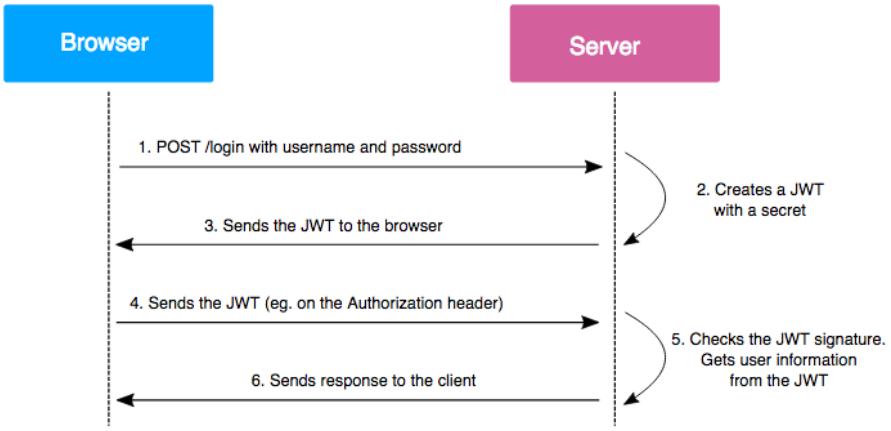
Security Analysis

Authentication of RESTFUL API Services

The plan we have to achieve this is shown in the diagram below. The server stores a token which is created whenever a user logs in. The user's browser can then store the token as a cookie, which will be used to authenticate access to the API services. This source has been quite useful in figuring out how to do this securely.



Each token contains the expiration date, which will be useful for automatically logging users out. This will also be used to keep users logged for a specific amount of time if they choose the "Stay logged in" option on the login page.

Currently authentication on our system still needs a lot of work before it is actually secure. We generate and store the tokens to keep track of logins, but currently, they are not being used to authenticate API requests properly. This is an area for improvement. We plan to use Jersey's role-based authentication since we have to distinguish between the privileges of the client, crew, and admin to restrict API access.

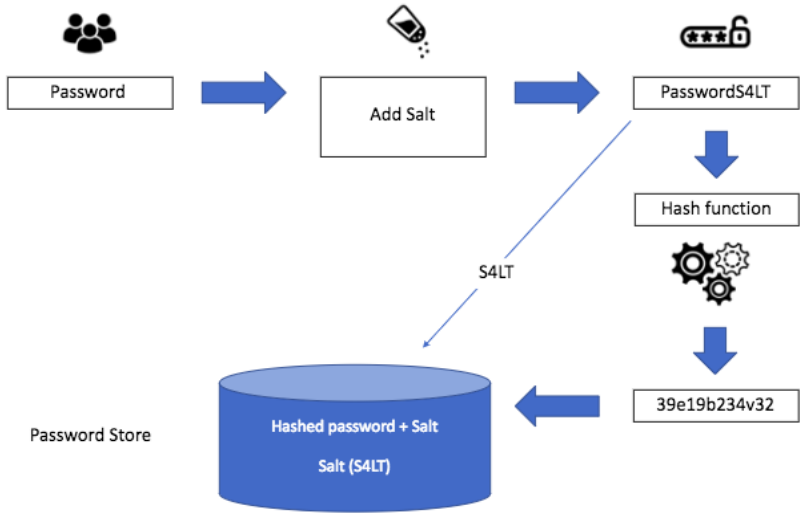
Cookies and user sessions

Currently, the application stores the account ID of the client as well as the authentication token. Because of this, there is a risk that an attacker could steal these cookies using injected scripts. This means that to keep the cookies secure, should be sanitizing our inputs and using prepared statements. In this regard, there is a lot of space of improvement since we haven't always been very careful in sanitizing our inputs and taking the proper precautions. It will be necessary to go over existing code to fix possible vulnerabilities and become stricter in the way we implement our features.

One insecure part of our application we discovered was that when issuing the cookie using HTTPS, we didn't add the Secure flag to it. This is important because it instructs browsers to never send the cookie in plain HTTP requests, making it more difficult for attackers to steal cookies. In addition, these cookies are used to verify user sessions, meaning that the same token can't be used with a different account.

