CSCE 606 Software Engineering
A Project Report on

PhD Review System

Team: **HyperText Assassins**
Ankur Kunder (729007628)
Sumedh Pendurkar (830000048)
David Ratliff (424004174)
Purav Zumkhawala (230002341)
Anna Shekhawat (930001909)
Suresh Gandhi (729007555)

Customer: Ms. Karrie Bourquin, Program Specialist I, Department of Computer Science and Engineering, Texas A&M University, College Station

Stakeholders: Ms. Karrie Bourquin, Dr. Duncan "Hank" Walker, Team HyperText Assassins, PhD students and the Faculty of the Department of Computer Science & Engineering.

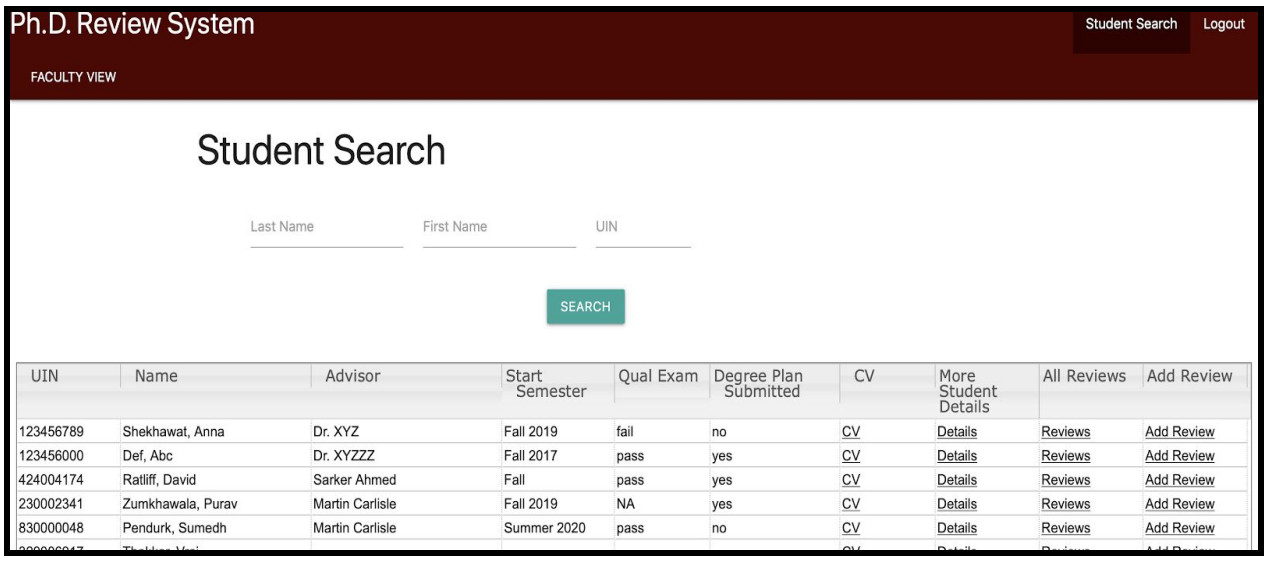
Summary

The customer needed a PhD Review System to facilitate the reviews for PhD Students in the Department of Computer Science & Engineering at Texas A&M University. The system should be accessible to the Students, the Faculties, the Program Specialists, and the Graduate Advisor.
In the implemented project, the PhD students will be able to login using their TAMU email addresses. They will be able to enter their information for the Review process. This includes entering relevant dates, uploading a CV and academic letters. The advisors should be able to access the system for evaluation of their students on a per semester basis and give them feedback. The Program Specialist and Graduate Advisor have administrative access to the system. Administrators will have direct access to the data with the help of Google Spreadsheets, using which they can edit access for users and update records for students. Information such as qualifier, prelim and final exam dates, proposal dates, GPA, advisor details, former reviews, current degree status, department decisions etc. are filled by the students, and visible to the reviewers. The students can also upload their Improvement plan and Report, and also view the reviews they are given. The reviewers will be able to see all the reviews for a student in a particular year. Due to FERPA law, the system uses TAMU Google Drive as its database. The implemented system will be used in the current review year, i.e. 2020.

