PGUA Final Report

Project Summary

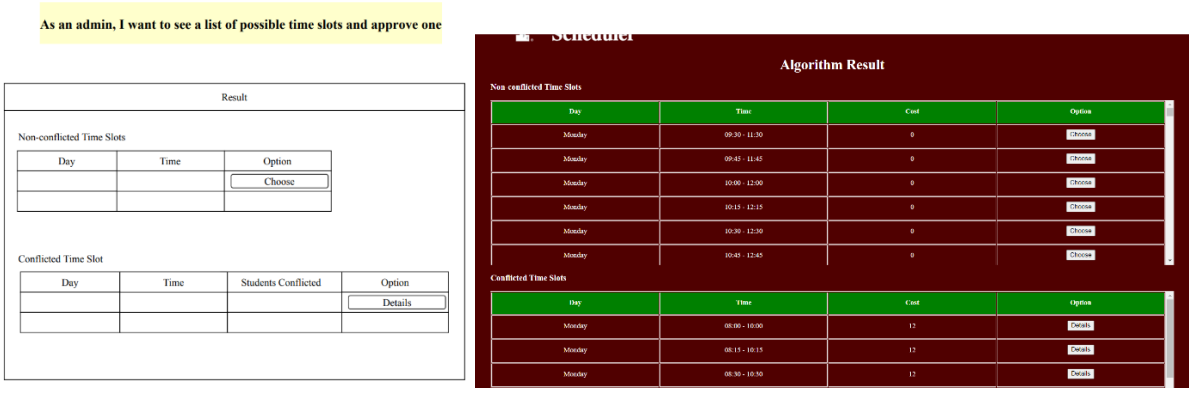
The stakeholders for this project are Seth Sullivan and the faculty/staff for the Zachary Leadership Program as well as the students in that program. They requested an application that would help them find a class time for their groups to meet during the semester. They currently search for this time slot manually, which is a long and frustrating process.

To address this need, our project allows administrators to create and open a new term, which allows the students to enter their schedules for the upcoming semester. The administrators can see which students have submitted schedules, and once they are ready, they can run an algorithm that returns a list of time slots for which no student has a conflicting class. The students can also input alternative schedules to help increase the likelihood of finding an available time slot.

User Stories

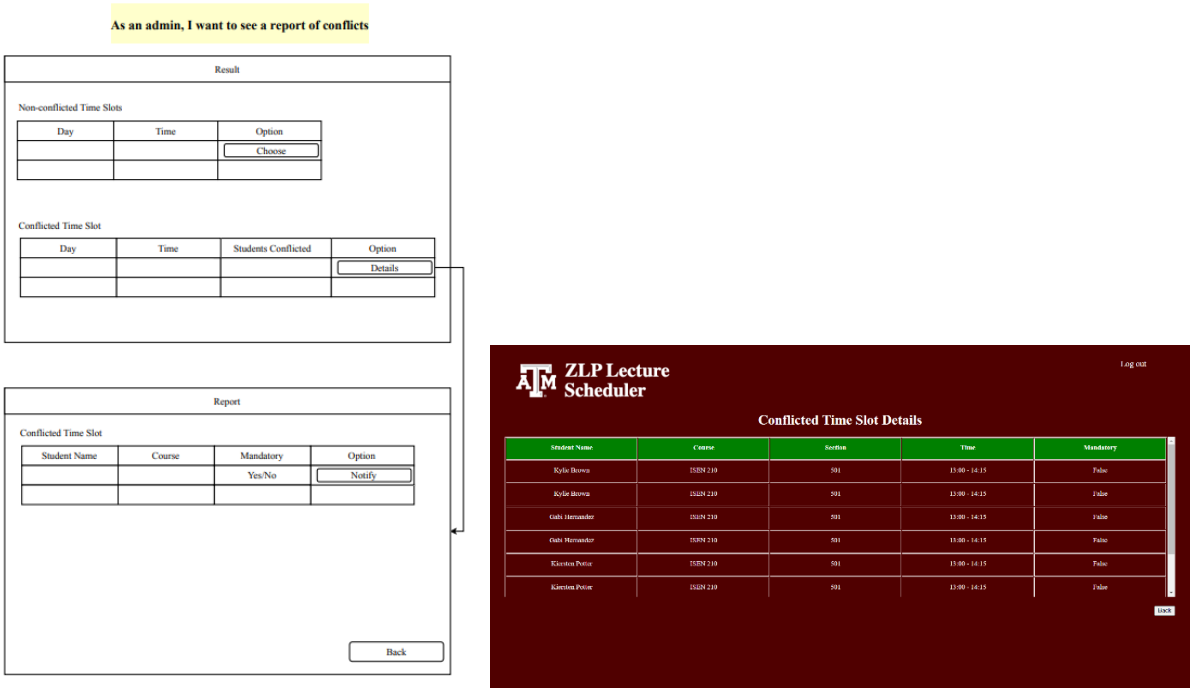
1. As an administrator so that I can determine the time slot to choose I want to see the results of the algorithm.

