**Software Design**

**Document**

**for**

**Aquila**

**Process Management System**

**Version 1 approved**

**Prepared by Bryan Bravo**

**Barry Boayes**

**Jurelly Tello**

**Sasha Nicole Cisneros**

**Jessica Palacios**

**Aquila**

**10/03/17**

Table of Contents

Revision History

1. Introduction
   1. Purpose
   2. Document Conventions
   3. Intended Audience and Reading Suggestions
   4. System Overview
2. Design Considerations
   1. Assumptions and dependencies
   2. General Constraints
   3. Goals and Guidelines
   4. Development Method
3. Architectural Strategies
4. System Architecture
5. Policies and Tactics
   1. Specific Products Used
   2. Requirements traceability
   3. Testing the software
   4. Engineering trade-offs
   5. Guidelines and conventions
   6. Protocols
   7. Maintaining the software
   8. Interfaces
   9. System's deliverables
   10. Abstraction
6. Detailed System Design
7. Detailed Lower level Component Design
8. User Interface
   1. Overview of User Interface
   2. Screen Frameworks or Images
   3. User Interface Flow Model
9. Database Design
10. Requirements Validation and Verification
11. Glossary
12. Reference

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
| Initial Draft | 12/08/17 |  | 1 |
| Final Draft | 04/08/18 | Revisions | 2 |
|  |  |  |  |
|  |  |  |  |

**1. Introduction**

**1.1 Purpose**

Aquila is a process management tool for UAS. Our web application will allow for UAS to manage and oversee their current projects and past projects. The forms will be filled out online. Submissions will also be done online.This will be our first revision of the project we have done.

Our software will improve efficiency for UAS and the applicant. This will be our first revision of the project we have done. The web application allows for better workflow. In our application will have a form of communication built in so it is easier for the applicant and UAS analyst to communicate on forms or any questions they have regarding the proposal. Our software will replace their current process of printing and using paper documents. Removing the paper work allows for our system to run online easier and quicker.

**1.2 Document Conventions**

The typographical conventions followed were given by our template. We used the same fonts and highlighting as shown on the template given to us by our instructors.

**1.3 Intended Audience and Reading Suggestions**

This document is to be read by the development team, the project managers, testers, and documentations writers. California State University, Los Angeles ECST Department, and UAS, may review the document to learn about the project and to understand the detailed list of requirements. The SRS has been organized approximately in order of increasing specificity. Developers and project managers need to become intimately familiar with the SRS. Other than requirements the SRS will give an inside scoop on what the project is and what it will do when completed.

The SRS first describes a little about the project then an overall description that goes into depth about the project. Then it describes the external interface requirements and the requirement specifications. Finally the SRS talks about the nonfunctional requirements. The best sequence for reading the SRS is to read the introduction and overall description to understand what our project is all about. Then the external interface requirements which details important information about the interfaces. After that the nonfunctional requirements which mainly focuses on the security aspect of our project. Finally the requirement specifications because this sections is more technical and goes in depth of what functional requirements we will meet and the database requirements.

**1.4 System Overview**

We will be making a web application where applicants will be able to go through the current Pre Award process online. They will begin by making an account, then they will be able to fill out all the forms online. Once finished they submit to a UAS analyst, a face to face meeting is scheduled. In this meeting, they will agree to dates for final submission of all the finalized forms. After the UAS analyst will be able to submit the entire application to the funding agency. On the administrative side, our application will allow administrators to view all of their projects. They will be able to access past projects through an archive. Our application will have a comment section allowing the applicant and administrator to have an easy means of communication and avoids doing email transactions for all feedback. This web application’s main goal is to help UAS make their current process more efficient.

**2. Design Considerations**

**2.1 Assumptions and Dependencies**

Later on in the project, we will have dependencies for SAML. The role of the user is a main factor in what operations can be made.

**2.2 General Constraints**

The user must have a valid Cal State LA email login. Our software does not include any hardware environments as long as the user has access to a computer and internet they will be able to access our software.

