

**Software Design
Document
for
Program Review Information
System Management**

Version 2.0 approved

**Prepared by
David, Leanne
McLees, Andrew
Sarenas, Justin
Solis, Ben Jair**

For Graduate Studies at California State University, Los Angeles

April 14, 2018

Table of Contents.....	2
Revision History.....	
3	
1. Introduction.....	4
1.1. Purpose.....	4
1.2. Document Conventions.....	4
1.3. Intended Audience and Reading Suggestions.....	4
1.4. System Overview.....	5
2. Design Considerations.....	6
2.1. Assumptions and dependencies.....	6
2.2. General Constraints.....	6
2.3. Goals and Guidelines.....	7
2.4. Development Methods.....	7
3. Architectural Strategies.....	8
4. System Architecture.....	10
5. Policies and Tactics.....	13
5.1. Specific Products Used.....	13
5.2. Requirements traceability.....	13
5.3. Testing the software.....	13
6. Detailed System Design.....	14
7. Detailed Lower level Component Design.....	15
8. User Interface.....	16
9. Database Design.....	17
10. Requirements Validation and Verification.....	19
11. Glossary.....	20
12. References.....	20

Revision History

Name	Date	Reason For Changes	Version
Initial Version	12/8/2017		1
Final Version	4/14/2018	Completion of project	2

1. Introduction

1.1 Purpose

The purpose of this document is to describe the implementation details of the Program Review Information System Management (PRISM) as described in the Software System Requirements for Program Review Information System Management. This document describes the entire system and its implementation in detail.

1.2 Document Conventions

This document is formatted according the CSULA Computer Science department's 2017-2018 senior design software design document template.

1.3 Intended Audience and Reading Suggestions

This document is intended for individuals directly involved in the development of PRISM. This includes software developers, team managers, testers, and documentation writers. This document does not need to be read sequentially; users are encouraged to jump to any section they find relevant. Below is a brief overview of each part of the document.

- Part 1 (Introduction)
 - This section offers an overview of PRISM.
- Part 2 (Design Considerations)
 - Readers interested in the considerations accounted for while designing the system should consult this section.
- Part 3 (Architectural Strategies)
 - This section describes the design decisions and strategies used for the organization of the system and its higher-level structures.
- Part 4 (System Architecture)
 - This section provides a high-level overview of how the functionalities and responsibilities of the system were partitioned and then assigned to components.
- Part 5 (Policies and Tactics)
 - This section discusses any relevant political or tactical decisions made and the judgement used behind these decisions.
- Part 6 (Detailed System Design)
 - This section discusses the lower-level details of the system. Each system component is described in detail and may have an accompanying DFD.
- Part 7 (Detailed Lower Level Component Design)
 - This section discusses other lower-level details: components and subcomponents.
- Part 8 (Database Design)
 - This section covers all of the details related to the schema of the database.
- Part 9 (User Interface)
 - This section covers all of the details related to the structure of the graphical user interface (GUI). Readers can view this section for a tentative glimpse of what the final product will look like.
- Part 10 (Requirements Validation and Verification)

- This section details the requirements and the methods used to test that these requirements are met.
- Part 11 (Glossary)
 - Readers unfamiliar with software engineering terminology should consult this section.
- Part 12 (References)
 - This section includes any additional information which may be helpful to readers.

1.4 System Overview

PRISM is a workflow management tool including an optimized document storage and management system where users will be able to store and download all files related to program reviews.

PRISM is a full-stack web application that is accessed via a web browser. The system can be broken up into its backend and frontend side. The frontend side of the system provides users with an interface for PRISM; functionalities and user input are handled directly here. Whereas the backend side of the system offers CRUD operations on data in the database.

The following list summarizes the major functions of the system.

- 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 e-mail notifications upon changes
- Track which programs are due for review
- Send e-mail notifications upon events relevant to the user

2. Design Considerations

2.1 Assumptions and Dependencies

The following assumptions are being made in the design of PRISM.

- There will be a server capable of running the following software in accordance with the performance requirements of the software:
 - A MEAN stack as specified in section 3 of this document.
 - An e-mail server
- The server will be running a distribution of GNU/Linux
- The server will have sufficiently powerful hardware such that PRISM will not need to undergo extensive optimization to meet performance requirements
- The end users of this software will be limited to:
 - External reviewers
 - CSULA College Deans
 - CSULA Department Chairs
 - PRS members
 - PRS committee chairs/Provost appointees (Administrators)

2.2 General Constraints

End-user Environment

The browsers used by end-users must be supported. This will require additional E2E testing and polyfills in the Angular application to support certain browsers specified in the SRS.

Review Process

The review process has many irregularities and exceptions in timing which must be accounted for in designing time-based parts of the system.

Testing

Testing will be used to ensure reliable and quality of the software. The goal of testing will encourage splitting functionality into smaller modules.

Network Communication

The Angular application will communicate with the server via a REST API. This makes design of the REST API important.

Authentication with CSULA Credentials

As the system must allow users to log in with CSULA credentials, PRISM will need to interface with the university's Shibboleth identity provider server via SAML. This implies certain changes in the user model and the authentication flow.

2.3 Goals and Guidelines

Delivery by the End of the Senior Design Class

The PRS expects the software to be completed in time for it to be used as a part of the 2018-2019 program review process. This will require faster development methods to be utilized.

Emphasis on Simplicity

The user count of the software is likely to never reach 50 concurrent users. This means that performance on the server side will not be a large issue. If faced with situations such as authentication or development of the REST API where access list control or pagination could potentially increase complexity but improve performance or functionality not specified in the requirements, the functionality will only be implemented after all requirements are met. This emphasis is based on the notion that keeping the software simple will speed development.

Colors

As web developers, we believe color to be a critical component of any web application. As such, before designing a UI component, we ask ourselves: "What colors best illustrate the actions being performed?" We also consider look-and-feel consistency across the web to be important, so we chose to use Twitter Bootstrap as a basis for our UI color toolbox, taking advantage of its colors' position as the lingua franca of the modern web.

2.4 Development Methods

Due to the project's developers being students rather than full-time developers, it was not feasible to meet daily to discuss project progress. The project was organized into week-long sprints (planned at weekly meetings) with ad-hoc additional meetings and information exchange over instant messaging.

3. Architectural Strategies

Use of a particular type of product

PRISM will be using a MEAN stack which is composed of MongoDB, Express, Angular 4, and NodeJS. MongoDB will be interfaced with through Mongoose to assist with data organization and validation. PRISM will also use Twitter Bootstrap for UI components for Angular because it provides rich functionality integrated with Angular.

The primary alternative to a MEAN stack would have been a Java-based stack. The decision to use a MEAN stack was ultimately made because it provides a more responsive user experience, a more flexible software environment, and is widely used in industry. Stacks based in other languages are also possible but lack the market share and relative ease-of-configuration present in the options considered.

Reuse of existing software components to implement various parts/features of the system

We will not be reusing existing software components to implement various part/features of the system.

Future plans for extending or enhancing the software

Future plans for PRISM may include but are not limited to:

- A real time collaborative text editor

User interface paradigms (or system input and output models)

PRISM will be accessible through the user's web browser and will take in user input and output the data requested.

Hardware and/or software interface paradigms

PRISM will not have any hardware paradigms. TypeScript is a programming language which may be used to develop JavaScript applications. Javascript supports multiple programming paradigms for ease of development.

Error detection and recovery

Error handling is an integral part of PRISM's HTTP API. All requests contain extensive error handling for validation of user input, database failures, and other runtime exceptions. Errors propagate from the backend to the frontend and are also handled on the frontend, providing feedback usable by the user. All unexpected errors are logged so that any bug reports can be thoroughly investigated.

Memory management policies

Memory management is not necessary since our server will be able to store all data related to our system.

External databases and/or data storage management and persistence

No external databases or data storage management will be used.

Distributed data or control over a network

Distributed data will not be necessary. Control over a network is given to the administrators of the system which they can access through their Cal State LA user login and password.

Generalized approaches to control

Access is granted by the administrators with the user's Cal State LA credentials.

Concurrency and synchronization

PRISM will need synchronization with the commenting system in the case of two people commenting at the same time.

