Setting up your Mirror API Glass Project for CS50

By Christopher Bartholomew, Teaching Fellow

cbartholomew@g.harvard.edu

Contents

Section I: Creating a Project in the Google Cloud Console 2

Section II: Setting up the Mirror API with your project 3

Section III: Setting up App Registration and Open Authentication 4

Section IV: Using the Glass Developer Playground 6

Section V: Pro Tips 7

Section I: Creating a Project in the Google Cloud Console

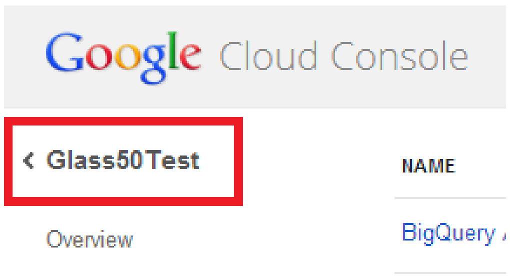
1. The credentials you are going to be using are the following:
   Important - DO NOT SHARE WITH ANYONE ELSE UNLESS THEY ARE YOUR PARTNER OR YOU'VE ASKED ME FIRST

Username: glassware50@gmail.com

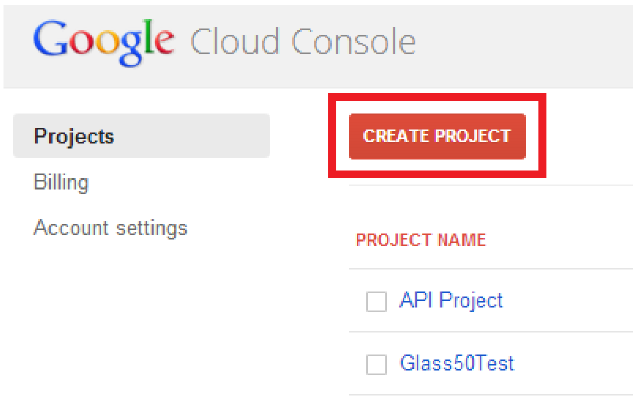
Password: iwearglass

2. Head over to https://cloud.google.com and login with the given credentials above.

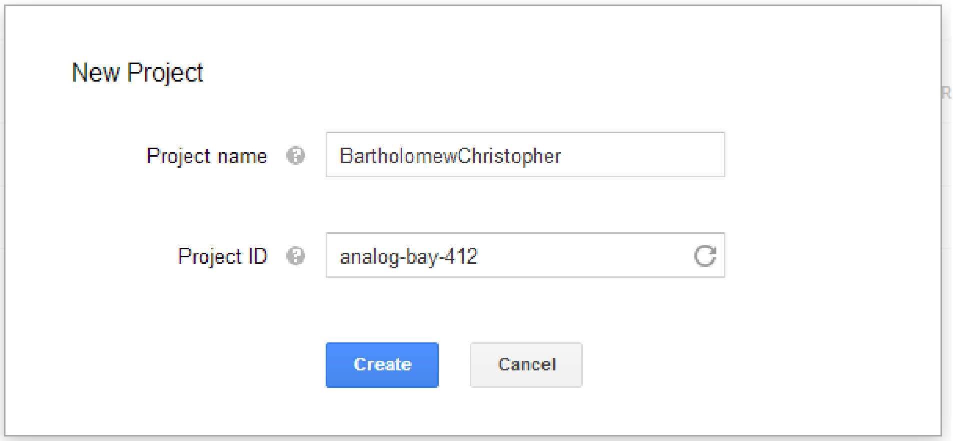
3. In the top left hand corner, you may see the last project that was logged in using this account, if this is the case, back out of it by clicking the project name.



4. Otherwise, you'll be brought to the main project menu. Here, you'll create and define your new project by selecting "Create Project".



