

# **Tigerbook V2: An Extensive Student Directory Framework For Princeton University**

Ben Chan

Adviser: Jérémie Lumbroso

## **Abstract**

*This paper details the design and implementation of Tigerbook V2, an extensive student directory framework designed specifically for Princeton University. Building on the features of the original Tigerbook but with emphasis on privacy, this platform aggregates opt-in information from undergraduate students about their hometowns, current cities, extra-curriculars, interests, and research, providing a comprehensive resource for finding and connecting with other students who share similar backgrounds and interests. The directory has a user-friendly interface and is easy-to-navigate, with robust search tools that enable students to filter results based on a variety of criteria to find profile of other students. While the directory is only for Princeton undergraduates, there is the potential to extend the platform to include Princeton alumni and graduate students in the near future. Overall, Tigerbook V2 aims to foster a stronger sense of community among students and to start conversation among each other.*

## **1. Introduction**

Tigerbook V2 has been developed with six key features in mind, each designed to provide a beneficial cause to the Princeton University student body. One of the primary features of Tigerbook V2 is that it is not just a student directory. In addition to finding other students, users can also search for interests, extracurricular activities, and more. To continue improving the platform, users can suggest new categories through a prospective feature of an easy-to-fill form to add to the compendium.

Another important feature of Tigerbook V2 is its opt-in system. The platform balances privacy and valuable information by requiring user consent before sharing any information. Users have control

over access to their information, with the ability to prohibit all staff, all faculty, or particular students from seeing certain attributes of their profile. Tigerbook V2 does not share any user information with third-parties that are not affiliated with Princeton University and is built with awareness to the third-party software involved in the platform, such as opting for a custom map implementation instead Google Maps to visualize hometowns and current cities of Princeton students.

The platform also includes a smart search bar that allows users to search by PUID, email, full name, and Princeton NetID with dynamic search results. In addition, the interactive map feature of Tigerbook V2 enables users to visualize the hometowns and current locations of undergraduates around the world, making it easy to find and connect with other students over breaks or after graduation.

Finally, Tigerbook V2's prospective reverse-search feature enables users to see who and how many students belong to a particular extracurricular activity, share a particular interest, are in the same certificate program, or live in the same housing building, among other possibilities. These six key features work together to create a comprehensive platform that is designed to meet the needs of Princeton University students.

## **2. Background and Related Work**

### **2.1. Tigerbook V1 History**

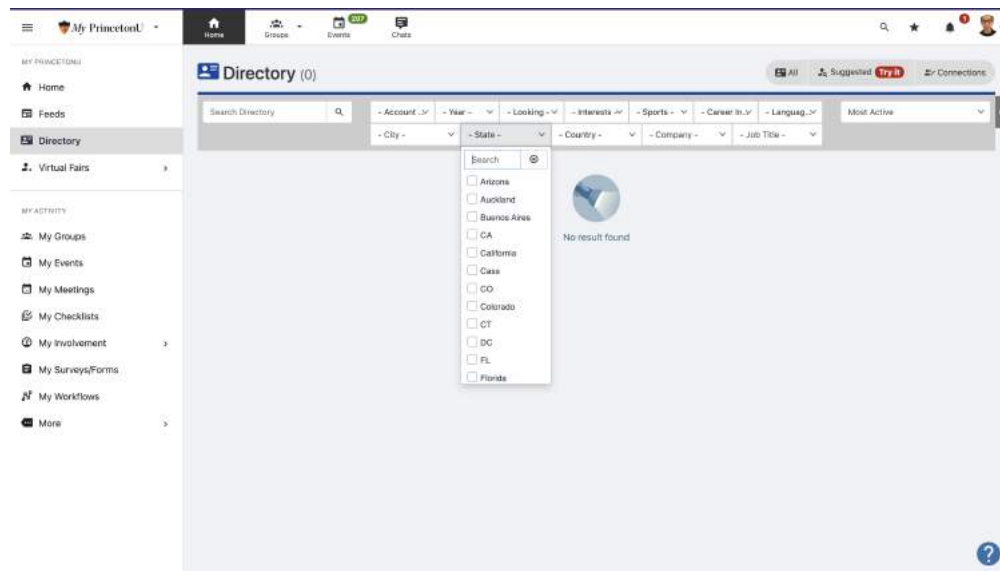
The original Tigerbook was a directory application that aimed to help students connect with their peers based on their backgrounds. By storing information such as hometowns, roommates, and dorm room numbers, Tigerbook V1 enabled students to connect with others more easily; however, the application faced a major setback in September 2019, when a privacy update was implemented in response to concerns from students about their personal information being shared [7]. As a result, most student information was removed from the application, leaving only names and photos, along with the most basic information: campus mailing and e-mail addresses. This update significantly reduced the value of Tigerbook V1 as a social resource for students, and many students were left feeling disappointed and frustrated. In response to this challenge, Tigerbook V2 was developed

with a focus on privacy and user control, allowing students to share information on an opt-in basis and providing powerful search tools for finding others who share similar backgrounds and interests.

## **2.2. My PrincetonU**

The university has a few alternatives to a student directory system, including My PrincetonU. My PrincetonU was introduced to the student body as a platform connecting a directory to events around campus. While it aimed to provide a similar service to Tigerbook, it had several pitfalls. The directory feature of My PrincetonU provided wrong filtering for queries and search results, even for attributes that were not attached to any user's profile. For instance, a search for the interest "Advertising" resulted in no profiles with that interest stated, instead displaying profiles with unrelated interests. Furthermore, in regards to its aesthetic, the cluttered user interface of My PrincetonU detracts from its main purpose as a student directory, and its functionality for social connection is limited because there are a myriad of fixed attributes to attach to one's profile such that people finding mutual interest in others would seem to be uncommon, e.g., using "CA" and "California" as filter queries when both are redundant. One observation to note is that there is a lot of users that are either staff or faculty on My PrincetonU which could lessen the exclusivity of using My PrincetonU. This may seem counter-intuitive; however, more exclusivity may induce more usage of the application when a student can more freely connect to their classmates; this is what Tigerbook V2 aims to solve. Finally, it appears that there is a lack of oversight in the creation of user-generated suggestions for My PrincetonU's search feature, which limits its usefulness as a student directory. For instance, one of the filter queries available on the platform is an emoji of an American flag, which does not provide any meaningful information about a user's interests, extracurriculars, or hometown. This implies a lack of consistency and standardization in the way filter queries are created and suggests that there is a need for a central body to govern the flow of information and ensure that the user-generated suggestions are accurate and relevant. Without proper oversight, the platform can become cluttered with irrelevant and inaccurate information, leading to a frustrating user experience and limiting the platform's effectiveness as a student directory. Tigerbook V2

addresses these issues by having a central team to be responsible for maintaining the platform and ensuring that the user-generated suggestions are accurate, standardized, and relevant. This ensures that the directory remains a useful tool for students looking to connect with one another over shared interests and backgrounds.



**Figure 1: "CA" and "California" are semantically the same, a redundancy in My PrincetonU**

### 2.3. Residential College Student Facebook

Another alternative is the Residential College Student Facebook (RCSF). Although it is not highly publicized, it is more accurate in regard to filtering compared to My PrincetonU. However, the end user is limited to three filters: class year, concentration, and residential college association. Its main advantages are its adherence to consistent naming of attaching a residential college, class year, concentration, track, e-mail address, and full name to everyone in the directory. This is in contrast to My PrincetonU where it is not mandatory to state one's concentration, whether planned or not, on their profile. The RCSF is still a, if not the most, robust solution for finding the most basic, up-to-date information on their student body. Anecdotally, there was one instance about a year ago in which the page after the next had shown an endless amount of duplicates and whether it was resolved, that seems to be the case. Tigerbook V2 is enhanced with information from the RCSF by

using most of the profile pictures of all undergraduate students as a default image for their profile on Tigerbook V2, along with their aforementioned pieces of information from the consistent naming.

Recently after the Tigerbook V1 API had been dismantled, the Princeton University organization ResInDe, a "student-run design consultancy" was faced with the problem, "How might we provide engaging information while balancing privacy concerns?" [6]. Their idea greatly influenced Tigerbook V2 by suggesting an opt-in system for sharing personal information. It stated the need to balance the privacy concerns of users with the need for valuable information to connect students. Additionally, they provided a clear vision of the prototype for Tigerbook V2, which helped guide the development process. Below is one of their user-interface mockups[6] for what an ideal Tigerbook that enables opt-in information would look like during the setup process.

**TIGERBOOK**

Welcome, John!  
Let's setup your profile

**Tigerbook has been redesigned!**

Please take a moment to setup your profile. All information is optional besides the information on the card above. Fill in the info you'd like to appear!  
Thank you for helping update Tigerbook!