Description of User Stories

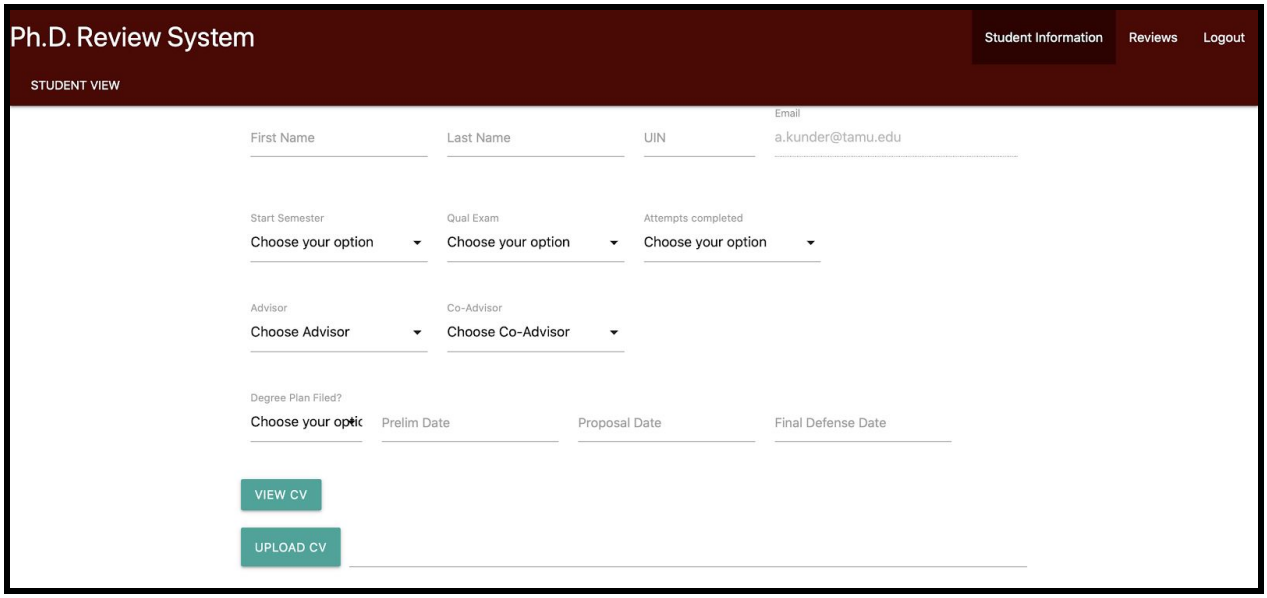
View List of PhD Students (7 points)
As a system administrator,
So that I can review the list of enrolled PhD students
I want to be able to search for students by setting filter criteria such as UIN number, first name/last name, and review year.

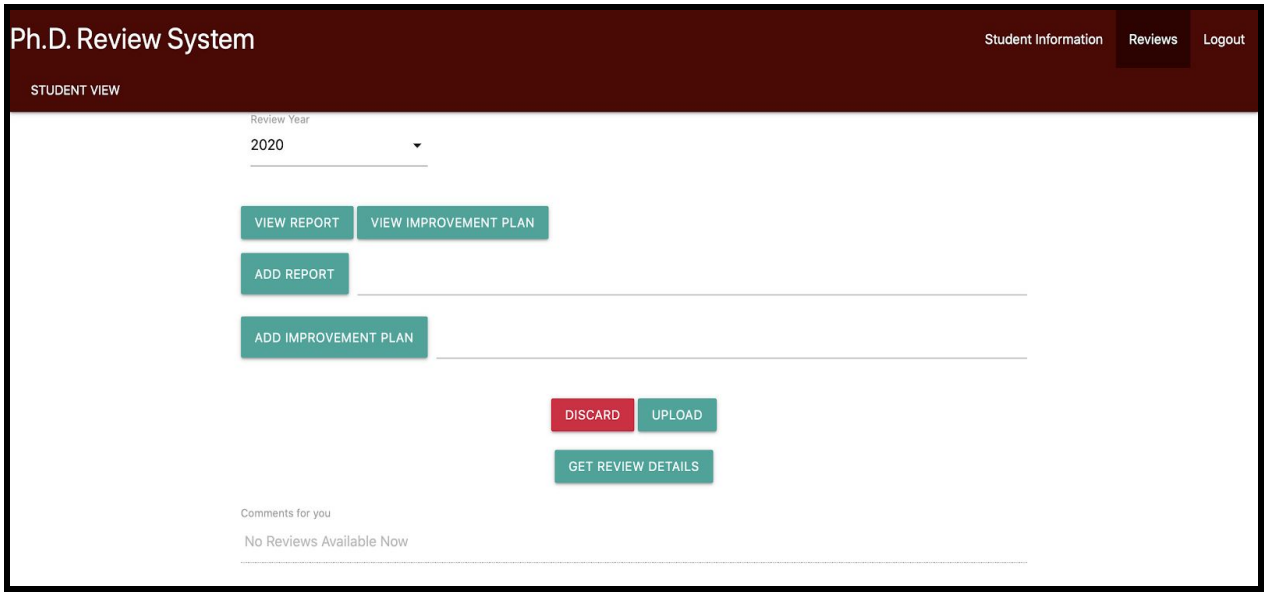
This story allows faculty and administrators to view all PhD students, and find students of interest. This story was modified to exclude the review year filter criteria, instead grouping all review years for a student under a single record. The modified story has been fully implemented.



