Spring 2020

LabLineup: An Intuitive Web Application for Queueing Help Requests in Academic Labs

Graham McDonald
University of South Carolina - Columbia, email@gmcdonald.net

LABLINEUP: AN INTUITIVE WEB APPLICATION FOR QUEUEING HELP REQUESTS IN ACADEMIC LABS

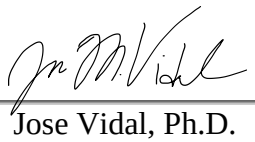
By

Graham McDonald

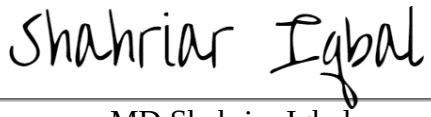
Submitted in Partial Fulfillment
of the Requirements for
Graduation with Honors from the
South Carolina Honors College

May 2020

Approved:



Jose Vidal, Ph.D.
Director of Thesis



MD Shahriar Iqbal
Second Reader

Steve Lynn, Dean
For the South Carolina Honors College

Table of Contents

Thesis Summary

LabLineup is a start-up project designed to address the needs of large, lab-based classes, where teaching assistants (TAs) struggle to assist students in the order in which they request help. LabLineup is a lightweight web application that allows TAs and professors to accept requests for help in order. LabLineup allows professors to view the requests for a lab and see frequently asked questions that can be addressed en masse rather than individually. LabLineup also allows students to provide TA feedback. LabLineup addresses issues arising from requesting help in large, lab-based classes in the most efficient manner possible.

Introduction

Problem

In many large labs, teaching assistants and professors struggle to assist students in the order in which they request help. In a lab with 40 to 50 students, the teaching assistants (TAs) could easily be searching through a sea of 40-100 hands raised in the air with no idea of who was next in line. This leads to some students having excessively long wait times. Sometimes, students may never get helped before the end of lab, despite the best efforts of the TAs. Additionally, as professors develop new lab assignments and introduce new technologies, multiple students frequently incur a single issue, and teaching assistants must address each student separately. This is extremely time consuming and inefficient. Also, course evaluations typically are not given for each teaching assistant individually. This causes all teaching assistants to receive the same feedback. Some excellent teaching assistants may be receiving negative feedback due to the sub-par work of another teaching assistant.

Solution

LabLineup is a lightweight web application that holds each student's request in a queue. Teaching assistants and professors simply click one button to accept the next request for help in the order in which the requests are received. LabLineup also allows professors to view the requests history for a lab and see frequently asked questions that can be addressed in class, over mass email, or in the future to proactively prevent issues from arising. LabLineup also allows students to provide feedback for each request. The feedback is given to the professor for the lab as a whole and for each teaching assistant individually. The professor can also choose to allow the teaching assistants to see their individual feedback. LabLineup aims to address issues arising from requesting help in large, lab-based classes in the most efficient manner possible.

Methods & Approach

Features

Features for Professors

LabLineup allows professors to create labs quickly and easily. Professors simply give their lab a name and description, and the lab is up and running. The professor can then generate a code to give to their students to join the lab. Teaching assistants can be added by a separate code or by

username if they already have an account. Professors can monitor the members of their labs and remove users as needed. They can also close the lab to prevent new users from joining.

Professors are able to accept the next request for help in a lab simply by clicking the “Accept Next Request” button. This process is significantly more efficient than the hand-raising method.

Professors are also able to receive notifications whenever a new request is submitted or when the number of requests reaches a specified level. This allows the professor to monitor the lab and to help the teaching assistant(s) if necessary. These notifications are sent via email.

Professors can view the students’ submitted feedback for the lab as a whole and for each teaching assistant individually. This feedback includes the average wait time and the number of requests closed. Additionally, the professor can choose whether the teaching assistants are able to see their feedback. Professors can also view every request submitted for a lab.

Features for Teaching Assistants

Teaching assistants, like professors are able to accept the next request for a lab simply by clicking the “Accept Next Request” button. This allows the TA to avoid searching through a sea of hands to determine which student was next in line for help.

Teaching assistants can receive notifications whenever a new request is submitted or when the number of requests reaches a specified level. These notifications are sent via email.

Teaching assistants are able to view their feedback and statistics, such as their average response time and the number of requests they’ve closed, if the professor allows it and provided they have received at least three ratings from students.

Features for Students

After joining a lab via the professor-provided lab code, students can submit a request simply by providing their station number and a brief description of the issue. Once the request is submitted, the student can see a summary of their request along with the estimated wait time and the number of requests ahead of them in the queue. If the student is never helped, the student can mark the request as such, and the professor will be notified via email. If the student solves their problem while waiting for the teaching assistant or professor, the student can cancel the request.

Students are also able to see their request history for each lab. Students are shown each request along with its submitted datetime, its completed datetime, who assisted them, the feedback they left, the station, and the problem description.

Technologies

Django (Python, HTML, JavaScript)

For our backend, we chose Django. Django is a Python framework designed to allow for rapid, scalable, and secure web development [1]. Django has been used to create many successful web apps, including Disqus, Instagram, Spotify, YouTube, BitBucket, DropBox, Eventbrite, Pinterest, NASA’s website, and many news media sites [2]. Django provides many features out-of-the-box, such as a built-in authentication system, cross-site request forgery prevention (CSRF tokens), and SQL injection prevention (form data cleaning) [1]. Django uses HTML (Hyper Text Markup

Language) templates to serve pages with dynamic content. JavaScript is also used in the templates for form submission and creating dynamic pages. Django also allows for the storage of user session cookies to save user-session specific data. Django stores persistent data as models (classes) that are held inside a database. We chose to use a MySQL instance running in Google Cloud Platform to host our data.

Google Cloud Platform

CloudSQL (MySQL)

We created an instance of CloudSQL in Google Cloud Platform to host our data. The relations were automatically created by Django migrations. The relations, while created and typically managed as Django models, can be administered using the standard MySQL commands. This flexibility allows us to create routines to clean-up expired data from the database. Google's CloudSQL handles the server management part of the MySQL instance, preventing any speed and capacity issues from arising.

Compute VMs

Our web server runs on a cloud-based virtual machine in Google Cloud Platform called Compute. This machine can be easily modified to support larger amount of virtual RAM and virtual CPU units. The machine runs Ubuntu (Linux). Due to the lightweight nature of LabLineup, we are able to use the web server built into Django.

