

# Book-A-Study Room

## **Deliverable #7**

December 3<sup>rd</sup>, 2018

The logo for HAXORS is displayed within a dark blue rectangular box with a light blue border. The word "HAXORS" is written in a bold, yellow, serif font.

Eliza Gaudio (1554032)

Elie Sader (6128748)

Jonnie Klein Quezada (1640380)

Kevin Hirsh (1634710)

Kyle Nancoo (1441915)

### **Client organization:**

Vanier College Library

### **Client name:**

Haritos Kavallos

## **Table of Contents**

Executive Overview.....	3-4
Final summary description of client/problem.....	4
Final narrative description of project.....	5
Client's comments.....	5-6
Future work to be completed.....	7-8
Appendix 1.....	8
Appendix 2.....	8-10
Appendix 3.....	10-13
Works Cited.....	14

## Executive Overview

This deliverable is our last deliverable for our project, and throughout this document, we discuss the final outcome of our prototype, and talk about our demonstration of the implementation to the client. We also talk about what work is left to be completed, what final business problem our prototype solves, and explains how to use the current system we implemented.

The first part of the deliverable talks about the final business problem that our prototype aims to solve. Initially, we had intended on creating a system for the Vanier College library, to allow students to be able to book study rooms. We wanted to attach it to Omnivox, but later on discovered the problems we would face by doing that, and proceeded to creating a separate website to access a reservation system. More details are explained throughout that paragraph, to give the user and the reader a better idea of what problem we wanted to solve with our prototype.

The second part discusses the final description of our project. In the beginning, we each had a bunch of ideas on how to implement this prototype, and what kind of system we wanted to design for Haritos. Towards the end, however, as we started having more meetings with our client, our ideas changed. Due to time constraints, we also did not code the entire prototype yet. If we decide to keep going with this project in the future, for our own benefit, we have lots left to do. For now, this section explains the gist of our prototype, and what parts we actually implemented throughout the last few weeks.

Following that, we have a section that summarizes our client's comments about the prototype we presented to him, and he also gave our team feedback on how to improve in the future, in the event that we work with other clients to build other prototypes. After finishing the implementation, we showed Haritos what our final product looked like, and he gave us very generous and informative feedback. In that paragraph, you can find all the information he provided to us.

Our document has a section that explains what work is left to be completed for this prototype. Given the fact that, as previously mentioned, we didn't have an unlimited amount of time to execute our actual product, and finalize the system, we chose what we deemed to be the most important to implement, and worked on the code for those sections. In this deliverable, you can find the explanation about which portions we coded.

In Appendix 1, we talk about the changes we made from our prototype to our actual implementation. The idea of our prototype and the vision we had significantly changed when doing the actual coding part of this assignment,

and throughout that section of this deliverable, you can see the description of how the changing process occurred.

Appendix 2 is an instruction manual on how to use the system. Even though it's fairly easy to understand how to work with our prototype, a particular user who may not be very tech savvy will need to know what they have to do to use our system. It's also good for our client to be able to follow fluently the process of using the prototype we created for him.

Last, but not least, Appendix 3 highlights which user stories we implemented, to make it easier to follow what we decided to code and execute in our project. For example, for the log in information, we implemented the user story for students to create an account. In this part of the deliverable, you'll be able to follow how we chose the sections to implement, and based on what stories that we wrote.

## **Final summary description of client/problem**

Throughout the semester, our goal for this project was to implement a system for Haritos and the Vanier College library. When we had first met with him to discuss a business problem, he and the librarians had mentioned that they had no previous system set up for bookings. Students showed up on the same day to book a room, and that was the only time they would know whether the room was available or not. After further discussion with the client, and more team meetings amongst the five of us, we decided to create a booking system. This system would allow students, teachers, librarians, and the admin (Haritos) to have access to room bookings for the study rooms in the library. The idea of the system is to facilitate the process of getting a study room. For example, if a student wanted to book a room to study for their exam period within a week's time from today's date, they would be able to access the booking website and plan out their study sessions. We had a slightly different plan at the beginning on how to implement the project. We wanted to attach the website to Omnivox, but then later on learnt that they had a third party software working with them, and had no way of attaching the booking system there for now. In the future, if Haritos decides to further pursue this prototype, we'd be able to work with them. After discovering that hurdle, we went on to deciding it was best to create a separate website to begin the implementation. Essentially, in retrospect, our new system would be able to solve their problem of not having a booking system for their library, and with more work and effort, we would be able to create an entire system that the college could use forever.

