PAID - Participation Tracker

Team: PAID - Participation Tracker

CSCE 431 – Software Engineering

Contents

Recommended outline for your reflection:

-Describe in one sentence (or as few as possible) what the activity/method/topic was about. You can include another sentence about how you felt as you were going through the activity.
-Describe what you did right.
-Describe what you would do differently.

1 Executive Summary

This project was developed in CSCE 431 - Software Engineering. This project was intended to provide students with a realistic, industry-like experience and develop a deep understanding of software engineering methodologies. There was also a strong focus on team collaboration and communication within the team and with the client.

The goal of this project was to develop an automated participation tracking system where PAID members can sign into events held by their organization. PAID members can sign into events without needing an account and PAID administrators can access or modify all event and participation data collected by the site. The site also interfaces with the PAID-Member Points Tracker through an API. To achieve this goal, the site was developed in Ruby on Rails and we used other tools like GitHub, Heroku, RSpec, Simplecov, Jira and Rubocop. The final product was demonstrated to the client, was well received and met all the requirements PAID had for the product.

Our client was the Professional Association of Industrial Distribution (PAID) at Texas A&M University. The main point of contact for the team was Wesley Massey, the Data Analytics officer. They wanted an automated system through which they can keep track of members' attendance to PAID events and the points earned by each member. The latter portion of their need was handled through our aforementioned collaboration with the PAID-Member Points Tracker team.

The participation tracker developed by our team provides an interface where PAID members can see upcoming and ongoing events and sign-in/out/up for them using their Texas A&M email account. Site administrators can create events and assign them points. Virtually tracking event participation is much faster with this web-application as it removes all human elements which were previously present. The application is hosted on Heroku and has been transferred to the account of a PAID admin. The source code for the application is available to PAID at their request if they want to independently develop updates.

1.1 Need

The main motivation for our project was the needs of the PAID organization. PAID holds events for its members and keeps track of their attendance as well as assigns members points based on their events attended. Prior to the creation of our web-application, the organization used Google spreadsheets and had to manually input and transform data which is slow and creates opportunities for human error. By creating an online system with which PAID can track member attendance and that can interface with the PAID-Member Points Tracker team we removed the need for manual labor and freed up PAID resources to focus on other things for the foreseeable future.

1.2 Recommendations for Future Work

If in the future further work is performed on the participation tracker then the main focus

should be on improving the aesthetics and usability of the site or adding new features. The core functionality correctly performs the necessary actions and does not need to be modified. Certain upgrades could involve adding Javascript elements to the site or formatting the site documentation to increase visual appeal.

For project teams who may develop something similar in the future if multiple teams need to coordinate their applications it is imperative to communicate with your partner team often in order to avoid confusion and increase the productivity of both teams. It is also recommended to determine the documentation required at the beginning of the sprint to avoid any confusion or panic near the end of the sprint when a new document is found. From an end user perspective it is very important to consider how the site will be used. If most users will be on their phones then it is necessary to account for that and make sure the site functions correctly on mobile devices.

Key lessons the team learned included learning about Rails and the Model-View-Controller Architecture, understanding web development methodology, developing a deep understanding of Agile, understanding the importance of effective, timely communication and time management, and dealing with fluid requirements in a realistic situation.

There are no other issues for further investigation, however, it is worth noting that the Heroku CI/CD pipeline is not compatible with the Texas A&M GitHub. For future work it is worth transitioning to commercial GitHub if implementing CI/CD is a top priority.

2 Stakeholders

Client: Wesley Massey, Data Analytics at TAMU PAID
Advising Faculty: Pauline Wade

Scrum Master	Product Owner	Team Members
Lauren Ruffner	Neil Dwivedi	Neil Dwivedi
		Rory Gatson
		Jack Greene
		Roni Phillips
		Lauren Ruffner

The only special role besides Scrum Master and Product Owner that we had on our team was the Github/Heroku master who handled deploying to Heroku each sprint and also ensured that our code was successfully transferred to GitHub Classroom. This role was held by Jack Greene. All team members contributed to the project by coding and in certain, more difficult, situations we used pair programming.

2.1 Stakeholder Analysis Matrix and Communication Plan

A key part of the success of the project was the use of the stakeholder analysis matrix and communication plan. It was imperative to the team to have methods in place to determine our stakeholders and understand their needs as well as methods to communicate with them.

For the communication plan these methods involved the information required by the stakeholders, the format through which we would communicate and how often that communication would occur.

To communicate with the customer we updated them weekly about the status of the project through sprint review meetings. We always performed very well at sprint reviews since we communicated the product features effectively. One thing the team would have done differently is communicate with the customer more often than once a week and get more of their input for a more personalized site.

When communicating with the instructor we had to give status reports which we did through GitHub, Jira and sprint documentation deliverables. Communicating with the TAs and graders was the same as communicating with the professor, sprint deliverables were audited at the end of each sprint. The team did a great job keeping on top of all the deliverables we had because of strong organization. One thing the team could have done differently was starting some documentation prior to the sprint start to provide more structure to each sprint.

The project team communicated through a Discord channel and Microsoft Teams on a daily basis. The team did a great job communicating daily and there were no difficulties in contacting fellow team members. One thing the team could have done better is utilizing the capabilities of MS Teams to improve communication.

The stakeholder analysis matrix methods involved gauging the level of power and interest of each stakeholder, determining their expectations and preferences, and strategies to ensure a positive relationship.

The customer, Wesley, and the members of PAID were the most important stakeholders for the project with expectations that we would build a bug free system to track member participation. The team did an amazing job meeting all their requirements and impressed Wesley each sprint review. One thing the team could have improved on is communicating with Wesley more to get more feedback during development itself or maybe try to reach out to PAID members to get their input as well.

The professor and the TAs were stakeholders with low levels of power and interest since their main concerns were finishing the project on time and sprint deliverables. The team did well meeting daily during sprints for SCRUM meetings and keeping all necessary documentation up to date. One thing the team could have improved on is reading all the class announcements to ensure knowledge on all required deliverables.

The project team were also major stakeholders in the project with high levels of power and interest in the project since we built the application. Again, the team did a great job communicating via SCRUM and keeping on top of deliverables leading to a bug free product at the end of every sprint. Something the team could have done differently is coordinate pulling and pushing on GitHub better to avoid having merge conflicts.

See the Annex for our Stakeholder Analysis Matrix and Communication Plan.

2.2 Acknowledgements

- Neil Dwivedi
    - Product Owner
    - Worked with heavily with Active Admin and added mail server
    - Level headed and strong communication with team members
- Lauren Ruffner
    - Kept the team organized
    - Scrum Master
    - Security master, did majority of security testing
    - Group comedian
- Jack Greene
    - GitHub/Heroku master
    - Developed an API for and communicated with the PAID-Member Points Tracker
    - Pair programmed with other team members
    - Implemented sign-ins and sign-outs

- Rory Gatson
    - CSS Master
    - Performed quick fixes at 12 AM
    - Implemented sign-ins and sign-outs
- Roni Phillips
    - Created and maintained models including DB migrations
    - Developed organization methods to keep track of all documentation
    - Rubocop and Simplecov Master
    - Kept DB changes easy

Every team member did a great job contributing throughout the process. We all specialized in different things but contributed equally. I would like to commend every team member for their amazing work this semester. Without the whole team it would not have been possible to deliver a project of the quality we did.

3 Scrum & User Stories

3.1 Release Planning

Release Planning involved capturing requirements as user stories which is a common language for all stakeholders, including nontechnical customers, on the requirements of the application.

Jira was the primary tool used, and was helpful in keeping track of specific tasks and who was assigned to them. Jira also helped in relation to time tracking and keeping a backlog of all possible future user stories. At the beginning of each sprint we would spend at least an hour compiling tasks and user stories for the sprint. This allowed us to have a very organized sprint.

When it comes to release planning, we used Jira to hold all of our requirements compiled through the many meetings with our client. As a team, we really liked the process of using Jira. It kept us extremely organized throughout the whole project. I would say that the one thing we could've done better in relation to release planning was making sure that we included more testing tasks instead of pushing it till the last sprint. We could always improve with more in depth testing during the process rather than pushing it off.

See the Annex for our user story map showing our epics and the user stories in each one.

3.2 Sprint Planning

Sprint Planning involved determining team roles and assigning people to each user story. When we started assigning tasks, we first let people "claim" tasks that they were interested in or had already looked into. Our focus was on not having people work on aspects that they would then have to do a lot of research on to catch up. After that, we assigned the

tasks that were left over to people who were willing and had time to take on more tasks. During this sprint planning we also talked about a timeline for when things expected to be done so that we made our respective deadlines.

In the initial sprints, one of the key activities was the set-up of our implementation environment, consisting of hardware, software, tools, which is described in more detail in a later section.

Sprint planning was a pivotal part of each sprint. Without the planning process we wouldn't have been able to complete as much as we did. We worked very well as a team when it came to planning out our sprint and completing the tasks that we expected to get finished. We had a good idea on what we could complete in the time for each sprint. None of our expectations were unrealistic. I don't think that we really made any mistakes when it came to our sprint planning. We were all attentive and focused on the purpose of each sprint in the assigning of roles and tasks.

The project involved four sprints, each described in the following sections.

In regards to releasing the application in 4 separate increments, it allowed our client (Wesley) to see and play with our project and see what we were creating. It gave him a sense of the work that was being put into it and the ability to watch it evolve over time with different requirements. As a team, we also liked the heroku releases because it gave us a way to test exactly how the project would perform for our client. We also were very diligent on releasing early or on time. If we could do something differently, we would try to deploy to heroku many times in one sprint rather than just once at the end.

3.3 Sprint 1:

Key activities included the following:

- Set development priorities
- Selected a Scrum Master and a Product Owner
- Set up Jira and the Github repository
- Set up code climate to rate our code - SimpleCov and Rubocop
- Set up meeting times with our customer and gathered their requirements, which we converted into user stories
- Create the database relational model to set up the backend for our project.
- Created all initial documentation in relation to metrics, stakeholders, risks, and communication plan.
- Discussed user stories 1, 2, and 3. The customer was pleased with our progress because they didn't have an idea about the functionality that could be created in a single sprint.
- We implemented the following 3 user stories:

User Story 1

Feature	Database Access to be Limited to Only Admins
As an administrator of the system	● So that member information is protected by a secure login ● I want to have to login to view the information of members.

User Story 2

Feature	Add Events
As an administrator of the system	● So that members could eventually sign in and out of those events. ● I want to be able to create future events that require attendance.

User Story 3

Feature	View Activity of Members (Sign-Ins)
As an administrator of the system	● So that the admin can make sure people are attending events ● I want users to sign in to events and admin to be able to view the attendance.

- For user story 1, we used a gem called ActiveAdmin that helped us create an admin side of the application that has access to the database models through a secure log-in. In sprint 1, there was not much to this admin view but it was a start as we built up our application.
- For user story 2, we created the CRUD for Events which then got implemented into the Admin Dashboard view since only admins should be able to CUD events. A view of all events was also added to the homepage for members.
- For user story 3, we created the initial model information and migrations for student sign ins. At this point in time we hadn't created the student model so complete sign-in functionality was pushed for later. The model CRUD was added to the admin view.

3.4 Sprint 2:

Key activities included the following:

- Discussed the Heroku app where the customer could view our progress without having to meet in person.
- Discussed user stories 4, 5, 6, and 7 with the client. With the implementation of these four user stories, all main functionality will be completed. Client is happy with the prospective product.
- We implemented the following 4 user stories:

User Story 4

Feature	Have Members Sign Up for Events

As an administrator of the system	● So that the admin has an estimate on how many are attending a future event ● I want events to have a sign-up option.

User Story 5

Feature	Export Data on Admin Side
As an administrator of the system	● So that the admin can do analysis on the data. ● I want a CSV export option on all admin table views.

User Story 6

Feature	See Upcoming Events on the Homepage
As a member	● So that the member can plan their week accordingly ● I want only future and current events to show.

User Story 7

Feature	Easily Sign In to Events
As a member	● So that the member gets attendance credit. ● I want a simple sign-in form for each event.

- For user story 4, we created a sign up option for any future events. The admin also has a dashboard view of these sign ups protected by their admin log in.
- For user story 5, we made sure that all tables in the admin dashboards were exportable by CSV and in a readable format.
- For user story 6, we made the events view on the homepage only show current and future events. These events were also sorted by date ascending.
- For user story 7, we created both the student model and the sign in form so that students could begin receiving attendance for their sign-ins to events.

3.5 Sprint 3

Key activities included the following:

- We discussed the user stories 8, 9, 10, and 11.
- Our client felt the feature set was progressing nicely.
- Our client was surprised by the CSS changes added. Really liked the layout of the changes.
- Our client was concerned about the possible connection between the other PAID app.
- We implemented the following 4 user stories:

User Story 8

Feature	Data Stored Properly

As an administrator of the system	● So that the data on the admin side is correct ● I want inputs to be in the proper format.

User Story 9

Feature	Information to be Secure
As an administrator of the system	● So that no one can log into the admin account without permission ● I want admin user and password to be non injectable and not in plain text.

