Data Sources for Advanced Programming Techniques

A Compilation of Available Princeton Sources of Information

Vinay Ramesh

Spring 2020

Independent Work Report

A Guide for Future COS 333 Students

Problem

The COS 333: Advanced Programming Techniques class provides its students with the unique opportunity to work in teams to collectively create a robust piece of software together. It is a distinctive chance in a computer science major's Princeton career to have real-world experience in software development which will help their future prospects of getting a job and performing well during summer internships. The project itself has several stages, outlined in the webpage https://www.cs.princeton.edu/courses/archive/spr20/cos333/project.html, which outlines the following considerations affecting the form of the project:

- Projects must have enough common structure to make it feasible for us to oversee a substantial number of projects, and to grade them fairly and uniformly.
- At the same time, the basic project should have plenty of room to try out interesting ideas, and freedom to use a broad spectrum of languages and tools.

- It must be relevant to the general themes and topics of the course, which are programming techniques, languages, tools, components and interfaces.
- It should be of some intrinsic interest and potential utility.

It comes as no surprise that one of the most important stages of creating this application is the planning stage. Groups must decide on a viable project idea that satisfies all of the above requirements and allows for novel creation in the realm of the expertise of the members of the group of a particular project. Over various semesters of the COS 333 course, groups have historically, as in with very high frequency, chosen project ideas related to enhancing the Princeton experience. For example, student groups have made projects around course selections (e.g. PrincetonCourses hosted on https://www.princetoncourses.com/), course scheduling (e.g. ReCal hosted on https://recal.io/landing), and room draw selection (e.g. http://rooms.tigerapps.org/map). A common issue during this planning stage is an inability to come up with a feasible project proposal that covers the group members' passions and/or beliefs, a belief that their project would benefit the Princeton community in a meaningful way. For any project group, past present or future, dealing with Princeton-related projects, a comprehensive list of "Princeton Data Sources" would be invaluable, which would be a complication of places to find information about Princeton such as public/private API's as well as documentation for how to us the API (if necessary). A list like this would make it much easier for future COS 333 students to decide what their project ideas will be. After all, if the data one has access to is more readily available, it would be easier to come up with an idea related to that data. Further examples of this

include dining hall menus, list and information of Princeton student clubs, the location of TigerTransit buses, art museum gallery information, etc..

The need for such a far-reaching list of "Princeton Data Sources" is apparent, a sentiment not only emphasized by this paper but also by previous students. In reaching out to previous project groups, comments like the following were common:

- "It seems like you're doing some really helpful work." (PrincetonPlants, Fall 2019)
- "I know something like this would have been very useful to my group." (FindItFast, Fall 2019)
- "This would have saved our project group so much time in finding an API and a topic!" (PrinceTour, 2018)

Further correspondence with these project groups will be outlined in the following "Chronological Narrative" section. The main takeaway from these findings is simple: the "Princeton Data Sources" list would save time for future COS 333 students in their project selection and API discovery, ultimately leading to more time in actual software development and less time in wearisome administrative matters including setting up meetings with select Princeton departments and writing dozens of emails to gain access to their data.

Chronological Narrative

The "Princeton Data Sources" list started with a search for an API for course offerings. In the Fall 2019 semester, I worked on a Princeton Course Planning mobile application, for which I used the WebFeeds Registry API (webfeeds.princeton.edu) to get this course information. From

course titles to professor information, this API allowed me to create a comprehensive course planner, through which students could browse through courses/professors, and create/edit custom schedules over various semesters.

This was a clear first API to add to the list of "Princeton Data Sources," however, upon browsing to the website, it returns a Page Not Found Error. After inquiring to Scott Karlin (Senior Manager of Computing Facilities) why this was the case, it was revealed that OIT is in the process of decommissioning the WebFeeds Registry, and so I was directed to George Kopf (Director of Software Infrastructure Services for OIT). A conversation with George Kopf disclosed that the legacy WebFeeds Registry has been replaced with a more modern API platform in which OIT can create custom APIs to meet COS 333 groups' exact needs. This new platform is called the OIT API Store (https://api-store.princeton.edu/store/). The OIT API Store provides a secure and organized way to consume Princeton-related APIs for COS 333 students, with endpoints protected by the OAuth2 security protocol. So far, the accessible APIs on this platform are called PrincetonInfo, ActiveDirectory, and MobileApp (more on the specifics of each API in the section "APIs in the OIT API Store"). George Kopf also iterated that himself and the OIT team would be happy to have a conversation with any student in COS 333 about what they may be looking for and how they can help (more on this in the following section "Working with OIT").

Additionally, if students are looking for any API that doesn't currently exist in the OIT API Store, then the API could be created. From this initial meeting with George Kopf, it was evident that it would be extremely useful to compile a list of desired APIs from both current and previous COS 333 students. The idea was to hand this list over to OIT so that they can begin

work on creating some of these APIs for the students. My search began by divulging into the COS 333 projects from previous years (https://www.cs.princeton.edu/courses/archive/spr20/cos333/projectprevious.html) in order to get an idea of which datasets students were able to get a hold of in the past. Below is a list of all groups contacted, and the information obtained from each of these groups:

- Clubhub (Fall 2019) - A COS 333 project that provides information about clubs on campus. This group indicated that there was no such API for student club/organization events. This is due to the fact that student clubs/organizations don't actually log its events into any particular system. Luckily, however, there is an endpoint in the API Store that retrieves event information for campus (not student clubs) events.
- FindItFast (Fall 2019) - Displays the locations of common utilities such as printers and water fountains at Princeton. This group also indicated that no such API existed. However, they provided the following websites from which they obtained certain pieces of data:
    - Emergency Phones: https://publicsafety.princeton.edu/safety-security/blue-light
    - Hotspots: https://hres.princeton.edu/sites/g/files/toruqf196/files/2020-01/hotspot-map.pdf
    - Printers: https://princeton.service-now.com/snap?id=kb_article&sys_id=ee132b1f4fe432408ef028928110c799
    - Filtered Water Stations: https://sustain.princeton.edu/resources/drink-local
    - AEDs: https://ehs.princeton.edu/emergencies-and-incidents/medical-emergencies-managing-injuries/automatic-external-defibrillators