5. Name the project using the following convention: **LastnameFirstname** or if you are with a partner do **LastnameAndLastnameOfPartner**. The maximum character limit is 30. Also, keep the Project ID default. Afterwards, select "Create". ***Note: You might have to hit "Refresh" if your project doesn't show up immediately.***


New Project

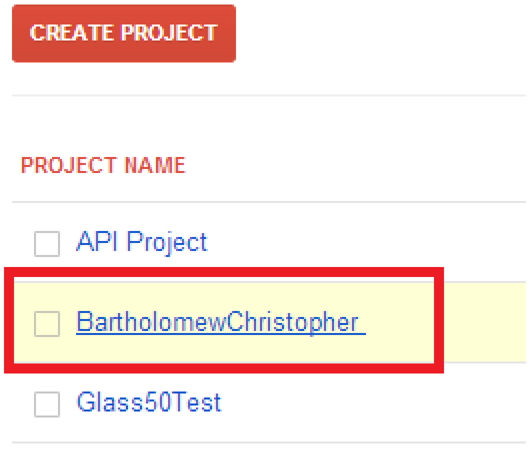
Project name: BartholomewChristopher

Project ID: analog-bay-412

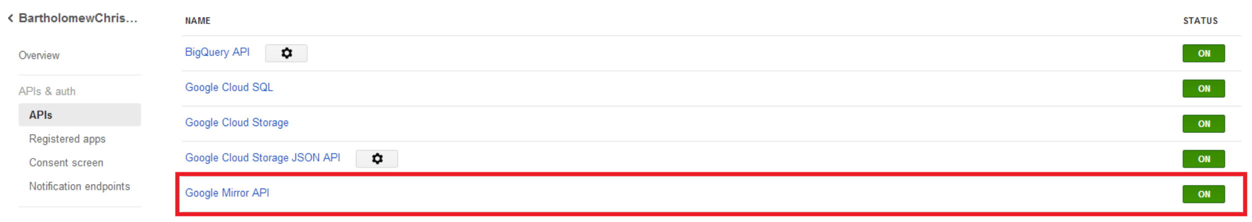
Section II: Setting up the Mirror API with your project

Note: If you have not completed Section I, please do so before continuing.

1. From the cloud console, click into your project that you've just created.



2. On the left hand navigation menu, select "***APIs & auth***" then "***APIs***". On the right hand side of the screen, a list of APIs will be displayed. Scroll down until you see "***Google Mirror API***", and then toggle the switch on the left hand side of it to "***On***". **Note: you might have to refresh the page for the toggle to show up as "On". This is because the API is then moved to the top of the screen.**

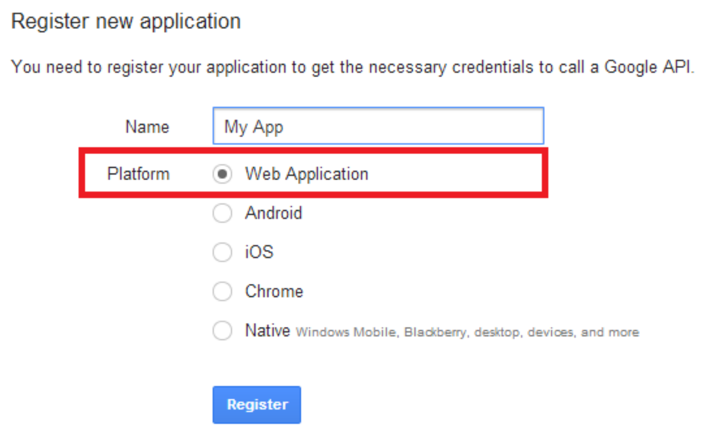


NAME	STATUS
BigQuery API	ON
Google Cloud SQL	ON
Google Cloud Storage	ON
Google Cloud Storage JSON API	ON
Google Mirror API	ON

Section III: Setting up App Registration and Open Authentication

Note: This step is extremely important if you want to interact with the playground and other Mirror API functionality.

1. If you haven't entered nor created your project yet, do so now (*Section I & II*). Once inside of your project console, go to the left hand navigation menu and select "***APIs & auth***". Then select "***Registered Apps***". If you do not have any existing applications, you'll be prompted to create a new one. Provide any name of your choosing, and ensure that "***Web Application***" is checked. Then select the "***Register***" button below.
2.



