

Team: ShadowCoders -- Force Request System

Product Manager: Niti Jain

Scrum Master: Shamsu Shahid Raja Mohamed

Other Members: Abdullah Abdul Kader, Karimi Abhishek Das, Nishant Aditya, Ramesh Ghimire, Sulav Adhikari

Relevant Links

Pivotal Tracker: <https://www.pivotaltracker.com/projects/2204401>

Github: <https://github.com/abkds/Force-Request-ChiUSDE>

Heroku: <https://force-request-2018.herokuapp.com/>

Demo Video: <https://vimeo.com/305527180>

Final Interview: <https://vimeo.com/305536016>

Two Paragraph Summary

The Force Request System is a Ruby on Rails legacy project which the Texas A&M's Computer Science Department would like to use as a replacement for the existing system. Our customers were Dr. James Raley Marek, Dr. Richard Furuta, Dr. Scott Schaefer, and Dr. Vivek Sarin, from the Computer Science Department. Our stakeholders consisted of the above-mentioned customers, the other computer science advisors as well as the students of Texas A&M University who wish to enroll in CSCE courses. The updated Force Request system is easier to use for students and provides additional functionality for the admins.

The legacy system enabled students to register using their student email address and verified the identity of the student with the TAMU directory. The students can then submit force requests and view their submitted requests and their status. Administrators can approve, deny or place the force request on hold and record notes for themselves and other administrators. They are also given the ability to filter the force requests based on the status as well as the course. They can then change the status of the force request and upon the change of status, an automated email is sent to the student as an update for the action taken. The main upgrades to the system that the customers needed were a feature to maintain a log of all the requests, ability to customize the email being sent to students, ability to limit the number of requests students can submit and for students to prioritize their requests. With the updated system, the admins are given more control over the system from the User Interface. The ability to set limits on the requests and edit the default email template has been added under the admin actions. Additionally, a priority level has been added to the requests which allow the student to prioritize their requests. The students are also given the ability to edit their requests and change the priority level. Lastly, a log for each request has been created which records when the request was created, edits if any, when an action was taken on the request and which admin took the action.

Understanding the Existing Code

The codebase that was inherited from the legacy project was well documented and adequately tested. We analyzed the flow of the code by dividing the team and using pair programming to run through the tests. However, the codebase contained a lot of code repetitions. Hence, part of the code was refactored by the DRYing fashion. Additionally, in the legacy codebase, a lot of the validations were performed at the controller level which made the controller unnecessarily FAT. We moved part of it to the javascript level thereby creating more specific and optimized controllers.

We also found a couple of bugs in the legacy codebase. Student registration did not work for students with middle names or students with space in their first name. We solved this by adding a validation technique of looking for the middle name and also considering the space in the first name. Additionally, there was no timeout for the Authentication user email sent when students sign up for the system. Hence, a student would be locked out of the system if he deletes the authentication email mistakenly. Therefore, a 20 min timeout period was added for this authentication after which the system would refresh and allow the user to resend the authentication email

Configuration Management Approach

For our configuration, we all set up the same environment, as shown in our Github documentation. In addition, for the main user stories like the emailer and the logging feature, there was a separate branch created which was branched off from the develop branch. We divided our team into pairs and worked on the respective user story assigned to us on a separate branch. After merging and after each release, all these branches were removed considering the task had been completed. We had four releases that were tagged appropriately according to the iteration. All the releases contained more than one user story and we only released at the end of the respective iteration in which they were completed

The Trouble with Cloud9, Github, Heroku, Pivotal Tracker

We did not face any issue with the cloud9 platform and Github. However, our free trial expired before the project so we were not able to completely update our user stories in the pivotal tracker. But for the initial days, we found pivotal tracker very convenient to update and work with user stories.

Since we used encryption in our design, it was initially difficult for us to set up the environment in Heroku. Eventually, with the help of the “figaro gem” we were able to configure the production environment with app-level encryption enabled.

GEMs and Other Tools

Most of the gems we used are standard. These include *rspec-rails*, *cucumber-rails*, and *pg*, which allowed us to write unit tests with RSpec, functional tests with Cucumber, and connect to a postgres database respectively. These came with the codebase when we received it. Over the course of the project, we added one gem: *gon*, which is used to send some data to javascript files without using views and parsing.

TDD and BDD Process

We followed the Test Driven approach for every iteration. Prior to coding the feature, we decided on possible test cases that the feature needed to satisfy. We followed this approach in every iteration. Finally, we were able to achieve total code coverage of 92%.