**2.3 Goals and Guidelines**

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. For each such goal or guideline, unless it is implicitly obvious, describe the reason for its desirability. Feel free to state and describe each goal in its own subsubsection if you wish. Such goals might be:

· The KISS principle ("Keep it simple stupid!")

· The Software has a mandatory delivery date that must be met (end of the cd3337 class)

· Emphasis on speed versus memory use

· The product should work, look, or "feel" like an existing product

Our primary goals is to have the Pre Award complete by the end of the year. The Pre Award is the process a PI facing on trying to apply for a grant.

After we will try to complete the Post Award process. This process happens after the grant/contract has been granted.

The software is due at the end of Spring 2018

**2.4 Development Methods**

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used.

These would be things such as the ‘Water Fall Development’ methods, ‘Agile Development’, ‘Unplanned Mad Scramble Development’, or other development models and variations. Describe how these were applied in the case of your project.

**3. Architectural Strategies**

Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

Aquila is built having the backend and frontend completely separate. This allows them to be developed independent of each other. This was decided to be able to manipulate data to certain endpoints to do CRUD requests. This way will also allow one section of the system for example the frontend, will not have to wait for the other section,the backend, to complete its task to continue working. Our previous plan was to create our system using MVC(model, view, controller) this involves using jsp and servlets. This plan was not developed because MVC requires the the frontend and backend to work concurrently and is very dependent on each other. This process would have made the system more tedious to create.

· Use of a particular type of product (programming language, database, library, etc. ...)

· 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.---digital signature

· Future plans for extending or enhancing the software

The future plans for Aquila include adding more of UAS processes to our system./post

· User interface paradigms (or system input and output models)

Aquila will be available through the user’s web browser and will take in user input and output the data requested.

· Hardware and/or software interface paradigms

Aquila does not have any hardware paradigms. For the software we will use TypeScript to develop the user interfaces. The backend will be developed using Java as its programming language.

