Maddox Scott

Ms. Sakthikumar

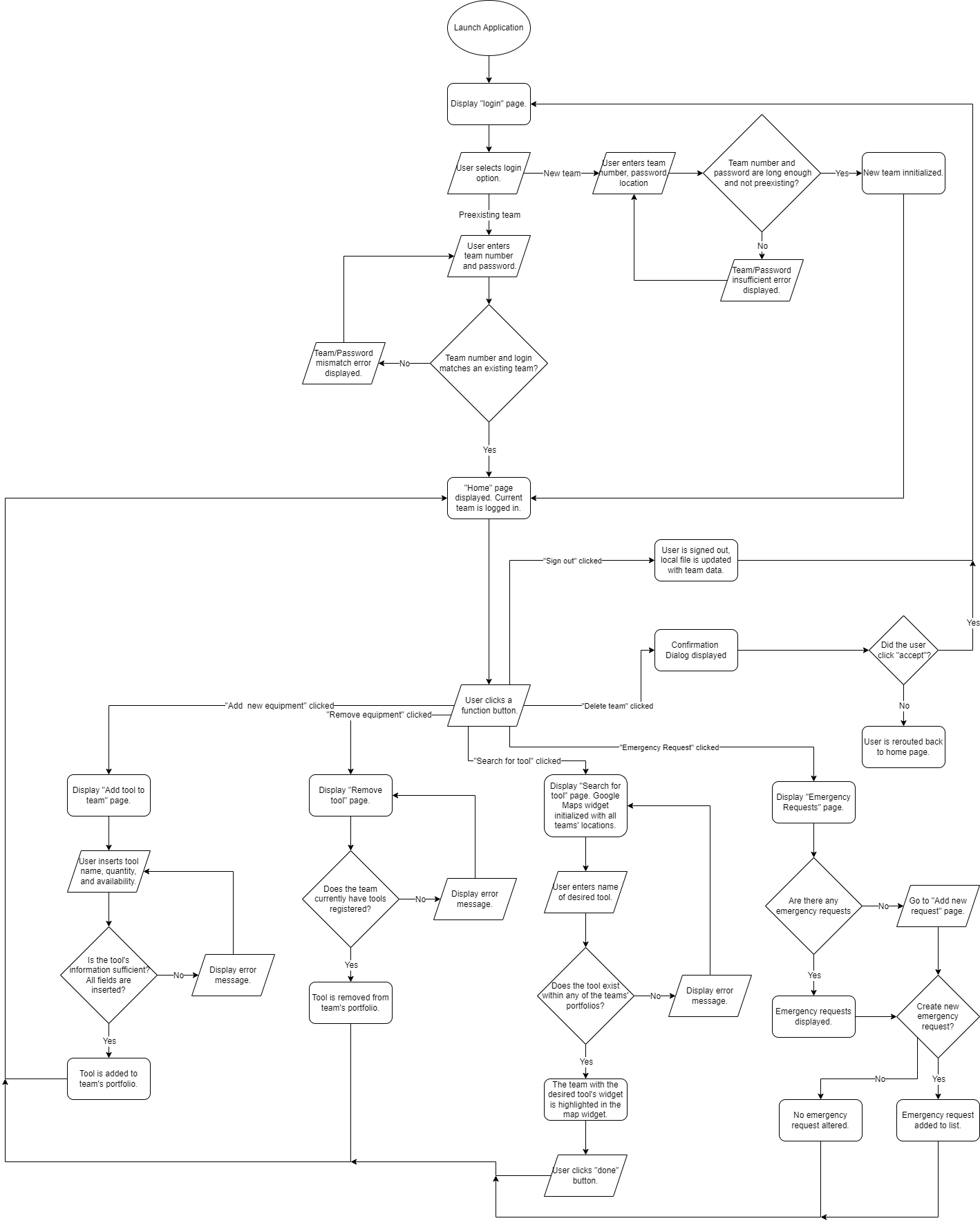
IB Computer Science, Period 5

20 March 2023

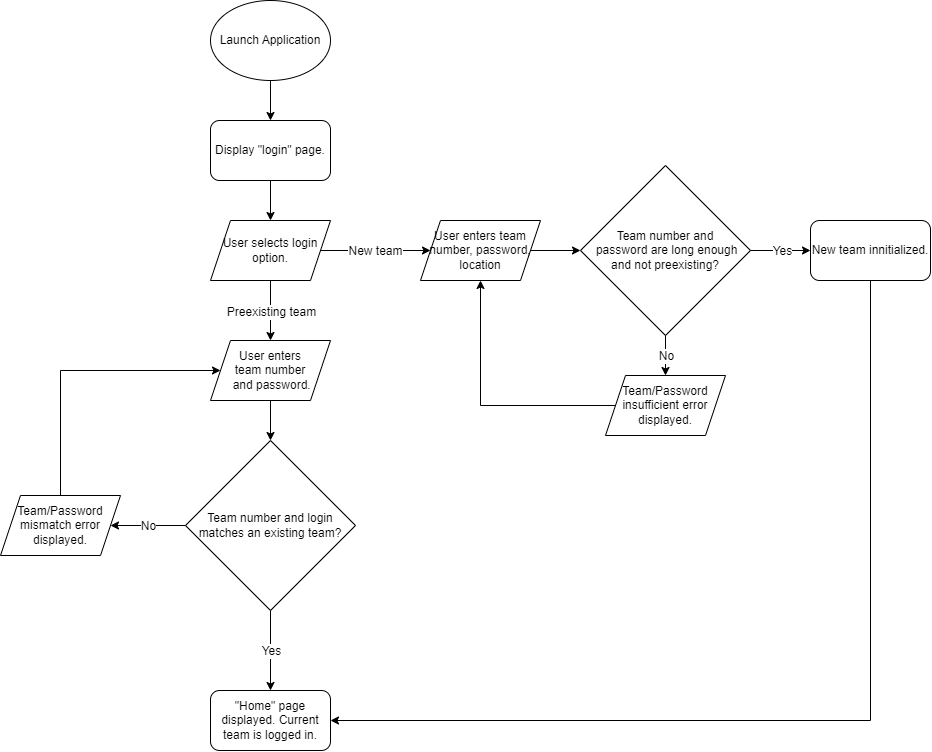
**Criterion B: Design**

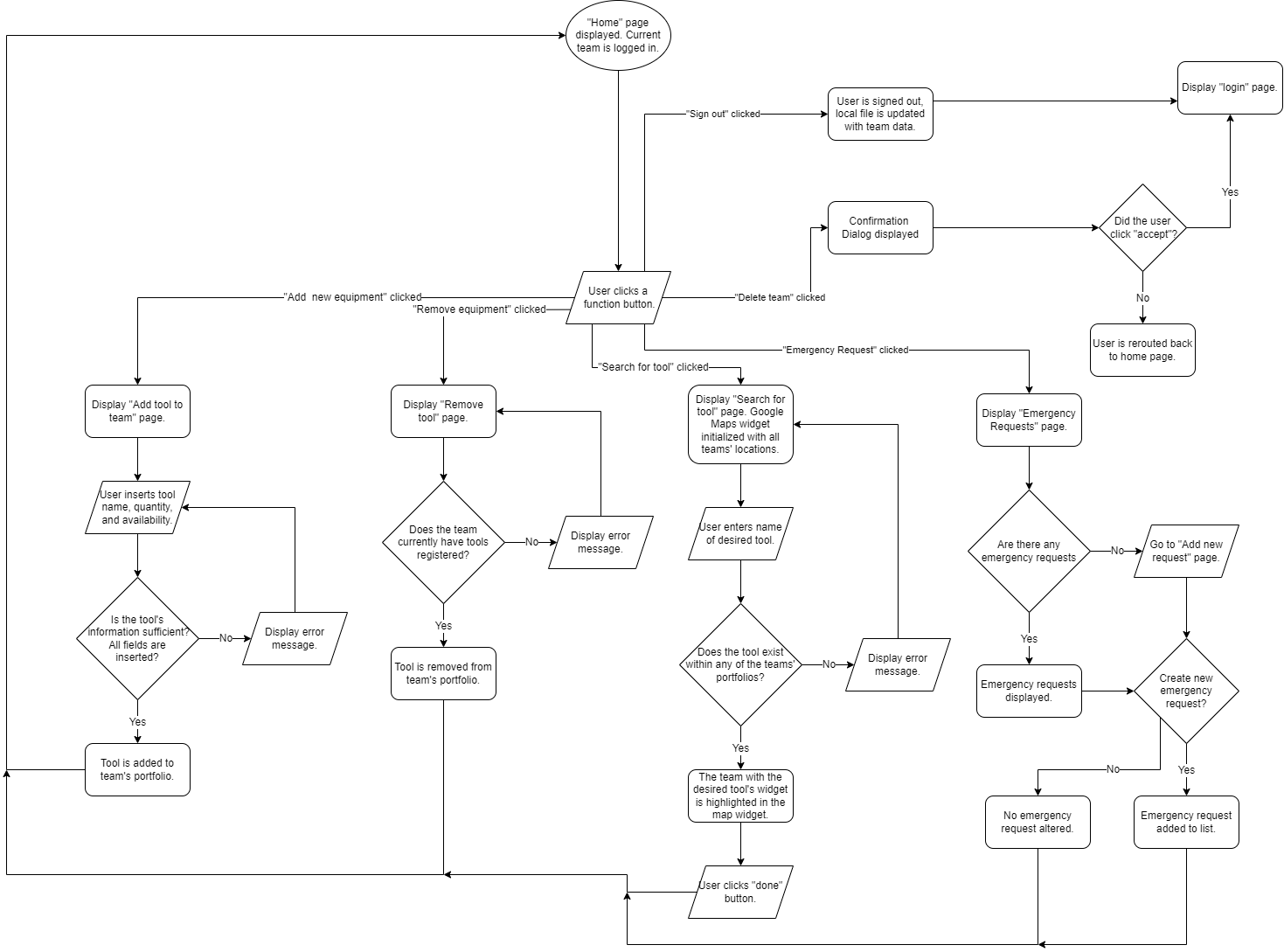
**Application Flowchart**

Comprehensive Flowchart: See Appendix (D.1) for the initial application flowchart.