Our version of the force request heavily makes use of sending email, we were unable to completely test client-side part of the email.

Iteration 1 (3 points completed)

Customer meeting: October 09, 2018 Tuesday at 9:00 a.m.

In our Iteration 0 meeting, we got an overall picture of the concerns of the customers regarding the legacy system. We were able to identify the key features that they wanted for the legacy system. We were unable to get the existing system up and running for the Iteration 0 meeting due to the name validation bug as we were not able to register on the system.

For Iteration 1, we analyzed the code thoroughly and uncovered a few bugs in the code that prevent certain users from registering for the system. The first bug was the middle name validation and the second bug we found was the lack of timeout in the user authentication email. We solved the first bug by adding a validation technique of looking for the middle name and also considering the space in the first name. The second bug was fixed by introducing a 20 min timeout period for the authentication after which the system would refresh and allow the user to resend the authentication email. Additionally, we worked on one of the features that were identified from Iteration 0. We added a priority level for the requests. The students can now choose from a drop-down of 5 priority levels for their requests.

Legacy code bugs

1. Student registration did not work for students with middle name: Whenever the student had the middle name, and registering for the force request system the legacy software

was not allowing him/her to register it. The software used the first name and last name to search the status of the student ignoring the middle name which was the main cause for the issue.

For example, The software could not find the student with the name “George W. Bush “ because it looked for the name “George Bush “. To fix this a validation technique was added that looks for the middle name and also considers a space in the first name.

2. No timeout in the Authentication user email: While registering to the force request system, the student was sent an email with the authentication code. If the email sent was deleted mistakenly, then there was not any mechanism for a timeout. The bugs were a reason for not testing the corner cases thoroughly as it is nearly impossible for getting 100% test coverage. This was fixed in the updated system by adding a 20 minute timeout period for the authentication of the user. The user should have to authenticate the confirmation email received within the 20 minutes, after which the authentication system would refresh and allowed the user to resend the authentication email

Features

1. Prioritization of force requests (3 points)

This user story was implemented with a drop-down menu defaulting to Very High Priority. Five options are provided on the drop-down menu; Very High Priority, High Priority, Normal, Low Priority and Very Low Priority.

The image displays two versions of a form titled 'Making New Request'. On the left is a hand-drawn sketch on lined paper, and on the right is a digital mockup of the same form.

Hand-drawn Sketch (Left):

- Header: 'An update to existing make request page.'
- Title: 'Making New Request'
- Fields: 'Full Name', 'Major', 'Classification', and 'Minor', each followed by a rectangular input box.
- Priority: A dropdown menu with a downward arrow, showing options 'Low', 'Medium', and 'High'.
- Buttons: 'save' and 'cancel' at the bottom.

Digital Mockup (Right):

- Header: 'Email' with the value 'ssrm@tamu.edu'.
- Fields: 'Expected Graduation*' (2018 Fall), 'Request Semester*' (2018 Fall), 'Course Id* (CSCE)' (i.e. to choose course CSCE 629-600, er), and 'Section Id*' (i.e. to choose course CSCE 629-600, er).
- Priority*: A dropdown menu with a downward arrow, showing options 'Very High' (selected), 'High', 'Normal', 'Low', and 'Very Low'.
- Notes: A text area at the bottom.

Iteration 2 (9 points completed)

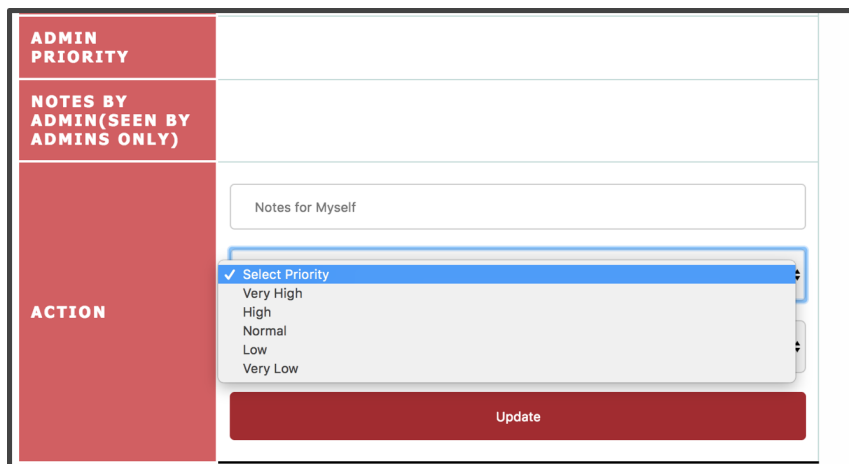
Customer meeting: October 23, 2018, Tuesday at 9:30 a.m