   This user story was 1 point because it mainly involved only displaying data. It is implemented, including passing Cucumber scenarios.



2. As an administrator so that I can pick an appropriate time slot I want to see a report of conflicts for each suggested time slot.

   This user story was 2 points because it would potentially involve some logic as well as displaying data. It is implemented, including passing Cucumber scenarios.



3. As an administrator so that I can determine the available time slots I want to be able to run the algorithm.

   This user story was 3 points because it involved determining the best algorithm to use and implementing that algorithm. It is implemented, along with passing rspec tests.

4. As an administrator so that I can effectively use the application I want the algorithm and the results page to be accessible from the current term page.

   This user story covers the parts of the above user stories that were not finished in a single iteration. This user story was 2 points because it involved connecting parts of the application. It is implemented, including passing Cucumber scenarios. It also spanned a few iterations, so new user stories were made for the parts left at the end of the iteration.

   4a. As an administrator so that I can get accurate results I want the conflict details page to show information from the database.

      This user story was 1 point since it required only UI changes. It is implemented.

   4b. As an administrator so that I can see the results page quickly I want the algorithm to be optimized.

      This user story was 2 points because it involved optimization of the algorithm. It is implemented, and the algorithm has a run time acceptable to the customer.

   4c. As an administrator so that I can view the results of a previously run scheduling algorithm I want to be able to visit the results page without the algorithm being run a second time.

      This user story was 2 points as it involved link function editing. It is implemented.

4d. As an administrator so that I can be sure the application is working correctly I want there to be testing for the results pages.

This user story was for testing, so it was 0 points. It is implemented. Cucumber scenarios were written and are all passing. This user story spanned multiple iterations, so a new user story was made for the parts done in a separate iteration.

4e. As a user so that I know the application is working correctly I want the buttons on the results page to be tested.

This user story was for testing, so it was 0 points. It is implemented. Cucumber scenarios were written and are all passing.

5. As a user so that I can use the system I want to be able to create an account.

   This user story was for testing legacy code, so it was 0 points. It is implemented. Cucumber scenarios were written and are all passing. Several bugs in the legacy code were found, fixed, and tested. One of these bugs was given its own user story:

   4a. As a student so that I can properly use the application I want to be able to register only once.

   This user story was for a bug, so it was 0 points. It is implemented.

6. As a student so that I can submit my schedules I want to be able to login to the system.

   This user story was for testing legacy code, so it was 0 points. It is implemented. Cucumber scenarios were written and are all passing.

7. As an administrator so that I can run the algorithm I want to be able to login to the system.

   This user story was for testing legacy code, so it was 0 points. It is implemented. Cucumber scenarios were written and are all passing.

8. As a user so that I can login to the system I want to be able to recover my password.

   This user story was for testing legacy code, so it was 0 points. It is implemented. Cucumber scenarios were written and are all passing. This functionality was not initially working in the legacy code, but we have fixed this issue. A user story was created for the email-sending portion of fixing this section of the legacy code:

   7a. As a student so that I can get into my account I want to receive an email when I have forgotten my password.