· Error detection and recovery

· Memory management policies

No duplicate entries to database

· External databases and/or data storage management and persistence

Mysql jpa hibernate

· Distributed data or control over a network

Hash for files security encrypting passwords

· Generalized approaches to control

· Concurrency and synchronization

· Communication mechanisms

Email notification

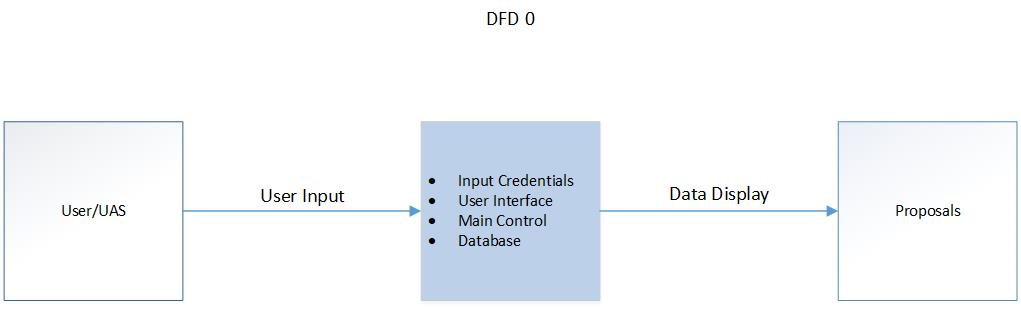
· Management of other resources

//third party documents

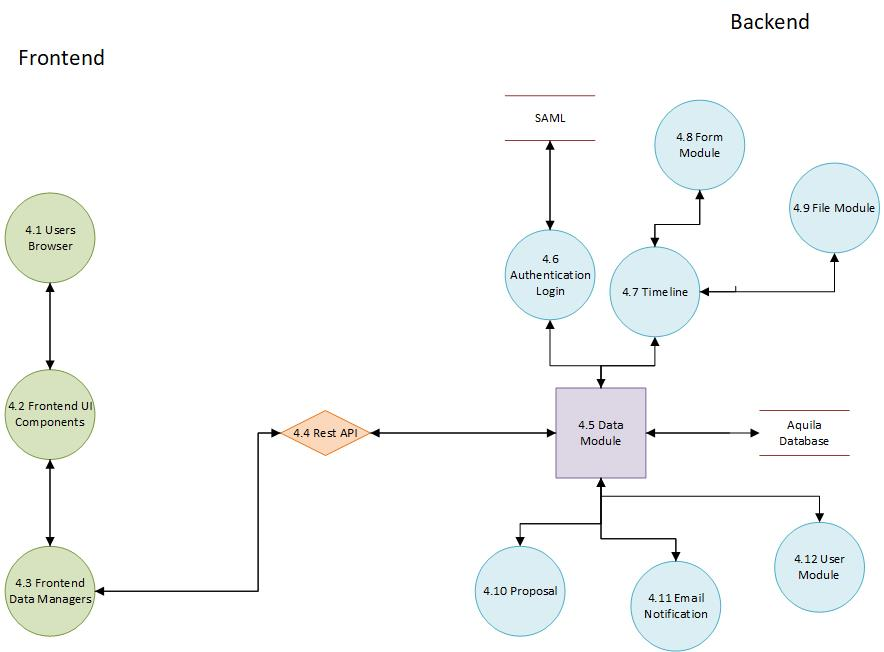
Each significant strategy employed should probably be discussed in its own subsection. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose).

**4. System Architecture**

The main responsibility for our system is to provide the user and the admin access to a workflow management system. For the Pre Award process the forms are broken down by request. Every form has its own UI components, request, controller, daos, and model.



All the input needed for reading and writing proposals/users is sent through HTTP requests. The backend of our system is what saves all the information(database entries and files) and performs whatever calculations are necessary. Upon the system persisting the necessary data, it is to be returned to the client and displayed.

DFD 1 

Our web application, Aquila, is divided into up into two sections, the frontend and the backend. The frontend and backend work concurrently with each other and communicate through a restful api.

4.1 Users Browser

The user’s browser is where a majority of data is either requested or modified and saved.

4.2 Front End UI components

Large portions of the application are broken up into their own respective components. These components all have their encapsulated logic, display and styling.

4.3 Front End Data Managers

These components have data injected into them via Angular services. This paradigm of dependency injection is also present in our server implementation via Inversion of Control.

4.4 RestFul Api

Restful Api is the design paradigm we are utilizing to have our User Interface communicate with the server via Http and Json representation of data.

4.5 Data Module

4.6

Authentication/Login Module - The login module is used to login and authenticate any of the users of our application. We will be using SAML to authenticate any of the CSULA account in addition to our internal database.

4.7

Timeline Module - Arguably the largest part of the application. The timeline modules will keep track of the users process on a proposal. Handles the uploading of files and creation of forms.

4.8

Form Module - The form module will hold all forms needed for the UAS process. These forms will be used in every application submitted to UAS.

4.9

File Module - The file module will contain any of the forms that will be submitted as a file. These forms include the budget file.

4.10

Proposal Module - The proposal module is where each proposal for a project is created. The forms and files are all included on the proposal module. The propose will take the user to the any of the forms they will need to fill out.

4.11

Email Notifications Module - The email notification module will be responsible of notifying all types of users of notifications in their best interest.

4.12

User Module - Most users will be created via authentication with SAML. There are exceptions such as UAS analysts being created.

**5. Policies and Tactics**

Our main tactic was separating our frontend and backend. The frontend and backend are going to developed completely independent from each other. The benefits of separating these two will be that if one crashes the other may continue developing their part of the system. Another benefit is there is no waiting for the other part system to be built. The frontend can continue to build their user interfaces until the backend is ready to connect to the frontend. The separation between sections also follows modern trends because it is easier to swap components and allows for independent development.

**5.1 Choice of which specific products used**

* Eclipse IDE
* Java
* MySQL
* Hibernate
* Java Persistence API
* Spring framework
* Angular
* Prime MG
* VS code
* Chrome Development Tools

**5.2 Plans for ensuring requirements traceability**

We plan on documenting the function and flow our software as best as possible within this document.

