CSCE 606: SOFTWARE ENGINEERING

Force Request System

Final Report

Team: TitaniumSouth

Course Instructor:

Prof. Hank Walker

Team Members:

Lianghao Zou (Product Owner)

Dongwei Qi (Scrum Master)

Jing Jiang

Fangsheng Wu

Cheng Chen

Jianghao Shen

Summary

Pivotal Tracker link: https://www.pivotaltracker.com/n/projects/2412339
Vimeo video link: https://vimeo.com/373490421
GitHub repository link: https://github.com/tomatoJr/Force-Request-ChiUSDE
Heroku App link: http://csceforcerequest.herokuapp.com
Demo Video: https://vimeo.com/378199320

Our Force Request system project is a Saas application and is based on ruby on rails framework development. Its implementation is a basic MVC framework. It is for students who want to take CSSE major courses but can not be enrolled in because seats are full. They could ask a force request and then wait for the CSSE department to process it at the beginning of the new semester. They can track their request status and get the notification whether the request is accepted or rejected. Some non-CSSE major students can also do this. The out main goal is to update the old system and add some new features of our customer. Our customers are Dr. Walker, Jennifer, and Dr. Song.

Students have to create a new account using their tamu UIN if they use it for the first time, and then fill in their name, UIN, major, minor and their expect graduate time and some other information. All of these must be completed when they are using the school internet since another internet will not allow them to do this. After they submit it, they have to specify their course name and section. After they submit a force request, an email notification will be sent to their tamu email. The administrator will process their request and can choose to approve, reject. We fix some bugs and improve the user interface, and also add some new features. For example, tests for code, delete canceled courses and so on. More details are provided below.

Development Environment and Tools Used

Our main development environment is AWS cloud 9 virtual linux development tools. We use pivotal tracker to assign each person's work so that everybody is clear. Word is used to write description of process and user stories. Github is used to store repository and we push each time's iteration to Github. We run the test on local AWS environment and push to Heroku after everything is good. Cucumber and rspec are used for testing. Unit testing was made using RSpec, while functional testing were made possible by Cucumber scenarios.

TDD and BDD Process

Test driven test is implemented for each iteration and we use unit testing for added features. Behavior driven development was implemented by writing cucumber tests to ensure all functions work.

Team Roles

In addition to contributing code to project's features, the team assigned the following roles for each member. The tasks mentioned below were added throughout the development cycle:

Lianghao Zou (Product Owner): Zou held Product Owner role, which involved communicating directly with the customers, and arranging iteration meetings. Zou worked mainly on implementing user stories throughout the project's iterations. Zou helped test the app, add identified some bugs that were later fixed. Zou performed legacy code review, and helped others in constructing the new features.

Dongwei Qi (SCRUM Master & Project Manager): Qi held the SCRUM Master role, as well as the Project Manager, which is a team leader equivalent role. Qi organized the group, handled partitioning work into iterations, as well as assigning tasks for each iteration. Qi built up the operation and maintenance environment, worked on iteration reports, poster, and final report. Managed weekly team meetings.

Fangsheng Wu (Test Team): In addition to contributing code to each user story, Wu ran tests for each iteration, and improved code test coverage. Wu also attended all customer meetings, and gave suggestions to customer feedback. Wu worked on iteration 2 user stories and final posters.

Jing Jiang (Customer Manger): As the customer manager, Jiang arranged meetings and communicated directly with customers, and helped built user stories. Jiang also helped maintain online discussion areas (PivotalTracker, Github).

Cheng Chen (Test Team): Chen worked with others on improving test coverage for the iterations. Chen modified existing legacy code for bug fixes, and made sure existing tests were passing. Chen also attended customer meetings, and worked on iterations.

Jianghao Shen (Maintenance): Shen's main task was working with Wu in testing the code, ensured app's functions were performing properly. Shen attended weekly meetings and helped maintained the environment (AWS) and final report.

Legacy Code Discussion

The legacy code in general was well organized and could run stably on Heroku. But initial code review revealed several bugs both for student side and admin side. The project was structured as classical model-view-controller (MVC) architecture with 4 controllers and 9 models and tens of routes. This project fully followed Restful API to organize the routes. As for potential bugs, on the student side, the app didn't make judgement on duplicate force request before making one and there was a missing row on the force request form. On the admin side, adding new force request button would direct to a site which did not exist and result in a web breaking down. The

courses information was hard coded. App initially had a coverage test in the high 80% for existing test. These bugs were patched in iterations 1,2,3 respectively. To address the existing problems, we refactored the code in the following way. We edited the view template of proposing a new force request to retrieve the missing row and added a verification before submitting a force request. Furthermore we refactored the code to make sure the link was redirected to correct address and added a new function to delete force requests on iteration 2 and iteration 3. The blind tests were conducted by graduated students from Department of Electrical Engineering and customers for admin functions.