   This user story was for fixing a bug, so it was 0 points. It is implemented and the application can now send password reset emails.

9. As a student so that the algorithm can be run I want to submit my schedules.

   This user story was for legacy code tests, so it was 0 points. It is implemented. Cucumber scenarios were written and are passing. Some bugs were discovered and fixed.

10. As a student so that I can see the schedules I've input into the system I want to be able to view my schedules.

This user story was for testing legacy code, so it was 0 points. It is implemented. Cucumber scenarios were written and are all passing.

11. As an administrator so that I can see which cohorts are in the current term I want to be able to view the current term.

    This user story was for testing legacy code, so it was 0 points. It is implemented. This functionality is achieved by an administrator login, so that user story covers this one too.

12. As an administrator so that I can run the algorithm I want to be able to create and open a new term.

    This user story was for legacy code tests, so it was 0 points. It is implemented. Cucumber scenarios were written and are passing. We found a bug, so a new user story was made:

    11a. As an administrator so that I can create a new term I want to be redirected to the form page when errors are detected.

        This user story was for a bug, so it was 0 points. It is implemented and the bug has been fixed. Tests were also written for this bug fix.

13. As a student so that a time can be found I want to input alternate schedules.

    This user story was for testing legacy code, so it was 0 points. It is implemented. The functionality described in this user story is the same as the functionality in the user story about students submitting schedules, so those tests covered this functionality as well.

14. As a student so that I can graduate on time I want the algorithm to account for any of my classes that are mandatory.

    This user story was 3 points because it could involve changing the structure of the algorithm. It is implemented. Mandatory classes have a higher cost than ones that are not.

15. As an administrator so that I can spend time with my family I want the algorithm to account for my meeting time preferences.

    This user story was 2 points because it involved making slight modifications to the algorithm. It is implemented. The cost of a time slot increases the longer after 5:00 pm it ends. Rspec tests were added for this code as well.

16. As an administrator so that only active administrators have access to the application I want to be able to add and delete other administrators.

    This user story was for legacy code tests, so it was 0 points. It is implemented. Cucumber scenarios were written and are passing. We found a bug, so a new user story was made:

    16a. As an administrator so that I can cancel editing an administrator I want the cancel button to link back to the manage administrators page.

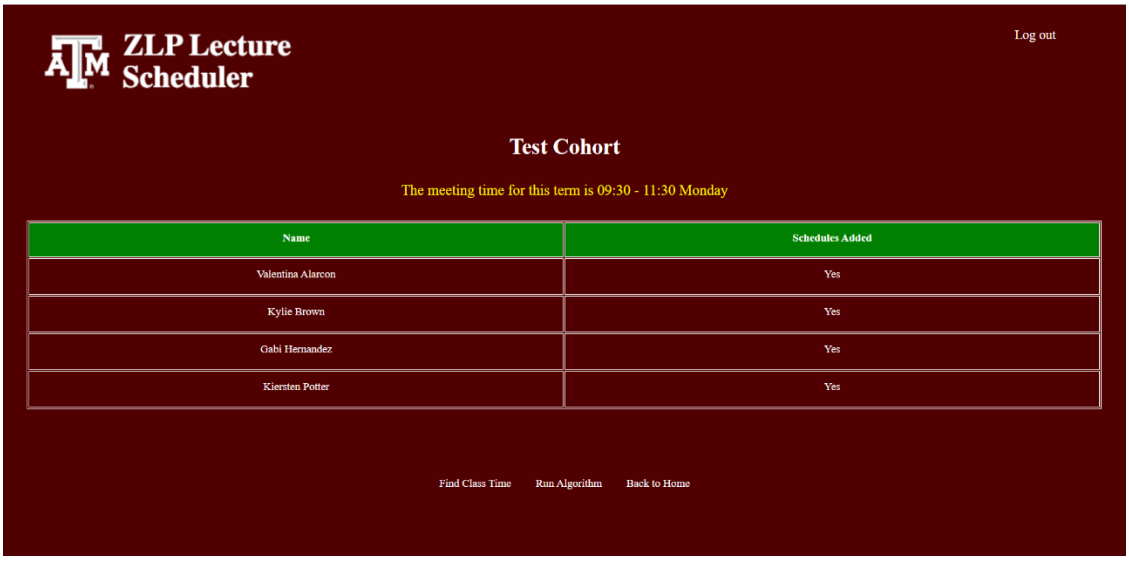
        This user story was for a bug, so it was 0 points. It is implemented and the bug has been fixed. Cucumber scenarios were also written for this bug fix.

