Program Review Information System Management (PRISM)

Team: Leanne David, Andrew McLees, Justin Sarenas, Ben Solis

Advisor: Dr. Chengyu Sun

Liaisons: Dr. Brown and Veronica Ramirez from
the CSULA office of Graduate Studies

Table of Contents

Chapter 1. Introduction

The sponsor of the PRISM project was the Office of Graduate Studies at California State University, Los Angeles (CSULA). Program review at CSULA is a complex process not supported by specialized software prior to the development of PRISM, thus, PRISM was envisioned as a way to streamline the review process. In short, it involved college deans, department chairs, program review subcommittee (PRS) members, and administrators collaborating on a number of documents. The process is highly asynchronous and fluidic; many reviews occur simultaneously and each review may have several documents in progress simultaneously. Additionally, deadlines for documents change often throughout the process. See chapter 2 of the PRISM Software Requirement Specification (SRS, see references for a hyperlink) for more details on the review process.

The existing solution to conducting the program review process was driven not by dedicated software, but by a combination of word-processing software and email. The complex nature of the review process made this collaboration difficult and confusing, particularly for new PRS members.

PRISM is a full-stack web application designed to streamline the review process at CSULA. It provides convenient representation, storage, and transmission of data within all reviews conducted and the tools to manipulate the data. The essential requirements of the system, as summarized in the PRISM SRS, are:

- Track and provide an interface to view the progress of each review as it proceeds through the review process
- Store, track the progress of, and allow collaboration on review documents
- Store and automatically source new documents from review document templates
- Store meeting agendas and minutes
- Maintain a calendar of PRS meetings and send email notifications upon changes
- Track which programs are due for review
- Send email notifications upon events relevant to the user

PRISM was implemented as a MEAN (Mongoose Express Angular Node) stack application. It consists of an HTTP API based on RESTful principles and a single-page client-side web application. It meets the requirements listed above and will be used starting during the academic year of 2018-2019 at CSULA.

The core benefits of PRISM are centralization of all review information, simpler management of review workflows for administrators, and automations of notifications and coordination previously conducted via email.

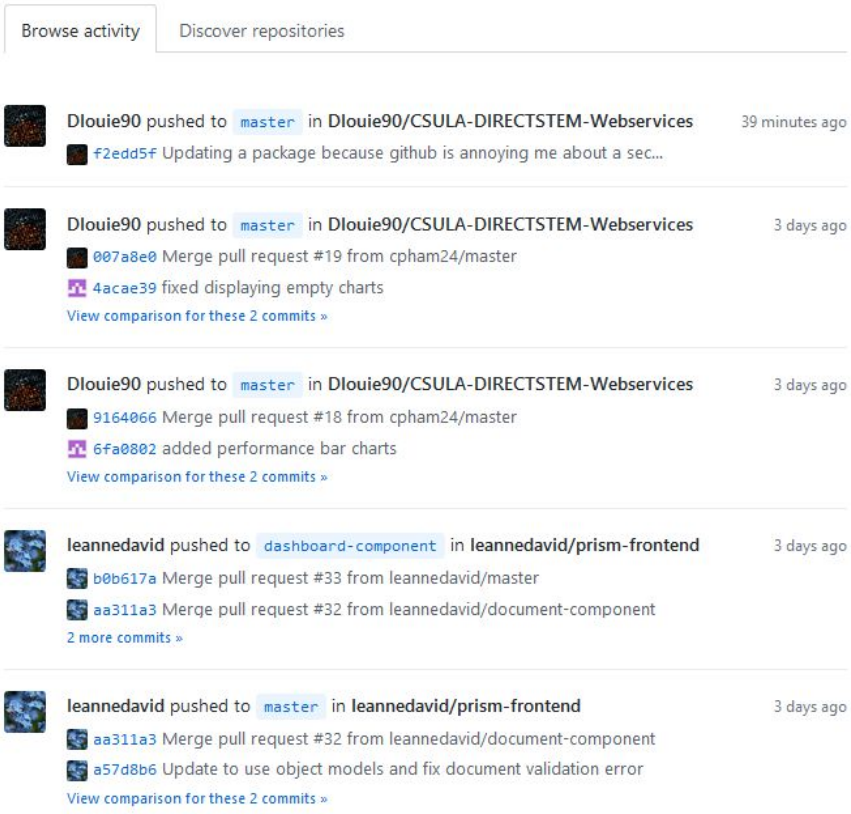
Chapter 2. Related Works and Technology

During the initial stages of PRISM's development, it was determined that a web application would best meet the needs of the Program Review Subcommittee. A web application provides:

- Centralization and reliable data storage in a proper database
- Simple deployment - only one server must be deployed to run the application
- Existing resources for User Interface (UI) development

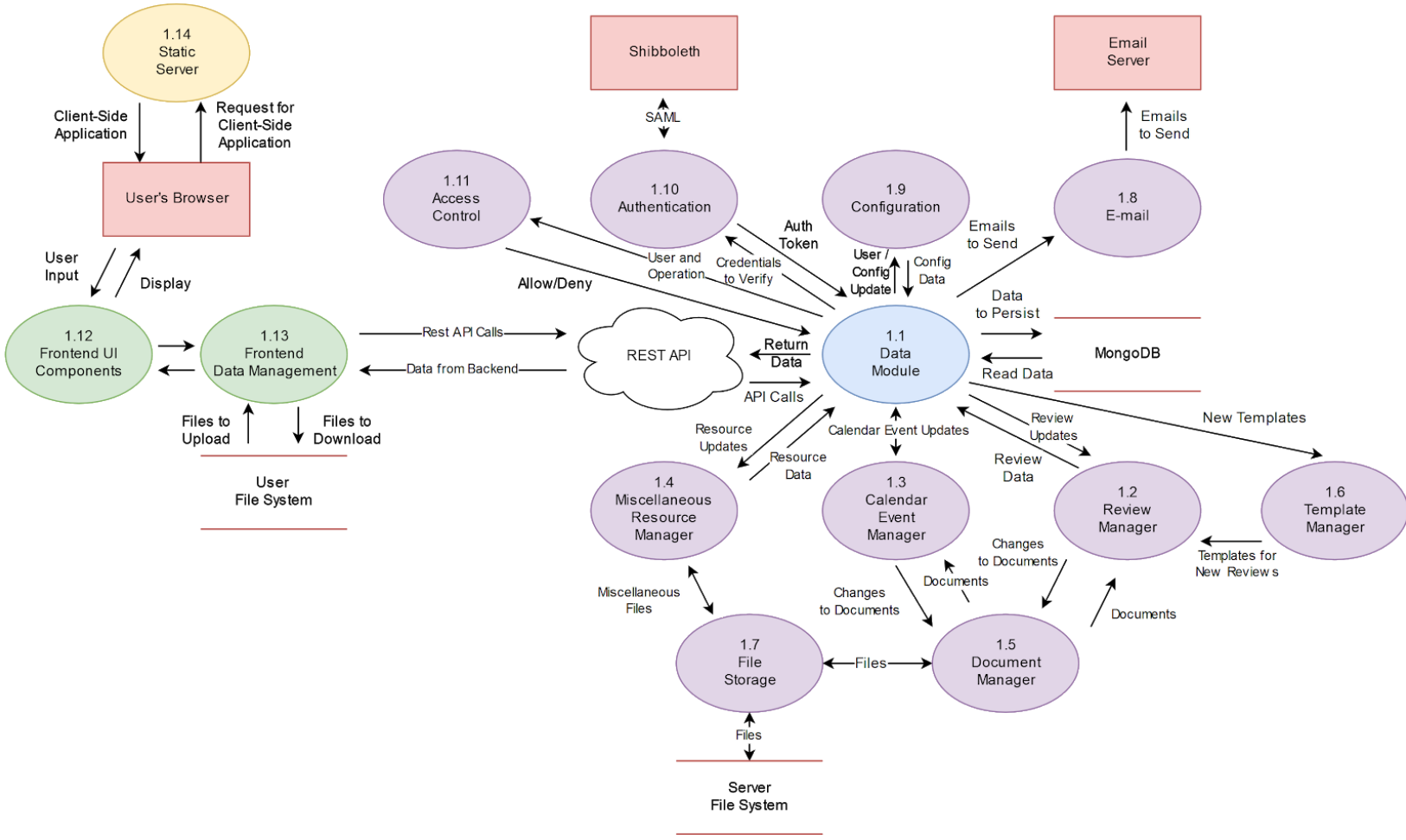
With a web application in mind, the next decision to be made was the software ecosystem. Though much of the development team lacked extensive experience in JavaScript, it was chosen as the primary language for development across the entire stack. Chapters 2, 3, and 4 of Appendix II. Software Design Document detail this choice further.