User Story 10

Feature	Sign-Ins Connected to the Given Member
As a member	● So that the member gets all of their attendance counted. ● I want email validation and no duplication of the model.

User Story 11

Feature	Site Layout to be Intuitive
As a member	● So that the member can record their information without any usability problems. ● I want a nice and readable sign in, sign out, and sign up pages for the user.

- For user story 8, we implemented input validation on all forms for the user. Admins also have the ability to edit any information in the table views in the admin dashboards.
- For user story 9, we made sure the admin login information was hashed and also the log in was non-injectable.
- For user story 10, we fixed how the sign ins connect to a student to prevent the duplication of a student. Connects by student tamu email.
- For user story 11, we focused on adding nice reactive CSS that allows the user to more easily use the application.

3.6 Sprint 4:

Key activities included the following:

- Connected our app with the other PAID team's app.
- Completed small bug fixes.
- Final demonstration to the client before the maintenance period.
- 100% code coverage.
- We implemented the following 2 user stories:

User Story 12

Feature	Admin Dashboards to be Consistent

As an administrator of the system	● So that there is a good feel across the panels. ● I want the dashboard to have the same options across the panels.

User Story 13

Feature	No Bugs / Errors
As a member	● So that there are no issues with attending events ● I want all things to be thoroughly tested.

- For user story 12, we made sure that the main dashboard has useful information for the admins. We also made sure all filters were standardized and each table had help text explaining the filters and CRUD options.
- For user story 13, we made sure we had 100% code coverage and all functionality worked as expected.

3.7 Scrum Ceremonies: Sprint Review, Retrospective, and Daily Scrum

Throughout the process of each sprint we used sprint reviews, sprint retrospectives, and daily scrum meetings. We used daily scrum meetings to make sure everyone was up to date on the latest changes and what was going on. Overall our daily scrum meetings were the most helpful when it came to team organization. Sprint reviews were extremely important due to the interaction with the client. It gave us a time to show Wesley exactly what we accomplished and what we planned going forward. This meant that if he had any additions needed, we could add them. The sprint retrospectives started feeling repetitive throughout the 4 sprints. We had a well organized team so we really didn't really have much to change between sprints. If we were to do any of these differently, I would've tried to push for more information from our client in the earlier sprint reviews. With our client not being of the CSCE world, I don't think he really knew what to expect or what to ask for. A lot of what we did was what we thought they would like. An earlier idea of expectations might've been more helpful in creating a more advanced version of our application.

See the Annex for slides presented during one of the sprint reviews and results of our final sprint retrospective.

3.8 Compliance

In relation to compliance of our requirements, we created a metric spreadsheet to make sure that we were doing what we needed to. Some of our indicators included the Definition of Done, Acceptance Criteria, and Testing Coverage. We also reviewed all of our requirements with our client throughout each of the sprint reviews. At the end of sprint 4, we had two evaluations sent to our client with the first one being the acceptance criteria and the second being a more team evaluation.

See the Annex for the results of the user acceptance test and customer evaluation of the

team's professionalism.

3.8.1 Acceptance Criteria

To help ensure compliance with user requirements, we used acceptance criteria which allowed us to know when we have properly completed one of our requirements.

Acceptance criteria helped our group know when we could confidently say that we implemented something. We used acceptance criteria to show our client that we completely understood and complied with the requirements that he wanted to see in our project. To use acceptance criteria to its fullest potential, we should've used them from the beginning of sprint 1. We only started using them to their fullest potential in sprint 3. If we used it in sprint 1, we could've had more functionality from the git go.

An example of an acceptance criteria we used for one of our user stories is shown below.

User story: As a site user, I would like the site layout to be intuitive so I can record my attendance
Acceptance criteria:

- Given I am a site user, I should see the current and upcoming events on the homepage
- Given I am a site user, I should be able to sign in/up/out for my specific event
- Given I am a site user, If I am confused there should be documentation to explain how to use the site correctly
- Given I am a site user, if I navigate to the incorrect page, I should be able to navigate back to where I was
- Given I am a site user, I should be able to show attendance for an event without having to sign in
- Given I am a site user, fields to fill in form sign in/up/out are clearly marked and their use is explicitly declared

Completion status: User story completed and verified based on predefined acceptance criteria.

4 Non-Functional Requirements

The non-functional requirements for our project can be described using the project objectives. These requirements were:

- Efficiency
- Correctness & Reliability
- Maintainability
- Integrity

4.1 Efficiency

Efficiency can be defined in this context as being continuously productive with a low waste of effort. Our team repeatedly rolled our system out on time every sprint, which aligned with this requirement. We split our user stories up evenly each sprint to divide the workload. We were very productive each sprint and almost never wasted effort. The one instance we worked on a function we didn't use was when we created database fields for the sign-in and sign-out times for events. We removed these fields because Rails already captures the time records are created and the time they are updated. Our team could have used this functionality initially so less time was spent on functions we didn't use in the end.

To properly measure how efficient our team was, we kept track of key performance indicators each sprint. First, we measured the actual release date versus the expected release date each sprint. For each sprint, we finished on time and released our product with zero variance from the expected release date. We also measured velocity and percentage of issues assigned to each team member. Our velocity varied each sprint, but stayed within the range of ten to twenty. As for issues assigned to each team member, we were each assigned an even amount of story points.

The last three performance indicators for efficiency are number of unresolved issues, burn down chart, and tasks completed over tasks assigned. Each sprint, our team resolved every one of the issues we assigned, also making our tasks completed consistently 100%.

4.2 Correctness & Reliability

To ensure we had correct and reliable code, we did code reviews every sprint. This is where we would test each team member's code to ensure it worked properly and that the finished product at the end of the sprint had no defects. We also used behavior-driven design and test-driven development to make sure our product aligned with the client's wishes and requirements.

4.2.1 Behavior-Driven Design (BDD)

Behavior-Driven design is a software development methodology in which an application is specified and designed by describing how its behavior should appear to an outside observer.

What we learned:

- Ask questions about behavior of the application before and during development to reduce miscommunication.
- Don't be afraid to communicate with the client after the initial requirements meeting. If more clarification is needed, it is better to communicate and have a clear understanding of the design before

implementing it.

- Communicate with your team thoroughly. Everyone in the team should be on the same page when it comes to the design of the application or the product will be implemented incorrectly.

4.2.2 Test-Driven Development (TDD)

Test-Driven development involves writing unit tests before coding (validation by testing), and involves the following:

1) The developer writes an (initially failing) automated test case that defines a desired improvement or new function;
2) He/she produces the minimum amount of code to pass that test;
3) Refactors the new code to acceptable standards.

What we learned:

- These unit tests are quite tedious to write versus just going straight to coding, however we found that it helped us better design our system. We were better prepared for the possible failures of the system.
- The configuration of the testing frameworks is just as important as the tests. If configured incorrectly, one failed test could fail all others as a result.

4.2.3 Reviews

Code Reviews were a big part of ensuring quality and involved reviewing our team's code with multiple pull requests. Each sprint, our team would commit directly into the dev branch. Near the end of the sprint, when we finished with our functionality, we would submit a pull request to the test branch. In this pull request, team members would comment on code based on the definition of done and coding conventions. After the pull request got approved by each member, we would merge to test and submit another pull request to master to review the code coverage and tests. Again, each member would review the code and approve based on the content of the request. After merging this pull request, we would push our code to Heroku.

What we learned:

- Code reviews allowed us to discover defects related to merge conflicts and verify that we followed the different standards set forth in the project, including the definition of done and the coding standard.
- More code reviews are better than less. Reviewing our team code frequently allowed us to catch defects up to the last few minutes of the sprint.

4.2.4 Definition of Done

The "Definition of Done" is a guide to determining completion of a user story and its tasks. It is very important to include when planning out your sprints because it is what defines success. If everyone on the team is on the same page about what needs to be done in order for the sprint to be completed then the process should go a lot smoother.

See the Annex for the project's "Definition of Done."

4.2.5 Test Coverage

Evaluating test coverage is one of the methods that indicate how well the code was tested. A tool we used to determine coverage was simplecov which tracks how many lines of code we have tested. After running rspec tests simplecov is able to determine which lines in the code it ran through and which it didn't. Simplecov determined that we have test coverage for 100% of our code.

See the Annex for the final simplecov output for our application.

4.3 Maintainability

One of the approaches we used to fulfill this requirement is the use of a code style checker, Rubocop, to detect code smells such as excessive nesting of conditional and looping constructs, methods with too many parameters, layout, etc.

See the Annex for our final RuboCop output.

We also reviewed our code against guidelines of clean code as specified in our coding standard, which is included in the Annex. Furthermore, we consistently tracked the mean time to repair any issues as well as the length of time for an emergency fix to production. For each sprint these metrics well exceeded our goals.

4.4 Integrity

Common vulnerabilities in our application included the fact that the event password could be guessed and sign in data could be edited. Some strategies we used to address these vulnerabilities included making sign-in and sign-out passwords separate the remove the ability for a user to guess a password correctly. We prevented the editing of sign-ins by creating tests for SQL injection and preventing all user ability to do so. We also tracked the number of security breaches and the number of security tests created during development.

4.5 Usability

[Reflect on how the team met this objective. Include the key performance indicators related to this objective.]

The team first gathered user expectations related to "usability" and reflected them as either user stories or acceptance criteria within particular user stories. Several UX models (e.g., Lo-fi mockups/storyboards, etc.) were developed, as shown in the Annex. Beside each form, is the corresponding UX model.

After each sprint, we also asked our client to complete a feedback form to give any complaints and score the usability. We tracked both of those metrics as well as compliance with acceptance criteria and completion of user stories. For all of these metrics we were able to consistently hit our goals for each sprint.

4.6 Monitoring and Control using Key Process Indicators

At the beginning of the project we created a backlog of all of the user stories that we wanted to complete by the end of the project. During each sprint we would take slightly more than a fourth of the stories to leave room for improvement. This way, we were able to ensure that if we completed each sprint on time, we would be able to complete the entire project plus some stretch goals within the four sprints.

Furthermore, within each sprint, we time tracked each task and divided the work equally so that we could finish all the stories for that sprint in time. We tracked this data through the use of Jira's burndown chart, which is located in the Annex.

5 Implementation Environment: Hardware, Software Tools, GEMs

Using Heroku, the initial release of our app (tamu-paid) was on 09/20/2020. The final demo was done on 11/10/2020 when all features were completed and tested.

We established development, test and production databases and used PostgreSQL to manage local databases.

The following packages and tools were used in the project:

- Ruby 2.7.1
- Rails 6.0.3
- PostgreSQL 9.3
- rspec-rails 3.9.0
- Heroku
- Git Hub

The following gems were also used:

- puma
- turbolinks
- sass-rails
- webpacker
- jbuilder
- activeadmin
- devise
- simple_form
- rack-cors
- pg
- bootsnap
- simplecov
- spring
- listen
- web-console
- capybara
- selenium-webdriver
- tzinfo-data

We used Jira to distribute the tasks among the team members and keep track of the progress in the development.

5.1 Heroku

Fortunately, we were able to deploy our application on Heroku without any problems. However, because our code repository was in the TAMU version of Github, we were unable to get CI/CD working due to Heroku being unable to locate our repository.

5.2 Continuous Integration (CI) / Continuous Delivery (CD)

CI / CD is the ability to continuously deploy and integrate your project to the test and master environments so that users and other developers can get rapid updates on any changes made to the code. Unfortunately, we were unable to get CI/CD working for our project because our code repository was located in the TAMU version of Github. Heroku was unable to find our repository there so we were unable to get CI/CD working.

6 Data Design

A relational database was used for the project using PostgreSQL DBMS for Heroku. To ensure completeness of the data model, and assess it for quality, an entity relationship diagram was

developed.

The data design for our system was developed easily after careful analysis of client needs.

When connecting with the other PAID team, we used an API. Our team created endpoints for the other team to access event data. This was slightly difficult at first because both teams had slightly different database designs. By the end, we configured our systems to work seamlessly.

See the Annex for our data design.

7 Risk Analysis

Risk Analysis was used to identify risks, which are potential problems that may occur. For each risk, we estimated the probability of occurrence, and impact should the risk become reality. Using this method helped us spend time critically thinking about potential flaws in our application and to analyze how we could minimize the related risks. Establishing risks early on helps team members identify how their actions during the sprint could affect other team members and the finished products.

Each risk was ranked based on the risk exposure score and multiplied by the impact score, after which a cutoff line was decided, with risks above the cutoff line considered important to mitigate, monitor, and manage. Once the risk table had been established, it was important for the team to constantly monitor the risks to see which risks will likely become reality. Usually we would find ways to reduce the risks fairly quickly in each sprint.