Network Load Balancer

We created a network load balancer inside of Google Cloud Platform to route network traffic. The network load balancer has two ingress points, one for HTTP and one for HTTPS. The HTTPS frontend has an SSL certificate attached to the domain, assuring users that the site is trusted. Both frontends direct traffic to a single backend: a group of Compute VM instances that run the web service. The number of Compute VMs can be easily scaled to support growth of the usage of the web app.

Bootstrap (CSS)

For the styling (Cascading Style Sheets), we chose to use Bootstrap. Bootstrap is the most widely used front-end library for web applications [3]. Bootstrap contains pre-defined CSS styles for HTML and built-in JavaScript functions and allows developers to build responsive web applications [3]. Due to its powerful feature-set, widespread use, and its simple integration, Bootstrap was our top choice for our front-end styling. We also modified some components of Bootstrap to support our unique needs.

MailGun

LabLineup has the ability to send professors and teaching assistants email notifications, alerting them to new requests, a lab that has reached a specified number of open requests, or if a request has been transferred to them from another professor or teaching assistant. To do this, we chose to use MailGun, primarily because it allows us to create impressive templates, allows 10,000 emails to be sent for free each month, and is used by many well-known companies, including GitHub (owned by Microsoft), Lyft, and Reddit [4].

Square API

LabLineup is primarily a free service. However, in order to offset the operation costs, LabLineup offers premium plans that allow professors to have more labs for a modest subscription fee. We chose to use Square for our payment processing. Using the Square API, we created a "Checkout" that allows subscribers to pay for their subscriptions. The entire payment transaction takes part on Square's domain, ensuring that the transaction is secure [5]. After the transaction is complete, the user is taken to a page on our site that confirms the transaction and activates the user's subscription.

Selenium IDE

LabLineup has a multitude of behavioral tests developed using Selenium IDE. Selenium IDE allows developers to create behavioral tests for any website/web application by recording actions on the target page as if the developers were users [6]. Selenium IDE, based on Selenium, allows developers to create powerful cross-browser tests quickly [6]. See Testing & Validation for more details.

App Layout

The entire app uses a responsive design to allow users to easily use the web app on their phones, computers, or tablets seamlessly. Professors, teaching assistant, and students will have similar views but with different functions.

Professor / Teaching Assistant View

The main app view for all users is the Select Lab page. This page shows all the labs in which the user is a professor, teaching assistant, or student. Users can have multiple roles (professor, teaching assistant, and student), but they are restricted to only having one role per lab.

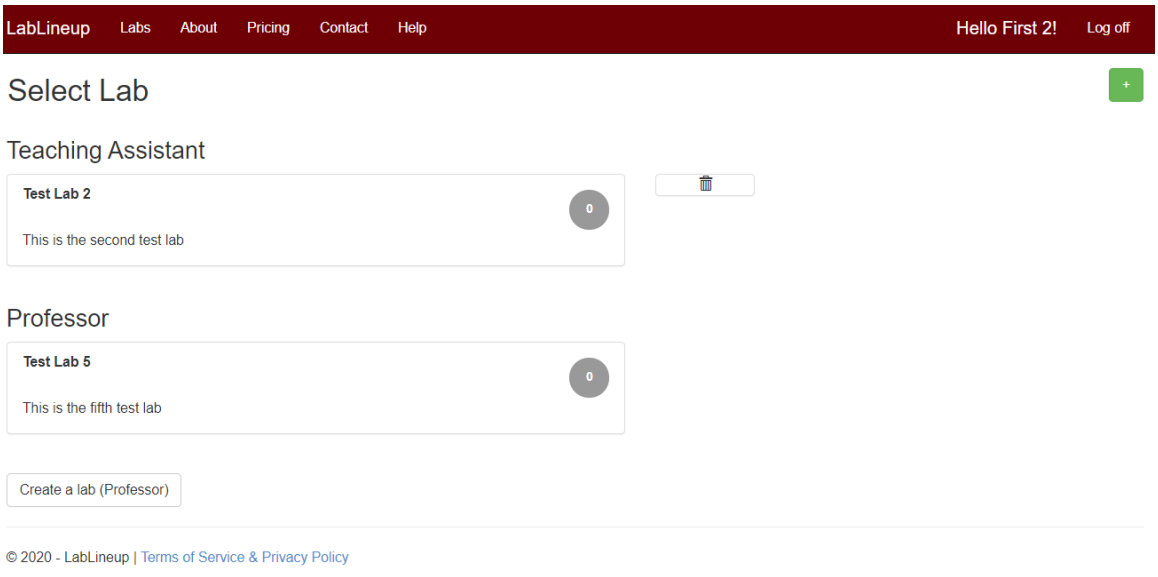


Figure 1. Select Lab (Professor/TA)

Professors can easily create a lab by clicking the "Create a lab (Professor)" shown in Figure 1.

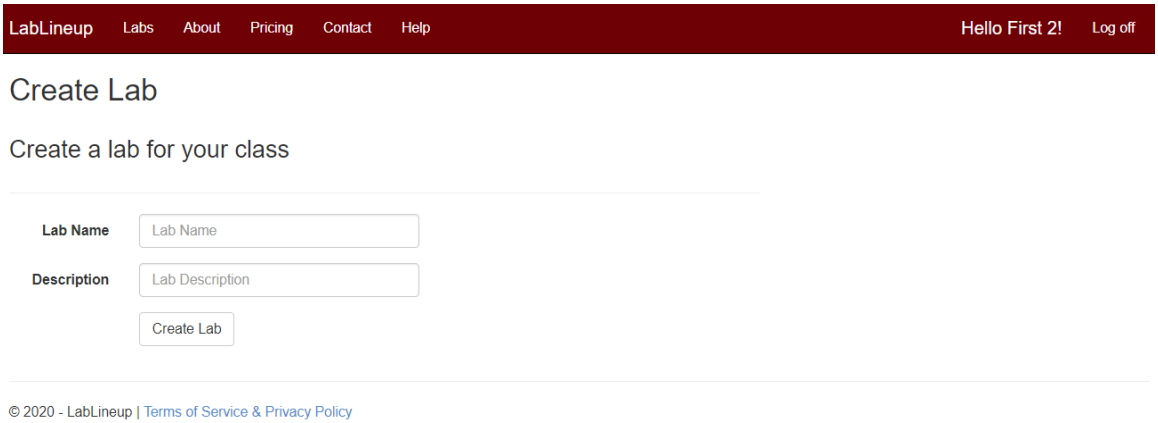


Figure 2. Create Lab

After creating a lab, professors can generate lab codes to be distributed to students and teaching assistants (two unique codes will be used for students and teaching assistants), request notifications for new requests and/or if there are a set number of open requests in the lab, add TAs by username, close the lab (preventing new users from joining), and remove users who do not belong in the lab.

