
OverAchiever Website

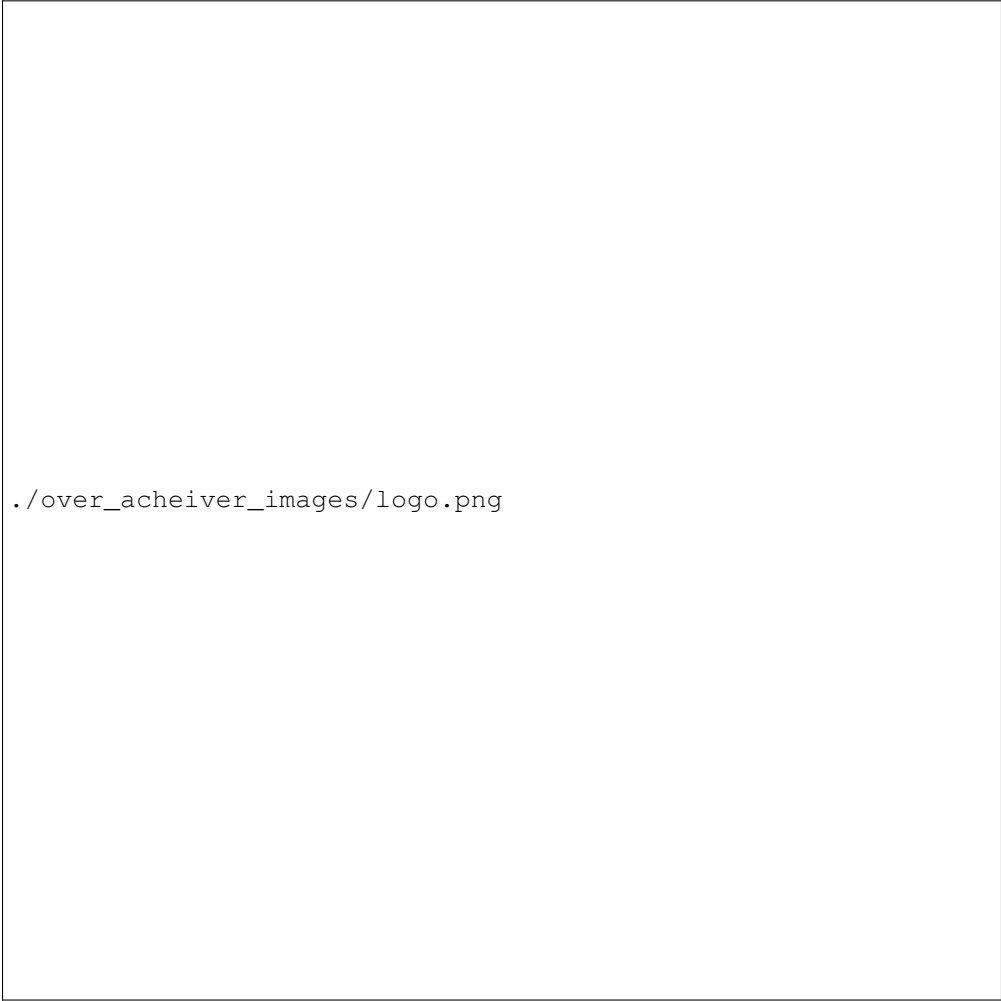
Trinity Armstrong

Jan 23, 2020

Contents

1	January 6th, 2020	3
2	January 7th 2020	5
3	January 8th 2020	7
4	January 9th, 2020	9
5	January 10th, 2020	11
6	January 13th, 2020	13
7	January 14th, 2020	15
8	January 15th, 2020	19
9	January 16th, 2020	21
10	January 17th, 2020	23
11	January 20th, 2020	25

In this journal I will be documenting what I have completed each day in regards to my serverless web app, Over-Achiever.



`./over_acheiver_images/logo.png`

CHAPTER 1

January 6th, 2020

1. I had created a new Cloud9 instance in my \$70 AWS starter account, I had named it “OverAchiever”. This where I can create and edit my code.
2. I connected my GitHub repo (<https://github.com/Trinity-Armstrong/ICS3U-2019-Group16>) to the root of my new Cloud9 instance. In other words, I had uploaded all the files and images that are stored in my github repo so I can make changes to them.