17. As a student so that I know when to input my schedules I want to only be able to input my schedules when my cohort is part of an open term.

This user story was for a bug, so it was 0 points. It is implemented and the bug has been fixed. Existing Cucumber scenarios were also modified to test this bug.

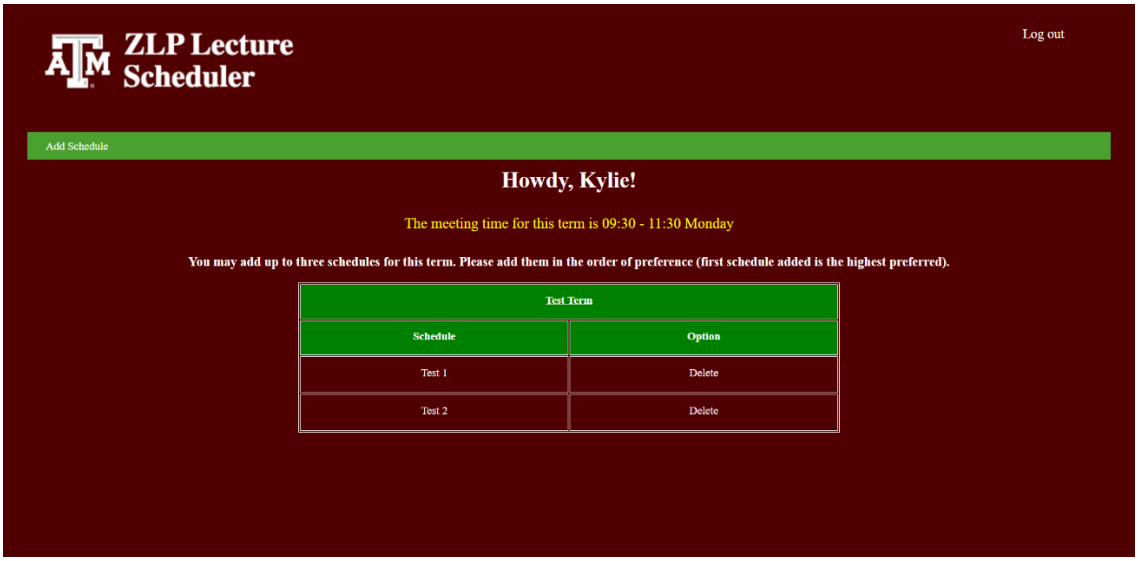
18. As an administrator so that I can inform the students of the picked time slot I want to be able to choose one of the time slots suggested by the algorithm.

    This user story was 1 point because it consisted of saving and displaying a single piece of data. It is implemented. Cucumber scenarios were written and are passing.



19. As a student so that I can know which classes to take I want to see the time slot picked by the algorithm/administrator.

    This user story was 1 point because it consisted of only displaying data. It is implemented. Cucumber scenarios were written and are passing.



20. As a user so that I can return to the homepage I want the logo to link back to the landing page for my role.

This user story was for a bug, so it was 0 points. It is implemented and the bug has been fixed. Cucumber scenarios were created to test this bug, and they are all passing.

21. As an administrator so that I can know when to run the algorithm I want to see which students have submitted their schedules.

    This user story was for testing legacy code, so it was 0 points. It is implemented. Cucumber scenarios were written, and they are all passing.

22. As an administrator so that I can account for changes in the cohorts I want to be able to create, delete, and edit both the cohorts and the students in the cohorts.

    This user story was for testing legacy code, so it was 0 points. It is implemented. Cucumber scenarios were written and are all passing. Several bugs were found and fixed.