## **Final narrative description of project**

We did not implement the entire prototype plan we had because of our time limit, however we worked on the parts we believed were important to display. Our prototype showcases the analytics for our system, as well as how to log in and view the booking schedule. It also shows how to update student information, how to delete a student from the database, and how to create a new account. Our goal was to create a system that would prompt a user to enter their account information for an existing account, and create a new account if they never signed up to book a room, and be able to access a system for study room bookings. Neither a student nor a teacher would be able to book a room, unless they were registered on the booking site. After that, once logged in, they would have the opportunity to pick whichever available time slot there is in order to get a study room, and they can see the schedule to view the availability of the rooms. A student would be able to cancel their reservation at any given time, or even change it, and they would also be able to create group bookings with their friends. Teachers would have the same opportunity. They would be able to book a study room to hold a lecture for a particular group of students. Librarians and admins would see the booking system differently than the students and teachers, however. A librarian would have access to deleting/modifying a student's reservation, especially if that student went against the rules in the past. They would also be able to see what type of room was booked, during what time it was booked, and they would get notified when a student surpasses their booking time limit. A librarian would also know if the student didn't show up, or showed up too late after their booking. In addition, a librarian would have the opportunity to add comments to a student's profile. An admin, in this case, Haritos, would be the only one able to see the analytics of the system. The analytics show how many times a student logs in to the system, and which program they come from. It will also eventually show how frequent bookings occur. Haritos would also be able to delete a student from the database, as we further develop the prototype, in the event that a student graduates from the college, and should no longer be part of the system. He sees everything about the system, since he is the administrator.

## **Client's comments**

Upon finishing our prototype, we requested feedback from our client, especially because we wanted to know what he thought of all the work we put into it, and to see if he wanted us to continue on with this prototype eventually.

Haritos began by thanking us for working so hard on this project all semester, and that he appreciated how much time and effort we spent working with him

throughout the last few weeks. We also gave him feedback on how to further pursue with this prototype, and he thanked us and wanted us to continue to work with him in the future, if we wished to continue with our idea.

He gave us very informative feedback on how to improve the system, as well as how to improve as a team. He also gave us such nice compliments on what we did well together as a team, and what we did well individually. He mentioned that our team planning was great. We didn't always get to meet with him in person, as our schedules sometimes ran into conflict, and he told us that we did very well on finding alternative ways to show him our ideas and ask him questions. We would always either email him or mio him, and we kept him up-to-date with everything we were doing throughout the semester. He told us that he loved being aware of our step-by-step process, and that we worked really well on communication.

Another thing he complimented us on was our design. When we presented him the prototype, he told us that the design was really nice, very simple, and esthetically pleasing to the eye. Furthermore, we complimented us on our overall sketches, and everything we did to get where we did. He told us that the sketches we provided to him before the actual implementation were clean and concise, and that we made a very serious effort on understanding the branding, and we incorporated good UI practices. Because he is a more visual person, he appreciated having pictures to the ideas we provided him.

He also mentioned how he really liked our willingness to ask him questions and improve on the prototype. We were always asking him for information and feedback, and it showed him how much we truly wanted to learn. He told us that we never asked too many questions, and we always asked the right questions. It made us happy knowing he appreciated our will to learn and make our project, as well as ourselves, better. In relation to that, he also mentioned how we each took initiative one time or another on being a team leader, and that he found it effortless/respectable how we each took a stand and took responsibility for our given deliverable and role in this project.

The one thing he did mention for us to improve on was how to plan meetings, and that we should have planned the meetings further in advance, or stick to our suggested meeting time slots, as future clients may not have as flexible a schedule as he had.

Other than that, Haritos was very pleased to have worked with us, and he wished us all well in our academic and business careers. He did mention that he would like to work with us again in the future, and congratulated us.

## **Future work to be completed**

Given our time constraints, and the complexity of the project, we were only able to implement the code for a select few aspects of the prototype that we had envisioned. Creating a booking system for the Vanier College library is a rather long process, as the system itself requires a lot of detail.