Review processes at many other universities, similarly to the CSULA program review process prior to the introduction of PRISM, depend largely on traditional document sharing via email and coordination via a set of hard deadlines set by those involved. This lack of similar systems led the team to examine the design of software accomplishing similar goals. One example is GitHub, an online host for Git repositories. GitHub, similarly to PRISM, provides tools for workflow management and hosts source-control repositories (similarly to how PRISM stores all versions of all documents as they are developed). The design of GitHub is similar to PRISM not only in versioning documents, but also in the way both systems display action feeds and collections of documents on their respective front pages. The figure below depicts the GitHub action feed.



Appendix I. Software Requirements Document Chapter 1.5 contains more details of resources consulted during the design and implementation of PRISM.

Chapter 3. System Architecture



The overall system was split into three categories of modules: frontend modules that run on the client side, backend modules running on the server, and a static server to serve the client-side code. This distinction was made to simplify development: the frontend and backend can be developed simultaneously with relative ease once the REST API has been specified.

The server was split into a variety of interconnected modules to effectively meet the requirements of the system. The reason for the large number of modules is to meet the single-responsibility principle, which states that each module should have a single responsibility. Having separate modules for different functionality makes development and testing each module easier, and thus each module was chosen to perform a specific set of functions corresponding to the requirements for the system.

Data Module
This module handles all the data interactions within the server. This connects all the modules together.

Review Manager

The review manager tracks all the states of the review. The review contains a nodes object with all the deadlines pertaining to the review. Administrators are able to extend deadlines. This module works with the email module in order to send notifications to relevant users of upcoming deadlines or completed documents.

Calendar Event Manager

This module tracks all the PRS meetings and events. Meetings and events can be created and scheduled on a calendar. This module works with the e-mail module in order to send out notifications of upcoming meetings and events.

Miscellaneous Resource Manager

This module handles all files accessible to everyone on PRS. This module interacts with the file storage module for the upload and download of files.

Document Manager

This module handles all files and revisions related to the review. Users will be able to upload revisions to a document. Administrators will be able to delete and restore to a previous revision.

Template Manager

This module handles the templates used in the reviews. Users will be able to download these templates.

File Storage

This module handles all file storage actions within the system. This module utilizes multer for file uploads to the server. File uploads are limited to extensions doc, docx, and pdf.

E-mail

This module handles all e-mail functionality within the backend. Automated e-mails are sent out when an event is triggered such as an upcoming deadline, meeting, document deadline, and etc. There are custom templates that correspond with the type of e-mail being sent out.

This module utilizes three packages named nodemailer, cron, and nodemailer-express-handlebars for ease of implementation.

Configuration

This module handles all user settings. Users may select to subscribe to a review for notifications regarding deadlines.

Authentication

This module handles authentication to access the software. Users will be able to log in and access the site using their CSULA credentials. Users will be given a token which will allow access to the different endpoints as each endpoint requires a token.

Access Control

This module handles all access control within the system. Users will only see data they are allowed access to. Access control consists of groups which have different privileges to certain data. Administrators are able to add or remove users in those groups.

Chapter 4. Results and Conclusions

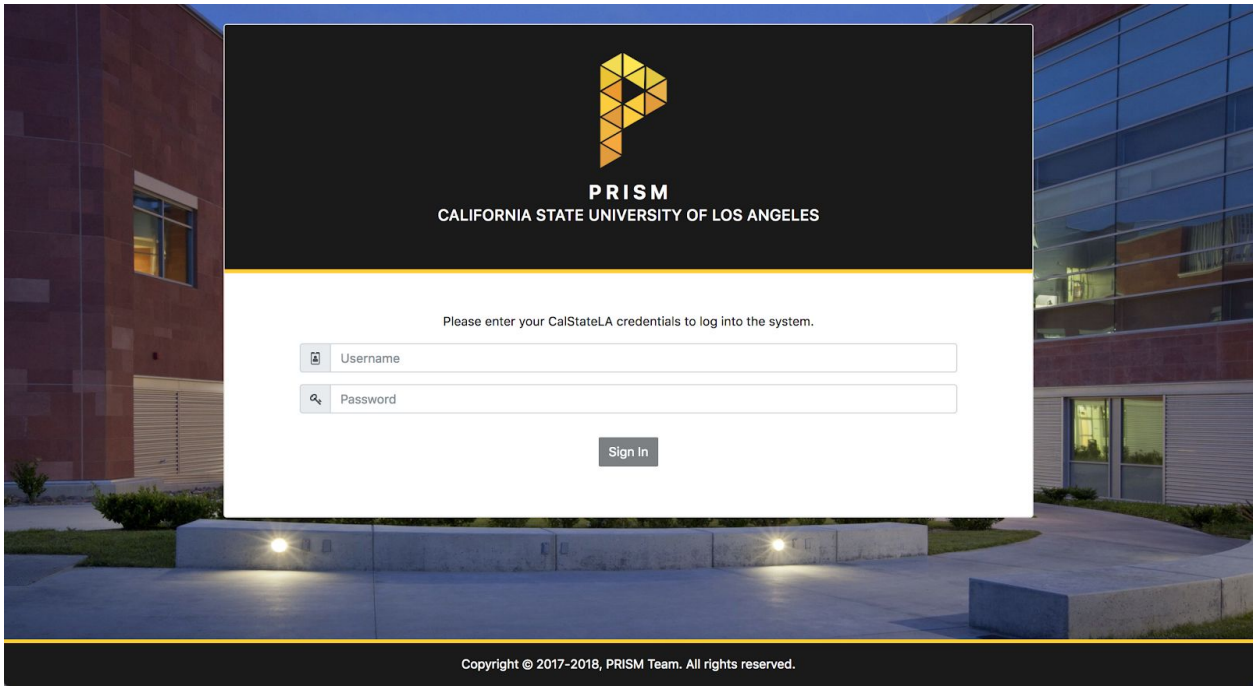
Results

PRISM is a complete web application that contains 80 endpoints, intricate data models, a RESTful-esque api, and a user-friendly interface. The majority of the endpoints implementation were completed in the winter; which allowed the front-end team to initiate the development of the user interface in a timely manner. The data models required custom validation modules and useful error handling code that logs errors and facilitates the debugging process. The user interface implementation was completed in the end of the academic year; which utilized a great amount of time for testing. The interface provides the user a standard navigation bar to facilitate the access to different components of the system. The elegance of the system's user interface is derived from a meticulous approach to its design. As one can see in the screenshots below, the user interface has a pleasant color schema and an intuitive layout.