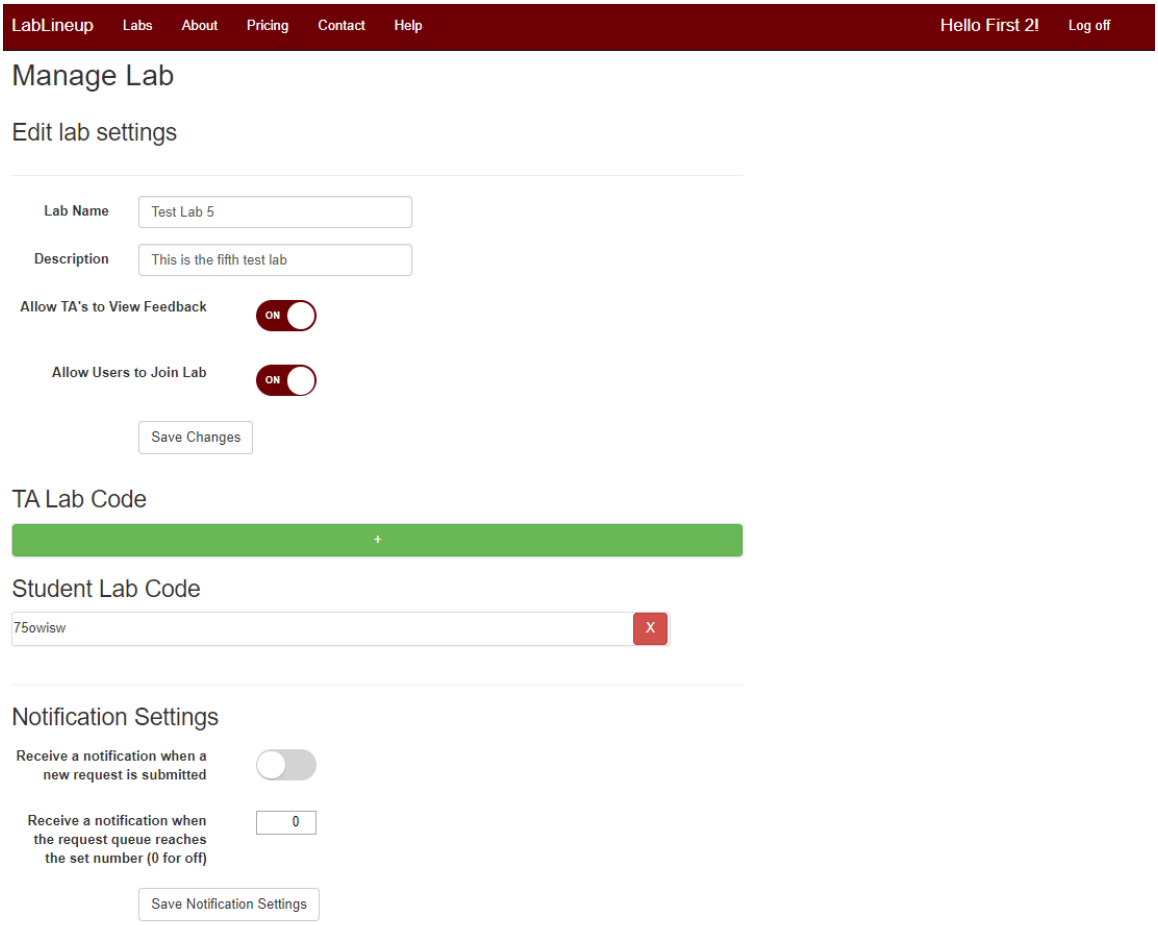


Figure 3. Manage Lab / Lab Settings

Once students are added to the lab and begin to submit requests for help, the professor and teaching assistants can easily monitor the lab and accept requests for help. To accept the next request, the professor or teaching assistant can simply click “Accept Next Request” as shown in Figure 4.

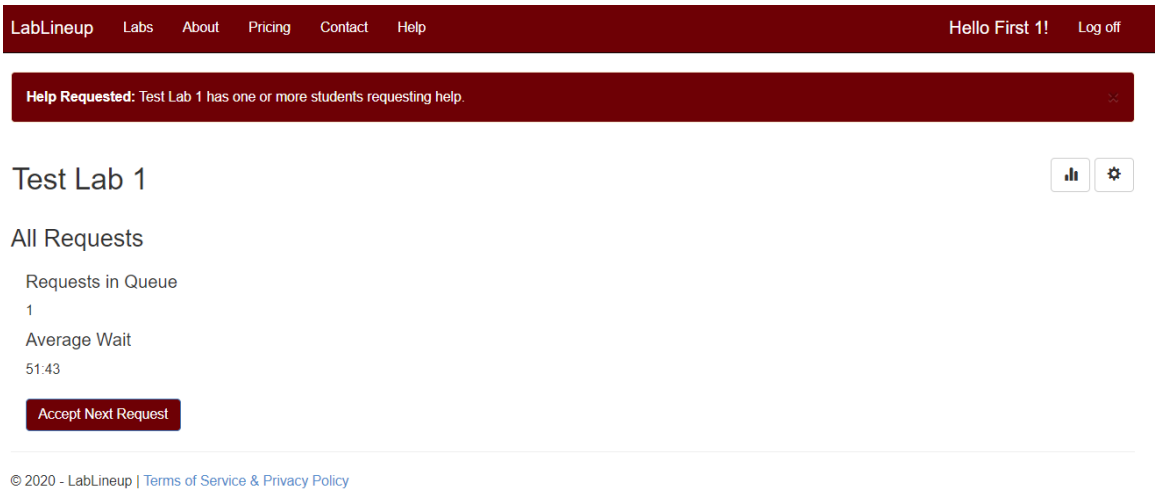


Figure 4. Lab Queue

Once the professor or teaching assistant has accepted the request, they can see the user, the user’s station number, the problem description, and when the request was submitted.