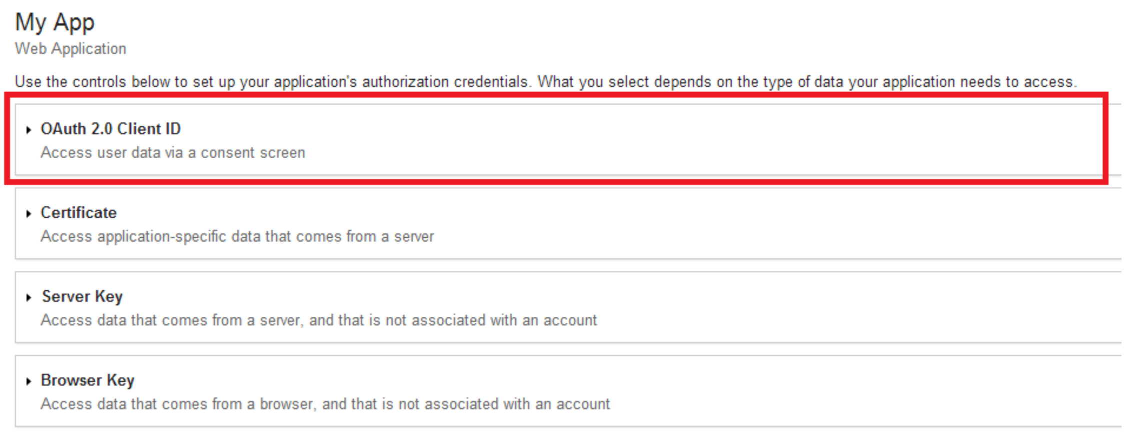
Register new application

You need to register your application to get the necessary credentials to call a Google API.

Name: My App

Platform: [✓ Web Application] Android iOS Chrome Native Windows Mobile, Blackberry, desktop, devices, and more

3. Once your application is registered, you'll be sent over to menu which has a few menu options. For now, we'll only need to focus on setting up the OAuth 2.0 Client ID. select "***OAuth 2.0 Client ID***" menu. ***Note: If you are not sure what OAuth is, please refer to the following wiki article: http://en.wikipedia.org/wiki/OAuth.***


My App
Web Application
Use the controls below to set up your application's authorization credentials. What you select depends on the type of data your application needs to access.

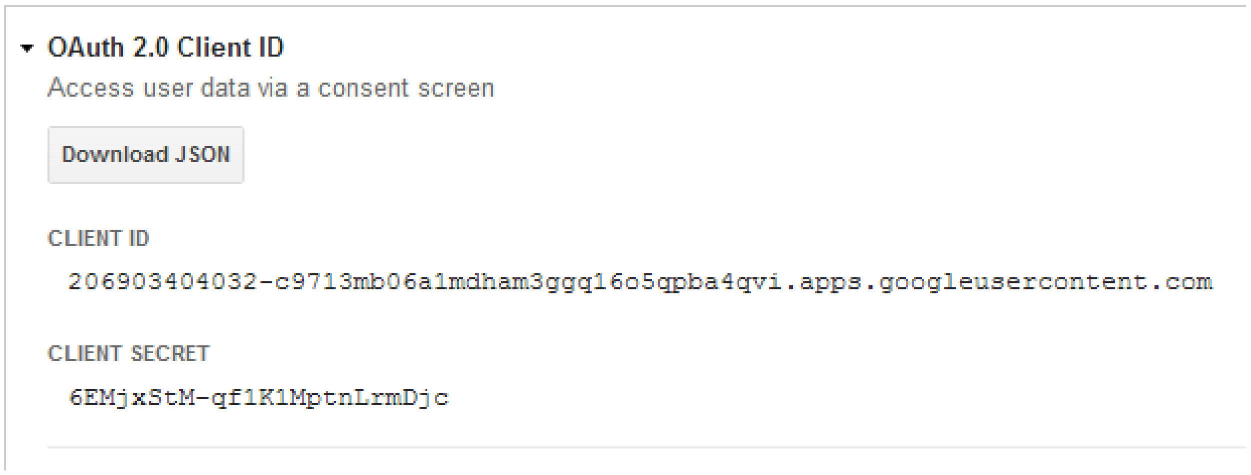
OAuth 2.0 Client ID
Access user data via a consent screen

Certificate
Access application-specific data that comes from a server

Server Key
Access data that comes from a server, and that is not associated with an account

Browser Key
Access data that comes from a browser, and that is not associated with an account

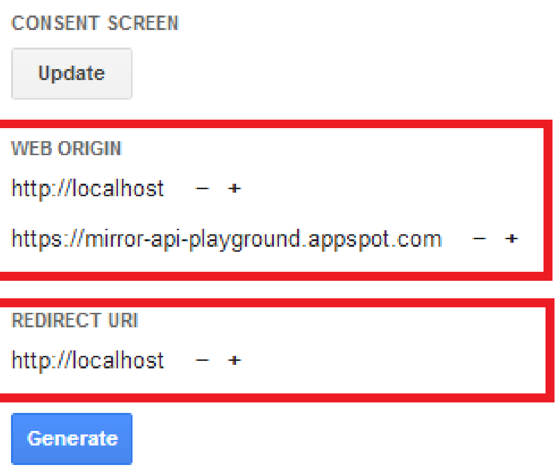
4. Next, you'll be showed a client id and a client secret. ***You'll need this later to interact with the API from your actual service***. ***Copy the Client ID and put it somewhere easily accessible to you in the future, perhaps in a code file within your development environment.*** As for your client secret, you can ignore this for now.