Submit CV and Degree Plan (6 points)
As a student,
So that I can view and update my details
I want to upload my CV and degree plan to the review system database

This user story, as shown above, was written incorrectly in Iteration 0. It is meant to read ‘improvement plan’ instead of ‘degree plan’.The option to upload student reports was also added. The modified user story has been fully implemented. Below are screenshots corresponding to this story:

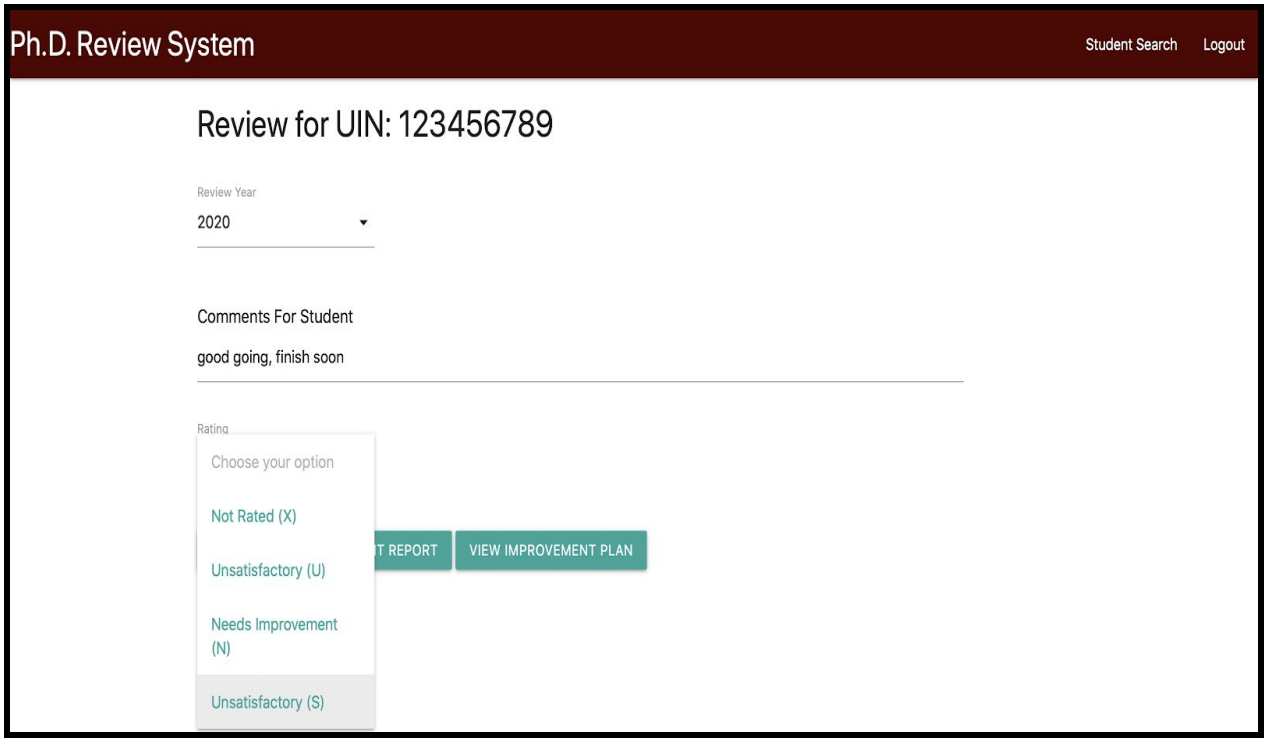




Rate Performance of Students (5 points)
As a teaching faculty,
So that I can evaluate students