Risk	Problem in %	Impact	RMMM Plan	Status
Customer discovers many defects	20%	High	Monitoring: Create tests during every spint Mitigation: Frequently do code reviews and test team code.	On Schedule, low probability of happening due to testing.
Release of Sprint 4 App will be delayed	10%	High	Monitoring: Conduct Daily Stand-Up Meetings Mitigation: Frequently communicate (SCRUM meetings) and encourage team members to ask for help if needed	On Schedule, low probability of happening.
Hours not logged properly	30%	Medium	Monitoring: Conduct Daily Stand-Up Meetings Mitigation: Frequently communicate (SCRUM meetings) and encourage team members to ask for help if needed	On Schedule
Team Member falls behind	10%	High	Monitoring: Conduct Daily Stand-Up Meetings Mitigation: Frequently communicate (SCRUM meetings) and encourage team members to ask for help if needed	On Schedule, no issues thus far
PAID Member signs in when not in attendance	10%	Medium	Monitoring: Create password validation for sign-ins Mitigation: Ensure that the incorrect event password does not work	Low probability due to addition of password for signing in and out of events
User creates multiple student models for the same person	20%	High	Monitoring: Admin checks user log Mitigation: Create user only based on email and edit otherwise	Only possible if student inputs incorrect email and if this happens an admin can easily go and delete the email.

For the majority of sprints we did not encounter any of our predefined risks. We did see team members fall behind on a few occasions but we were able to recover quickly because we were expecting it and had a mitigation plan. Our daily SCRUM meetings and Jira board made it very easy to see exactly how impactful the team member falling behind was.

See the Annex for the risk table with an example of how the risks were monitored.

8 Configuration Management

8.1 Central Repository

Our code was maintained in a GitHub Repository which can be found here.

Other artifacts (e.g., documents, designs) were maintained in MS Teams and in the project Google drive.

8.2 Version Control

In software engineering it is necessary, especially in big projects, to track and control the source code developed. In our project, we decided to use Github, which offers functionalities, such as integrated issue tracking, collaborative code review, team management, and highlighting of syntax. It allowed us to work on separate features of the application, track bugs, and manage coding tasks.

We decided to keep it simple with just three branches, dev, test, and master for all five members of the group to work in at once. We divided work based on features to avoid conflicts when pushing and pulling from the same branch. For more intensive features, we would occasionally create a separate branch named after that attribute.

To overcome the issues of merging different branches in GitHub, every team member would push and pull frequently. Once all tasks were completed in Jira and all tests were passed we would submit a pull request to the test branch. Once the code had been tested thoroughly, the code was merged to the master branch in preparation for deployment to production using Heroku. We did not have continuous Heroku integration setup because we were using the TAMU GitHub, so we would push the code to a non-org repository and use that GitHub to push to Heroku.

In addition, we all had 2 different databases in use: one for Heroku and an individual local PostgreSQL database for testing and development. Finally, the Morris chart library was used to generate statistics charts in GitHub.

8.3 Change Control process

The change control process was used to control changes to the baseline of the project which includes our software, data, and documentation.

When a change needed to be made the process we used would start with an identification of the problem or needed change. We would usually send a message in the team discord after identification and then would discuss who needed to make the change i.e. who owned

that component for that sprint. After finding out who needed to make the change, they would begin the process of coding the solution. If it was simple they would code in the master branch, but if it was more complex they would work on a separate branch. If the issue was very serious we would add the task to the next sprint's list of tasks. Once the change was made and merged with master, we would then re-integrate with Heroku.

9 Deployment, Support, and Maintenance, and Project Close

In Sprint 3, we developed plans for deployment, support, and maintenance, which described installation, conversion of data from the current system (if applicable), user documentation to be provided, training, and support after deployment.

Our strategy for deployment first involves the turning over of the heroku site to a PAID representative. We created an in-depth readme and extensive help page for the website to help future teams work with our code and users understand how our site works. On the admin side of the application, we added help tips on each page for clarity. We also released a training/demo video to aid PAID in teaching administrators how to use the site most effectively. For maintenance, PAID has the option to enroll in the program to have bugs fixed and improvements made next semester.

Microsoft Teams Deployment Plan

10 Other Important Links

Our Jira project can be found here.

Our application can be found in Heroku here.

A video of our application demo can be found here.

The first sprint video interviewing with the customer can be found here.

The final sprint video interviewing with the customer can be found here.

11 References

https://www.atlassian.com/software/jira
https://en.wikipedia.org/wiki/GitHub
https://activeadmin.info/
https://www.linkedin.com/learning/ruby-essential-training-part-1-the-basics/exit-a-running-script?u=74650722

https://www.linkedin.com/learning/ruby-testing-with-rspec/complex-expectations?u=74650722

12 Annex

12.1 Stakeholder Analysis Matrix and Communication Plan

Communication Plan

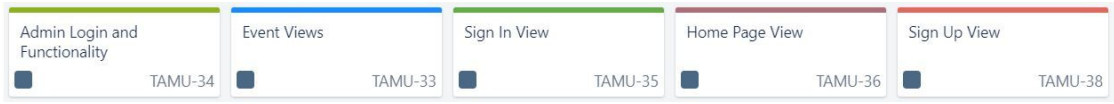
Stakeholder	Information Needed / Document Name	Document Format / Medium	Contact Person	Frequency / When Due	Current Progress
Customer (PAID)	Progress / Status Report Progress / Sprint Review Meeting	Email Microsoft Teams	PAID POC	Once a week/At the end of every sprint	Communicating More
Instructor	Progress / Status Report	Microsoft Teams Jira Github Sprint Deliverable Documents	Professor Wade	At the end of every sprint	On Track
TA / Grader	Progress / Status Report	Sprint Deliverable Documents	Naghma	At the end of every sprint	On Track
Project Team	Progress / Status Report	Discord / Microsoft Teams	Full Development Team	Daily	On Track

Stakeholder Analysis Matrix

Stakeholder	Level of Power / Level of Interest	Requirements / Expectations	Preferences	Strategies for Gaining Support / Reducing Obstacles	Status
Customer (PAID)	High / High	- Usable System without bugs - Must be able to track member participation in events	- User side for member sign in and sign out - Admin side to create events	- Stay in contact with the client - Conduct Sprint Reviews to ensure the client is getting the product they expected	On Track
Instructor (Professor Wade)	High / High	Project finished in time and delivered to customers	Finished quality project by November 20th	- Conduct daily SCRUM meetings to ensure team is on track - Ensure that sprints stay within the scope of the project so that they can be completed within the existing time frame	On Track
TA / Grader	Low / Low	Due dates are met by project team	Project turned in on time and well documented	- Conduct daily SCRUM meetings to ensure team is on track - Ensure that sprints stay within the scope of the project so that they can be completed within the existing time frame - Frequently update documentation so it stay up to date	On Track
Project Team	High / High	All customer needs are met and quality project is finished on time	Meeting all deadlines and complete functionality	- Conduct daily SCRUM meetings to ensure team is on track - Ensure that sprints stay within the scope of the project so that they can be completed within the existing time frame	On Track
Members of PAID	Low / High	Product is easy to use and has complete functionality	Product is easy to use and has complete functionality	- Stay in contact with the client - Conduct Sprint Reviews to ensure the client is getting the product they expected	On Track

12.2 Epics and User Stories

Epics



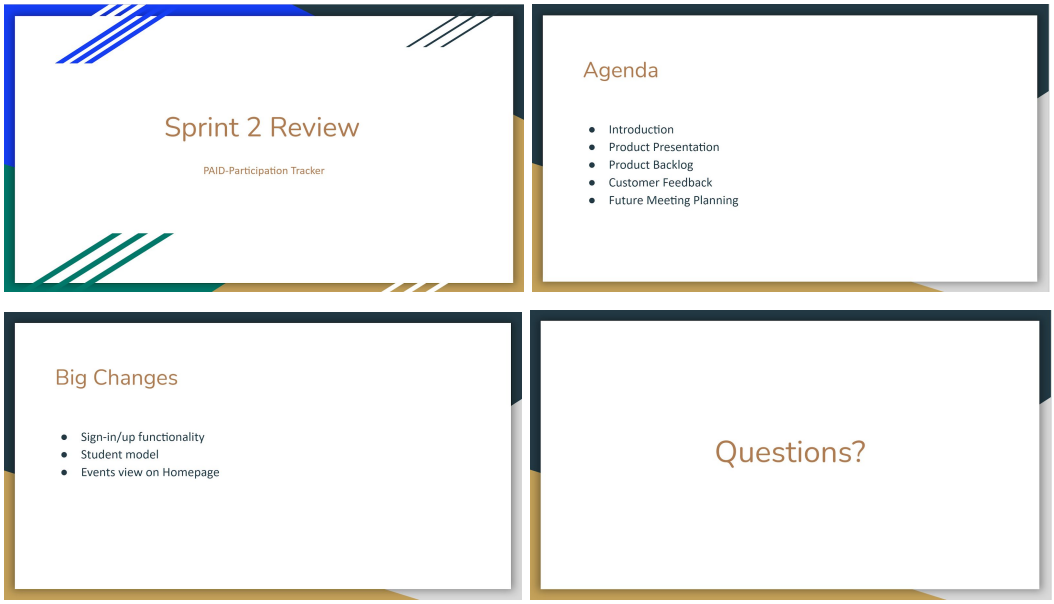
User Stories

1. As an administrator, I would like to be able to add events so that my members can sign in and out of those events
2. As an administrator, I would like database access to be limited to other administrators so member data is protected
3. As an administrator, I would like to be able to view everyone's activity attendance so that I can make sure everyone is attending the events
4. As a user, I would like to be able to easily log my info so that I get attendance credit
5. As an administrator, I would like to be able to easily export the data so I can do analysis on it
6. As an administrator, I would like to have people be able to sign up for events so I can have an estimate on how many will attend it
7. As a user, I would like to see upcoming events so that I can plan my week accordingly
8. As a site user, I would like the site layout to be intuitive so I can record my attendance easily

9. As an administrator, I would like to have the data stored properly so that I can trust the data I receive
10. As a user, I would like to be able to have all my sign-ins connected to me so that I get credit for all my attendance
11. As an admin of the organization, I would like my information to be secure so no one can log into my account without permission
12. As a site user, I don't want to encounter any bugs/errors while using the site
13. As an admin, I would like the admin dashboards to be consistent so that there is a good feel to the website.
14. As a user, I need sufficient user documentation, so that I can use the system even without the help of the development team.
15. As an admin, I need to have sufficient documentation, so that I can administer the system even without the help of the development team.
16. As an admin, I need to have sufficient 'system' documentation, so that I can set-up, maintain, and deal with application issues without the help of the development team

12.3 Sprint Review and Final Sprint Retrospective Results

Screenshots of Powerpoint from the Sprint 2 Review:



[Retrospective Minutes]

Sprint 4: Sprint Retrospective

11/8/20

What went well this sprint?

- most organized sprint
- got everything done very early
- limited the last minute fixes
- no big pair programming – everyone knew what they were doing
- helpful teammates
- good coordination
- better communication with the other team
- 100% code coverage
- better output with rubocop
- the smoothest sprint so far
- did well last couple of sprints, making this sprint much easier
- we finish sprints with completed things, never fixing huge bugs in the next sprints

What went wrong this sprint and what could we have done differently?

- API creation and the other team not having the date
  - we didn't know they had an issue until Thursday morning.
  - Could have been fixed sooner if they had communicated better
- Difficulty of cross-team communication (mentioned by 3 people)
  - Constant communication in many forms
  - could have had at least one stand-up a week with the other team
- Other teams had slideshows for presentations, which could have amplified our own presentation
  - Create a slide or two for any future presentations
  - Ensure final presentation is organized
- Missing final sprint review due to scheduling conflicts
  - Mention schedule conflict to Pauline early.

What did we learn this sprint?

- Lauren:
  - learned more about CSS, Active Admin, more technical stuff
- Rory:
  - hadn't worked on active admin before, so learned more about the gem
- Jack:
  - Learned more about communication
  - Learned about apis and rspec
- Neil
  - Learned about implementing mailers
- Roni:
  - Learned more about active admin
  - learned more about Simplecov
- All:
  - learned how to be flexible and work on things on the fly

Now What?

