SW Engineering CSC648/848 Spring 2024 Section 2

Olympic Quest College Search

Team 02

Team Lead - Shriya Dandin (sdandin@sfsu.edu)
Front End Lead / System Administrator - Victoria Bialorucki (vbialorucki@mail.sfsu.edu)
Back End Lead - Sam Zandiasadabadi (szandiasadabadi@sfsu.edu)
Front End Team - Erika Chan (echan10@mail.sfsu.edu)
Github Master - Jake Klopukh (jklopukh@sfsu.edu)
Back End Team - Omar Ahmed (oahmed1@sfsu.edu)
Database Engineer - Subodh Khadka (skhadka@sfsu.edu)

Milestone 2
3/12/2024

Date Submitted	Date Revised
3/12/24	

1. Functional Requirements - prioritized

Functional Requirements:

1. Allow athletes to create accounts - giving each user-athlete the ability to create an account using their name and information to be seen.

2. Allow athletes to log-in to their accounts - giving each user-athlete the ability to log-in to their account given their account registration information.

3. Allow athletes to modify their accounts whenever - giving each user-athlete the ability to add, remove, or update their personal information whenever they need to.

4. Allow recruiters to create accounts - giving each user-recruiter the ability to create an account using their name and information to be seen.

5. Allow recruiters to log-in to their accounts - giving each user-recruiter the ability to log-in to their account given their account registration information.

6. Allow recruiters to modify their accounts whenever - giving each user-recruiter the ability to add, remove, or update their personal information, or guidelines for accepting/scouting athletes whenever they need to.

7. Allow admins to be assigned a unique master account - assigning an admin for the website who would be in charge of maintaining the website.

8. Allow admins to modify all users' posts - giving the admin the ability to modify both the athletes, and the recruiters profiles.

9. Allow athletes to apply/reach out to colleges/recruiters - allowing the user-athletes to reach out and connect to college recruiters to learn more about their program.

10. Allow athletes to apply/reach out to multiple colleges/recruiters simultaneously - similar to the previous requirement, however, ensuring that the user-athlete can contact multiple college recruiters at the same time.

11. Allow recruiters to reach out to athletes - allowing the user-recruiters to reach out and connect to athletes in order to introduce them to their program and talk to them about their needs.

12. Allow recruiters to reach out to multiple athletes simultaneously - similar to the previous requirement, however, ensuring that the user-recruiter can contact multiple athletes at the same time.

13. Allow athletes to search for specific colleges - allowing the user-athletes to search for a specific college to learn about their program or to connect with their recruiter.

14. Allow recruiters to search for specific players - allowing the user-recruiters to search for a specific athlete in order to learn more about them and connect with them if necessary.

15. Allow admins to search for specific players or recruiters - allowing the website admins to search for both user-athletes and user-recruiters in case they need to identify, modify or remove any accounts.

16. Allow website users to sort data based on name - allow all users to sort the data within the website based on the name of the college to easier identify the college they are interested in.

17. Allow website users to sort data based on state - allow all users to sort the data within the website based on the state a college is located in order to showcase all available colleges in that specific state.

18. Allow website users to sort data based on year - allow all users to sort the data within the website based on the year they would be qualified to attend a specific college.

19. Allow website users to sort data based on sport - allow all users to sort the data within the website based on the sport they are interested in playing/recruiting.

20. Generate a list of colleges that matches an athlete’s preference/checklist - generating a list of colleges that best matches what a user-athlete is looking for.

21. Generate a list of athletes that matches a recruiter’s preference/checklist - generating a list of athletes that best matches what a user-recruiter is looking for.

22. Mark certain words as offensive and unacceptable - marking offensive, discriminatory, and unacceptable words.

23. Ensure no user can type/use offensive and unacceptable language - deleting or censoring words that have been marked as offensive or unacceptable.

Prioritized Requirements:

Priority 1 - Must Have:

Athletes:

1. Allow athletes to create accounts - giving each user-athlete the ability to create an account using their name and information to be seen.
2. Allow athletes to log-in to their accounts - giving each user-athlete the ability to log-in to their account given their account registration information.
9. Allow athletes to apply/reach out to colleges/recruiters - allowing the user-athletes to reach out and connect to college recruiters to learn more about their program.
13. Allow athletes to search for specific colleges - allowing the user-athletes to search for a specific college to learn about their program or to connect with their recruiter.
16. Allow website users to sort data based on name - allow all users to sort the data within the website based on the name of the college to easier identify the college they are interested in.
17. Allow website users to sort data based on state - allow all users to sort the data within the website based on the state a college is located in order to showcase all available colleges in that specific state.
18. Allow website users to sort data based on year - allow all users to sort the data within the website based on the year they would be qualified to attend a specific college.
19. Allow website users to sort data based on sport - allow all users to sort the data within the website based on the sport they are interested in playing/recruiting

Athletic Staff:

4. Allow recruiters to create accounts - giving each user-recruiter the ability to create an account using their name and information to be seen.
5. Allow recruiters to log-in to their accounts - giving each user-recruiter the ability to log-in to their account given their account registration information.

Admin:

7. Allow admins to be assigned a unique master account - assigning an admin for the website who would be in charge of maintaining the website.

Priority 2 - Desired:

Athletes:

3. Allow athletes to modify their accounts whenever - giving each user-athlete the ability to add, remove, or update their personal information whenever they need to.
10. Allow athletes to apply/reach out to multiple colleges/recruiters simultaneously - similar to the previous requirement, however, ensuring that the user-athlete can contact multiple college recruiters at the same time.

Athletic Staff:

6. Allow recruiters to modify their accounts whenever - giving each user-recruiter the ability to add, remove, or update their personal information, or guidelines for accepting/scouting athletes whenever they need to.
11. Allow recruiters to reach out to athletes - allowing the user-recruiters to reach out and connect to athletes in order to introduce them to their program and talk to them about their needs.
12. Allow recruiters to reach out to multiple athletes simultaneously - similar to the previous requirement, however, ensuring that the user-recruiter can contact multiple athletes at the same time.

Admin:

8. Allow admins to modify all users' posts - giving the admin the ability to modify both the athletes, and the recruiters profiles.
15. Allow admins to search for specific players or recruiters - allowing the website admins to search for both user-athletes and user-recruiters in case they need to identify, modify or remove any accounts.

Priority 3 - Opportunistic:

Athletes:

20. Generate a list of colleges that matches an athlete's preference/checklist - generating a list of colleges that best matches what a user-athlete is looking for.

Athletic Staff:

14. Allow recruiters to search for specific players - allowing the user-recruiters to search for a specific athlete in order to learn more about them and connect with them if necessary.

21. Generate a list of athletes that matches a recruiter’s preference/checklist - generating a list of athletes that best matches what a user-recruiter is looking for.