Communication mechanisms

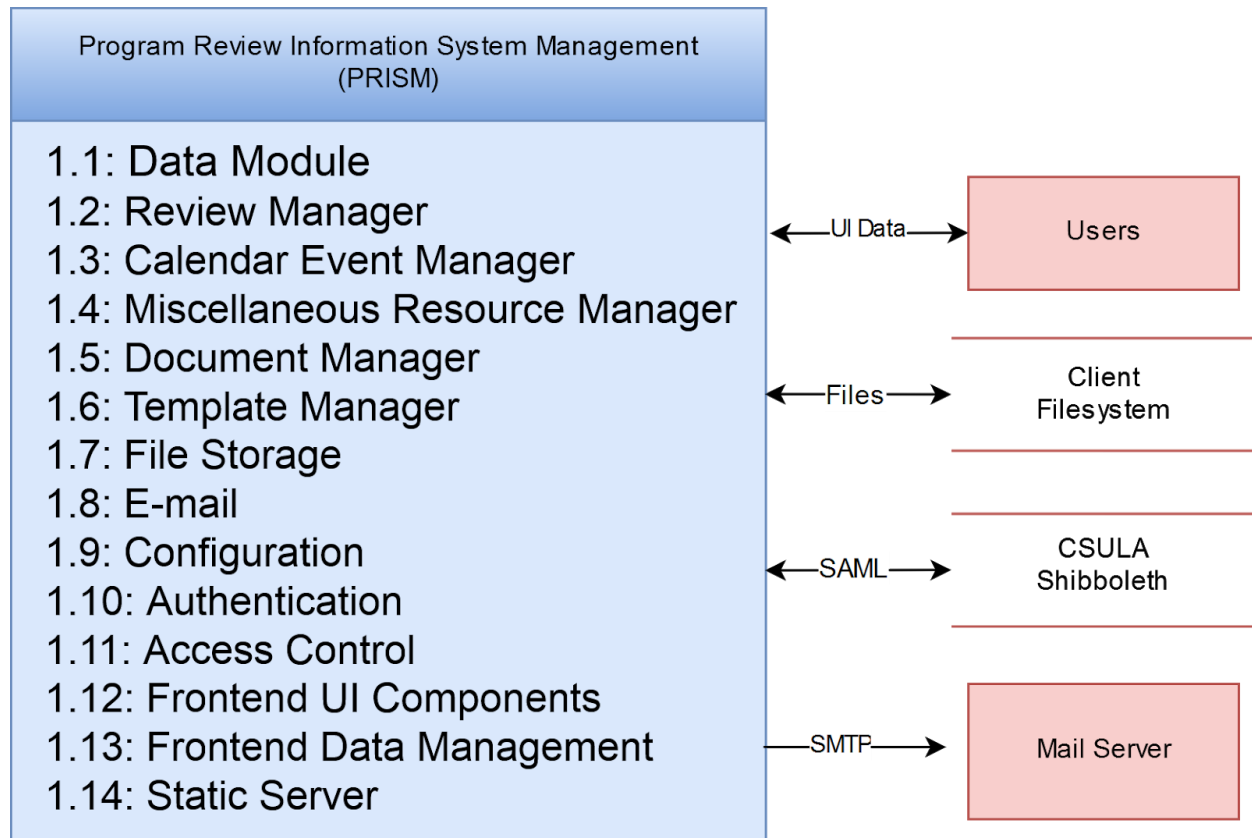
Communication is handled through the web browser.

Management of other resources

There will be no other resources that need to be managed.

4. System Architecture

DFD Level 0

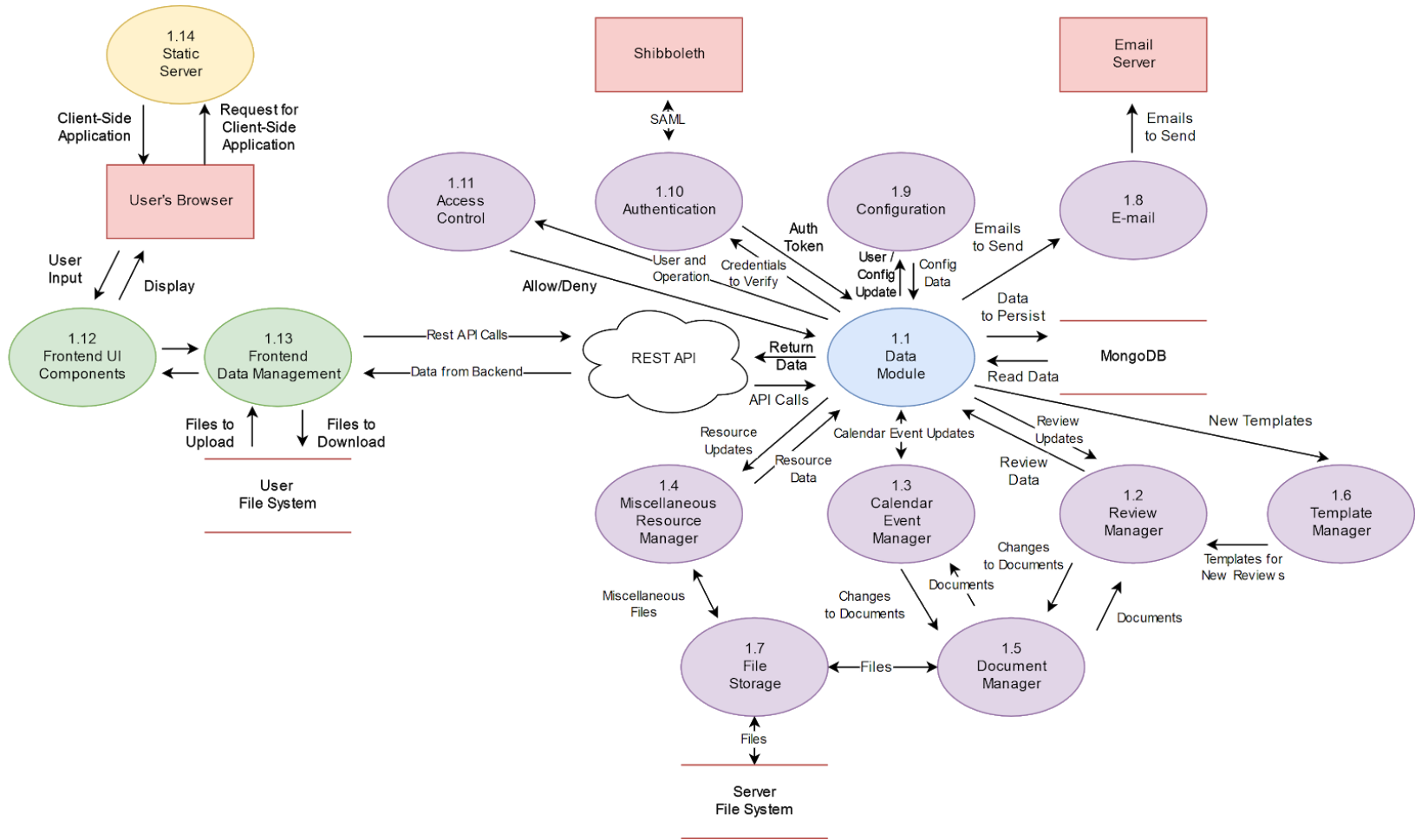


The system was split into modules based on functionality corresponding to categories of requirements (e.g. Review Manager, Calendar Event Manager, Miscellaneous Resource Manager) and functionality that can be separated from the rest of the code base for simplicity (e.g. Authentication, E-mail, etc.). The modules that run on the client side communicate with the server-side modules, and most of the data processing is handled on the server side.

- 4.1.1 Data Module
 - The Data Module handles persisting data to the database and connects all other modules.
- 4.1.2 Review Manager
 - The Review Manager tracks the state of each review.
- 4.1.3 Calendar Event Manager
 - The Calendar Event Manager tracks PRS meetings and their corresponding meeting agendas and minutes.
- 4.1.4 Miscellaneous Resource Manager

- The Miscellaneous Resource Manager handles files which must be made available to all PRS members (e.g. orientation slideshows).
- 4.1.5 Document Manager
 - The Document Manager handles revisioning and comments for each document stored in PRISM.
- 4.1.6 Template Manager
 - The Template Manager stores and provides templates for the documents involved in reviews.
- 4.1.7 File Storage
 - The File Storage manager handles storage of files on the server filesystem.
- 4.1.8 E-mail
 - The E-mail module handles sending of e-mail to users of the system upon events specified in the requirements occurring.
- 4.1.9 Configuration
 - The Configuration module tracks user and administrator configurations (e.g. e-mail preferences, naming preferences, etc.).
- 4.1.10 Authentication
 - The Authentication modules provides authentication for users both through PRISM directly and through university-provides credentials.
- 4.1.11 Access Control
 - The Access Control module determines whether a user has permission to request a resource or perform an action.
- 4.1.12 Frontend UI Components
 - The Frontend UI Component interact directly with the user, taking actions to execute and form data from the user.
- 4.1.13 Frontend Data Management
 - The Frontend Data Management module deals with data being sent and received on the client side. Building the client side with Angular 4 will allow some processing to be done on the client side to improve the user experience.
- 4.1.14 Static Server
 - The Static Server serves the client-side portion of PRISM to the user over HTTP(S).

