Zachry Leadership Program Scheduler

1 Summary of the Project

We had a legacy project to work on which had issues while selecting the available time. We first started by working on this bug. On the web app, there are two user roles: admin and student. In the current state, our project allows students to enter their schedules from a set of confirmed TAMU courses for the current term. The system would then automatically determine what time slots are available for all students. It also provides the best alternatives that only a few students need to change their schedules to find a time slot if there is no single time slot available.
Moreover, we discussed ways that can improve the project and implemented new user stories discussed during our meetings. For examples, the admin can enable or disable student entries when they want. There is also a new page that shows past student activities, and we also added a feature that let students edit their schedules. Besides the bug fixes and new features, we simplified the new term selection mechanism which now requires just one selection.

2 User Stories

2.1 Feature: Select a common time slot

As an admin, so that I can decide when to schedule the meeting, I want to see available common time slots of students.
This user story was for legacy code tests, so it was 0 points. It is implemented and the bug has been fixed. Cucumber scenarios were modified to test this bug.

2.2 Feature: Determine almost acceptable time slots

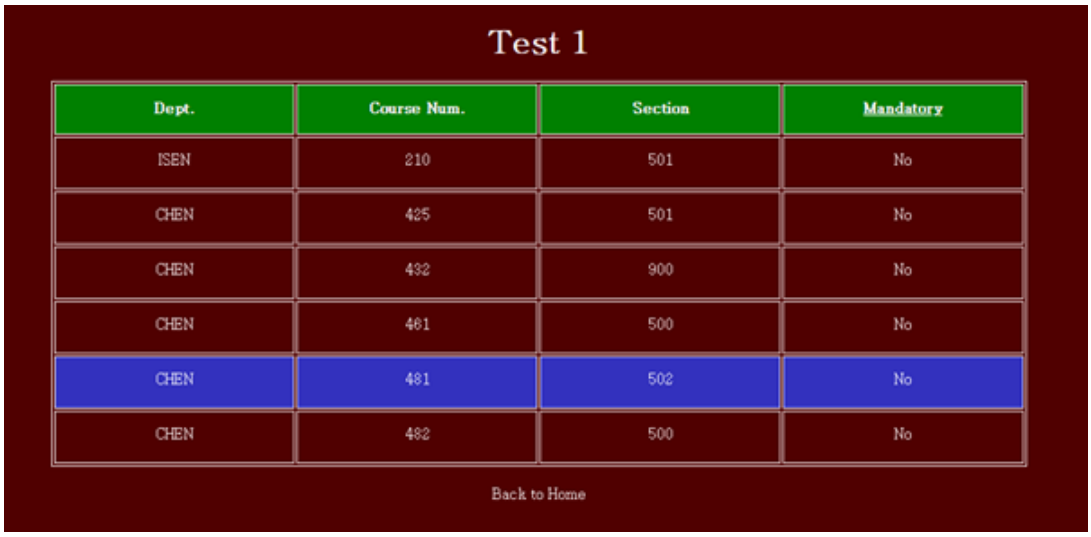
As an admin, so that I can ask students to modify their schedules, I want to see time slots that work for all but up to 4 students.
(I logged in with Lauren's id, as an administrator, to run the algorithm and find the time slots that work for all students and for all but up to 4 students)

2.3 Feature: Login to the system

As a student so that I can submit my schedules I want to be able to login to the system
(I logged in with Kiersten's id, and checked if I am able to login and submit my new schedules. Works totally fine.)

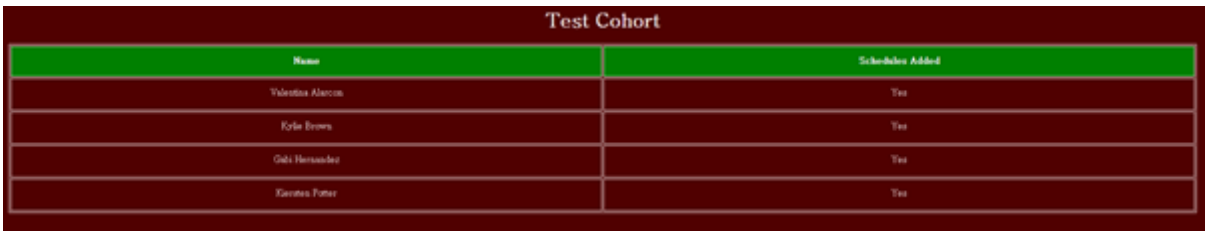
2.4 Feature: View my schedule

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.
(I logged in with Kiersten's id, and checked to see if I am able to see the schedule that he has inputted into the system. Works totally fine.)