    - Kitchens & Laundry Locations: https://drive.google.com/file/d/11KF0p16FDdGMwcz_U6LX_tYwo7k9KY2B/view?usp=sharing
- PrincetonPlants (Fall 2019) - Discloses information about the locations of every bush, tree, and plant on campus. This group indicated that they reached out to Facilities, who uses a third-party service name TreePlotter in order to obtain a static .csv file which gave them the necessary information about each plant (Plant name, Latin name, and description). Communications with this group proved to eventually reveal a way in which future COS 333 students may programmatically retrieve this information, rather than through a static .csv file (a more desirable method as the application would be up to date). More on this later in the "Other APIs" section of this document.
- Princeton ArtFinder (Spring 2019) - Conveys information about the locations plus descriptions of every piece of art not only in Princeton's Art Museum but also around the campus itself. Communications with this group revealed that an API for this indeed does exist. Created by Daniel Brennan, the API documentation is hosted on the following GutHub website: https://github.com/Princeton-University-Art-Museum/puam-api-docs. More on this API later in the "Other APIs" section of this document.
- CalcuLateMeal (Spring 2019) - Displays information about the price of every dish during late meal at Frist. This group indicated that although they believed that Dining Services has some database with foods/prices, at the time of their project they didn't have time to meet with the group. Therefore, the group decided go to Frist themselves and manually log the menu items/prices, including any deals during late meal time.

- PrinceTour (Fall 2018) - Gives any user a tour around campus by showing various landmarks in Princeton. In communicating with this group, it surfaced that such an API for landmarks around campus does not exist. The information about the landmarks coordinates, descriptions, and photos were all compiled manually. Below are those pieces of information:
    - Coordinates: https://drive.google.com/file/d/1g0bp38qmzCg4mf7-rxFpRCbDwDIFzSFQ/view?usp=sharing
    - Descriptions: https://drive.google.com/file/d/1y1I-kKbhA0Zk6JNtASdG7Y3Sl9it1CON/view?usp=sharing
    - Photos: https://drive.google.com/file/d/1_YK-4-KBUV4SOfPU-Hs88R4ZE2VGGKVo/view?usp=sharing
- PASS (Spring 2018) - Retrieves and displays information about Princeton alumni, from what major they pursued, to where they had internships, to where they are working in the present day. This group indicated to me that this data was obtained through Princeton Career Services, with the condition that they would not share or reuse the data for other projects. Therefore, should any student be interested in this type of information, they suggest reaching out to either Career Services or the Princeton Alumni Association.

After this arduous journey to uncover the various APIs used by past COS 333 students, I had a better idea of what sorts of information could be accessed. The next step was to survey the current Spring 2020 COS 333 class to see which Princeton-related APIs they wished were available, and/or some APIs they were already aware of. Understandably, there weren't too many responses to this survey, but I was still able to compile a few bits of information from it. The

survey is here: https://forms.gle/1kQra7XF3zFVgsYJ6. The following is a summary of the responses collected from the survey. As expected, there were more wishes for APIs than there were reports of known APIs:

Wishes

- Wish for an API for finding a list of members to different listservs. This would serve as a means to categorize what social groups people are in on campus. An API like this doesn't exist, for the reason that this information isn't public, so one will not be released in the foreseeable future.
- Wish for an API to query different events on campus. This API does indeed exist, as mentioned earlier in this section in the description of my interactions with Clubhub.
- Wish for an API that reports how many people swipe into which dining hall at what point. This information is possible to obtain, although I hadn't inquired into this. Later in this paper (section "Working with OIT") I will document how to conduct an inquiry into new Princeton-related APIs.
- Wish for an API to access student class data. This API exists.
- Wish for an API to return current classes offered. This API exists.
- Wish for a messaging API for messaging between students. This is not a feasible API that could be created in the future, as students could just email each other.
- Wish for an API for prices of different foods in venues in/around the University. An API like this doesn't exist just yet. Currently, students would have to screen scrape online menus on the

Princeton Dining Services website in order to obtain this information. More on obtaining an API like this in the "Working with OIT" section.

- Wish for an API to check menus for eating clubs. This would be an API that most likely wouldn't be made public. The more likely scenario is a student who is a member of a club, or who is friends with a member of a club, manually filling out the menu information.
- Wish for an API for accessing a student's Class Schedule from Tigerhub through Princeton CAS. This API will not be created, nor does it exist because of privacy reasons. The administration does not want to make this information public.
- Wish for a more suitable API for accessing dining hall menus. This group had the idea of screen scraping TigerMenus (hosted on https://tigermenus.herokuapp.com/) in order to obtain dining hall information. However, an API for this indeed does exist on the OIT API Store. More on this API in the "APIs in the OIT API Store" section of this document.

Known APIs

- There exists an API from the Mudd Library for accessing old thesis titles
- There exists an API that returns TigerTransit bus routes.
- There exists an API about housing information. I'm not sure that one like this actually exists to the general public. If a student wants this information, they should follow the steps outlined in the "Working with OIT" section.
- There exists an API with printer locations. This is actually not true. The printer locations obtained by the FindItFast group were from the Princeton Facilities website on a particular web page.

