GAME 3110

Assignment 4

Stack of serverless services

Ricardo Shimoda Nakasako 101128885

Git Repository: https://github.com/ricardoshimoda/gbc_game3110.git

User Management

All services here are connected to the dynamodb table "Players".

1. Login User

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/Login

This service only accepts a POST and in the body of the request you need to send the username and password like this:

```
{
        "Username" : "RicardoShimoda",
        "Password":"password"
}
```

When successful, it returns the a message with the following format:

```
{
    "result": "User logged in successfully: (68d6057a-207c-445c-a84c-fd8f4787c2b5)"
}
```

2. Logout User

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/Logout

This service only accepts a GET and you have to send a Username parameter in the QueryString:

```
https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/Logout?Username=<user_to_logout>
```

When successful, it returns the a message with the following format:

```
{
    "result": "User logged out successfully"
}
```

3. Register User

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/UserMgm

This service only accepts a PUT and in the body of the request you need to send user data (username, password and email) in the following format:

```
{
 "Username": "anotherlogin",
 "Password":"password",
 "Email":"test@test.com"
}
```

When successful, it returns the a message with the following format:

```
{
    "result": "User registered successfully"
}
```

4. Update User

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/UserMgm

This service only accepts a POST and in the body of the request you need to send user data (username, password and email) in the following format:

```
{
 "Username": "anotherlogin",
 "Password":"password",
 "Email":"test@test.com"
}
```

When successful, it returns the a message with the following format:

```
{
    "result": "User updated successfully"
}
```

5. Retrieve User Profile

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/UserMgm

This service only accepts a GET and in the QueryString of the request you need to send the username in the following format:

```
https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/UserMgm?Username=<user_whose_profile_we_want_to_get>
```

When successful, it returns the a message with the following format:

```
{
    "password": "password",
    "user_id": "username_chosen_by_the_user",
    "session_tk": "-",
    "email": "test_test@test.ca",
    "login_date": "-"
}
```

If the user is currently logged in, this will also show his token and the login date.

Score Management

All services here are connected to the dynamodb table "Score".

1. User score retrieval:

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/Score

This service only accepts a GET and in the QueryString of the request you need to send the username and the game name in the following format:

https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/Score?Username=<username>&GameName=<game_name>

When successful, it returns the a message with the following format:

```
{
    "Score": 2048.0,
    "UserGame": "RicardoShimoda|Battleship"
}
```

The user has to belong to the table "Players" and the game has to have an entry in the table "RemoteSettings" otherwise this service will return an error message

2. Score update

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/Score

This service only accepts a POST and in the body of the request you need to send the username, the game name, and the score in the following format:

```
{
        "GameName":"Battleship",
        "Username": "RicardoShimoda",
        "Score": 2048
}
```

When successful, it returns the a message with the following format:

```
{"result": Score created successfully for user RicardoShimoda in game Battleship "}
```
OR
```
{"result": Score updated successfully for user RicardoShimoda in game Battleship "}
```

Depending on whether it's the first time the player plays this game or not. As with the previous service, the user has to belong to the table "Players" and the game has to have an entry in the table "RemoteSettings" otherwise this service will return an error message

Analytics

This service is directly linked to the DynamoDB table "Analytics". All services check existence in tables "Players", "RemoteSettings", and "Event" to verify existence of Username, Game Name and Event Type respectively, before creating an item in the analytics table.

The types of event that can be logged on are:

Event	Body

Bug	<pre>{
	  "Exception":"Yes",
	  "Text" : "NullReferenceException line 87829820"
	}</pre>

GamePurchase	<pre>{
	  "Merchant": "AppleStore",
	  "Promotion": "Yes",
	  "Price": 0.0
	}</pre>

GameQueue	<pre>{
	  "Timestamp": "20200405134024"
	}</pre>

GameStart	<pre>{
	  "Timestamp": "20200405134512",
	  "Opponent" : "UserOpponent"
	}</pre>

GameEnd	<pre>{
	  "Timestamp": "20200405135839",
	  "Result" : "Lost",
	  "Score" : 110
	}</pre>

ItemPurchase	<pre>{
	  "Merchant": "AppleStore",
	  "Item" : "Sacred Graal"
	  "Promotion": "Yes",
	  "Price": 120.0
	}</pre>