2.5 Feature: See which students have submitted their schedules

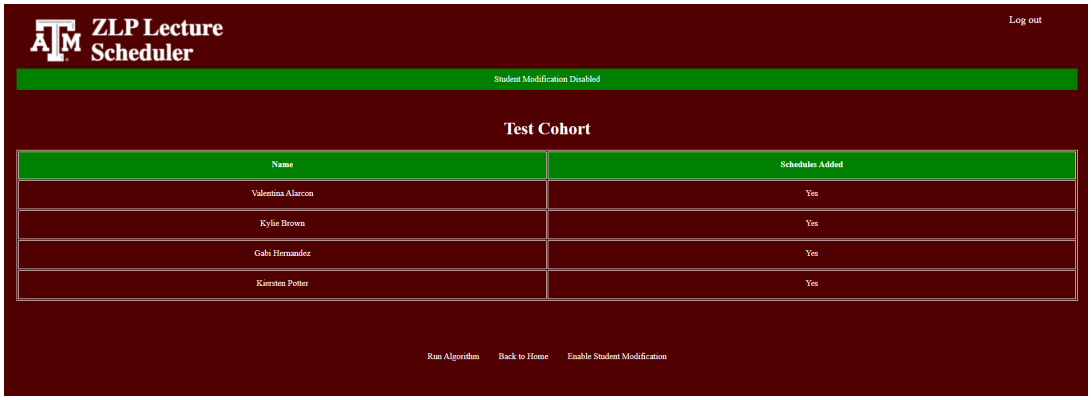
As an administrator so that I can know when to run the algorithm I want to see which students have submitted their schedules
(I logged in with Lauren's id, as an administrator, and went to the cohort page to see which students have submitted their schedules.)



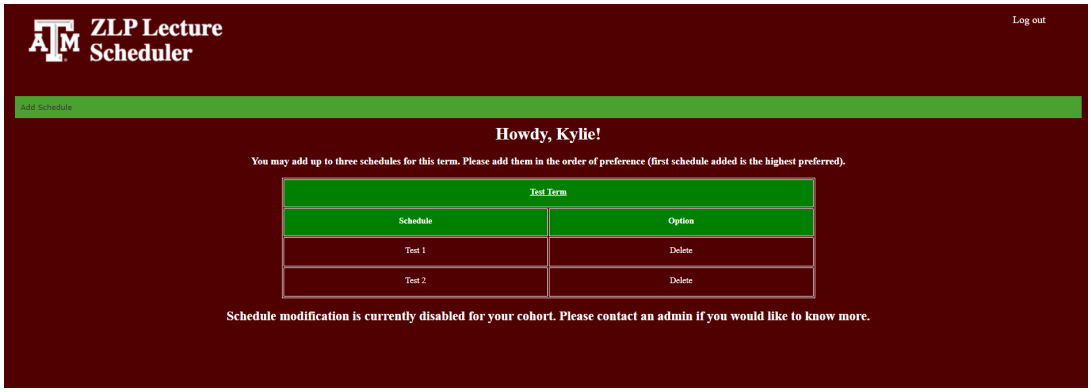
2.6 Feature: Admin to open or close student input

As an admin so that I can manage when students can enter their schedules I want to have a switch on the admin dashboard to open or close student input.

Admin view of pressing "Disable Student Modification" button with notification:



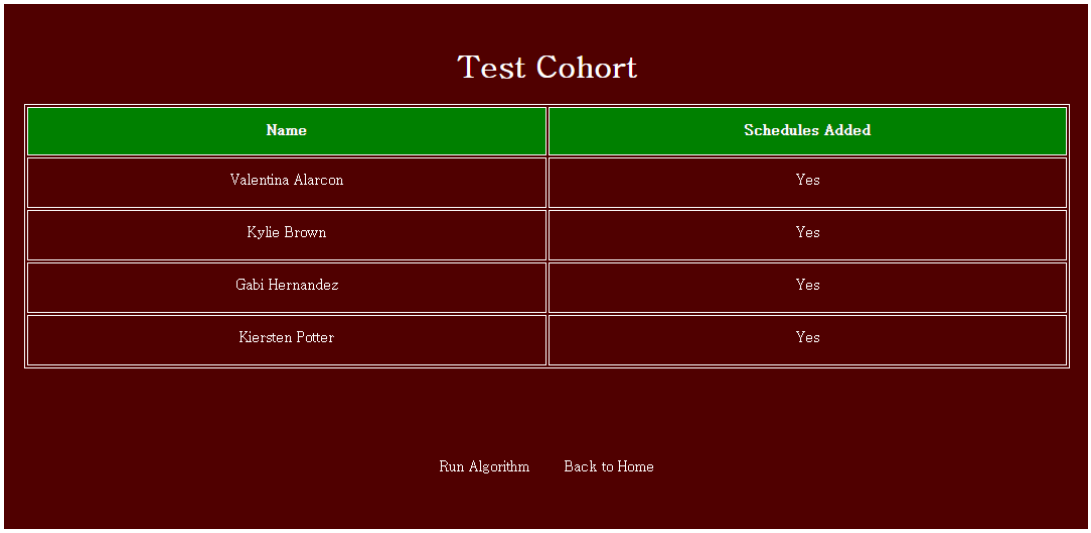
Student view where modifying schedules is disabled (cannot add or delete):