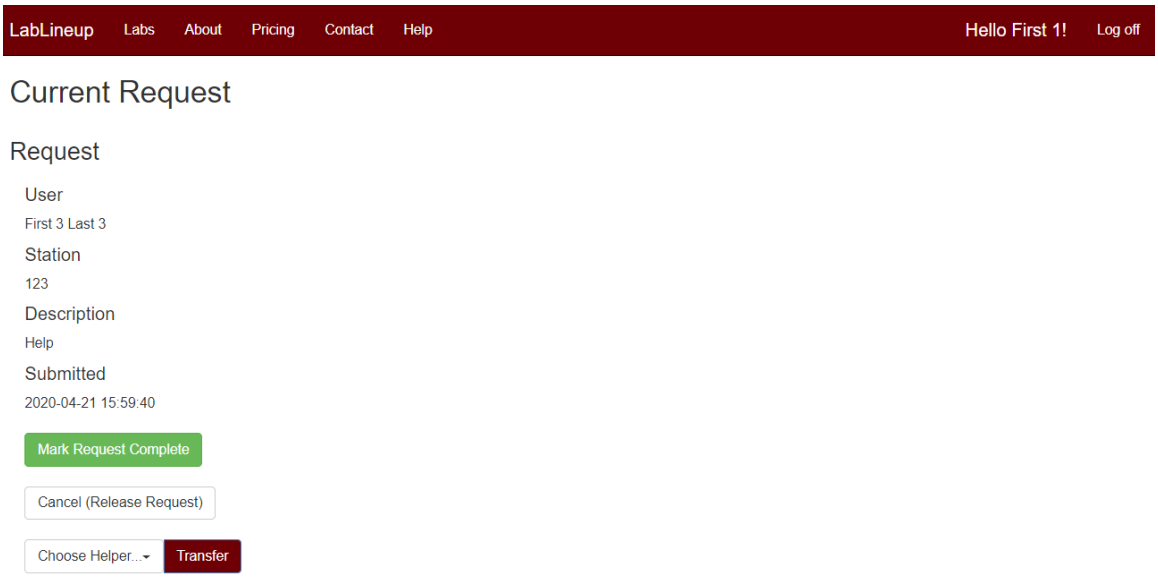


Figure 5. Current Request

Once they help the student, they can mark the request as complete. If they are unable to help the student, they can transfer the request to another professor or TA. If they accidently click accept request, they can release the request, sending it back to the front of the queue.

Professors can also view the average wait time, the average feedback rating left by students, the number of requests closed, and the number of open requests for each lab. This can also be filtered by teaching assistant. Moreover, the professor can choose whether teaching assistants can see their feedback. This setting is shown in Figure 3. Professors can also view all of the requests submitted in the lab. This allows the professor to determine if there are any common issues students are having.

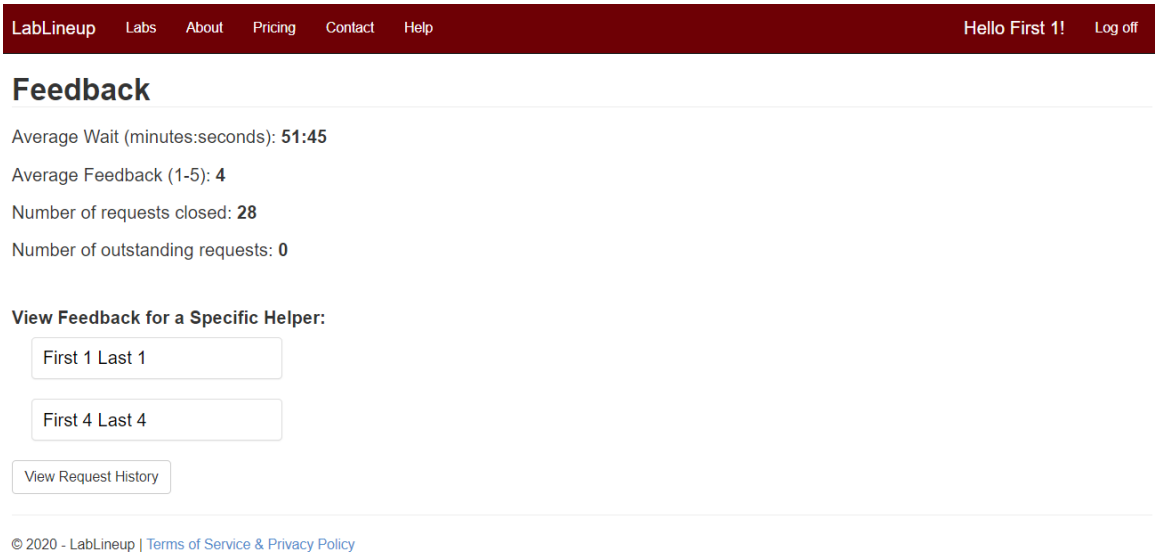


Figure 6. Lab Feedback Report

Student View

The main app view for all users, including students is the Select Lab page. From this page, students are able to select a lab in which they would like to submit a request for help.

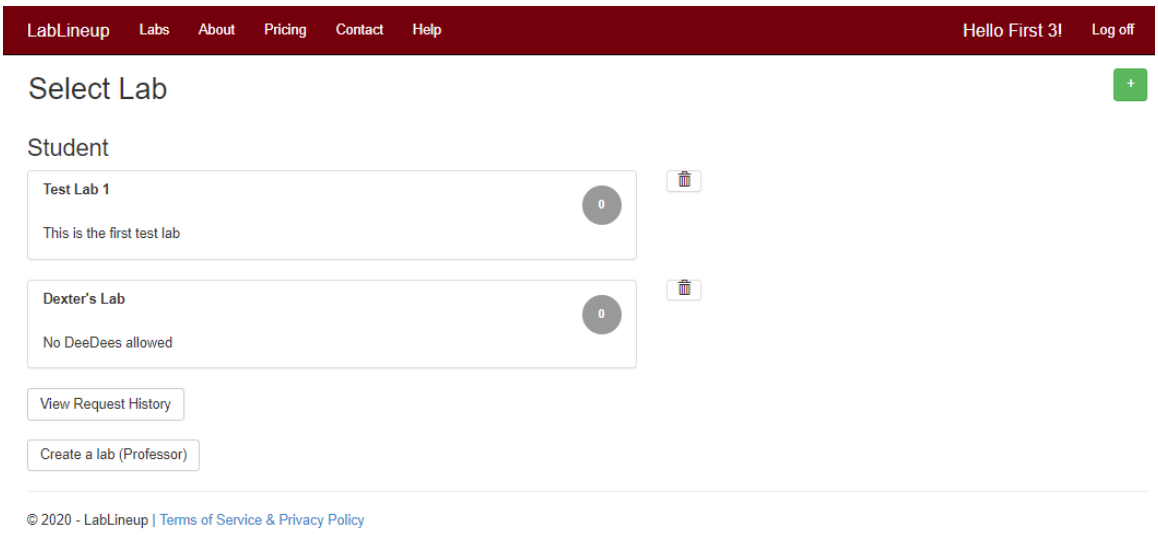


Figure 7. Student select lab view

First, students will add a lab to their account by clicking the green + icon in the top right corner of Figure 7.