Listing 1: How to connect Cloud9 instance root to GitHub repo

```
vocstartsoft:~/environment $ git init
Initialized empty Git repository in /home/ubuntu/environment/.git/
vocstartsoft:~/environment (master) $ git remote add origin https://github.com/Mr-
↪Coxall/Amplify-Test
vocstartsoft:~/environment (master) $ git pull origin master
```


CHAPTER 2

January 7th 2020

1. I created a index.html file in the root of my Cloud9. This contains the code for “Hello, World!” that I will be using to test my new website once I have connected to Amplify.

Listing 1: index.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>AWS Serverless Web App</title>
5   </head>
6   <body>
7     Hello, World!
8   </body>
9 </html>
```

2. Next, I pushed all the changes I have made to my github repo in my master terminal. This includes my index.html file.
3. Finally, I connected this instance to Amplify. I did this by going to my dashboard and opening AWS amplify through the services tab. From there, I was able to create my Amplify instance connected to GitHub and deploy my hello world program. I was able to confirm through the provided URL that my code is up and running correctly.
4. Created a role in IAM called ‘AWS_Serverless_Web_App’ through AWS Identity and Access Management (IAM). After creating my database, I will be able to use this role for AWS Lambda to access it.
5. I created a lambda function in python that returns “Hello, World!” on AWS
6. I was able to test and confirm that this function was running correctly using test cases
7. using configured test event, I passed my name Trinity Armstrong into the function. It outputted “Hello, Trinity Armstrong”.

CHAPTER 3

January 8th 2020

1. Created a DynamoDB table called “chocolate_user” with email as my primary key
2. Created rows in the table to ensure that is working correctly. I used the following information: first_name, last_name and age of the user
3. I changed the capacity of my table from 5 to 1 in order to save money, as that is more than enough for the services I will be needing
4. Created a new lambda function called get_user_info

Listing 1: hello_world.py Lambda function

```
1  #!/usr/bin/env python3
2
3  # Created by: Trinity Armstrong
4  # Created on: Jan 2020
5  # This function is the Hello, World! Lambda function
6
7  import json
8
9  def lambda_handler(event, context):
10     # TODO implement
11
12     return_var = {
13         'statusCode': 200,
14         'body': json.dumps('Hello, ' + event['name'])
15     }
16
17     return return_var
```

5. copy pasted my code from the hello_world lambda function into my new function
6. Tested my new lambda function by outputting “Hello, Trinity Armstrong”

CHAPTER 4

January 9th, 2020

1. Coded a function in the lambda function `get_user_info` that returns a row from our `chocolate_user` DynamoDB
2. Tested my code and it correctly outputted the row I had selected in DynamoDB (user: John Smith) Output: { 'Item': { 'last_name': 'Smith', 'email': 'john.smith@gmail.com', 'first_name': 'John', 'age': Decimal('35') }}
3. Did a second test with a different row, it was a success (user: Jane Smith) Output: { 'Item': { 'last_name': 'Smith', 'email': 'jane.smith@gmail.com', 'first_name': 'Jane', 'age': Decimal('25') }}
4. Changed code so that an incorrect email will result in a blank row and a existing email will result in the output of the corresponding row
5. Next, I created a new API gateway, that is fully functioning. When you enter a parameter, it outputs the corresponding information and when you enter NO parameter, there is a nice response. It is now published on the internet and later I will be able to use this URL while coding in HTML. URL: https://y8s2gy3mi8.execute-api.us-east-1.amazonaws.com/prod/user-profile?user_email=mr.coxall@mths.ca
6. I altered my lambda code to trap errors with a try catch statement
7. Used Javascript “Fetch” through `index.html` to call my API and present the data for “jane.smith@gmail.com” on my website

Listing 1: `get_user_info.py` Lambda function

```
1  #!/usr/bin/env python3
2
3  # Created by: Trinity Armstrong
4  # Created on: Jan 2020
5  # This function returns a row from our chocolate_user DynamoDB
6
7  import json
8  import boto3
9  import decimal
10
11
12 def replace_decimals(obj):
13     # Helper class to Decimals in an arbitrary object
```

(continues on next page)

(continued from previous page)

