Auth0 authentication for DesignAssist

# What is Auth0?

Auth0 is a third-party authentication service

# Using Auth0 Online Manager

For developer access, use manage.auth0.com to login with Google ([webmaster@tempest-tech.com](mailto:webmaster@tempest-tech.com) account). The dashboard on the initial login shows a snapshot of the activity of active logins.

## Applications and Settings

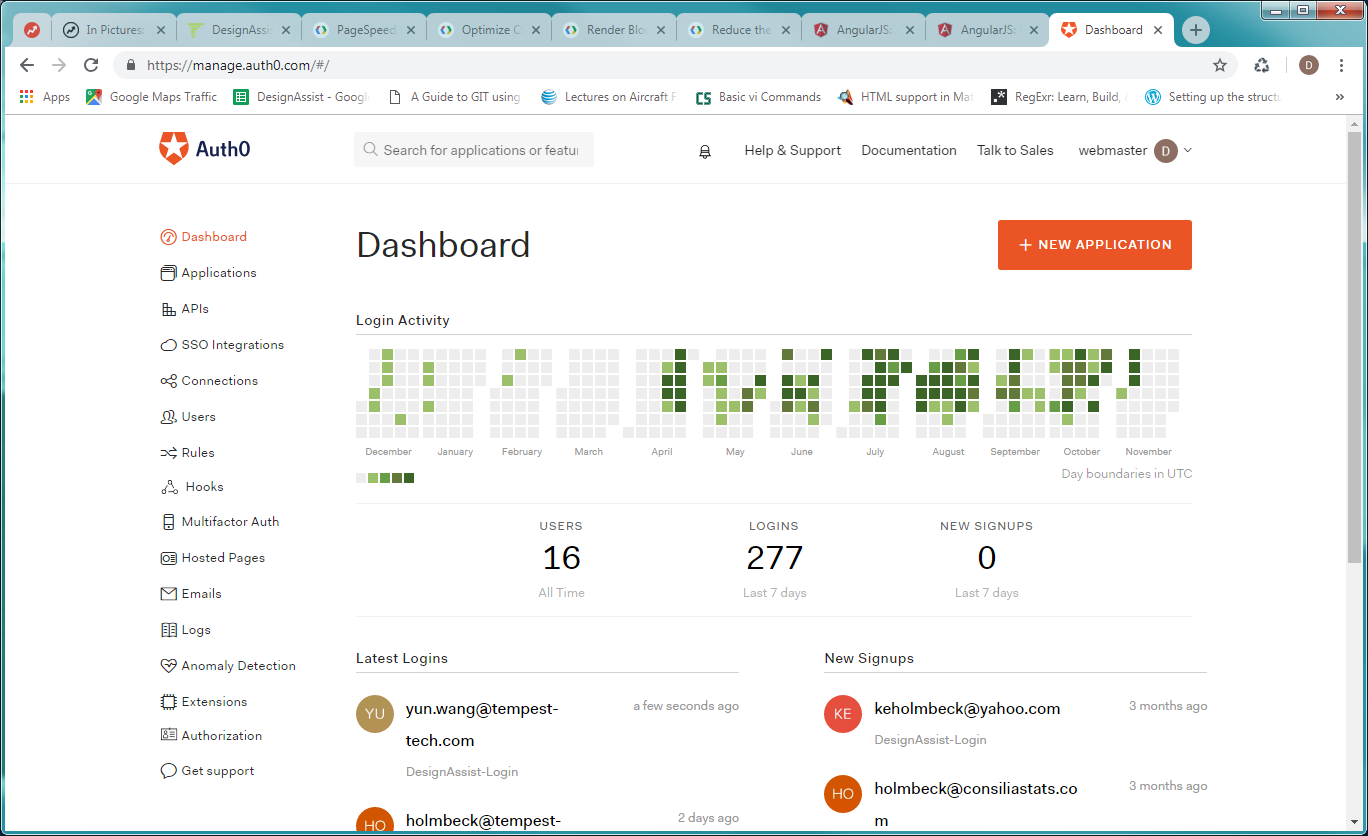


Figure 1: Auth0 Dashboard

In order to create our application, click on *Applications* in the sidebar. For our application, we created a *Regular Web Application* called DesignAssist. (It is important to note that while AngualrJS is a single-page application (SPA), we need to use Auth0 under "regular" web application for backend API access in Python, which we will cover in a later section.)

Upon generation, a client ID and secret key is created. We can access these anytime in the *Settings* section of an application. The purpose of these keys is to access Auth0 securely.