Screen Frameworks or Images

Login Component

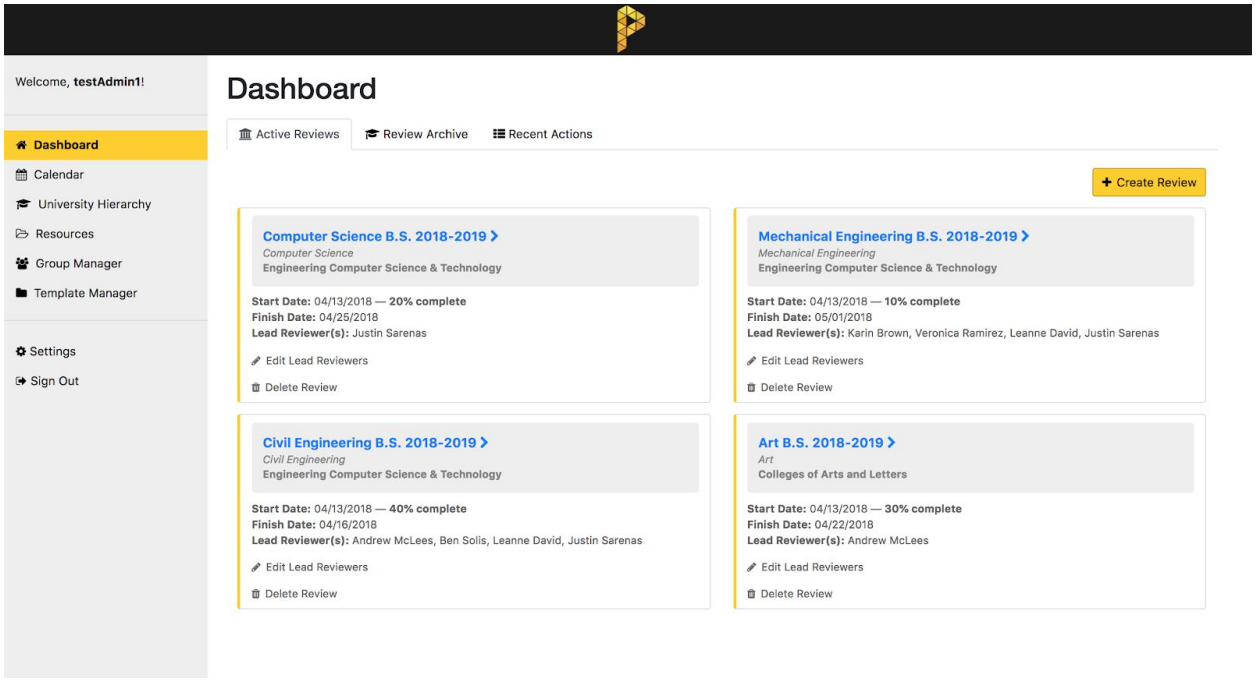
User enters their login credentials, typically their Cal State LA information, to log into the system.



Dashboard

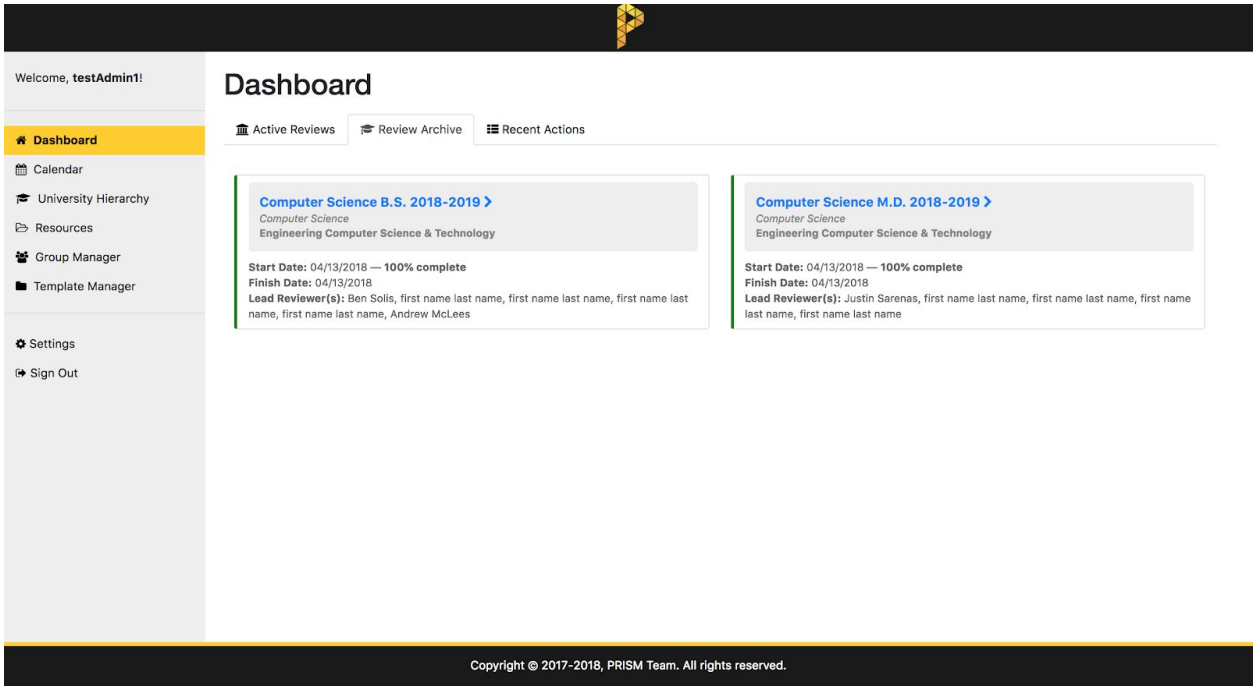
Active Reviews Tab

Tab consists of current active reviews the user has a role in. As an Administrator, they are allowed to edit lead reviewers, add and delete reviews.