OAuth 2.0 Client ID
Access user data via a consent screen

CLIENT ID
206903404032-c9713mb06a1mdham3ggq16o5qpba4qvi.apps.googleusercontent.com

CLIENT SECRET
6EMjxStM-qf1K1MptnLrmDjc

5. Next, you'll need to update the "***Web Origin***" and "***Redirect URI***". When you've published your service, you'll need to change this; however, for now – Enter "**http://localhost**" for both the "***Web Origin***" and the "***Redirect URI***". Afterwards, select the "***Generate***" button. After the page has been generated, select the "**+**" (plus) to add a new "***Web Origin***", you'll need to do this in order to use the developer playground. In the new "***Web Origin***" text field, enter the following end point: ***https://mirror-api-playground.appspot.com*** and then select the "***Generate***" button once again. The output should resemble the following:

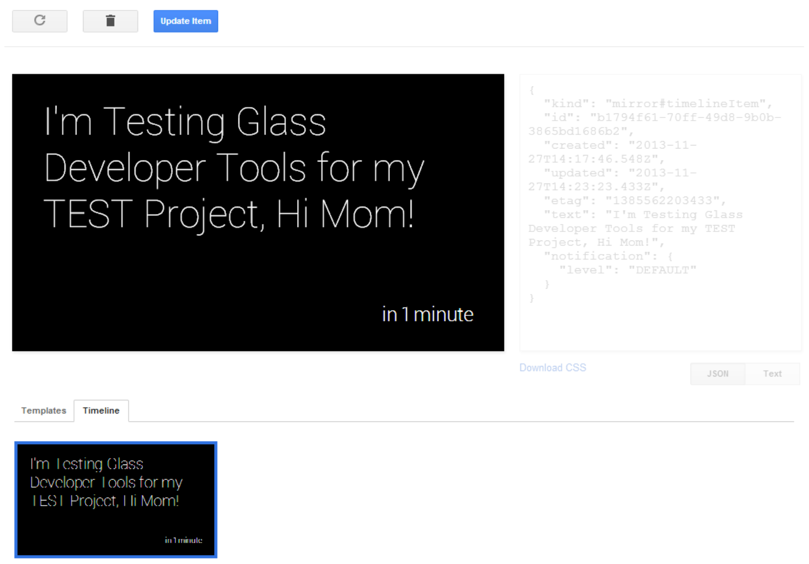


6. If you later want to customize your own "***Consent Screen***" i.e. when a user selects to subscribe to your service, you can select "***Update***". For now, however, we've completed the OAuth 2.0 Setup, and we can now use the Playground.

Section IV: Using the Glass Developer Playground

Note: The developer playground will allow you to do many things such as insert, update, and remove timeline cards. Furthermore, it will even allow you to use a "Proxy" so that you can test your Glass service and how it interacts with Glass itself. Ensure you have completed Sections I-III before attempting to use, otherwise, this will not work.

1. If you haven't done so already, make sure you copy your "***Client ID***" (not client secret) that you've created in your previous project. Afterwards, head over to the Glass's developer playground: https://developers.google.com/glass/tools-downloads/playground
2. On the playground page, there is a text box where it will ask you to input your "***Client ID***" paste your "***Client ID***" into the text field and select "***Authorize***". Once you've selected "***Authorize***", a consent window will appear asking you consent to the specific "***Scope***" that was provided. In this case, the "***Scope***" (https://developers.google.com/accounts/docs/OAuth2Login#authenticationuriparameters) is "View and manage your Glass timeline". Here, you'll just want to select the "***Accept***" button.
3. Once you've accepted the consent, you're ready to use the playground. Here, you'll be able to formulate and test inserting, updating, and reviewing items on your Glass's timeline. You'll also be able to select a variety of different templates that will accommodate the various styles. Because your service will be using the same "**Client ID**" as what you've just entered, anything that is done from your service will be sent to your glass and viewable here in the timeline. This is all due to everything being bounded to that "**Client ID**".



