

SW Engineering CSC648/848 Milestone 5

Project title: **Momentum**

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	2
2	Coding Style	3
3	Unit Tests	4
4	Integration Testing	8

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.

```
1  const functions = require("../functions");
2  const expect = require('expect');
3  const path = require("path");
4
5  test('All User Collection', done => {
6    functions.allCollection(2, function(jason) {
7      try {
8        expect(jason).toEqual(
9          expect.arrayContaining([
10             expect.objectContaining({
11               userID: 2
12             })
13           ])
14        );
15        done();
16      } catch (error) {
17        done(error);
18      }
19    });
20  });
21
22  test('New Collection', done => {
23    functions.newCollection("testlol", 2, function(jason) {
24      try {
25        expect(jason).toEqual([]);
26        done();
27      } catch (error) {
28        done(error);
29      }
30    });
31  });
32
33  test('Append Collection', done => {
34    functions.appendCollection(2, 3, function(jason) {
35      try {
36        expect(jason).toEqual(
37          expect.arrayContaining([
38             expect.objectContaining({
39               postID: 3
40             })
41           ])
42        );
43        done();
44      } catch (error) {
45        done(error);
46      }
47    });
48  });
49
50  test('View Collection', done => {
51    functions.viewCollection(2, function(jason) {
52      try {
53        expect(jason).toEqual(
54          expect.arrayContaining([
55             expect.objectContaining({
56               postID: 7
57             })
58           ])
59        );
60        done();
61      } catch (error) {
62        done(error);
63      }
64    });
65  });
66
```

```

67 test('Post By Id', done => {
68   functions.postById(7, function(jason) {
69     try {
70       expect(jason).toEqual(
71         expect.arrayContaining([
72           expect.objectContaining({
73             postID: 7
74           })
75         ])
76       );
77       done();
78     } catch (error) {
79       done(error);
80     }
81   });
82 });
83
84 test('Post By User', done => {
85   functions.postByUser(2, function(jason) {
86     try {
87       expect(jason).toEqual(
88         expect.arrayContaining([
89           expect.objectContaining({
90             userID: 2
91           })
92         ])
93       );
94       done();
95     } catch (error) {
96       done(error);
97     }
98   });
99 });
100
101 test('Get Profile', done => {
102   functions.getProfile(2, function(jason) {
103     try {
104       expect(jason).toEqual(
105         expect.arrayContaining([
106           expect.objectContaining({
107             userID: 2
108           })
109         ])
110       );
111       done();
112     } catch (error) {
113       done(error);
114     }
115   });
116 });
117
118 test('Logs in User', done => {
119   functions.login("cd", "cd", function(jason) {
120     try {
121       expect(jason).toEqual(
122         expect.arrayContaining([
123           expect.objectContaining({
124             userID: 2
125           })
126         ])
127       );
128       done();
129     } catch (error) {
130       done(error);
131     }
132   });
133 });

```

```

135 test('Favorite', done => {
136     functions.favorite(7, function(jason) {
137         try {
138             expect(jason).toEqual(
139                 expect.arrayContaining([
140                     expect.objectContaining({
141                         postID: 7
142                     })
143                 ])
144             );
145             done();
146         } catch (error) {
147             done(error);
148         }
149     });
150 });
151
152 test('Get Favorite Post', done => {
153     functions.getFavPost(6, function(jason) {
154         try {
155             expect(jason).toEqual(
156                 expect.arrayContaining([
157                     expect.objectContaining({
158                         postID: 6
159                     })
160                 ])
161             );
162             done();
163         } catch (error) {
164             done(error);
165         }
166     });
167 });
168

```

```

169 test('Get ALL Favorite', done => {
170     functions.getAllFav(1, function(jason) {
171         try {
172             expect(jason).toEqual(
173                 expect.arrayContaining([
174                     expect.objectContaining({
175                         userID: 2
176                     })
177                 ])
178             );
179             done();
180         } catch (error) {
181             done(error);
182         }
183     });
184 });
185
186 test('Registers new User', done => {
187     functions.register("test@gmail.com", "testUName", "test", "testFirst",
188         try {
189             expect(jason).toEqual(
190                 expect.arrayContaining([
191                     expect.objectContaining({
192                         username: "testUName"
193                     })
194                 ])
195             );
196             done();
197         } catch (error) {
198             done(error);
199         }
200     });
201 });
202

```

```

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.

```
21:41:29:~/Documents/GitHub/CSC648GroupProject % > npm test functions-int.test.js
> momentus@1.0.0 test
> jest "functions-int.test.js"

PASS ./functions-int.test.js
  ✓ User register, log out, then log back in with new account (98 ms)

Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 1.181 s
Ran all test suites matching /functions-int.test.js/i.
```

- 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.

```
21:41:59:~/Documents/GitHub/CSC648GroupProject % > npm test functions-int.test.js
> momentus@1.0.0 test
> jest "functions-int.test.js"

PASS ./functions-int.test.js
  ✓ User register, log out, then log back in with new account (121 ms)
  ✓ User login, creates a collection, creates a post, adds the post to the collection, logs out (172 ms)

Test Suites: 1 passed, 1 total
Tests: 2 passed, 2 total
Snapshots: 0 total
Time: 3.022 s
Ran all test suites matching /functions-int.test.js/i.
22:03:33:~/Documents/GitHub/CSC648GroupProject % >
```