DFD Level 1



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.

5. Policies and Tactics

5.1 Choice of which specific products used

The chosen development tools for which PRISM will be developed in are as follows:

- MEAN stack (Mongoose/MongoDB, Express, Angular 4, NodeJS)
- Atom text editor
- Angular CLI
- Clang-Format
- ESLint
- Twitter Bootstrap
- Passport

The main decision made for the software implementation was deciding whether to use Java or JavaScript-based technologies. Ultimately JavaScript was chosen because of its flexibility and the larger range of tools it provides. It is also used more often in current web development trends such as JavaScript frameworks with Angular 4 and its supporting libraries compared to Java servlets.

For writing the code, the text editor that will be used is Atom. Another text editor candidate was neovim but was set aside due to Atom having a smaller learning curve and being simpler to configure. Atom provides the means to download plugins to easily customize the workspace such as clang-format for beautifying and ESLint for linting; this was another factor in choosing Atom as the text editor for developing PRISM.

5.2 Plans for ensuring requirements traceability

Plans for tracking requirements include traversing through the list of requirements and ensuring each case is accounted for.

5.3 Plans for testing the software

Testing the software shall be executed using two testing frameworks—Protractor and Jasmine. Protractor is an end-to-end testing framework that uses test suites for Angular applications. These tests execute while the software is running on the browser and perform scenarios such as a user would. The test code shall be written using Jasmine because of the assertions and functions it provides. Using Protractor and Jasmine seems to be a popular combination which aided in making decisions on which testing frameworks to use.

6. Detailed System Design

6.1.1 Data Module

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

6.1.2 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. I

6.1.3 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.

6.1.4 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.

6.1.5 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.

6.1.6 Template Manager

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

6.1.7 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.

6.1.8 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.

6.1.9 Configuration

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

6.1.10 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.

6.1.11 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.

7. Detailed Lower level Component Design

Detailed lower level component design was completed alongside implementation due to the large number of changes that will be imposed by technical issues. Due to the large number of lower level components, they are detailed here according to the file structure of the project. Our goal in implementation was to have the code be self-documenting where possible. The organization of the code should be clear from the file structure, and comments are present where unapparent problems were encountered.

Folders are denoted with a /, and files have an extension. Comments begin with a #.

- prism-api/
 - # The backend repository
 - bin/
 - www # Entry point for the application generated by Express
 - create_programs.js # Script to import Programs, Colleges, and Departments
 - raw-programs.csv # Raw data used by the create_programs.js script
 - db_setup_development.js # Script to set up the database for development
 - db_setup_production.js # Script to set up the database for a production environment
 - lib/
 - config/
 - passport.js # Configuration file for Passport, the authentication package used by PRISM
 - settings.js # Global application settings container
 - cron/
 - cronexample.js # File used as a reference when creating email cron jobs
 - document_deadline.js # Cron job for sending notifications when documents near their deadlines
 - question_notification.js # Cron job for notifying parties in the department question exchange of a review
 - index.js # Import helper
 - templates/
 - # Folder for email templates
 - access.js # Contains generators for Express middleware for access control
 - action_logger.js # Handles access logging requests from endpoints
 - base_review.js # Defines the structure of a review
 - document_factory.js # Document factory
 - review_date_estimation.js
 - # Estimates completion dates of nodes in a Review
 - # Given n nodes in a Review, runs in O(n) time with no extra database requests
 - # O(c) memory (the Review itself is O(n))
 - review_factory.js # Review factory

- # Very database-intensive. Despite this, response times of 15ms were measured for the route that uses it (POST /review)
 - review_node.js # Type definition for review nodes
 - subscribe_middleware_factory # Factory for middleware that lets users subscribe to emails from any model
 - token_cache.js # API for in-memory cache of JWT tokens used to provide more granular control over authentication tokens (expiration and uniqueness)
 - user_factory.js # User factory
 - models/
 - # All models are defined here
 - # Models are registered with Mongoose for usage elsewhere
 - routes/
 - # All API endpoints are defined here
 - # Each file is named <name>.route.js
 - # Endpoints are registered with Express
 - # Access control is defined for each endpoint
 - specs/
 - # Automatic tests are defined here
 - .env # This file contains environmental variables for the application
 - app.js # The core of the application. It imports all modules and handles initialization and deinitialization
 - db.js # Handles database connection setup
 - error_handler.js # Handles HTTP responses and error logging for errors from endpoints
 - log.js # Configuration for Winston logger
 - package.json # Contains standard project metadata and dependencies
 - README.md # Readme for the project with setup information
 - teardown.js # Handles database teardown
- prism-frontend/
 - # The frontend repository
 - e2e/
 - # End to end tests
 - src/
 - # Application source code
 - app/
 - calendar/ # Calendar component
 - colleges/ # College, department, and program components
 - committee/ # PRS member directory component
 - dashboard/ # Dashboard component
 - document/ # Document component
 - external-upload/ # External upload component
 - group-manager/ # Group manager component
 - layout/
 - private/ # Layout component for authenticated users

- public/ # Layout component for unauthenticated users
- login/ # Login component
- models/
 - # All request/response models are in this folder
- page-not-found/ # 404 page component
- resources/ # Resources component
- review/ # Review component
- review-list/ # Review list component
- settings/ # User settings component
- shared/
 - # Shared source files
 - app.global.ts # Injectable global variable container
 - filter.pipe.ts # Text filter pipe for searching an array of objects with titles
 - shared.service.ts # Shared service for tracking the current user and caching response data
- template-manager/ # Template manager component
- user-selector/ # User selector component
 - # Allows selection of specific users as a reusable form element
- app-routing.module.ts # All routing configuration
- app.component.css # Application component CSS
- app.component.html # Application component HTML
- app.component.ts # Application component TypeScript
- app.module.ts # Application module definition
- index.html # Entry point for HTML
- main.ts # Entry point for application
- styles.css # Global CSS
- package.json # Contains standard project metadata and dependencies

8. Database Design

See the PRISM requirements specification document for an overview of the PRISM database design. Detailed database design is not complete; below is a list of models currently implemented.

The following Mongoose schemas have been implemented. All Mongoose schemas have a MongoDB ObjectId associated with them by default; the schemas below are not an exception. Square brackets ([]) indicate arrays in Mongoose schemas. ObjectId fields are indicative of references to other objects. What these references are to is apparent by their field name.

- Action
 - text: String
 - date: Date
 - user: ObjectId
 - type: string
 - object: ObjectId
- College
 - name: String
 - abbreviation: String
 - deans: [ObjectId]
- Department
 - name: String
 - abbreviation: String
 - college: ObjectId
 - chairs: [ObjectId]
- Document
 - title: String
 - revision: Array of
 - message: String
 - filename: String
 - originalFilename: String
 - dateUploaded: Date
 - uploader: ObjectId
 - template: Boolean
 - comments: Array of
 - text: String
 - author: Array of
 - _id: ObjectId
 - username: String
 - name: String
 - createDate: Date
 - revision: Number
 - originalFilename: String
 - subscribers: ObjectId

- template: Boolean
 - coreTemplate: Boolean
 - completionEstimate: Number
 - locked: Boolean
 - groups: String
- Event
 - title: String
 - date: Date
 - canceled: Boolean
 - documents: ObjectId
 - groups: ObjectId
 - people: ObjectId
- External Upload
 - token: String
 - message: String
 - user: ObjectId
 - document: ObjectId
 - completed: Boolean
- Group
 - name: String
 - members: [ObjectId]
 - access: Boolean
- Program
 - name: String
 - department: ObjectId
 - nextReviewDate: Date
- Resource
 - title: String
 - files: Array of
 - message: String
 - filename: String
 - originalFilename: String
 - dateUploaded: Date
 - uploader: ObjectId
 - groups: ObjectId
- Review
 - program: ObjectId
 - startDate: Date
 - finishDate: Date
 - externalUploads: [ObjectId]
 - leadReviewers: [ObjectId]
 - endNodes: [ObjectId]
 - nodes: Object mapping ObjectId to:
 - startDate: Date
 - finishDate: Date

- completionEstimate: Number
 - finishDateOverridden: Boolean
 - finalized: Boolean
 - document: ObjectId
 - prerequisites: [ObjectId]
 - title: String
- deleted: Boolean
- User
 - username: String
 - email: String
 - name:
 - first: String
 - last: String
 - internal: Boolean
 - root: Boolean
 - groups: [ObjectId]
 - disabled: Boolean
 - config: Array of
 - email: Array of
 - documentFinalized: Boolean
 - newComment: Boolean
 - meetingChange: Boolean
 - samlType: String
 - passwordHash: String

9. User Interface

9.1 Overview of User Interface

The system provides various components to aid users in the completion of their tasks to the overall program review process. Any internal user will be able to access certain Reviews, Dashboard, Calendar, Resources, Committee, and Settings at any page via side menu.