2.7 Feature: See the list of students and if they uploaded their schedules or not

As an admin so that I can warn students who didn't upload their schedules I want to see the list of students and if they uploaded their schedules or not.

- We implemented this as a new tab on the admin dashboard and then realised this feature already exists when you view a cohort.
- When an admin chooses a specific cohort, the admin can see the list of students within the cohort and if they uploaded their schedules or not.



2.8 Feature: See a notifications tab on the admin dashboard page with the latest student actions

As an admin so that I can keep track of student actions I want to see a notifications tab on the admin dashboard page with the latest student actions listed in it

- We implemented this user story as a new tab on the Admin dashboard named “Student Actions" that shows the list of student’s activities with respect to time.

2.9 Feature: Upload a list to the system with the information of students in cohorts

As an admin so that we can restrict who can register to the system I want to upload a list to the system with the information of students in cohorts

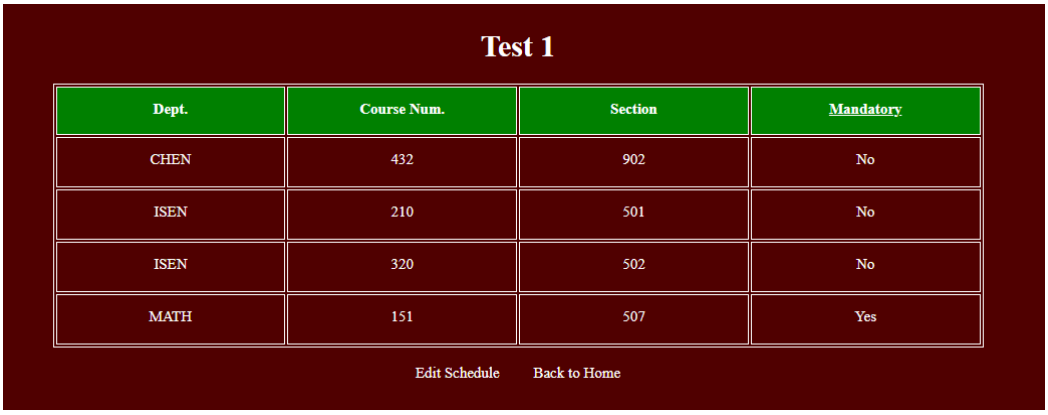
- This was already an implemented feature and we showed him the usage in our meeting.

- Student registration is actually just an action process. Without the admin uploading the student list, they cannot register to the system.

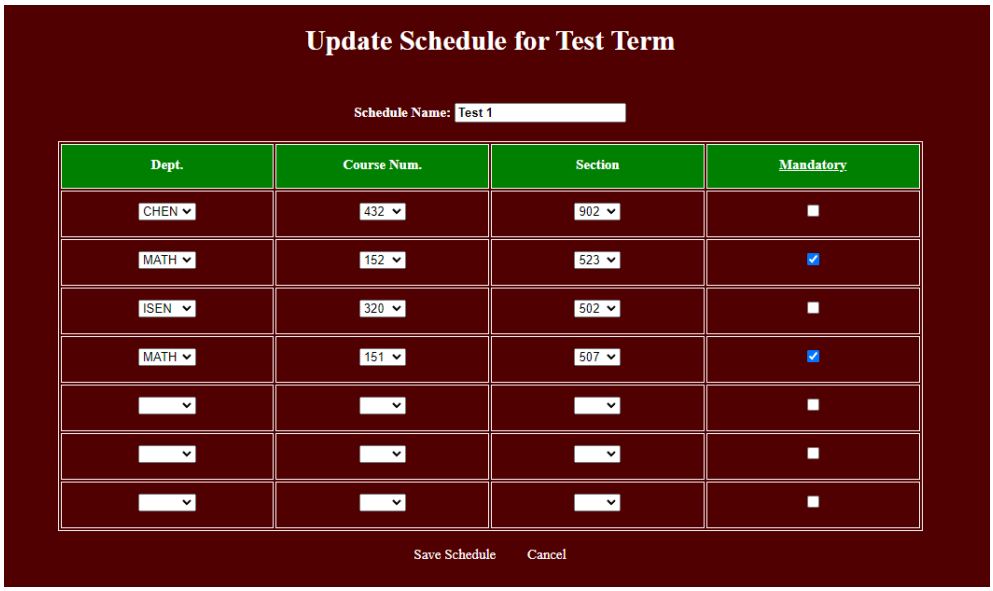
2.10 Feature: Upload a list to the system with the information of students in cohorts

As a student, so that I can make modifications on my schedule, I want to be able to edit my existing schedules.