**BIOGRAPHICAL INFORMATION**

Hometown

Country

Nickname

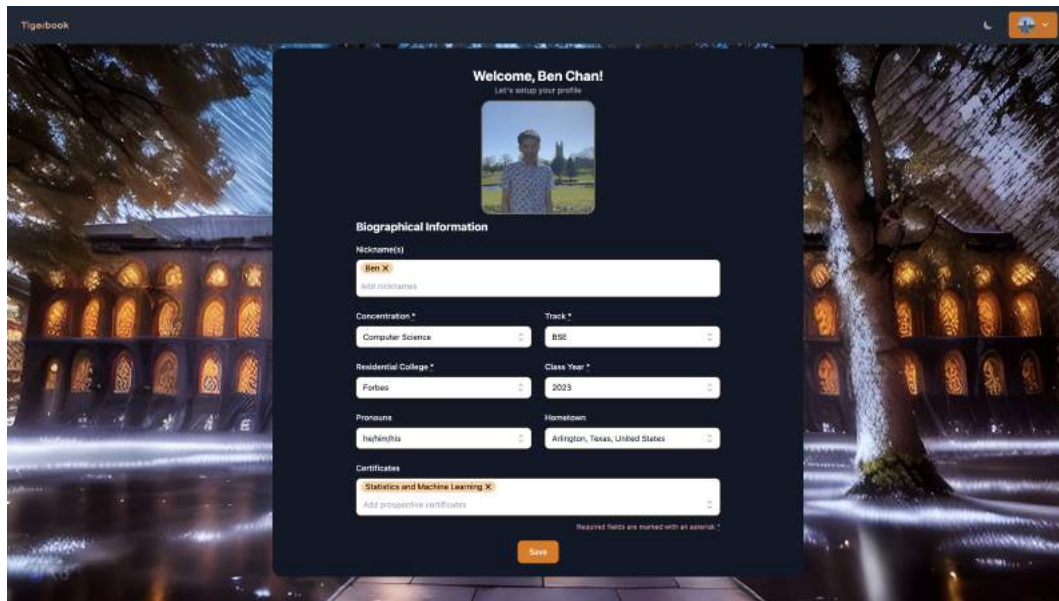
Pronouns

Certificates

Next

**Figure 2: ResInDe's setup page mockup of their perceived ideal Tigerbook.**

In comparison, below is my setup page that borrows some aesthetic similarities but with drastic design differences.



**Figure 3: The first setup page for Tigerbook V2 with some form fields populated**

### 3. Approach

I started this journey back in November 2022. I saw this an opportunity to have a profound effect on the Princeton community. To execute the plan of building the ideal Tigerbook for privacy and interaction, I had to quickly learn additional technologies that had a steep learning curve to them. My adviser had suggested a decoupled backend and frontend. In the end, I decided to settle for this architecture (that is, a Next.js frontend and Django REST API backend) because it seemed to me as future-proof and would easily extend to any other Princeton student wanting to build within or from the student directory space with JSON web tokens being the crux of the appeal for their future usage.

#### 3.1. Web-Scraping (Google Puppeteer on Node.js)

To lower the barrier for the end user to become accustomed to Tigerbook V2 quickly, I decided to scrape the Residential College Student Facebook for the existing profile pictures of Princeton undergraduate students. That is, if the end user did not want to upload an image of their own besides

the one on the RCSF, they could use this default profile picture for Tigerbook V2. For this, with Node.js, I used Google Puppeteer with a headless Chrome browser that can automate actions in web browsing. To enable this, at the start of the program I had created for Puppeteer, I had to provide my session cookie key-value pairs that I had since I was logged into RCSF. Then, it was able to log in from that point in a browser not under my manual control. It was able to browse a few hundred pages and save all the images on the page before each click to the next page. After web-scraping the images onto my local computer, I had the challenge of figuring out how to save it on the cloud. For that, after saving their image URLs on my database, I used Amazon S3 through the backend to serve these images to the client.

### **3.2. Backend (Django REST Framework)**

The backend should be built in a way such that it can handle both session authentication and token authentication, so I decided to use the Django REST Framework because it has robust libraries that can handle these production-ready situations. Session authentication would be for those users browsing on the site, and token authentication would be for those users retrieving Tigerbook V2 data outside of its web application. With the constraints of Princeton CAS authentication, I found out that to retrieve JSON web tokens for outside use, the user has to log in with CAS Authentication first, and the only way to do that with the way Duo Security MFA is set up for Princeton students is through a browser, with session authentication first, before using token authentication. That is, a Princeton undergraduate student cannot retrieve JSON web tokens merely by their terminal using *cURL* or any similar means; they would have to log in first through a browser. Of course, we could require the student sign up for Tigerbook V2 with different credentials, but they would have to log in with credentials other than their CAS login, thereby a hindrance. As an aside, I not only planned to have undergraduates to have their profiles on Tigerbook V2, but also I wanted to potentially let in Princeton alumni and graduate students, and this was made possible by the Django Uniauth package[9], which can allow Princeton alumni with their *@alumni.princeton.edu* email to sign up for Tigerbook V2. For the graduate students, their user model in Django would differ

greatly from that of the undergraduates because they have fields that are different, such as their housing location, program names, degree names, etc. I took this in mind and found similarities that all groups shared. For instance, all are based in a city (for the most part) and all have at least one hometown. Taking account of these commonalities, I created some appropriate shared tables that generalizes with the student body for future use, although they are purely experimental and not production-ready. Overall, the Uniauth, created by Princeton University Master's graduate, Lance Goodridge, provides the scaffolding for Tigerbook V2 to be extensible, such that alumni, graduate students, and undergraduates could showcase their profiles in one place.

### **3.3. Frontend (Next.js Framework)**

For the frontend to be fast both in terms of setting up and running smoothly for the end user, I opted to use the Next.js framework. The advantages are that it can handle switch between either handling server-side rendering or client-side rendering all within one application. Server-side rendering is useful to see if the client is authenticated to access a specific part of Tigerbook V2; this ensures that all data is fetched or validated before a client is shown a page. Client-side rendering is useful to lower the time it takes to see the first render of a page after it is done loading since the server plays less of a part using computational resources and is more on the client to render the data on their own computer or device. I use server-side rendering for every web page on Tigerbook V2 to check whether the end user has been authenticated with Princeton CAS in combination with client-side rendering to fetch less valuable data later. Using this combination is a secure way to provide for the end user while not sacrificing the speed that it takes to load the web page at the first render. Also, I use React Hooks throughout my project, since Next.js is built on the React language, to simplify my logic of my page. Next.js is especially important for Tigerbook V2 because it is essentially a single-page app with app state spread across its pages, which provides a smooth developer experience. This app state is important, especially when saving the settings of whether the user has the dark mode theme enabled or saving and displaying the notification pop-up state when there is an error in posting a form submission. Lastly, all of the frontend were styled



either manually with Tailwind CSS[5] or with Flowbite React components[2]. The logos were all from Heroicons, an open-source project[3]. The background image of Nassau Hall in a seemingly magical environment was AI generated on a cloud computer with the original image taken by me. The interface design influenced by Tailwind CSS is essential in creating a friendly and welcoming atmosphere for users. Through a streamlined and user-friendly design, students can easily navigate the platform and find relevant information to connect with their peers.

## 4. Implementation


We start at the backend which is the foundation of the system and work our way into the frontend for each of the following sections. As of right now, the backend is hosted at <https://www.api.tiger-book.com/> on a **Render** server with debug mode enabled, such that one can see the API routes available to them. Of course, when the system is in full production, i.e., when Tigerbook V2 is publicized to the general student body, the API routes will be documented on GitHub and debug mode will be disabled. The backend relies on a PostgreSQL database to store user information and their relevant data to populate their profile pages. The frontend is hosted at <https://www.tiger-book.com/> on **Vercel**.

### 4.1. Logging in and Out

Whenever an end user logs in via Princeton CAS, they would have to enter through the [/accounts/login/](#) route and logout through the [/accounts/logout/](#) route which both are provided by the Uniauth Django app. To provide a custom user experience for the Princeton community, I entirely copied the Uniauth app into my `./backend` folder relative to my root directory on GitHub. Then, I modified the CAS login procedure within the Uniauth views to classify whether the user is within one of the five categories: faculty, undergraduate, graduate, staff, or service account. Each correspond to the following constants declared within my code in the same respective order: `PU_STATUS_FACULTY`, `PU_STATUS_UNDERGRADUATE`, `PU_STATUS_GRADUATE`, `PU_STATUS_STAFF`, and `PU_STATUS_SERVICE_ACCOUNT`. When a user logs in for the first time, they create a entry in the `auth_user` table that the Django framework provides. Only users

that are undergraduates can access the setup pages for their profile. No user within the other four categories can create a profile as of currently. Users can logout of Tigerbook V2 by clicking on the drop-down menu on the top-right and then clicking "Logout".