At this point, I was armed with plenty of information with which to proceed. My next step was to contact other Princeton departments in order to find out whether or not they had APIs of their own to share with the future COS 333 students. There were many departments that I had contacted, but many of them would not respond to my correspondence. The two departments that did indeed respond were The Princeton University Art Museum and Princeton Facilities. The overall goal was to either document any existing APIs these departments had and/or to create an API out of an existing dataset these departments had, which would hopefully be introduced into the OIT API Store.

The Princeton University Art Museum correspondence was through Daniel Brennan, the Art Museum's Application Developer. He indicated high interest in the OIT API store, and was willing to participate in the potential addition of one of their datasets to be hosted on the API Store. In order to initiate this process of bringing one of their datasets into the Store, I inquired to Daniel Brennan whether or not there was any data not publicly offered on the Github-hosted Art Museum API that he would like to be available to COS 333 students. It could be any private data that he doesn't want to be released to the general public, but that he would like future COS 333 students to have access to in order to create their applications should they choose to do an Art Museum related project. If so, the data could be available through the OIT API Store, and that OIT would be willing to work with him to create the API in order to make it available on their platform so that the information is streamlined. Unfortunately, Daniel Brennan's response was essentially no; most of the data the Art Museum holds back is either of a nature where it would always remain internal-facing (insurance values, storage locations) or relates to objects that the museum technically doesn't own and so there are some questions of legality to be answered at

the institutional level before they can make them available. Daniel Brennan then went on to state that this second group (the objects the museum doesn't technically own) will likely be made available for users on campus which would include the COS 333 students, but that he doesn't yet have a clear timeline on when the Art Museum will be ready to move forward with that. In light of this answer from Daniel Brennan, I requested a directory that lists the objects, makers, and packages (the items available in the Art Museum API. This will be further discussed in the "Other APIs" section of the document) in the format of (id, name). He in fact was able to send me a .json file, but with only the ids of all objects, makers, and packages and so I created a script that updated the directory in the format of (id, name), which is placed in my COS333-API-Examples Github repository (hosted on https://github.com/vr2amesh/COS333-API-Code-Examples).

The correspondence with Princeton Facilities was through Kristi Wiedemann and Jennifer Broome Chung. The both of them likewise also indicated high interest in the OIT API Store, and potentially incorporating some of their data into the Store. I had a phone meeting with these two administrators and attempted to initiate the process of bringing one of their datasets to the Store via an API. The phone call seemed promising, however it did seem as though this project wouldn't be something high on their priority list. In order to expedite the process, I sent over a list of datasets overseen by Facilities that could potentially be introduced into the Store. Here is the list below:

- Facilities Data
    - Location of every plant, tree, bush on campus
    - Energy consumption of each building on campus (note: currently an xml feed)
    - Frequency of building repairs

    - Type of building repairs
    - Quantity of landfill from each building (tons)
    - Quantify of recycling of consumer items from each building (tons)
    - Water usage of each building (gal)
    - Locations of filtered water stations
    - Locations of bathrooms – Possibly Building Services

- SAP metrics

    - Campus CO2 emissions (MT)
    - Campus water consumption (gal)
    - Campus landfill waste (tons)
    - Recycled soil from construction sites (%)
    - Recycled consumer items (%)
    - Recycled construction and demolition debris (%)
    - Area under enhanced stormwater management (acres)
    - Tracked forest area with improved habitat connectivity and quality (acres)

Kristi Wiedemann attempted to determine which datasets would seem to make the most sense to inquire about turning into APIs in the immediate future, based on her understanding of the status of the data, and wanted to run that by Jennifer Broome Chung and myself (please see green text in the above list). For example, several of the datasets for the SAP metrics in particular are "crunched" annually, and there is currently no automated feed/process, which seems like it would make it less practical to turn these into APIs at the current moment. Jennifer Broome Chung indicated that the plant, tree, and bush API could be relatively easy to accomplish, and that building repairs and energy data could be more complex in the type of data requested. Depending on the source of the data itself, the API could be created by either a third-party vendor or Facilities and subsequently shared with OIT in their API Store if approved by their leadership. As indicated earlier, this would be a project that would need to be prioritized and approved before work could begin on it. Given the other projects and priorities already underway

at the time, Facilities would have to get back to me to see what was reasonable for a potential timeline. Eventually, I was able to contact Jennifer Broome Chung's boss, Robert Kuhn, the Manager for Account Management and Delivery Business Services for Facilities. A meeting with Robert Kuhn and the third party vendor TreePlotter was scheduled in order to figure out if an API for every plant, tree, and bush on campus could be created. In fact, the TreePlotter vendor has an internal database that pinpoints all the plants, trees, and bushes by longitude and species name. With a username and password to enter their system, a potential COS 333 student could import this database into a CSV file, much like the group PrincetonPlants (Fall 2019) did for their project. Therefore, although an actual API does not exist for this plant, tree, and bush information, there does exist a mechanism to download the entire database. More information on how to exactly do this will be elaborated in the "Other APIs" section of this document.

Working with OIT

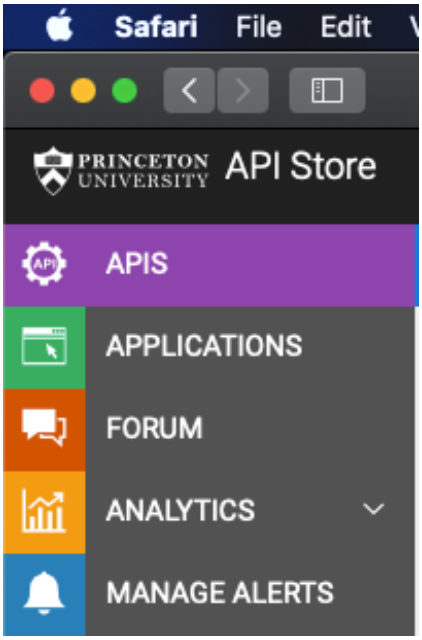
This section will cover working with OIT in the context of being a part of a COS 333 project group. There are several ways in which a student would want to communicate with OIT for their project. The first of which is obtaining a service account for the project. A service account is a separate Princeton netid which does not correspond to an actual student or faculty member, but rather is created to be linked with a particular application. It would be much better for a group to have shared login information for a service account rather than the group sharing the login information of one of its team members. Additionally, a service account can be made permanent, while student accounts expire after they graduate. Many Princeton-related APIs

require a user to use a netid in order to authenticate themselves as part of the Princeton community, including those APIs in the OIT API Store.