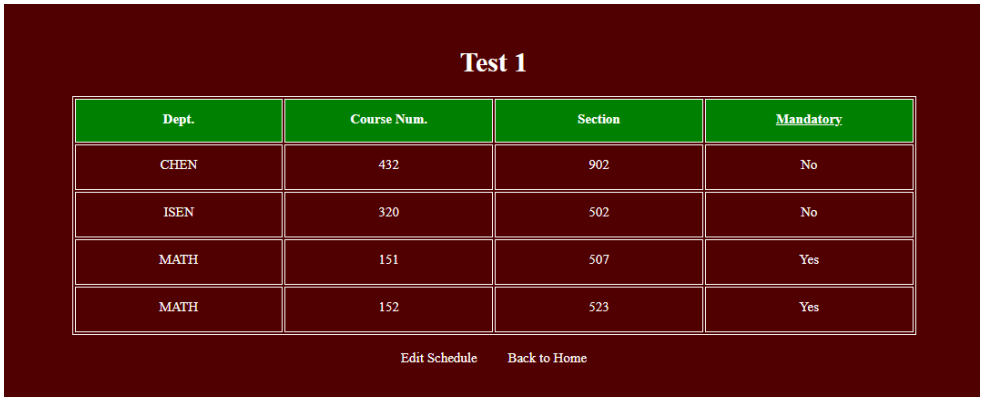
- When viewing schedules as a student, the schedule can now be edited by clicking the “Edit Schedule" button, if modification is enabled.



- This brings up the schedule form with the selected scheulde’s information already entered. Modifications can be made and either saved or discarded.



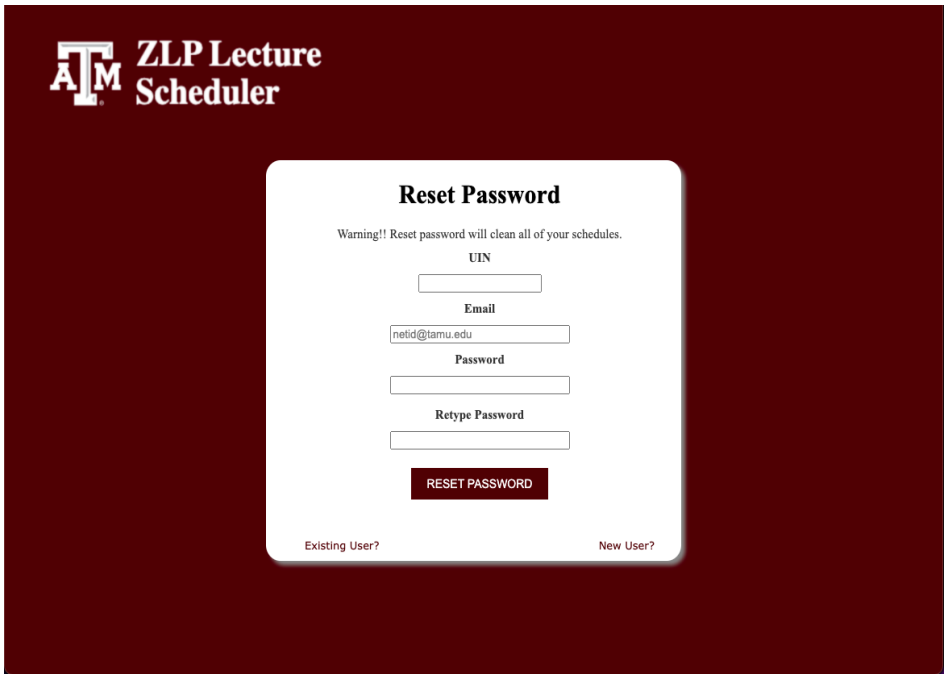
- The newly modified schedule is properly updated and visible to the student.



2.11 Feature: Login to the system

As a student I can reset my password, so I can login to the system

- Previously the students can reset passwords through an email link, but the email service is unstable under a free plan.
- We now let the students reset passwords using UIN and email, as shown below, and their existing schedules will be removed after that.