OpenCart	<pre>{
	  "Timestamp": "20200405134024"
	}</pre>

1.Track Event

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/AnalyticsFunc

This service only accepts a POST and in the body of the request you need to send the username, the game name, the event name and the event data in the following format:

```
{
        "GameName": "game_name",
        "Username": "username",
        "EventName": "event_name_from_the_list_above",
        "EventData": <one_of_the_object>
}
```

For instance:

```
{
        "GameName": "Battleship",
        "Username": "RicardoShimoda",
        "EventName": "Bug",
        "EventData": {
          "Exception":"Yes",
          "Text" : "NullReferenceException line 3213"
         }
}
```

When successful, it returns the a message with the following format:

```
{
    "result": "Event logged successfully"
}
```

2. Event Retrieval

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/AnalyticsFunc

This service only accepts a GET request and the parameters to filter results have to be defined in the QueryString. All parameters are optional:

- GameName: the name of the game
- Username : the user whose events are requested
- DateTimeStart: establishes the initial date and time of retrieval request - in the format: yyyymmddHHMMSS (i.e: April 5th, 2020, 11:09:34 PM would be 20200405230934)
- DateTimeEnd: established the final date and time of interval request - once more in the format:
  yyyymmddHHMMSS (i.e: April 5th, 2020, 11:09:34 PM would be 20200405230934)

When successful, this service simply dumps the information requested in the response body:


```
{
    "Items": [
        {
            "EventData": {
                "Merchant": "AppleStore",
                "Promotion": "Yes",
                "Price": 0.0
            },
            "EventDate": "20200403031439",
            "EventName": "GamePurchase",
            "GameName": "Battleship",
            "Username": "RicardoShimoda"
        },
        {
            "EventData": {
                "Exception": "Yes",
                "Text": "NullReferenceException line 87829820"
            },
            "EventDate": "20200405134024",
            "EventName": "Bug",
            "GameName": "Battleship",
            "Username": "RicardoShimoda"
        },
        {
            "EventData": {
                "Exception": "Yes",
                "Text": "NullReferenceException line 87829820"
            },
            "EventDate": "20200403033706",
            "EventName": "Bug",
            "GameName": "Battleship",
            "Username": "RicardoShimoda"
        }
    ],
    "Count": 3,
    "ScannedCount": 4,
    "ResponseMetadata": {
        "RequestId": "5IFSBNKE056DNU4SIT1K4OGNJNVV4KQNSO5AEMVJF66Q9ASUAAJG",
        "HTTPStatusCode": 200,
        "HTTPHeaders": {
            "server": "Server",
            "date": "Sun, 05 Apr 2020 14:45:43 GMT",
            "content-type": "application/x-amz-json-1.0",
            "content-length": "706",
            "connection": "keep-alive",
            "x-amzn-requestid": "5IFSBNKE056DNU4SIT1K4OGNJNVV4KQNSO5AEMVJF66Q9ASUAAJG",
            "x-amz-crc32": "1428065602"
        },
        "RetryAttempts": 0
    }
}
```


Where "Items" is an array with all the needed data.

Remote Settings

This service deals directly with the DynamoDB "RemoteSettings" - this same table is used throughout other services to see whether a game really exists or not. Insertion has to be done manually in this table - services only read and alter existing entries.

1. Updating Values:

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/RemoteSettingsFunc

This service only accepts a POST and in the body of the request you need to send the game name, the key - name of the property you want to create or update - and the value - its respective current value - in the following format:

```
{
        "GameName":"game_name",
        "Key": "remote_property",
        "Value": "value_for_remote_property"
}
```

When successful, it returns a message with the following format:

```
{
    "result": "game_name updated remote_property successfully"
}
```

2. Retrieving Values:

URL: https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/RemoteSettingsFunc

This service only accepts a GET request and it has to have the GameName as a parameter:

```
https://zfum6anmbl.execute-api.us-east-1.amazonaws.com/default/RemoteSettingsFunc?GameName=<game_name>
```

When successful, it returns a dictionary with all the properties stored for that particular gamet:

```
{
    "prop1": "value1",
    "prop2": "value2",
    "prop3": "value3",
    "GameName": "game_anme"
}
```