9.1.1 Dashboard

The dashboard offers quick access to program review tools. Users can easily navigate to an active program review, archived reviews, and recent actions.

9.1.1.1 Active Reviews

This tab allows administrators to create new program reviews. Non-admins are limited to making progress on an existing review to which they are assigned. An alert message informs the administrators when there are not any active reviews in existence.

9.1.1.2 Review Archive

This tab lists all past program reviews and allows users to access their data. An alert message informs the administrators when there are not any archived reviews in existence.

9.1.1.3 Recent Actions

This tab lists useful information about the user's contribution history and access to related data. An alert message informs users when there are not any archived reviews. Administrators have exclusive information displayed about other user's contribution and equipped with a search bar to find a specified user's contribution.

9.1.2 Calendar

This calendar tool allows users to create events for PRS meetings with notifications containing an attached file for the event. The calendar contains marks that represent events that are active in yellow and canceled events in red. Event details are displayed in a pop up window.

9.1.3 University Hierarchy

This tool is for administrators to manipulate college, departments, and program level data stored in the system. This component is not displayed to non-administrators.

9.1.4 Group Manager

This tool allows administrators to create and manage the grouping of members of the PRS or administrators. An accordion is used to handle the data with panel titles displaying group names and the panel contents holding the roster information.

9.1.5 Document

Review documents are accessible by clicking on a node on the flowchart of a program review. New documents can be added by clicking the add document button and providing form data. New documents created will appear in the flowchart as separate nodes. These documents can be tracked and untracked by users, marked as finalized and have their estimated completion time

extended for deadline extensions. The document is partitioned into three tabs: main version, previous version, and comments. The documents are all downloadable regardless the version.

9.1.5.1 Main Version

This tab holds information about the latest document in use for a program review. Users are able to update the latest document used via upload or reverting from a previous version.

9.1.5.2 Previous Revisions

This tab holds information about previous documents used in a program review. The users can delete, or revert to this documents as the main version.

9.1.5.3 Comments

This tab holds comments directed toward a revision and allows users to edit or delete these comments.

9.1.6 Resources

This tool allows users to download resource files and create new resources. Multiple resource deletion and download is supported. A dynamic search bar allows users to filter the resources and a counter helps users keep track of the correct amount of files or resources are stored in the system.

9.1.7 Review

Program reviews are displayed in a colored flowchart that has completed documents in green and non completed documents in gray. Documents currently being worked on are colored blue. The flowchart contains clickable nodes that represent key stages in the program review process. These nodes will display document information below the chart when the user clicks on a node

9.1.8 Settings

This component allows users to configure profile settings such as their own password and username.

9.1.9 Template Manager

This tool allows administrators to create templates for documents that will be used in program reviews. Templates are archived as new revisions are uploaded. Selecting from the list of templates will take administrators to a page for uploading the template file or revision file.

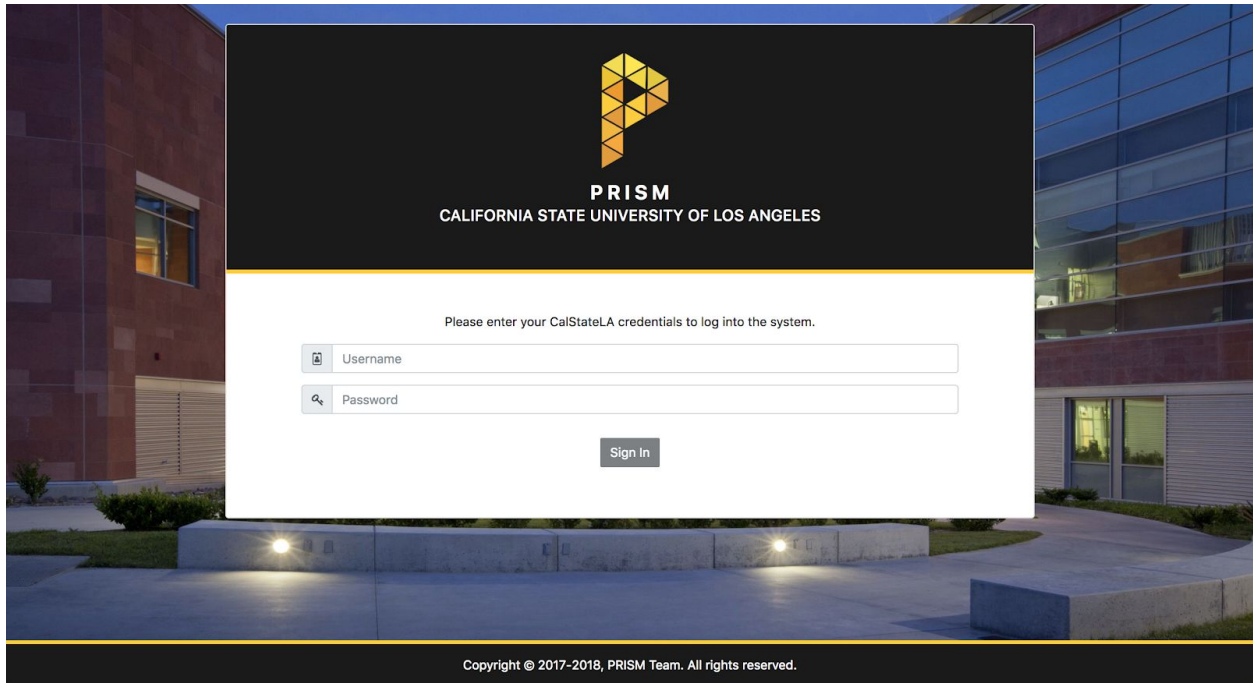
9.1.10 Committee

This interface displays a table of all PRS members and their email.

9.2 Screen Frameworks or Images

9.2.1 Login Component


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

The image shows a login interface for the PRISM system at California State University of Los Angeles. The interface is overlaid on a background image of a modern building at dusk. The login form has a dark header with the PRISM logo and the text 'PRISM CALIFORNIA STATE UNIVERSITY OF LOS ANGELES'. Below the header, there is a prompt: 'Please enter your CalStateLA credentials to log into the system.' The form contains two input fields: 'Username' and 'Password'. The 'Password' field has a small icon of a key and a toggle for password visibility. A 'Sign In' button is located below the input fields. At the bottom of the form, there is a copyright notice: 'Copyright © 2017-2018, PRISM Team. All rights reserved.'

9.2.2. Dashboard

9.2.2.1 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.



Welcome, testAdmin!

Dashboard

Active Reviews Review Archive Recent Actions

[+ Create Review](#)

Computer Science B.S. 2018-2019

Computer Science
Engineering Computer Science & Technology

Start Date: 04/13/2018 — 20% complete
Finish Date: 04/25/2018
Lead Reviewer(s): Justin Sarenas

Edit Lead Reviewers
Delete Review

Mechanical Engineering B.S. 2018-2019

Mechanical Engineering
Engineering Computer Science & Technology

Start Date: 04/13/2018 — 10% complete
Finish Date: 05/01/2018
Lead Reviewer(s): Karin Brown, Veronica Ramirez, Leanne David, Justin Sarenas

Edit Lead Reviewers
Delete Review

Civil Engineering B.S. 2018-2019

Civil Engineering
Engineering Computer Science & Technology

Start Date: 04/13/2018 — 40% complete
Finish Date: 04/16/2018
Lead Reviewer(s): Andrew McLees, Ben Solis, Leanne David, Justin Sarenas

Edit Lead Reviewers
Delete Review

Art B.S. 2018-2019


Art
Colleges of Arts and Letters

Start Date: 04/13/2018 — 30% complete
Finish Date: 04/22/2018
Lead Reviewer(s): Andrew McLees

Edit Lead Reviewers
Delete Review

9.2.2.2 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.



Welcome, testAdmin!

Dashboard

Active Reviews Review Archive Recent Actions

Computer Science B.S. 2018-2019

Computer Science
Engineering Computer Science & Technology

Start Date: 04/13/2018 — 100% complete
Finish Date: 04/13/2018
Lead Reviewer(s): Ben Solis, first name last name, first name last name, first name last name, first name last name, Andrew McLees

Computer Science M.D. 2018-2019

Computer Science
Engineering Computer Science & Technology

Start Date: 04/13/2018 — 100% complete
Finish Date: 04/13/2018
Lead Reviewer(s): Justin Sarenas, first name last name, first name last name, first name last name, first name last name

Copyright © 2017-2018, PRISM Team. All rights reserved.

9.2.2.3 Recent Actions tab

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

Welcome, testAdmin1!