- Good job to the team!
- Probably good for the maintenance period

Review Minutes

- **Pauline's Slide Presentation**
  - Timeline
    - Deliver app by 11/9
    - UAT from 11/9 to 11/15
    - Collection of UAT Form on 11/16
    - Maintenance Phase 11/16 to 11/22
    - Official Project close 11/22
    - Collect Customer Evaluation and any more UAT by 11/24
    - End of Support 11/24
  - Proceed with Demos
  - Loop back to forms
- **PAID-Participation Tracker**
  - Website similar to end of Sprint 3
  - New Features:
    - Implemented Mailer for Forgot Password Emails
    - Successful Password Change
    - Added links to Admin Dashboard
    - Exporting Data – Updated CSV Data Download Format
    - Event Creation – Added AM PM functionality
    - Help Text added to admin panel
    - Events now do not require sign out password
      - With no password, so sign out will be required
  - Project Backlog
    - Empty Backlog
    - All requested functionality implemented

12.4 User Acceptance Test Results & Customer's Evaluation of the Team's Professionalism

User Acceptance Test

User Acceptance Tests

All tests accepted by customer

Number	User Story with it's acceptance criteria	Critical		Test Result		Comments
		Yes	No	Accept	Reject	
1	As an administrator, I would like to be able to add events so that my members can sign in and out of those events					
	Acceptance Criteria:					
	Given I am an org admin, I should be able to create, modify and delete events from the admin panel	x		x		
	Given I am an org admin, I should be able to see all events in the admin panel.	x		x		
	Given I am a site user, I should not be able to create, modify or delete events without the proper credentials	x		x		
2	As an administrator, I would like database access to be limited to other administrators so member data is protected					
	Acceptance Criteria:					
	Given I am an org admin, the view of sign ins and sign outs should be password protected	x		x		
	Given I am a site user, my attendance should not be accessible by other students.	x		x		
	Given I am an org admin, I should have acess to all information with the proper admin login.	x		x		
3	As an administrator, I would like to be able to view everyone's activity attendance so that I can make sure everyone is attending the events					
	Acceptance Criteria:					
	Given I am an org admin, I should be able to view all sign ins and sign outs for events	x		x		
	Given I am an org admin, I should be able to filter the sign ins and sign outs by the student I am looking for	x		x		
	Given I am an org admin, I can export the activity attendace so that I can run other stats tests on the information	x		x		Really appreciated this getting done
4	As a user, I would like to be able to easily log my info so that I get attendance credit					
	Acceptance Criteria:					
	Given I am a site user, I should be able to sign up for a specific event	x		x		
	Given I am a site user, I should be able to sign in for a specific event	x		x		
	Given I am a site user, I should be able to sign out of a specific event	x		x		
5	As an administrator, I would like to be able to easily export the data so I can do analysis on it					
	Acceptance Criteria:					
	Given I am an org admin, I should be able to export a csv file for the sign-ins of a specific event	x		x		
	Given I am an org admin, I should be able to export a csv file for the sign-ups of a specific event	x		x		
	Given I am an org admin, I should be able to export a csv file for all the events a student has signed up for	x		x		
6	As an administrator, I would like to have people be able to sign up for events so I can have an estimate on how many will attend it					
	Acceptance Criteria:					
	Given I am an org admin, I should be able to see which students have signed up for each event	x		x		
	Given I am a site user, I should be able to sign up for a specific event	x		x		
	Given I am an org admin, I should be able to export a csv file for the sign-ups of a specific event	x		x		
7	As a user, I would like to see upcoming events so that I can plan my week accordingly					
	Acceptance Criteria:					
	Given I am a site user, I should be able to see all upcoming events on the homepage		x	x		Not needed, but makes it easier!!
	Given I am a site user, the events on the homepage should be in order of occurence so I can properly plan	x		x		
	Given I am an org admin, I should be able to see the upcoming events for the week on the admin page [redacted]	x		x		
8	As a site user, I would like the site layout to be intuitive so I can record my attendance easily					
	Acceptance Criteria:					
	Given I am a site user, I should see the current and upcoming events on the homepage	x		x		
	Given I am a site user, I should be able to sign in/up/out for my specific event	x		x		
	Given I am a site user, if I am confused there should be documentation to explain how to use the site correctly	x		x		
	Given I am a site user, if I navigate to the incorrect page, I should be able to navigate back to where I was	x		x		
	Given I am a site user, I should be able to show attendance for an event without having to sign in	x		x		
	Given I am a site user, fields to fill in form sign in/up/out are clearly marked and their use is explicitly declared	x		x		
9	As an administrator, I would like to have the data stored properly so that I can trust the data I receive					
	Acceptance Criteria:					
	Given I am an org admin, the data stored in the admin panel remains constant and consistent	x		x		
	Given I am an org admin, I would like to be able to export my data in various formats (CSV)	x		x		
	Given I am an org admin, the data stored in the admin panel is secure from external threats (SQL Injection)		x	x		This was amazing, thank you.
	Given I am an org admin, I should be able to create, modify and delete events from the admin panel	x		x		
	Given I am an org admin, the sign in/up/out data should contain valid Texas A&M email addresses	x		x		This helps a lot for organization and confusion.
10	As a user, I would like to be able to have all my sign-ins connected to me so that I get credit for all my attendance					
	Acceptance Criteria:					
	Given I am a site user, my attendance is successfully recording when I sign into an event	x		x		
	Given I am a site user, my attendance information should match with the PAID points tracker site information	x		x		
	Given I am a site user, I should receive some visual conformation of my successful sign in/up/out	x		x		
11	As an admin of the organization, I would like my information to be secure so no one can log into my account without permission					
	Acceptance Criteria:					
	Given I am an org admin, I want only authorized users to be able to sign in	x		x		Protects from outside users, thank you.
	Given I am an org admin, invalid users should not be able to access the admin panel	x		x		
	Given I am an org admin, the data stored in the admin panel remains constant and consistent	x		x		
	Given I am an org admin, if an admin account is deauthorized then it should be removed from the admin users	x		x		
12	As a site user, I don't want to encounter any bugs/errors while using the site					
	Acceptance Criteria:					
	Given I am a site user, I should not be able to sign-in to any events that are not currently ongoing.	x		x		Yes, people would sign in for wrong one.
	Given I am a site user, I should be able to sign-in/up/out for my specific event.	x		x		
	Given I am a site user, I should not be able to sign-up for any events that are currently ongoing.	x		x		
	Given I am a site user, the site visuals should remain consistent throughout the extent of my usage.	x		x		
	Given I am a site user without admin privileges, I should not be able to acces the admin panel.	x		x		
	Given I am a site user with admin privileges, I should be able to successfully login to the admin panel.	x		x		
	Given I am a site user, I should recieve a green feedback notice if I succesfully sign-in/up/out for my specific event.	x		x		
	Given I am a site user, I should recieve a red feedback notice if I incorrectly sign-in/up/out for my specific event.	x		x		
13	As an admin, I would like the admin dashboards to be consistent so that there is a good feel to the website.					
	Acceptance Criteria:					
	Given I am a site admin, all dashboard sections should maintain consistent formatting.		x	x		Makes it prettier though, thank you.
	Given I am a site admin, all dashboard sections should maintain a consisten color scheme.		x	x		
	Given I am a site admin, I should be able to filter all dashboard sections.	x		x		
	Given I am a site admin, I should be able to export data from all dashboard tables.	x		x		
	Given I am a site admin, I should be able to edit the data in all dashboard tables.	x		x		
	Given I am a site admin I should be able to delete the data in all dashboard tables.	x		x		
	Given I am a site admin, the main dashboard should contain only relevent information.	x		x		
	Given I am a site admin, I should be able to safely logout from the admin panel.	x		x		
	Given I am a site admin, I should be able to create/update/delete new or other admin users.	x		x		
14	As a user, I need sufficient user documentation, so that I can use the system even without the help of the development team.					
15	As an admin, I need to have sufficient documentation, so that I can administer the system even without the help of the development team.					
	Acceptance Criteria					
	-Transferring admin privileges to another member	x		x		This was a must, thank you for getting this done.
	-Managing members (for example, add/delete/modify)	x		x		
	-Other admin functions specific to my organization	x		x		
16	As an admin, I need to have sufficient 'system' documentation, so that I can set-up, maintain, and deal with application issues without the help of the development team					
	Acceptance Criteria					
	-Transfer of application to another Heroku account, in the event that the member is no longer with the organization	x		x		
	-Recovering / restarting the application in the event it crashes or is consistently slow or how to reach Heroku support	x		x		
	-Details on how often we need to backup the data		x		x	
	-Backing up the data using Heroku functionality and restoring		x		x	
	-Creating a local backup and restoring (e.g., export/import)		x		x	

Evaluation of Professionalism

1. Customer satisfaction *

Delivers requirements that contributes the most to customer satisfaction; find ways to exceed expectations without making unrealistic commitments

- ○ 0
- ○ 1
- ○ 2
- ◉ 3

2. Agility *

Responds to customers and/or implement requirements with a sense of urgency.

- ○ 0
- ○ 1
- ○ 2
- ◉ 3

3. Acts with integrity *

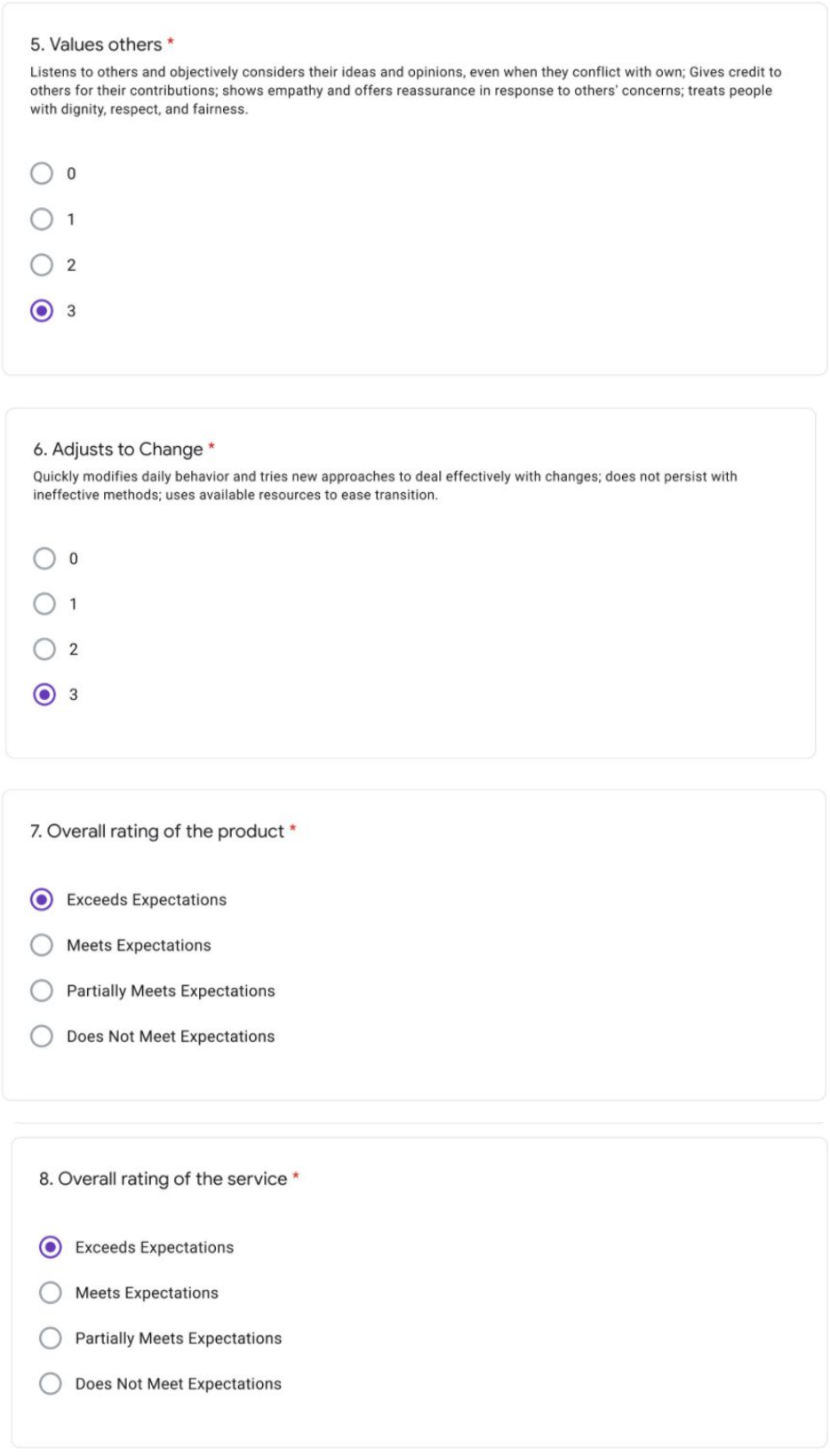
Demonstrates honesty; behaves in a consistent manner; keeps sensitive information confidential; adheres to moral, ethical, and professional standards, regulations, and organizational policies.

- ○ 0
- ○ 1
- ○ 2
- ◉ 3

4. Discloses own positions *

Shares thoughts, feelings, experiences, and rationale so that others understand personal positions and feel comfortable sharing similar information; accepts responsibility for outcomes (positive or negative) of one's work; admits mistakes and refocuses efforts when appropriate.