23. As an administrator so that I can input the correct cohort file I want the application to tell me when I've selected an incorrect file type.

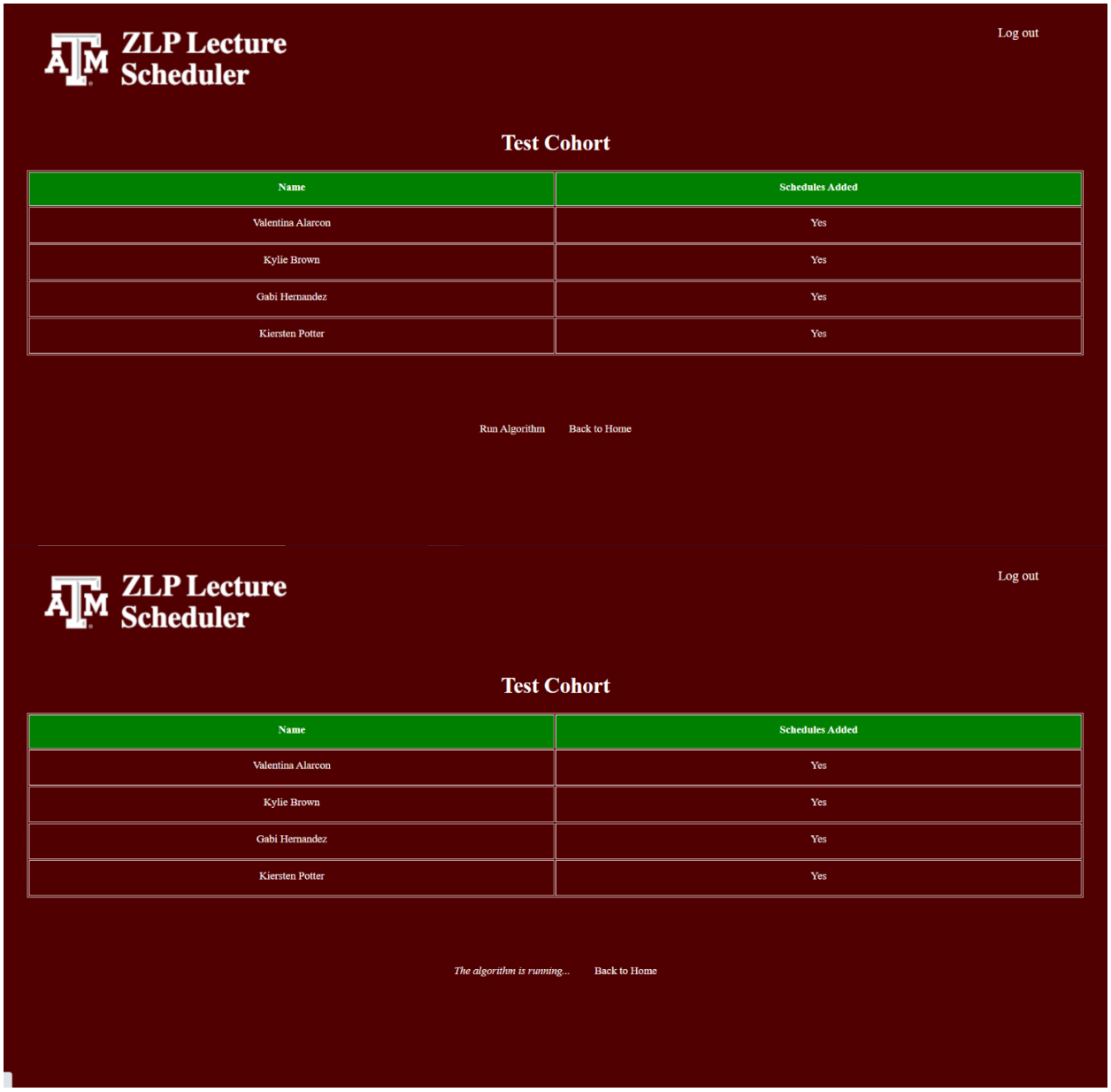
    This user story is for a bug, so it was 0 points. It is implemented. The bug has been fixed.