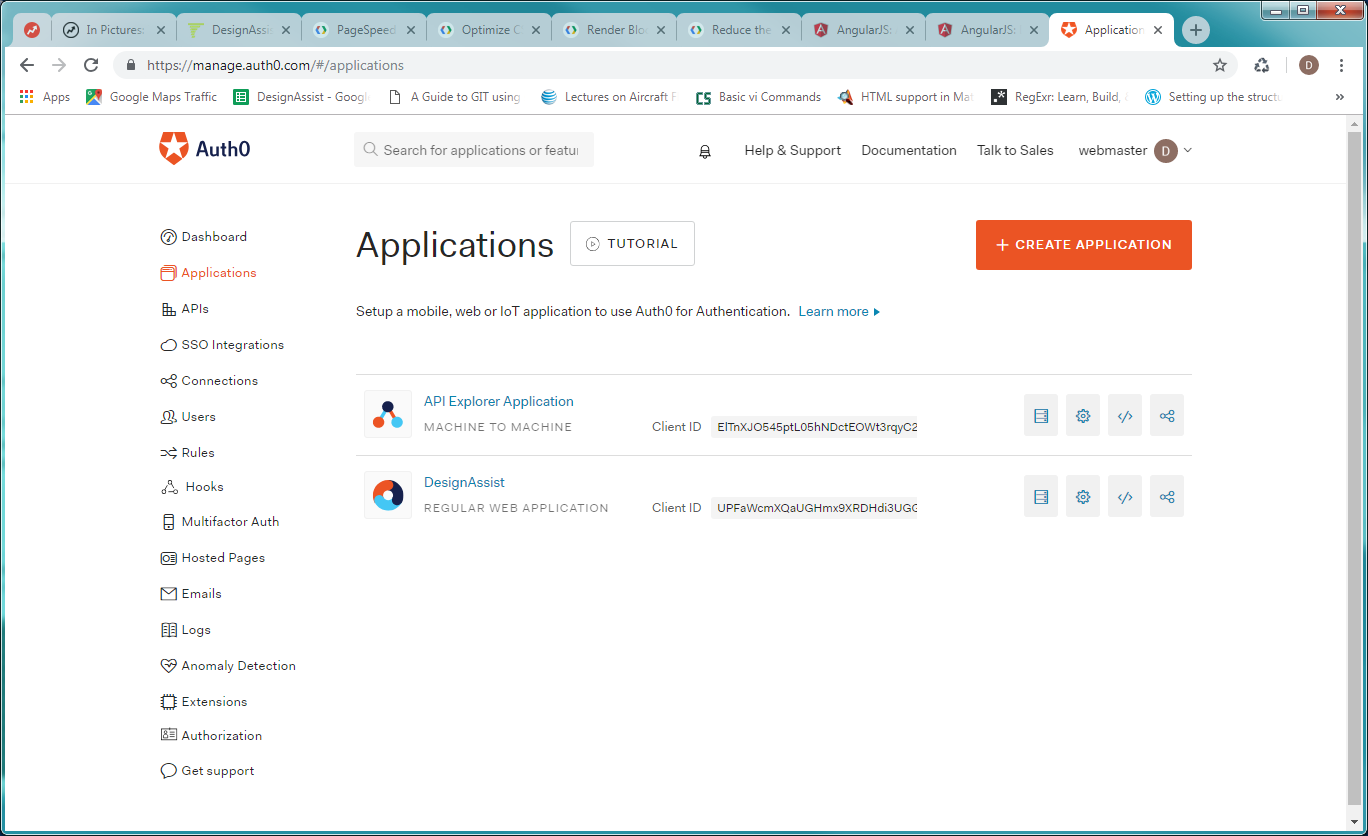


Figure 2: Auth0 Applications

In the Settings for this particular application, Allowed URLs and Origins indicate which website(s) can access Auth0 under our client credentials key. This prevents any site from using our Auth0 login, unless it's added to the list. For testing, we need to add localhost and any variants we use for the port number.