## 4.2. Setting up Undergraduate Profiles

At the first time of logging in, undergraduates are greeted by the following page. They cannot access any other page of Tigerbook V2 until after they complete the two setup pages in total. The backend connects the email of the logged in user to that of the email found under their Residential College Student Facebook profile, if they have one. If they have a profile picture on RCSF, then it will be shown on this first setup page; if they don't have a profile picture on RCSF, they will be shown this placeholder . If the user does not have a profile on RCSF, then the pre-populated fields of concentration, track, residential college, and class year would not show up on their frontend, so they would have to add these to the form themselves.

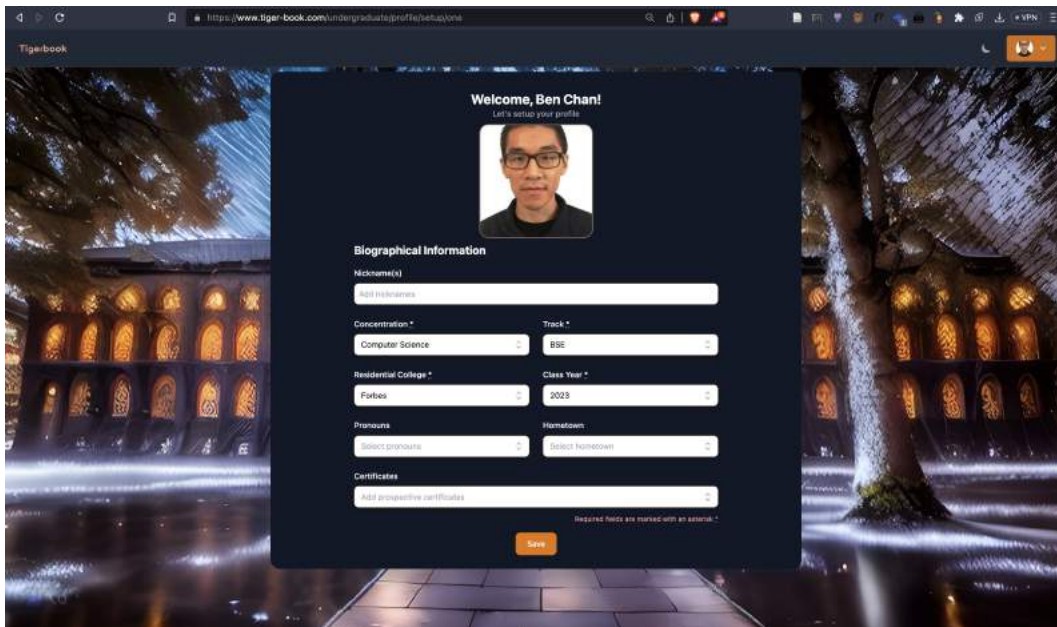


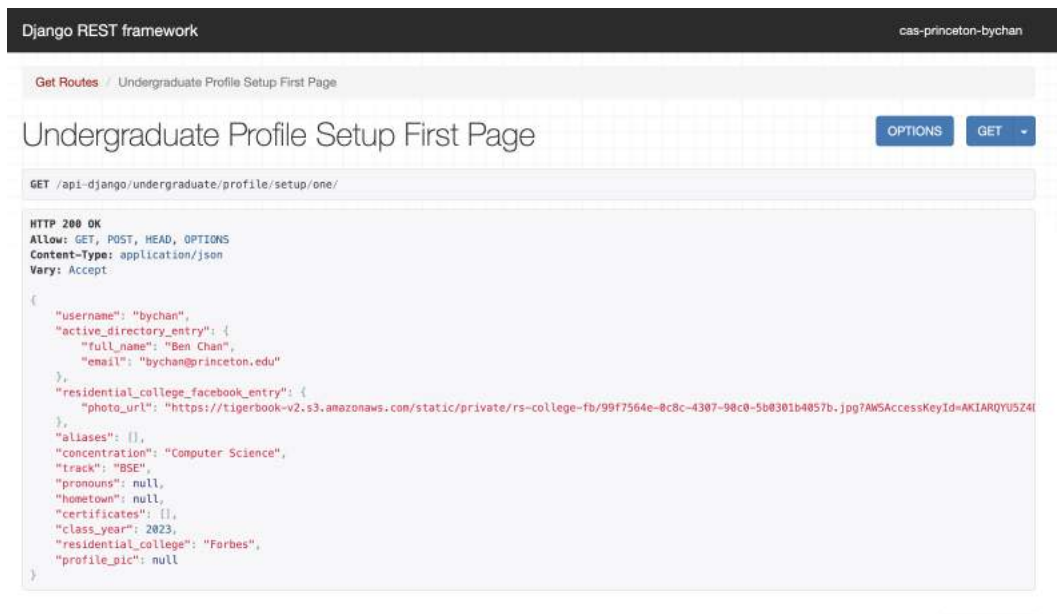
Figure 4: The first setup page for Tigerbook V2

All the form fields, except for "Aliases" have preset options to select from. These preset options are gathered for **concentrations**, **tracks**, **residential colleges**, **class years**, **pronouns**, **hometowns**, and **certificates**. All of these direct to <https://api.tiger-book.com/> of which contains the

`/api-django/` segment appended to it. These preset options are designed to enforce consistency so that students can easily filter within a group and expect exhaustible results. A student can put any name for aliases. In the backend, any free-form field like that of "aliases" will be validated for hyperlinks and HTML tags so that students cannot enact their malicious intentions of changing or leaking other end users' personal information on their browser.

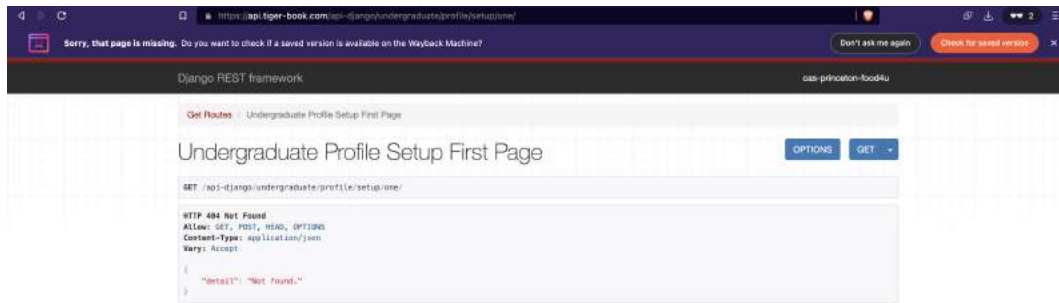
The first setup page populates its frontend through the API route

`/api-django/undergraduate/profile/setup/one/` as shown below.



**Figure 5: The main API route view for the first setup page**

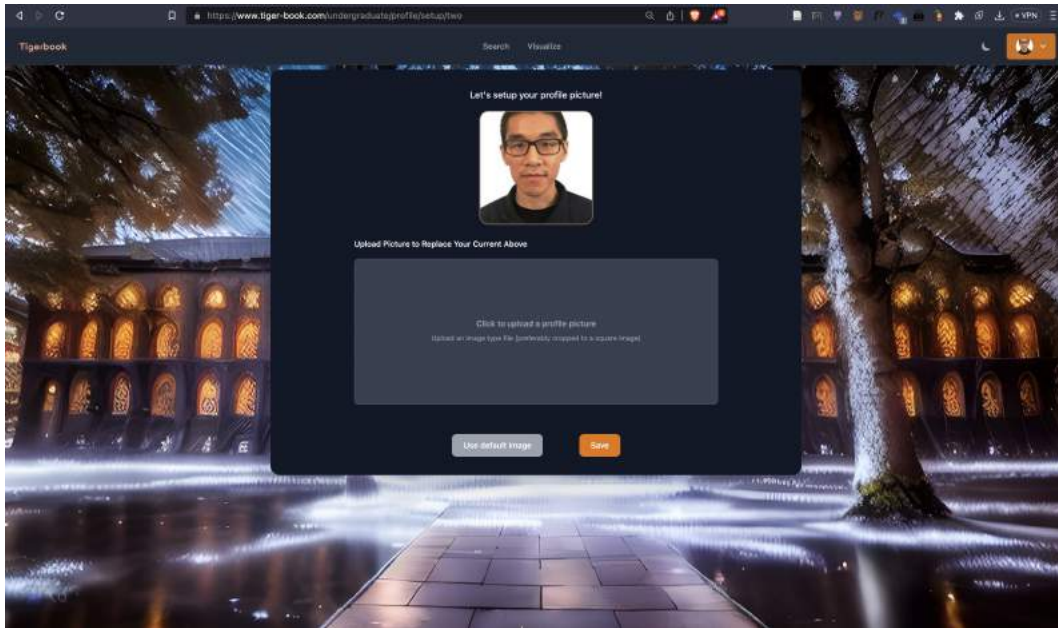
You will not see the such following if you are not a Princeton undergraduate student. If you are either any of the four groups other than an undergraduate student. You will see the following in Figure 6.