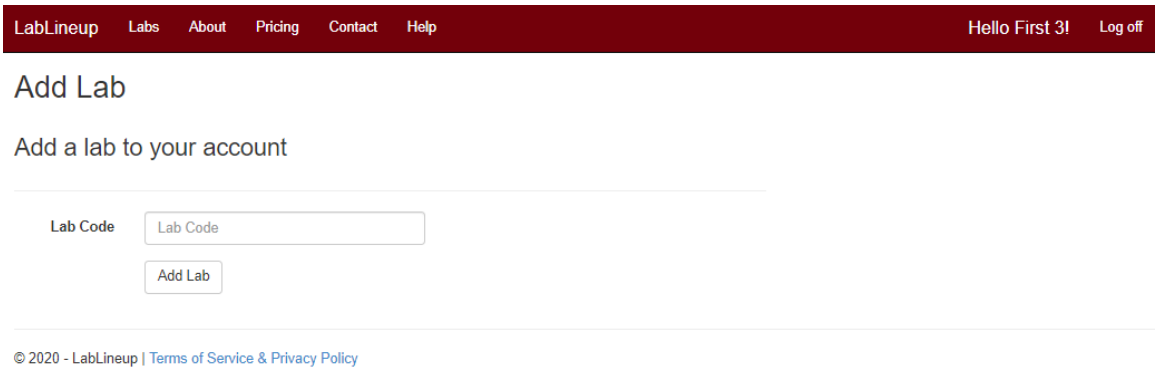


Figure 8. Add Lab to Account

Students will then enter the lab code provided by their professor in the field as shown in Figure 8. After successfully adding the lab to their account, students can select that lab from the Select Lab page (shown in Figure 7). Selecting a lab will take the students to a simple form that will allow them to submit a request for help (shown in Figure 9).

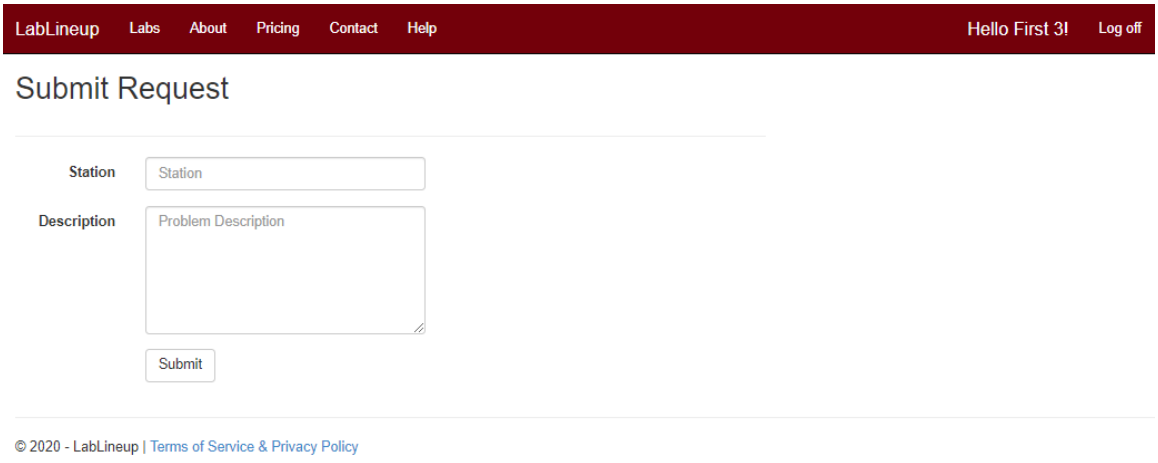


Figure 9. Student Submit Request for Help

After submitting a request for help, students will be taken to a waiting page (shown in Figure 10) that will show them their estimated wait time and the number of requests before them. Additionally, students can cancel their request or notify the professor that they were never helped.

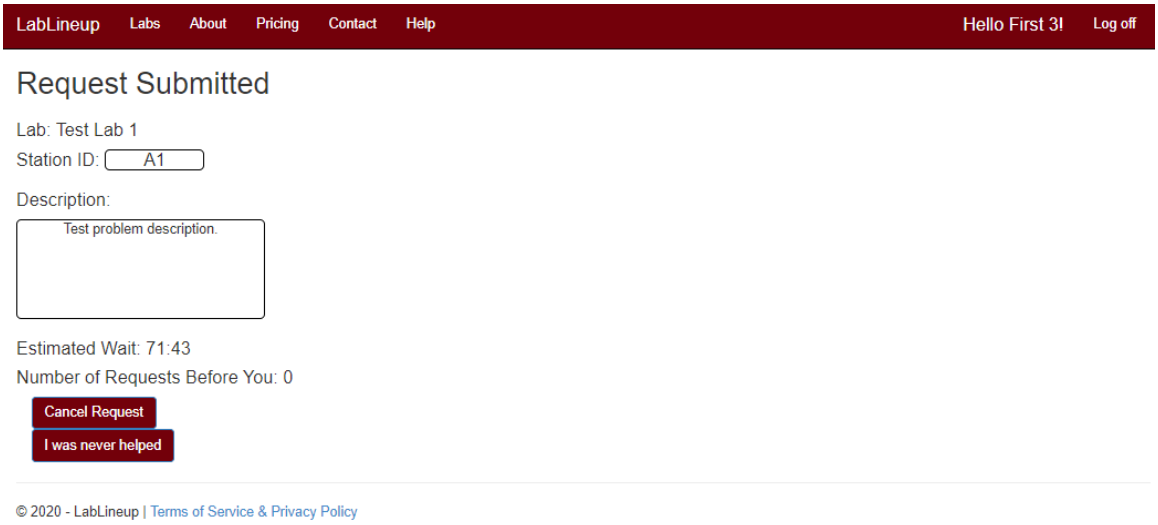


Figure 10. Student Request Submitted

After the student has been help, the student will be redirect to a page to allow them to submit feedback for the professor or teaching assistant who helped them by clicking a number from 1 to 5 as shown in Figure 11.