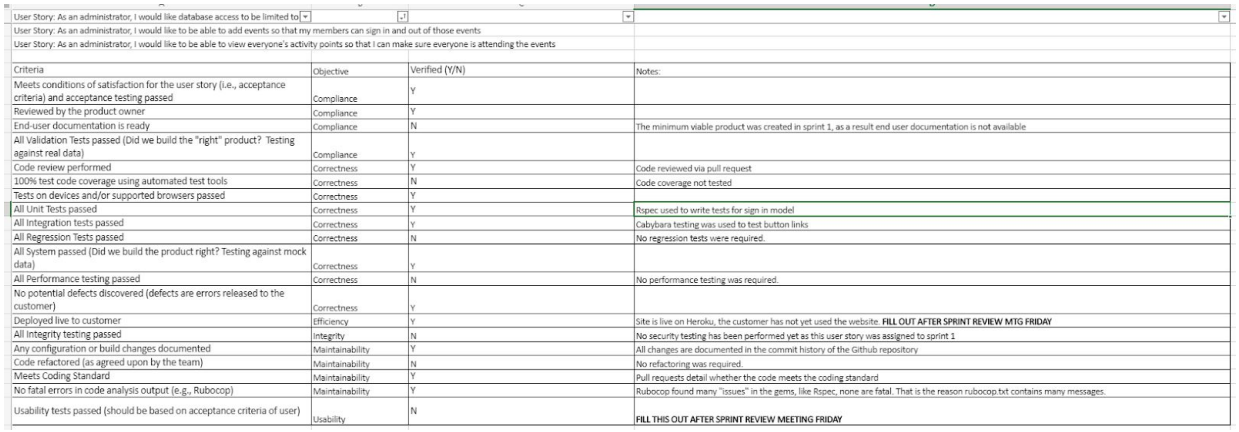
- ○ 0
- ○ 1
- ○ 2
- ◉ 3



12.5 Definition of Done

[Sprint 1]
[Sprint 2]
[Sprint 3]
[Sprint 4]

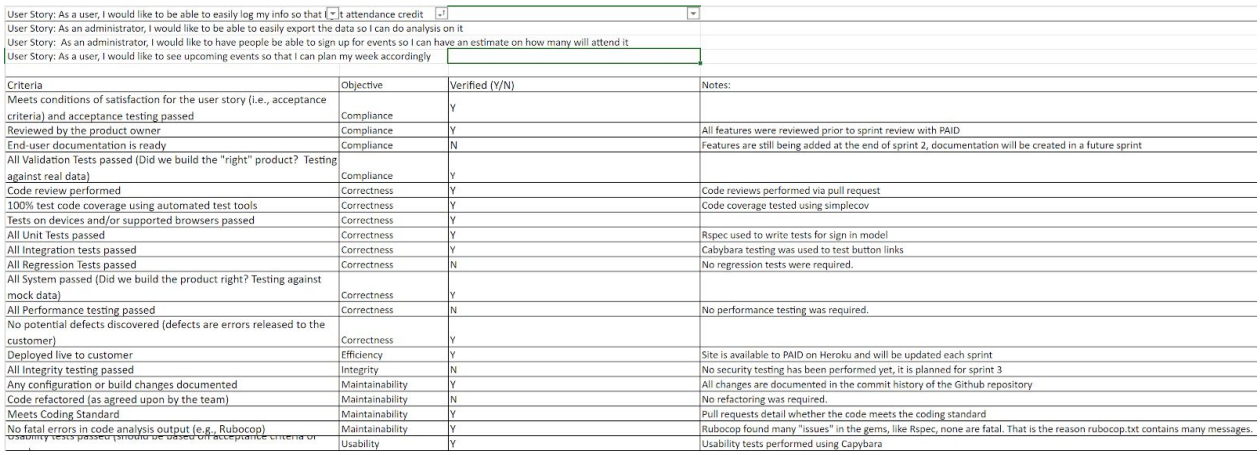
Sprint 1's Definition of Done:



User Story: As an administrator, I would like database access to be limited to
User Story: As an administrator, I would like to be able to add events so that my members can sign in and out of those events
User Story: As an administrator, I would like to be able to view everyone's activity points so that I can make sure everyone is attending the events

Criteria	Objective	Verified (Y/N)	Notes:
Meets conditions of satisfaction for the user story (i.e., acceptance criteria) and acceptance testing passed	Compliance	Y	
Reviewed by the product owner	Compliance	Y	
End-user documentation is ready	Compliance	N	The minimum viable product was created in sprint 1, as a result end user documentation is not available
All Validation Tests passed (Did we build the "right" product? Testing against real data)	Compliance	Y	
Code review performed	Correctness	Y	Code reviewed via pull request
100% test code coverage using automated test tools	Correctness	N	Code coverage not tested
Tests on devices and/or supported browsers passed	Correctness	Y	
All Unit Tests passed	Correctness	Y	Rspec used to write tests for sign in model
All Integration tests passed	Correctness	Y	Cabybara testing was used to test button links
All Regression Tests passed	Correctness	N	No regression tests were required.
All System passed (Did we build the product right? Testing against mock data)	Correctness	Y	
All Performance testing passed	Correctness	N	No performance testing was required.
No potential defects discovered (defects are errors released to the customer)	Correctness	Y	
Deployed live to customer	Efficiency	Y	Site is live on Heroku, the customer has not yet used the website. **FILL OUT AFTER SPRINT REVIEW MTG FRIDAY**
All Integrity testing passed	Integrity	N	No security testing has been performed yet as this user story was assigned to sprint 1
Any configuration or build changes documented	Maintainability	Y	All changes are documented in the commit history of the Github repository
Code refactored (as agreed upon by the team)	Maintainability	N	No refactoring was required.
Meets Coding Standard	Maintainability	Y	Pull requests detail whether the code meets the coding standard
No fatal errors in code analysis output (e.g., Rubocop)	Maintainability	Y	Rubocop found many "issues" in the gems, like Rspec, none are fatal. That is the reason rubocop.txt contains many messages.
Usability tests passed (should be based on acceptance criteria of user)	Usability	N	**FILL THIS OUT AFTER SPRINT REVIEW MEETING FRIDAY**

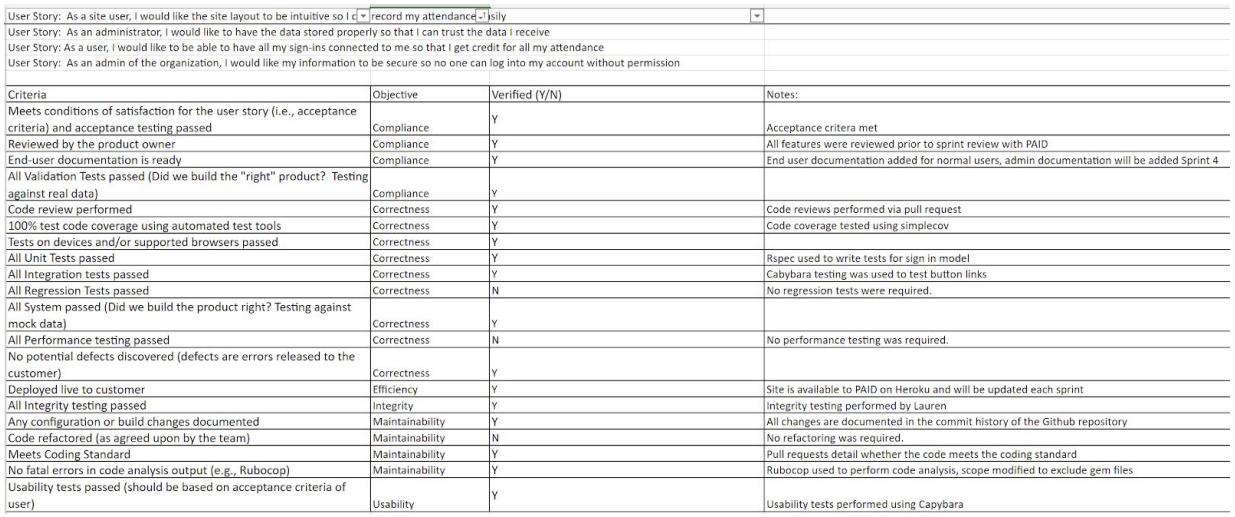
Sprint 2's Definition of Done:



User Story: As a user, I would like to be able to easily log my info so that I get attendance credit
User Story: As an administrator, I would like to be able to easily export the data so I can do analysis on it
User Story: As an administrator, I would like to have people be able to sign up for events so I can have an estimate on how many will attend it
User Story: As a user, I would like to see upcoming events so that I can plan my week accordingly

Criteria	Objective	Verified (Y/N)	Notes:
Meets conditions of satisfaction for the user story (i.e., acceptance criteria) and acceptance testing passed	Compliance	Y	
Reviewed by the product owner	Compliance	Y	All features were reviewed prior to sprint review with PAID
End-user documentation is ready	Compliance	N	Features are still being added at the end of sprint 2, documentation will be created in a future sprint
All Validation Tests passed (Did we build the "right" product? Testing against real data)	Compliance	Y	
Code review performed	Correctness	Y	Code reviews performed via pull request
100% test code coverage using automated test tools	Correctness	Y	Code coverage tested using simplecov
Tests on devices and/or supported browsers passed	Correctness	Y	
All Unit Tests passed	Correctness	Y	Rspec used to write tests for sign in model
All Integration tests passed	Correctness	Y	Cabybara testing was used to test button links
All Regression Tests passed	Correctness	N	No regression tests were required.
All System passed (Did we build the product right? Testing against mock data)	Correctness	Y	
All Performance testing passed	Correctness	N	No performance testing was required.
No potential defects discovered (defects are errors released to the customer)	Correctness	Y	
Deployed live to customer	Efficiency	Y	Site is available to PAID on Heroku and will be updated each sprint
All Integrity testing passed	Integrity	N	No security testing has been performed yet, it is planned for sprint 3
Any configuration or build changes documented	Maintainability	Y	All changes are documented in the commit history of the Github repository
Code refactored (as agreed upon by the team)	Maintainability	N	No refactoring was required.
Meets Coding Standard	Maintainability	Y	Pull requests detail whether the code meets the coding standard
No fatal errors in code analysis output (e.g., Rubocop)	Maintainability	Y	Rubocop found many "issues" in the gems, like Rspec, none are fatal. That is the reason rubocop.txt contains many messages.
Usability tests passed (should be based on acceptance criteria of user)	Usability	Y	Usability tests performed using Capybara

Sprint 3's Definition of Done:



User Story: As a site user, I would like the site layout to be intuitive so I can record my attendance easily
User Story: As an administrator, I would like to have the data stored properly so that I can trust the data I receive
User Story: As a user, I would like to be able to have all my sign-ins connected to me so that I get credit for all my attendance
User Story: As an admin of the organization, I would like my information to be secure so no one can log into my account without permission

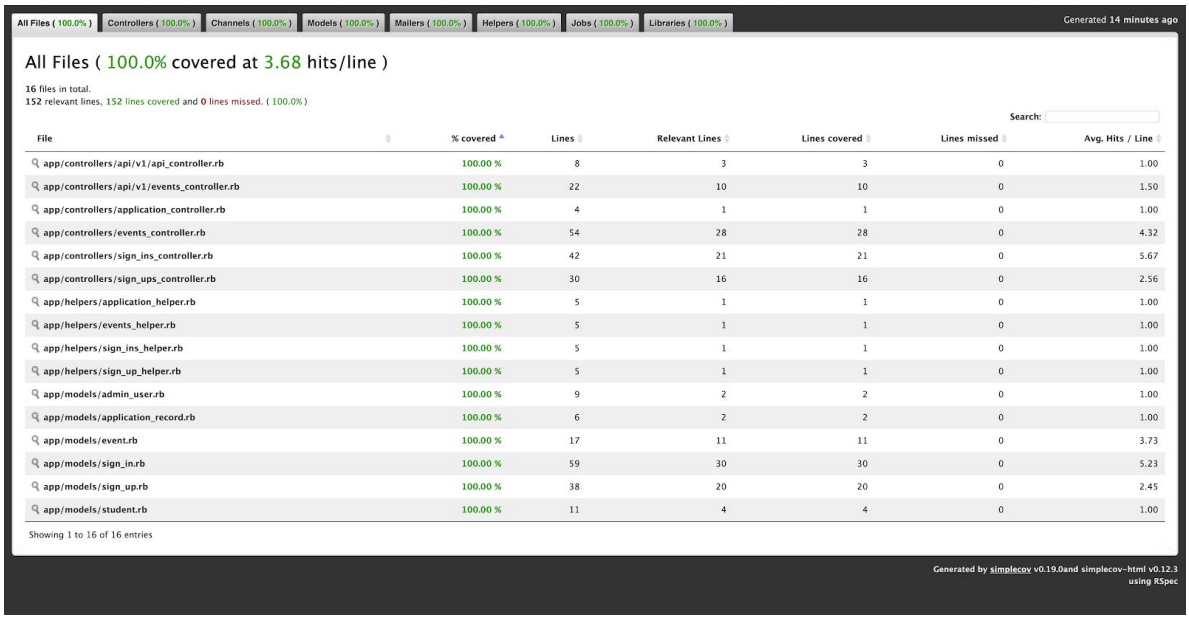
Criteria	Objective	Verified (Y/N)	Notes:
Meets conditions of satisfaction for the user story (i.e., acceptance criteria) and acceptance testing passed	Compliance	Y	Acceptance critera met
Reviewed by the product owner	Compliance	Y	All features were reviewed prior to sprint review with PAID
End-user documentation is ready	Compliance	Y	End user documentation added for normal users, admin documentation will be added Sprint 4
All Validation Tests passed (Did we build the "right" product? Testing against real data)	Compliance	Y	
Code review performed	Correctness	Y	Code reviews performed via pull request
100% test code coverage using automated test tools	Correctness	Y	Code coverage tested using simplecov
Tests on devices and/or supported browsers passed	Correctness	Y	
All Unit Tests passed	Correctness	Y	Rspec used to write tests for sign in model
All Integration tests passed	Correctness	Y	Cabybara testing was used to test button links
All Regression Tests passed	Correctness	N	No regression tests were required.
All System passed (Did we build the product right? Testing against mock data)	Correctness	Y	
All Performance testing passed	Correctness	N	No performance testing was required.
No potential defects discovered (defects are errors released to the customer)	Correctness	Y	
Deployed live to customer	Efficiency	Y	Site is available to PAID on Heroku and will be updated each sprint
All Integrity testing passed	Integrity	Y	Integrity testing performed by Lauren
Any configuration or build changes documented	Maintainability	Y	All changes are documented in the commit history of the Github repository
Code refactored (as agreed upon by the team)	Maintainability	N	No refactoring was required.
Meets Coding Standard	Maintainability	Y	Pull requests detail whether the code meets the coding standard
No fatal errors in code analysis output (e.g., Rubocop)	Maintainability	Y	Rubocop used to perform code analysis, scope modified to exclude gem files
Usability tests passed (should be based on acceptance criteria of user)	Usability	Y	Usability tests performed using Capybara