The best way to obtain a service account is to go to the following website: https://princeton.service-now.com/service?sys_id=f44539ab4ff81640f56c0ad14210c77c&id=sc_cat_item&table=sc_cat_item, and fill out the form. On it, make sure to do the following:

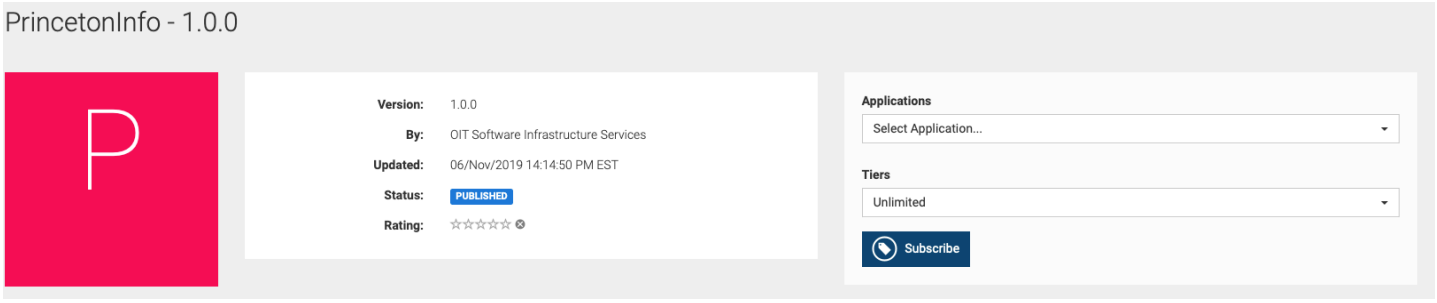
- Indicate that it's an account for General Use.
- Indicate the current COS 333 Professor as the sponsor for the account.
- Indicate the account will have a permanent duration (this will be necessary if the group intends to have their project "live on" beyond the duration of the COS 333 course).
- Indicate a business reason for the account, which in this case would be a project for the COS 333 course.

Now that a service account has been created, we can move on to consuming APIs in the OIT API Store. The API Store is hosted on this website: https://api-store.princeton.edu. In order to access this website, one must be either on the Princeton VPN or on the Princeton eduroam WiFi network. Login to the website with the Princeton CAS authentication using the service account you just created. Now, click the Applications tab on the left hand side of the screen (it should



be a green button), and edit the name of the default application into a name suitable for your COS 333 project by clicking the Edit icon. Then, click on the APIs tab on the left hand side of the screen (it should be a purple button), and you should see two APIs listed, ActiveDirectory and PrincetonInfo (more on what exactly is in each API in the following section "APIs in the OIT API Store").

Now we must subscribe to one of these APIs. Click on either one of these APIs, and then click the dropdown tab over to the right hand side of the screen (that says Select Application…) and choose the application name you just created. Then, click on the button "Subscribe" to successfully subscribe to the API.



Apart from the ActiveDirectory and PrincetonInfo APIs, there is one more API called MobileApp. This API gives information on courses, dining hall menus, events on campus, and places on campus that are currently open/closed. This API will not be seen at first, and you must ask OIT for explicit access to this API. In order to do this, send an email to George R. Kopf (or whoever the current Director for Software Infrastructure Services is) and ask him to add your service account netid to the approved accounts for the MobileApp API.

Now, let's say that after looking at the available endpoints in each of these APIs, your group decides that what you are looking for is not available in the OIT API Store. In this case, **OIT will be willing to work with you to see if they can add a new API for some other**

Princeton dataset, but START EARLY. In conversations with OIT, it was apparent that the administration has the desire to help out students in this capacity. Whether this is because OIT eventually desires control over all Princeton-related data is currently unclear, but what is known is that OIT is currently training many of its employees to develop APIs on the Store. In order to do this, first send an email to George R. Kopf indicating that you are a COS 333 project group looking for a new API. George would need to know a few things about your request:

- The full set of data fields you want to access.
- The full set of input parameters and whether they are optional or required.
- Permission from the data owner (if OIT isn't the owner of the dataset).

It could be possible that OIT does not look over the dataset that you desire, and so permission from the data owner would be required. Typically and if necessary, after the email to George R. Kopf, he can direct you to the appropriate administrator from which this permission needs to be acquired.

There are three scenarios that could come up when talking to this administrator:

- They grant you permission for access to the data, and OIT is allowed to gain access to their database in order to create an API for you in their Store. In this case, schedule a meeting (probably over Zoom) between George Kopf and the administrator in order to discuss further steps. In this meeting, be sure to indicate that the matter is time-critical, and is very important for the success of your project. This is why its important to start this process early in the semester!

- They grant you permission for access to the data, but they are unwilling to allow OIT to gain access to their database and create an API in the Store. If this is true, there could be a separate API for this information, which you should use for your project if it exists. If such an API does not exist, then you should request for special access to this information for the purposes of your project. However, make sure that you can gain access to the information relatively quickly, as there have been several project groups in the past who have been promised information that in the end was never given.
- They don't grant you permission of any kind to their data, in which case neither them nor OIT can help any further. If this is the unfortunate circumstance you find yourself in, then it is not recommend to pursue this dataset further. At this point, change your project idea and/or the dataset you intend to use for your project.