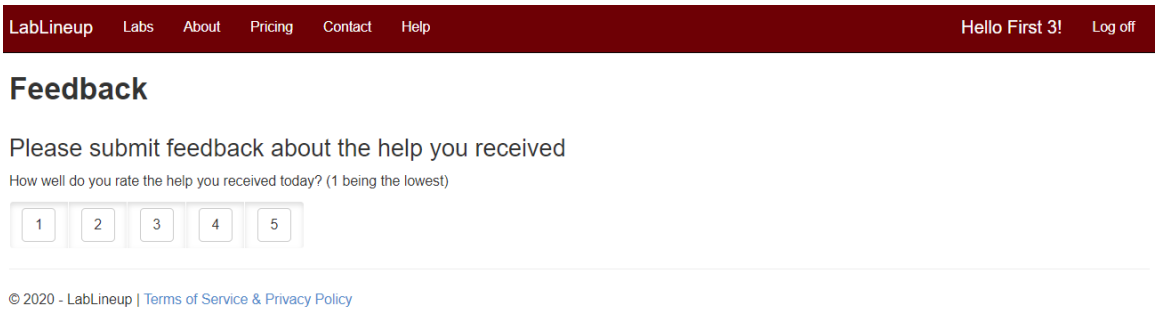


Figure 11. Student Submit Feedback

Students can also view their request history by clicking the “View Request History” button at the bottom of the Select Lab page as shown in Figure 7.

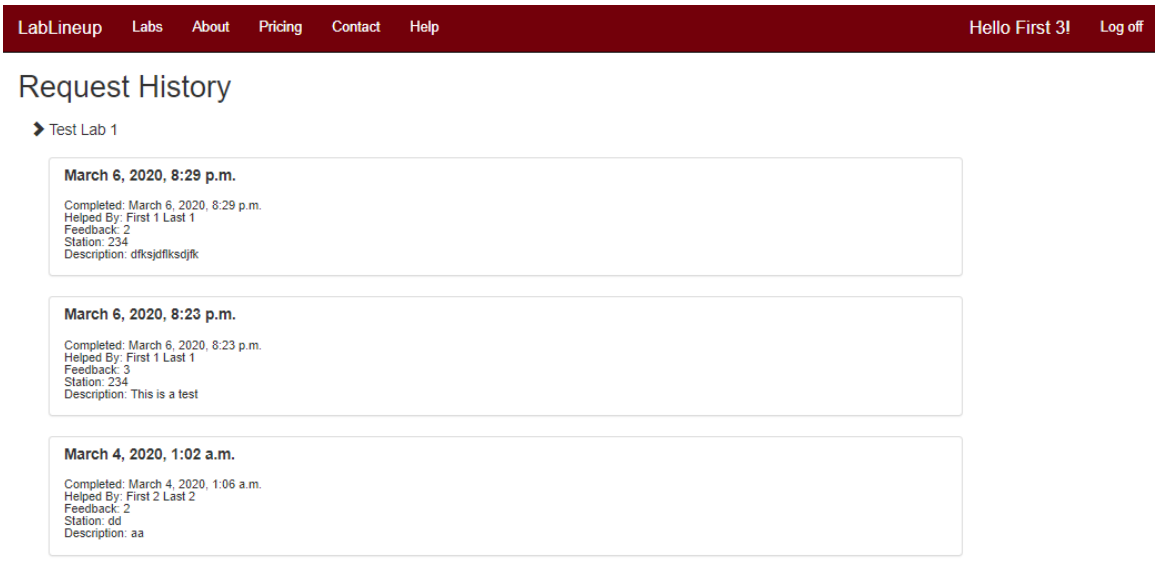


Figure 12. Student View Request History

Technical Obstacles

Obstacles & Solutions

User Account Roles / Permissions

Due to the need for each user to be assigned one of three roles for each individual lab, the standard built-in permissions system Django provides is impractical for managing access control for LabLineup. To use Django's standard access control system by itself, we would have to create three roles (professor, teaching assistant, student) for each lab in the authentication user permissions relation. Instead, we are using a table detailing each user's role in each lab of which they are a member (see the Role relation in Appendix: Database Schema). This relation has four fields: a unique id, the lab id, the user's id, and the user's role for the specified lab id. This allows us to manage lab permissions in an organized and efficient manner.

Notifications

Since LabLineup is a web application, we are unable to use push/toast notifications as most operating systems use today. Additionally, according to research by OneSignal, a customer engagement service provider used by companies such as Verizon, Volkswagen, Bose, and many others, users do not opt-in to approximately 90% of web push notifications (compared to 9.9% for Android and 56.1% for iOS) [7]. To provide professors and teaching assistants with necessary notifications, we send email notifications using MailGun.

Accepting Payments

While LabLineup is primarily a free application, we offer premium subscriptions for professors in order to cover operating costs. To accept payments for the subscriptions, we use Square's API. All credit card payments are required to be PCI (Payment Card Industry) compliant. Compliance has 90 requirements [8]. To avoid this issue, we use Square's Checkout API. This allows us to send the order information to Square. The user will complete the transaction on Square's site (Square maintains PCI compliance). The order will then be confirmed on our servers.

Testing & Validation

Unit Tests

To ensure LabLineup performs as expected, we have many unit tests using the unit testing framework built into Python and Django. These tests primarily focus on the modelFunc.py file, the file that acts as a layer between the models and the rest of the application.

Behavioral Testing

To confirm all parts of the application works, including the front-end, we run behavioral tests created using Selenium-IDE. The tests, along with the unit tests, are run immediately prior to each release to check for any bugs that may have been introduced since the previous release along with any bugs from new features.