There are quite a few parts of the prototype left to implement. For example, we wanted to create a way to distinguish whether it's a teacher, an admin, or a student logging on to the booking system. If a student logs on, it would redirect them immediately to a page where they would see a schedule, and have the opportunity to book a room. They'd also be able to modify their profile, see their reservation, etc. If a librarian would log on, they would be able to delete that particular student's reservation, in the event that they are not eligible to actually book because of an infraction, edit a reservation, manage the availability of the rooms, see which type of room was booked, and would be notified of things like a breach of reservation time. As for admin, he would be able to see the analytics page, and have access to the main database. In our current prototype, anybody creating an account or logging in can see these pages. Because of the time constraints, we didn't manage to separate the different types of logins. With more time, eventually there will be a way to differentiate what type of user is accessing the system.

We wanted to create a section for the teachers to be able to create a group booking. Essentially, when a teacher would log in to the booking system to access the calendar and book a study room, they would be able to create a group booking and select which students from their class would be allowed in to that study room. Since we were limited with our time, a teacher would just be able to log in and book a room, with no way to create a group.

In addition to the previous paragraph, students can't yet create group bookings either. They're only able to log in to the system, or create their account, and then from there, book a single study room for themselves. In the future, it would be nice to have the possibility of booking for their group of friends, so that those who wish to study together have the opportunity.

Librarians would have also had a different way of viewing information on the booking system. We had wanted to have them be able to add notes on a student's profile, in case they breached an agreement, or did something that would merit them a warning from booking any future study rooms. Furthermore, if the librarian took a look at student bookings on the calendar, and saw that the particular student with a warning tried to book another room,

they would have had the option to remove their reservation altogether. This is another work in progress.

We would also need to add pre-loader animations. A pre-loader animation is what you see on a page while content is still loading. For example, when you log in to outlook, when you've entered your email and your password, an animation appears while you are waiting for your information to load. The reason we left that out is because the backend is in local. That changes things since requests happen instantaneously. If we decide to go further with this prototype, it would need a pre-loader so that the user doesn't think the site crashed while their request is processing.

## **Appendix 1**

The theme of the website was changed, due to time restrictions. We used a template from a website builder as the theme, instead of our initial idea of creating a whole separate website, or wanting to attach it to Omnivox (as mentioned in previous paragraphs).

We also did not implement the booking system for the students, because we thought having a fully functional administrator account over a student was more crucial to demonstrate our system. For example, the reservation schedule displays pre-set data, and you cannot book a reservation in the current implementation we have. However, you are able to see what it would look like if the student/teacher had the opportunity to book an actual study room. Within our implementation, we included proper charts to provide information regarding logins and course demographics.

What is also different is that the logins with the current implementation is set for admins only, where even trying to log in as a student would give admin privileges. In the future, we intend on separating the account privileges, as we do not want to run into complications, and have a student accidentally delete another student from the database, or have access to personal information, especially things like analytics and student information.

## **Appendix 2**

Although the system may appear to be easy to use, it still requires an instruction manual, that way someone who is unsure of what to do has a set of guidelines.

The project is conveniently divided into two distinct parts: the frontend and the backend. The technologies used for the frontend is React.js.



As for the backend, Node.js was used. Although there are clear benefits to this decision, it has not come without repercussions. For instance, Node.js is a single-threaded run-time environment which means it is not suitable for CPU-intensive tasks such as graphics development. Fortunately, the system is using Node.js in an I/O heavy environment, where it excels. Each tool is as useful as its master's intent. By using Node.js in a way that matches what it was conceived for, we can say for certain that the server will be able to handle requests at a faster rate and bigger ratio than by using Ruby on Rails or PHP. As a data-driven solution, if we were to use it for advanced complex calculations, it would significantly lower its performance to the point that it will be unfeasible to use it over a multi-threaded solution such as PHP.

We have put an emphasis on scalability. We want the system to work, as intended, without setbacks, no matter how big the user base is. Using technologies that support or accentuate the data-driven concept of the application will only ensure that it handles the users' needs. With the little time we had to work on the implementation, there are many improvements to be made. For instance, let's take the analytics page as an example. One problem with the analytics page is that it sends multiple requests to the backend for data that are closely related. It sends requests to get the number of logins, get all the users, etc. And so, instead of sending 5 or more requests to the backend each time the analytics page loads, it would be much more efficient to have a route handler `/getAnalytics`, which will pass all the data the analytics page requires.

To explain more about how the system works (the parts we coded and actually implemented), you start off by accessing the page, where you will be prompted to enter your account information. For a user who already has an account, all they have to do is enter their student ID and their password, and they will be redirected to the schedule page immediately. For a user without an account, they will have to create their account by using their ID, selecting their program, entering their name, and then their account will be created. After they do that, they too will be able to see the schedule for bookings.

