GAME 3110

Assignment 3 Bonus

Analytic micro services

Salick Talhah 101214166

Git Repository: https://github.com/Talha991s/Game3110_Assignment3.git

PostMan Link to collection:

https://www.getpostman.com/collections/3db89c3c787deeb04524

Analytics

URL: https://6260kxb034.execute-api.us-east-1.amazonaws.com/default/AnalyticsFunc

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://6260kxb034.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": "Battle",
        "Username": "SalickTalhah",
        "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://6260kxb034.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: November 30th, 2020, 11:09 PM would be 202011302309)
- DateTimeEnd: established the final date and time of interval request - once more in the format:
  yyyymmddHHMMSS (i.e: November 30th, 2020, 11:09 PM would be 202011302309)

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

```
{
```

```
    "Items": [
        {
            "EventData": {
                "Merchant": "AppleStore",
                "Promotion": "Yes",
                "Price": 0.0
            },
            "EventDate": "202011300041",
            "EventName": "GamePurchase",
            "GameName": "TicTacToe",
            "Username": "SalickTalhah"
        },
        {
            "EventData": {
                "Exception": "Yes",
                "Text": "NullReferenceException line 87829820"
            },
            "EventDate": "202011300125",
            "EventName": "Bug",
            "GameName": "Battle",
            "Username": "SalickTalhah"
        },
        {
            "EventData": {
                "Exception": "Yes",
                "Text": "NullReferenceException line 87829820"
            },
            "EventDate": "202011300143",
            "EventName": "Bug",
            "GameName": "Battle",
            "Username": "SalickTalhah"
        }
    ],
    "Count": 3,
    "ScannedCount": 3,
    "ResponseMetadata": {
        "RequestId": "EfVV2R......",
        "HTTPStatusCode": 200,
        "HTTPHeaders": {
            "server": "Server",
            "date": "Mon, 30 Nov 2020 11 :50 :30 GMT",
            "content-type": "application/x-amz-json-1.0",
            "content-length": "702",
            "connection": "keep-alive",
            "x-amzn-requestid": "EFVV2R……",
            "x-amz-crc32": "1064544340"
        },
        "RetryAttempts": 0
    }
}
```


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

Remote Settings

(I am having a bug in it, I couldn’t fix it) But here are the expected result.

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://34robn3uki.execute-api.us-east-1.amazonaws.com/default/RemoteSettings

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://34robn3uki.execute-api.us-east-1.amazonaws.com/default/RemoteSettings

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

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

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