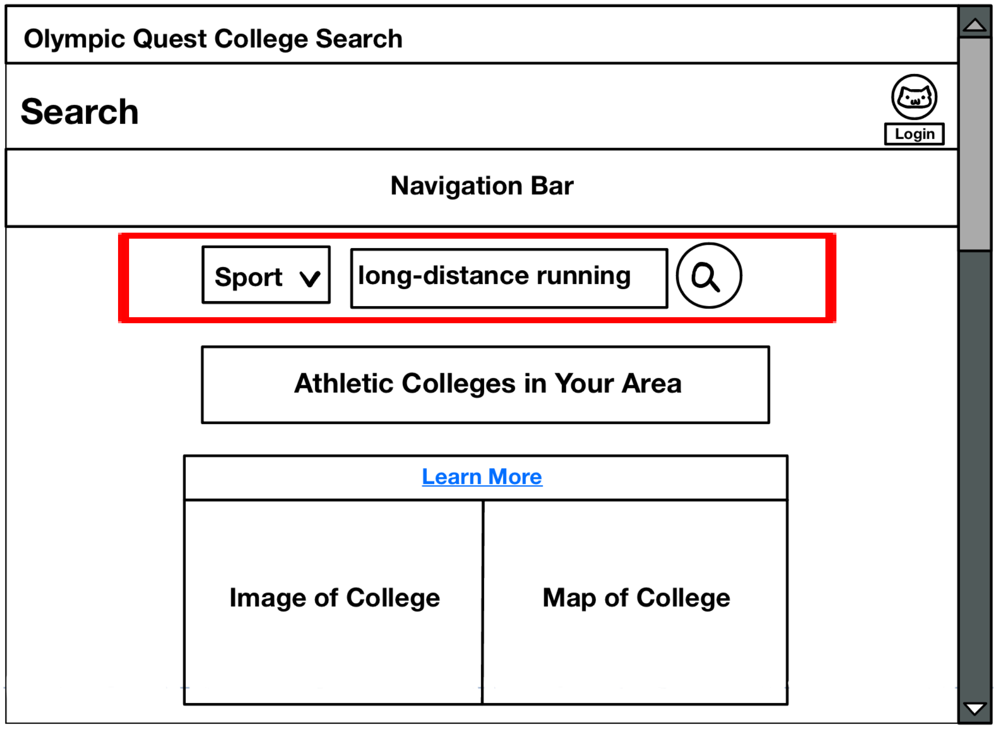
Admin:

22. Mark certain words as offensive and unacceptable - marking offensive, discriminatory, and unacceptable words.
23. Ensure no user can type/use offensive and unacceptable language - deleting or censoring words that have been marked as offensive or unacceptable.

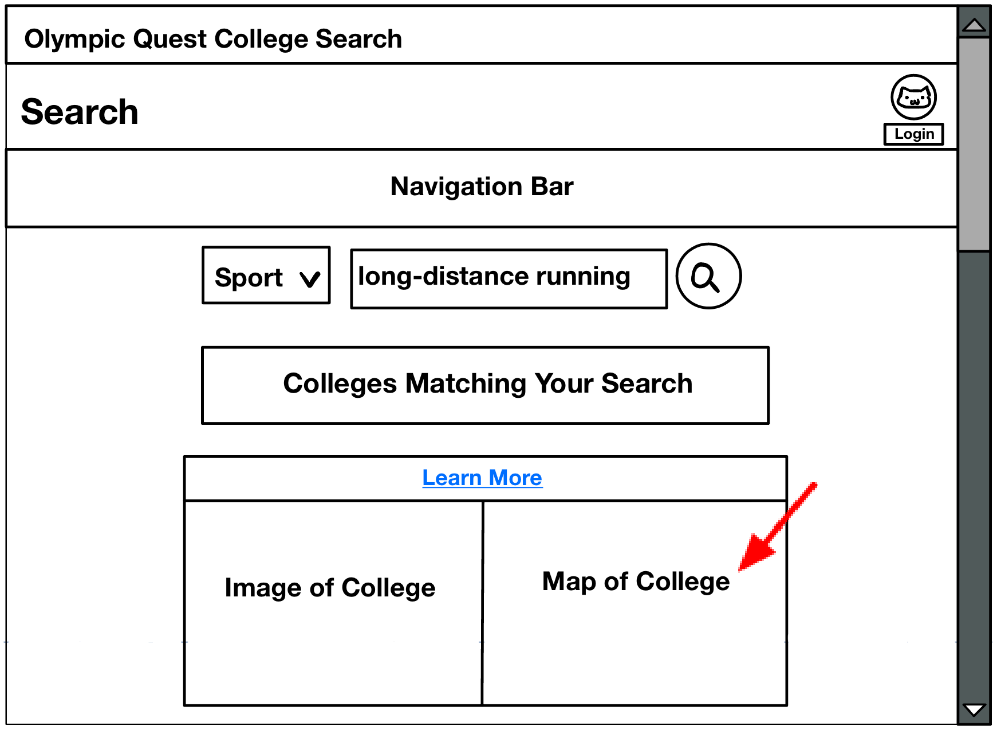
2. UI Mockups and Storyboards (high level only)

1. School Searching

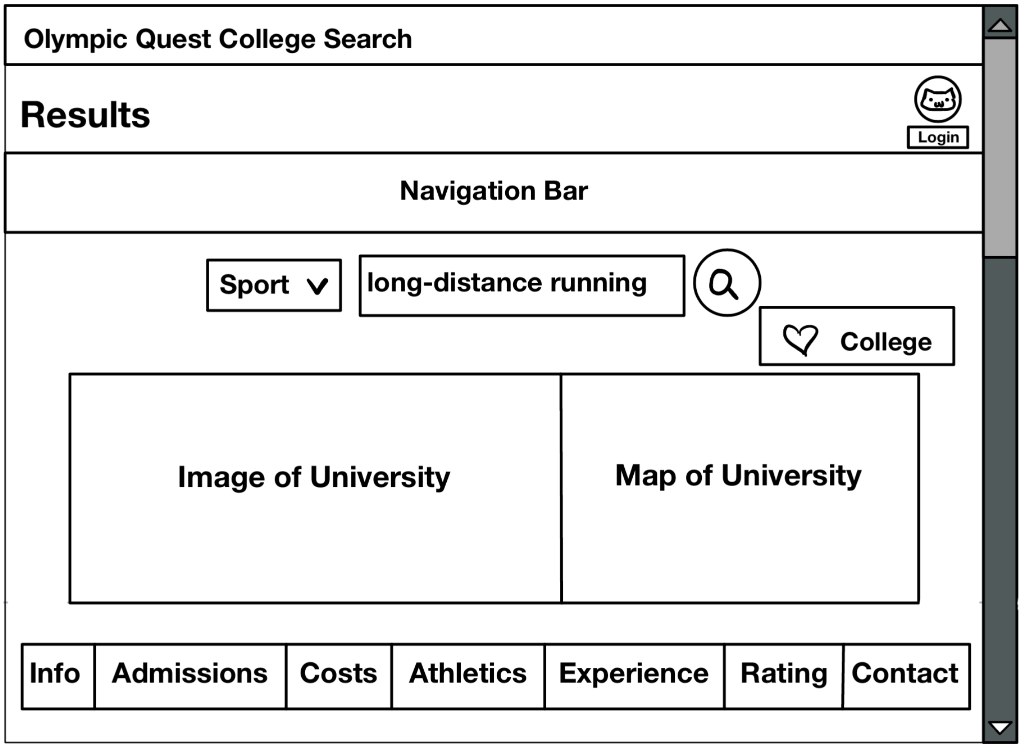
- Claire searches for long-distance colleges that specialize in running:



- Claire chooses the first college in the list:



- Claire can now see the college and details about it for non-logged-in users:



Sport
long-distance running

Summary

- By default, the search page will display the closest college based on ip address.

- Claire is a future student looking at the non-logged-in search. She is searching for long-distance running, particularly colleges that specialize in this sport.