Dashboard

Calendar

University Hierarchy

Resources

Group Manager

Template Manager

Settings

Sign Out

Dashboard

Active Reviews

Review Archive

Recent Actions

Search for...

All

Search

Search for a user to return all their action history.

Leanne David updated a review

testAdmin1 on 04/13/2018 05:54:47

Leanne David created a new review

testAdmin1 on 04/13/2018 05:54:47

Leanne David updated a review

testAdmin1 on 04/13/2018 05:54:29

Leanne David created a new review

testAdmin1 on 04/13/2018 05:54:29

Leanne David created a new college Charter College of Education

testAdmin1 on 04/13/2018 05:46:21

9.2.3 Calendar Component

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

Welcome, testAdmin1!

Dashboard

Calendar

University Hierarchy

Resources

Group Manager

Template Manager

Settings

Sign Out

Previous

Today

Next

April 2018

Month

Week

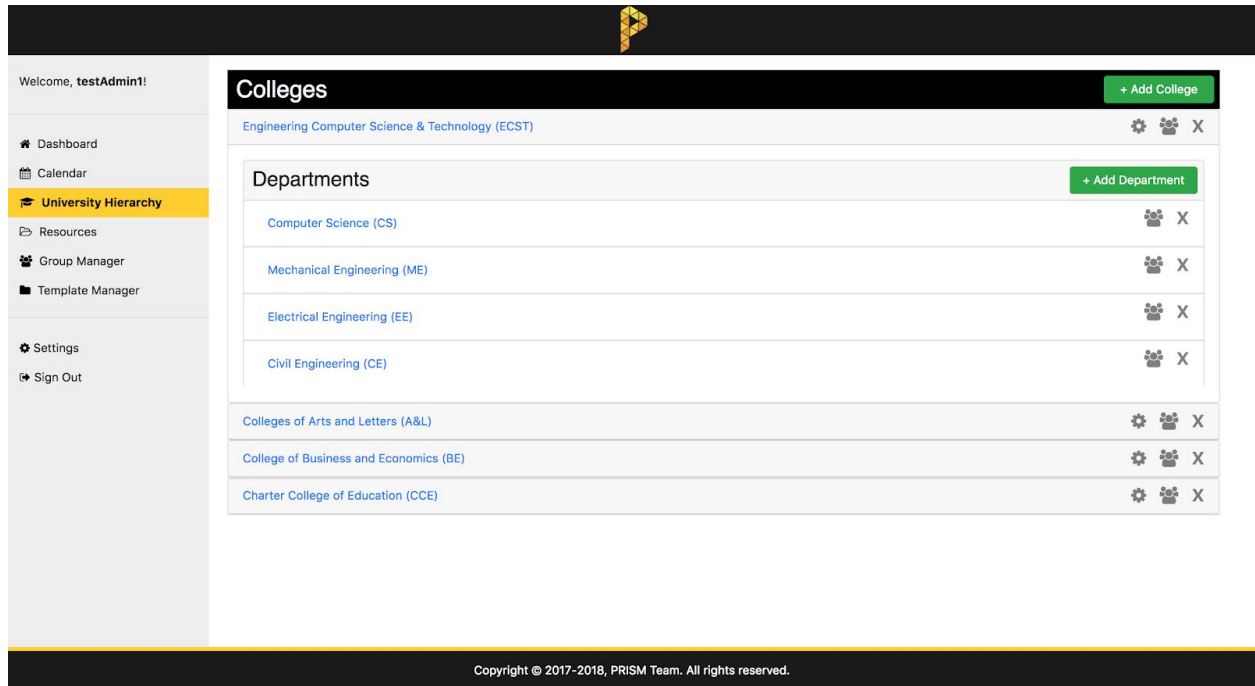
Day

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2	3	4	5	6	7
8	9	10	11	12	13	14
<div> <div>11:59 PM Senior Design Poster Due</div> <div>11:59 PM Senior Design Presentation Slides</div> </div>						
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

New Event

9.2.4 University Hierarchy Component

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



Welcome, testAdmin1!

Colleges + Add College

Engineering Computer Science & Technology (ECST)

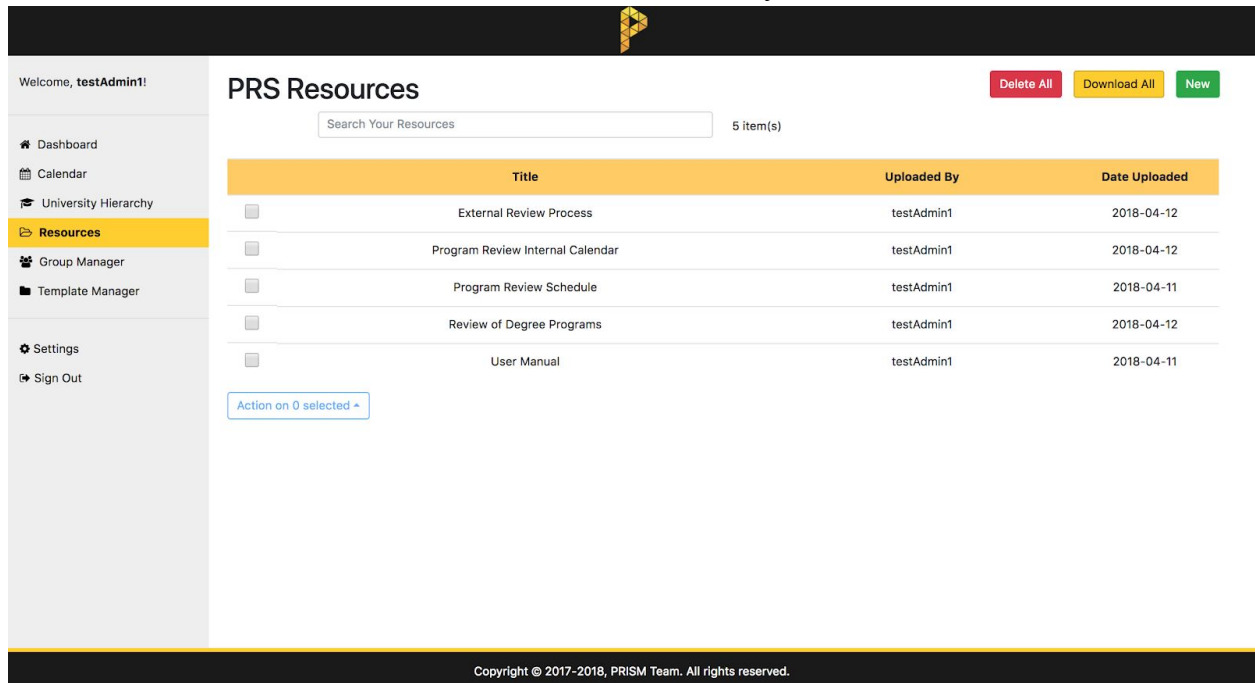
Departments + Add Department

Computer Science (CS)			
Mechanical Engineering (ME)			
Electrical Engineering (EE)			
Civil Engineering (CE)			
Colleges of Arts and Letters (A&L)			
College of Business and Economics (BE)			
Charter College of Education (CCE)			

Copyright © 2017-2018, PRISM Team. All rights reserved.

9.2.5 Resources Component

Tab consists of additional resources of documents users may need.



Welcome, testAdmin1!

PRS Resources Delete All Download All New

Search Your Resources 5 item(s)

	Title	Uploaded By	Date Uploaded
<input type="checkbox"/>	External Review Process	testAdmin1	2018-04-12
<input type="checkbox"/>	Program Review Internal Calendar	testAdmin1	2018-04-12
<input type="checkbox"/>	Program Review Schedule	testAdmin1	2018-04-11
<input type="checkbox"/>	Review of Degree Programs	testAdmin1	2018-04-12
<input type="checkbox"/>	User Manual	testAdmin1	2018-04-11

Action on 0 selected

Copyright © 2017-2018, PRISM Team. All rights reserved.

9.2.6 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.

Welcome, testAdmin!

Dashboard
Calendar
University Hierarchy
Resources
Group Manager
Template Manager

Settings
Sign Out

GROUP MANAGER

PROGRAM REVIEW SUBCOMMITTEE

Username	Name	E-mail	
testPrs1	Andrew McLees	email@example.com	
testPrs2	Ben Solis	email@example.com	
testPrs5	Karin Brown	email@example.com	
testPrs4	Leanne David	email@example.com	
testPrs3	Justin Sarenas	email@example.com	
testPrs6	Veronica Ramirez	email@example.com	
testPrs9	first name last name	email@example.com	
testPrs7	first name last name	email@example.com	
testPrs8	first name last name	email@example.com	

9.2.7 Template Manager Component

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

Welcome, testAdmin!

