SW Engineering CSC648/848 Milestone 5

Project title: **Momentus**
Team 4
Matt Bolles: Team Lead
Raza Ali: Github Master
Natalie Christie: Scrum Master
Jo Ey Chong: Front End Lead
Eric Chie: Back End Lead
Adam Belaid: Back End

05/20/2021

Version History

Milestone/Version	Date
M5V1	05/20/2021

Contents

1 Committed Features

- Login / Register
- Text / Photo Post Creation
- Collections
- User Profiles
- Comments / Favorites

2 Coding Style

Due to lack of experience, our team did not designate a strict coding style, as we felt that would be a scenario of "reinventing the wheel". Instead, we followed and stuck to established paradigms - whether for JavaScript, React, or any other technology or framework we used. When writing code, if similar code or functions had previously been written by other team members, we made sure to follow that style, unless we had a good reason to change it.

3 Unit Tests

Unit tests are run using Jest, a JavaScript testing tool. Each test checks a major user function.

```
const functions = require("./functions");
const expect = require('expect');
const path = require("path");

test('All User Collection', done => {
    functions.allCollection(2, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        userID: 2
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

test('New Collection', done => {
    functions.newCollection("testlol", 2, function(jason) {
        try {
            expect(jason).toEqual([]);
            done();
        } catch (error) {
            done(error);
        }
    });
});
```

```
test('Append Collection', done => {
    functions.appendCollection(2, 3, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        postID: 3
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

test('View Collection', done => {
    functions.viewCollection(2, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        postID: 7
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

```

```
test('Post By Id', done => {
    functions.postById(7, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        postID: 7
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

test('Post By User', done => {
    functions.postByUser(2, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        userID: 2
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});
```

```
test('Get Profile', done => {
    functions.getProfile(2, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        userID: 2
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

test('Logs in User', done => {
    functions.login("cd", "cd", function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        userID: 2
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});
```

```
test('Favorite', done => {
    functions.favorite(7, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        postID: 7
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

test('Get Favorite Post', done => {
    functions.getFavPost(6, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        postID: 6
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

```

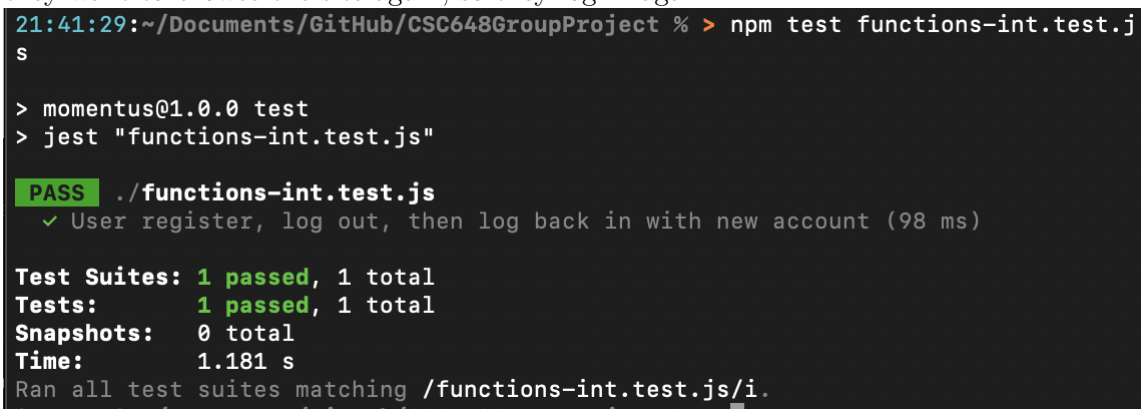
```
test('Get ALL Favorite', done => {
    functions.getAllFav(1, function(jason) {
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        userID: 2
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

test('Registers new User', done => {
    functions.register("test@gmail.com", "testUName", "test", "testFirst",
        try {
            expect(jason).toEqual(
                expect.arrayContaining([
                    expect.objectContaining({
                        username: "testUName"
                    })
                ])
            );
            done();
        } catch (error) {
            done(error);
        }
    });
});

```

```
202
203   test('Creates New Text Post', done => {
204       functions.newPostText("testTextTitle", "testContent", function(jason) {
205           try {
206               expect(jason).toEqual(
207                   expect.arrayContaining([
208                       expect.objectContaining({
209                           title: "testTextTitle",
210                           content: "testContent"
211                       })
212                   ])
213               );
214               done();
215           } catch (error) {
216               done(error);
217           }
218       });
219   });
220
221   test('Creates New Image Post Link', done => {
222       functions.newPostImageLink("testImageTitle", "testCaption", "http://mattrbolles.com/bluecircle.png", function(jason) {
223           try {
224               expect(jason).toEqual(
225                   expect.arrayContaining([
226                       expect.objectContaining({
227                           title: "testImageTitle",
228                           caption: "testCaption"
229                       })
230                   ])
231               );
232               done();
233           } catch (error) {
234               done(error);
235           }
236       });
237   });
238
239
```

```
[20:39:26:~/Documents/GitHub/CSC648GroupProject % > npm test functions.test.js

> momentus@1.0.0 test
> jest "functions.test.js"

 PASS  ./functions.test.js
  ✓ All User Collection (52 ms)
  ✓ New Collection (41 ms)
  ✓ Append Collection (41 ms)
  ✓ View Collection (19 ms)
  ✓ Post By Id (37 ms)
  ✓ Post By User (43 ms)
  ✓ Get Profile (25 ms)
  ✓ Logs in User (30 ms)
  ✓ Favorite (45 ms)
  ✓ Get Favorite Post (17 ms)
  ✓ Get ALL Favorite (35 ms)
  ✓ Registers new User (42 ms)
  ✓ Creates New Text Post (54 ms)
  ✓ Creates New Image Post Link (62 ms)

Test Suites: 1 passed, 1 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        1.831 s, estimated 2 s
Ran all test suites matching /functions.test.js/i.
20:39:35:~/Documents/GitHub/CSC648GroupProject % >
```

4 Integration Testing

- In this scenario, a user visits the Momentus site without an account. They successfully create an account, which automatically logs them in. They then log out after browsing the site. Later on, they want to browse the site again, so they log in again.



- In this scenario, an existing user wants to browse the Momentus site, so they log in. They then make a new collection to add their photos to. Realizing they have no posts yet, they create a post, then add it to the collection they just made.