Security Tests

Authorization

In order to test authorization, we wrote unit test that test the access of different controller's function by different users. We test these accesses by logging in as these different user roles and ensure that when an unauthorized user attempts to access pages they are denied access to that they are redirected to the main page. As these tests are included in the unit tests, they are run with every build and are passing.

Avoiding SQL Injection

Through the usage of cakephp's built-in methods to access and modify our databases, we effectively avoid SQL injection from occurring. Cakephp's method handles SQL escaping which can occur if you write raw SQL statements to handle your database. No issues were revealed during testing.

Handling Malformed Input

Our system handles any malformed input by using validations to ensure that the inserted values are valid and meet the requirements. Also, we avoid malformed input by constraining the type of the input that they can insert for certain inputs. Furthermore, if any invalid input is attempted, a message is shown displaying what the error was and the system does not process anything until the inputs are all correct. Many unit tests were created to test any invalid inputs. As with the authorization testing, the inclusion of these tests in the unit tests means they are run for every build and are passing.

Password Security

In order to ensure that the user's password are secure, we hashed these password into the database. Thus, these password will only be known by the user. No issues were revealed during testing.

Usability

Sign Up/Login

In early stages of testing, the previous Captcha proved problematic for users. Some reported trouble with the system accept the correct response, leading to the adoption of the current Captcha. The new version is both easier to use and clearer, while still maintaining the necessary functionality. The only reported bug still outstanding regarded a "password reset loop" while attempting to login. It was reported once, and since then neither the users nor the developers have succeeded in reproducing it. This suggests it was a complication the originated outside of the system, and the customer has indicated they are not worried about it.

Reservation

A later stage of testing revealed a design oversight present in the requirements document; one-way or multi-leg trips were not possible. This was revealed during more focused testing recently by JAUNT's "power users" and was subsequently corrected quickly. While handling multi-leg trips is still complicated, as it requires multiple one-way trips to be made for each leg, both the users and JAUNT are satisfied with the result.

Timeoff

Overall the system has been well received. While the reordering of time-off preferences, in the "Make a Time Off Request" page, is not always initially intuitive, the users have had good reactions to the system ultimately.

Installation

The test plan has been tested with an Amazon EC2 instance in addition to the later steps done on the pegasus server.

Requirements

A comparison of the requirements document and the final system shows that all listed requirements have been completed. The customer has stated they are satisfied with how the system meets their needs and are in the process of introducing it fully both internally and to their clients.

Compatibility

The system functions on every tested browser.