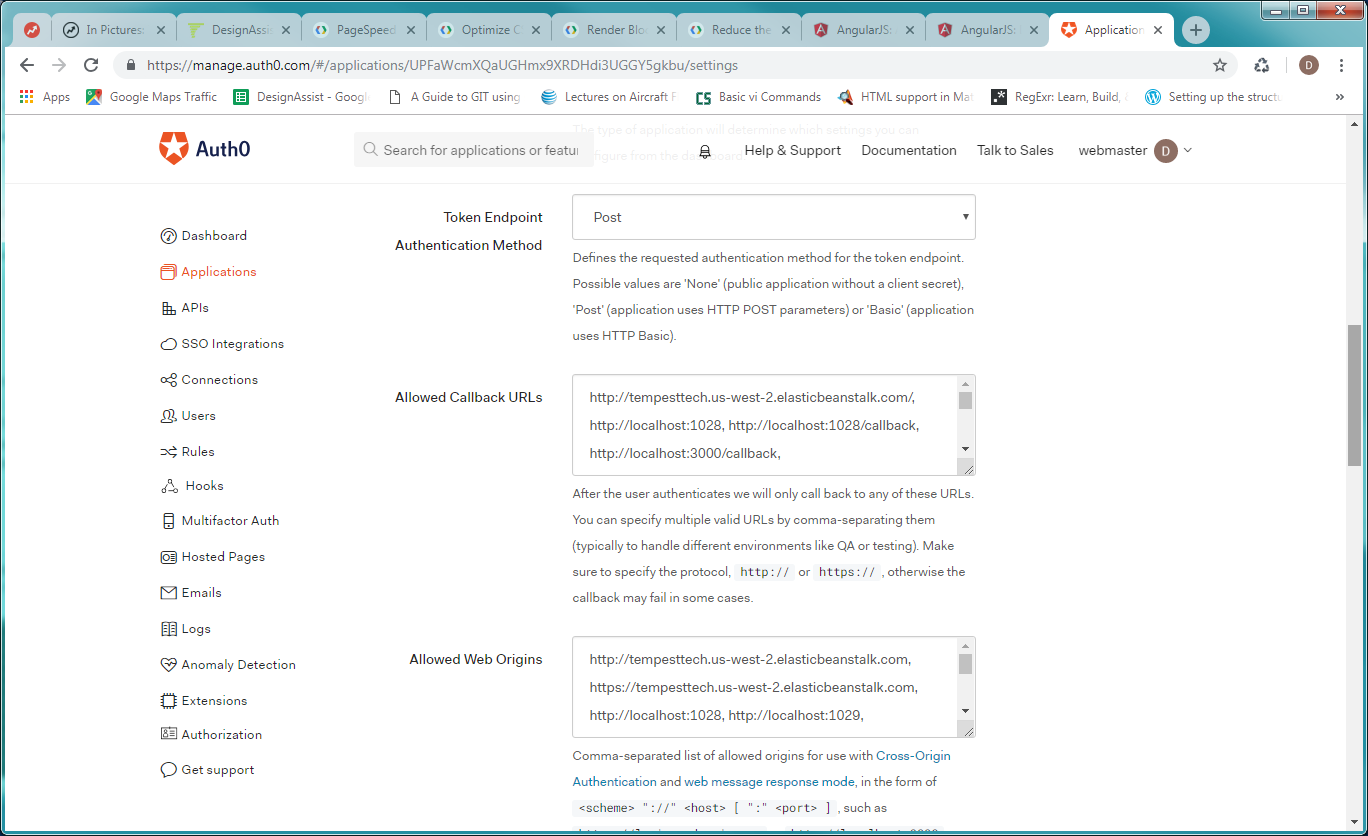


Figure 3: Auth0 Application Details

In the Advanced Settings section, the Grant Types must be set as follows for our application: Implicit, Authorization Code, Refresh Token, Client Credentials, Password, and MFA. Some of these are included by default any application creation.

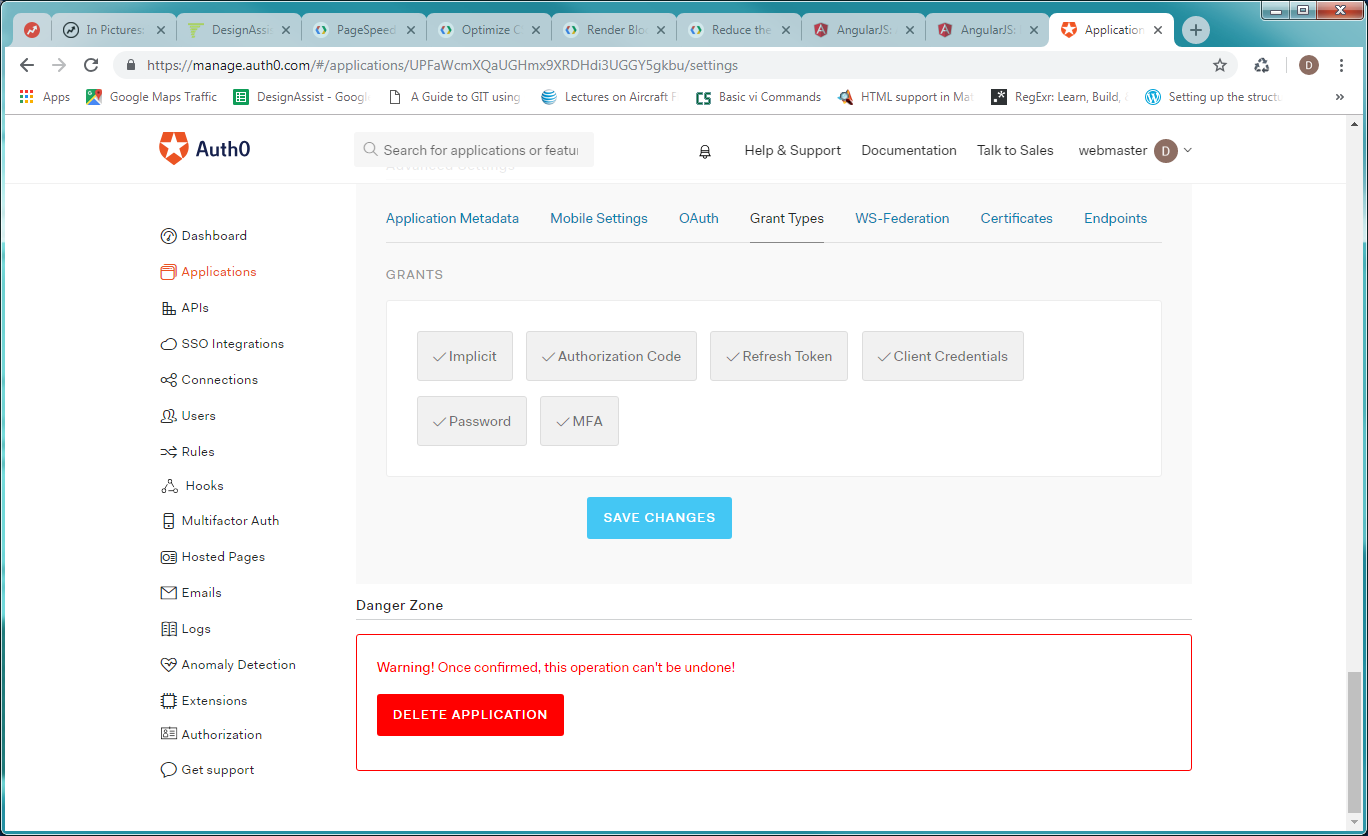


Figure 4: Auth0 Application Grant Types (Advanced Settings)

## Management API

The Auth0 Management API under the *APIs* section is used to call Auth0 programmatically through a POST method. This will be discussed further in section User signup through Python API.