**Figure 6: The 404 error page for a route restricted to Princeton undergraduates**

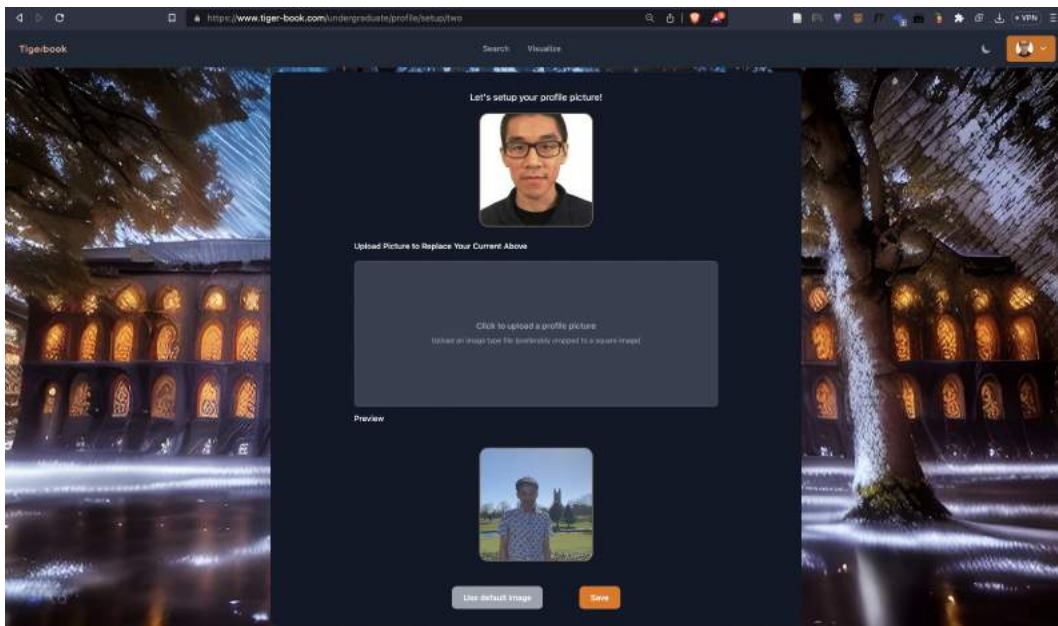
As one can see, the *username* field is retrieved from their CAS credentials via the Uniauth implementation. The *active\_directory\_entry* field information retrieved from Princeton OIT's Active Directory API [1], which one can register for. The *residential\_college\_facebook\_entry* field is populated by the data from the web-scraper that scrapes through the Residential College Student Facebook, found in the *.web-scraper directory* relative to the root directory. Briefly, the way images are uploaded from the web-scraping is through the Django management script called *add\_current\_residential\_college\_facebook.py*. This goes through the images and the details of each profile found by the web-scraper and adds it to Django's associated PostgreSQL database; the images are sent to Amazon's S3 cloud service to host images through a CDN, which optimizes the performance for image delivery. Additionally, the fields *concentration*, *track*, *class\_year*, and *residential\_college* were from the data gathered from the web-scraper. All the other fields are either null and empty, and for every login from thereon, they are able to access the first setup page again and are able to see these all other fields populated if they have filled them out previously. An aside, the *aliases* field has the array of nicknames that the frontend form field of "Nickname(s)" uses.

After saving information on the first login page, the user will then see the second page of the setup process in Figure 7.



**Figure 7: The second setup page for Tigerbook V2**

At this point, the user can either use the default image or use an image they have at hand. Here, I upload an image of my own liking. The form will preview with the selected picture dynamically.



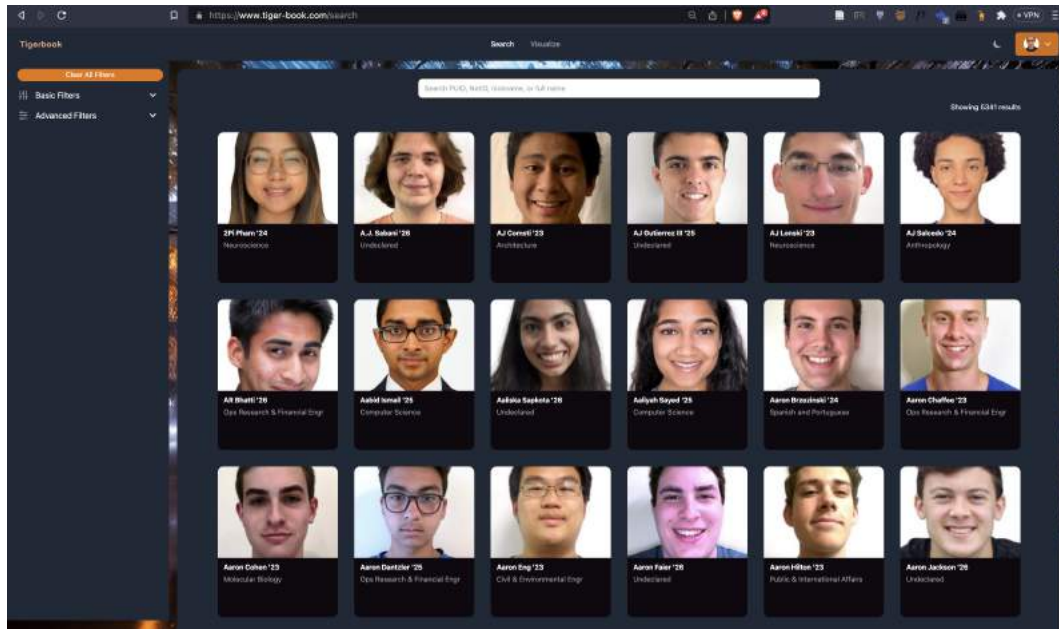
**Figure 8: The second setup page updated for Tigerbook V2**

From then, I can either save or use the default image. I choose to save.



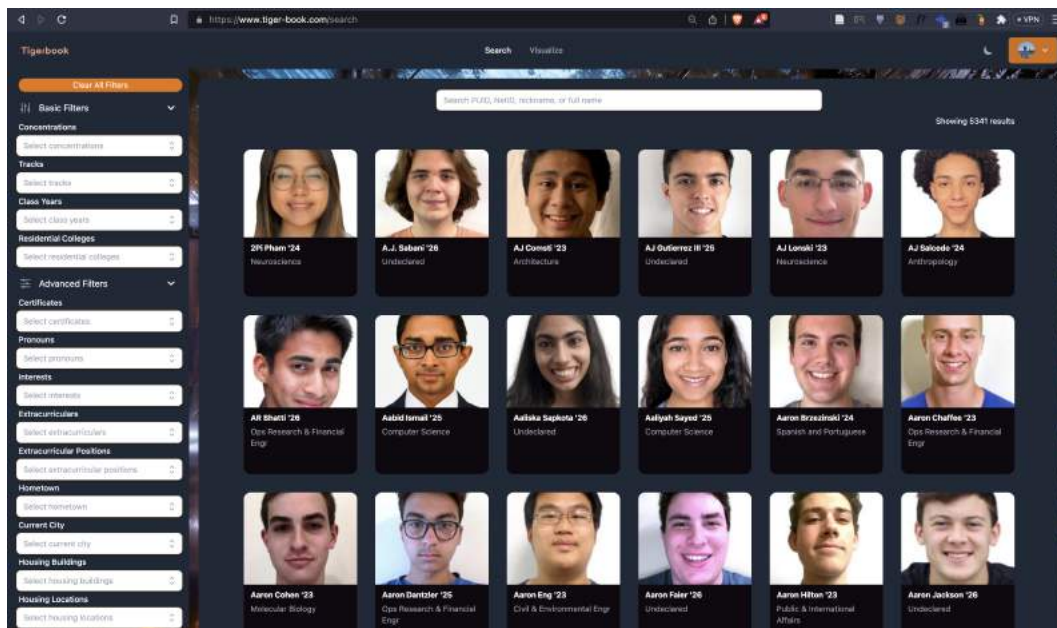
### 4.3. Using the Search Page

After setting up their profile pages, the undergraduate can expect to see the main page or the search page. This is also the first page that authenticated users will see if they are not an undergraduate.