Dashboard
Calendar
University Hierarchy
Resources
Group Manager
Template Manager

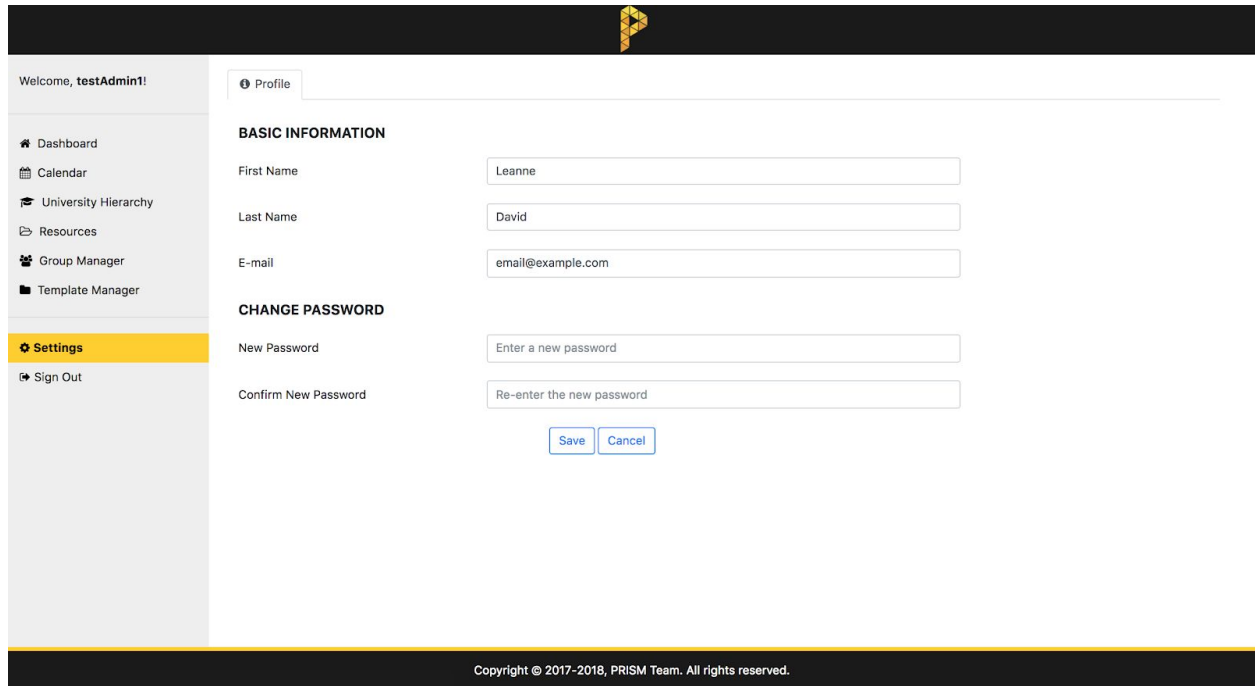
Settings
Sign Out

TEMPLATE MANAGER

Template Title	Estimated Completion	File Message	Uploader	Date Uploaded	Groups	Delete
External Review Report	3 days	N/A	N/A	Unknown	Administrators	
Questions	3 days	N/A	N/A	Unknown	Administrators	
Response to Questions	3 days	N/A	N/A	Unknown	Administrators	
Follow-up Questions	3 days	N/A	N/A	Unknown	Administrators	
Response to Follow-up Questions	3 days	N/A	N/A	Unknown	Administrators	
Commendations and Recommendations	3 days	N/A	N/A	Unknown	Administrators	
Self-study Document	3 days	N/A	N/A	Unknown	Administrators	
Draft Summary Report	3 days	N/A	N/A	Unknown	Administrators	

9.2.8 Settings Component

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



The screenshot shows a web application interface for user profile management. At the top, a dark header bar contains a yellow 'P' logo. Below the header, a sidebar on the left lists navigation options: 'Welcome, testAdmin1!', 'Dashboard', 'Calendar', 'University Hierarchy', 'Resources', 'Group Manager', 'Template Manager', 'Settings' (highlighted in yellow), and 'Sign Out'. The main content area is titled 'Profile' and contains two sections: 'BASIC INFORMATION' and 'CHANGE PASSWORD'. The 'BASIC INFORMATION' section has three input fields: 'First Name' (containing 'Leanne'), 'Last Name' (containing 'David'), and 'E-mail' (containing 'email@example.com'). The 'CHANGE PASSWORD' section has two input fields: 'New Password' (containing 'Enter a new password') and 'Confirm New Password' (containing 'Re-enter the new password'). Below these fields are two buttons: 'Save' and 'Cancel'. At the bottom of the page, a dark footer bar contains the text 'Copyright © 2017-2018, PRISM Team. All rights reserved.'

9.2.9. Review Component

9.2.9.1. Document Component - Main Version Tab

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

9.2.9.3. Document Component - Comments Tab

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

The screenshot shows the 'Comments Tab' for a document titled 'Computer Science B.S. Review 2018-2019'. The interface includes a sidebar with navigation options like Dashboard, Calendar, University Hierarchy, Resources, Group Manager, Template Manager, Settings, and Sign Out. The main content area displays a flowchart of the review process, starting from 'External Review Report' and 'Self-study Document' through various steps like 'Questionnaire', 'Response to Questions', 'Follow-up Questions', 'Response to Follow-up Questions', 'Recommendations and Recommendations', 'Draft Summary Report', and 'Final Summary Report', all leading to a 'Memorandum of Understanding'. Below the flowchart, there are buttons for 'Add Document', 'Edit', 'Change Estimated Completion', 'Finalize', and 'Watch'. The 'Comments' tab is selected, showing a 'New Comment' button and a dropdown menu for 'Most Recent: External report submitted'. A message indicates that comments are displayed from most recent to oldest. Two comments by 'Leanne David' are visible, both commenting on 'External report submitted' for Revision #3 on 04/13/2018 at 06:21:33 and 06:21:26 respectively.

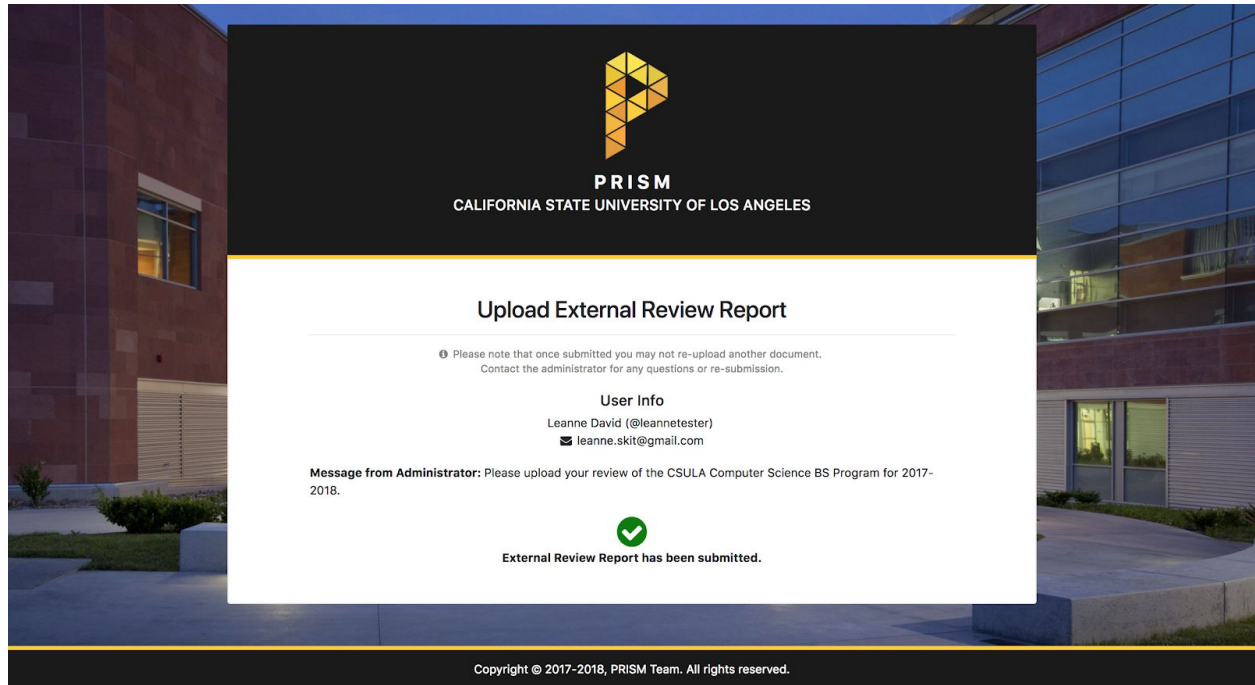
9.2.9.4 Document Component - Create External Upload Model

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.