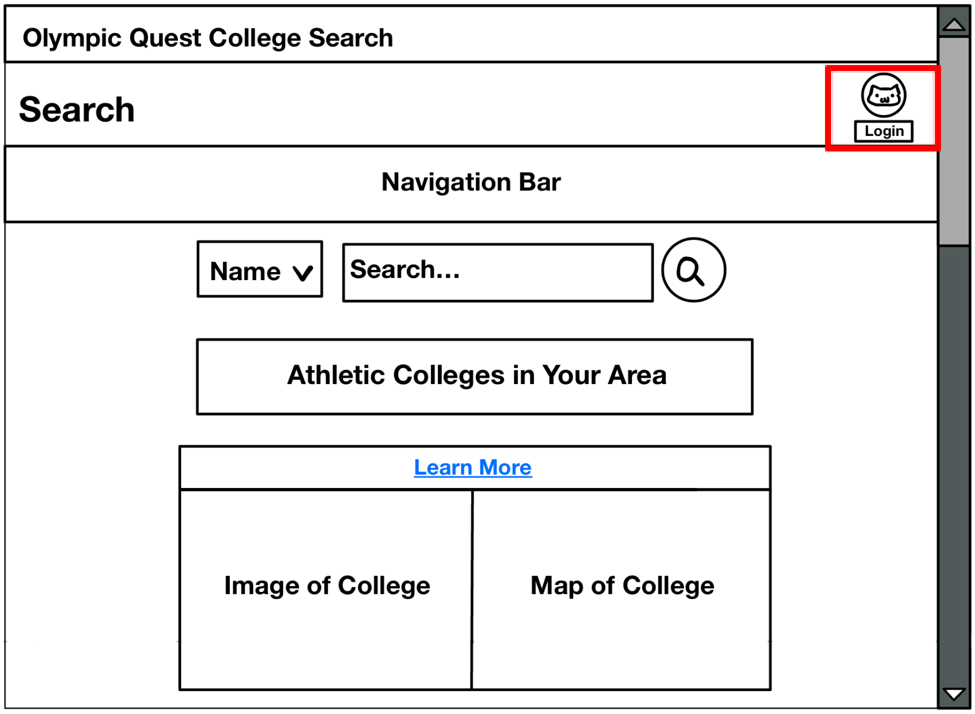
- Claire searches on the basic search functionality. This lacks a rating and messaging systems, and also doesn't allow for contacting college representatives.

- She finds a college in the list she wants to see, so she presses the “learn more” link for it.

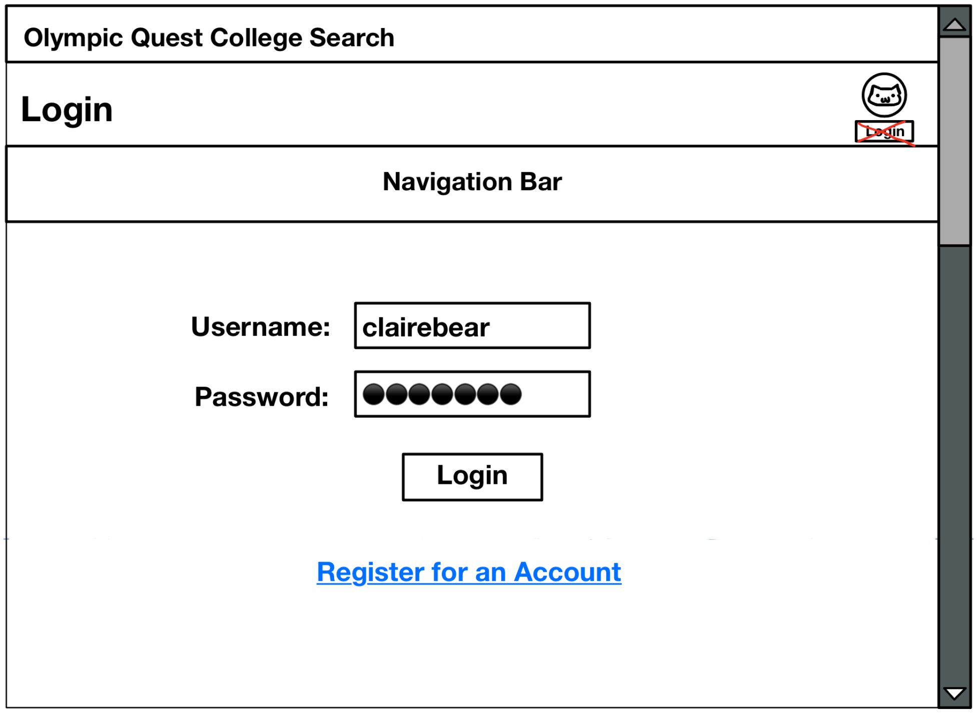
- The link takes her to the details page for this particular college. This details page has various tabs pertaining to: brief info about the school, admissions info, cost to attend, athletics that are specialized in, the overall experience of being at the college, and a rating and contact tab that can only be accessed if the user is logged in.

2. Student-School Contacting

- Claire wants to contact the school, but this requires her to be logged into her account:



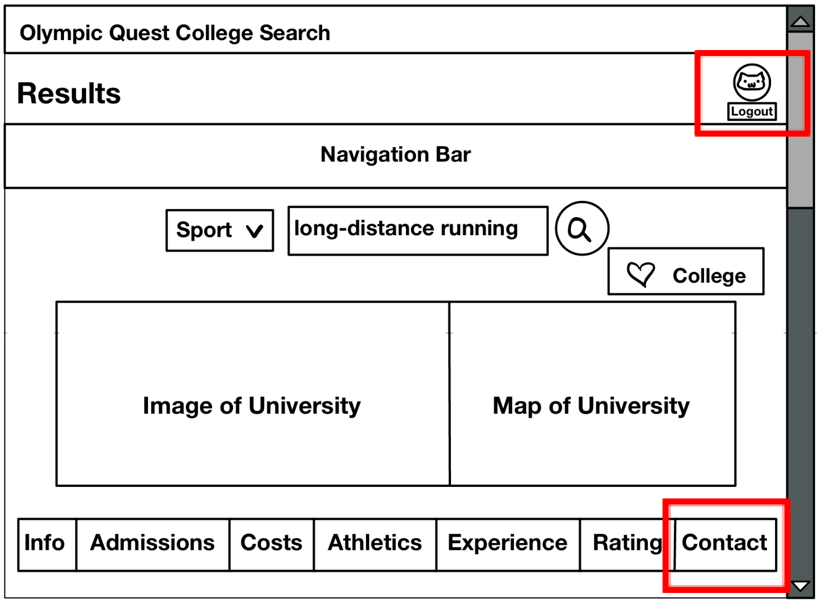
- Login is disabled in the upper-right, as this can cause issues. She now logs into her account:



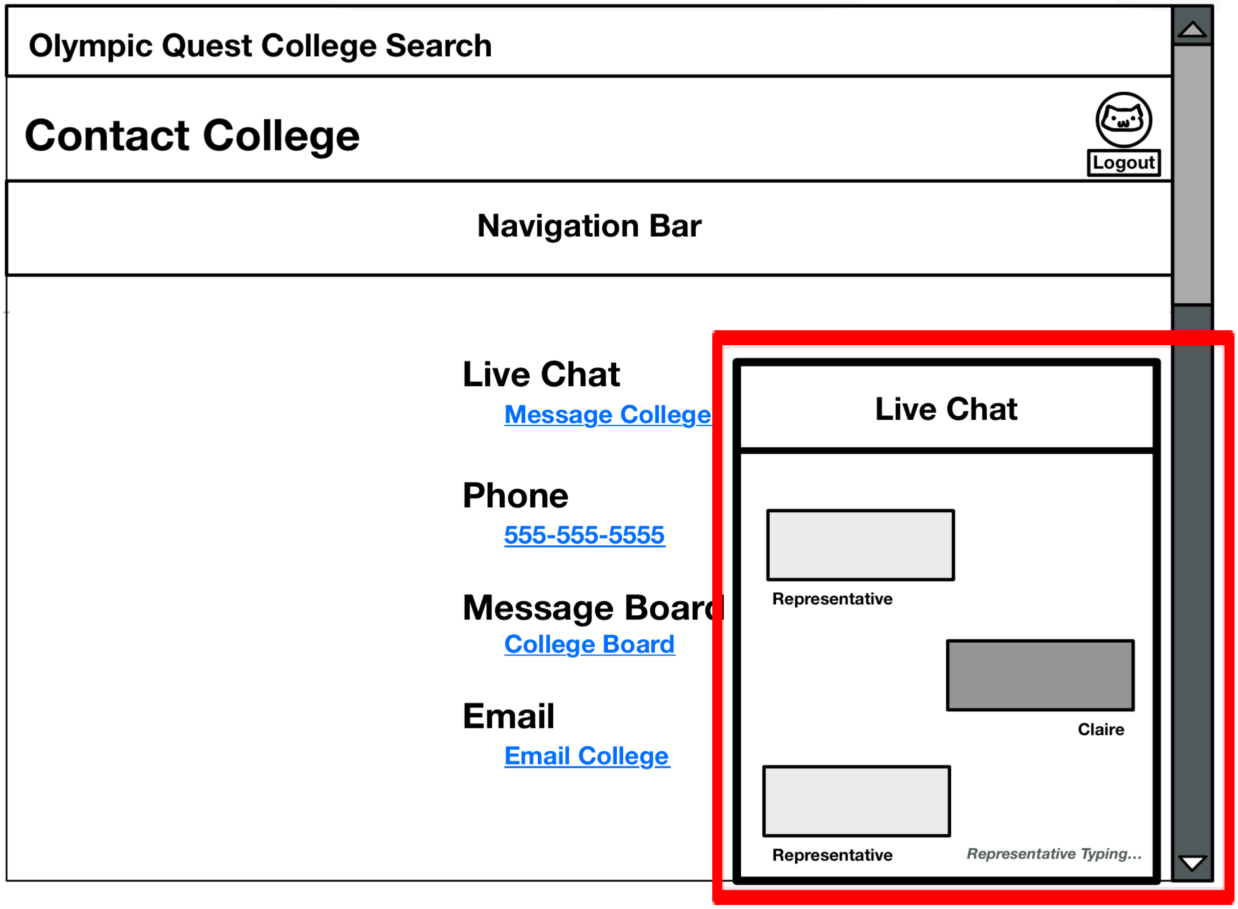
Username: clairebear

Password: ●●●●●●●

- Claire chooses a college based on her previous search. Since she is logged in, she can now reach out to the college via the “Contact” tab.



- Claire decides she wants to do a live chat, which allows her to connect with staff at the college (during business hours). The window for the live chat pops up at the bottom-right, where she can speak to a representative:

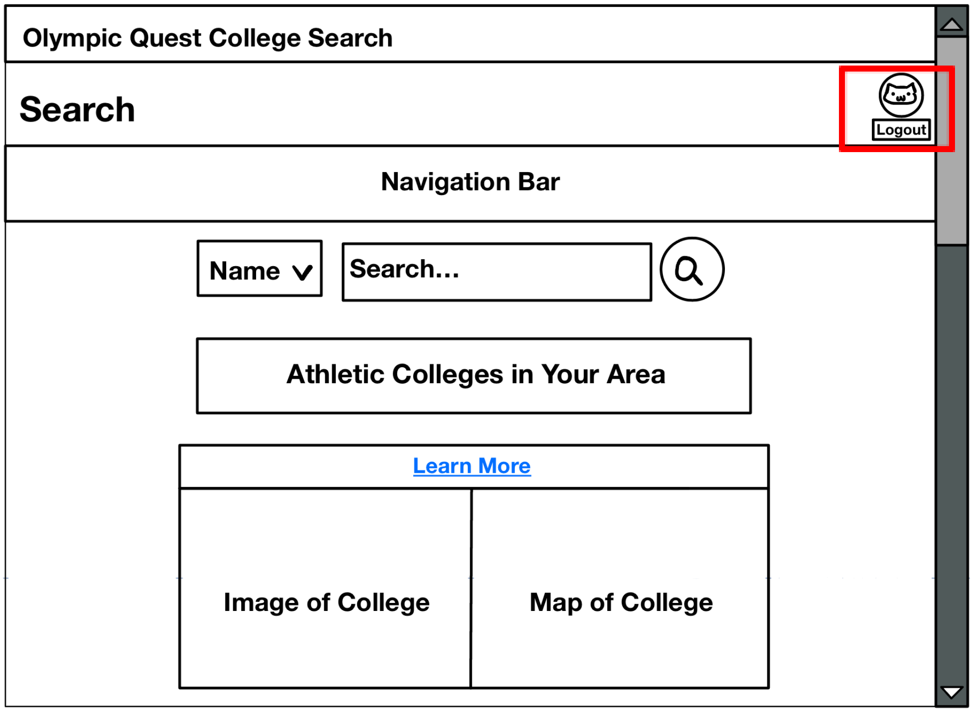


Summary

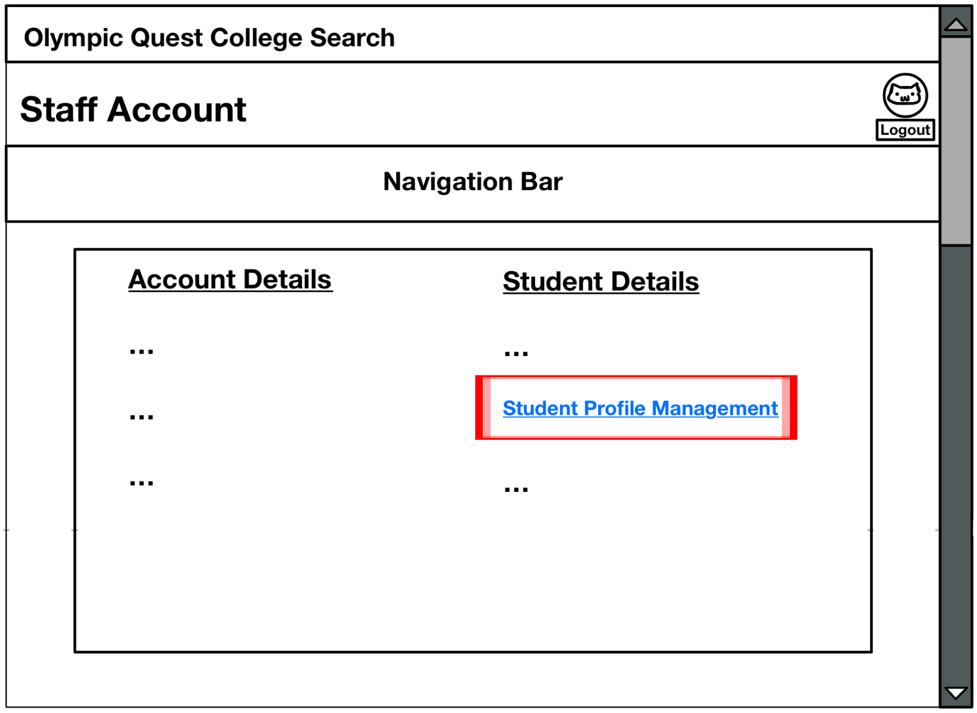
- Claire wanted to reach out to the college that she chose from her search.
- She noticed that she couldn’t reach out to the college she logged into her account.
- After she logged into her account, she was able to then access the “Contact” button on the navigation bar under the specific college’s details.
- After this, she then had the option to choose from: live chat with a representative from the college, call via phone, contact via the message board, or email. She chose the live chat in the example above.