Beta Version Deployment

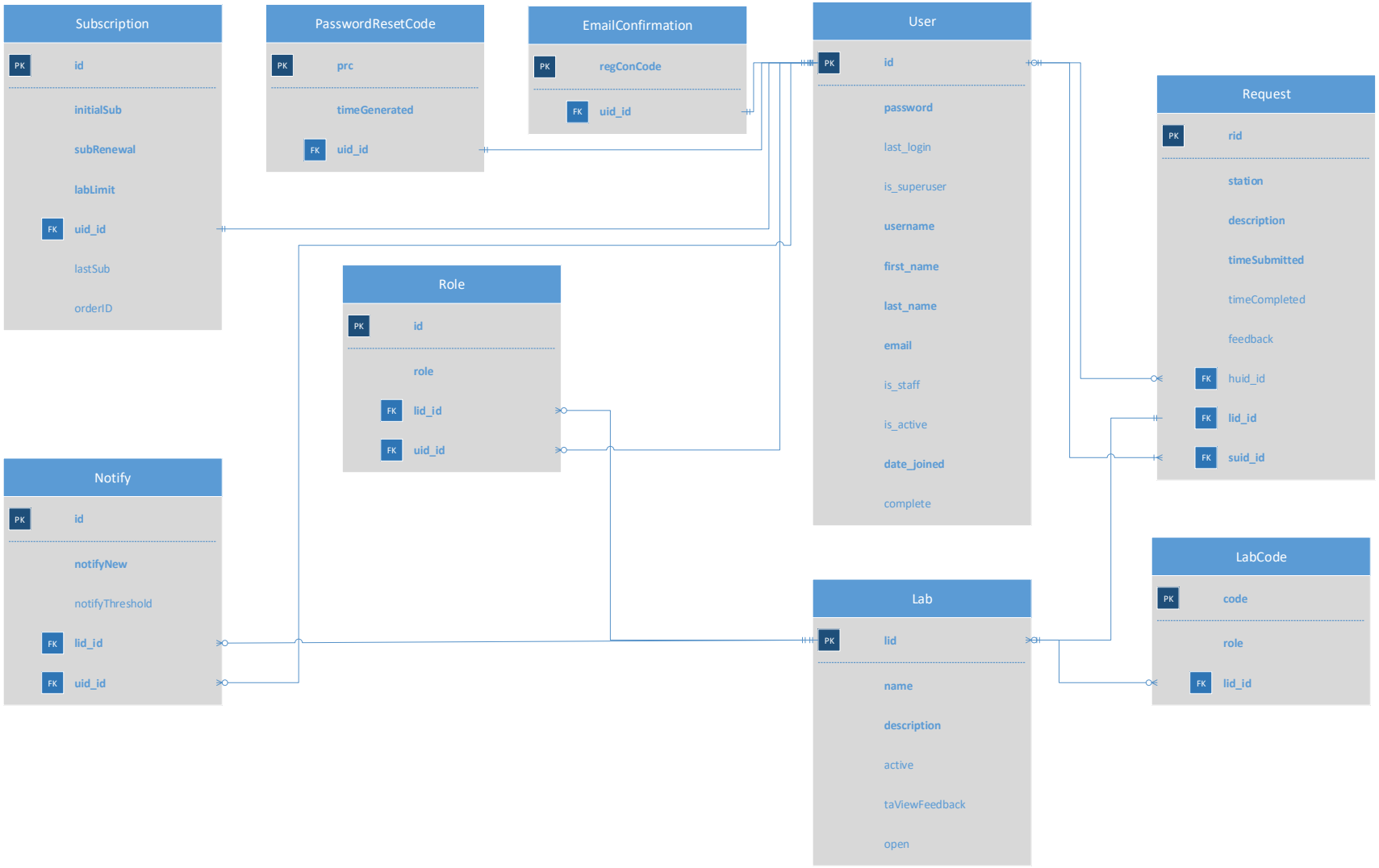
After deploying the beta version of the application, it was to be tested in several academic labs under the direction of Dr. Jason Bakos and his teaching assistants. However, due to COVID-19, they were unable to meet in person and use LabLineup in their labs. The following release was tested extensively by another team of students. All of the bugs they found have been resolved.

Conclusion

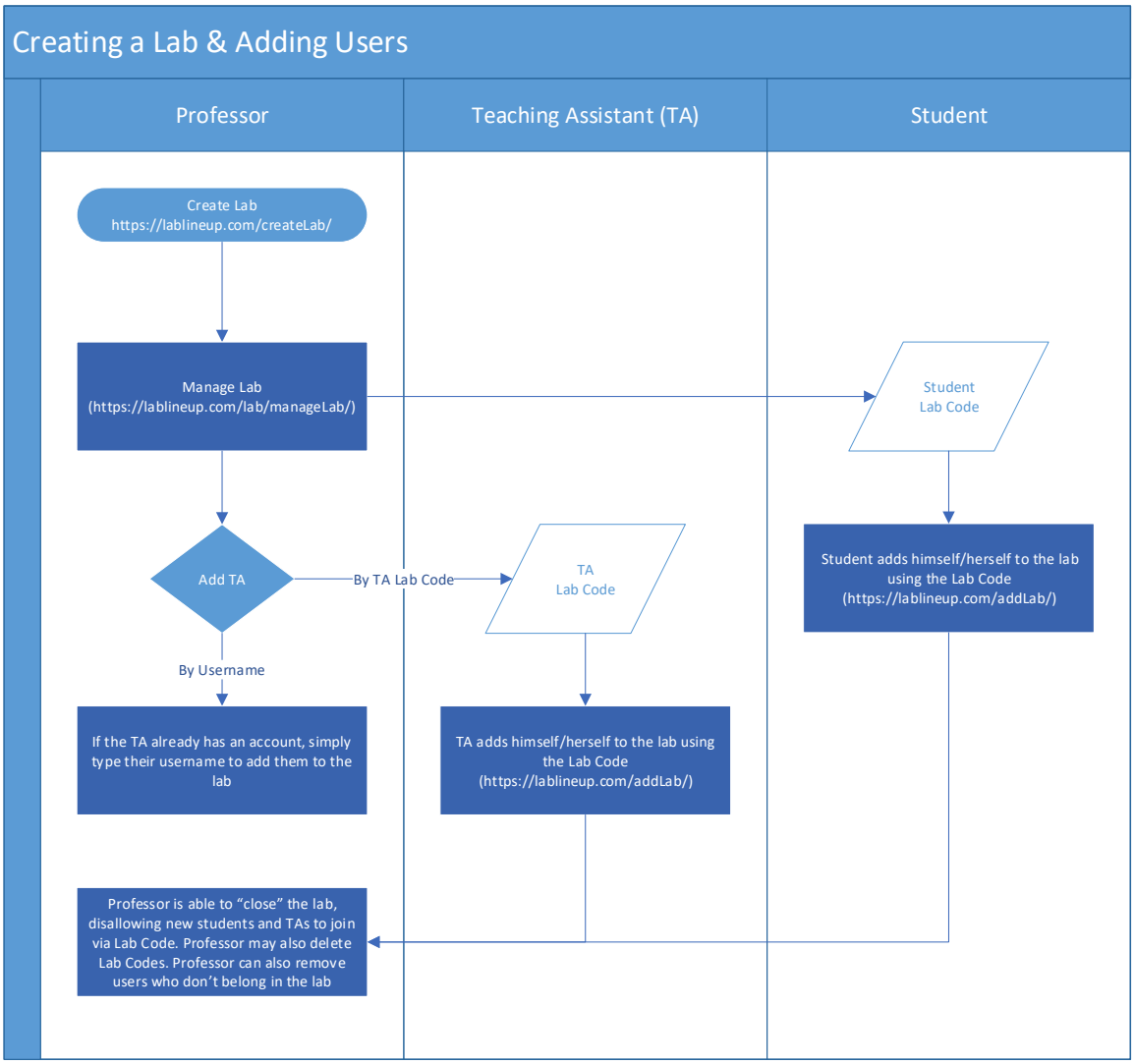
LabLineup aims to help professors, teaching assistants, and students in academic labs more efficiently handle requests for help. LabLineup is a responsive web application that can easily be used on a computer, phone, or tablet. LabLineup allows professors to easily create and manage labs. Students can join the labs and submit requests for help. Professors and teaching assistants can easily accept the next request for help with a click of a button. Professors and teaching assistants no longer have to search through a sea of hands to see who was next in line for help. LabLineup is a free SaaS but may be upgraded to a paid subscription in order to add more labs. Student and teaching assistant accounts are completely free. LabLineup solves the common issues in large, academic labs.