**5.3 Plans for testing the software**

The backend uses Restful API and this is tested with Postman to do CRUD requests. As for the frontend, there are many ways of testing it. There’s cross browser compatibility we will be testing for. We will also be performing end-to-end testing which will run random requests to see if anything crashes via Karma.js

5.3.1 Coding guidelines and conventions

Traditional Java conventions and TypeScript conventions defined by TS LINT.

5.3.2 The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

Modularization of controllers, Database Accessor Objects, Reusable UI components.

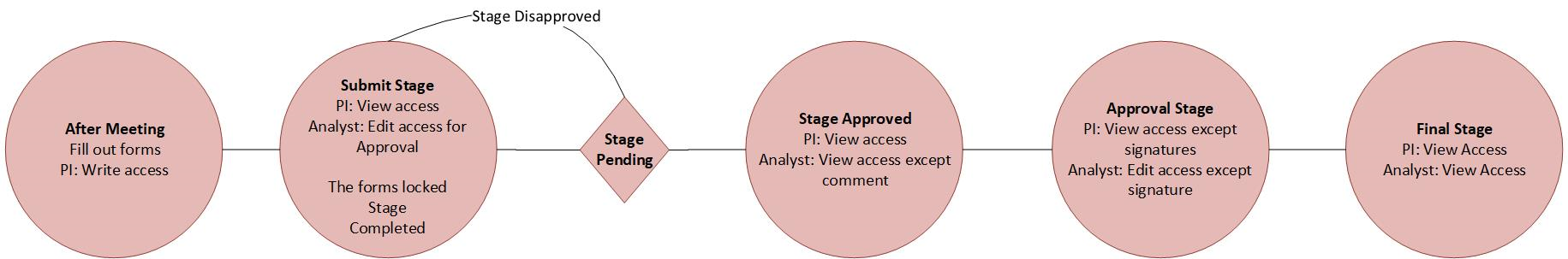
5.3.3 Interfaces for end-users, software, hardware, and communications

Due to our components having data injected to them in the UI. Maintenance would be easier for any potential future developers.

**6. Detailed System Design**

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD’s will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.



**6.x Name of Component (Module)**

**6.x.1 Responsibilities**

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

**6.x.2 Constraints**

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

**6.x.3 Composition**

A description of the use and meaning of the subcomponents that are a part of this component.

**6.x.4 Uses/Interactions**

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclasses, and metaclasses.

**6.x.5 Resources**

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

**6.x.6 Interface/Exports**

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that

**7. Detailed Lower level Component Design**

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”.) Note that there isn't necessarily a one-to-one correspondence between packages and components.

**7.x Name of Class or File**

**7.x.1 Classification**

The kind of component, such as a subsystem, class, package, function, file, etc.

**7.x.2 Processing Narrative (PSPEC)**

A process specification (PSPEC) can be used to specify the processing details

**7.x.3 Interface Description**

**7.x.4 Processing Detail**

**7.x.4.1 Design Class Hierarchy**

Class inheritance: parent or child classes.

**7.x.4.2 Restrictions/Limitations**

**7.x.4.3 Performance Issues**

**7.x.4.4 Design Constraints**

**7.x.4.5 Processing Detail For Each Operation**

**8. Database Design**

Include details about any databases used by the software. Include tables and descriptions.

// talk about hibernate stuff

//put dfd

**9. User Interface**

The user interface is hosted in the user’s web browser with internet access. The browser is used to navigate through system and communicate with database.

Common features that will be found in our website will be installed to help users interact with our application. We will have a process bar that will show the status of the applicant’s proposal. We will have a comment section that allows easy communication between the PI and UAS analyst. There will be navigation buttons for the user to go through the pages of the forms. For the archive, the user will have a search bar that allow user to search by name, date, etc.