User Stories and Iteration Description

Below is a detailed description of the six user stories implemented throughout the project's five iterations. Figures were added to visually-representative user stories:

Iteration 0

Customer meeting date: October 17, 2019, Thursday at 9:00am

In this iteration, we figured out that the goal is to reach a test deployment with probably some desired new features, to determine whether we should go with this system or the LaserFiche-based system that other departments are trying.

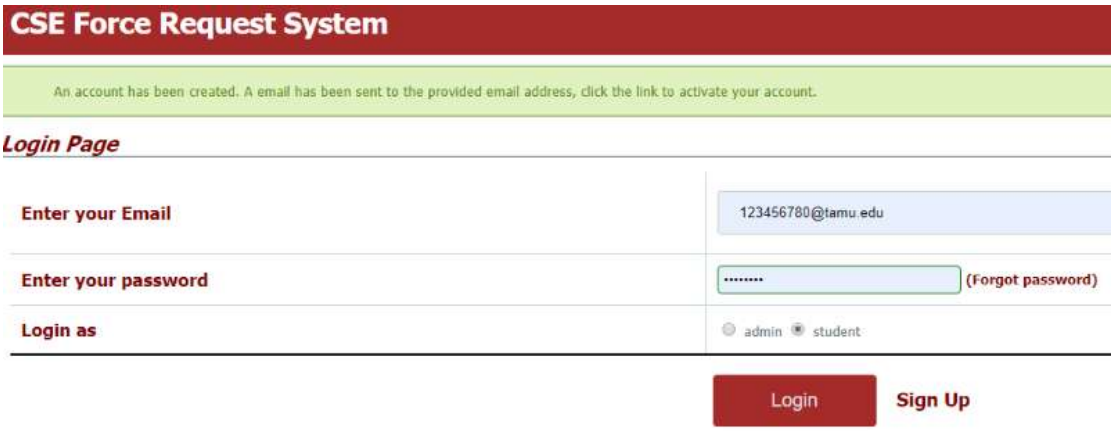
Through communications with the grad advising office, the main content would include a pilot-test on a subset of grad courses. After a discussion with TA, we focused on running the project on AWS and deployment. We also focused on how to write test cases and collect user stories.

Iteration 1

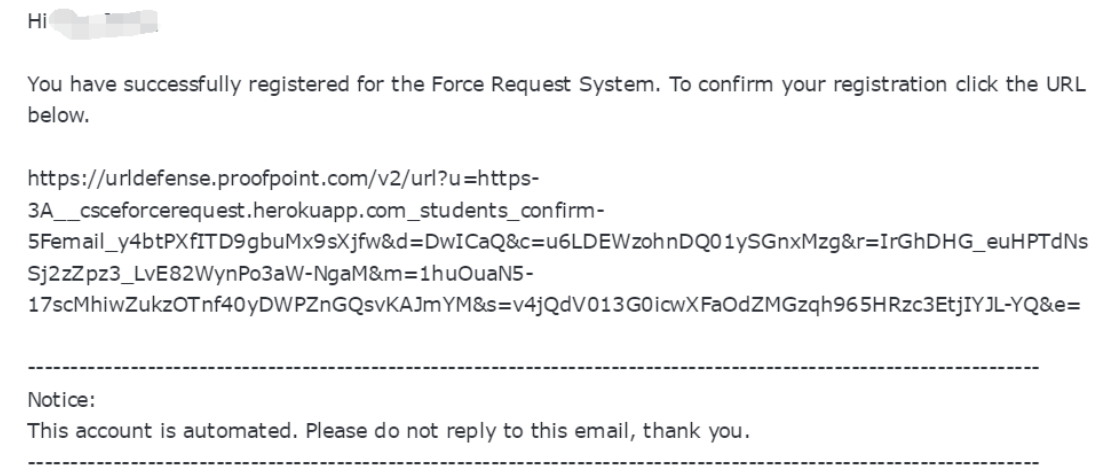
Customer meeting date: October 30, 2019, Thursday at 9:00 am

Deployed the system on Heroku: https://peaceful-fjord-63136.herokuapp.com/

Found a legacy code bug: Users did not receive confirmation email during signup.