The screenshot shows the 'Create External Upload for External Reviewer' form. The form is titled 'CREATE EXTERNAL UPLOAD FOR EXTERNAL REVIEWER' and includes a message: 'This will create a unique account for the external reviewer to upload their report for the given review. Once the account is created, they shall receive an email with a link to this document.' The form fields include: 'First name' and 'Last name' (with a note 'Enter their first and last name'), 'Enter a username' (with a note 'Create a unique username for the external reviewer'), 'email@example.com' (with a note 'Enter an email'), and a large text area for 'Input a message to send to the external reviewer'. At the bottom, there are 'Create' and 'Cancel' buttons. The background shows the same sidebar and flowchart as the previous screenshot.

9.2.10. External Upload Component

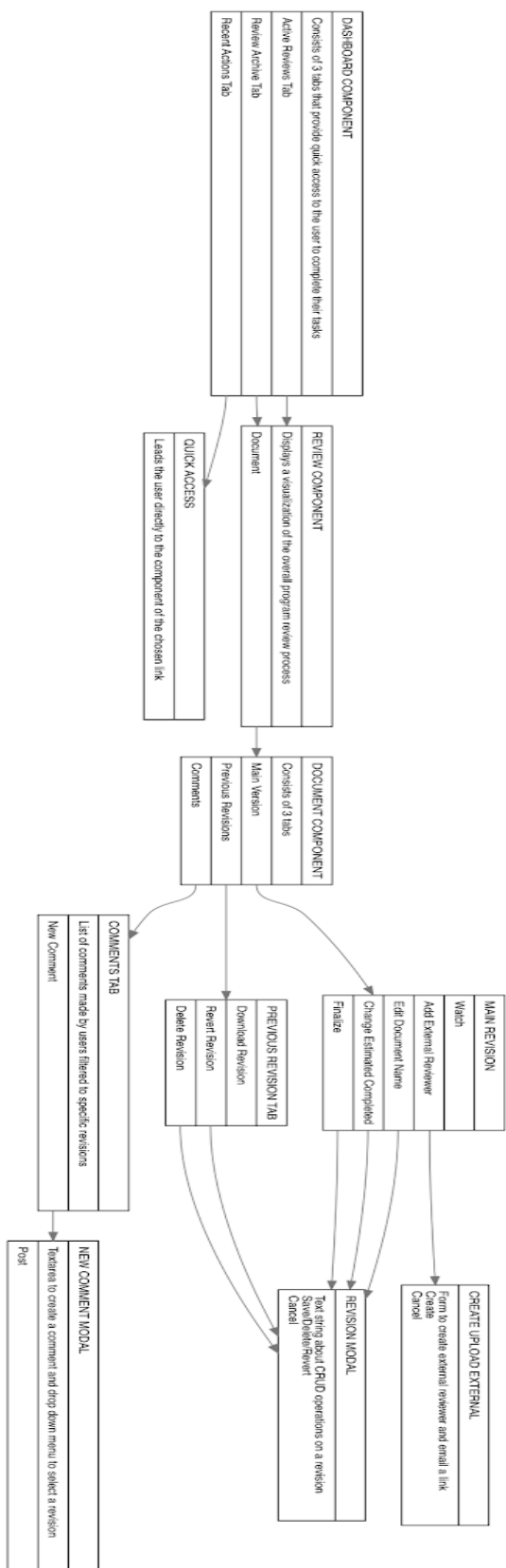
Page which allows external reviewers to upload their document.



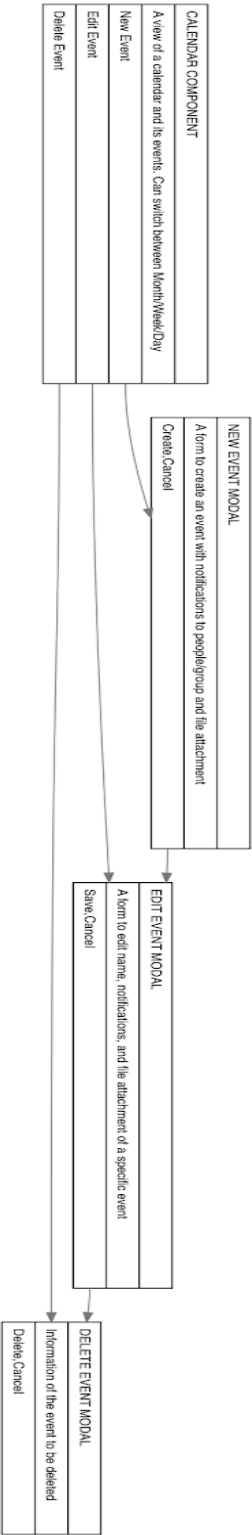
9.3 User Interface Flow Model

The user interface is as stateless as possible where there are no sessions stored for the sake of simplicity. The UI flow consists entirely of selecting menu buttons which are then followed by the appropriate modals. Any actions that require creating, manipulating, or deleting data will update the database and are logged to the overall history of actions done by users. This will provide quick shortcuts to users that will directly link them to where their last actions were taken.