The next page is for admins. Currently, everyone can see the analytics page. However, that is something we will change and improve, as we continue on (if we decide to) with this prototype. You can see by accessing the analytics page that you can view which program the user belongs to, and also see how often they log in.

The scheduling page shows time slots for the user to be able to pick and book. You can see the schedule, and you are able to change the month, as well, to book for a specific period of time in advance. A student has to click on a

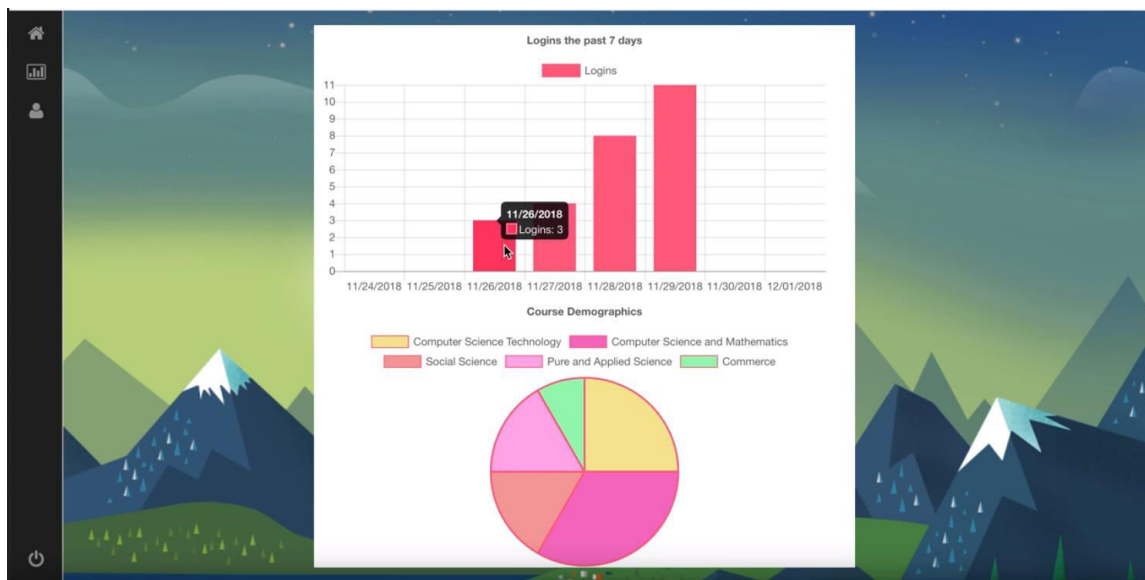
specific day, and they can select which time they want to book a study room for. They can't book too far in advance, however. We had discussed with Haritos that we wanted to set a time restriction for bookings, so that students would get a fair chance to request a study room.

All in all, the user guide seems fairly simple, since we did not have the time to implement all the sections we wanted to. In the future, our instructions would be much more in depth, because there would be a lot of different sections to consider in our prototype.