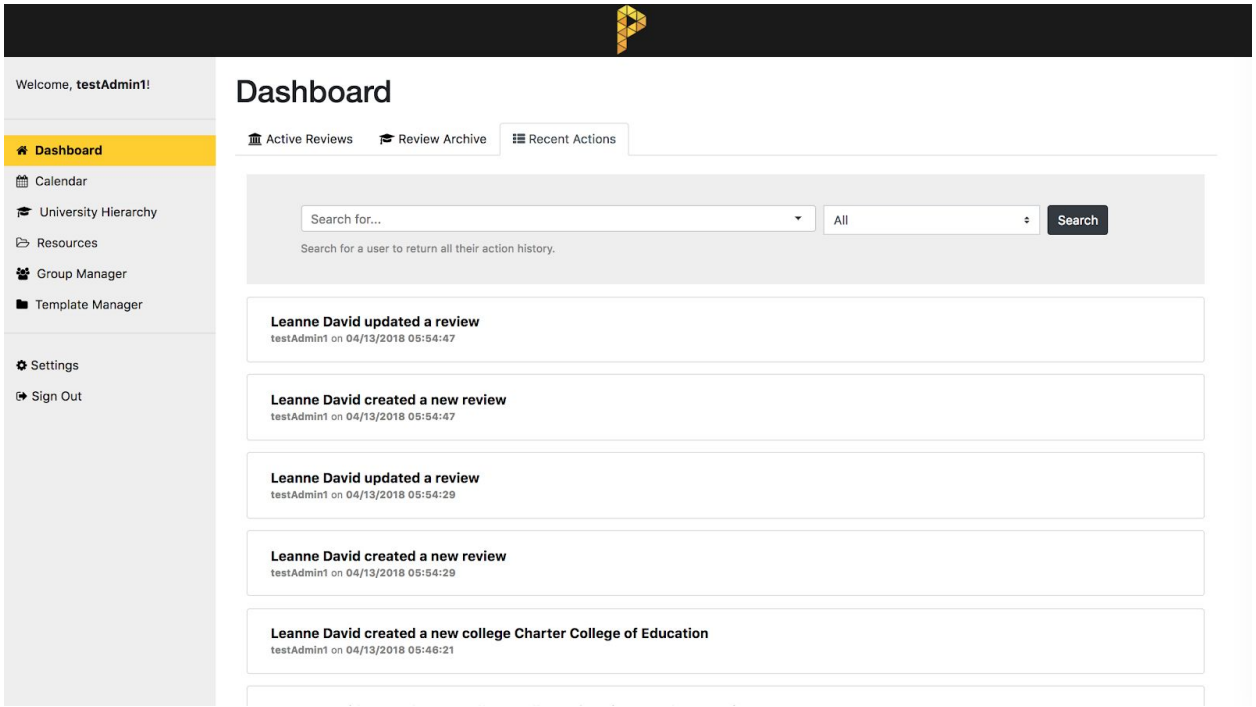
Review Archive Tab

Tab consists of previously completed reviews for archival purposes. Users may look at previous reviews as a template for an ongoing review.



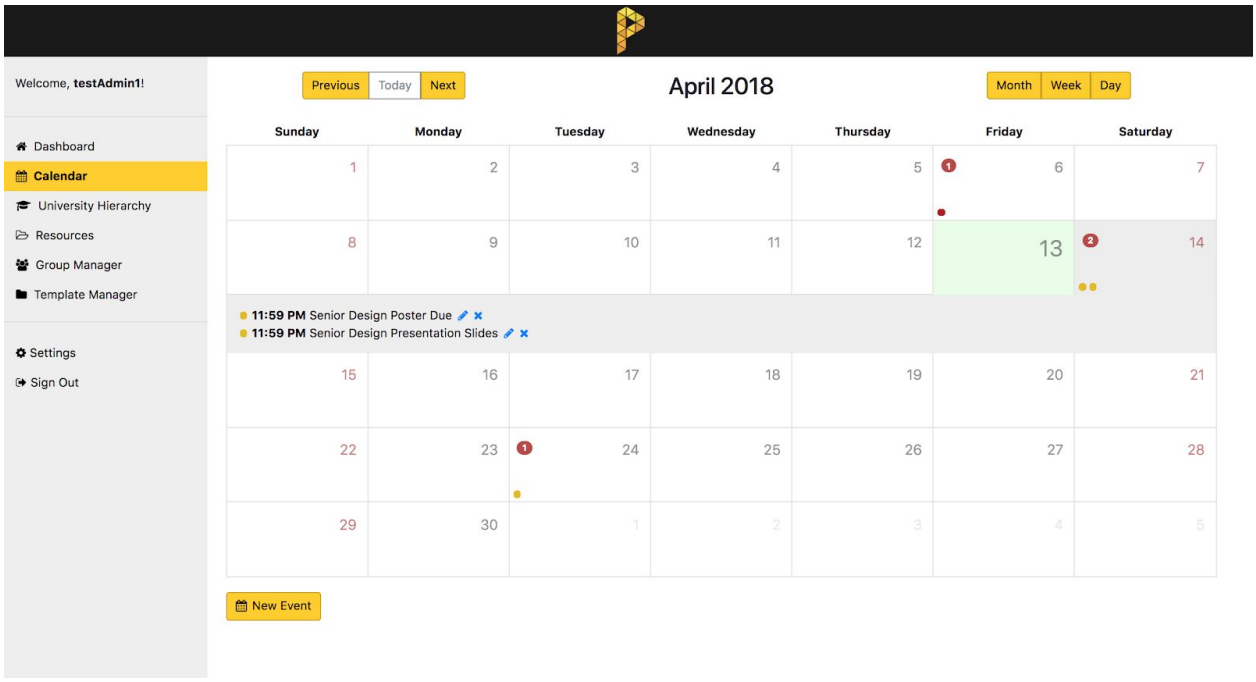
Recent Actions tab

Tab displays all the recent actions users have taken within the system such as file uploads, updates, and deletions.



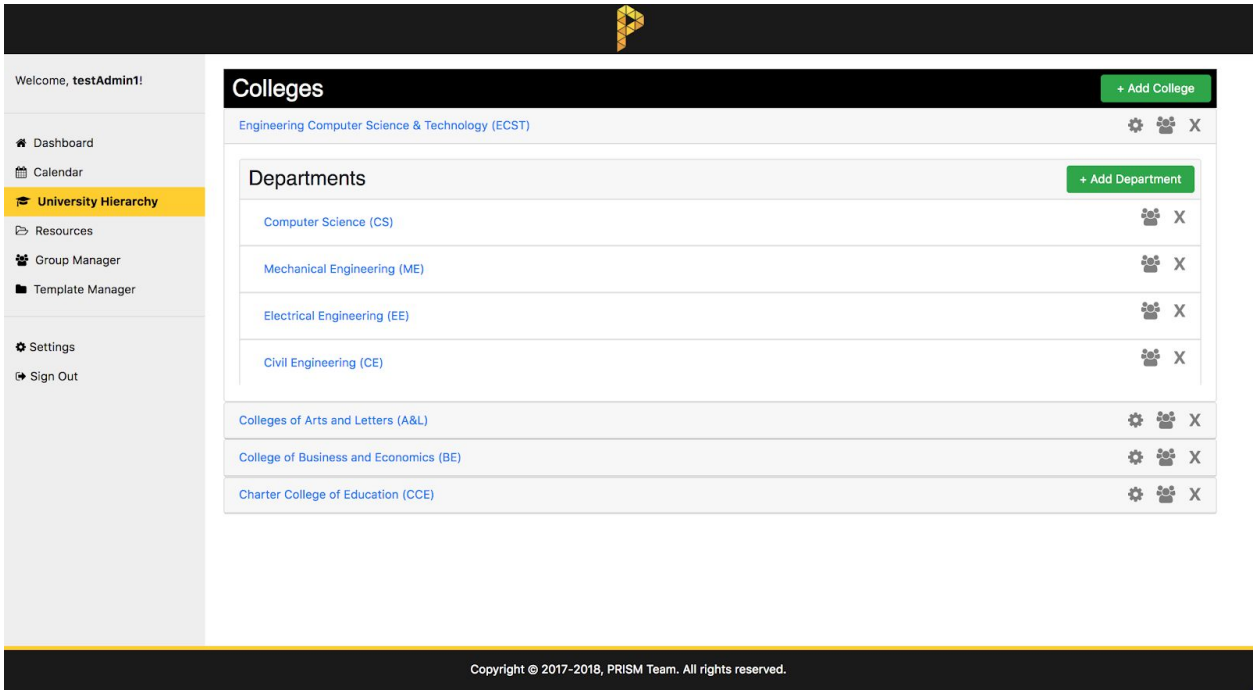
Calendar Component

Tab displays a calendar with upcoming events complete with email notifications.