Although it showed that the confirmation email has been sent, the user could not receive it. After debugging, we found that the email was blocked by google. Then we remove the block and the project works well.



Hi [redacted]

You have successfully registered for the Force Request System. To confirm your registration click the URL below.

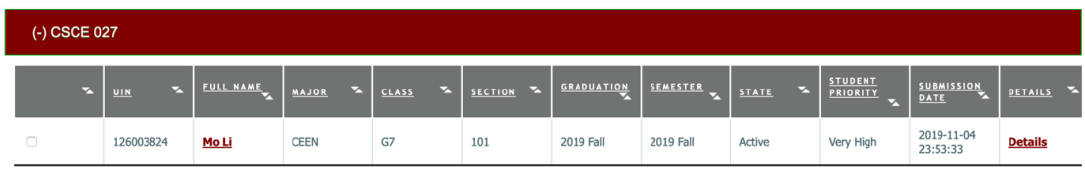
https://urldefense.proofpoint.com/v2/url?u=https-3A__csceforcerequest.herokuapp.com_students_confirm-5Femail_y4btPXfITD9gbuMx9sXjfw&d=DwICaQ&c=u6LDEWzohnDQ01ySGnxMzg&r=IrGhDHG_euHPTdNsSj2zZpz3_LvE82WynPo3aW-NgaM&m=1huOuaN5-17scMhiwZukzOTnf40yDWPZnGQsvKAJmYM&s=v4jQdV013G0icwXFaOdZMGzqh965HRzc3EtjIYJL-YQ&e=

Notice:
This account is automated. Please do not reply to this email, thank you.

Iteration 2

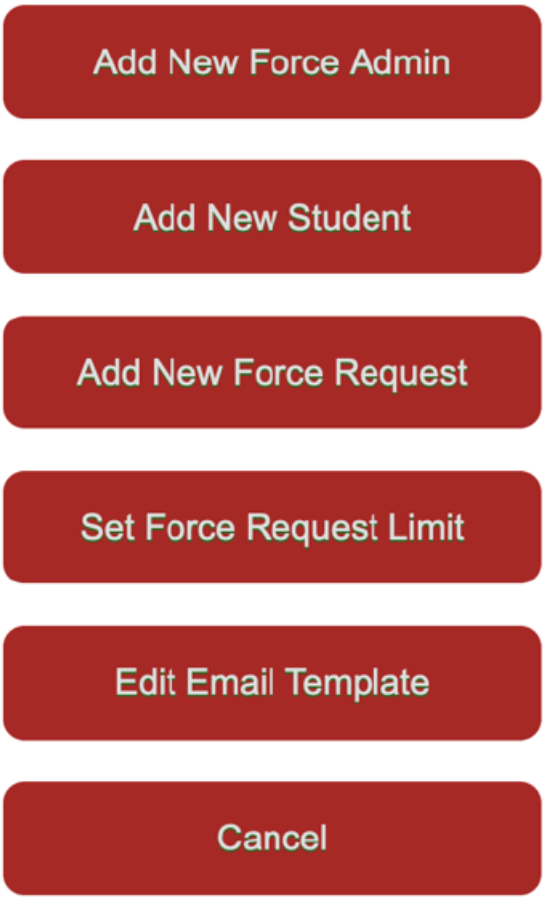
Customer meeting date: November 8, 2019, Friday at 9:00am

Made the detail links clickable:



We implemented this function such that the “Details” link of each request instance should have the attribute of “clickable” making it more indicative.

Added force request for the admins:

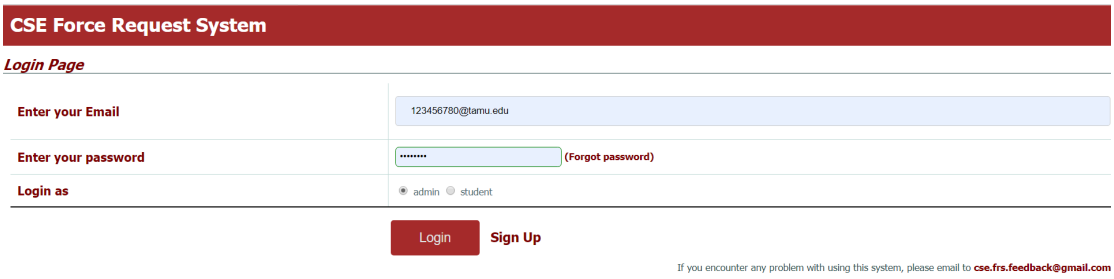


We implemented this function such that administrators can add new force requests through the “More Actions” operation.