## User Management

We can manage users individually through Auth0, including resetting passwords, viewing login history, and editing the user metadata which is important for our application. In general, metadata is additional associated data that we might want to attach. For our application, the metadata field is used to store the Stripe customer\_id field which is needed to access user payment and billing information. More details on Stripe can be found in the Stripe document.

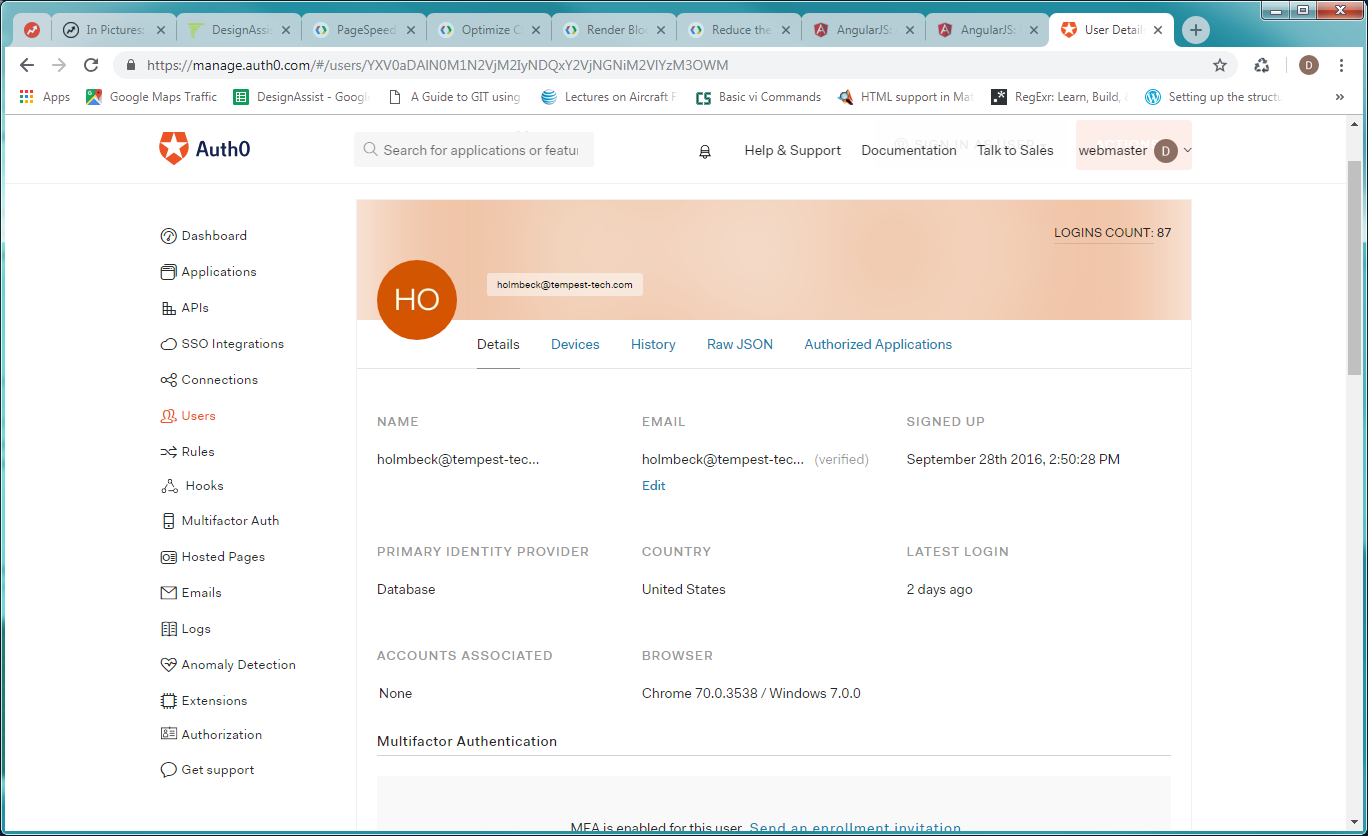


Figure 5: User Dashboard

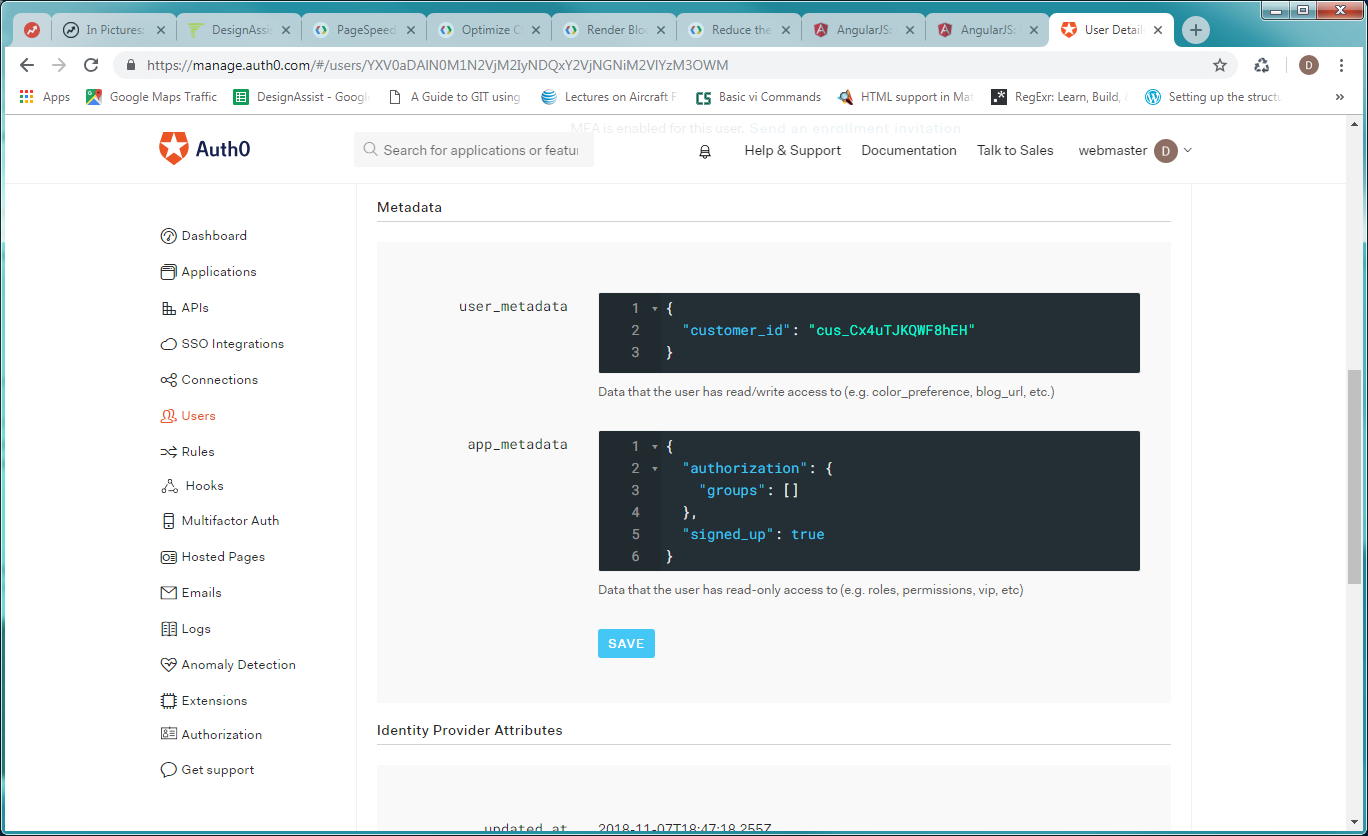


Figure 6: User Metadata

When a user must reset their password, Auth0 manages the communication by sending an email to the user. There are default templates, but we can also tell Auth0 to use a custom email template that includes, for example, our logo and contact information. This is done in the *Emails* > *Templates* section (Figure 7).