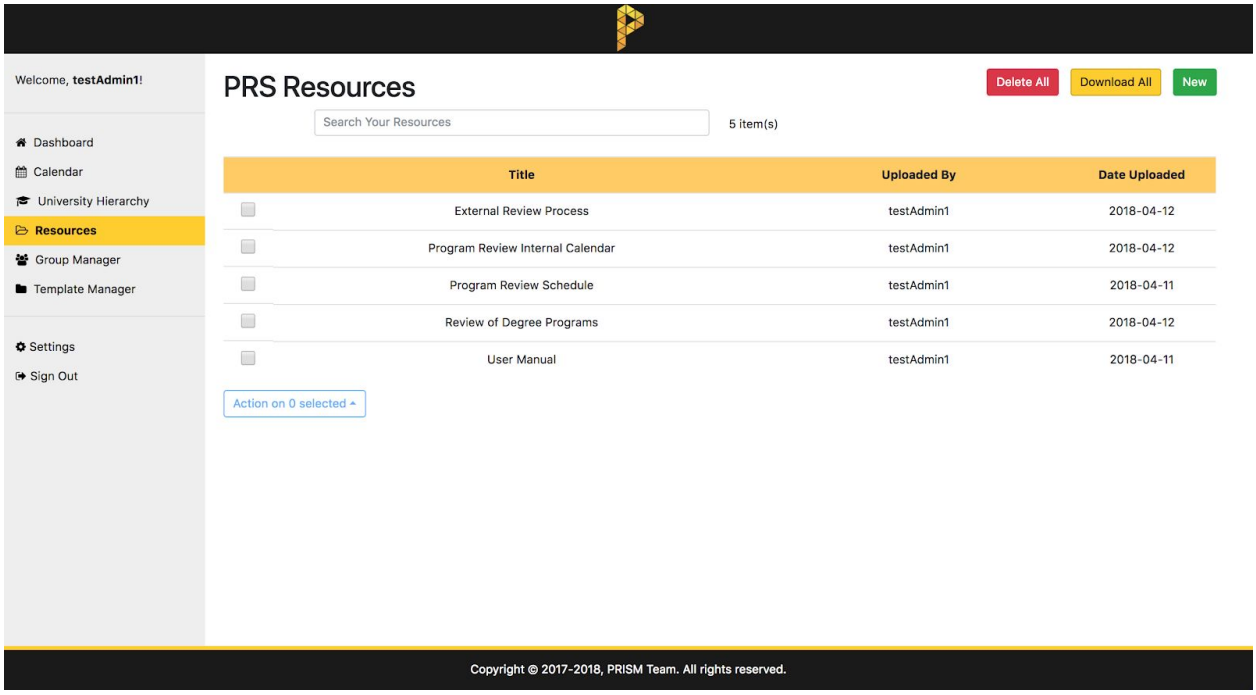
University Hierarchy Component

Tab consists of the hierarchy at CSULA. Users will be able to navigate through the different colleges, departments, and programs.



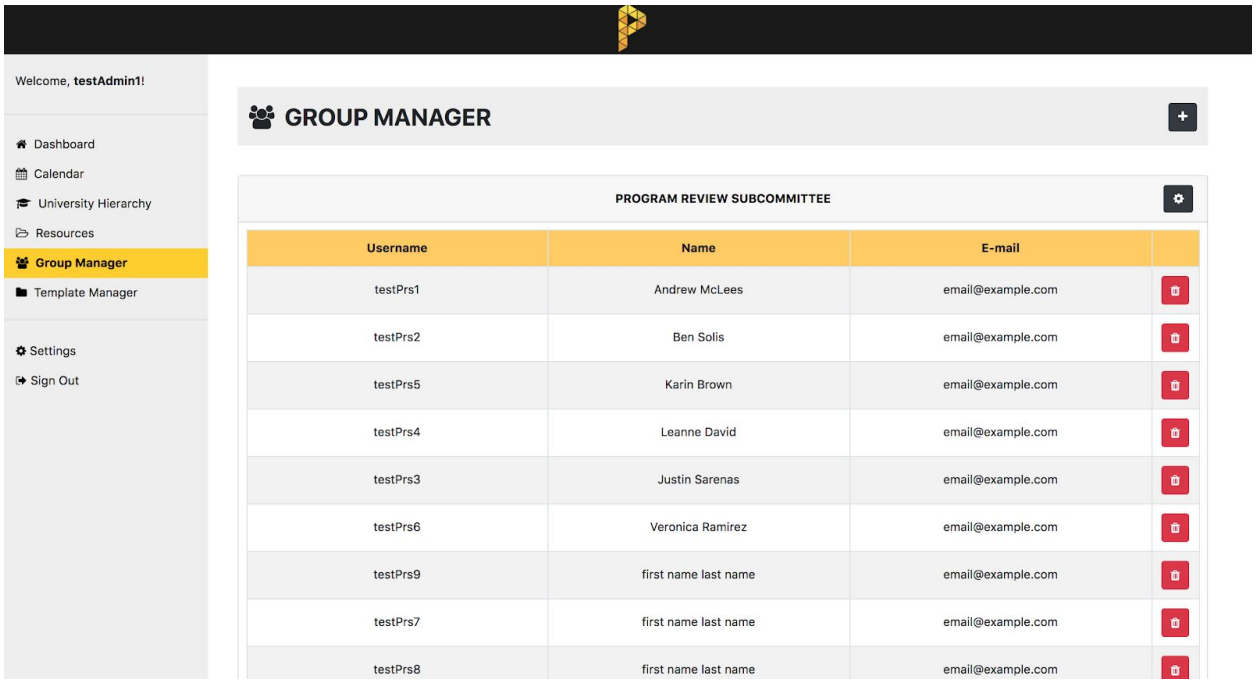
Resources Component

Tab consists of additional resources of documents users may need.



