**Online unity demo – running the experiment**

**1) Amazon AWS – DynamoDB set up**

* Create an amazon AWS account: <https://aws.amazon.com/>
* Navigate to “Console” and type “DynamoDB” into the search bar
* You’ll need to create three new tables:
  + First has a table name “schoolUserData” with a primary key “tokenId”(string) and sort key “email”(string)
  + Second has a table name “schoolTrialData” with a primary key “tokenId”(string) and sort key “trial”(string)
  + Third has a table name “tokenTable” with a primary key “tokenId”(string)
* Return to the console and type “IAM” in the search bar
* Add a new user with administrator access and copy and paste your ACCESS KEY and SECRET KEY into a .txt file

**2) Uploading your generated tokens to DynamoDB**

* Install node.js: <https://nodejs.org/en/>
* Open the command prompt and type “npm install aws-sdk”
* Open “genericTokenGenerator.js” from the root directory in a code editor that will let you run code snippets through node (Visual Studio code will do the trick)
* Copy and paste your ACCESS KEY and SECRET KEY from earlier intothe generateTokens() function. Note this isn’t suitable for production because it exposes your database. You may also change the server “region” and “endpoint” settings if you’re not hosting on us-east-2 (my default).
* Input the number of tokens you require in the generateTokens() function at the bottom
* Ctrl+alt+n will allow you to run the code
* Open your browser and check the “tokenTable” table in DynamoDB to confirm all tokens have been updated

**3) Unity project**

* Download the unity project folder onto your computer
* Navigate to Assets -> Plugins -> jsplugin.jslib and open in your code editor
* Copy and paste your ACCESS KEY and SECRET KEY from earlier into **all four** functions. Note this isn’t suitable for production because it exposes your database. You may also change the server “region” and “endpoint” settings if you’re not hosting on us-east-2 (my default).
* Now open the project using unity ver. 2019.4.4f1
* File -> build settings -> select WebGL
* Make sure both scenes are selected and then click build
* Navigate to index.html in the folder you’ve just built to and open in a code editor
* Insert the following line in the <head> of your .html (at the top of the <script src=> stack):
  + <script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

This will allow you to interface with the aws SDK using javascript

* Now go back to the parent folder of index.html and archive it along with the “Build” and “TemplateData” folders into a .zip file. **NOTE: this does not require you to zip the parent folder of your project, rather, you archive the separate constituents together within your parent folder**

**4) Amazon AWS – static hosting**

* Navigate to the AWS console
* Type “Amplify” into the search bar
* “Deploy” an app -> deploy without Git provider
* Give your app a name and an environment name (neither matter)
* Click drag and drop and drag your.zip folder in -> save and deploy
* Once it has finished creating your app you’ll be able to follow a link to your unity game
* Test it out and check the data is being saved to the database correctly. You should have a single entry in the userData table for each user, and 10 entries in the trialData table for each user (one for each trial)

*NOTES ABOUT IMPLEMENTATION*

So the way this works is by interfacing with the AWS SDK for DynamoDB using javascript. You can’t normally execute javascript in Unity so we’ve had to write a plugin (“jsplugin.jslib”) which defines the functions we’d like to call in javascript. We can then import these functions from Unity classes by [DllImport(“\_\_Internal”)] using System.InteropServices. Then at any point in our script we can call them as we would any other function.

Another feature of this is the use of singletons to pass userdata between scenes. This is handled by the UserInfo.cs script attatched to an empty object (SaveData) in the first scene. We can then call any of these parameters in future scenes by use of UserInfo.Instance.yourparameterhere. This is especially useful when we’re working with two separate tables that we we’ll want to link on processing.

Finally, at present this is bypassing completely the uxf framework and the experimentcontroller handling logic is messy and potentially unwieldy when the complexity increases. Matthew and I will be working together to try and get UXF working using the AWS platform.

JP