I'm Testing Glass Developer Tools for my TEST Project, Hi Mom!

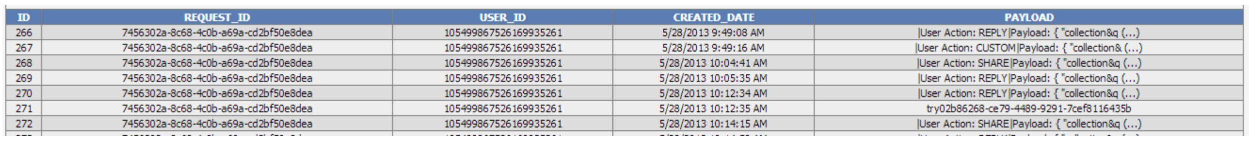
in 1 minute

```
{
  "kind": "mirror#timelineItem",
  "id": "b1794f61-70ff-49d8-9b0b-
3865bd1686b2",
  "created": "2013-11-
27T14:17:46.548Z",
  "updated": "2013-11-
27T14:23:23.433Z",
  "etag": "1385562203433",
  "text": "I'm Testing Glass
Developer Tools for my TEST
Project, Hi Mom!",
  "notification": {
    "level": "DEFAULT"
  }
}
```

4. When you have published your service to an external resource (other than localhost), and you want extend functionality such as allowing "***Subscriptions***" then you'll need to pre-append you're your subscription URLS with the Google Proxy (***This is of course not needed if you are already using a server that has an SSL Certificate, i.e. HTTPS)*** https://developers.google.com/glass/tools-downloads/subscription-proxy

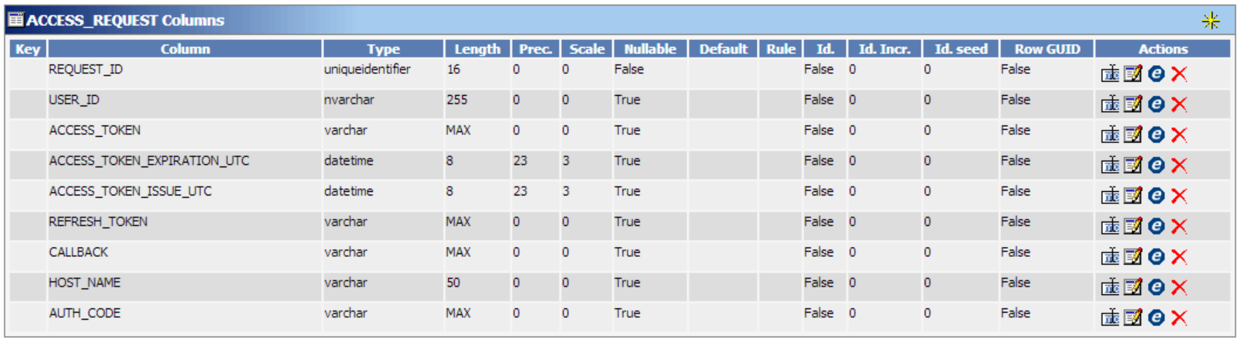
Section V: Pro Tips

1. **Make a default Chrome Profile for Glassware50**. To do this, Open Chrome, go to the URL, and type: chrome://settings/. Afterwards, scroll down to the "**Users**" section and select "***Add new user…"*** Select a theme and name. Then sign into your new Chrome profile. This way, you'll be able to swap out of Chrome Profiles without using Incognito Mode. Makes development really easy.
2. **Log all incoming JSON requests to your service handlers inside of the database.** Using the Authorization Token and User ID as additional fields. For example, in the blogger Glassware that I created (https://myglassapps.com/#blogger), I log all incoming requests so that I can debug why something isn't working correctly. Meaning, I can take that same JSON payload, dump it into some javascript page, and send the request to my "***localhost***". I can also send the JSON dump directly to my service locally and test it there. There database table that I created for logging requests is the following:



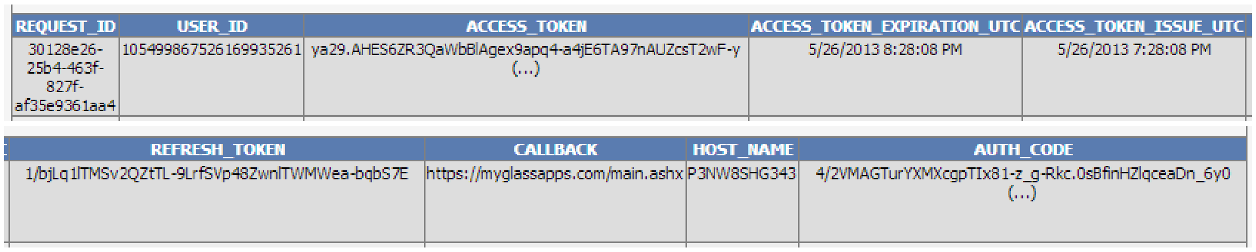
ID	REQUEST_ID	USER_ID	CREATED_DATE	PAYLOAD
266	7456302a-8c68-4c0b-a69a-cd2bf50e8dea	105499867526169935261	5/28/2013 9:49:08 AM	|User Action: REPLY|Payload: { "collection&q (...)
267	7456302a-8c68-4c0b-a69a-cd2bf50e8dea	105499867526169935261	5/28/2013 9:49:16 AM	|User Action: CUSTOM|Payload: { "collection& (...)
268	7456302a-8c68-4c0b-a69a-cd2bf50e8dea	105499867526169935261	5/28/2013 10:04:41 AM	|User Action: SHARE|Payload: { "collection&q (...)
269	7456302a-8c68-4c0b-a69a-cd2bf50e8dea	105499867526169935261	5/28/2013 10:05:35 AM	|User Action: REPLY|Payload: { "collection&q (...)
270	7456302a-8c68-4c0b-a69a-cd2bf50e8dea	105499867526169935261	5/28/2013 10:12:34 AM	|User Action: REPLY|Payload: { "collection&q (...)
271	7456302a-8c68-4c0b-a69a-cd2bf50e8dea	105499867526169935261	5/28/2013 10:12:35 AM	try02b86268-ce79-4489-9291-7cef8116435b
272	7456302a-8c68-4c0b-a69a-cd2bf50e8dea	105499867526169935261	5/28/2013 10:14:15 AM	|User Action: SHARE|Payload: { "collection&q (...)

3. **The table where you are storing Access Tokens is very important for using OAuth,** it should resemble the following structure:



ACCESS_REQUEST Columns

Key	Column	Type	Length	Prec.	Scale	Nullable	Default	Rule	Id.	Id. Incr.	Id. seed	Row GUID	Actions
	REQUEST_ID	uniqueidentifier	16	0	0	False			False	0	0	False	
	USER_ID	nvarchar	255	0	0	True			False	0	0	False	
	ACCESS_TOKEN	varchar	MAX	0	0	True			False	0	0	False	
	ACCESS_TOKEN_EXPIRATION_UTC	datetime	8	23	3	True			False	0	0	False	
	ACCESS_TOKEN_ISSUE_UTC	datetime	8	23	3	True			False	0	0	False	
	REFRESH_TOKEN	varchar	MAX	0	0	True			False	0	0	False	
	CALLBACK	varchar	MAX	0	0	True			False	0	0	False	
	HOST_NAME	varchar	50	0	0	True			False	0	0	False	
	AUTH_CODE	varchar	MAX	0	0	True			False	0	0	False	



REQUEST_ID	USER_ID	ACCESS_TOKEN	ACCESS_TOKEN_EXPIRATION_UTC	ACCESS_TOKEN_ISSUE_UTC
30128e26-25b4-463f-827f-af35e9361aa4	105499867526169935261	ya29.AHES6ZR3QaWbBlAgex9apq4-a4jE6TA97nAUZcsT2wF-y (...)	5/26/2013 8:28:08 PM	5/26/2013 7:28:08 PM

REFRESH_TOKEN	CALLBACK	HOST_NAME	AUTH_CODE
1/bjLq1lTMSv2QZtTL-9LrfSVp48ZwnlTWMWea-bqbS7E	https://myglassapps.com/main.ashx	P3NW8SHG343	4/2VMAGTurYXMXcgpTIx81-z_g-Rkc.0sBfinHZlqceaDn_6y0 (...)

Note: when any future request comes in – you should check this table to see if an access token exists and is valid. If for some reason the user is – re-subscribing to your service, you should remove the users' old token and create a new one.

4. **If all else fails, read, read, and read https://developers.google.com/glass/develop/mirror/index – or contact me ☺ cbartholomew@g.harvard.edu**