I want to access the list of students enrolled under my guidance and rate their performance as SATISFACTORY/UNSATISFACTORY/NEEDS IMPROVEMENT

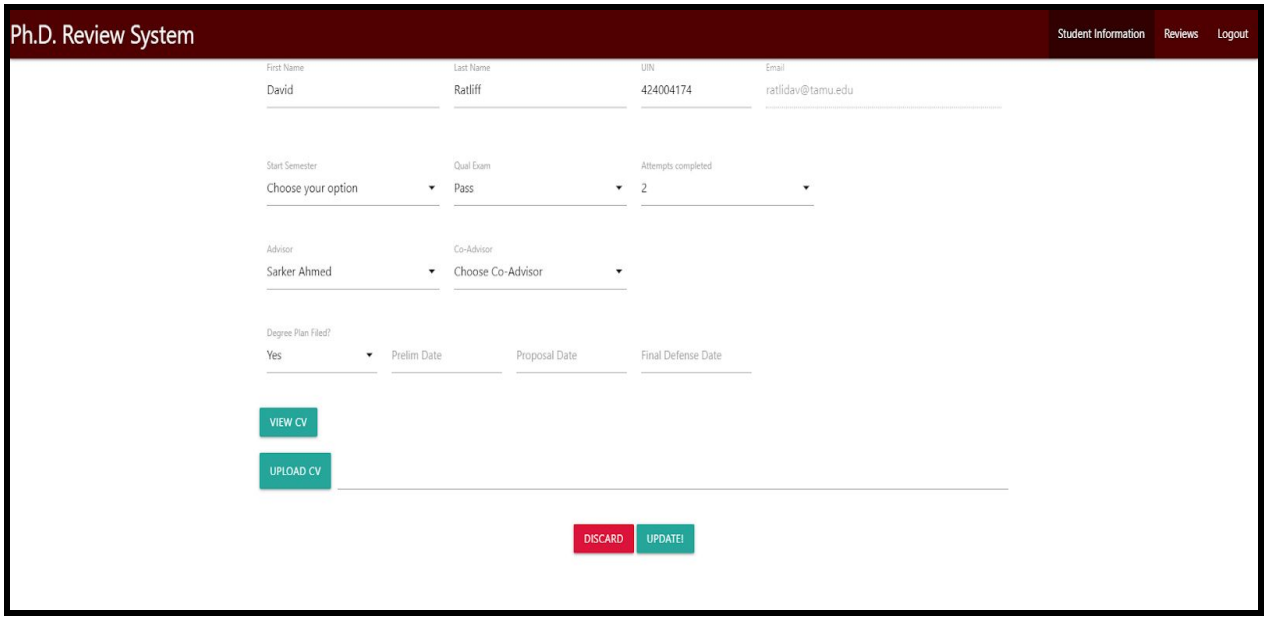
This user story was modified to remove a special view for showing a faculty member their own students. Instead, faculty are given a searchable list view of all students, as implemented in the first user story. In addition to the student rating, faculty can also add comments to the student's

review. They also have the ability to select certain common phrases to add to the student's generated review letter. New reviews can be created per academic year.