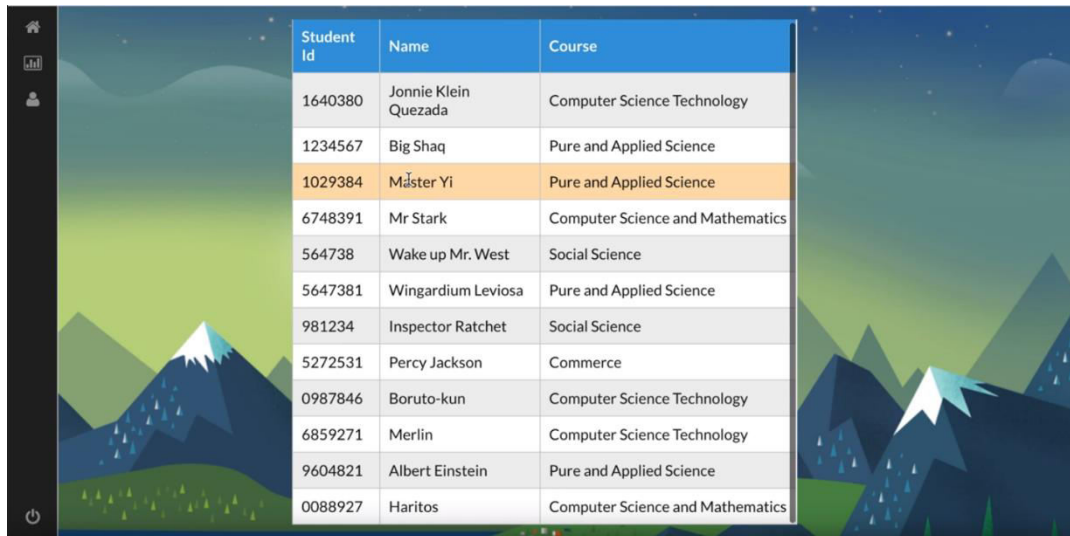
## Appendix 3

1. As an admin, I want the analytics page to have a “demographics” graph in order to see the number of students in each program reserving study rooms.
2. As an admin I want the analytics page to have a graph indicating the number of reservations each hour so that we can learn the peak times or peak periods.

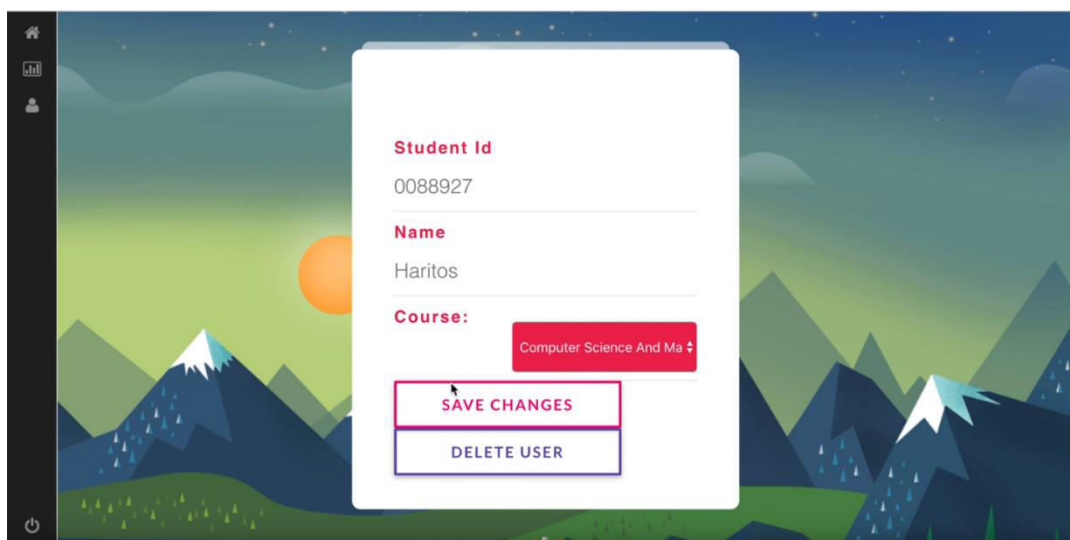
(For both 1 and 2)



3. As an admin I would like to have access to the database to remove students that have graduated and update student information.



Student Id	Name	Course
1640380	Jonnie Klein Quezada	Computer Science Technology
1234567	Big Shaq	Pure and Applied Science
1029384	Master Yi	Pure and Applied Science
6748391	Mr Stark	Computer Science and Mathematics
564738	Wake up Mr. West	Social Science
5647381	Wingardium Leviosa	Pure and Applied Science
981234	Inspector Ratchet	Social Science
5272531	Percy Jackson	Commerce
0987846	Boruto-kun	Computer Science Technology
6859271	Merlin	Computer Science Technology
9604821	Albert Einstein	Pure and Applied Science
0088927	Haritos	Computer Science and Mathematics