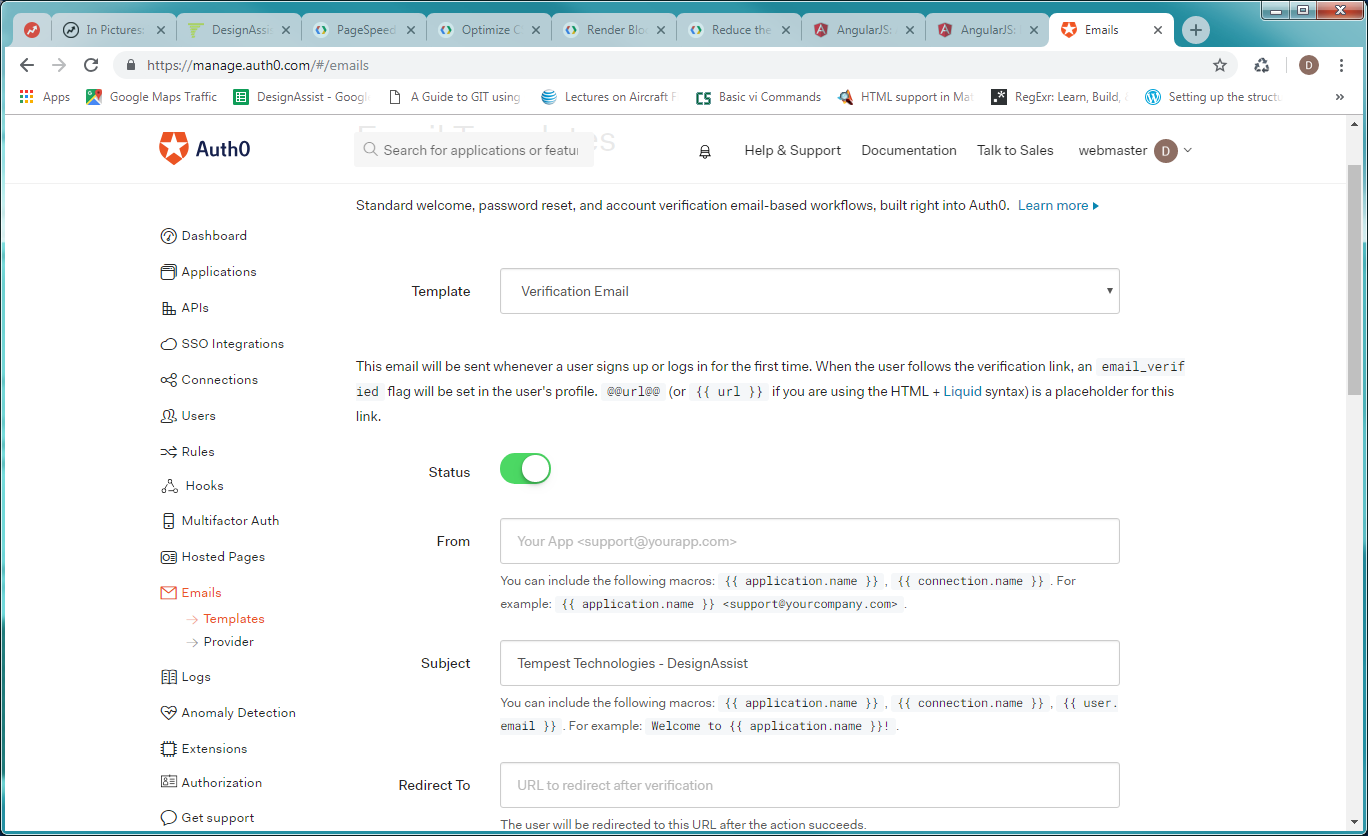


Figure 7: Auth0 Email Templates

## Change Email or/and Password

A user’s email and password could be modified by clicking the ellipsis next to the user.

# Auth0 Lock

"Lock is an embeddable login form […] recommended for use in single page apps" (<https://auth0.com/docs/libraries/lock/v11>). Due to using AngularJS, the most efficient method of user login is through Auth0's Lock library which we import into the Javascript portion of our application. The code implementation is located in the repository at: svn://NIDA\_web/branches/eggplant/app/static/module\_auth/authService.js.

The sign in page looks like:



# User signup through Python API

## Initialization

Upon payment, users also create a password for their Auth0 account, and the user registration and signup occurs through the Python backend in the Authlib *pip* package. In the initial customization, we must specify which OAuth method we are using, including providing the client ID and secret key information discussed in previous sections. (At the time of writing, Figure 8 shows the correct information required.)

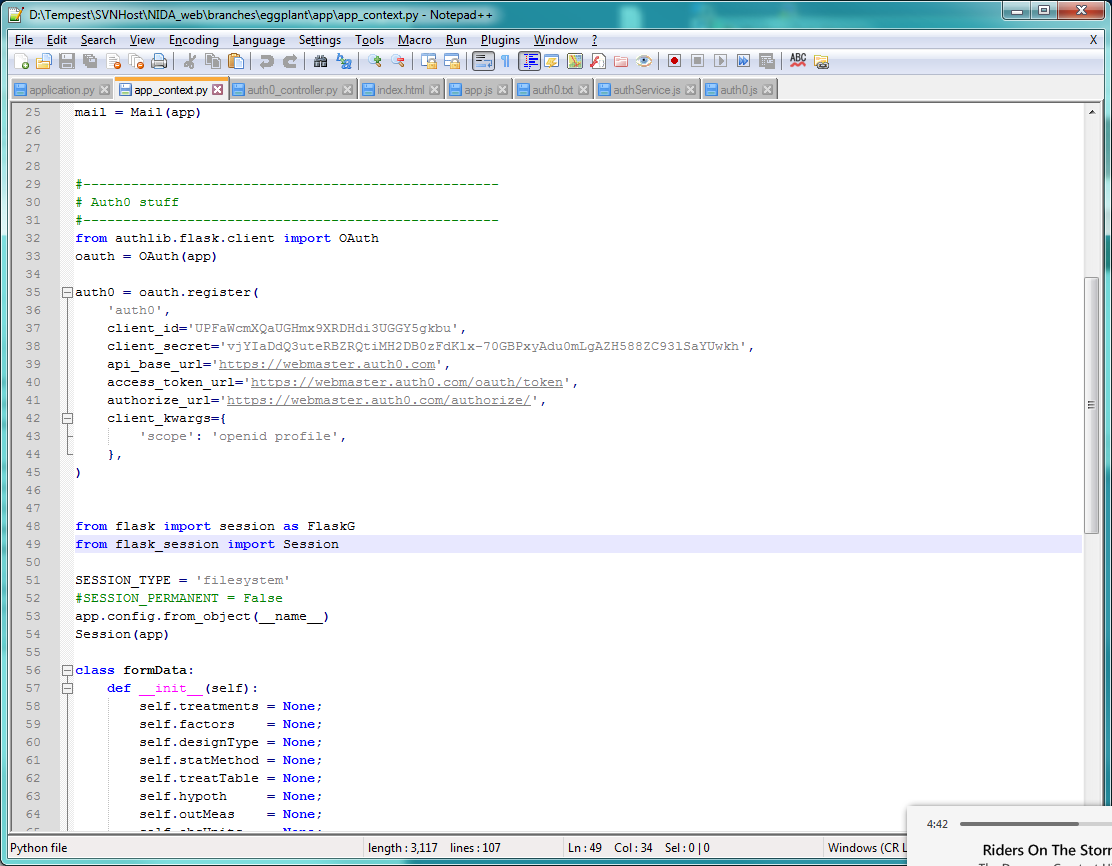


Figure 8: OAuth Flask initialization

After this information is set up, we can call the API and communicate directly with Auth0.