Having the system up after fixing the bugs and adding the student priority feature to the requests, we were able to work on more features this iteration. After the demo of the previous iteration with the customers, they requested a new feature wherein admins can add a priority level to the request. Additionally, we also added the custom email feature from the admin side. Lastly, we also implemented the limits on the number of requests a student can submit.

Features

1. Admin Priority of Requests (3 points)

This user story was implemented such that an Admin can add a priority to a particular request. Five options are provided to the admin in a drop down menu: Very High Priority, High Priority, Normal, Low Priority and Very Low Priority



The screenshot shows a web form for setting admin priority. It has a red sidebar with three sections: 'ADMIN PRIORITY', 'NOTES BY ADMIN(SEEN BY ADMINS ONLY)', and 'ACTION'. The 'ADMIN PRIORITY' section is empty. The 'NOTES BY ADMIN(SEEN BY ADMINS ONLY)' section contains a text input field labeled 'Notes for Myself'. The 'ACTION' section contains a dropdown menu with the following options: 'Select Priority' (highlighted with a blue bar and a checkmark), 'Very High', 'High', 'Normal', 'Low', and 'Very Low'. Below the dropdown is a red 'Update' button.

2. Custom email for admins (3 points)

This user story was implemented such that whenever an admin is taking any action on a particular request, the admin has the option to include a custom message that will be added to the default email in a special notes section as displayed below.

CSE Force Request System

Compose Message

Added you to the course, good luck!

Confirm

force.request2018@gmail.com
to me ▾

Hi Shamshu Shahid Raja Mohamed,

The state of Force request for CSCE 630-600 has been update to Approved .
Special Notes["Added you to the course, good luck!"]

3. Limiting student requests (3 points)

This user story was implemented such that whenever a graduate student is submitting more than 3 force requests or an undergraduate student is submitting more than 5 force requests then they will get an error message indicating that they have reached their maximum limit of requests.

CSE Force Request System

Maximum limit of force request reached

Iteration 3 (16 points completed)

Customer meeting: November 7, 2018, Wednesday at 9:30 a.m.

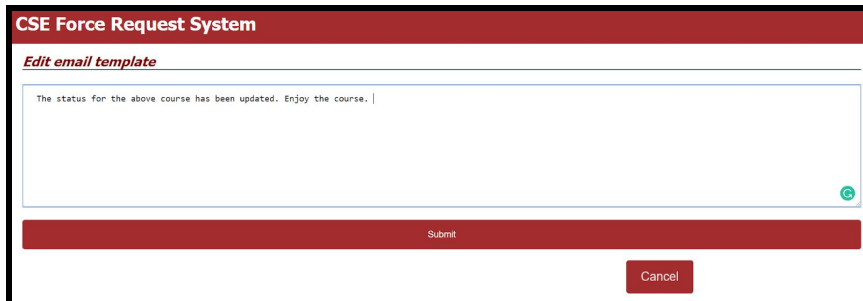
After the demo of the previous iteration, the customers wanted a few changes with the features we implemented in the previous iteration. Firstly, the customers wanted the template for the email being sent to the students be editable by the admins. Hence, we added an action wherein the admin can edit the default template being sent. Additionally, once the admin selects an action for the request, the email pops up and the admin can further customize the email and then send it. The customers also didn't want a static limit for the requests for the students. They requested that the limit is set by the admins according to the priority level of the courses.

In addition to the changes above, we also added the feature through which students can go back and edit the requests that they have submitted. Lastly, we also implemented a logging feature for each request. The admins can select a request and see when it was created and who took what action on the request.