APIs in the OIT API Store

In the OIT API Store, assuming your service account has already gained access to the MobileApp API, there should be 3 APIs listed. In order to see sample code of how to consume these APIs, check out this Github repository: https://github.com/vr2amesh/COS333-API-Code-Examples. Before delving into the details of each of the endpoint of these 3 APIs, it is important to cover very briefly the security protocol used by the OIT API Store. The API Store uses the OAuth2 security protocol in order to protect their endpoints. This protocol includes the use of an access token which needs to be passed into the header of each request to the API. Below is a

```
import requests
req = requests.get(
    self.configs.BASE_URL + endpoint,
    params=kwargs if "kwargs" not in kwargs else kwargs["kwargs"],
    headers={
        "Authorization": "Bearer " + self.configs.ACCESS_TOKEN
    },
)
text = req.text
```

small code snippet of how to use the access token in the header of a request in python. The final value text represents the return value from the endpoint in string form. The variable kwargs is a dictionary of key word arguments that represent the parameters in the request. For example, if a request is made to BASE_URL + endpoint with the parameter fmt=json (in order to perhaps have the return value in json format instead of XML), then kwargs would be the dictionary {“fmt”: “json”}. The access token only lasts one hour, so it’s important to make sure it's up to date. In order to retrieve the up to date access token for your application, make a request to the following endpoint: https://api.princeton.edu:443/token. Below is a code snippet in Python to retrieve an access token for your application. In this case, kwargs should be the dictionary {“grant_type”: “client_credentials”} and the header includes the following base64 encoded string: CONSUMER_KEY + “:” + CONSUMER_SECRET. The sample code in the Github repository illustrates further how to use an up to date access token for each request made.

```
req = requests.post(
    self.REFRESH_TOKEN_URL,
    data=kwargs,
    headers={
        "Authorization": "Basic " + base64.b64encode(bytes(self.CONSUMER_KEY + ":" + self.CONSUMER_SECRET, "utf-8")).decode("utf-8")
    },
)
text = req.text
response = json.loads(text)
self.ACCESS_TOKEN = response["access_token"]
```

Coupled with the access token is the Consumer Key and the Consumer Secret. In order to get these values, browse over to the OIT API Store, and click the Applications tab. Then, click on the application name that you renamed earlier (from the default name). At this point, you should be able to see a series of tabs which are Details, Production Keys, Sandbox Keys, and Subscriptions. The Production Keys are meant to be used in a deployed application context, and the Sandbox Keys are meant to be used in a local development context. To start, use the production keys. Click the production keys tab and click the button that says “Generate Keys” in order to generate your Consumer Key, Consumer Secret, and Access Token. The Consumer Key and Consumer Secret values do not change throughout the duration of the application, but as stated earlier, the Access Token indeed does change every one hour. Therefore, the Consumer Key and Consumer Secret can be hard coded into your application code, but not the Access Token. Refer to the Github sample code for some examples on how to deal with these three values to consume the APIs on the Store. Please refer to the ReqLib.java/req_lib.py and Configs.java/configs.py files in particular in the ActiveDirectory, MobileApp, and PrincetonInfo folders.

Below is a list of the APIs available on the OIT API Store:

<u>ActiveDirectory:</u>

Base URL - https://api.princeton.edu:443/active-directory/1.0.2

- /groups - This endpoint returns all users that belong to a particular group on campus. By using this endpoint, you can see if a particular user is a part of a certain group. The only parameter of this endpoint is the following: name (name

of the group). The correct name of the group is necessary when using this endpoint. For example, one of the groups name is “Undergraduate Class of 2020”.

- /users/full - This endpoint returns information about a user within the Princeton community. This endpoint returns the full information about a particular user, far mar than the endpoint /users. The only parameter that the endpoint requires is the query netid. The parameter’s name is “uid”. The return value has the following information about the user: displayname (Full name of the user), universityid (PUID number), mail (user’s email address), pustatus (is the user a graduate, undergraduate, or faculty?), department (which department the user belongs to), eduPersonPrimaryAffiliation (whether the user is a student or faculty), streetAddress (office number and location if it is a faculty member), telephoneNumber (phone number if it is a faculty member), title (name of position at Princeton if it is a faculty member), eduPersonAffiliation (an array that shows all types of affiliation with the university. For example, faculty, employee, and student), departmentNumber (number of the department in which the user belongs), and memberOf (all groups on campus that the user is a part of. For example, Computer Science FacStaff, DuoEnabledAutomatically, or Office365ExchangeStandardEnabled, etc.).
- /users/basic - This endpoint also returns information about a user within the Princeton community. It returns much less information and the only parameter that the endpoint requires is the query netid. The parameter’s name is “uid”. The

return value has the following information about the user: displayname (Full name of the user), universityid (PUID number), and mail (user's email address).

- /users - This endpoint also returns information about a user within the Princeton community. The only parameter that the endpoint requires is the query netid. The parameter's name is "uid". The return value has the following information about the user: displayname (Full name of the user), universityid (PUID number), mail (user's email address), pustatus (is the user a graduate, undergraduate, or faculty?), department (which department the user belongs to), eduPersonPrimaryAffiliation (whether the user is a student or faculty), streetAddress (office number and location if it is a faculty member), and telephoneNumber (phone number if it is a faculty member).

PrincetonInfo:

Base URL - https://api.princeton.edu:443/princeton-info/1.0.0

- /department - This API is pretty straightforward. This is the only endpoint within the PrincetonInfo API. It returns a list of all of the departments within Princeton University, such as COS, ART, etc.. Each item in the list is a dictionary with key "dept" whose value is the name of the actual department. This endpoint does not take any parameters.