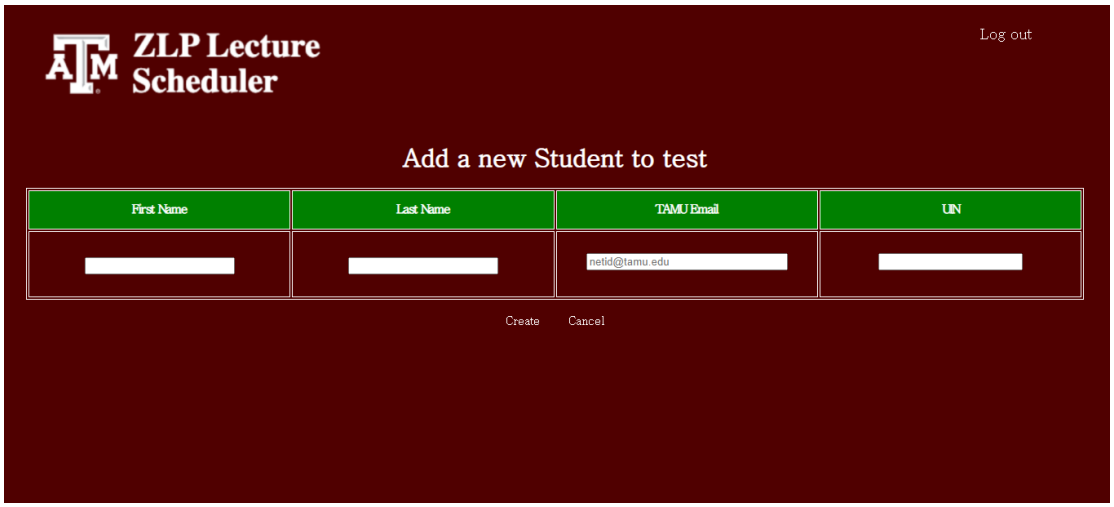


2.12 Feature: Edit the cohort

As an administrator so that I can add new students in specific cohorts, I want to be able to edit the cohort.

- When viewing Mange Cohorts page as an administrator, administrators can now add new students to specific cohorts by clicking on the 'Add student' button.

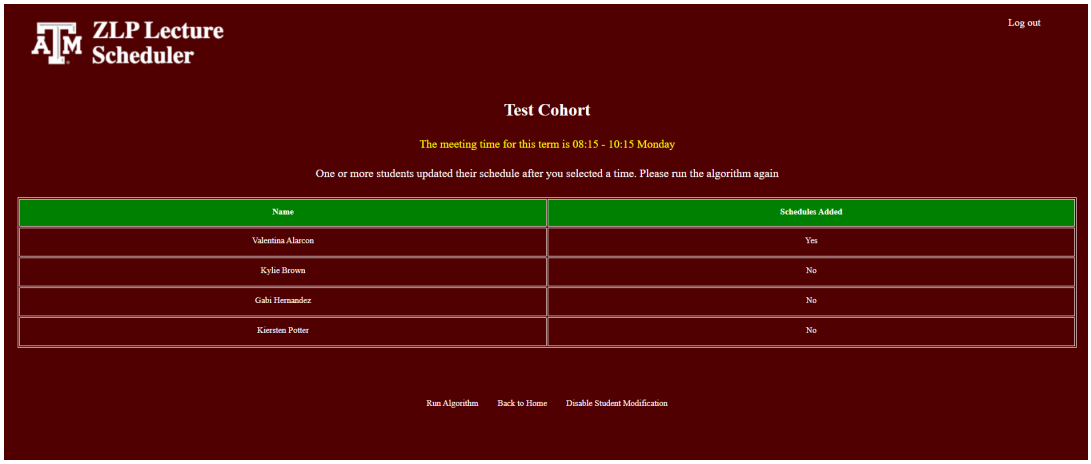
- When clicking on the 'Add student' button', there should be four input fields where a new student information should be added.



- The newly modified cohorts with students list is properly updated with a new added student information in it.

2.13 Feature: Get notification if students change their schedule(s)

As an administrator, I want to be notified if one or many students in a cohort change their schedule(s) after I select a time for that cohort.

