APPSO Portal Monitoring and Control Metrics from Sprint 2

Key measurables and metrics:

Efficiency: Minimal Waste; Maximum Throughput

Compliance: Degree to which a program starts **providing value** to the customers

Correctness: Degree to which a program operates according to specification

Integrity: Degree to which a program is **impervious** to outside attack

Usability: Degree to which a program is easy to use by the intended audience

Maintainability: The degree to which a program is amendable to change

Team Objectives for maintaining such metrics:

Efficiency: Velocity of 75 story points per sprint

Why 75 story points? This is the sprint where we need to do the most work. Members collectively agree on this reasoning.

Compliance: 100% of the **most valuable** user stories deployed

Correctness: 0 errors **escaped** to the user

Integrity: 0 security breaches

Usability: Intended user learns how to use the system in 15 minutes or less

Maintainability: Mean Time to Repair (MTTR) - from assignment of ticket to deployment of fix – of 5 hours.

TEAM OBJECTIVE RESULTS (to be filled out throughout the sprint 2):

Efficiency:

On the last day of Sprint 2 (10/17), the number of story points we have completed this sprint is: **113 story points!** Our team did an amazing job focusing on the critical user stories needed to have a fully working website that the customer can use. The sprint burndown chart below shows the progress we made over time.



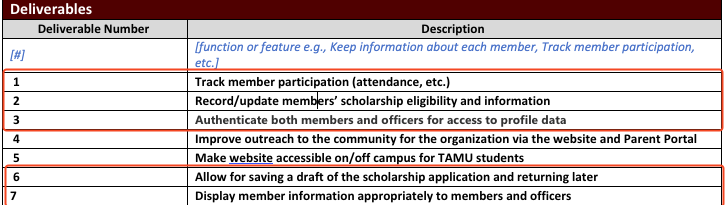
Note: we marked off several user stories at a time, especially during/after SCRUM meetings.

We did have some left over user stories we haven’t completed, but these stories were not deemed essential or critical, when looking at the project scope. This will be completed in sprint 3, especially since these tasks are more front-end oriented.

Compliance:

Most Valuable User Stories needed to complete:

From the project scope:



These are stories we need to complete:

“As an APPSO officer, I need to be able to check off if a member attended a event/meeting, in order to keep track of member attendance.”

“As an APPSO president/admin, I need to be able approve/deny new user registrations so that I can verify member identity and roles.”

“As an APPSO member, I need to be able to log into my member account so that I can view my participation data and scholarship application.”

“As an APPSO Member, I need to be able to log into my application without worrying about errors or bugs preventing login, in order to create a bug-free and seamless login experience.”