MobileApp:

Base URL - https://api.princeton.edu:443/mobile-app/1.0.0

- /courses/courses - This endpoint takes up to three parameters: term, subject, and search. Term is a required parameter. The other two parameters work a slightly different way. You should only provide one of these two parameters in order to make a valid query to this endpoint. If you provide only the subject parameter, the endpoint will return all courses within that subject for the term. That is, if for example you pass in “COS”, all classes within the COS department will be returned. Keep in mind that this must be all capital letters. If you provide only the search parameter, the endpoint will query all courses in the Registrar and return all courses which match the search query. For example, a search value of “intro” in either the Course Title, Course Description, Professor Name, or the Course Department Code. If both the subject and search parameters are provided, then the endpoint will return an OR of the two. That is, a course will be returned if EITHER it matches the subject parameter OR the search parameter.
- /courses/terms - This endpoint will return information of the current term. Each term in the return value has the following parameters. Code (the id number of the term according to the Registrar), suffix (formatted version of the term as such: TermYear [e.g. S2020, F2019, F2018, etc.]), name (formatted term as such: [e.g. S19-20, F19-20, F18-19, etc.]), cal_name (formatted term as such: Term Year [e.g. Spring 2020, Fall 2019, Fall 2018, etc.]), reg_name (formatted term as such: Years Term [e.g. 19-20 Spr, 19-20 Fall, 18-19 Fall, etc.]),

start_date (start date formatted YYYY-MM-DD), and end_date (end date formatted YYYY-MM-DD).

- /dining/locations - This route does not return data back in a JSON format, but rather in an XML format. This route returns dining locations along with its latitude/longitude information, payment options, building name, etc.. The only parameter this endpoint takes is categoryID, a type of dining location (some categories include 2: dining halls, 3: cafes, 4: vending machines, 6: shows amenities of each hall on campus such as printers, Mac clusters, scanners, and wheelchair accessible hallways).
- /dining/events - This endpoint on the MobileApp API returns an iCal stream (essentially just txt) of dining venue open hours. The only parameter is placeID, which is an id number given to a particular place on campus. Experiment with different placeID values to learn which placeID values correspond to which places on campus.
- /dining/menu - This endpoint returns dining menus. The parameters are locationID (which dining hall you are querying) and menuID (Breakfast, Lunch, or Dinner formatted with the current date in the format YYYY-MM-DD-MEAL). The return value is a list of each food item in the menu with the following parameters: id (id number of the food), name, description, link (url to menu item on Princeton website), and icons (vegan, vegetarian, etc.).
- /places/open - This endpoint returns information about all places on campus that are currently open/closed. The endpoint takes no parameters. For each place that

is returned in the return value, the parameters are name (name of the place), id (unique id number of the place), and open (indicates whether or not the place is open. Not a boolean, it is a text value “yes” or “no”.

- /events/events - This endpoint on the MobileApp API returns an iCal stream (essentially a txt stream) of dining venue open hours. The parameters accepted are “from” and “to”. The “from” and “to” parameters are dates formatted in the following way: YYYYMMDD.

Other APIs

Other APIs are also available to the COS 333 students. This section will cover two of those APIs as well as an additional data set that one could import into their computer programmatically. These data sources are all Princeton-related data sources: Princeton Art Museums’ API, Princeton TigerBook API, and a dataset that shows all plants, trees, and bushes on campus through Princeton Facilities. Each of these “Princeton Data Sources” has its own separate security protocol and method for consuming the API. Below is some documentation on the endpoints of these datasets. For further code samples, please refer to the Github repository: https://github.com/vr2amesh/COS333-API-Code-Examples.

Princeton Art Museum API: This is a public API, so there is no security protocol to be aware of in order to consume it. This API is very well documented in the Github repository page: https://github.com/Princeton-University-Art-Museum/puam-api-docs. Therefore, below are simple explanations of each of the endpoints in the API. In order to know which object ids, maker ids,

and packages ids refer to which items, refer to the files objects.json, makers.json, and packages.json in the ArtMuseum folder of the following Github repository: https://github.com/vr2amesh/COS333-API-Code-Examples.

BASE URL - https://data.artmuseum.princeton.edu

- /objects/{ObjectID} - Returns information related to objects in the Princeton Art Museum's collection. An object is any art piece that is within the Art Museum itself, or any art piece that is around the Princeton campus.
- /makers/{ConstituentID} - Returns information related to makers in the Princeton Art Museum's collection. A maker is any painter, sculptor, or architect that has art work on the Princeton University campus.
- /packages/{PackageID} - Returns information related to packages in the Princeton Art Museum's collection. A package is any collection of objects in the Art Museum, categorized by some common property. For example, all East Asian Ming Dynasty art could be one package, and all Spanish Renaissance Art could be another.
- /search - One can use this search endpoint in order to search for objects according to their type among other things. The parameters are as follows: q (query string), type (this is the type of object, which can be either artobjects, makers, packages, or all).

TigerBook API: The TigerBook API is based on the website TigerBook: https://tigerbook.herokuapp.com/. This API returns information about undergraduate students at

Princeton, such as Major, Major Type, Full Name, Residential College, and link to their photo in the TigerBook website. This API is documented on the following Github website: https://github.com/alibresco/tigerbook-api. In order to consume this API, it is recommended that you first have already obtained your service account. The sample code uses the service account cos333_fall2020. First, ensure that you are logged into CAS with your service account. Then, browse to this URL: https://tigerbook.herokuapp.com/api/v1/getkey/{agent}. This agent should be something related to the application name; the sample code used the agent cos333APIcodeExamples. After having browsed to this URL, the browser should show a random 32-character key. Save this key in your code, you will need it for all subsequent requests made to the API.