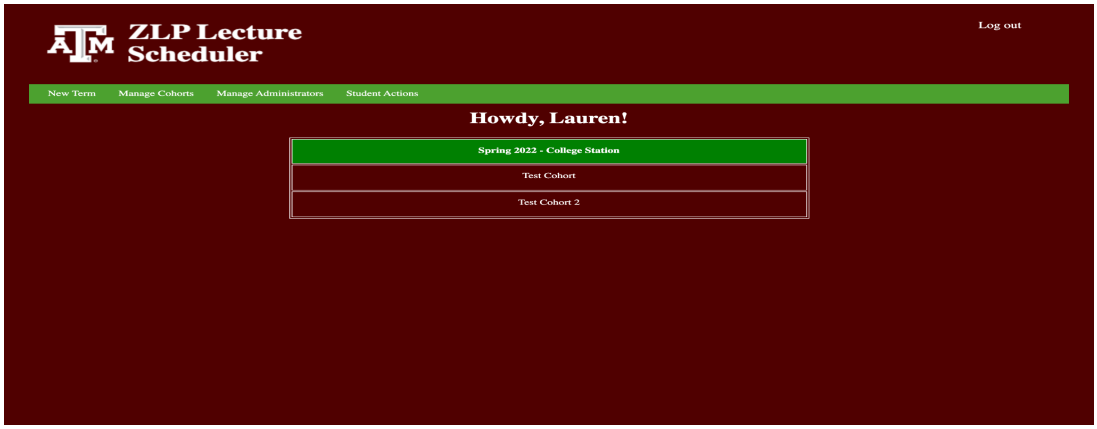


- After the admin selects a time slot for a cohort, if one or many students add, edit or delete their schedule(s), the admin sees the text “One or many students updated their schedule after you selected a time. Please run the algorithm again" when he views the cohort.

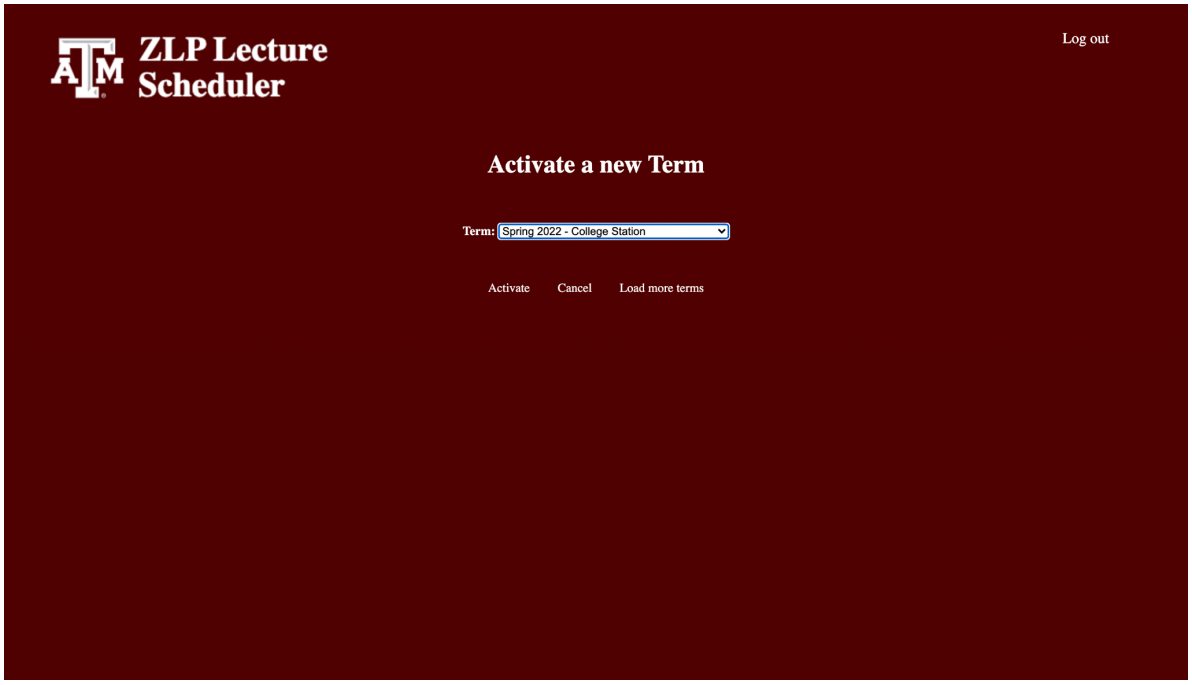
2.14 Feature: Select the current term in one step

As an admin, so that students can select their courses, I want to be able to select the current term in one step.