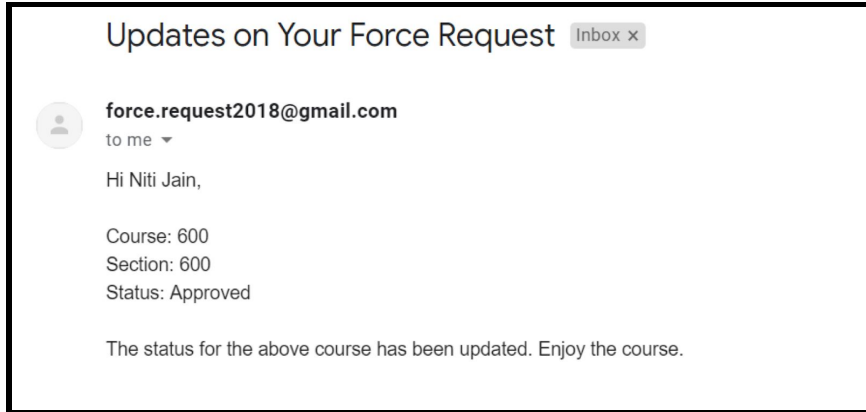
Features

1. Email template customization (5 points)

This user story was implemented such that an admin has the privilege to update the default email template which will be sent to all the students. Additionally, once the admin decides to take an action on a request, the email pops up and the admin can customize the email further before sending the email.



The screenshot shows a web interface for the 'CSE Force Request System'. At the top is a red header bar with the text 'CSE Force Request System'. Below the header is a section titled 'Edit email template'. Inside this section is a large text area containing the placeholder text: 'The status for the above course has been updated. Enjoy the course. |'. To the right of the text area is a small green circular icon with a white 'G' inside. Below the text area is a red bar with the word 'Submit' in white. At the bottom right of the form is a red button with the word 'Cancel' in white.



2. Limits on requests per priority level (3 points)

This user story was implemented such that an admin has the privilege to set the thresholds for the number of student requests that undergraduate and graduate student can make for each priority level. This grants the admin the option to change the limits as per the need during a particular semester.

The screenshot shows a web interface titled 'CSE Force Request System'. Below the title is a section 'Set limit for requests'. It features a dropdown menu for 'Select the classification' currently set to 'U1'. Below this are five rows for priority levels: 'Very High', 'High', 'Normal', 'Low', and 'Very Low'. Each row has a corresponding input field for setting a limit. The values entered are 2, 3, 2, 1, and 10 respectively. A 'Submit' button is located at the bottom of the form.

Priority Level	Limit
Very High	2
High	3
Normal	2
Low	1
Very Low	10

3. Options for students to edit requests (3 points)

This user story was implemented such that students can update their active force requests with updated information. The students are allowed to edit their expected graduation date, the priority level for the course and the notes for the request.

Howdy! Welcome back Niti Jain

[New Force Request](#)
[View Your Profile](#)
[Change Your Password](#)

COURSE_ID	SECTION_ID	REQUEST SEMESTER	REQUEST TIME	STATE	PRIORITY	MY NOTES	ACTION
628	600	2018 Fall	2018-10-22 16:46:48	Active	Very High		Delete Edit