Now, in order to make a proper request to the API, a header must be properly formed that follows the X-WSSE security protocol. Consult the following code snippet to see how to properly form this header in Python. USERNAME is the netid of the service account, AGENT is the agent value you inputted when browsing the /getkey/{agent} URL for the key, and KEY is the key obtained from the URL:

```
def genheaders():
    created = datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%SZ')
    nonce = ''.join(
        [
            random.choice('0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+/=')
            for i in range(32)
        ]
    ).encode("utf-8")
    username = USERNAME + "+" + AGENT
    password = KEY
    generated_digest = b64encode(
        hashlib.sha256(
            nonce + created.encode("utf-8") + password.encode("utf-8")
        ).digest()
    )
    return {
        'Authorization': 'WSSE profile="UsernameToken"',
        'X-WSSE': 'UsernameToken Username="%s", PasswordDigest="%s", Nonce="%s", Created="%s"'
        % (username, generated_digest.decode("utf-8"), b64encode(nonce).decode("utf-8"), created)
    }
```

This API has two endpoints. Full documentation on https://github.com/alibresco/tigerbook-api:
Base URL - https://tigerbook.herokuapp.com

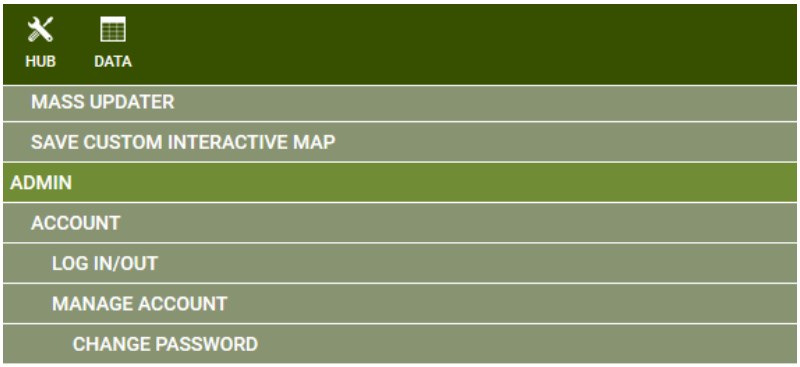
- /api/v1/undergraduates/{netid} - Returns a JSON dictionary representing the queried student. Consult the TigerBook Github repository for the dictionary fields.
- /api/v1/undergraduates - Returns a JSON list of dictionaries, with each dictionary representing a student. The JSON will contain information about every undergraduate student at Princeton. Consult the TigerBook Github repository for the dictionary fields.

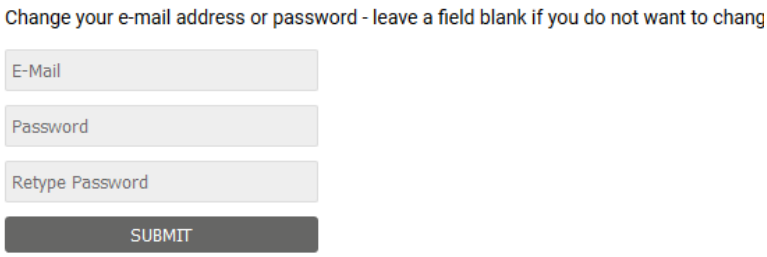
Plants, Trees, and Bushes: This is not an API, but rather a place from which to obtain the Princeton groundskeeping internal database. The database can be imported from the third-party vendor TreePlotter's website as a CSV file into your local computer to be used in whichever way your group sees fit. The CSV file gives the following information about the plants, trees, and bushes: address, common name, date planted, genus name, geometry, latin name, coordinates, species, and current status (alive or dead). If your COS 333 group wishes to use this information in their project, you must follow these steps below:

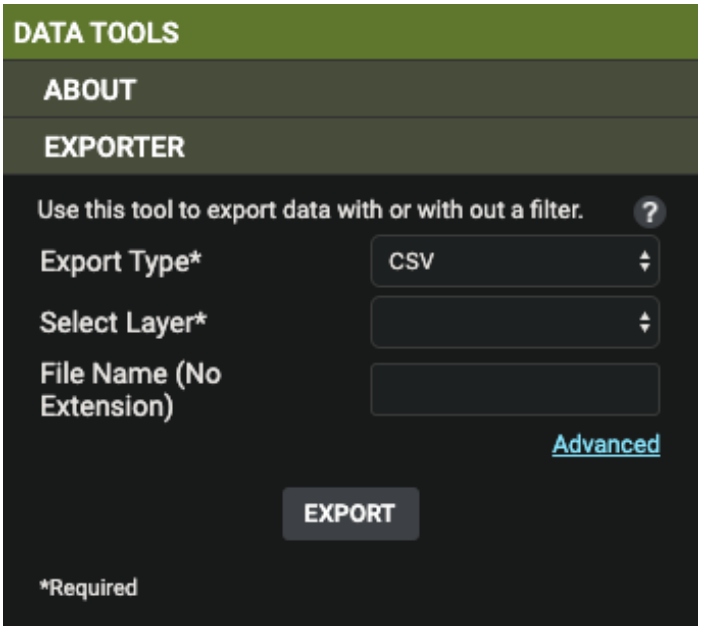
- Send an email to Robert Kuhn, the Manager for Account Management and Delivery Business Services for Facilities, and request login information for the following website: https://pg-cloud.com/PrincetonTreeMap/. Make sure to indicate that you are doing this for the purposes of a COS 333 project.

- Once you receive the login information, please update your password after first login from the Hub on the left-side. Select Admin, Account, Manage Account, and then Change Password.

- After you update the password, click the Hub button on the top left corner. Then, select Data Tools in the dropdown, and then select Exporter. Choose the CSV file export type, select whichever layer you wish, and write a file name you'd like to call the CSV.