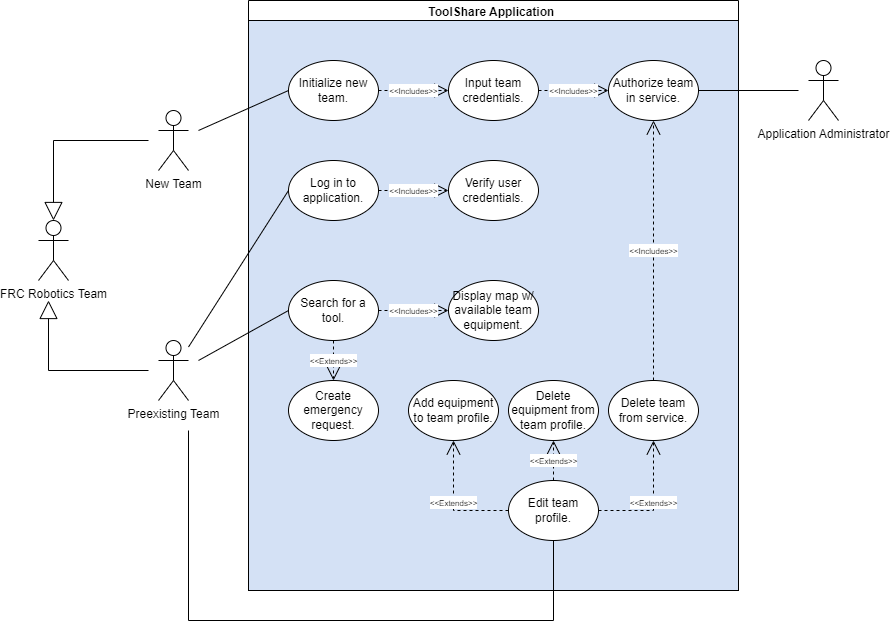
**Algorithms**

User Authentication Algorithm: Upon booting the application, the user is presented with a login screen where they provide their team’s number and password through two text fields. If the user enters a number and password which do not correlate to an existing team, an error message is displayed. This process repeats itself until either a valid team number and password have been entered, or the user clicks the “new team” button, routing them to a page for instantiating a new user into the application. After the user logs in or creates a new team successfully, they are finally routed to the home page.