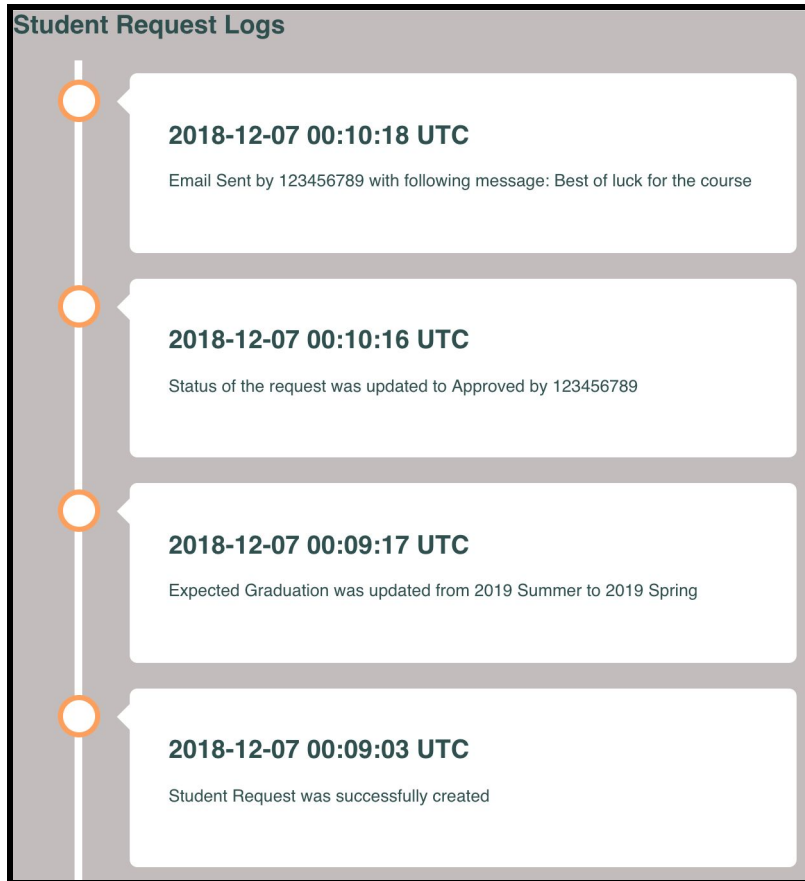
Edit Force Request

Full Name	Niti Jain
Major	Computer Science
Classification	G7
Minor	None
Email	nj305@tamu.edu
Expected Graduation*	2018 Fall
Request Semester*	2018 Fall
Course Id*	628
Section Id*	600
Priority*	Very High
Notes	

4. Log each action for a particular request (5 points)

This user story was implemented such that the admins can view a log in the form of a timeline that shows when the request was created by the student when/what edit took place on the request, what action was taken on the request and who took the action.

(-) CSCE 622													
	UIN	FULL NAME	MAJOR	CLASS	SECTION	GRADUATION	SEMESTER	STATE	STUDENT PRIORITY	ADMIN PRIORITY	ADMIN NOTES	LOGS	ACTIONS
<input type="checkbox"/>	926001674	Shamshu Shahid Raja Mohamed	Computer Engineering	G7	600	2019 Spring	2018 Fall	Approved	Very High	(View details to add priority)	(View details to add notes)	View Logs	View Details Approve Hold Reject
<div> <div>Select State</div> <div>Update Selected</div> <div>Email Selected</div> </div>													
(-) CSCE 629													
	UIN	FULL NAME	MAJOR	CLASS	SECTION	GRADUATION	SEMESTER	STATE	STUDENT PRIORITY	ADMIN PRIORITY	ADMIN NOTES	LOGS	ACTIONS
<input type="checkbox"/>	926001674	Shamshu Shahid Raja Mohamed	Computer Engineering	G7	600	2018 Fall	2018 Fall	Approved	Very High	High	vbv	View Logs	View Details Approve Hold Reject



Iteration 4 (13 points completed)

Customer meeting: November 21, 2018, Wednesday at 9:30 a.m.

Due to the fact that this was the final iteration for this project, the customers' did not have any new large tasks. They had a few small changes to the features and the UI. Additionally, although we have been testing each iteration using cucumber and rspec, we had to write additional tests for some of the features we modified and bring the coverage to 90% in this iteration.

Features

1. Ability to download logs in csv format (5 points)

This user story was implemented such that the admins can download all the logged data in a csv format.

(-) CSCE 799

	UIN	FULL NAME	MAJOR	CLASS	SECTION	GRADUATION	SEMESTER	STATE	STUDENT PRIORITY	ADMIN PRIORITY	ADMIN NOTES	LOGS	DETAILS	ACTIONS
<input type="checkbox"/>	926001674	Shamshu Shahid Raja Mohamed	Computer Engineering	G7	601	2023 Spring	2018 Fall	Hold	Very Low	(View details to add priority)	(View details to add notes)	View Logs	View Details	Approve Hold Reject

Select State

Update Selected

Email Selected

[Download the entire Force Request System data as an Excel Sheet](#)

[Download the entire Force Request System Logs data as an Excel Sheet](#)

This user story was implemented such that the status of the requests are color coded. We chose green for active, yellow for hold and red for rejected requests.

UIN

Full Name

Major

Class

Section

Graduation

Semester

State

Student Priority

Admin Priority

Admin Notes

Logs

Details

Actions

926001674

Shamshu Shahid Raja Mohamed

Computer Engineerin
g

G7

600

2019 Spring

2018 Fall

Approved

Very High

(View details to add priority)

(View details to add notes)

View Logs

View Details

Approve
Hold
Reject

Select State

Update Selected

Email Selected

UIN

Full Name

Major

Class

Section

Graduation

Semester

State

Student Priority

Admin Priority

Admin Notes

Logs

Details

Actions

926001674

Shamshu Shahid Raja Mohamed

Computer Engineerin
g

G7

600

2018 Fall

2018 Fall

Approved

Very High

High

vbv

View Logs

View Details

Approve
Hold
Reject

3. Automated email to be sent for Approve/Reject only (2 points)

This user story was implemented such that the email is sent to students only when the status of the requests change to Approved or Rejected. Previously, the email was being sent for the change of status to Hold as well.