Sprint 4's Definition of Done:

User Story: As a site user, I don't want to encounter any bugs/errors while using the site			
User Story: As an admin, I would like the admin dashboards to be consistent so that there is a good feel to the website.			
Criteria	Objective	Verified (Y/N)	Notes:
Meets conditions of satisfaction for the user story (i.e., acceptance criteria) and acceptance testing passed	Compliance	Y	Acceptance critera met for each user story
Reviewed by the product owner	Compliance	Y	All features were reviewed prior to sprint review with PAID
End-user documentation is ready	Compliance	Y	End user documentation present for admin and normal users
All Validation Tests passed (Did we build the "right" product? Testing against real data)	Compliance	Y	
Code review performed	Correctness	Y	Code reviews performed via pull request
100% test code coverage using automated test tools	Correctness	Y	Code coverage tested using simplecov
Tests on devices and/or supported browsers passed	Correctness	Y	
All Unit Tests passed	Correctness	Y	Rspec used to write tests for sign in model
All Integration tests passed	Correctness	Y	Cabybara testing was used to test button links
All Regression Tests passed	Correctness	Y	No regression tests were required.
All System passed (Did we build the product right? Testing against mock data)	Correctness	Y	
All Performance testing passed	Correctness	Y	No performance testing was required.
No potential defects discovered (defects are errors released to the customer)	Correctness	Y	
Deployed live to customer	Efficiency	Y	Site is transferred to our PAID correspondent's Heroku account.
All Integrity testing passed	Integrity	Y	Integrity testing performed by Lauren
Any configuration or build changes documented	Maintainability	Y	All changes are documented in the commit history of the Github repository
Code refactored (as agreed upon by the team)	Maintainability	Y	No refactoring was required.
Meets Coding Standard	Maintainability	Y	Pull requests detail whether the code meets the coding standard
No fatal errors in code analysis output (e.g., Rubocop)	Maintainability	Y	Rubocop used to perform code analysis, scope modified to exclude gem files
Usability tests passed (should be based on acceptance criteria of user)	Usability	Y	Usability tests performed using Capybara