```
14         # from: https://github.com/boto/boto3/issues/369
15
16     if isinstance(obj, list):
17         for i in range(len(obj)):
18             obj[i] = replace_decimals(obj[i])
19         return obj
20     elif isinstance(obj, dict):
21         for k, v in obj.items():
22             obj[k] = replace_decimals(v)
23         return obj
24     elif isinstance(obj, set):
25         return set(replace_decimals(i) for i in obj)
26     elif isinstance(obj, decimal.Decimal):
27         if obj % 1 == 0:
28             return int(obj)
29         else:
30             return float(obj)
31     else:
32         return obj
33
34
35 def lambda_handler(event, context):
36     # get a row from our chocolates_user table
37
38     dynamodb = boto3.resource('dynamodb')
39     table = dynamodb.Table('chocolate_users')
40     response = table.get_item(
41         Key = {
42             'email': event['email_address']
43         }
44     )
45
46     try:
47         results = response["Item"]
48         results = replace_decimals(results)
49     except:
50         results = {}
51
52     return {
53         'statusCode': 200,
54         'body': json.dumps(results)
55     }
```

CHAPTER 5

January 10th, 2020

1. I created an AWS Cognito user pool called “cognitoPool”
2. I used the Cognito built-in signup URL to add myself as a user through my trinity.armstrong@ocsbstudent.ca email. I successfully recieved a link and was confirmed as a user
3. I confirmed that this new user exists in the Cognito pool

CHAPTER 6

January 13th, 2020

Updated my readthedocs webpage

CHAPTER 7

January 14th, 2020

1. I created a javascript folder called “js”
2. I downloaded the JavaScript libraries “amazon-cognito-auth.min”, “amazon-cognito-identity.min”, “and config.js” into the folder “js”
3. I updated the “config.js” file with my app information from AWS Cognito
4. Typed the code for my sign in page in the file “sign-in.html”. This runs 2 input boxes for your email address and password and a sign-in button
5. I signed into my confirmed account and it successfully outputted “You are logged in as: [trinity.armstrong@ocsbstudent.ca](#)”

Listing 1: sign-in.html

```
1 <!DOCTYPE html>
2
3 <html lang="en">
4   <head>
5     <meta charset="utf-8">
6
7     <!-- Javascript SDKs-->
8     <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
9     <script src="js/amazon-cognito-auth.min.js"></script>
10    <script src="https://sdk.amazonaws.com/js/aws-sdk-2.596.0.min.js"></script>
11    <script src="js/amazon-cognito-identity.min.js"></script>
12    <script src="js/config.js"></script>
13  </head>
14
15  <body>
16    <form>
17      <h1>Please sign in</h1>
18
19      <input type="text" id="inputUsername" placeholder="Email address" name=
    ↪ "username" required autofocus>
```

(continues on next page)

(continued from previous page)

```

20     <input type="password" id="inputPassword" placeholder="Password" name="password
↪" required>
21     <button type="button" onclick="signInButton()">Sign in</button>
22 </form>
23
24 <br>
25 <div id='logged-in'>
26     <p></p>
27 </div>
28
29 <p>
30     <a href="./profile.html">Profile</a>
31 </p>
32
33 <br>
34 <div id='home'>
35     <p>
36         <a href='./index.html'>Home</a>
37     </p>
38 </div>
39
40 <script>
41
42     var data = {
43         UserPoolId : _config.cognito.userPoolId,
44         ClientId : _config.cognito.clientId
45     };
46     var userPool = new AmazonCognitoIdentity.CognitoUserPool(data);
47     var cognitoUser = userPool.getCurrentUser();
48
49     function signInButton() {
50         // sign-in to AWS Cognito
51
52         var authenticationData = {
53             Username : document.getElementById("inputUsername").value,
54             Password : document.getElementById("inputPassword").value,
55         };
56
57         var authenticationDetails = new AmazonCognitoIdentity.
↪AuthenticationDetails(authenticationData);
58
59         var poolData = {
60             UserPoolId : _config.cognito.userPoolId, // Your user pool id here
61             ClientId : _config.cognito.clientId, // Your client id here
62         };
63
64         var userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);
65
66         var userData = {
67             Username : document.getElementById("inputUsername").value,
68             Pool : userPool,
69         };
70
71         var cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);
72
73         cognitoUser.authenticateUser(authenticationDetails, {
74             onSuccess: function (result) {

```

(continues on next page)

(continued from previous page)

```
75     var accessToken = result.getAccessToken().getJwtToken();
76     console.log(result);
77
78     //get user info, to show that you are logged in
79     cognitoUser.getUserAttributes(function(err, result) {
80         if (err) {
81             console.log(err);
82             return;
83         }
84         console.log(result);
85         document.getElementById("logged-in").innerHTML = "You are logged in_
↪as: " + result[2].getValue();
86     });
87
88     },
89     onFailure: function(err) {
90         alert(err.message || JSON.stringify(err));
91     },
92     });
93 }
94 </script>
95
96 </body>
97 </html>
```


CHAPTER 8

January 15th, 2020

1. I typed the code for my sign out page in the file “sign-out.html”

Listing 1: sign-out.html