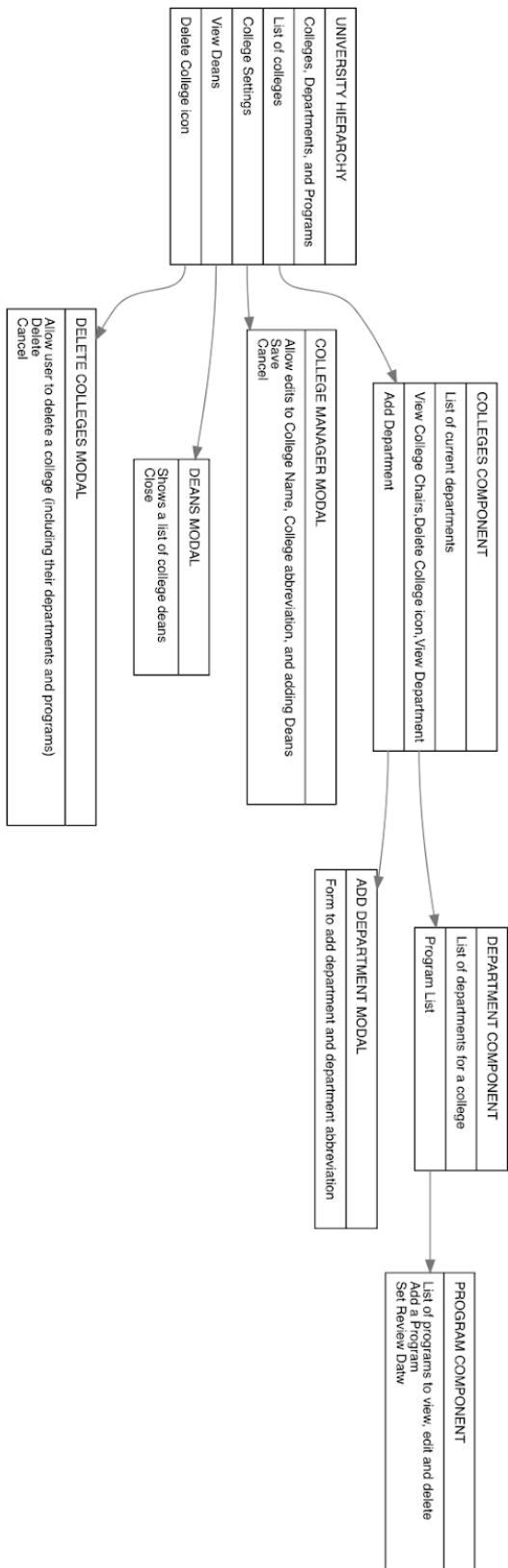
9.3.1 Dashboard UI Flow Model



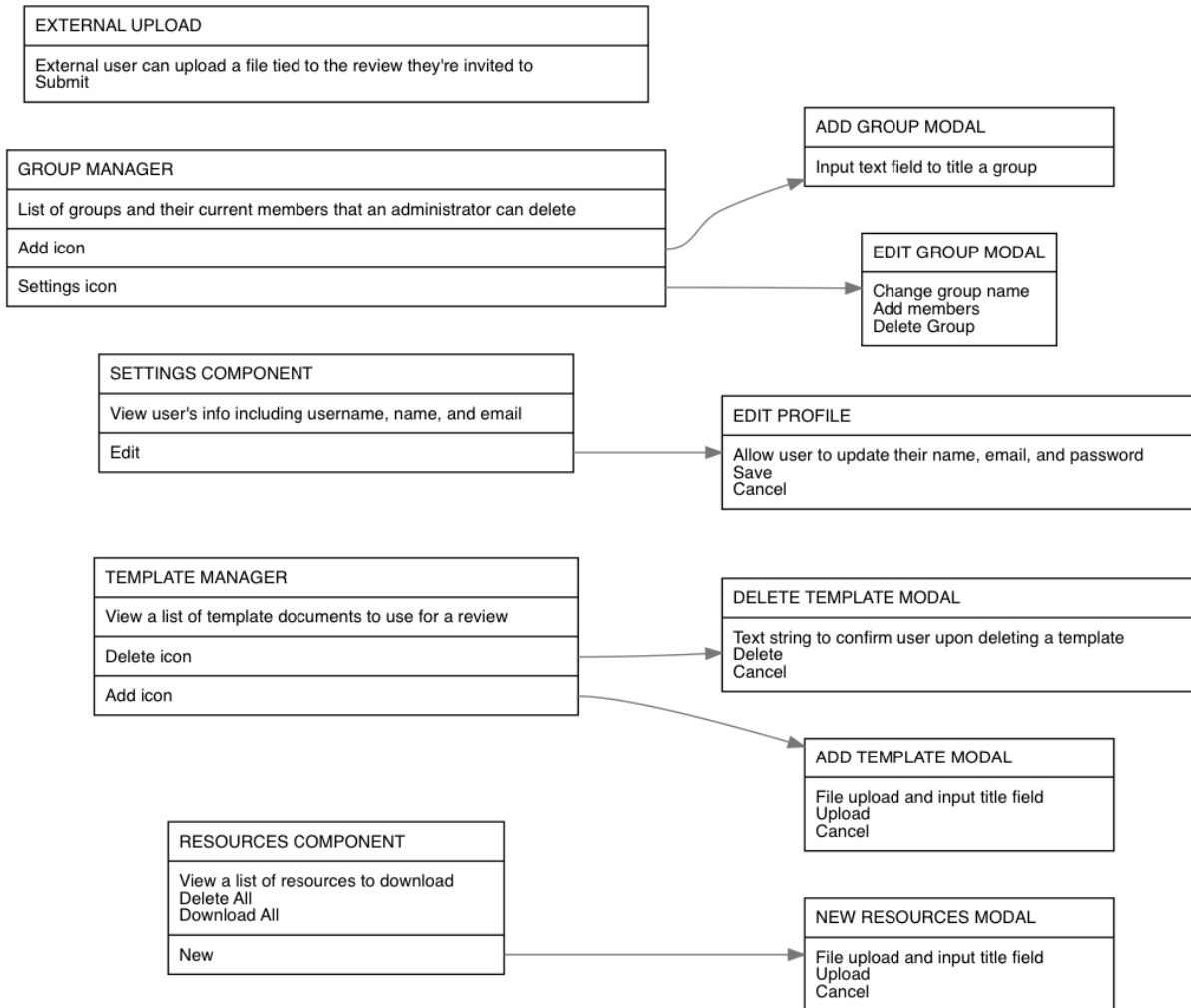
9.3.2 Calendar UI Flow Model



9.3.3 University Hierarchy UI Flow Model



9.3.4 External Upload, Group Manager, Settings, Template Manager, and Resources UI Flow Model



10. Requirements Validation and Verification

Requirements Related to the Review Process (1.1, 1.2, ...)		
Requirement No.	Requirement Description	V&V Methodology
1.1	The administrator shall be able to create a new review.	Demonstration
1.2	The administrator shall be able to access any review.	Demonstration
1.3	The administrator shall be able to update any review.	Demonstration
1.4	The administrator shall be able to delete any review.	Demonstration
1.5	The system shall track the status of each document pertaining to a review.	Testing
1.6	The system shall display the status of a review.	Demonstration
1.7	The system shall display the documents of a review.	Demonstration
1.8	The system shall track the estimated completion date of each document.	Analysis
1.9	The system shall allow lead reviewers and administrators to change the estimated completion date of each document.	Testing
1.10	When a review is initiated, the system shall import the templated documents to be part of the review's documents.	Testing
1.11	The system shall allow users to upload non-standard documents.	Demonstration
1.12	The system shall allow users to access non-standard documents.	Demonstration
1.13	The system shall allow users to update non-standard documents.	Demonstration
1.14	The system shall allow users to delete non-standard documents.	Demonstration
1.15	The system shall allow administrators to upload admin-controlled templates.	Demonstration
1.16	The system shall allow administrators to access admin-controlled templates.	Demonstration
1.17	The system shall allow administrators to update admin-controlled templates.	Demonstration
1.18	The system shall allow administrators to delete admin-controlled templates.	Demonstration
1.19	The system shall allow administrators to finalize a document.	Demonstration
1.20	When a review is initiated, the system shall be able to send an e-mail to the chair of the department of the program under review.	Testing
1.21	When the administrator creates new reviews, the system shall provide a list of program which should be up for review during the current year.	Demonstration

Requirements Related to Document Collaboration (2.1, 2.2, ...)		
Requirement No.	Requirement Description	V&V Methodology
2.1	The system shall allow users to upload new revisions of a document.	Demonstration
2.2	The system shall store all revisions of a document.	Demonstration
2.3	The system shall allow users to download the current revision of a document.	Demonstration
2.4	The system shall be able to send e-mail notifications to selected PRS members when a document is uploaded.	Testing
2.5	The system shall allow users to subscribe to e-mails about documents concerning them	Testing

Requirements Related to Miscellaneous Storage (3.1, 3.2, ...)		
Requirement No.	Requirement Description	V&V Methodology
3.1	The system shall be able to store agendas and meeting minutes uploaded by a user.	Demonstration
3.2	The system shall be able to store resources to be made available to all PRS members.	Demonstration
3.3	The system shall be able to send e-mail notifications to selected PRS members when an agenda is uploaded or changed.	Testing
3.4	When sending e-mails, the system shall be able to attach a document to the e-mail.	Testing

Requirements Related to Calendar (4.1, 4.2, ...)		
Requirement No.	Requirement Description	V&V Methodology
4.1	The system shall store the dates and times of meetings of the PRS.	Demonstration
4.2	Users shall be able to create a new meeting for the calendar to track.	Demonstration
4.3	The system shall display all the dates and times of meetings of the PRS to PRS members.	Analysis
4.4	Users shall be able to update the date and time of a meeting.	Demonstration
4.5	Users shall be able to delete a meeting.	Demonstration
4.6	The system shall be able to associate a list of meeting agendas or minutes with each meeting.	Analysis
4.7	Users shall be able to modify the list of associated meeting agendas and minutes for a meeting.	Demonstration
4.8	The system shall send e-mail reminders to PRS members 24 hours before a meeting with all associated meeting agendas as attachments to the e-mail.	Testing
4.9	The system shall send e-mail notification to all PRS members when a meeting is deleted.	Testing

Requirements Related to Comment System (5.1, 5.2, ...)		
Requirement No.	Requirement Description	V&V Methodology
5.1	The system shall allow users to create comments on a document.	Demonstration
5.2	The system shall display the comments of a specified document to users.	Demonstration
5.3	The system shall display the author of each comment.	Demonstration
5.4	The system shall display the date of each comment.	Demonstration
5.5	The system shall allow users to edit their comments.	Demonstration
5.6	The name of the current revision at the time of the creation of the comment shall be displayed with each top-level comment.	Testing

Requirements Related to the User Authentication (6.1, 6.2, ...)		
Requirement No.	Requirement Description	V&V Methodology
6.1	The system shall direct users to a login page if that user is not currently logged in.	Demonstration
6.2	The user shall be blocked from accessing the system if the user is not logged in.	Testing
6.3	The system shall allow users to log in using their CSULA-provided credentials if they are CSULA faculty.	Testing

11. Glossary

Acronyms:

- CSULA - California State University of Los Angeles
- E2E - End-to-End (Testing)
- HTTP - HyperText Transfer Protocol
- HTTPS - HyperText Transfer Protocol over TLS
- MOU - Memorandum of Understanding
- PRISM - Program Review Information System Management
- PRS - Program Review Subcommittee
- SAML - Security Assertion Markup Language
- SMTP - Simple Mail Transfer Protocol as specified in RFC 5321
- TLS - Transport Layer Security

Definitions:

- Document - A single document being produced that contains multiple revisions of its file. (i.e. Self Study, Summary Report, etc.)
- File - Consists of arbitrary data that holds content for a document (i.e. Self Study v1.docx, Self Study v2.docx, etc.)
- Program - A degree program under specific college departments (i.e. Computer Science, Electrical Engineering, etc.)
- Review - A single review process for a department's degree program during a specific year (i.e. CS MS 2018, CS BS 2024, etc.)
- Shibboleth - An implementation of SAML authentication that provides an Identity Provider among other products

12. References

Google JavaScript Style Guide (<https://google.github.io/styleguide/jsguide.html>)