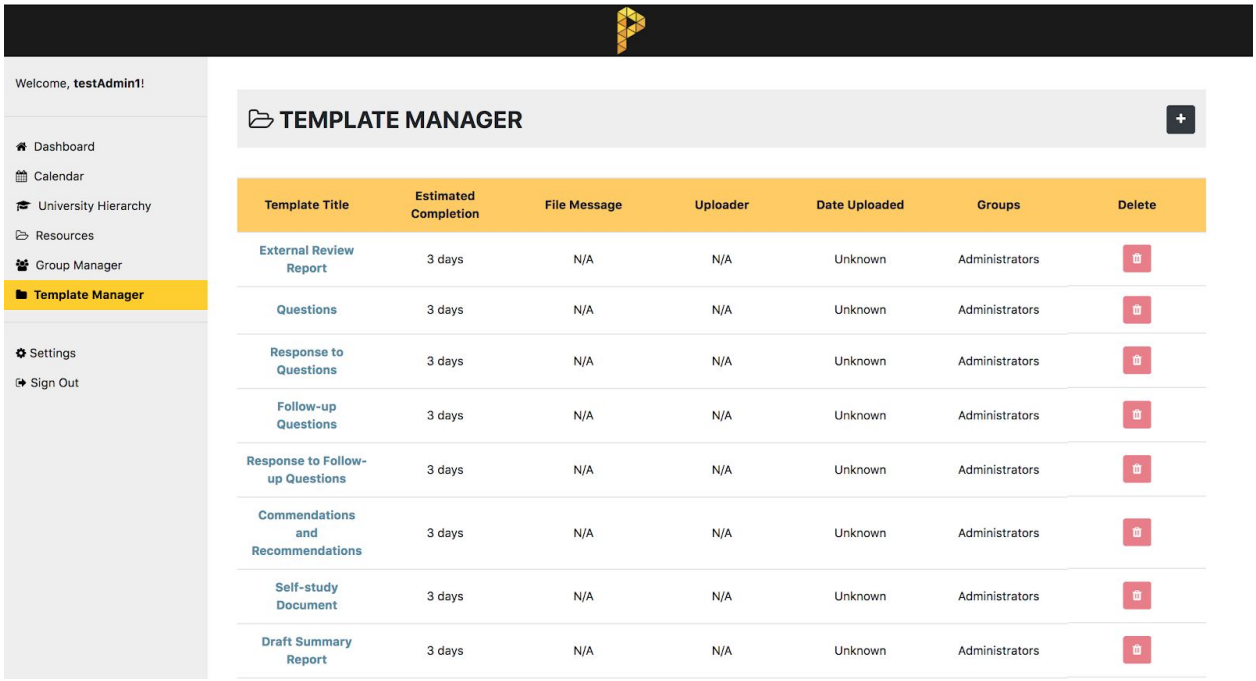
Group Manager Component

Tab consists of access control for the system. Administrators are able to add and remove users from groups. Groups restrict access to certain data within the system.



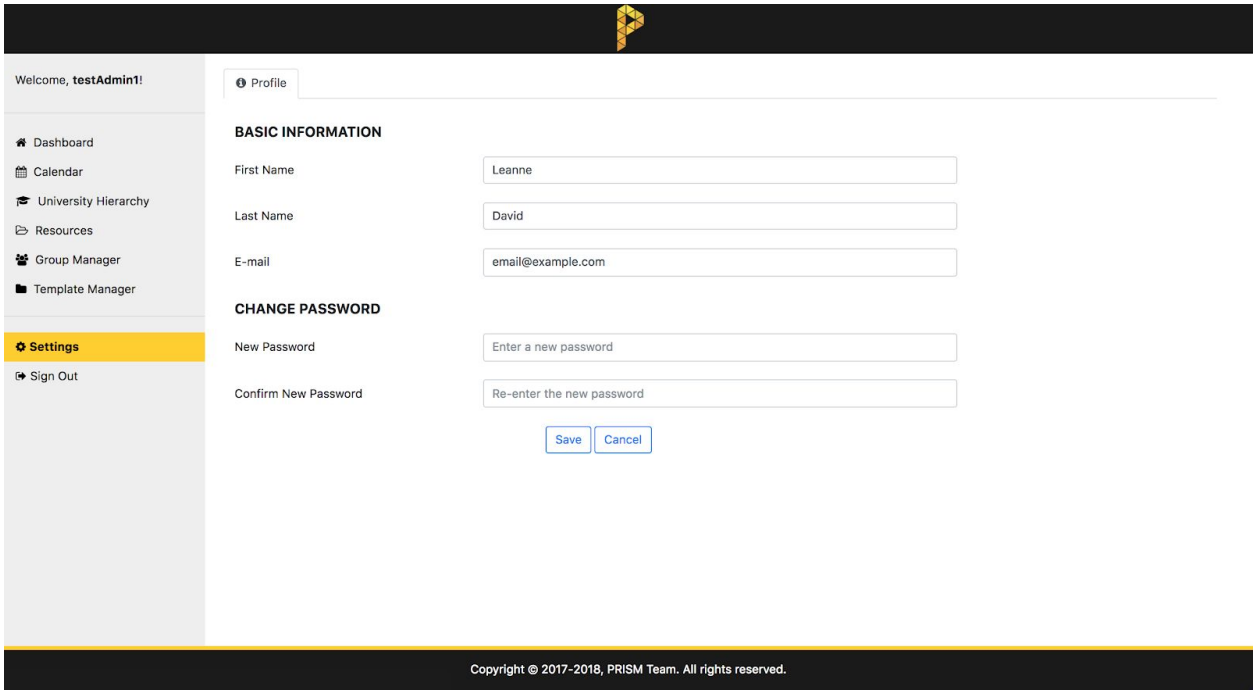
Template Manager Component

Tab consists of templates for previous reviews which users are able to download.



Settings Component

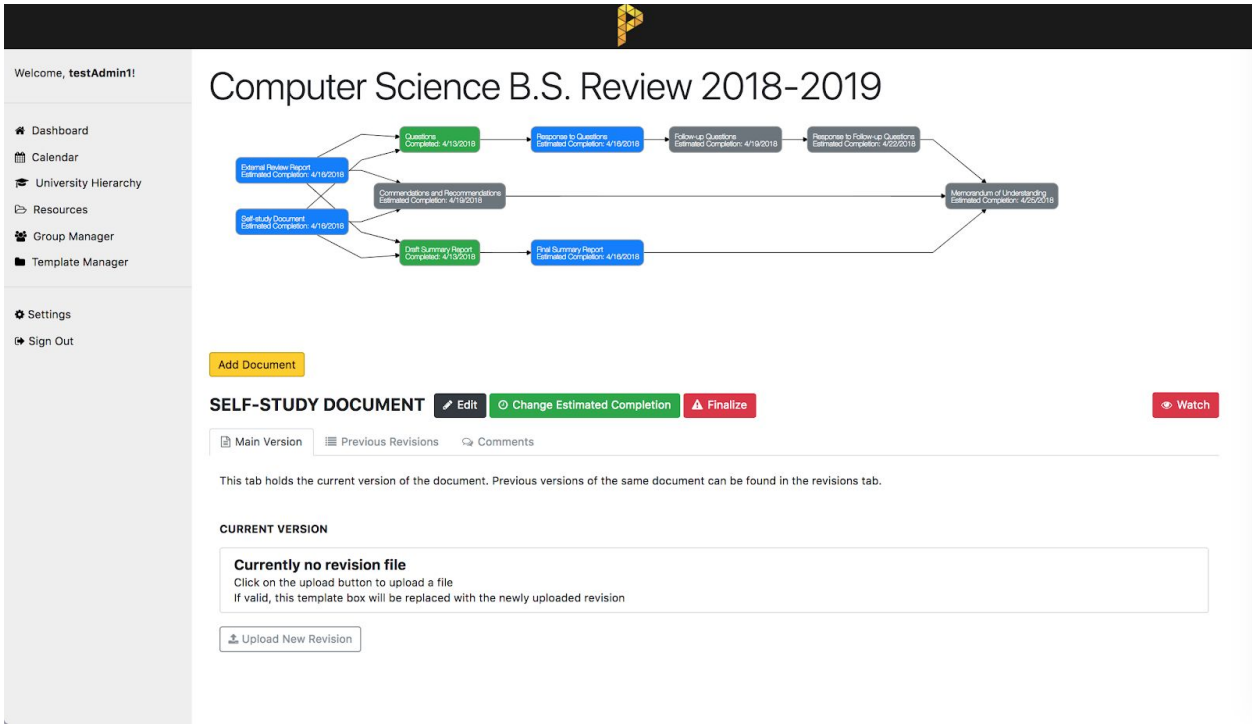
Tab consists of user configurations which users will be able to edit such as their name, email, and password.