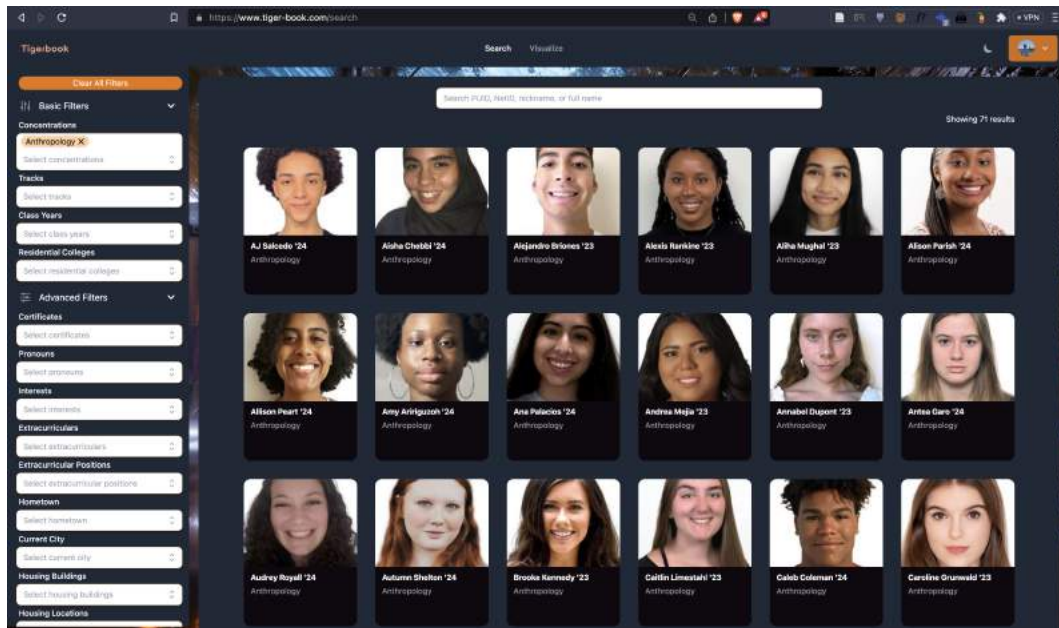
**Figure 9: The search or main page for Tigerbook V2**

All of the students listed will be in alphabetical order by their first name. The user can scroll down more to load more student directory profiles, notably called "infinite scrolling". This list may take a while to load. This is because it is also populating the filters on the far left which will constrain the backend server more. They are expanded in Figure 10.



**Figure 10: The search page with filters expanded of Tigerbook V2**

The search filters populate from the same API routes I described previously in addition to **interests**, **extracurriculars**, **extracurricular positions**, **housing buildings**, **housing locations**, and **research types**. On my computer, it takes about ten seconds for all the search filters to populate; this is the least I could do regardless of the computation power of the backend since I'm running these fetch requests in parallel. The "current city" search filter uses the same API **route** as hometown, as seen during the first setup profile page. Anytime I click to add a filter on the left, my search result of students update for that filter, as seen in Figure 11. To clear these search filters, the user can press the "Clear All Filters" button. This will return to the search results as before, unfiltered. The top four filters on Tigerbook V2 are considered "basic" filters because they are the primary attributes required for a student to create a profile on the platform. The remaining filters, known as "advanced" filters, are optional and require the user to input additional information to be searchable by these attributes. These advanced filters allow students to showcase their interests, extracurricular activities, research, and other unique attributes that make them stand out.



**Figure 11: The search page with filters activated on Tigerbook V2**

The search filters on the left all have preset options. On the other hand, the search bar above the student images can be freely typed into. This search bar is dynamic both in the sense that it can look up the four pieces of information simultaneously for the end user and can display search results as the user is typing. The four pieces of information are the Princeton PUID, Princeton NetID, their alias or nickname, and their full name. The algorithm can also find substrings of any of these four. One could use the search bar in combination with the search filters to narrow down their results. Whenever a search result is clicked it will bring the user to their profile page.



#### 4.4. The Basic Student Profile Page

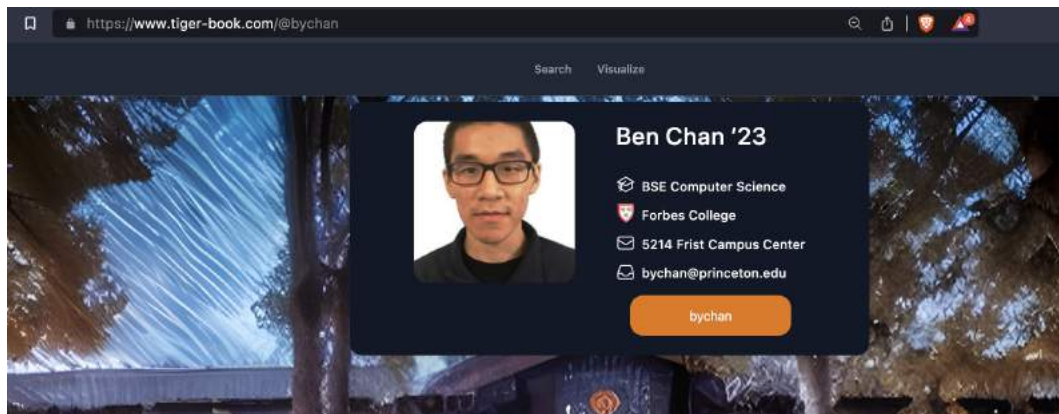


Figure 12: A basic student profile page on Tigerbook V2

In Figure 12, as one can see this is the most barebone profile one could find, assuming they have an RCSF profile and have their campus mailing address and email address in the OIT Active Directory (unfortunately, there are some students that do not have any of these). Alternatively, the API route to access this information is `/api-django/<str:username>` where the username parameter would be "bychan" as in Figure 12. Additionally, one can see that the URL contains the username followed by the @ symbol. This was intentional in letting profile viewers navigate to the profile they had in mind quickly or to share with others the link to this profile easily. Tigerbook V1 had campus mailing addresses before, so this was their idea I wanted to implement onto this platform. Of course, one could find a Princeton-affiliated student's campus mailing address with Princeton's public search functionality [4], but not many people know about this. Thus, it is more straightforward to include it in one's Tigerbook V2 profile, assuming this will gain traction among the student body. Their email address is also conveniently placed here. In addition to the four pieces of information that are required during setup, the username or NetID for undergraduates within the orange box is going to play a key role in Tigerbook V2's future extensibility, where alumni can have profiles of their own as well with their username ending in `@alumni.princeton.edu`.

#### 4.5. Editing the Student Profile Page

To edit their student profile page, the user will have to click on the drop down menu in the top-right corner as shown and click on "Edit Profile" in Figure 13. Again, this is available only to Princeton undergraduate students, as well as the page for the "Preview Full Profile" button, which will be discussed later.