24. As an administrator so that I can verify that the application will work in the production environment I want the application to have been load tested.

    This user story was for testing, so it was 0 points. It is implemented. The testing for this user story was done manually.

25. As an administrator so that I don't run the algorithm unnecessarily I want the application to tell me when the algorithm is running.

    This user story was 2 points since it possibly involved JavaScript. It is implemented. The link for running the algorithm changes to let the user know the algorithm is running.



26. As an administrator so that I can verify the application is working correctly I want the entire workflow of the application to be tested.

    This user story was for testing, so it was 0 points. It is implemented. The testing for this user story was done manually.

27. As an administrator so that I can use the application in the future I want a user guide to be available.

    This user story was 1 point since it only involved describing how to use the application. It is implemented, and the user guide has been sent to the customer.

28. As a student so that I can know how to use the application in the future I want a user guide to be available.

This user story was 1 point since it only involved describing how to use the application. It is implemented, and the user guide has been sent to the customer.

29. As an administrator so that I can make informed decisions I want to see the class time of the courses that are in conflict with a given time slot.

    This user story was 1 point because it only involved displaying data. It is implemented.

30. As a student so that I can correct a mistake I want to edit my schedules.

    This user story was 1 point since it mostly involved UI changes. It was not implemented. This user story was low on our priority list, and we did not have time to implement it.

31. As a student so that I can graduate on time I want to denote which classes on my schedules are mandatory.

    This user story was 1 point because it involved collecting and saving data. It was not implemented because it was already implemented in the legacy code.

32. As an administrator so that I can spend time with my family I want to input meeting time preferences.

    This user story was 1 point since it mostly involved UI changes. It was not implemented. As time preferences were factored into the algorithm, the customer said that being able to change these parameters was optional, and we did not have time to implement this.

33. As a student so that I can know which classes to take I want to see which of my schedules resulted in the time slot picked by the algorithm/administrator.

    This user story was 1 point since it mostly involved UI changes. It was not implemented. This user story was not a priority, and we did not have time to implement it.

Legacy Code Discussion

When we first started looking at the code, we noticed that there wasn't much testing. So, we decided to add testing to the legacy code as a way to help us understand the existing code. As we discovered them, we fixed any bugs we found in the code. We also modified the code that loaded data from an external API to save to the database in batches to decrease the runtime of that step.

Team Roles

Minh Dang – Product Owner
Bethany Witemeyer – Scrum Master
Sanghyeon Lee – Developer
Shih-Chiang Wei – Developer
Austin Dauzat – Developer
Han-Yi Wang – Developer
Litong Zhang – Developer

These roles did not change over the course of the project.

Iteration Summaries

Iteration 0
This iteration we met with our customer and established the main need the application was serving. We also made a list of user stories and determined their priorities. We generated low fidelity mock-ups for some of our user stories. Because no software development took place this iteration, 0 points were completed.

Iteration 1
During this iteration we generated the algorithm that would find potential time slots. We also generated web pages to display the results. These components were not connected to each other or the legacy code at this point. We also increased the testing on the legacy code this iteration. We completed 6 points this iteration. We also completed 5 zero-point user stories this iteration.

Iteration 2
During this iteration we connected most of the pieces of the application developed in the previous iteration. We also updated the algorithm to account for mandatory classes and meeting time preferences. We continued to add testing to the legacy code and fixed bugs that we discovered in the existing code. We completed 7 points this iteration. We also completed 2 zero-point user stories this iteration.

Iteration 3
During this iteration we added the ability to chose a time slot as well as finished connecting the pieces developed in Iteration 1. We also optimized the algorithm and prevented the algorithm from running every time the results page was loaded. We continued to add testing to the legacy code and fixed any bug that we found. We also figured out how to set up an email server for password reset emails this iteration. We completed 7 points this iteration. We also completed 8 zero-point user stories this iteration.

Iteration 4
This iteration we focused on testing. We also added a notification about when the algorithm is running and created user guides for the customer. In addition to testing legacy code and the new code added in the previous iterations, we also did some manual load testing and workflow testing. We completed 5 points this iteration. We also completed 8 zero-point user stories this iteration.