Passwords

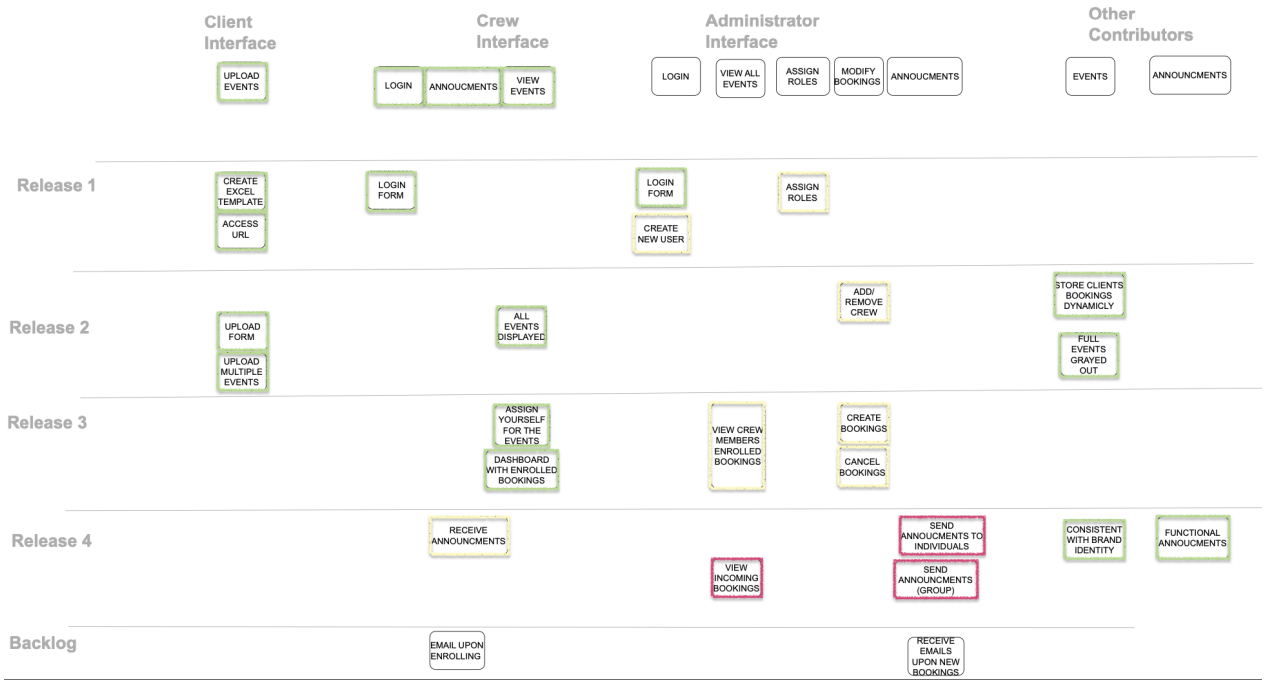
We planned to perform salting and hashing on all passwords to help secure the passwords and prevent rainbow table attacks. Currently, we have implemented the hashing of the passwords, and these hashes (which are stored on the database) are being used to authenticate user login. However, we still haven't implemented the automatic salting on the database side or server side. This means that our application is still vulnerable to rainbow table attacks, making it can area for improvement.



One option to further secure our application other than salting (which is coming soon) is peppering. However, given our time constraints we consider implementing this as a low priority, instead focusing on more function-critical parts of the application. Additionally, as long as we perform all the salting and hashing securely it should already be secure enough. If we have time we will add it.

Software Testing

User Stories



The table above shows is what we have done after the 3rd sprint review. We are mostly focusing on the crew member side first after that go through the administrator side. That is because there are some overlapping functions between the crew and administrator.

According to the user stories, we are done with the interface for clients. Which client can upload a single event and multiple events based on the requirements. But sometimes, the URL is not working somehow, so we will focus on this problem to fix it as soon as possible. And also we need to consider UI design for clients that can easily recognize and access the website without explanation.

For the crew interface, we mostly finished the interfaces and functions but we still need to fix some problems like viewing the different sizes of the window. We figured out that problems are based on the margin and padding in CSS. That is because the descriptions are not the same length for every event. Except for those little problems the crew side mostly completed.

With administrator interface, except login interface, we are still working on the administrator interfaces and connect with back end and front end. For the administrator interface we will finish them in the upcoming week we are working on them. After that we will have some time to test them to make sure all requirements are functioning. It will take less than the crew side, that is because we already had experience with front/back end for implementing that we want to do for web application.

For the other contributors, those things are more related to the web design based on the requirements. We are done with those features when we implement the crew member side for web application.