## Requests and Endpoints

The list below contains the types of requests that this application uses. Note that there are many other potential requests not outlined here.

|  |  |
| --- | --- |
| Request | Used when… |
| Find user by email | Check if user registered with Auth0 before payment / free trial. |
| Create Auth0 user | After payment successful |
| Change Auth0 password | Change password directly instead of using "Forgot Password" with Auth0 email. |

## Making a request

For more detailed information, refer to <https://auth0.com/docs/api/management/v2>. The basic procedure is that to post a request to the API, we need the (1) endpoint / URL, (2) the proper headers, and (3) the data or parameters to pass with the URL.

### Obtain an access token

In order to use any of the endpoints, an access token is required to access the Auth0 API. Before any request is made, we will call the get\_access\_token() function, shown in Figure 9.

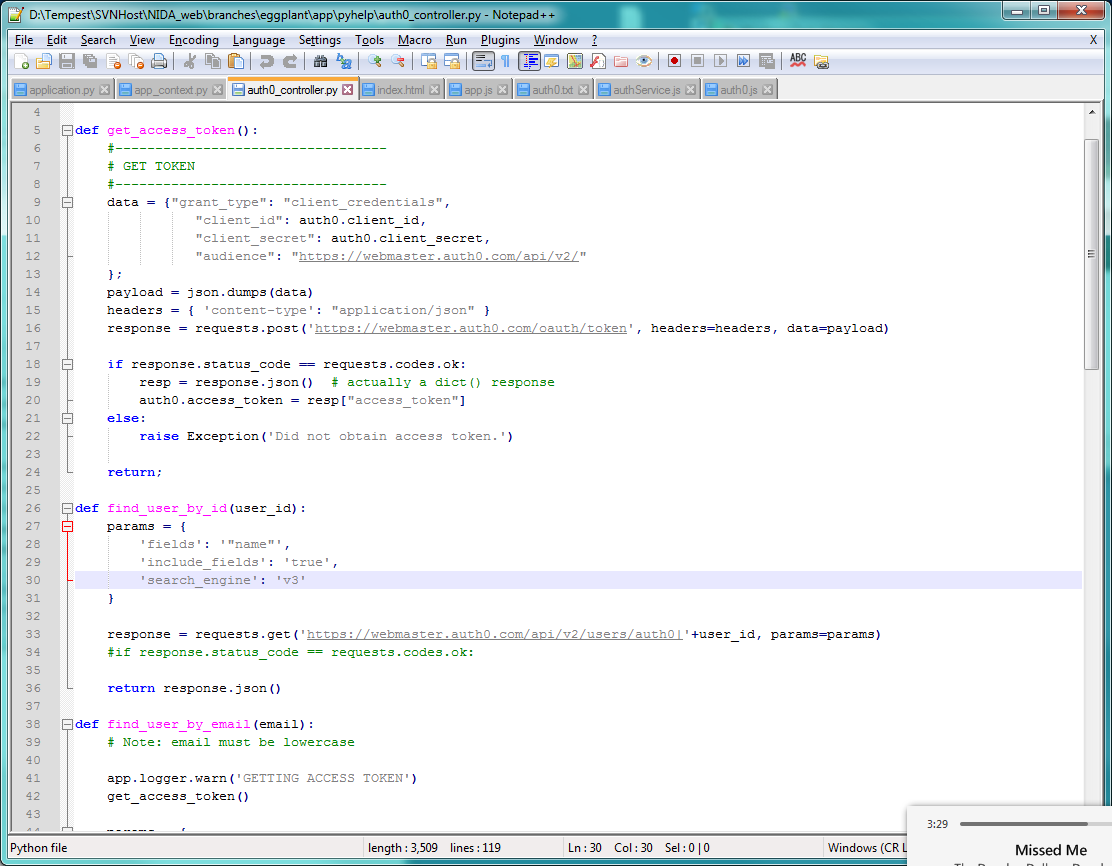


Figure 9: get\_access\_token()

### Creating a user

After the access token is obtained, we can make a request. In this particular section, we will outline the necessary requirements for making a request for user creation.

After passing the user name, email, Auth0 password, and any metadata (Stripe customer ID, for example), we can create a data object as shown in Figure 10.