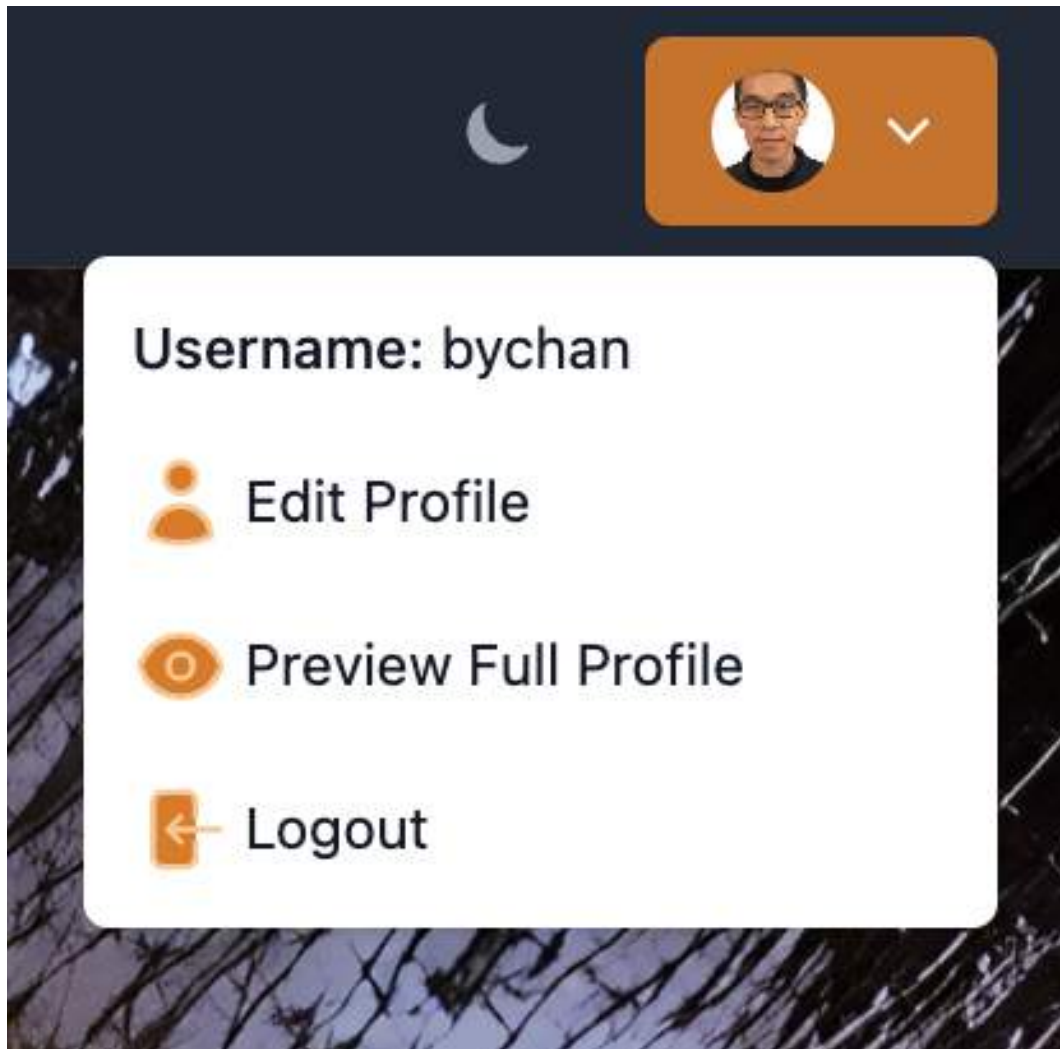
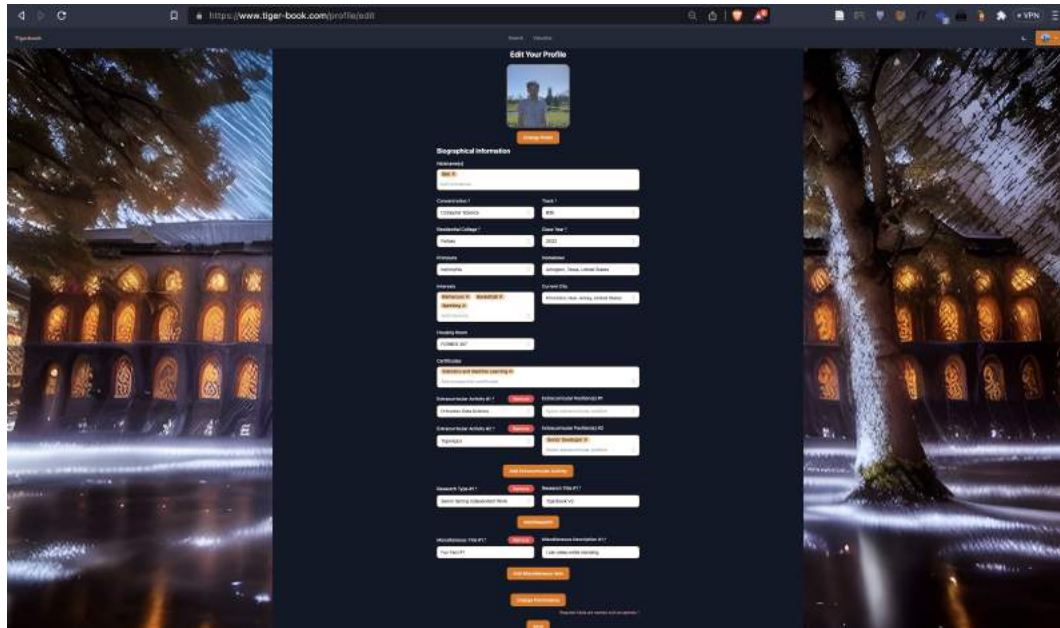


Figure 13: Locating the drop-down menu

After clicking "Edit Profile", the user will see this page shown in Figure 14, where some of the fields are already filled out.



**Figure 14: The profile edit page in full view**

There are several options available here. In addition to the basic profile information one can edit as seen in the previous setup pages. The user can also add their current city, which can be useful for get-togethers during the summer or any other break. The interests form field is also available to those who plan being search by those who have shared interests on the search page. Likewise with similar intention, the other new form fields are housing room, extracurricular activity and positions, research types and titles, and miscellaneous information. This is further zoomed-in in Figure 15.

The form is divided into several sections, each with a title and a list of input fields or buttons:

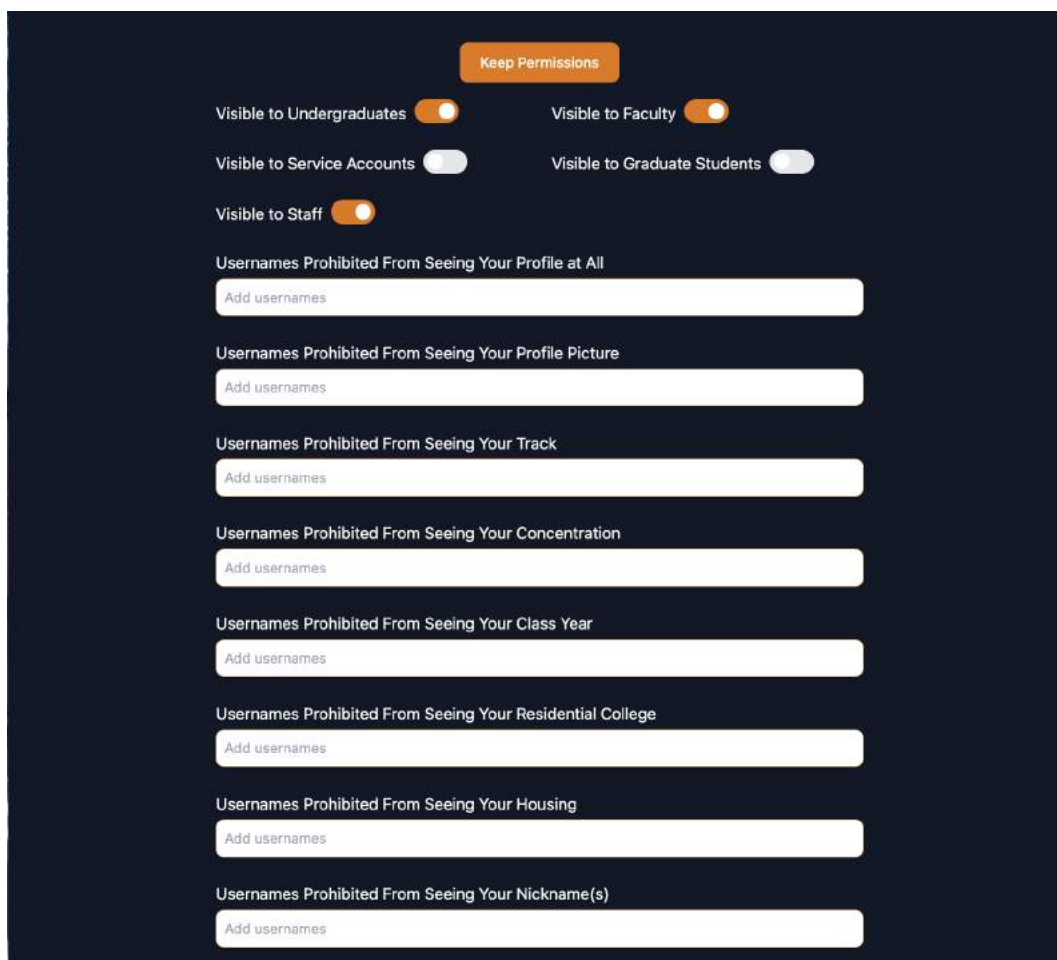
- Interests:** A container with three tags: "Barbecues X", "Basketball X", and "Sprinting X". Below the tags is a text input field labeled "Add interests" and a small up/down arrow icon.
- Current City:** A dropdown menu showing "Princeton, New Jersey, United States" with an up/down arrow icon.
- Housing Room:** A dropdown menu showing "FORBES 307" with an up/down arrow icon.
- Certificates:** A container with one tag: "Statistics and Machine Learning X". Below the tag is a text input field labeled "Add prospective certificates" and a small up/down arrow icon.
- Extracurricular Activity #1 \*:** A dropdown menu showing "Princeton Data Science" with an up/down arrow icon. To its right is a red "Remove" button.
- Extracurricular Position(s) #1:** A dropdown menu showing "Select extracurricular position" with an up/down arrow icon.
- Extracurricular Activity #2 \*:** A dropdown menu showing "TigerApps" with an up/down arrow icon. To its right is a red "Remove" button.
- Extracurricular Position(s) #2:** A dropdown menu showing "Senior Developer X" with an up/down arrow icon. Below it is a text input field labeled "Select extracurricular position".
- Add Extracurricular Activity:** An orange button.
- Research Type #1 \*:** A dropdown menu showing "Senior Spring Independent Work" with an up/down arrow icon. To its right is a red "Remove" button.
- Research Title #1 \*:** A text input field showing "Tigerbook V2".
- Add Research:** An orange button.
- Miscellaneous Title #1 \*:** A text input field showing "Fun Fact #1". To its right is a red "Remove" button.
- Miscellaneous Description #1 \*:** A text input field showing "I can sleep while standing".
- Add Miscellaneous Item:** An orange button.