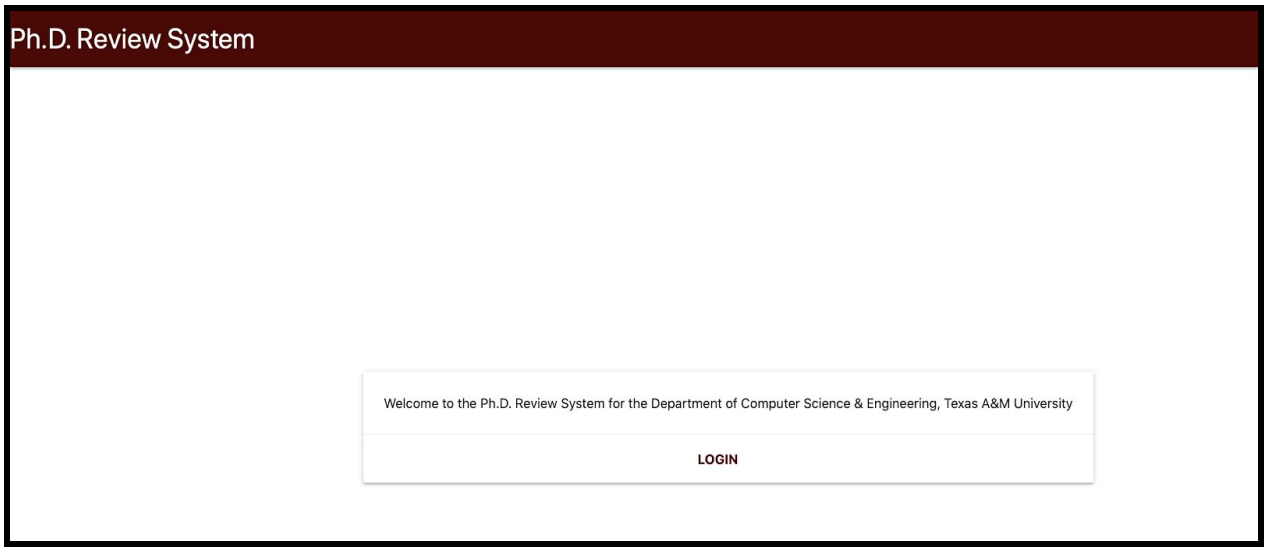
Enter Student Details (11 points)
As a system administrator,
So that I can keep the student details updated
I want to be able to enter prelim exam date, qualifier exam passing status, proposal date corresponding to each PhD student, and final exam date

This story also includes entering student data such as name and UIN. After consulting with the client, the final exam date was changed to the final defense date, and the responsibility for entering this information was delegated to students. In addition, a view was implemented for faculty to inspect a student and view their details.



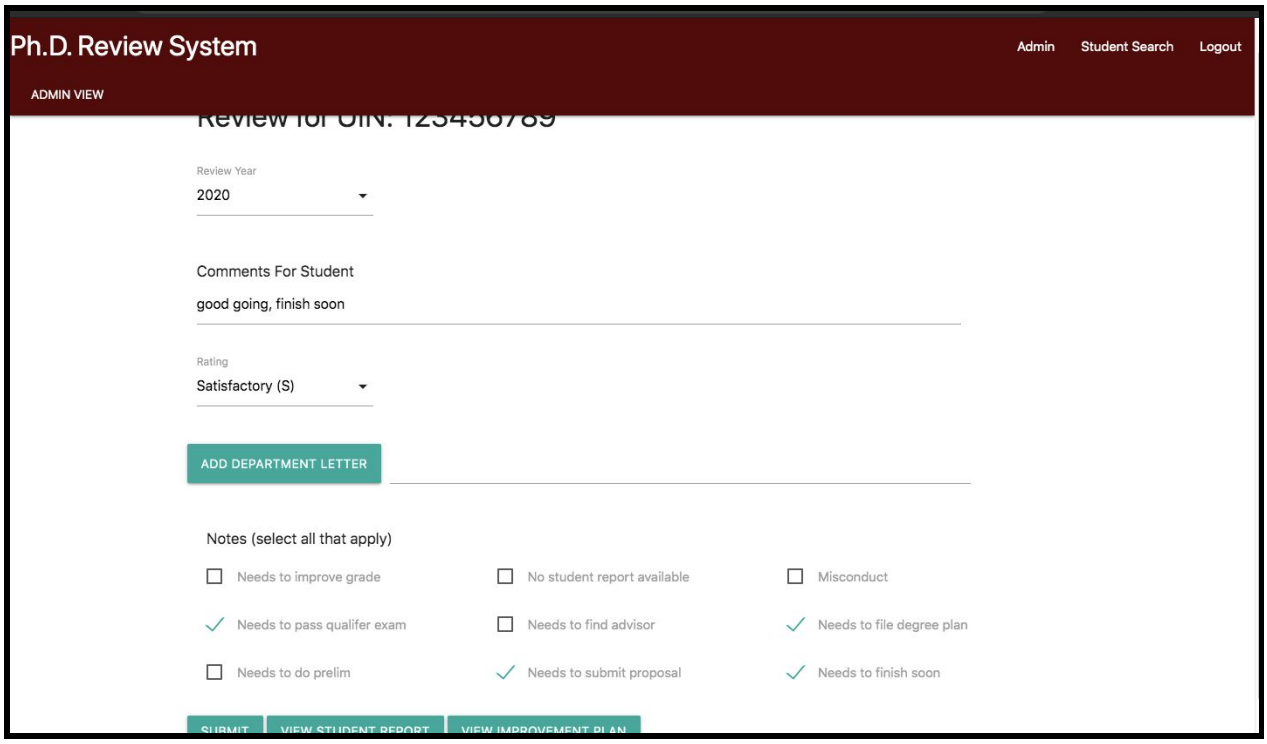
Login Path with Authorization/Redirection (5 points)
As a teaching faculty/student,
So that I can login to the web app,
I want to get access to student records with faculty/student credentials