Iteration 3

Customer meeting date: November 28, 2019, Thursday at 9:00am

Created admin test accounts:



We created the admin login ID so that the professor can see the 606 and 629 force requests, and start processing them for Spring 2020.

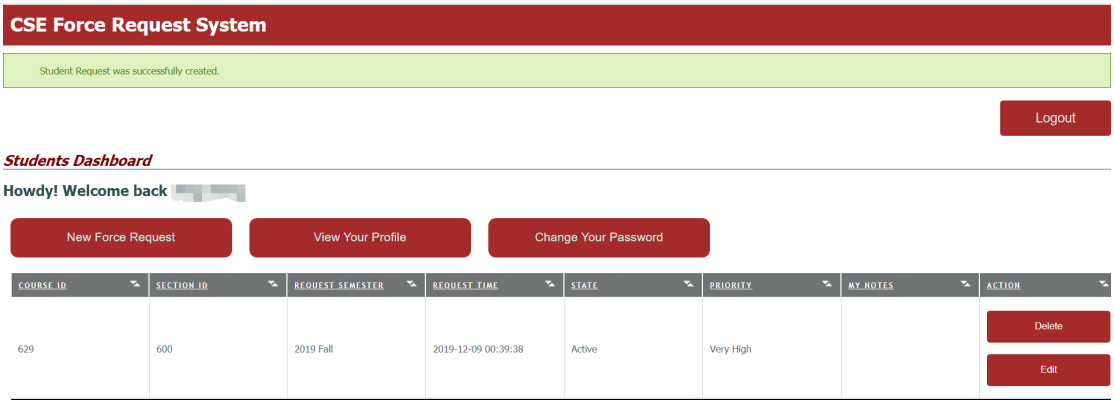
Rejected duplicated force requests for the same sections:

We implemented a function such that any students cannot duplicate request for one section of one same course.

Iteration 4

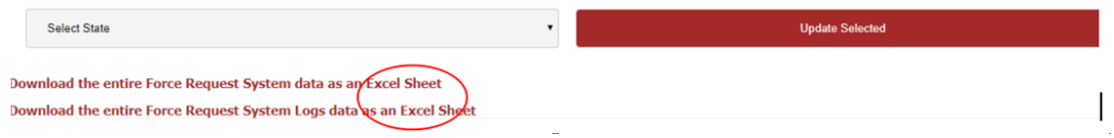
Customer meeting date: December 9, 2019, Monday at 9:00am

Withdraw an existing force request:



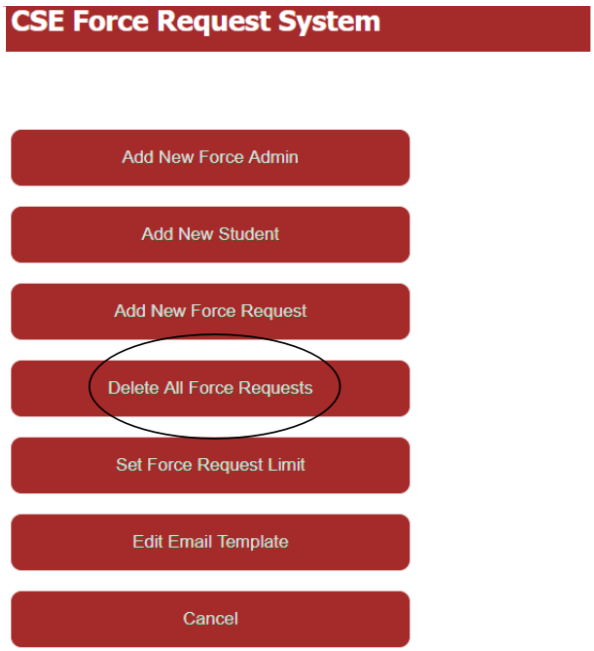
When a student submitted a force request, he or she can click “delete” so that the request could be withdrawn.

Archived requests log into excel:



We implemented a function to archive the requests for each year, and dump them into an excel spreadsheet. Professors can use past force requests to estimate future class demand.

Also we developed a way to for admin to wipe the database to empty the last semester's requests with confirmation.