```
1  <!doctype html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8">
5      <!--Cognito JavaScript-->
6      <script src="js/amazon-cognito-identity.min.js"></script>
7      <script src="js/config.js"></script>
8    </head>
9
10   <body>
11     <div class="container">
12       <div>
13         <h1>Sign Out</h1>
14         <p>Successfully signed-out</p>
15       </div>
16
17       <br>
18       <div id='home'>
19         <p>
20           <a href='./index.html'>Home</a>
21         </p>
22       </div>
23     </div>
24
25     <script>
26       var data = {
27         UserPoolId : _config.cognito.userPoolId,
28         ClientId : _config.cognito.clientId
29       };
30       var userPool = new AmazonCognitoIdentity.CognitoUserPool(data);
```

(continues on next page)

(continued from previous page)

```
31  var cognitoUser = userPool.getCurrentUser();
32
33  window.onload = function() {
34    if (cognitoUser !== null) {
35      cognitoUser.getSession(function(err, session) {
36        if (err) {
37          alert(err);
38          return;
39        }
40        console.log('session validity: ' + session.isValid());
41
42        // sign out
43        cognitoUser.signOut();
44        console.log("Signed-out");
45      });
46    } else {
47      console.log("Already signed-out")
48    }
49  }
50  </script>
51
52  </body>
53  </html>
```

2. I signed into my account, then called “signout()” function and successfully signed out. The output is “Successfully signed-out”, this proves that the code is working.

CHAPTER 9

January 16th, 2020

1. I created a `profile.html` file in order to add our user's profile to my webpage once signed in
2. I did this by copying my `sign-out.html` code
3. removing the sign-out code from my `profile.html` file
4. copying `getUserAttributes()` function from `sign-in.html` to add to my file
5. copying `getUser()` function from `temp.html` to add to my file
6. I confirmed that my code was running correctly by signing in, and checking the profile page that had successfully outputted my profile information

Listing 1: `profile.html`

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <!--Cognito JavaScript-->
6     <script src="js/amazon-cognito-identity.min.js"></script>
7     <script src="js/config.js"></script>
8   </head>
9
10  <body>
11    <div class="container">
12      <div>
13        <h1>Profile</h1>
14      </div>
15      <div id='profile'>
16        <p></p>
17      </div>
18    <div>
19
20      <br>
21      <div id='home'>
```

(continues on next page)

```

22     <p>
23         <a href='./index.html'>Home</a>
24     </p>
25 </div>
26
27 <script>
28
29     async function getUser(email_address) {
30         // get the user info from API Gate
31
32         const api_url = 'https://gonvpjbyuf.execute-api.us-east-1.amazonaws.com/prod/
↪user-profile?user_email=' + email_address;
33         const api_response = await fetch(api_url);
34         const api_data = await (api_response).json();
35         console.log(api_data);
36
37         const div_user_info = document.getElementById('profile');
38         div_user_info.innerHTML = api_data['body'];
39     }
40
41     var data = {
42         UserPoolId : _config.cognito.userPoolId,
43         ClientId : _config.cognito.clientId
44     };
45     var userPool = new AmazonCognitoIdentity.CognitoUserPool(data);
46     var cognitoUser = userPool.getCurrentUser();
47
48     window.onload = function() {
49         if (cognitoUser !== null) {
50             cognitoUser.getSession(function(err, session) {
51                 if (err) {
52                     alert(err);
53                     return;
54                 }
55                 //console.log('session validity: ' + session.isValid());
56
57                 cognitoUser.getUserAttributes(function(err, result) {
58                     if (err) {
59                         console.log(err);
60                         return;
61                     }
62                     // user email address
63                     console.log(result[2].getValue());
64                     getUser(result[2].getValue())
65                 });
66
67             });
68         } else {
69             console.log("Already signed-out")
70         }
71     }
72 </script>
73
74 </body>
75 </html>

```

CHAPTER 10

January 17th, 2020

1. I removed javascript code from my HTML files and moved them into seperate .js files
2. Used google's MDL code to act as the css of my website
4. Created home page, about page , Sign in page, Sign out page and Profile page on website (html and javascript)

CHAPTER 11

January 20th, 2020

1. Created register and settings page on website
2. Added background's to my website pages
3. Added my OverAchiever logo to the header of each page
4. Changed fonts and margins to make text more visually appealing
5. Added a button link to the registration page on the home page