12.6 Test Coverage - Output of simplecov

Simplecov Output After Sprint 4

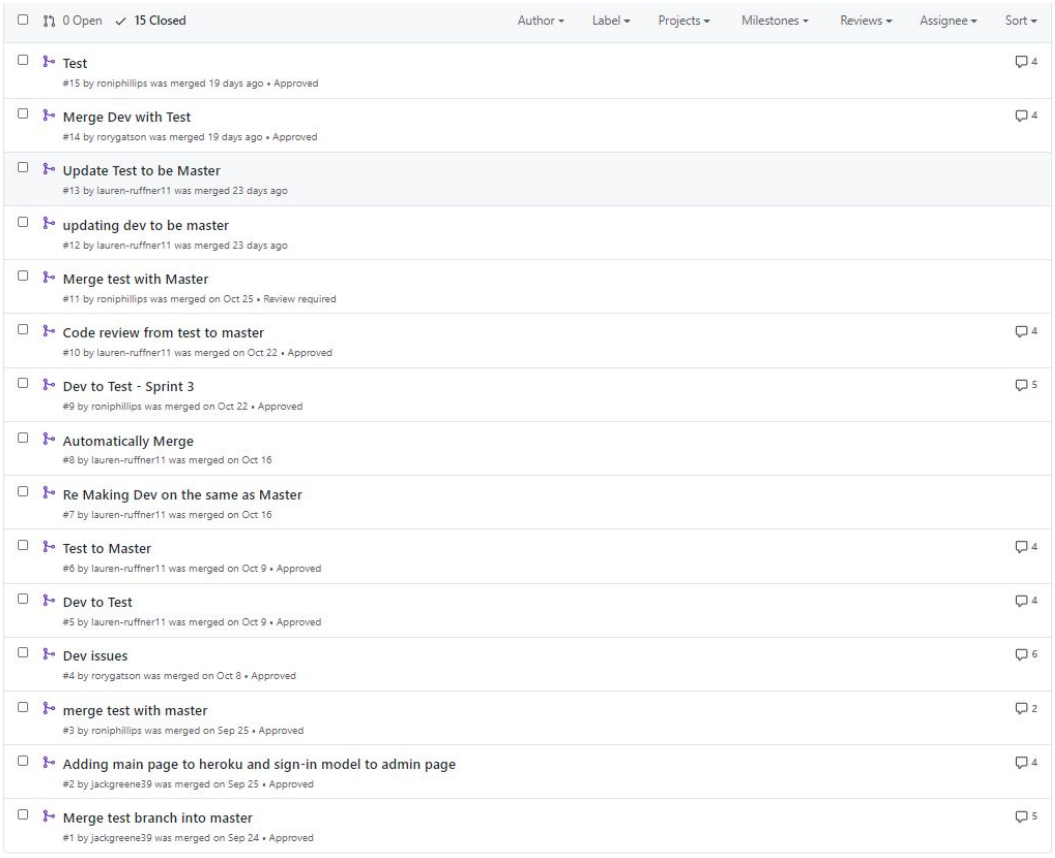


12.7 RuboCop Output

Rubocop Output After Sprint 4

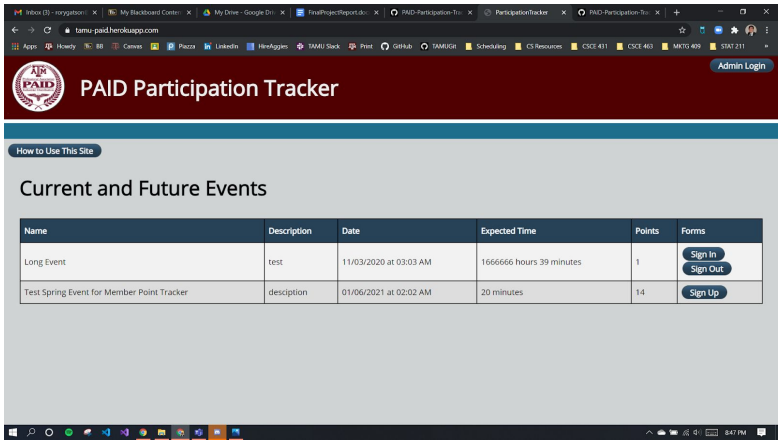
```
Inspecting 47 files
........CCC..............C.CCCC..............

Offenses:

app/controllers/events_controller.rb:10:3: C: Metrics/AbcSize: Assignment Branch Condition size for sign_out is too high. [<7, 44, 8> 45.27/17]
  def sign_out ...
  ^^^^^^^^^^^^
app/controllers/events_controller.rb:10:3: C: Metrics/MethodLength: Method has too many lines. [27/10]
  def sign_out ...
  ^^^^^^^^^^^^
app/controllers/events_controller.rb:34:5: C: Style/GuardClause: Use a guard clause (return unless @sign_in.save) instead of wrapping the code inside a conditional expression.
    if @sign_in.save
    ^^
app/controllers/sign_ins_controller.rb:9:3: C: Metrics/AbcSize: Assignment Branch Condition size for create is too high. [<6, 39, 8> 40.26/17]
  def create ...
  ^^^^^^^^^^
app/controllers/sign_ins_controller.rb:9:3: C: Metrics/MethodLength: Method has too many lines. [19/10]
  def create ...
  ^^^^^^^^^^
app/controllers/sign_ups_controller.rb:10:3: C: Metrics/AbcSize: Assignment Branch Condition size for create is too high. [<4, 27, 6> 27.95/17]
  def create ...
  ^^^^^^^^^^
app/controllers/sign_ups_controller.rb:10:3: C: Metrics/MethodLength: Method has too many lines. [13/10]
  def create ...
  ^^^^^^^^^^
spec/controllers/api/v1/api_spec.rb:5:1: C: Metrics/BlockLength: Block has too many lines. [40/25]
RSpec.describe Api::V1::EventsController, type: :api do ...
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
spec/homepage_spec.rb:6:1: C: Metrics/BlockLength: Block has too many lines. [376/25]
RSpec.describe 'events', type: :system do ...
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
spec/homepage_spec.rb:7:3: C: Metrics/BlockLength: Block has too many lines. [374/25]
  describe 'homepage' do ...
  ^^^^^^^^^^^^^^^^^^^^^^^^
spec/homepage_spec.rb:277:5: C: Metrics/BlockLength: Block has too many lines. [39/25]
    it 'sign in to an event and sign out of another' do ...
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
spec/homepage_spec.rb:348:5: C: Metrics/BlockLength: Block has too many lines. [26/25]
    it 'sign in twice stay on page' do ...
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
spec/models/event_spec.rb:5:1: C: Metrics/BlockLength: Block has too many lines. [34/25]
RSpec.describe Event, type: :model do ...
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

12.8 Coding Standard

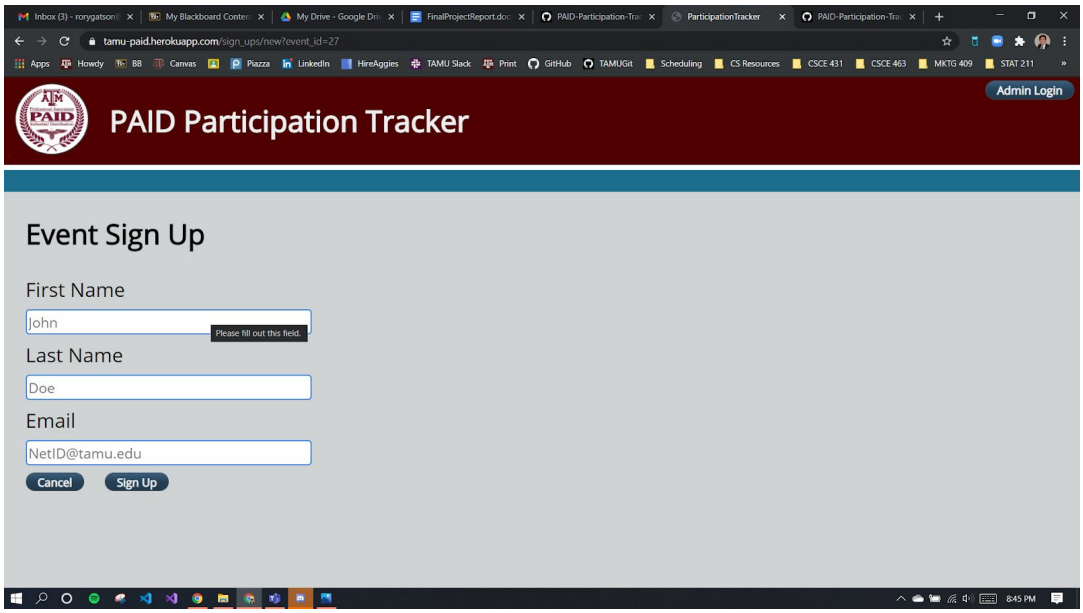


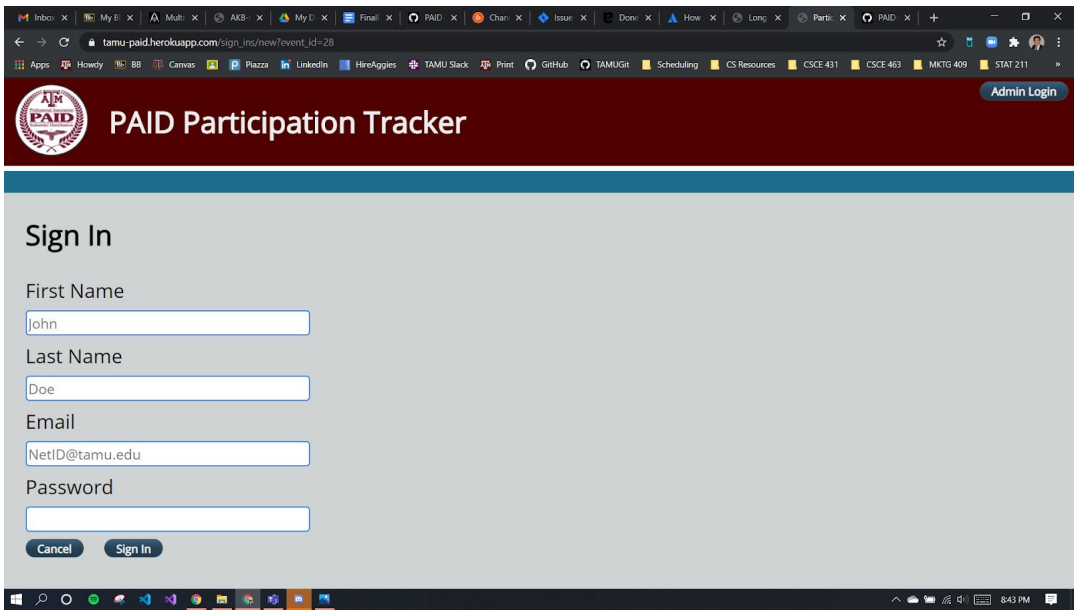
12.9 Usability: Mockups and Screenshots of Actual Forms

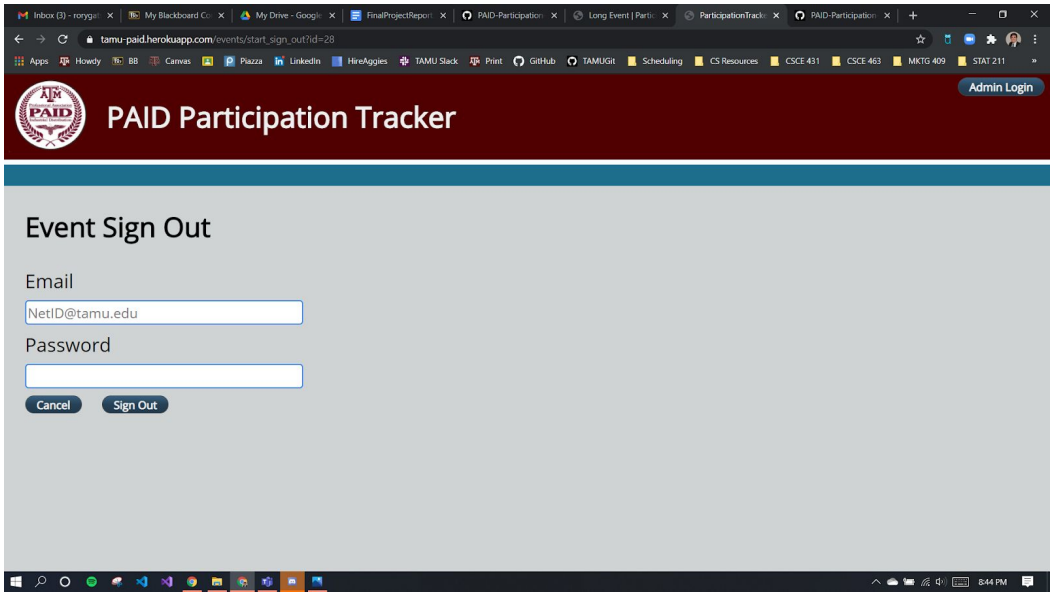
Homepage


PAID Participation Tracker
Admin Login
How to Use This Site
Current and Future Events
Name
Description
Date
Expected Time
Points
Forms
Long Event
test
11/03/2020 at 03:03 AM
1666666 hours 39 minutes
1
Sign In
Sign Out
Test Spring Event for Member Point Tracker
desciption
01/06/2021 at 02:02 AM
20 minutes
14
Sign Up

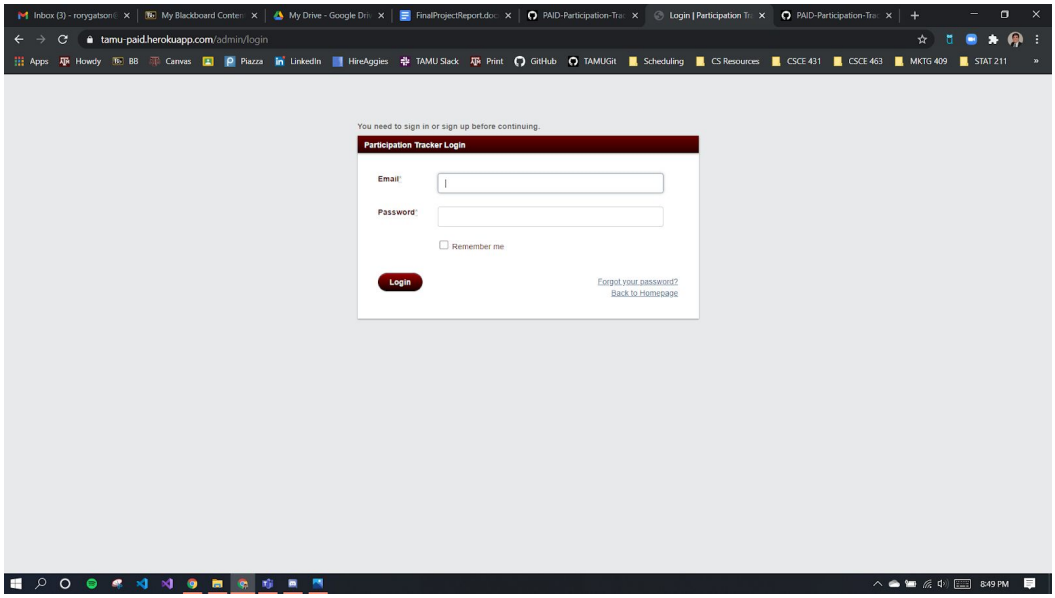
User Forms


tamu-paid.herokuapp.com/sign_ups/new?event_id=27
PAID Participation Tracker
Admin Login
Event Sign Up
First Name
John
Please fill out this field.
Last Name
Doe
Email
NetID@tamu.edu
Cancel
Sign Up

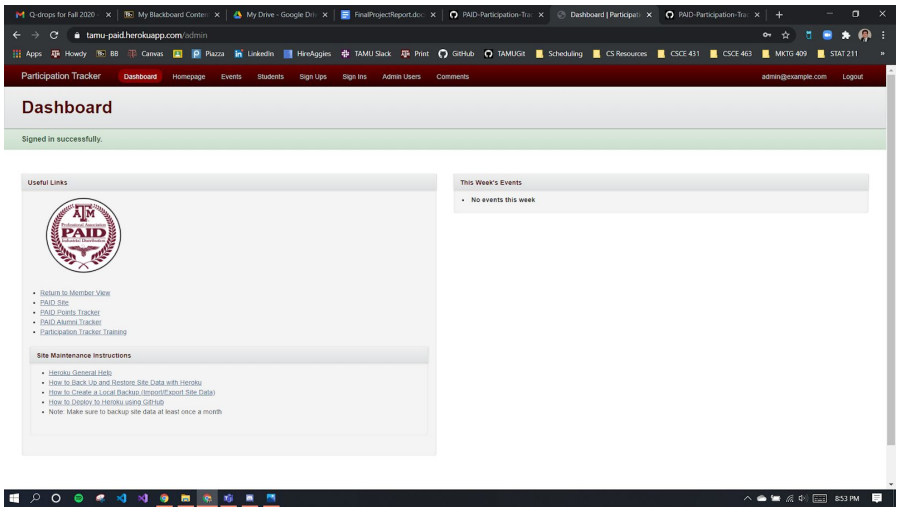

tamu-paid.herokuapp.com/sign_ins/new?event_id=28
PAID Participation Tracker
Admin Login
Sign In
First Name
John
Last Name
Doe
Email
NetID@tamu.edu
Password
Cancel
Sign In

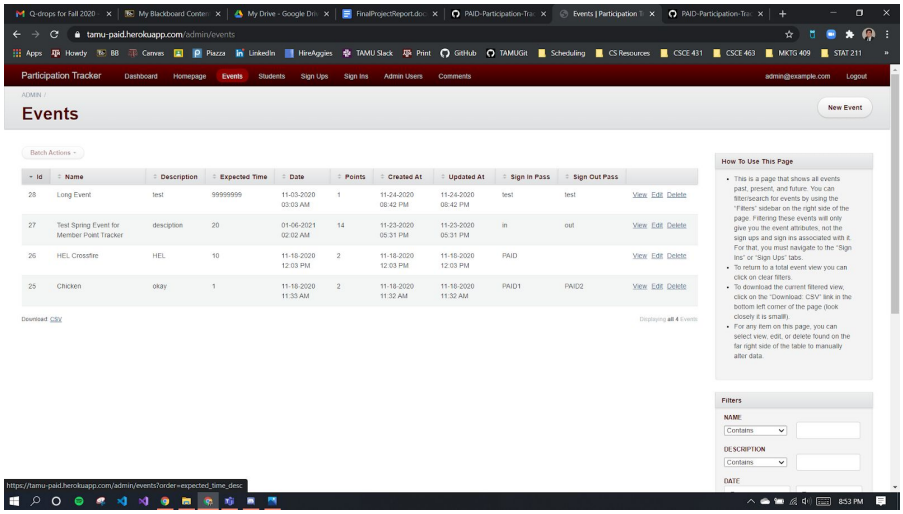


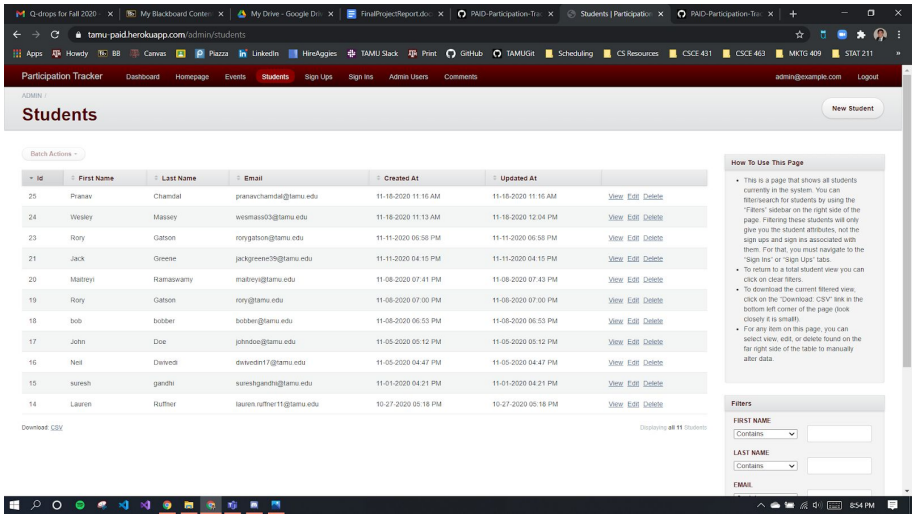
Administrator Login Page

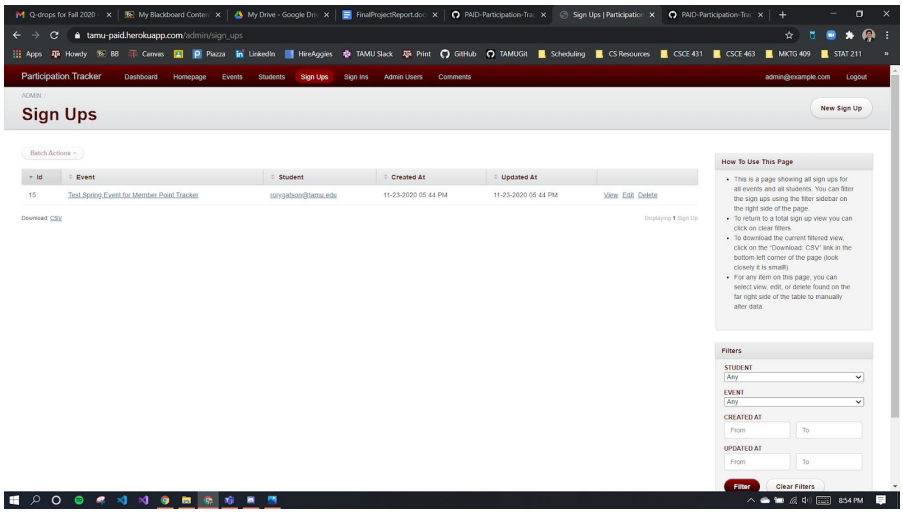


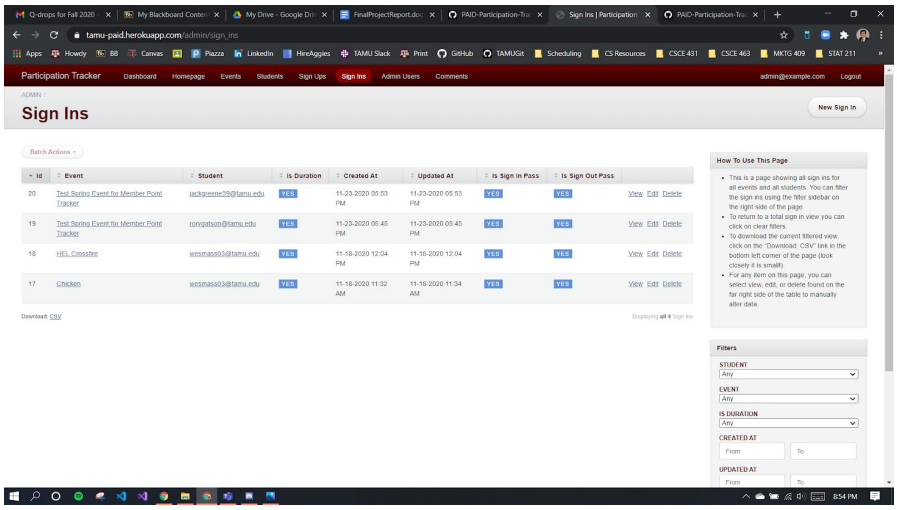
Admin Model Pages

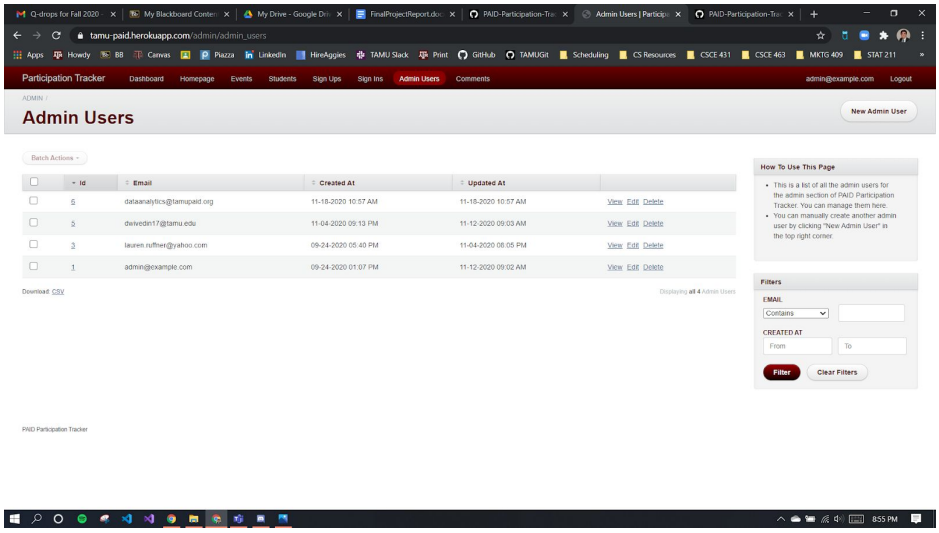







tamu-paid.herokuapp.com/admin/sign_ups
Participation Tracker
Dashboard
Homepage
Events
Students
Sign Ups
Sign Ins
Admin Users
Comments
admin@example.com
Logout
Sign Ups
New Sign Up
Batch Actions
Id
Event
Student
Created At
Updated At
15
Test Spring Event for Member Point Tracker
rongatson@tamu.edu
11-23-2020 05:44 PM
11-23-2020 05:44 PM
View Edit Delete
Download CSV
How To Use This Page
This is a page showing all sign ups for all events and all students. You can filter the sign ups using the filter sidebar on the right side of the page.
To return to a total sign up view you can click on clear filters.
To download the current filtered view, click on the "Download: CSV" link in the bottom left corner of the page (look closely it is small!).
For any item on this page, you can select view, edit, or delete found on the far right side of the table to manually alter data.
Filters
STUDENT
Any
EVENT
Any
CREATED AT
From
To
UPDATED AT
From
To
Filter
Clear Filters
8:54 PM

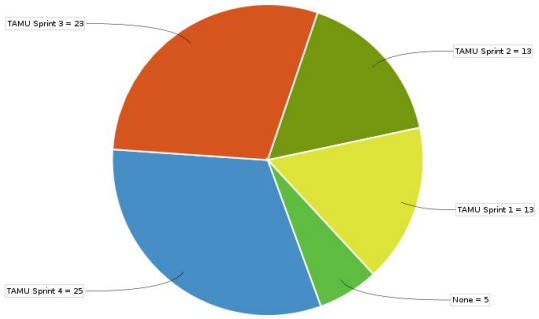

tamu-paid.herokuapp.com/admin/sign_ins
Participation Tracker