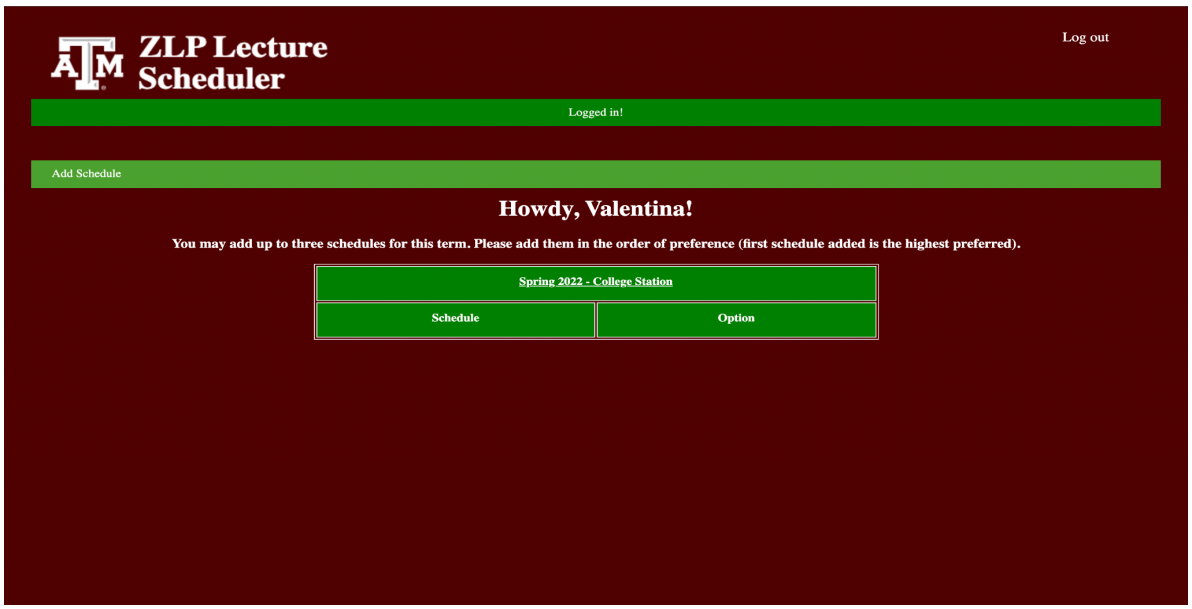
- Previously, The admin needed to select the cohorts for starting the new term. Afterwards, the admin needed to manually select the registration open date and close dates so that the students can start adding their schedules. But now the admin can do it just in one step.
- The admin starts at the home page, there is no ‘Open’ button now. Instead there is only ‘New term’.



- The admin goes to 'New Term' and selects the particular term of interest and activates it by clicking the 'Activate' button. No cohort selection is required.



- Thus the new term will be activated for all the cohorts. Students can log in and enter their schedules.



3 Team Roles

Scrum Master: Jordan Griffin
Product Owner: Md Messal Monem Miah
Developer: Fardeen Hasib Mozumder
Developer: Ali Furkan Budak
Developer: Adrita Anika
Developer: Ankur Nath
Developer: Jongyong Park
Developer: Han-Yi Wang

4 Iterations

4.1 Iteration 0

We met with our client for the first time and learned the requirements from him. We created our Pivotal Tracker project and created a general user story for fixing the bug that was occurring while running the available time finder algorithm. Since the legacy project was quite complex, this iteration we did not implement any new features but tried to understand how the existing project works.

4.2 Iteration 1

In this iteration we fixed some issues to have a working version of the legacy project. We deployed the web app into our own Heroku account to demonstrate the current version to the client. In our meeting we discussed and itemized our client's needs and created our first actual user stories. We also set up our test data to make the development process easier. In parallel, we continued to inspect the legacy code to have a better understanding.

4.3 Iteration 2

In this iteration we fixed the crucial bug that was causing an issue while running the algorithm. Besides, we implemented three user stories: Enabling/disabling student schedule modifications, a new Student Actions page that shows the list of student actions (creating, deleting, editing schedules) in chronological order, displaying which students haven't added their schedules to the system. However, we found out that the last one was already existing in the current version in a different page, so we removed our implementation.

Points Completed: 7

4.4 Iteration 3

In this iteration we worked on adding new user stories to improve the usability of our web app and improved the test coverage.

In total we implemented 5 new user stories: ability to edit an existing schedule, replacing the email verification service with a UIN based account activation & password reset system, ability to add a new student to an existing cohort, showing a warning text if a student changes their schedule after a meeting time is selected for that student's cohort, simplifying the term change mechanism into just one step.

To improve the test coverage besides writing new tests for the features we implemented, we refactored the code and removed the unnecessary parts. This process increased the test coverage to 83.29%. We also did some minor after the iteration report which increased the test coverage to 84.63%.

Points Completed: 11

5 Customer Meetings

5.1 10/25/2021 10.30am

In our first meeting, we learned in detail what is the purpose of the project and what is the state of the current legacy project. Seth told us the problems of the legacy project and we discussed the ways that we can improve the project.

5.2 11/10/2021 1.30pm

We fixed some bugs that were preventing running the project. We did our first demo to Seth where we showed the first working version of the web app. We also defined new user stories to work on.

5.3 11/17/2021 1.50pm

In this meeting, we demonstrated the fixes we made and the user stories we implemented in Iteration 2. We also defined new user stories.

