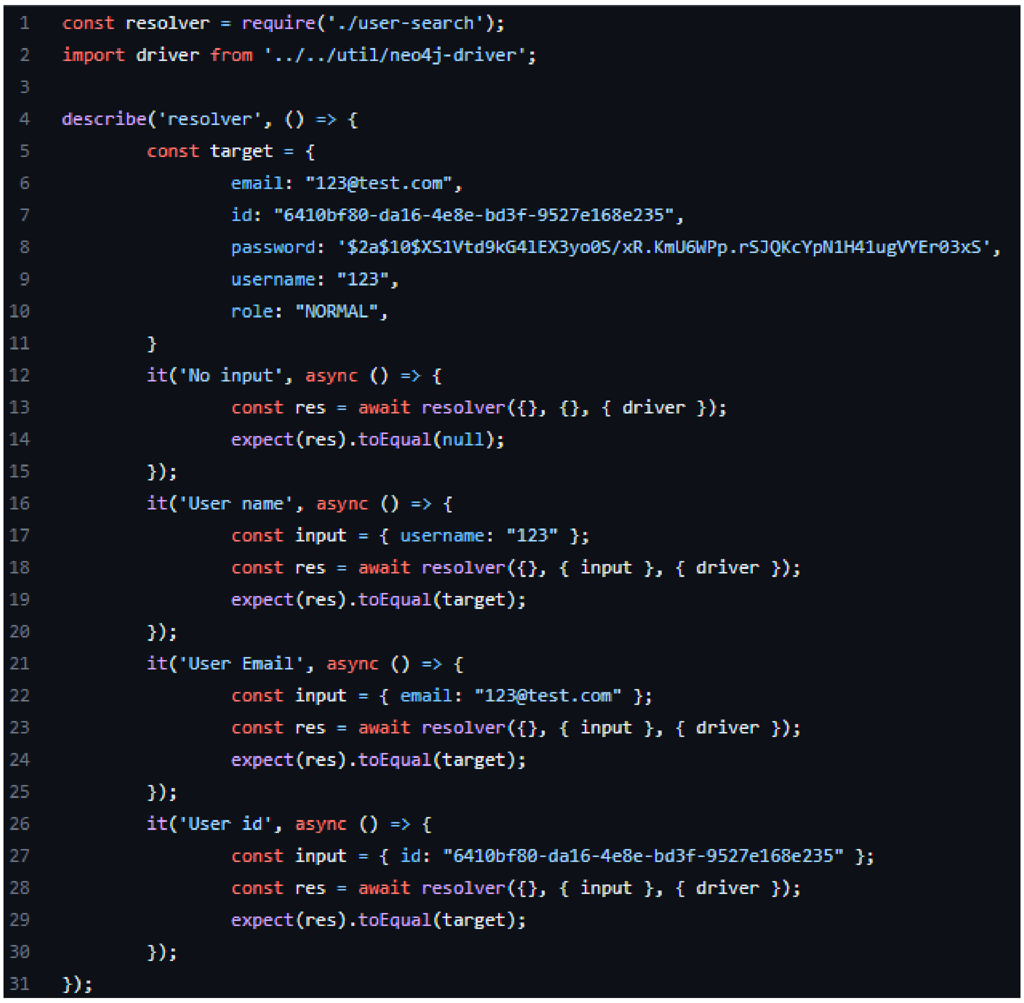
Test case 1 : [Comment with text]
The expected output should be the commented text will show under the commented Bit, with comment user information and the comment text. If there are comment under the Bits, the new comment should show under the previous comment. Therefore, the passing criteria should be showing the comment correctly and the comments should show according to the comment time. And, the fail criteria would be not comment show under the Bit and wrong order of the comments.

Test case 2 : [Comment without text]
The expected output should be no comment show under the Bits, as there is not text input to the comment box. Therefore the passing criteria is unsuccess comment, and not comment is shown. The fail criteria should be comment made under the Bits and other cases.

3.6 Case - User Search

The purpose of the checking the user search function is the ensure the search function is workable, including able to search the user with username, user ID and user email.

3.6.1 Input



```javascript
const resolver = require('./user-search');
import driver from '../../util/neo4j-driver';

describe('resolver', () => {
        const target = {
                email: "123@test.com",
                id: "6410bf80-da16-4e8e-bd3f-9527e168e235",
                password: '$2a$10$XS1Vtd9kG41EX3yo0S/xR.KmU6WPp.rSJQKcYpN1H41ugVYEr03xS',
                username: "123",
                role: "NORMAL",
        }
        it('No input', async () => {
                const res = await resolver({}, {}, { driver });
                expect(res).toEqual(null);
        });
        it('User name', async () => {
                const input = { username: "123" };
                const res = await resolver({}, { input }, { driver });
                expect(res).toEqual(target);
        });
        it('User Email', async () => {
                const input = { email: "123@test.com" };
                const res = await resolver({}, { input }, { driver });
                expect(res).toEqual(target);
        });
        it('User id', async () => {
                const input = { id: "6410bf80-da16-4e8e-bd3f-9527e168e235" };
                const res = await resolver({}, { input }, { driver });
                expect(res).toEqual(target);
        });
});
```

3.6.2 Expected Outputs & Pass/Fail Criteria

Test case 1 : [No input]
The expected output is ”Null” for this test case as not input for this test case, user should fill up the input field. Therefore the passing criteria is showing the same UI as origin to the user.

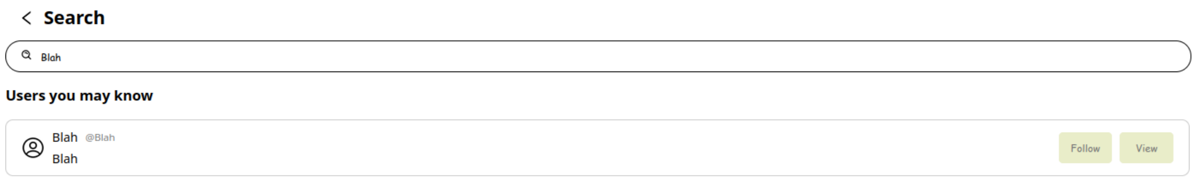
Test case 2 : [Wrong username]
The expected output is showing ”No User Found” and the user is not able to search the user. For the passing criteria, BeaConnect should show the expected output, the passing criteria is a window showing ”No User Found”.

Test case 3 : [Wrong user id]

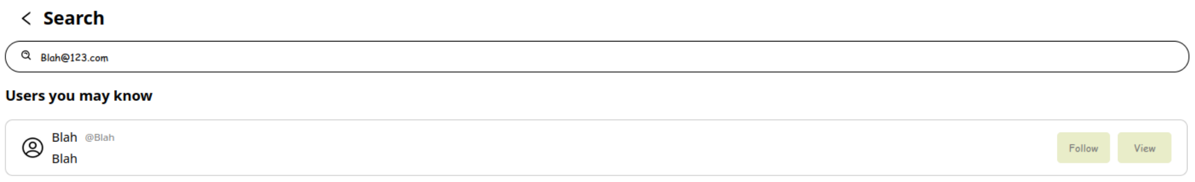
Same as wrong username.

Test case 4 : [Wrong Email]
Same as wrong username.

Test case 5 : [User name]



The expected output is the user with username ”Blah” with matching user name, And user can click follow or view by the button at the rightmost corner. The passing criteria is showing the required user in the UI. Test case 5 : [User Email]



Basically same with matching user name, the input changes from user name to user email.