This feature is fully implemented. Administrators must maintain a list of authorized faculty and student emails. Users accessing the application must be signed into one of these emails. Upon opening the app, it detects the user's email and redirects them to the appropriate view depending on whether they are a student or faculty, or to a landing page if they are not authorized.



Enter Student Details (7 points)
As a system administrator,
So that I can upload the Department Letter
I want to be able to upload the Letter, and mark the comments corresponding to each PhD student

This feature allows the Admin to upload the Department Letter for every PhD Student, as well as give them comments by selecting appropriate check boxes for each comment. This feature is only available to Admins and a faculty will not be able to see them.



Team Roles

Scrum Master: Ankur Kunder, **Product Owner:** Sumedh Pendurkar
Developers: Suresh Gandhi, David Ratliff, Anna Shekhawat, Purav Zumkhawala

Iteration Summary

Iteration 0: In iteration 0, we created Lo-Fi UIs, User Stories and made the Story Board according to the descriptions provided by the customer. We decided to try out frameworks such as Django, Google Apps Script, and Ruby-on-Rails for our project.
Points Completed: 0

Iteration 1: In iteration 1, we started off with deciding the prospective framework for the project discussing the pros and cons. We initially experimented with Python Django, given the popularity and support as well as a good background of most of the team members in the same. However, we concluded that the integration of Django with GSuite (Drive and Spreadsheet) posed compatibility problems (because the framework isn't designed to work directly and well enough with GSuite) in addition to latency issues. We then switched to Google Apps Script which, inevitably, provided a cure to these problems. All the team members spent time learning the Google Apps Script. The only major drawback of Apps Script is the lack of documentation and community support due to which our team members had to spend most of the time learning and getting acquainted with this new framework.
Points Completed: 7

Iteration 2: In iteration 2, the team implemented a majority of the Views. There was work done on the backend as well. The integration is still left as everyone was working individually. After this iteration, the Student will be able to enter her Data and upload a CV to Google Drive. This data can be viewed by the Faculty members. The Backend for Faculty to review the Students is still left, although the respective View was completed by this iteration.
Points Completed: 11

Iteration 3: In iteration 3, the team finalised the views and integrated the front-end and back-end to have a working application. We worked in pairs to accomplish this task. The Student view, and Faculty views will have restricted access. The students will be able to fill their details and the faculties will be able to submit their reviews.
Points Completed: 11

Iteration 4: In iteration 4, we handled the minor and major bugs that occurred due to the integration process. This helped to have a better application. We also contacted the customers and got their inputs. They were happy with the usability of the application and requested one last feature, which we will be submitting in the final submission.
Points Completed: 10

Customer Meetings

Meeting No.	Date	Location	Details
1	Feb 28, 2020	EAB-A Conference Room	● Met with the customer and got apprised of basic customer requirements. ● Customer shared screenshots of the current system for an overall idea. ● Discussed about the Google