Review Component

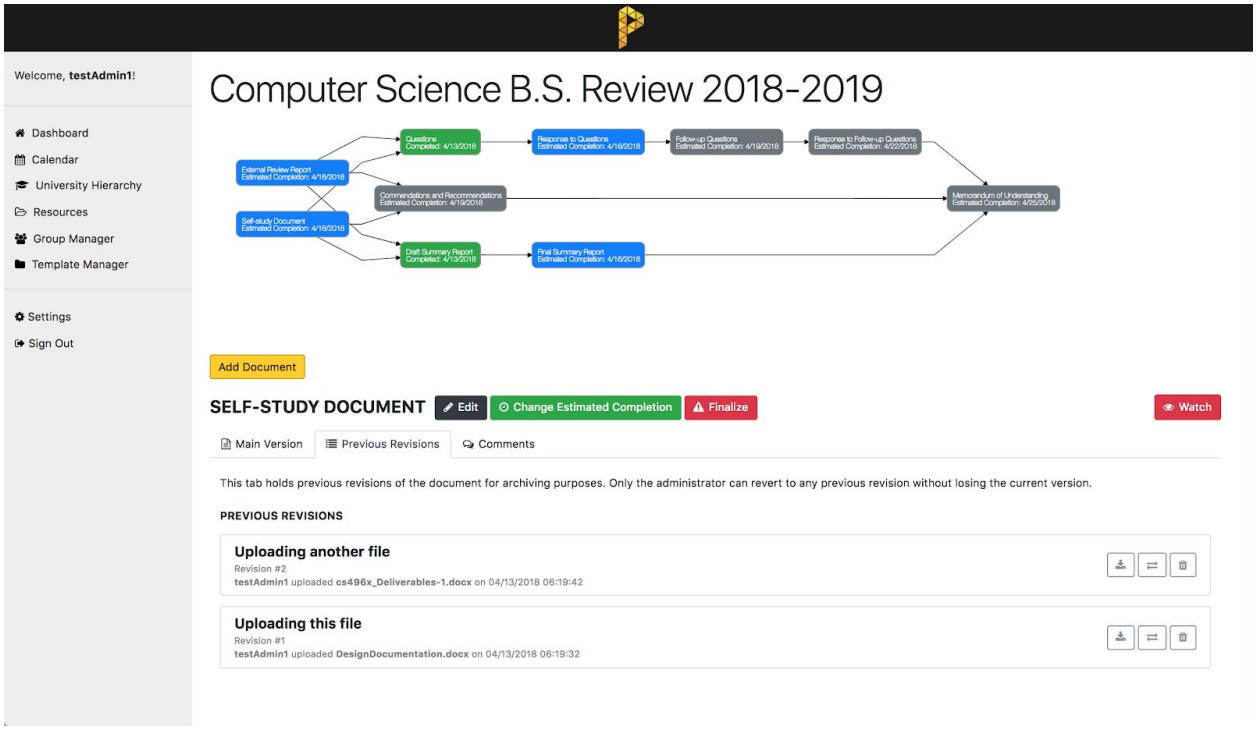
Document Component - Main Version Tab

Tab consists of the program review process. The main version displays the current accepted revision.



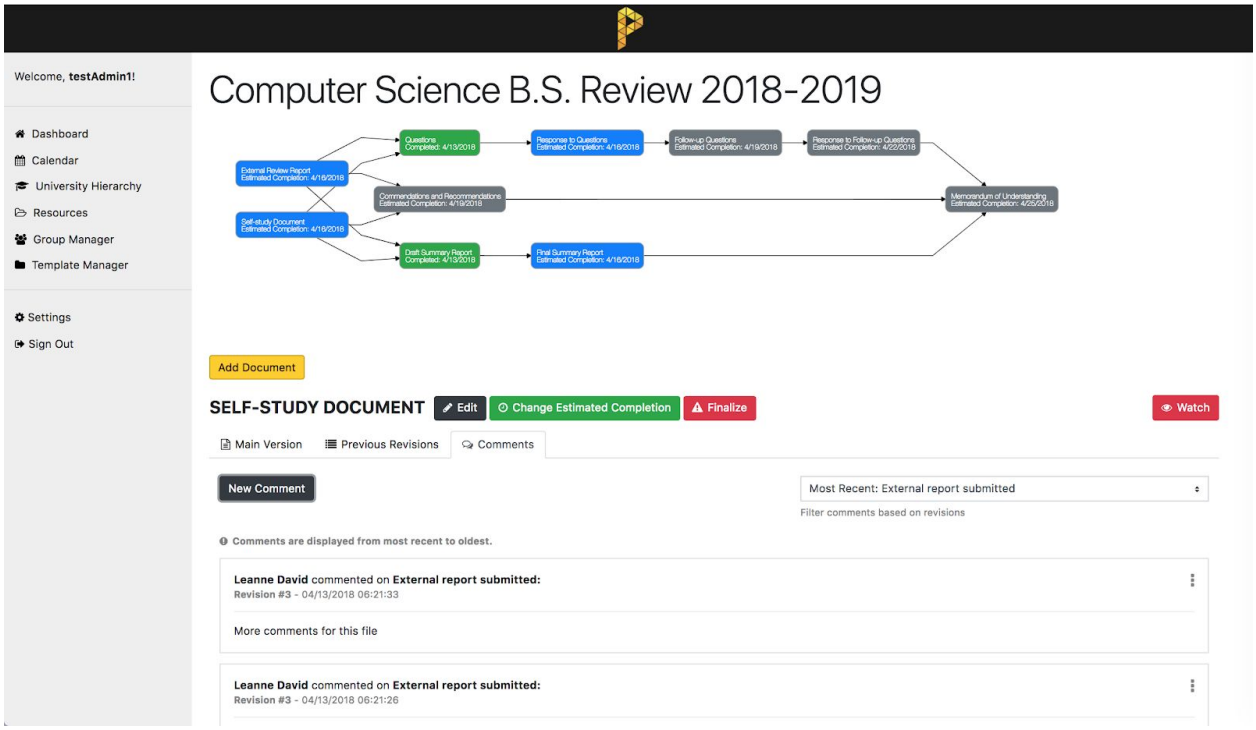
Document Component - Previous Revisions Tab

Tab displays all previous revisions that were uploaded. Administrator can delete and revert a previous revision as the current one.