HomePage-Routing Algorithm: See Appendix (D.1) for initial iteration. After the user has been authenticated, the home page routing algorithm is responsible for displaying the correct pages, graphics, and interactives based on user input. The application’s homepage consists of six elevated buttons, each sending the user to different pages of the application. The composition of several pages, such as the “Delete Tool” page, depends on whether or not the logged in team has initialized any tools. The algorithm passes a conditional to see whether or not tools exist, and if they do not, a simple error message is displayed instead of the page itself. After the user has finished using each page, they are routed back to the home page to begin the algorithm again. The algorithm terminates when the user either chooses to delete their team or sign out.

**Use Case Diagram**

Preexisting teams begin their interaction with the application by logging into their accounts which were already instantiated. New teams are capable of interacting with the application to the same level as existing teams, except they must first create a new team in the application to do so. As the administrator of the application, I am capable of adding, altering, or deleting teams from the application by manipulating the file which information is stored within.



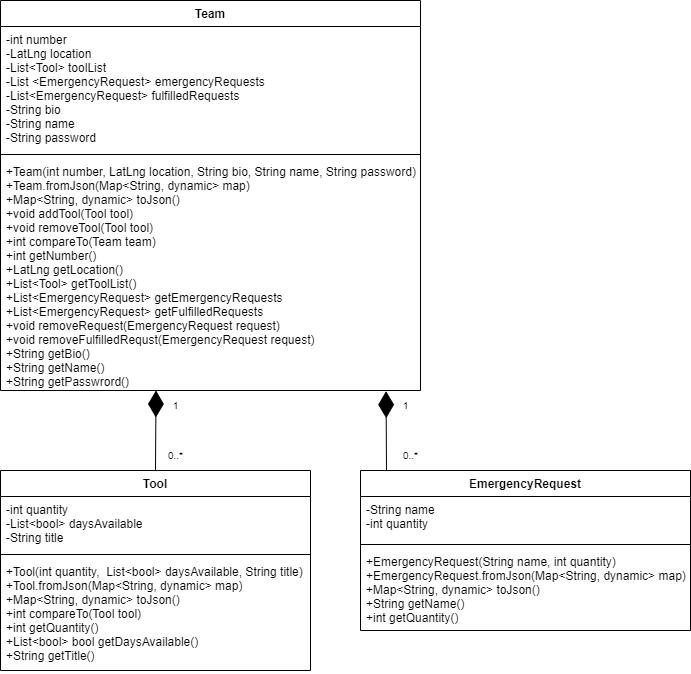
**Data Structures**

A hashmap data structure is used for storing and accessing each instance of the “Team” class. Every team’s number is used to populate the key values of the hashmap, and each key corresponds to the team’s “Team” class instance. Hashmaps do not allow for duplicate key values, which is beneficial for depicting the FRC numbering structure where every team has their own unique number. Additionally, a team’s information can be looked up without needing to iterate through the hashmap by utilizing the team’s number as the hashmap’s key, providing a constant Big O time complexity of O(1).

Each instance of the Team class possesses an ArrayList of Tools and an ArrayList of EmergencyRequests. Unlike regular arrays which have fixed sizes, ArrayLists are able to grow and retract as tools are added and removed from the list.

A stack data structure is utilized for keeping track of what pages the user visited in what order. When the user visits a new page, it is pushed onto the stack, and when the user clicks the page’s back button, the top page of the stack is popped off, and the application returns to the page preceding it.

**Objects:** See Appendix (D.3) for earlier iterations of the classes’ UML diagram. Originally, emergency requests did not possess their own class, and were stored as an ArrayList of strings.