Dashboard
Homepage
Events
Students
Sign Ups
Sign Ins
Admin Users
Comments
admin@example.com
Logout
Sign Ins
New Sign In
Batch Actions
Id
Event
Student
Is Duration
Created At
Updated At
Is Sign In Pass
Is Sign Out Pass
20
Test Spring Event for Member Point Tracker
jackgreene35@tamu.edu
YES
11-23-2020 05:53 PM
11-23-2020 05:53 PM
YES
YES
View Edit Delete
19
Test Spring Event for Member Point Tracker
rongatson@tamu.edu
YES
11-23-2020 05:45 PM
11-23-2020 05:45 PM
YES
YES
View Edit Delete
18
HEL Crossfire
wesmass03@tamu.edu
YES
11-18-2020 12:04 PM
11-18-2020 12:04 PM
YES
YES
View Edit Delete
17
Chicken
wesmass03@tamu.edu
YES
11-18-2020 11:32 AM
11-18-2020 11:34 AM
YES
YES
View Edit Delete
Download CSV
How To Use This Page
This is a page showing all sign ins for all events and all students. You can filter the sign ins using the filter sidebar on the right side of the page.
To return to a total sign in view you can click on clear filters.
To download the current filtered view, click on the "Download: CSV" link in the bottom left corner of the page (look closely it is small!).
For any item on this page, you can select view, edit, or delete found on the far right side of the table to manually alter data.
Filters
STUDENT
Any
EVENT
Any
IS DURATION
Any
CREATED AT
From
To
UPDATED AT
From
To
8:54 PM


tamu-paid.herokuapp.com/admin/admin_users
Participation Tracker
Dashboard
Homepage
Events
Students
Sign Ups
Sign Ins
Admin Users
Comments
admin@example.com
Logout
Admin Users
New Admin User
Batch Actions
Id
Email
Created At
Updated At
6
dataanalytics@tamupaid.org
11-18-2020 10:57 AM
11-18-2020 10:57 AM
View Edit Delete
5
dwivedin17@tamu.edu
11-04-2020 09:13 PM
11-12-2020 09:03 AM
View Edit Delete
3
lauren.ruffner@yahoo.com
09-24-2020 05:40 PM
11-04-2020 08:05 PM
View Edit Delete
1
admin@example.com
09-24-2020 01:07 PM
11-12-2020 09:02 AM
View Edit Delete
Download CSV
How To Use This Page
This is a list of all the admin users for the admin section of PAID Participation Tracker. You can manage them here.
You can manually create another admin user by clicking "New Admin User" in the top right corner.
Filters
EMAIL
Contains
CREATED AT
From
To
Filter
Clear Filters
PAID Participation Tracker
8:55 PM

12.10 Monitoring and Control through Jira and Other Reports

Project: PAID - Participation Tracker (Sprint)

Chart

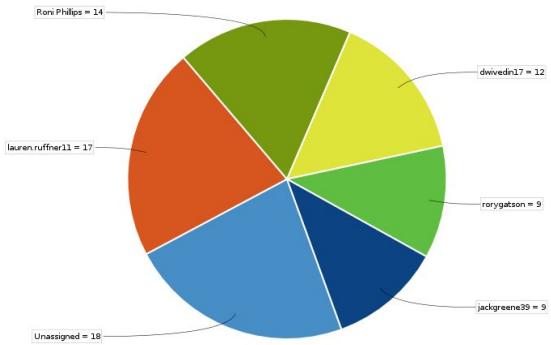


Data Table

	Issues	%
TAMU Sprint 4	25	31%
TAMU Sprint 3	23	29%
TAMU Sprint 2	13	16%
TAMU Sprint 1	13	16%
None	5	6%

Project: PAID - Participation Tracker (Assignee)

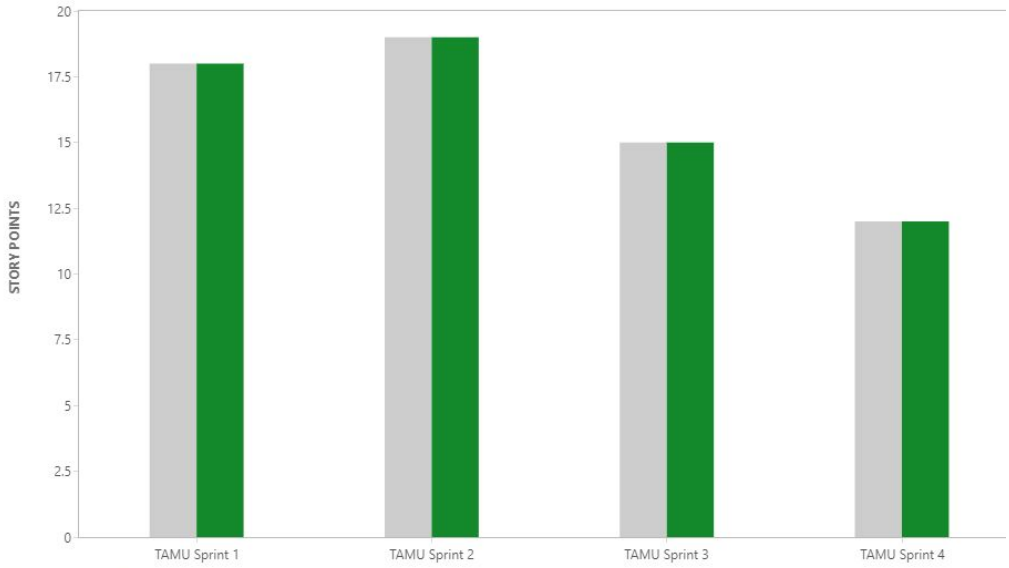
Chart



Data Table

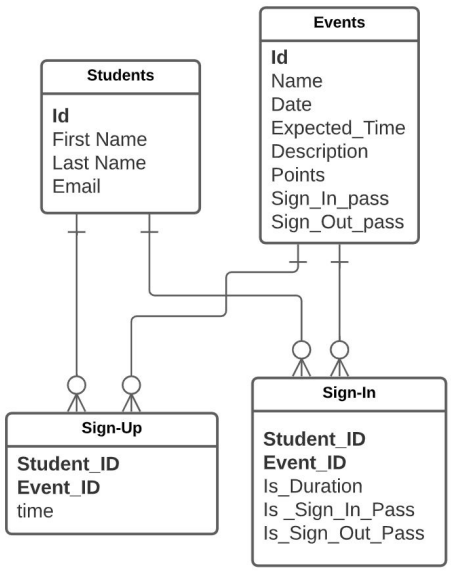
	Issues	%
Unassigned	18	22%
lauren.ruffner11	17	21%
Roni Phillips	14	17%
dwivedin17	12	15%
rorygatson	9	11%
jackgreene39	9	11%

Velocity Chart



Sprint	Commitment	Completed
TAMU Sprint 1	18	18
TAMU Sprint 2	19	19
TAMU Sprint 3	15	15
TAMU Sprint 4	12	12

12.11 Data Design



Relationship Explanation:
- Each Event entry can have 0 or more Sign Up entries but each Sign Up entry is connected to only one Event entry
- Each Student entry can have 0 or more Sign Up entries but each Sign Up entry is connected to only one Student entry
- Each Event entry entry can have 0 or more Sign In entries but each Sign In entry is connected to only one Event entry
- Each Student entry can have 0 or more Sign In entries but each Sign In entry is connected to only one Student entry

12.12 Risk Plan

Risk	Problem in %	Impact	RMMM Plan	Status
Customer discovers many defects	20%	High	Monitoring: Create tests during every spint Mitigation: Frequently do code reviews and test team code.	On Schedule, low probability of happening due to testing.
Release of Sprint 4 App will be delayed	10%	High	Monitoring: Conduct Daily Stand-Up Meetings Mitigation: Frequently communicate (SCRUM meetings) and encourage team members to ask for help if needed	On Schedule, low probability of happening.
Hours not logged properly	30%	Medium	Monitoring: Conduct Daily Stand-Up Meetings Mitigation: Frequently communicate (SCRUM meetings) and encourage team members to ask for help if needed	On Schedule
Team Member falls behind	10%	High	Monitoring: Conduct Daily Stand-Up Meetings Mitigation: Frequently communicate (SCRUM meetings) and encourage team members to ask for help if needed	On Schedule, no issues thus far
PAID Member signs in when not in attendance	10%	Medium	Monitoring: Create password validation for sign-ins Mitigation: Ensure that the incorrect event password does not work	Low probability due to addition of password for signing in and out of events
User creates multiple student models for the same person	20%	High	Monitoring: Admin checks user log Mitigation: Create user only based on email and edit otherwise	Only possible if student inputs incorrect email and if this happens an admin can easily go and delete the email.