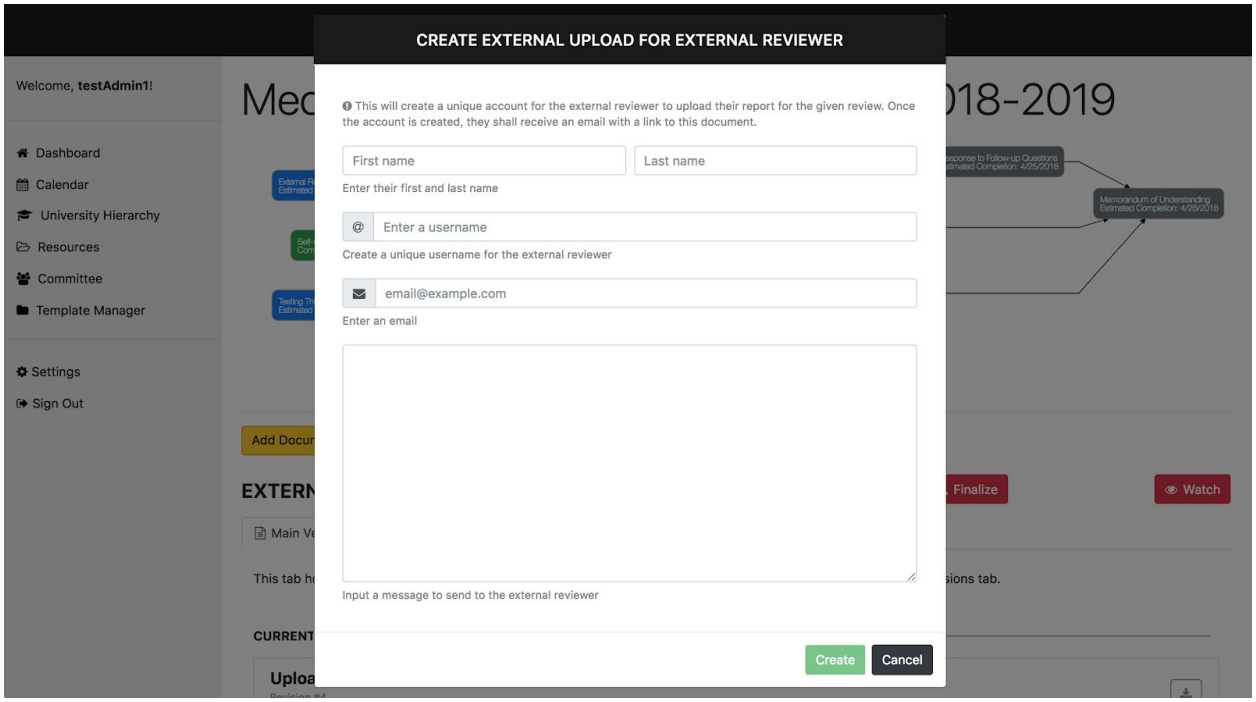
Document Component - Comments Tab

Tab displays the comments that are made by users under a certain document revision.



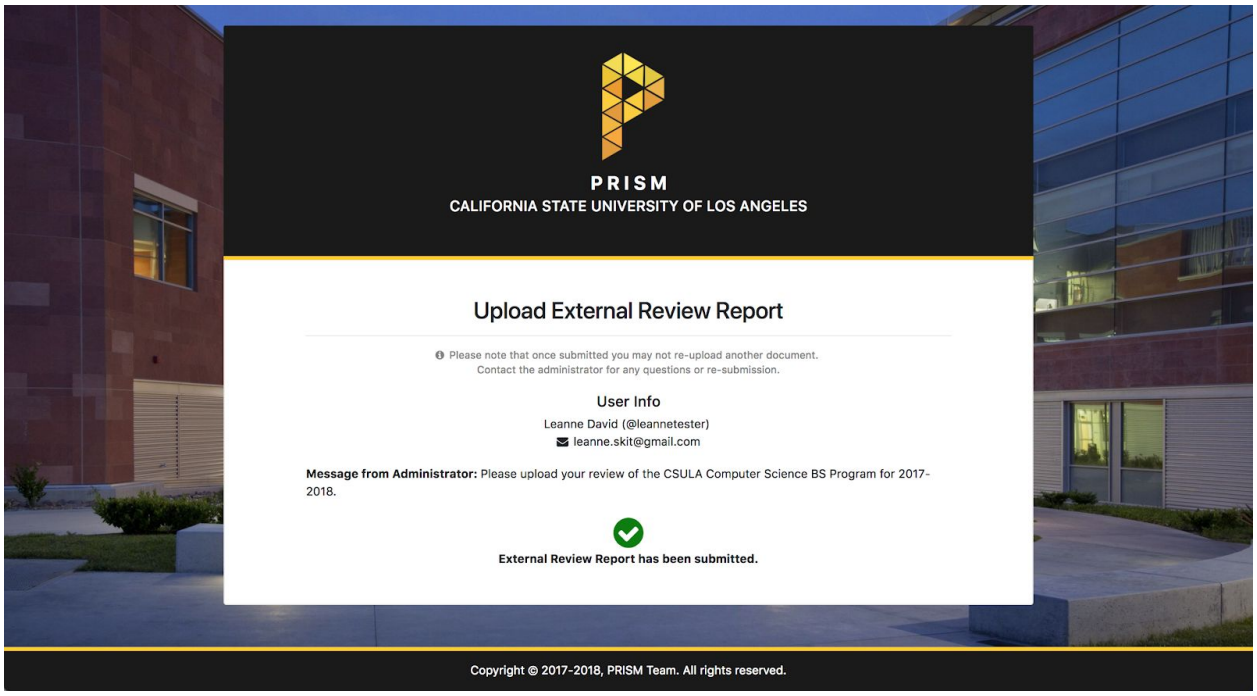
Document Component - Create External Upload Model

Tab displays form where administrators will be able to fill out in order to let an external reviewer access the site and upload their document.



External Upload Component

Page which allows external reviewers to upload their document.



Conclusion

PRISM is a web application developed for the PRS to use in order to streamline the processing of program reviews. The system was built using the MEAN stack; which is a popular trend for web app development, since its use of only one programming language, Javascript, throughout the stack makes development in a short time frame feasible. The system was split into three categories of modules: frontend modules that run on the client side, backend modules running on the server, and a static server to serve the client-side code. Furthermore, the server was split into a variety of interconnected modules to effectively meet the requirements of the project.

The main highlight of this project is the success of the review graph that was implemented with major challenges. Its implementation entailed adapting the d3.js library into the Angular platform. Another important highlight in the project is the success of using the ng-bootstrap library to replace the problematic PrimeNG library midway through the front-end development. PrimeNG failed to be the reliable user interface suite for Angular and made front-end development unreasonably difficult. The interaction between PRISM and shibboleth was not implemented due to administrative difficulties with the Cal State LA networking authorities.

The most important follow-up work this project needs is the implementation of adopting shibboleth for user authorization. This would allow users to sign in with their existing Cal State LA credentials. A useful add-on to incorporate into PRISM is a real-time in-browser document editor utilizing WebSockets. This would allow users to work on only one document until it is finalized; rather than working on different versions of the same document. Another useful capability to add on would be the customization of the review graph. This feature would allow the users to generate a review graph that reflects a more detailed level of the review process.

Chapter 5. References

The PRISM SRS and SRD were both references heavily in the creation of this document. The intent of this document is to consolidate the various materials about PRISM into a single document.

Andrew McLees's Honors College thesis on PRISM was referenced heavily in the writing of chapter 2 as it had similar goals to this document.