Challenge

Several challenges which are met during software development are as follows:

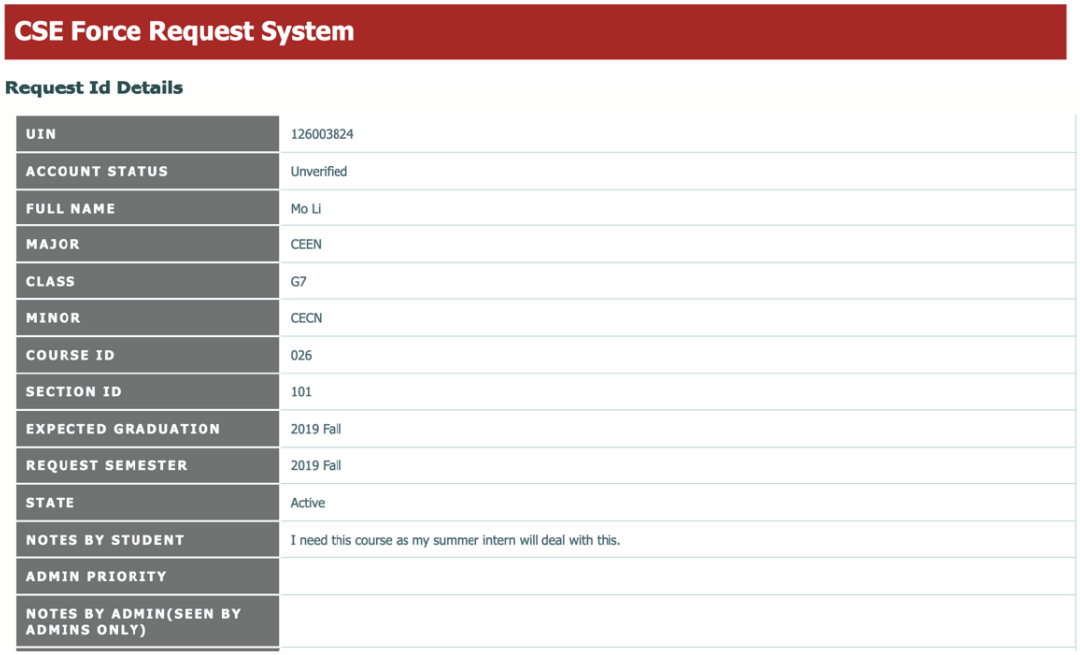
- **Development Environment Establishment Problem:** When we were trying to launch our app on a local environment, username with corresponding password in the seeds file could not access to index page. We made a change on code to judge whether to run encryption function based on the environment we are in. This change was followed DRY concepts to separate development and production environment and the problem was successfully addressed.
- **Hard Coded Courses:** Courses was hard coded in legacy code, and customer wanted a dynamic courses input and deletion. We established a new databases *courses* to meet customer demands. This change followed with CRUD concepts and Restful API to work smoothly with current function. This feature was partly finished and would require future work.

- **Clear Force Requests Efficiently:** Customer required a function to clear all force request with a confirmation box. We addressed this problem by adding a new route *delete* to delete all force requests with one click. To avoid mistakenly click, we added a confirmation box to let admin make sure. This change is subject to Restful API concepts and the feature was successfully added to running app.

Future Work

For future work, there are some features to be finished:

Notes for admin should be editable:



CSE Force Request System

Request Id Details

UIN	126003824
ACCOUNT STATUS	Unverified
FULL NAME	Mo Li
MAJOR	CEEN
CLASS	G7
MINOR	CECN
COURSE ID	026
SECTION ID	101
EXPECTED GRADUATION	2019 Fall
REQUEST SEMESTER	2019 Fall
STATE	Active
NOTES BY STUDENT	I need this course as my summer intern will deal with this.
ADMIN PRIORITY	
NOTES BY ADMIN(SEEN BY ADMINS ONLY)	

This user story should be implemented such that administrators can add notes for any force request instance and can be seen by students who raised it.

Get rid of the level classifications:

	UIN	FULL NAME	MAJOR	CLASS	SECTION	GRADUATION	SEMESTER	STATE	STUDENT PRIORITY	SUBMISSION DATE	DETAILS
☐	126003824	Mo Li	CEEN	G7	101	2019 Fall	2019 Fall	Active	Very High	2019-11-04 23:53:33	Details

This user story should be implemented that students have no access to determine the class(grade) on their own.

Upload all course information from excel:

The admin needs a way to load all the courses and sections for a semester in a batch load from a file of some format. The admin also needs the interface to add individual courses and sections.

Admin can wipe the database:

The admin needs a way to delete courses and sections, e.g. when the course is canceled. That way students don't try to submit force requests for a course that is "0 0 0" in Howdy, that is, no seats allocated, which means canceled.