Appendix

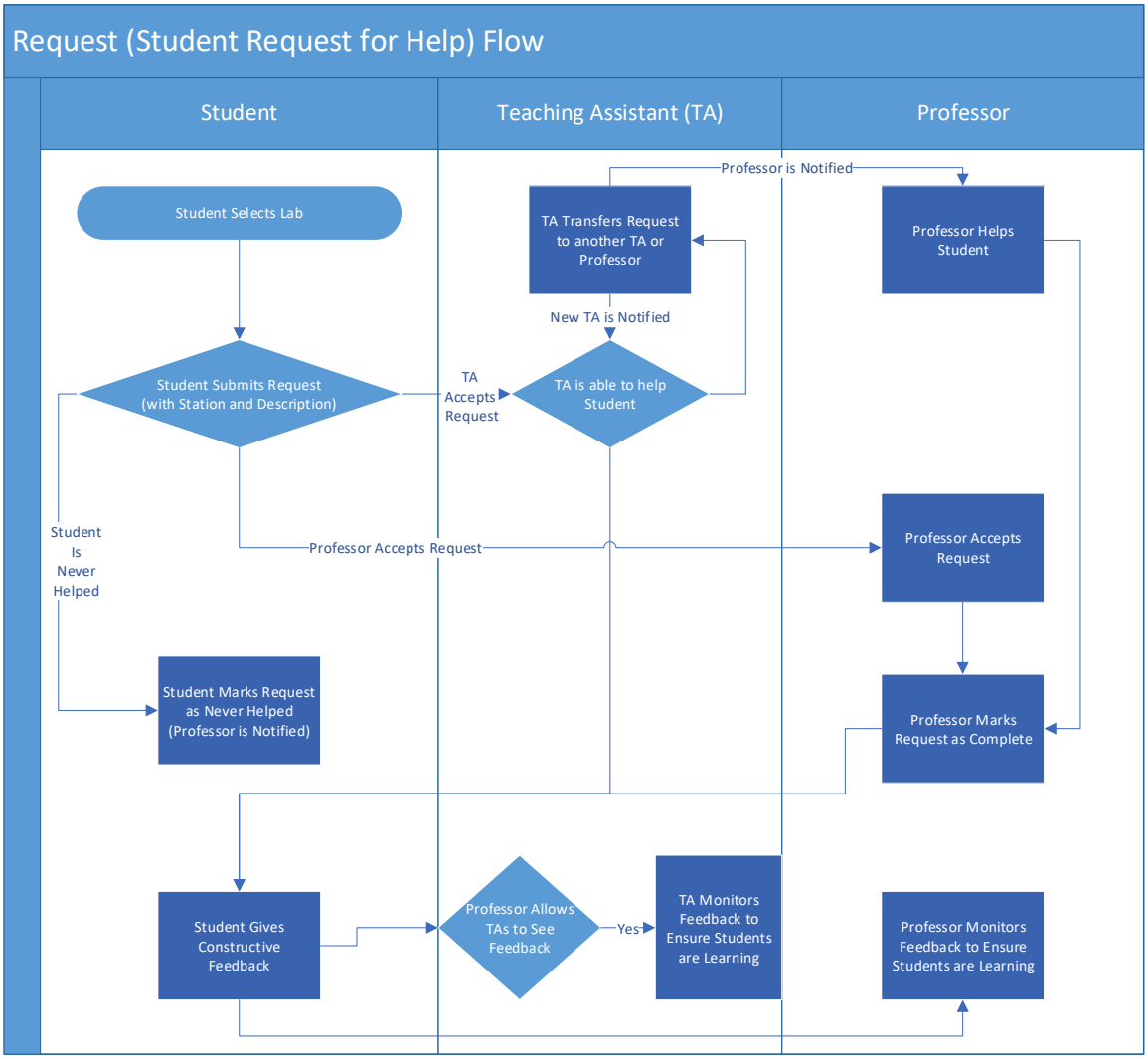
Database Schema



Creating a Lab & Adding Users Process Flow



Request Process Flow



References

[1] Django Project, "Django Overview," Django Software Foundation, [Online]. Available: https://www.djangoproject.com/start/overview/.

[2] Django Stars, "10 Popular Websites Built With Django," Django Stars, [Online]. Available: https://djangostars.com/blog/10-popular-sites-made-on-django/.

[3] Bootstrap, "Bootstrap - The most popular HTML, CSS, and JS library in the world," Bootstrap, [Online]. Available: https://getbootstrap.com/.

[4] MailGun, "Transactional Email API Service For Developers," MailGun, [Online]. Available: https://www.mailgun.com/.

[5] Square, "Checkout API," Square, [Online]. Available: https://developer.squareup.com/docs/checkout-api-overview.

[6] Selenium, "Selenium IDE," Selenium, [Online]. Available: https://www.selenium.dev/selenium-ide/.

[7] OneSignal, "How to Increase Opt-In Rates for Push Notifications," OneSignal, 27 November 2019. [Online]. Available: https://onesignal.com/blog/increase-opt-in-rates-for-push-notifications/.

[8] Investopedia, "PCI Compliance Definition," Investopedia, [Online]. Available: https://www.investopedia.com/terms/p/pci-compliance.asp.