3. School Staffer Page Maintenance

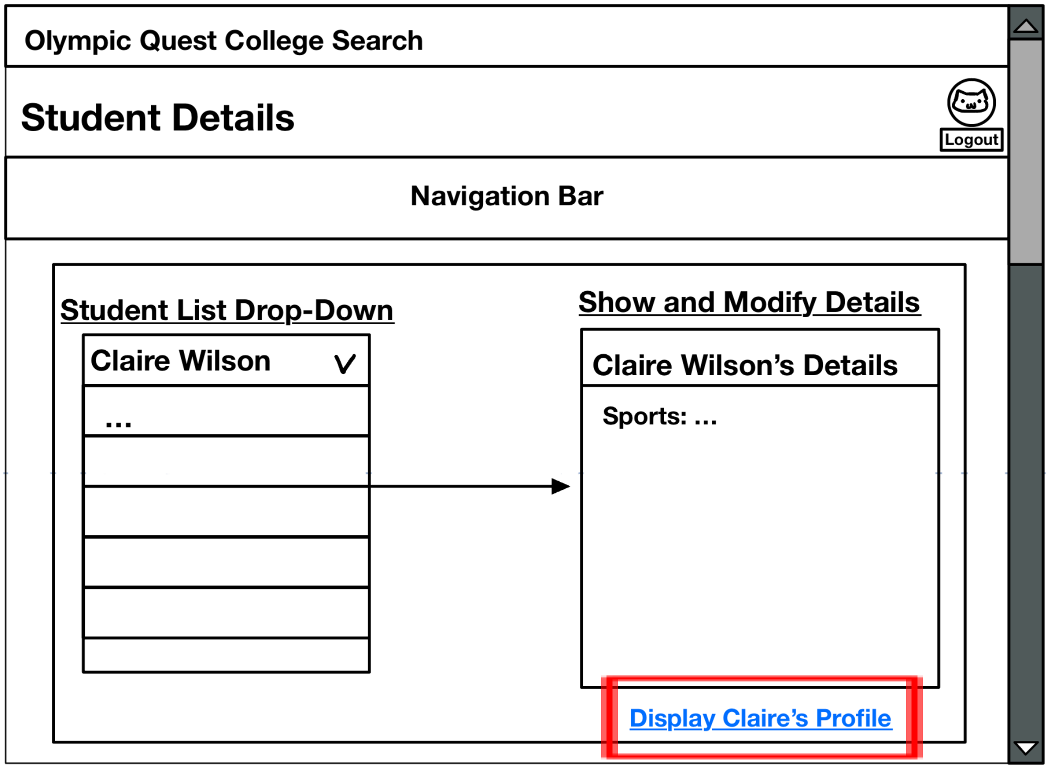
- Robert is logged into his staff account. Since he is a staff member, he is able to modify students' profiles at his college. He wants to add new student entries into the database. Once he modifies the student details page, it is reflected on the website. He can navigate to accounts via the avatar at the upper-right corner:



- He navigates to Student Profile Management:



- Claire was one of the new students added recently. He chose to modify her student details:



Olympic Quest College Search

Student Details

Logout

Navigation Bar

Student List Drop-Down

Claire Wilson V

...

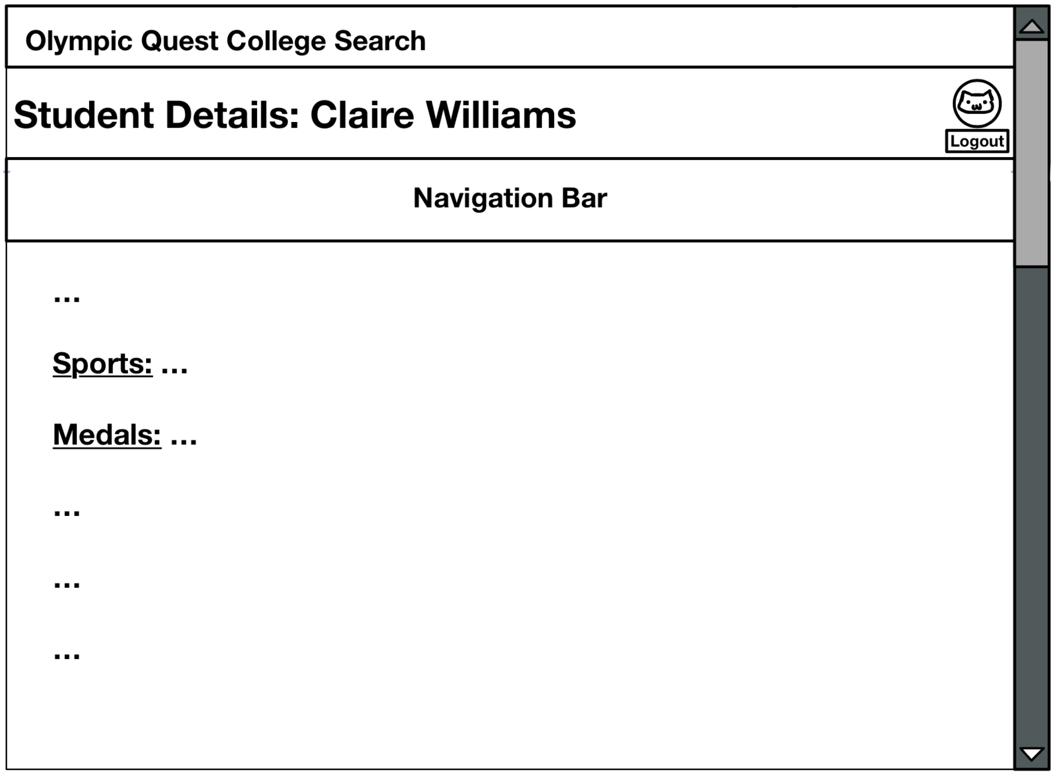
Show and Modify Details

Claire Wilson's Details

Sports: ...

Display Claire's Profile

- Robert is now able to see Claire's full, modified profile:



Olympic Quest College Search

Student Details: Claire Williams

Logout

Navigation Bar

...

Sports: ...

Medals: ...

...

...

...

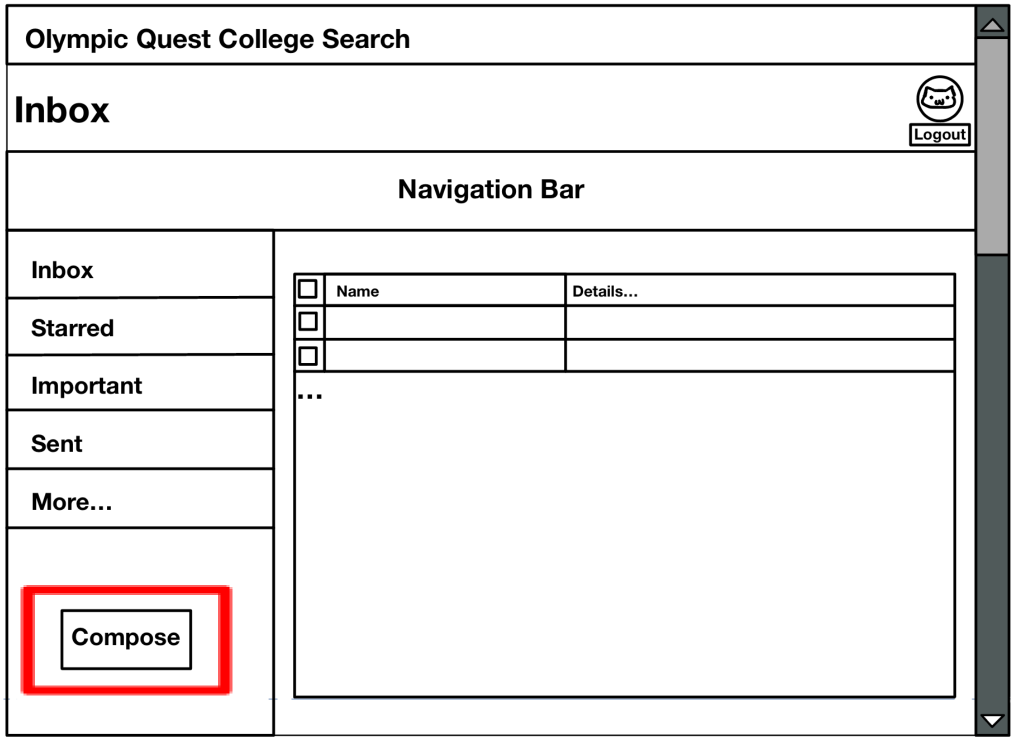
Summary

- Robert is a staff member who wants to change the details for certain students that attend the college. He mainly wants to update their Olympic achievements.