5.4 12/08/2021 1.30pm

We demonstrated the final state of the web app which included the 5 user stories we implemented during Iteration 3.

6 Testing

The legacy project had 59.6% test coverage. With every feature we implemented we made sure it was covered with behavioral tests. And at the end, we refactored the legacy code and removed unnecessary parts that were not covered by any testing. It resulted in 84.6% test coverage.

Since there were already written step definitions we relied mostly on behavioral testing instead of unit testing. We implemented our cucumber tests without writing too many new step definitions and we mostly reused old step definitions to create our new tests. One side effect of this was the length of our cucumber tests became too long because steps were not tailored for our tests. This both increased the complexity of tests and the memory usage of running cucumber tests. Our free AWS instance would run out of memory if we were to run them repeatedly.
Other than that, we usually wrote tests for a feature after implementing the feature which is not the ideal approach. However, every team member was aware of the edge cases of their tasks and our approach didn't cause us any unexpected issues.

7 Configuration Management

Our configuration management approach was to create a branch for every new feature. We pulled requests to allow for code review before the branch was merged. We did not do any spikes. We ended up with 10 feature branches. Among these, eight branches were merged. The unmerged two branches were used as backup by our developers for two of the features.

8 Heroku Deployment

The legacy project was uploaded to Seth's Heroku account. Initially we didn't have the credentials to that account so we deployed the web app to one of our team members' account. Before our final iteration report, we deployed to Seth's account. But we realised the free tier's database size won't be enough with real student data and mentioned this to Seth.

9 Github

The free tier of Cloud9 has low storage & memory capacity. And our Cucumber tests were complex and memory heavy. Running tests repeatedly was consuming the resources and we would need to terminate and rerun the instance.
The ZLP scheduler repository was under the TAMU-ZLP organization. We weren't able to push new commits initially. Afterwards, Han, who was a member of the ZLP scheduler team in the previous semester, invited us to the repository.

10 Tools and Gems

We inherited the project from the previous two groups, so we mostly use the same tools and gems. We use cucumber-rails, capybara, rspec-rails, and database_cleaner for common ruby-on-rails gems. We also used factory_bot_rails, faker, and selenium-webdriver in the testing environment.
We had two major changes to the gems. First, we upgrade rails to 5.2.4.4 and bundler to 2.2.15 and many other gems to fix the vulnerabilities. Second, we remove the send-grid

gem because we have removed the email feature due to the limitation of the SendGrid email service.
For the tools, we followed the last group to use CodeClimate and SimpleCov to collect data about any smells we had in our code and the test coverage of our code.

11 GitHub Repository

11.1 Project Organization

The Documentation folder contains images, iteration reports, meeting minutes etc. If you were to pick up this project in Fall 2020, you would want to add a subdirectory to Documentation called Fall2020. This would be where you can upload relevant reports and documentations during your semester.
The zlp-scheduler folder is where the Rails app lives. This folder was created with the standard Rails 5 rails new zlp-scheduler. Due to the project organization, the Heroku deploy step is slightly different. See that section to deploy to Heroku.

11.2 Deploy to Heroku

The caveat here is the git subtree command which will specify to git to push the sub-directory zlp-scheduler as the root directory to the heroku remote.
Run from root directory of project
> heroku apps:create
> git remote add heroku #{HEROKU_GIT_REPO}
> git subtree push –prefix zlp-scheduler heroku master

After that, you can setup the database
> heroku run db:create db:migrate db:seed

11.3 Development

To develop new features, you first need to create the development environment on Cloud9, the steps are described below

- Install PostgreSQL on cloud 9
- Install firefox and geckodriver for Selenium for some of the cucumber tests
- Clone source code and install bundler

Then, you can install the application

11.3.1 Install required gems

cd zlp-schedule
bundle install

11.3.2 Reset database

rake db:drop db:create db:migrate db:seed

11.3.3 Run server

rails s
The test username and password is specified in seeds.rb, you can find more test data from the history of the file

11.3.4 Run Cucumber tests

cucumber features

11.3.5 Run Unit tests

We test the algorithm using a unit test.
rspec
For more details, you can refer to the link below.
`https://github.com/tamu-zlp/zlp-scheduler/blob/master/zlp-scheduler/development.md`

12 Links

Pivotal Tracker: `https://www.pivotaltracker.com/n/projects/2536613`
GitHub: `https://github.com/tamu-zlp/zlp-scheduler`
Heroku: `https://zlp-scheduler.herokuapp.com/`
Poster: `https://github.com/tamu-zlp/zlp-scheduler/blob/master/documentation/Fall2021/POSTER.pptx`
Video: `https://youtu.be/4y9GKXJFMJ0`