**Figure 15: The new settings apart from the basic settings during the setup process**

The user can add additional fields of extracurriculars, research involvement, and miscellaneous to their profile by clicking on their respective "Add" button. Miscellaneous information, like the

aliases form field and the research title field, also takes in free-form input that is validated. Here, the user can add unique information onto their profile.

To increase the privacy of Tigerbook V2 to users that have such profiles, Tigerbook V2 introduces privacy settings shown in Figures 16 and 17. By default, mimicking the settings of RCSF, the profile is viewable only to undergraduates, faculty, and staff, and disabled for service accounts and graduate students. Furthermore, the user can set which usernames or Princeton NetIDs of those to restrict some pieces of information from them. Again, these settings are only for undergraduate students, as all other no one in any other group has such profiles. The second part of these permissions is shown. As one can see, any one of their attributes is controlled by adding more usernames of those that the user wants to prevent from seeing key parts of their profile.



The screenshot displays a dark-themed settings panel for a user's profile. At the top, there is an orange button labeled "Keep Permissions". Below this, there are six toggle switches arranged in two columns. The first column contains "Visible to Undergraduates" (on), "Visible to Service Accounts" (off), and "Visible to Staff" (on). The second column contains "Visible to Faculty" (on), "Visible to Graduate Students" (off), and "Visible to Staff" (on). Below the toggles, there are eight sections, each with a title and a text input field. The titles are: "Usernames Prohibited From Seeing Your Profile at All", "Usernames Prohibited From Seeing Your Profile Picture", "Usernames Prohibited From Seeing Your Track", "Usernames Prohibited From Seeing Your Concentration", "Usernames Prohibited From Seeing Your Class Year", "Usernames Prohibited From Seeing Your Residential College", "Usernames Prohibited From Seeing Your Housing", and "Usernames Prohibited From Seeing Your Nickname(s)". Each input field contains the placeholder text "Add usernames".

**Figure 16: The first part of permission settings**

**Username** **Prohibited From Seeing Your Pronouns**

Add usernames

**Username** **Prohibited From Seeing Your Certificates**

Add usernames

**Username** **Prohibited From Seeing Your Hometown**

Add usernames

**Username** **Prohibited From Seeing Your Current City**

Add usernames

**Username** **Prohibited From Seeing Your Interests**

Add usernames

**Username** **Prohibited From Seeing Your Extracurriculars**

Add usernames

**Username** **Prohibited From Seeing Your Research**

Add usernames

**Username** **Prohibited From Seeing Your Miscellaneous**

Add usernames

Required fields are marked with an asterisk \*

Save

**Figure 17: The second part of permission settings**



## 4.6. The Full Profile View

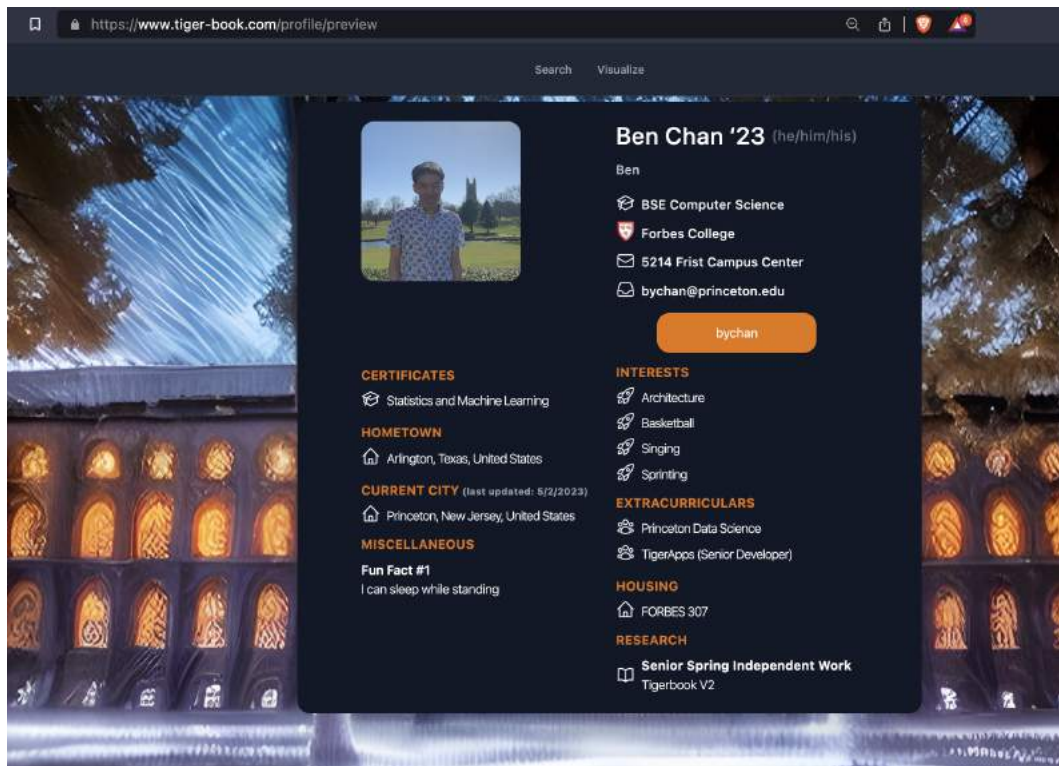


Figure 18: The desktop view of the full profile page

After submitting the form in the edit profile page, the user can expect to be redirected to the page shown in 18. This page can also be navigated to by the drop-down menu mentioned before. This page was also made to be responsive in the mobile view, as seen in Figure 19. Another feature to mention is the current city information in the profile that automatically shows when it has been updated. This is particularly useful when someone who is seeing this may want to question whether that user is still in the current city.

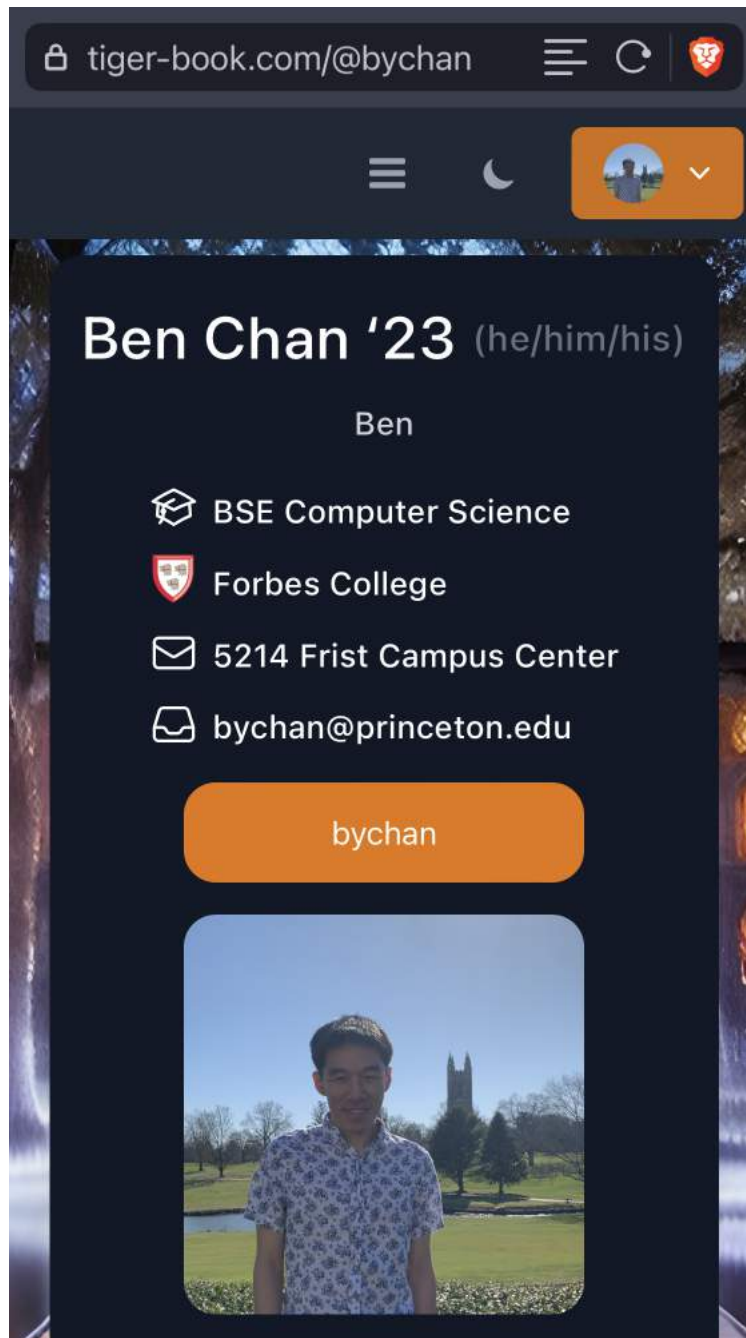
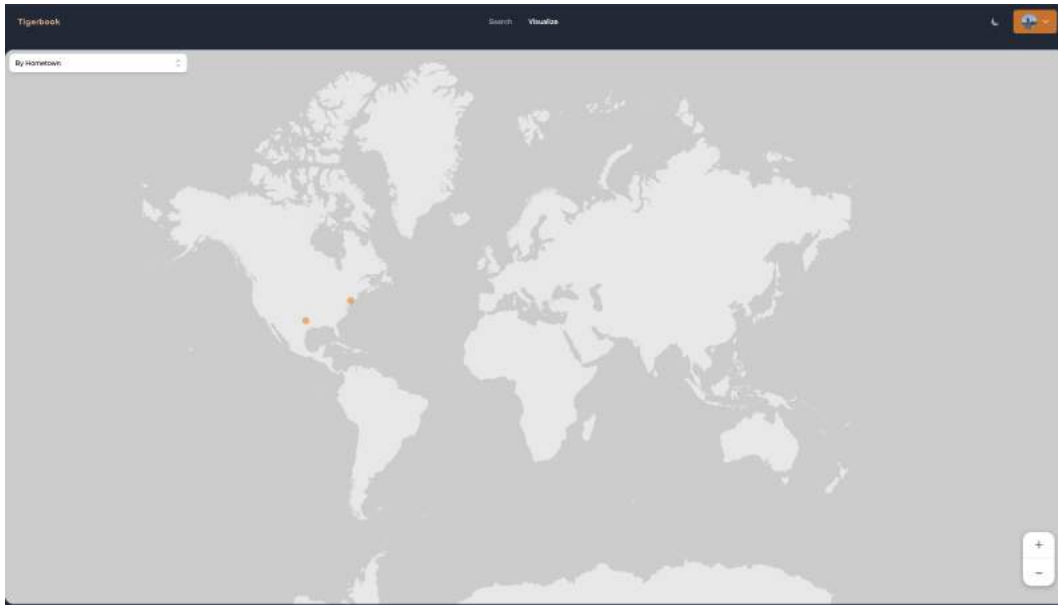