| Class | Description |
| --- | --- |
| Team | * Depicts an individual FRC robotics team, possessing any number of tools to lend to other teams. Each Team possesses a number integer and password String for login authentication. Both of these variables are private and final to prevent any outside tampering. Each Team also possesses a LatLng coordinate representing their geographic location. * The method “toJson()” and the constructor “fromJson()” allow for Team instances to be translated to and from a text format used for saving data locally in a nonvolatile txt file. * Every team possesses three ArrayLists: one for the team’s Tools, one for the team's unfulfilled emergency requests, and one for the fulfilled requests. * Every variable within the class is private to encapsulate everything belonging to the class in one location, unable to be tampered with from outside sources. |
| Tool | * Represents a unique tool, equipment, or asset owned by an FRC robotics team. Each Tool possesses a corresponding integer quantity, string name of the tool, and an array of the days which the tool can be borrowed. * “daysAvailable” is an array with seven elements, each of which is a boolean value indicating whether or not the tool can be borrowed on that day. The array’s index represents the day of the week, with i = 0 being Sunday, and i = 6 being Saturday. * Just like the Team class, the Tool class possesses a method for converting itself into a Json string format, and a constructor for creating a new Tool instance from a Json string. * The method “compareTo()” is used for sorting tools alphabetically by their titles. |
| EmergencyRequest | * Contains only a String for the name of the requested tool and integer quantity of tools needed. |

**UI Flowcharts**: See Appendix (C.1) earlier iterations and client feedback which drove changes.



**Test Plan**

| Success Criteria | Test Plan |
| --- | --- |
| 1 | **Plan:**  Initialize three teams into the application: Team 123, team 456, and team 789. Each team’s location will be somewhere in the Seattle area. Afterwards, delete team 789 from the application.  **Positive Outcome:**  Upon loading the map, all three teams will be displayed within one map, each with their respective locations. After deleting team 789 and reloading the map, team 789 will not be present.  **Negative Outcome:**  Less than three teams appear when first loading the app, or team 789 remains within the map even after deleting the team. |
| 2 | **Plan:**  Team 123 will add a tool named “drill” to their account, with a quantity of three and may only be borrowed on the weekend. Then, the user will route to the “Delete Tool” page.  **Positive Outcome:**  On the “Delete Tool” page, a tool of type “drill” with a quantity of three is displayed and able to be deleted.  **Negative Outcome:**  The “Delete Tool” page reports that team 123 has not yet added any tools to their account, so there are no tools to be deleted. |
| 3 | **Plan:**  After team 123 has added three drills to their account, team 123 will sign out and team 456 will sign in and navigate to the “Search for Tool” page and type in the word “drill.”  **Positive Outcome:**  The Google Map will refocus onto the location of team 123 and a dialog box will appear stating that they possess three drills which can be borrowed on weekends.  **Negative Outcome:**  The dialog box will state that no team possesses any drills, and will offer to submit an emergency request. |
| 4 | **Plan:**  Team 456 will route to “Emergency Requests” page and submit a new request for five wrenches. Afterwards, team 123 will log back in and attempt to fulfill the request. Lastly, team 456 will log back into the application.  **Positive Outcome:**  Upon logging back into the application, team 456 will be greeted with a message saying that their emergency request has been fulfilled. The emergency request for wrenches will no longer appear within the list of emergency requests.  **Negative Outcome:**  Team 456 will not be greeted with a message notifying them of their fulfilled request, and their emergency request will still appear in the list of unfulfilled emergency requests. |
| 5 | **Plan:**  After team 123 has been initialized into the app, the mobile device emulator will be restarted, thus terminating the application. Upon reloading the application, the user is presented with the “Login” page, and will attempt to sign in using the number 123 and password “pass,” which was used when creating team 123.  **Positive Outcome:**  Team 123 will successfully login, proving that their login credentials persisted between application runtimes.  **Negative Outcome:**  The application will fail to login and will report that there is no team within the application with the number 123. |
| 6 | **Plan:**  Signed in as team 789, the user will click the “Delete Team” button present within the application. After being sent back to the login page, the user will attempt to sign in with team 789’s credentials.  **Positive Outcome:**  The application will fail to sign in, as team 456 was successfully deleted and no longer exists within the application.  **Negative Outcome:**  The user successfully logs in, indicating that team 456 was never deleted. |
| 7 | **Plan:**  Search all pages of the application for button sizes less than 36px in height or 50 px in width.  **Positive Outcome:**  All buttons and text fields are larger than the parameters above. High contrast distinguishes them from their backgrounds. Simple animations mark each element’s appearance.  **Negative Outcome:**  There is illegible text, small buttons, or a lack of animation within any page of the application. |