			Spreadsheets Backend requirement and TAMU CAS login.
2	March 20, 2020	Email	• We confirmed the fields required for the student to fill out on the form and if any other fields were required to be visible apart from those. • We also asked about the accounts that would access the application and their access levels, if any.
3	April 2, 2020	Zoom Meeting	• Walked the customer through the Views (login page, search page, view details page) etc. • Received feedback and discussed about the features such as different keywords for search, application access, and ease of use for the admins as well.
4	April 15, 2020 and April 17, 2020	Email	• We decided on the format of the data that was required for the user to Login and what was the email type that we will require to verify a user's access rights to the system.
5	April 24, 2020	Zoom Meeting	• We showed our application to the customer and got feedback. They were happy with the usability of the application and requested one last feature, "View reviews of all faculties for a student", which we will be submitting in the final submission.

Technologies Used

The following tools and technologies were utilized in the project:

- Google Apps Script

- Google Drive/Sheets API
- jQuery
- Materialize (front-end framework)
- SlickGrid (grid UI component)
- GitHub

How To: Google Apps Script

Google Apps Script(GAS) is javascript with built in libraries for G Suite applications like Gmail, Calendar, Drive, Spreadsheets and more. There's no installation needed—they give you a code editor right in your browser, and your scripts run on Google's servers.
For more information, please visit the following links:

1. GAS Tutorials to get started : Covers a Beginner Example, loading and writing data using a Spreadsheet, searching for data, creating views, creating routes, rendering views in a Web App, etc.
2. Google Apps Script Website : Includes how to set up, and basic tutorials like How to extend Google Sheets and other G-Suite Applications.
3. Access GAS Locally from a command line vis Clasp.
4. Best Practices.
5. 2D Arrays Library.

Using https://script.google.com you can directly deploy your project on Google Servers and check the logs from the view tab. A temporarily deployed link is also available to test the latest changes which eases the development process.

For Google Chrome, there is a plugin available to check-in and push code from the scripts browser IDE itself. In other cases, you can use Clasp to maintain your code.

Using Google Apps Script limited our options for testing frameworks, as it is not designed for developing robust, complex applications. While there are a few third-party libraries that could be adapted for testing GAS, we couldn't use them due to the following issues:

- Limited types/capabilities of tests
- Not suitable for use with GAS webapps
- No longer actively maintained
- Considerable time was required to learn about them

Development Process and its Benefits/Problems

We followed a BDD approach during the entire development process. We periodically got feedback from the customer and made changes to keep up with the suggestions. This helped us minimize the issues down the road and minimize bugs in the project. But, as the project progressed it became increasingly difficult to accommodate those requests.

Configuration Management Approach

We have used GitHub for configuration and version control. We started working with the master branch and incrementally kept adding new branches for each functionality. At the end of each iteration, we used to merge it into the main branch. For testing purposes, we deployed the app to Google App Scripts provided framework. The final deployment would be done as per customer's convenience.

Issues related to releases

This project is deployed as a Google Cloud Project. As such, there is no requirement to explicitly manage a production environment. We had no issues with deployment.

Issues Related to GitHub

Using GitHub in combination with Google Apps Script presents some obstacles. Project files were pushed/pulled between GitHub and the local machine. We then had to use a tool called Clasp to synchronize local files with the online GAS project. Clasp automatically converts between .gs and .js files when pushing and pulling, causing confusion on what changes were made in what files. Several members of the team were also unfamiliar with GitHub, so it took some time to establish proper workflows.

Tools Used

- Clasp: Used for pushing and pulling GAS projects between a local machine and the online project in Google Drive.
- Pivotal Tracker
- GitHub
- Slack
- Google Apps Script Coding Environment, script.google.com
- Zoom

Important Links

Link to Video Poster Presentation and Video Demo:
https://drive.google.com/file/d/1IQwlaSCYSlub2BkxpBVmRbd894C3hcxm/view?usp=sharing

GitHub Repo Link: https://github.com/sumedhpendurkar/phd-review-system

Pivotal Tracker Link: https://www.pivotaltracker.com/n/projects/2437038

Deployed App:
https://script.google.com/a/tamu.edu/macros/s/AKfycbzpjSfnSZp1Klq1nmS6tIuNhGSTIsbl8q-jrpCjbNU-OekbOZY/exec