**Student Id**

0088927

**Name**

Haritos

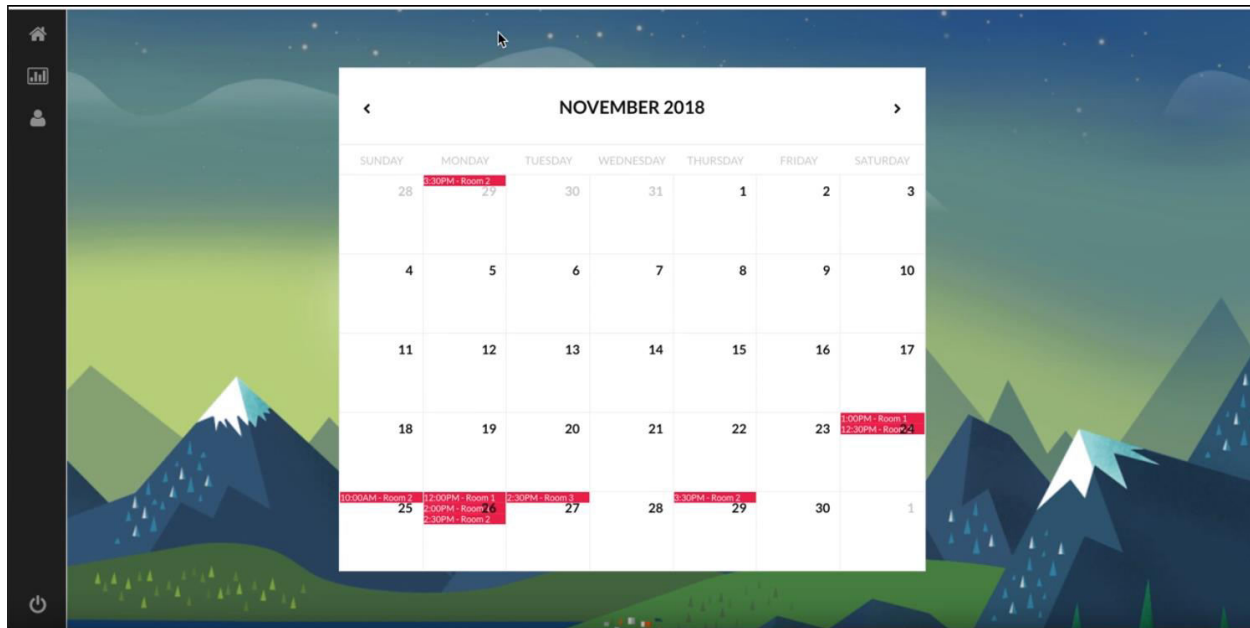
**Course:**

Computer Science And Ma

**SAVE CHANGES**

**DELETE USER**

4. As a librarian I can add/remove students from a reservation (students and teachers can delete their reservations, but librarians can remove them if they have a valid reason to).
5. As a librarian I can change the hours in which rooms can be booked (also for a student and for a teacher).
6. As a librarian I would like to see the schedule times (also for a student and for a teacher).



7. As a librarian I would like to have a registration system in order to see who is reserving a room.

**REGISTER** +

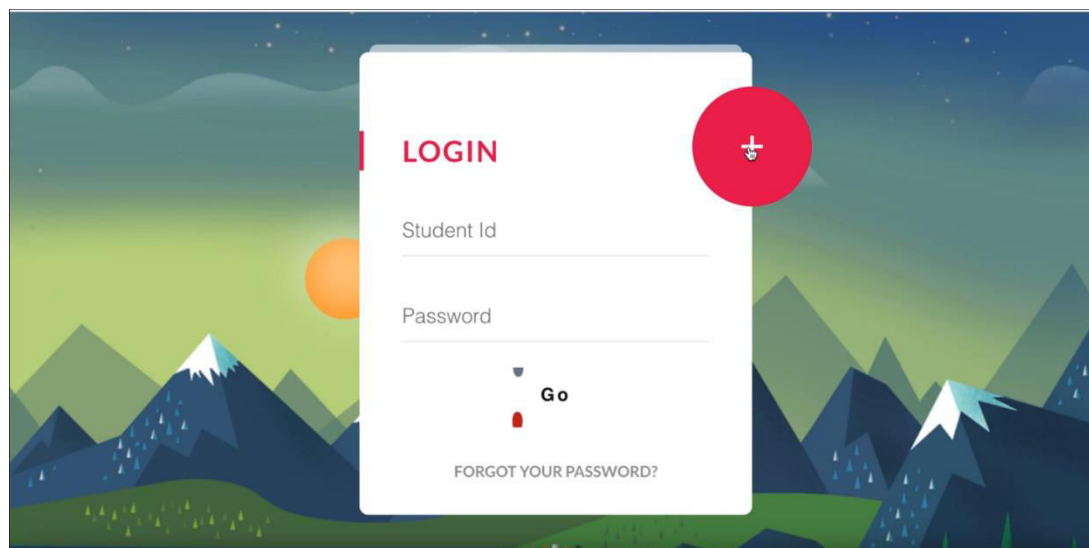
Student Id

Full Name

Course:

Password

Confirm Password



## **Works Cited**

- No references used for this deliverable.