You could work around this limitation by programmatically following the steps outlined above. That is, programmatically logging in, clicking the export button, and dealing with the CSV file. One could accomplish this by inspecting the elements of the HTML page, and determining which buttons on the page need to be clicked in order to export the CSV file. These buttons could be clicked programmatically using JavaScript and tools such as JQuery. It is a unique programmatic challenge to figure out how to do this, and so it is left up to the COS 333 group if they wish to implement this feature.

If you decide not to implement this programmatically, then it is possible to just download the CSV file once, and use this "snapshot" of the database for the entirety of your project. However, keep in mind that it would mean that the database is not up to date. An application that periodically retrieves this CSV file would indeed have an up to date database, and would result in a more robust application.

Conclusion

In conclusion, this document shows the various Princeton-related APIs that are available for COS 333 projects. My work this semester will hopefully make it much easier for future COS 333 students to not only have an easier time finding these APIs, but also save time in finding them. After all, these students' times are much better spent developing their applications rather than searching and inquiring about these APIs. It is much better for them to have these APIs at their fingertips, ready to go. My work this semester not only will help future COS 333 project groups, but I also did work to help the current COS 333 project groups.

Throughout the semester, I was active on the COS 333 Piazza page for any issues on Princeton-related APIs. Several student groups this semester had questions on how to access Princeton Student Data. The answer was to lead these students to the existing TigerBook API (Please consult Piazza Thread @153). I made it clear to these students that the information on this API is publicly available (to those in the Princeton community), and that certain information about students will never be available due to privacy and security reasons (such as course enrollment information, and recently even hometown, personal phone numbers, dorm room on campus, etc.). Several students posted many follow up questions to this thread ranging from questions about functionality of the API, to how to form the X-WSSE headers in order to make valid requests, to how to make network requests in Python.

Additionally, Piazza Thread @203 surrounded around helping students with obtaining service accounts for their projects. On it, I outlined the steps necessary to getting one of these service accounts. In my own experiments in obtaining a service account, I was able to get it without indicating Professor Robert Dondero as the sponsor. A subsequent call with George Kopf had backed up this detail. So in this Piazza Thread I initially reported that one could self-report as their own sponsor. Later on in the course of the semester, students reported that their requests for student accounts were getting rejected due to the fact that students indeed were NOT allowed to self-report as their own sponsor. Out of all of this, ultimately, we were able to come up with an organized protocol through which a COS 333 student may obtain a service account. Professor Dondero proposed this protocol in an email exchange with Russell Wells, in which I was also involved in order to represent the point of view of a COS 333 student. This protocol should

expedite the process of service account retrieval for future semesters, and should increase student awareness of the existence of these service accounts in the first place.

The other Piazza posts I was involved in revolved around Princeton CAS. In these posts, I made it clear to the COS 333 students that one cannot receive First and Last name information from the CAS /login route. If they wish to get this information, they should use a different API such as the ActiveDirectory in the OIT API Store or the TigerBook API.

My contributions to the COS 333 went further than just Piazza as well. Several students reached out to me via email inquiring about help with their respective projects. Here is a short compilation of the groups that reach out and the help I was able to provide:

- Devin Plumb: Inquired about whether or not an API that reveals a student's PU prox barcode number exists. I informed him that one indeed does not exist, as this is private information unavailable to the public Princeton community.
- Thomas Johnson: Created an API client that, given a Princeton netid, determined if the user was an undergrad, graduate student, or faculty member. While I thanked Thomas for his great work on creating the client, I informed him about the OIT API Store, and that this information is obtainable through the ActiveDirectory API mentioned earlier.
- Vikash Modi: Was using the TigerBook API for his project. However, was unable to get it working in the Heroku server application context, even though he could get it working in the localhost context. For this student, I had an email exchange (Professor Dondero was CC'd in this chain) in which I explained how

to use the TigerBook API, from initially getting a key using an agent to consuming the API in an application context on Heroku.

- Rohan Joshi: Inquired about whether there was an API that returned all the courses a particular instructor at Princeton teaches. I informed this student that although such an API with this specific endpoint does not exist, the MobileApp API on the OIT API Store returns the courses within a given semester, which shows who the instructor is for each course. Rohan would need to extract the data of ALL courses in the term to a local database, and then run a custom query against the local database that retrieves the courses of a particular instructor. I additionally had a Zoom meeting with him and OIT, in which I gave some of my insight on how to obtain a service account and consume the MobileApp API.

The “Princeton Data Sources” will prove to be an invaluable asset to the COS 333 experience. One long-term goal is that through many semesters of students working with OIT (following the steps outline in “Working with OIT”), the API Store will grow to host more APIs and future students will have ready access to more information via the Store. Ideally, once the amount of APIs in the Store is large enough, future COS 333 students will contact OIT and other departments less frequently for these datasets, as they will already be available to them on the OIT API Store. Ultimately, this should result in more creative and interesting projects with groups that have more time to work on development rather than mundane administrative matters.

Acknowledgements

- Professor Robert Dondero - Thank you for all your assistance and guidance during this semester. I have enjoyed my time working with you throughout my Princeton career, which started all the way in Sophomore Year in COS 217!
- OIT - Thank you for working with me and introducing me to the OIT API Store. I sincerely appreciate the efforts OIT put in to help future COS 333 students.
- Kristi Wiedemann & Jennifer Broome Chung - Thank you for working with me to develop a way to obtain information about all the plants, tress, and bushes on campus.
- Daniel Brennan - Thank you for explaining the Art Museum API to me, and providing me with the barebones directory for the objects, makers, and packages in the Princeton University Art Museum collection.
- Previous COS 333 Project Groups - I thank all the groups that had correspondence with me during the semester.
- Current COS 333 students - I thank the current students for participating in my “Data Sources” survey. It provided valuable insight into the issues and needs while developing the “Princeton Data Sources”.