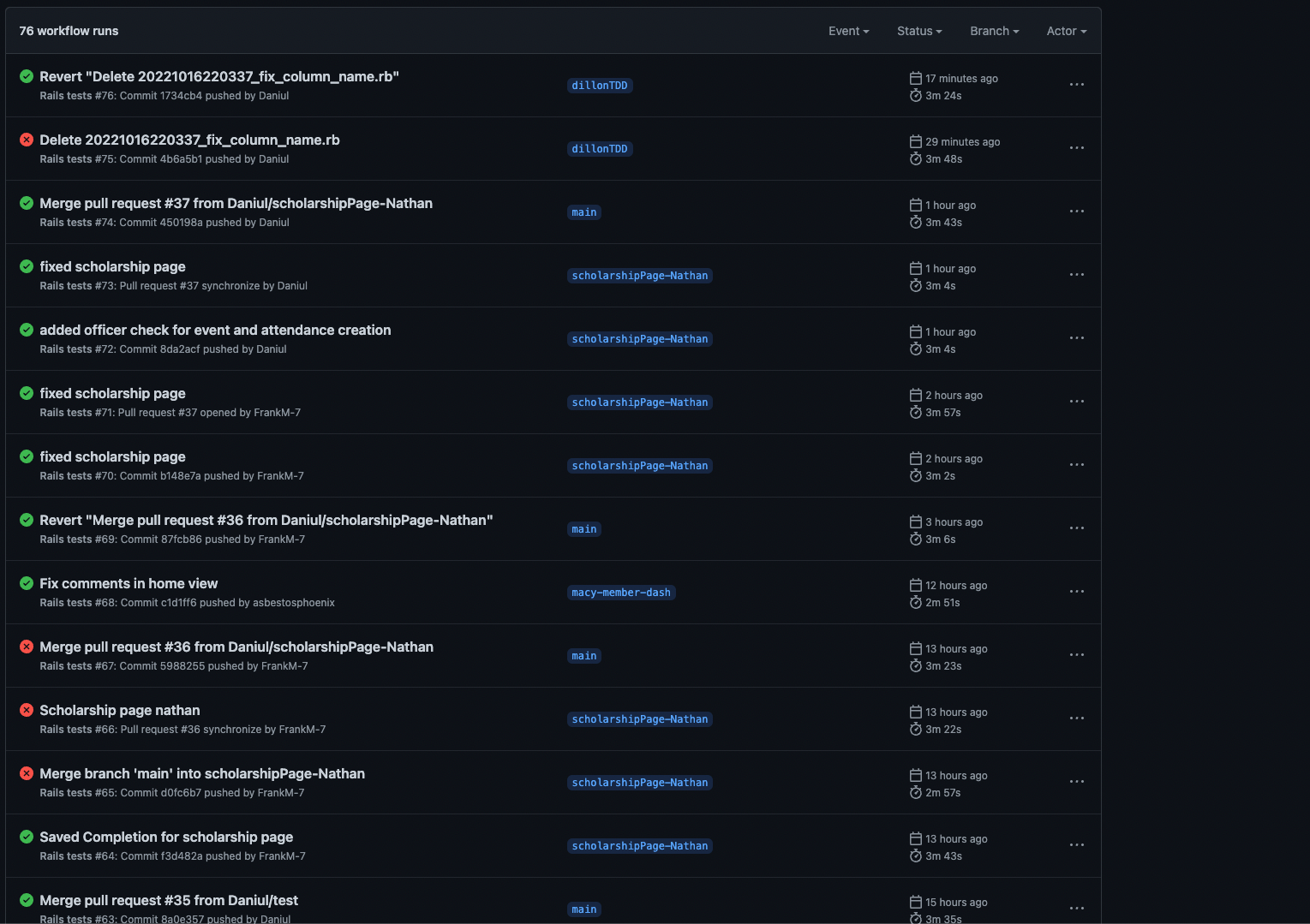
“As a new APPSO member, I need to be able request to be an admin, or officer and go through signup process.”

“As an APPSO user, I need to be able to log-in to Parent Portal securely, in order to keep my username and password from being exposed.”

“As an APPSO member, I'd like to be able to see my personal participation stats tracked in an easy-to-read format so that I can know how to improve my standing.”

**Green represents the most valuable stories we have completed.** The story we have not completed is in relation to Displaying membership information in one concise dashboard. We do currently display basic information such as email in the index, but we believe we did not satisfy the user story of presenting a member’s personal participation stats, or any other quick information in a “profile”. We plan on **designing** a more sophisticated design in sprint 3, which should be designated for UI/UX design focus. **So, we did NOT achieve 100% of our most valuable user stories deployed, but we are still proud that we completed basically most of the user stories.**

Correctness:



We did have sometimes during **development** that errors would pop up to the user, but by the end of sprint 2, we have managed to fix most of those errors (especially ones regarding approving/rejecting new users, as Thierry had trouble with debugging those issues without having a vulnerability issue).

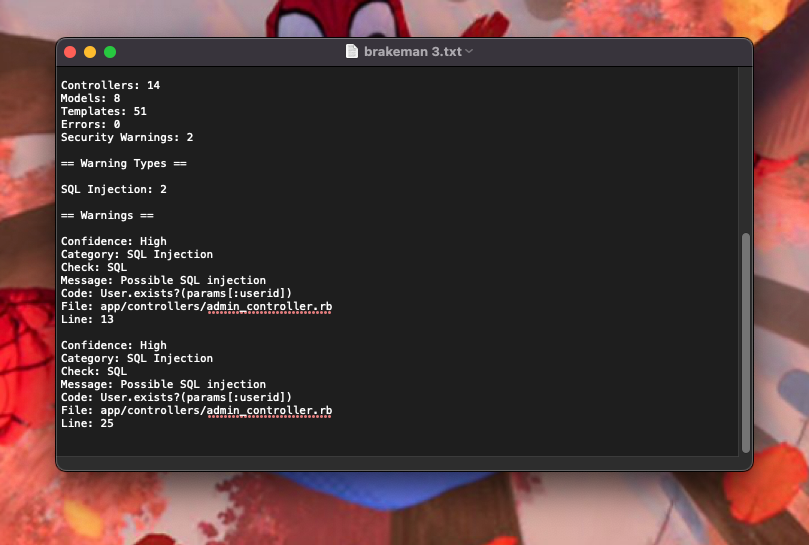
<https://github.com/Daniul/APPSO-Parent-Portal/pull/32>

This pull request is a **great example** of how a user story/feature was tested against the acceptance criteria, and the Definition of Done. Thierry noticed that only there were errors when user clicked certain buttons, there was a critical SQL injection vulnerability, which he managed to fix after getting the warning from Brakeman. **Utilizing the Acceptance Criteria and the Definition of Done allowed us to ensure that the program operates according to specification**.