Customer Meetings
September 23, 2020 at 2:00 pm
This was our initial customer meeting. We discussed what the application needed to do and the main use cases. We also talked with the customer about what to expect during the semester.

September 25, 2020 at 10:30 am
This meeting was with our customer's assistant and a member of the previous team. The existing code was demoed for us, and we were able to ask questions about it.

October 13, 2020 at 8:30 am
At this meeting, we demoed the unconnected pieces of the application that we had developed.

October 27, 2020 at 11:30 am
This meeting, we were able to demo the application as mostly working with a couple buttons not yet connected. We were also given some feedback on the user interface from the customer.

November 10, 2020 at 1:00 pm
This meeting was with the entire administrative team of the Zachary Leadership Program. We demoed the full but not quite finished application, discussed what the customer wanted us to focus on for the final iteration, and talked a little about the process of handing off the application.

November 20, 2020 at 3:45 pm
This was our final meeting with the customer. We did a full demo of the application and discussed the details of handing off the application to the customer at the end of the semester.

BDD/TDD Process
For the most part, we were writing tests after the code was written. This was due in large part to the fact that most of our testing was for legacy code. This process worked for us, and even revealed some bugs that may not have been found otherwise. Writing the tests also gave us a better understanding of how the code worked. We focused mostly on Cucumber scenarios because those were easier to write. However, we did include rspec tests when we thought they were necessary.

We did run into some problems with getting tests to run properly on pages with JavaScript. We utilized several GEMs to help us in testing the application. We also did some manual testing as we developed the application and as we were getting ready to hand it off.

Configuration Management
Our configuration management approach was to create a branch for every new feature. Some branches were reused for similar features, but only after the first feature's code was merged. We used pull requests to allow for code review before the code was merged. We did not do any official spikes, but we did do some trial runs of the algorithm in a language we were more familiar with before writing the code in Ruby. We ended up with 41 development branches. We did not have any releases, unless one counts the tags placed on each iteration.

Heroku
For the most part, we did not have any issues in deploying our code to Heroku. The previous team left good instructions on how to get the code to deploy. We did have some problems with running the database migrations the first time we tried to deploy. Something had happened with the previous team's migration files that was causing errors we hadn't seen in the development environment. However, we were able to reproduce these errors and determine what was causing them. After those initial problems, though, deploying to Heroku went relatively smoothly.

AWS Cloud9 and GitHub
Initially, we ran into some issues getting the development environment set up. The previous team was using PostgreSQL in the development environment, and that took some work to figure out how to set up. We also ran into an issue with using all the space available on a Cloud9 instance,

but that could easily be solved by deleting files, increasing the size of the instance, or creating an entirely new instance. We didn't encounter many other problems while using AWS Cloud9.

When using GitHub, we did run into some issues with merge conflicts, but those were usually easy to resolve. There were a couple of times where we had some trouble with branching, but those issues were able to be solved quickly using Google. GitHub did not give us any major problems.

GEMs

In addition to cucumber-rails, capybara, rsepc-rails, and database_cleaner, we also used factory_bot_rails, faker, timecop, email_spec, and selenium-webdriver in the testing environment. Factory_bot_rails and faker allowed us to create fake data to use in our tests. Doing so helped us ensure that our tests were not dependent on any preloaded data. Timecop and email-spec were used for testing password reset emails. Timecop was used to test the expiration of a password reset email. Selenium-webdriver was vital in getting our tests to work with the JavaScript used in the application.

We also used the sendgrid-ruby GEM to connect with Twilio SendGrids API, which was used to send emails from within the application. This API was able to be integrated seamlessly with Heroku. Finally, we used both CodeClimate and SimpleCov to collect data about any smells we had in our code and the test coverage of our code.

Links

Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2467184
GitHub: https://github.com/tamu-zlp/zlp-scheduler
Heroku: https://zlp-scheduler.herokuapp.com/

Poster and Demo Video: https://youtu.be/BkUdBXmixfg