Figure 19: The mobile view of the full profile page

#### 4.7. Visualizing Current Cities and Hometowns

Navigating to the "Visualize" tab on the top bar will guide you to this web page shown in Figure 20.





**Figure 20: The full map view**

From there, one can see the hometown or current cities of people by clicking on the drop-down menu in the top-left. This map was implemented with the help of a library called D3.js [8]. The user can interact with the map by zooming in and out by clicking on the appropriate zoom buttons on the bottom-right or dragging the map with their cursor. The size of the circles are determined by how many of whom had listed the city that that circle indicates as their hometown or current city. Hovering over these circles gives the name of that city along with the frequency of the people that had indicated it.

## **5. Evaluation**

From my evaluation, this project exceeded the expectations that ResInDe initially laid out. If I had more time, I would add more permission settings for the student to manage their profile. If these permissions were to be exhaustible, then Tigerbook V2 would need to have switch buttons for prohibiting the five groups of users mentioned from seeing certain attributes too. The two notable attributes missing in the permission settings are the campus mailing address and email address. This will be resolved in the near future. Additionally, if I were to satisfy the SML certificate, I implement

a machine learning model to match users to each other based on the similarity percentage of their miscellaneous information.

There were many features that my adviser and I wanted to implement but could not within the given time-frame. Originally, my adviser wanted Tigerbook V2 to be implemented with a "notes" features such that users can save profiles of certain users and group them together with a title and description attached so that they could view them for future reference. This is already implemented in the backend with the four routes, */api-django/notes*, *note/create/<str:username>/*, *note/update/<str:id>/*, and *note/delete/<str:id>/*. The first route lists the notes that the current user has taken with each note having a group of student profiles, each displaying the basic information seen in the main page. The second route creates the note starting with the first one in the note being user with associated with the parameterized username. The third routes updates the note with the parameterized "id" with more or less student profiles to save for reference and more or less users to whom the note is being shared with (and only the user or the users whom the note is shared with can add more and less student profiles). Finally, the fourth route deletes the note entirely (and only the user or the users whom the note is shared with can delete this note). All that is left is to implement these user interactions in the frontend.

In addition, there was a suggestion feature I had wanted so that users can seek permission to add to the list of options to add to their profile and for search filters. They can post a form to the API route */api-django/category-submission/create/* stating the category to suggest for and the suggestion for any of the following categories: tracks, concentrations, residential colleges, certificates, research types, interests, extracurriculars, extracurricular subgroups, extracurricular positions, housing, and cities. The extracurricular subgroups was not stated before; this delineates the sub-category that an extracurricular corresponds to, e.g. the extracurricular subgroup "Technology" for the extracurricular "Princeton Data Science". It will be added to the list of filters on the search page after this suggestion feature is implemented in the frontend. For the central body that is administering Tigerbook V2, by adding their NetIDs to the environmental variable *TIGERBOOK\_ADMIN\_NETIDS*, they will be able to see, delete, and approve through the API routes in the respective order of

*/api-django/category-submission/list/*, */api-django/category-submission/delete/<str:id>/*, and */api-django/category-submission/approve/<str:id>/*, where the "id" of the submission is parameterized. All of this could be implemented within an admin dashboard, whereby the users that have their submission approved or declined will be emailed automatically of the decisions by the admins. This suggestion algorithm will be significant for the Princeton community at large because the existing solution do not capture the involvement of students in minute detail, including what their positions are and the existing extracurricular are available unbeknownst to the Princeton student body at large. Unfortunately, like the notes feature, the suggestion feature is implemented only for the backend and not the frontend. Undoubtedly, this feature will be prioritized over the notes feature because it will be more practical in Tigerbook V2 becoming an encyclopedia for Princeton commonalities in the student body.

Finally, the */api-django/token* API route allows users to retrieve JSON tokens for external use. The additional API routes */api-django/refresh/* and */api-django/verify/* allow users to refresh and validate their tokens respectively. This would allow anyone in the Princeton community to build applications off of the data that Tigerbook V2 provides. For example, the housing information is available such that a Princeton student in COS333 can create from the data a room review application.

## **6. Conclusion and Future Work**

Overall, besides having a simple and informative user interface styled using Tailwind CSS, this project hit four of the six main features stated in the introduction. The two privacy objectives were satisfied, along with the search functionality and map-based feature. The two features remaining are the suggestion and reverse-search functionality. The suggestion feature, as described previously, is already implemented in the backend; however, it would be a beneficial to click on an extracurricular or interest on a student profile and be hyper-linked to the search page, using this information to query. Other future work is creating profiles for alumni and graduates would be of significant interest. It would take additional work by modeling database tables for their profile, since their

attributes will greatly differ in comparison to the undergraduates; although they may share the same interests and extra-curricular activities as the undergraduates.

## **7. Acknowledgements**

I would like to thank Professor Jérémie Lumbroso for his guidance on the framework of Tigerbook V2. Additionally I would like to thank the TigerApps team for their suggestions for the frontend and backend, most of these suggestions were brought forward on the user-interface and user-interaction of this project. In the end, I had an enjoyable experience developing Tigerbook V2.

## 8. Appendix

### 8.1. Code

All of the code for this project is available on GitHub but its visibility remains private due to the privacy concerns of students.

### 8.2. Honor Code

I pledge my honor that I have not violated the Honor Code.

Ben Chan

## References

- [1] “API Store Listing.” [Online]. Available: <https://api-store.princeton.edu/store/>
- [2] “Flowbite React Components.” [Online]. Available: <https://flowbite-react.com/>
- [3] “Heroicons.” [Online]. Available: <https://heroicons.com>
- [4] “Site Search.” [Online]. Available: <https://www.princeton.edu/search>
- [5] “Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.” [Online]. Available: <https://tailwindcss.com/>
- [6] “Tigerbook.” [Online]. Available: <https://www.princetonresinde.com/work/tigerbook/>
- [7] “The Daily Princetonian: September 13, 2019 by The Daily Princetonian - Issuu,” Sep. 2019. [Online]. Available: [https://issuu.com/dailyprincetonian/docs/sep\\_13\\_issuu](https://issuu.com/dailyprincetonian/docs/sep_13_issuu)
- [8] M. Bostock, “D3.js - Data-Driven Documents.” [Online]. Available: <https://d3js.org/>
- [9] L. Goodridge, “django-uniauth,” Apr. 2023, original-date: 2018-12-26T15:09:08Z. [Online]. Available: <https://github.com/lgoodridge/django-uniauth>