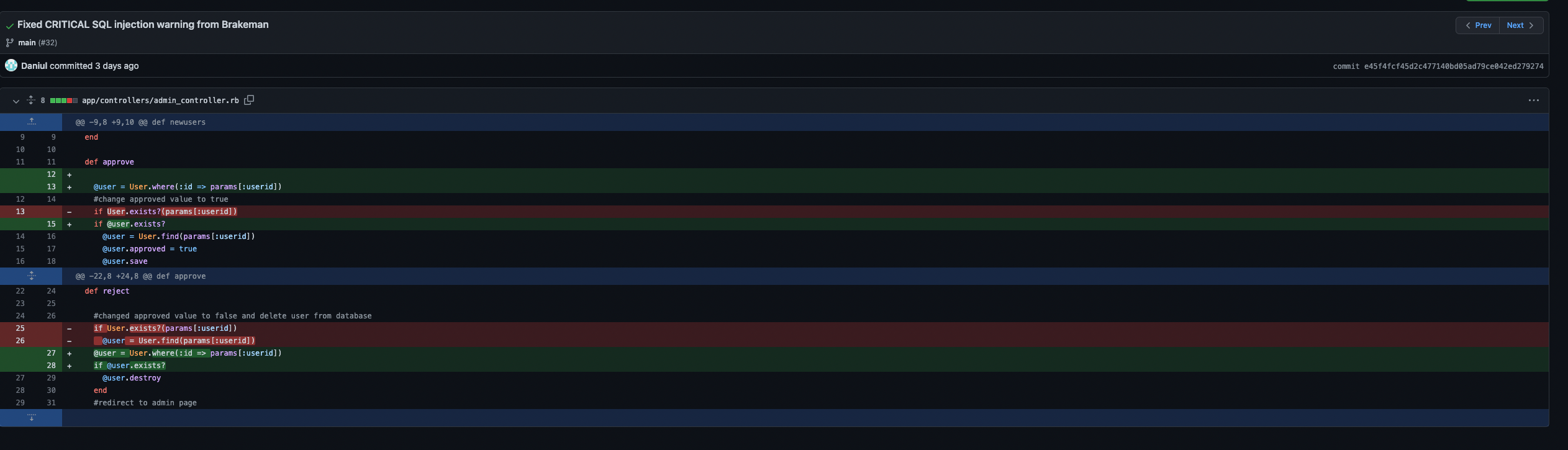
**In conclusion, as of the last day of the sprint, we do have zero errors (seen so far) escaped to the user**.

Integrity:

During development there were a few security breaches that appeared, especially when marking a user present or absent.

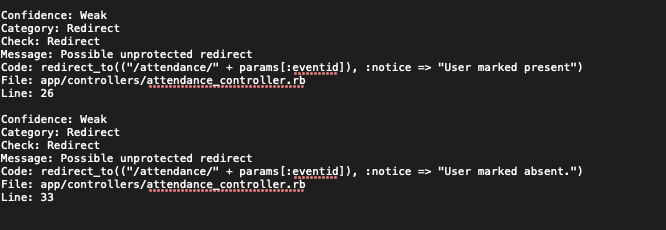


We had possible SQL injection vulnerabilities after implementing admin approval/rejection of new user registrations. The way we fixed this was to ensure that the params were converted and filtered into the right datatype when calling a equivalent SQL command.



10/16/22:

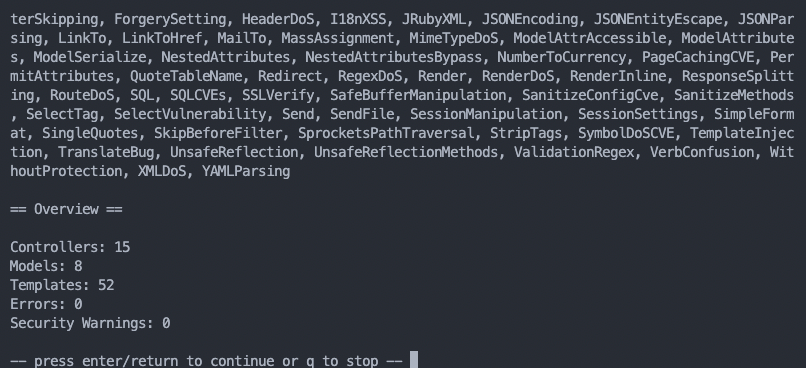
There was a possible unprotected redirect warning from brakeman, when implementing alerts to show officer a user was marked present/absent.