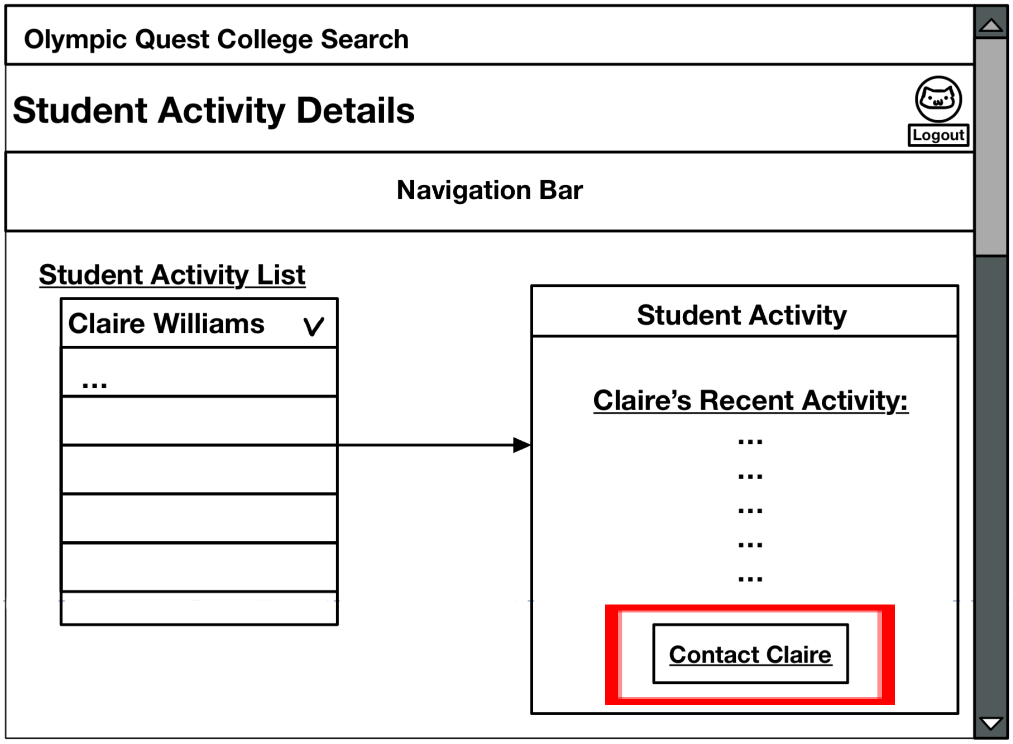
- He navigates to the student details page in his user account from the avatar button, then enters new information about each student. Once he is done, the student’s page is updated.

4. School-Student Contacting

- Via the nav bar, Robert finds his inbox. He can contact students with compose:



- Robert can see Claire’s activity, and also contact her for college recruitment:



Summary

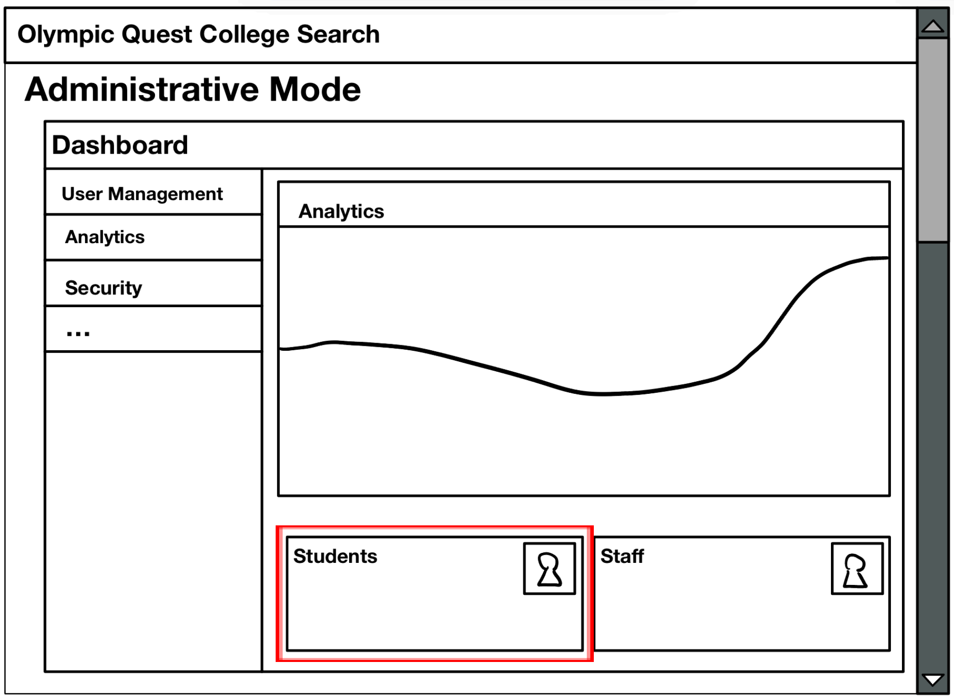
- Robert wants to get in contact with students interested in the college and recruit them.

- With our website's interface, he can message and email students who have reached out.

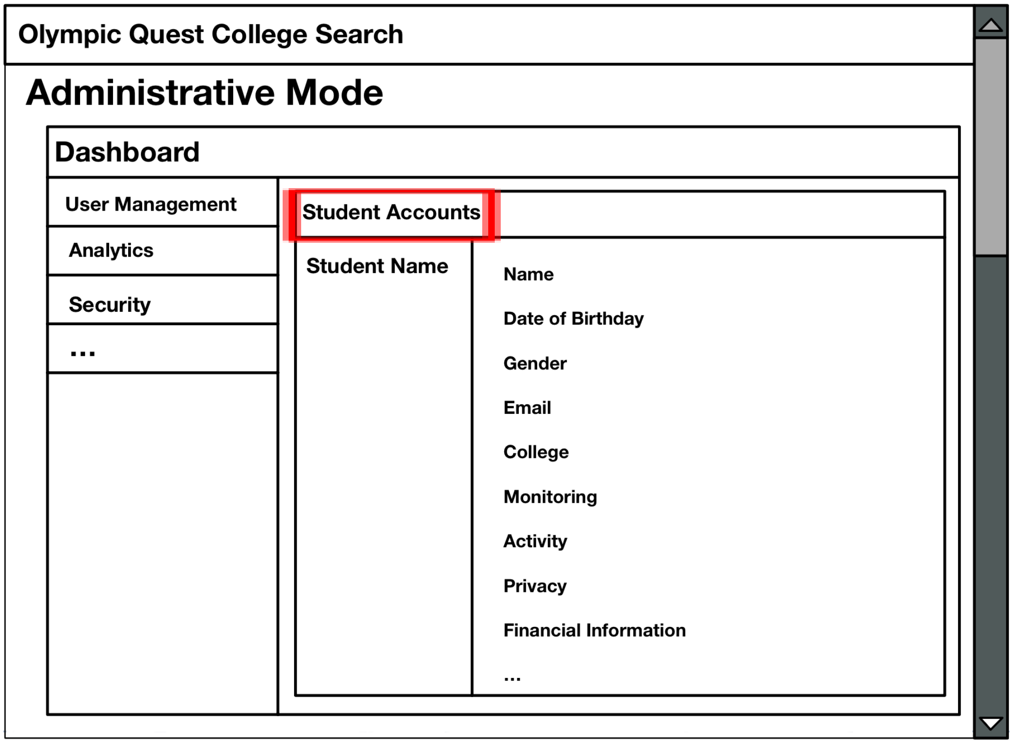
- He can also see the activity of students who have logged into their account and viewed the college he works for. WIth this feature, he is able to proactively recruit students or send messages to them.