4. Added additional student classifications (2 points)

Previously, only 4 Undergraduate classifications (U1, U2, U3, U4) and one Graduate classification (G7) was recognized by the system. However, we have added an additional undergraduate (U5) and a graduate (G8) classification in the system. Additionally, we also added 4 letter acronyms to the major list in the system.

5. UI changes to font and color (2 points)

Lastly, several minor UI changes like font and colors were changed according to the preference of the customer as well as the University Brand Guide of TAMU.

Future Work

Verification of courses being offered

Students should be able to submit requests only for courses that are being offered that particular semester that they are requesting for. Currently, there is no such verification if the courses are valid and are being offered. We contacted Jake Leland, a graduate student from CSCE who has done something similar and obtained an idea of how this can be implemented.

The endpoint to use: https://compass-ssb.tamu.edu/pls/PROD/bwckschd.p_get_crse_unsec

You'll need to submit a POST request to this endpoint with the following request body:

```
sel_subj:dummy sel_day:dummy sel_schd:dummy sel_insm:dummy sel_camp:dummy  
sel_lvl:dummy sel_sess:dummy sel_instr:dummy sel_ptrm:dummy sel_attr:dummy  
term_in:201911 sel_subj:CSCE sel_crse:121 sel_title: sel_schd:% sel_insm:% sel_from_cred:  
sel_to_cred: sel_camp:% sel_lvl:% sel_ptrm:% sel_instr:% sel_attr:% begin_hh:0 begin_mi:0  
begin_ap:a end_hh:0 end_mi:0 end_ap:a sel_day: sel_sess:
```

- Authentication is not needed.
- This query actually returns the information for all of the sections of those classes. If the class is invalid, the page will display "No classes were found that meet your search criteria".
- The three params: term_in, sel_subj, and sel_crse are the only three you should ever need to change.
- The term is coded as follows: 201911, where digits 1-4 are the year, digit 5 is the semester (1 = spring, 2 = summer, 3 = fall, 4 = full-year), digit 6 is the campus (1 = College Station, 2 = Galveston, 3 = Qatar).
- To search for a particular course, enter the course number in the sel_crse field. To search for all CSCE courses, leave the sel_crse field blank.
- Postman (getpostman.com) is a super helpful tool when it comes to playing around and testing HTTP requests.
- Depending on the and the module being used to generate the HTTP requests, the app might complain that there are duplicate keys in the request body. This is true – all of the “dummy” values at the top of the request body are duplicates, but the endpoint won't work unless they're included. If you run into this issue, an easy workaround is to include those dummy values as URL parameters instead of as a part of your request body.
- To summarize, submit your POST request to:
https://compass-ssb.tamu.edu/pls/PROD/bwckschd.p_get_crse_unsec?sel_subj=dummy&

sel_day=dummy&sel_schd=dummy&sel_insm=dummy&sel_camp=dummy&sel_lvl=dummy&sel_sess=dummy&sel_instr=dummy&sel_ptrm=dummy&sel_attr=dummy

With the request body:

term_in:201911 sel_subj:CSCE sel_crse:121 sel_title: sel_schd:% sel_insm:%
sel_from_cred: sel_to_cred: sel_camp:% sel_lvl:% sel_ptrm:% sel_instr:% sel_attr:%
begin_hh:0 begin_mi:0 begin_ap:a end_hh:0 end_mi:0 end_ap:a sel_day: sel_sess:

Update the emailer to include admin in CC

Currently, the email is being sent automatically to each student whenever an action is being taken. However, the customers wanted the admin to be cc'ed whenever an email is being sent. If the admins select multiple requests and take an action on them, one email should be sent to all students with the admin being cc'ed.

Stress Testing the system

Now that the system has been populated with a majority of the features that the customers need, stress testing can be done on the system. The pre-registration period at TAMU can be quite busy for the system and most of the students would want to submit force requests for the courses that they couldn't register for. This could potentially lead to a very high load and hence, stress testing should be done to ensure that the system's behavior is not affected by intense loads.

Ability to customize emails if bounced back

The customers also wanted a way to be notified if there was an issue with the email being sent to the students. If the email was not sent successfully they wanted to be emailed regarding this issue.

Warn students if they have already registered for the course with a different section

The customers would like the students to be warned while submitting their force requests if the student has already registered for the same course under a different section