The way Thierry fixed this was to change the redirect to path, utilizing a bult in rails verbiage, to refresh the website and show the warning, without having to rely on params.



So as of the end of this sprint, we have **ZERO** security vulnerabilities



It’s great that we managed to have zero vulnerabilities at the end of this sprint. We will definitely be looking and monitoring brakeman warnings as they come.

Usability: Intended user learns how to use the system in 15 minutes or less

10/16: Thierry asked roommate to interact with the website.

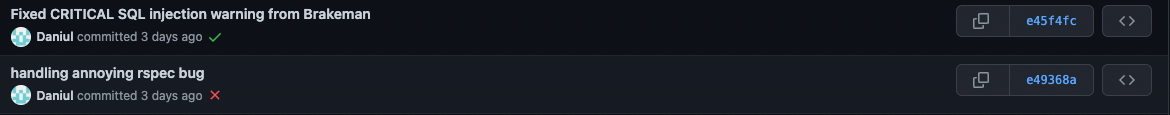
Thierry: I gave basic information on what the website was about, asked them to place themselves in the form of a parent/faculty, and try to access the website in the form of either an officer or a member. My roommate was also able to login with his tamu.edu, and ensured that no other gmail account can access the website. Even if they used the ordinary login/password flow, they confirmed they still had to be confirmed. I acted as an admin who approved his account. He told me although the website looks basic, he was able to use it quickly. I timed how long it took him to get acclimated to the system in **10 minutes**, whiich included me explaining some things **about** the purpose of the website.

This time metric of 15 minutes or less to get used to the system **will vary, especially with real parents and members of the organization**. Some users may not be as tech savvy as others, variety matters.

**But our test run with Thierry’s roommate confirmed to low time to acclimation.**

Maintainability: Mean Time to Repair (MTTR) - from assignment of ticket to deployment of fix – of 5 hours.

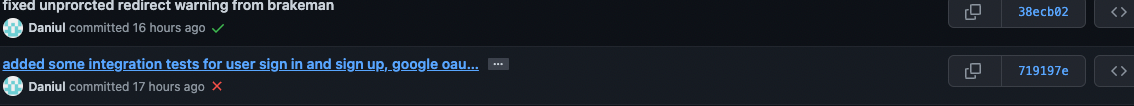
**BUG FIX EVENT #1: SQL Injection warning**



From Assignment of ticket to deployment of fix: ~1-2 hours.

Thierry: As mentioned earlier, I was the one who helped fix this error, I spent almost 1-2 hours receiving the error from my team, and fixing it.

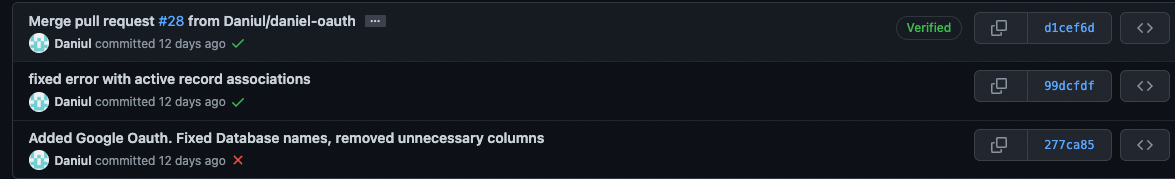
**EVENT #2: Redirect warning**



From Assignment of ticket to deployment of fix: ~1-2 hours.

Thierry: This was an annoying ticket to fix, since rails documentation of fixing this issue was vague, had to search across stack overflow and other websites to find solutions to this issue.

EVENT #3: Google OAuth not working correctly



From Assignment of ticket to deployment of fix: 1 hour

Thierry: Frank helped a lot in reviewing my pull request. My pull request had an error with Google OAuth while testing against Acceptance Criteria. I was able to fix it quickly with assistance from Frank. Here is proof where other members also checked just in case <https://github.com/Daniul/APPSO-Parent-Portal/pull/28> , since this was a big feature to implement.

The above are 3 events of MTTR. We had other errors /scenarios where we had a ticket to fix them, but they are denoted in our commits.

**SO MTTR was around 1-2 hours, I would say 1 hour usually since we were always on top of things when it comes to detecting unexpected errors.**