5. Website Management

- Will can manage and moderate the website. Since he is logged in as admin, he can modify user info for staff members, students, and colleges. He can also monitor various types of activity on the website:



- Will can manage college, staff, and student accounts with admin privileges:

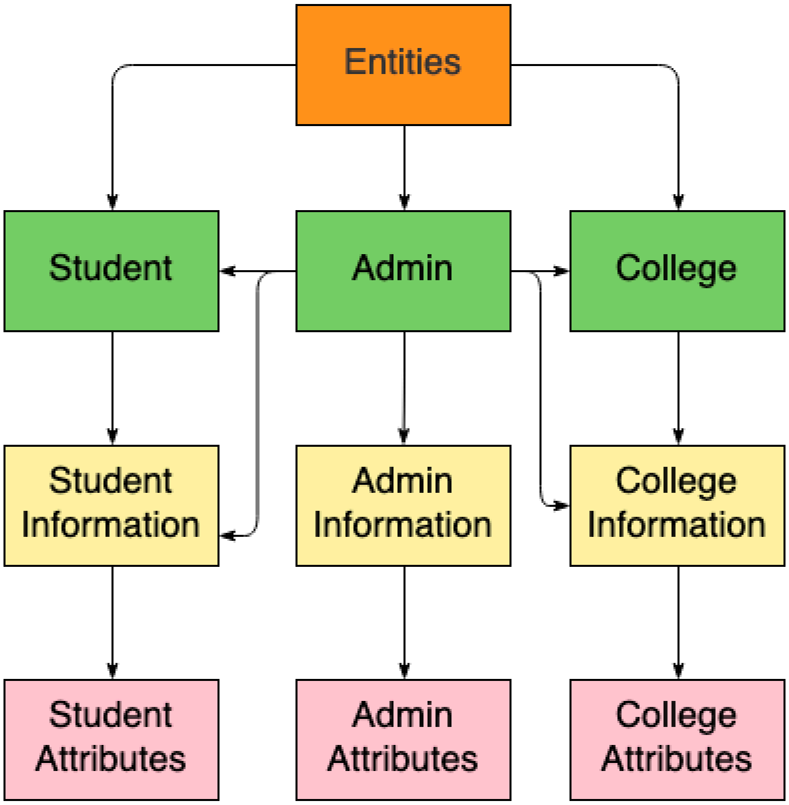


Summary

- Will can manage and moderate the website.
- Will can login as admin, which allows him to access and modify student, staff and college records.
- He can delete and modify users and schools as he needs to.
- He can change permissions and monitor activity on the website.
- He has full control and access to what happens on the website. He is also expected to maintain it and make sure things are running smoothly.

3. High Level Architecture, Database Organization

- **A very basic prototype of the database organization is pictured below:**



In the above prototype, we see that we must keep track of the users of the website and ensure all types of users have their own unique information and roles in the database. This part is shown in the orange box named Entity as we continue to branch out.

Firstly we need to recognize and differentiate the three users so that they cannot be confused or mixed with one another. We do this by creating three different branches known as Student, Admin, and College. This part is shown in green boxes.

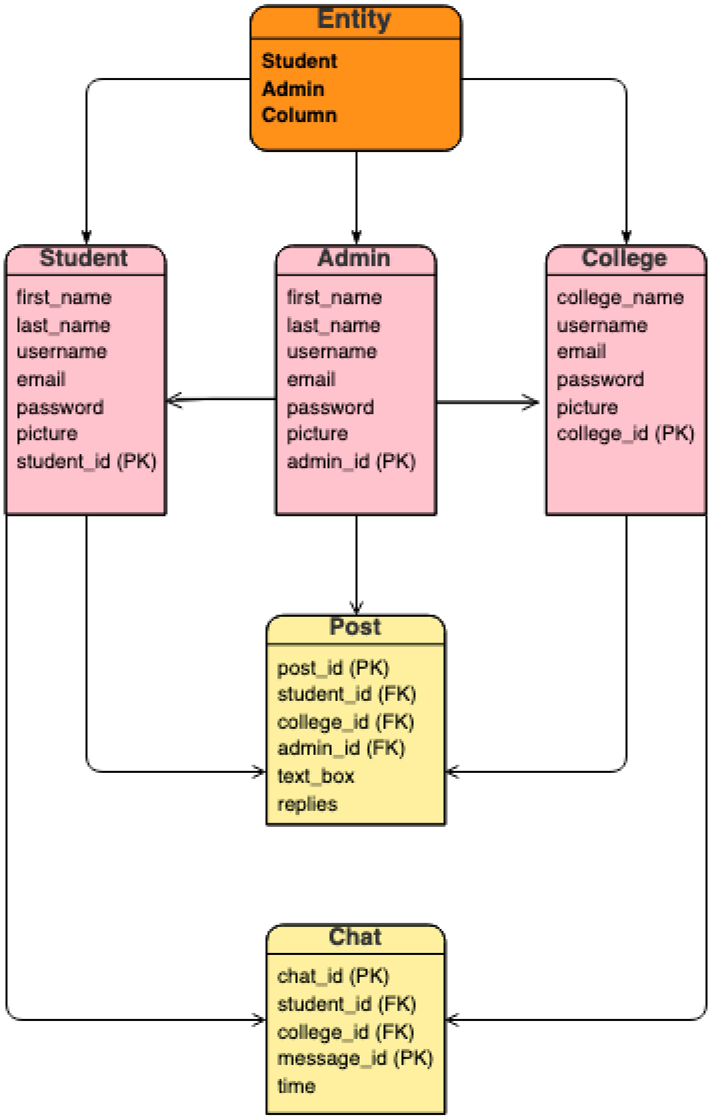
The Student entity needs to represent a student-athlete who has signed up on the website and wants to display their information to the possible college recruiters. We need to ensure that this user can in fact write and save their information on the website and be able to modify their changes.

The College entity represents colleges that are accepting and are recruiting student-athletes based on the sport that they are interested in. Similar to students, the colleges should also be able to write information about themselves and be able to modify this information.

The Admin entity is a person who has unique privileges for the website. The Admin should be able to see, modify, and maybe even delete the information posted by both the Student and College entities. The admin themselves should also be able to have a profile and post as a user as well. All of the above entities' information will be saved and kept in the database, as shown in the yellow boxes.

Lastly, in order to properly and safely store this information, we use attributes that are made specifically with the purpose of keeping the data safe and stored in memory. This part of the prototype is shown in the pink boxes.