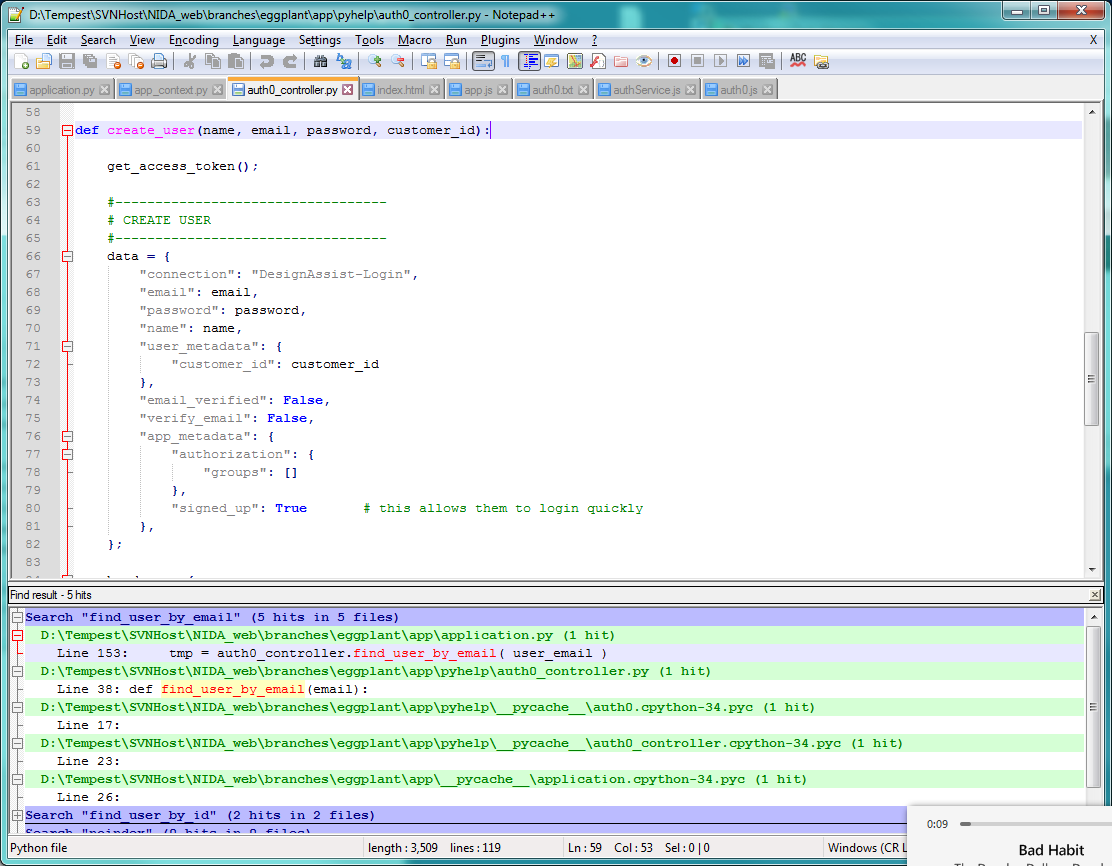


Figure 10: Data passed for Auth0 API user creation

Next, we require the proper headers which include the access token obtained in the previous section. After dumping the data object into JSON format, we can post our request.

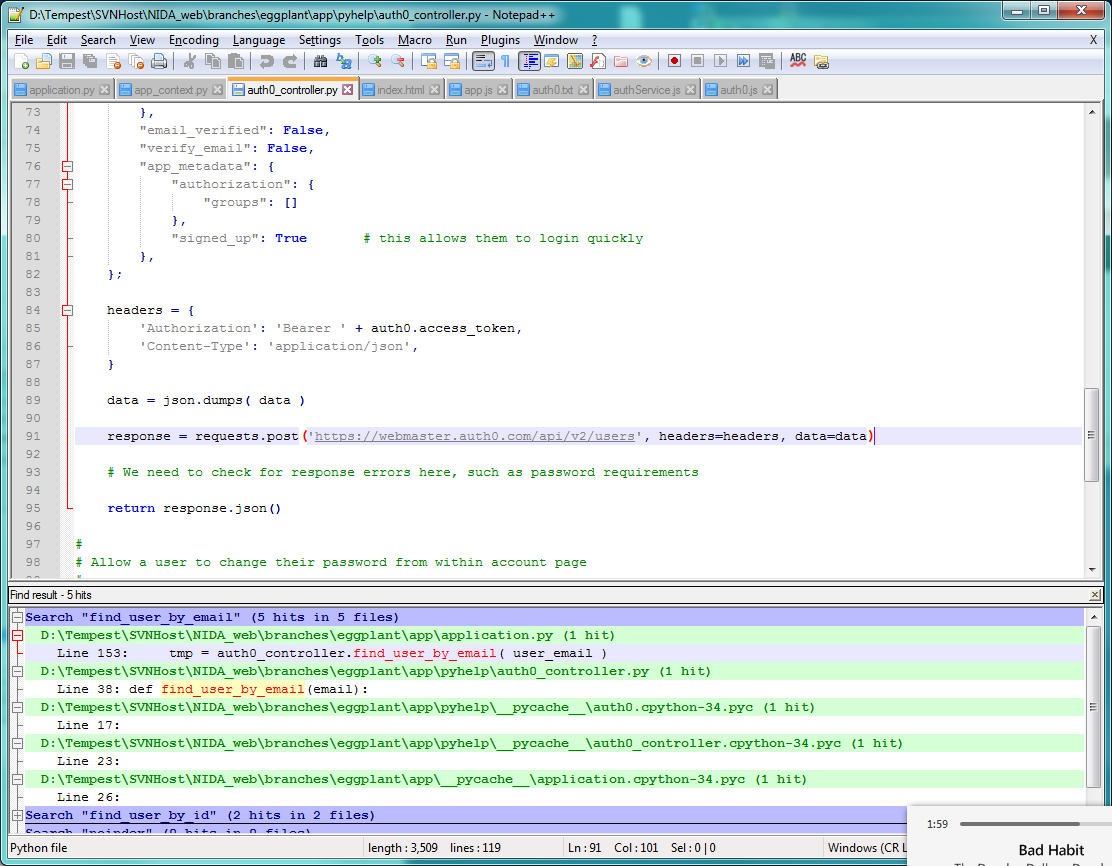


Figure 11: Posting request for Auth0 user creation

# Conclusion

In conclusion, we need two levels or methods of access to third-party Auth0 authentication: (1) Auth0 Lock, a Javascript plugin which works with AngularJS, and (2) the Authlib package to access the Auth0 API. We also briefly discussed how to navigate the Auth0 online dashboard for user management.