- **More specific, and actual database organization:**



Student Entity Table:

Name	Data Type	Description
first_name	String	First name of the student
last_name	String	Last name of the student
username	String	Username of the student
email	String	Email of the student
password	String	Password for the student
picture	String	Profile picture (URL)
student_id (PK)	Integer	ID assigned to student (Primary Key)

Admin Entity Table:

Name	Data Type	Description
first_name	String	First name of the admin
last_name	String	Last name of the admin
username	String	Username of the admin
email	String	Email of the admin
password	String	Password for the admin
picture	String	Profile picture (URL)
admin_id (PK)	Integer	ID assigned to admin (Primary Key)

College Entity Table:

Name	Data Type	Description
college_name	String	First name of the college
username	String	Username of the college
email	String	Email of the college
password	String	Password for the college
picture	String	Profile picture (URL)

college_id (PK)	Integer	ID assigned to college (Primary Key)

Post Table:

Name	Data Type	Description
Post_id (PK)	Integer	ID assigned to the post (Primary Key)
student_id (FK)	Integer	ID of the student involving the post (Foreign Key)
college_id (FK)	Integer	ID of the college involving the post (Foreign Key)
admin_id (FK)	Integer	ID of the admin (Foreign Key)
text_box	String	Text box where the post is typed
replies	String	Replies made to the post

Chat Table:

Name	Data Type	Description
chat_id (PK)	Integer	ID assigned to the chat (Primary Key)
student_id (FK)	Integer	ID of the student involving the chat (Foreign Key)
college_id (FK)	Integer	ID of the college involving the chat (Foreign Key)
message_id (PK)	Integer	ID of the message (Primary Key)
time	Timestamp	Timestamp of when a message is sent

- **Media Storage:**

  We will keep any images, videos, or audios in the file system in order to keep it simple and easy to use. Additionally to the three listed media types, we will also be using Google Maps, and analytics as well. All mentioned types will be stored in the file system.

- **Search architecture and implementation:**

  Given the format of our database, the algorithm that will be implemented for the search function is to go through the entries in our database, based on the specific item that is being searched. For example, if the user is interested in searching for a specific college, they have the option of searching through either names of colleges, the state they reside in, the year they were founded, or the sport that they offer. This way they can search for and see a various list of colleges that offer what they are looking for.

  The exact process of this search is based on what entry the user chooses to use. If they choose to search for the names, then we will go to the database and iterate through the data we have under our College entity that refers to the name of the college until we see a match. The same exact algorithm is used for the rest of the offered ways in which the user can look for a college.

- **Our Own API's:**

  We will create our own API's for various different cases. For example, we plan to create our own API's for functions such as:

  - User Authentication
  - Olympic participant entry for staff working on student entries
  - Searching
  - Messaging

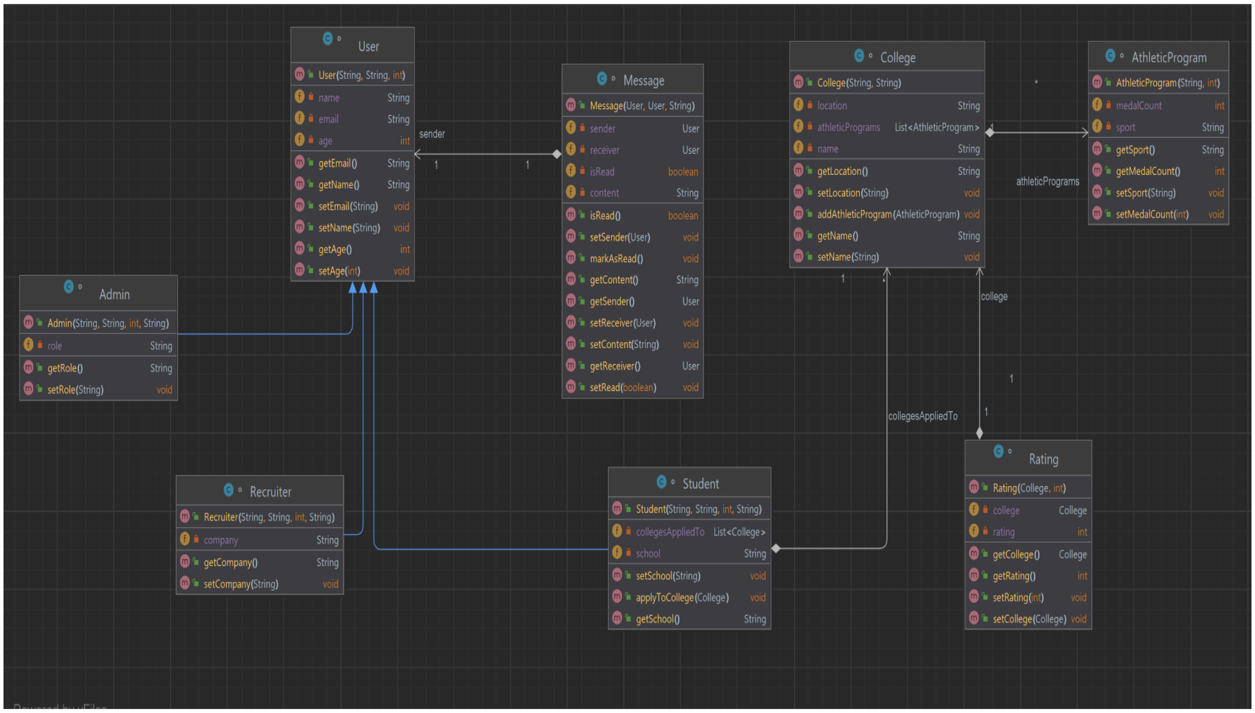
- **Significant non-trivial algorithms:**

  We plan to provide a rating for colleges that are looking for athletes. For example, a college could be rated 2 out of 5, which would entail that this college is probably not one of the most sought-after options. For this function, we will either take the score from online sources such as Google, or have previous students/alumni rank their experience and the effectiveness of the program in their university.

  Additionally, we also intend to recommend colleges that best suit a student's goals/checkbox. This could be done by trying to find a match between the requirements/extracurriculars a college is looking for and with what the student-athletes have listed on their profiles.

4. High Level UML Diagrams

This kind of diagram shows classes, their properties, methods, and the connections between them in order to illustrate a system's structure. It's a standard tool in object-oriented system design as well as in software engineering.



5. Identify Actual Key Risks for Your Project at This Time

- **Skill risks**
    - **Do we have the right skills?**
        - When delegating roles, we make sure to communicate our skill strengths and weaknesses. This provides strong structure in terms of teamwork.
- **Schedule risks**
    - **Are we able to get things done on time?**
        - Due to us being piled with work/assignments from other classes and our personal lives, it is hard to be on top of things all the time. To address this risk, our team will use team communication to check in with each other to get our parts done and reminding each other of due dates/times.
- **Technical risks**
    - **Is what we have set up working properly? Does it work without errors?**
        - This ties to team communication, but making sure backend and frontend teams (everyone) are working together. This ensures that we’re making not only progress in our project, but also what we have so far is working properly without errors.
- **Legal/Content risks**
    - **Are we able to obtain content we need (legally) with proper licensing and copyright?**
        - Making sure we have rights to any images we use for the project. Again, staying on top of communication, we can ask our teammates if we are unsure or let each other know.

6. **Project Management**

Our team has been doing well managing the tasks of this project. Our clear communication and time management is a strength that has been really benefiting our team. At the moment, we are using Miro for our tasks. Miro offers a visual interface where tasks are organized into boards with customizable sections. It first started as a dashboard to view all of our front end tasks but we later implemented it for our back-end and other full stack tasks as well so that all of us can use it for recording our tasks and be clear on what tasks are at hand. Another important aspect of product management is splitting up the tasks.

During the beginning of each milestone, either during our class meeting or our weekly meeting, the milestone is broken down into seven parts assigned to each team member according to their role and strengths. Then, the team lead records everything into our Discord and Miro so that everyone knows what their part is at all times. Our team meets twice a week to ensure we are all on the same page and to address any issues we may have with our parts of the project. In addition, we also are very active on Discord and ask any questions necessary when needed. Doing all these things ensures efficient project management and ensures that all tasks are getting done on time and in a resource-efficient manner.