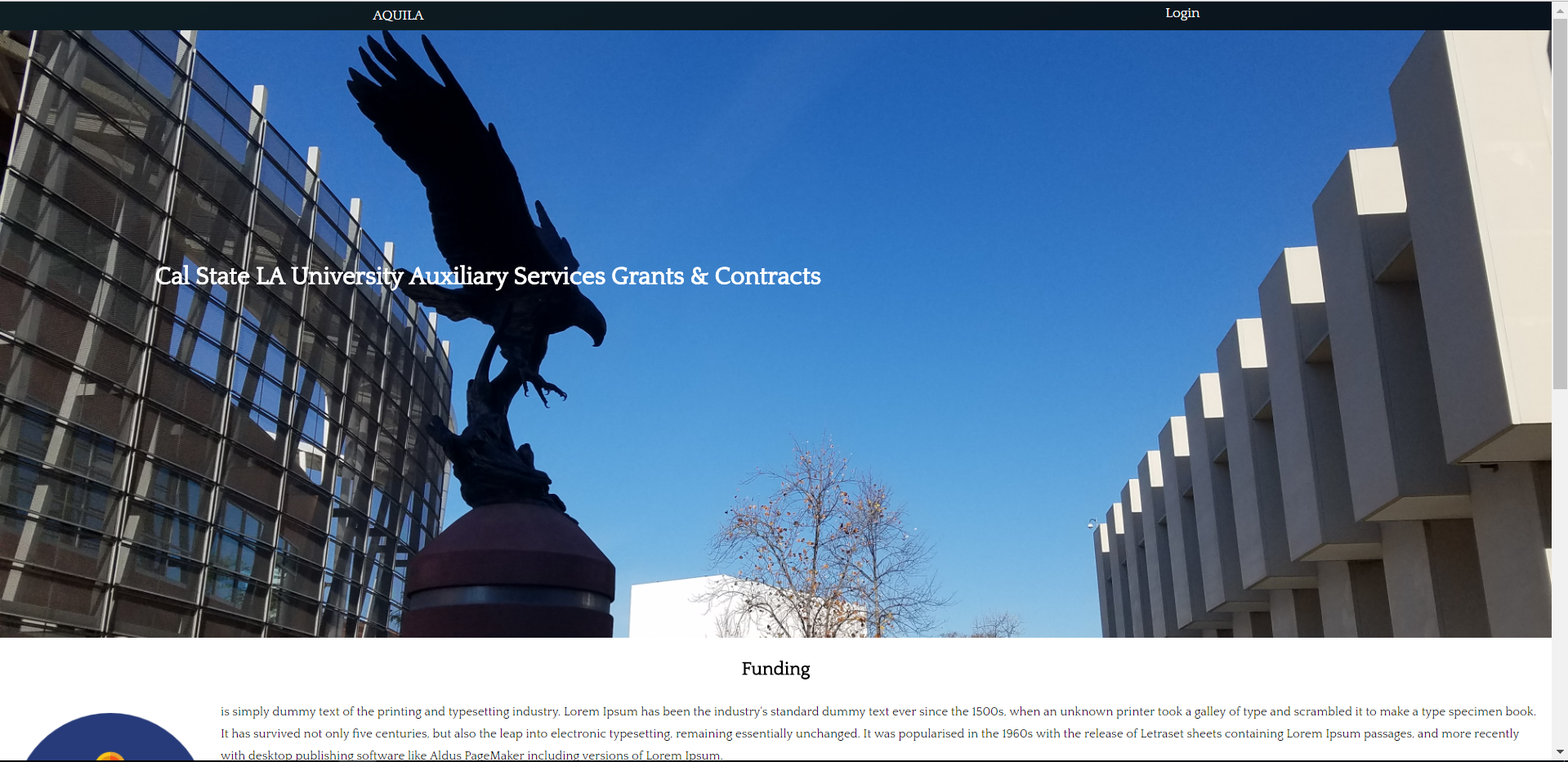
Everything in the UI will be displayed in an organized fashion, that is easy to use for the user to understand and use.

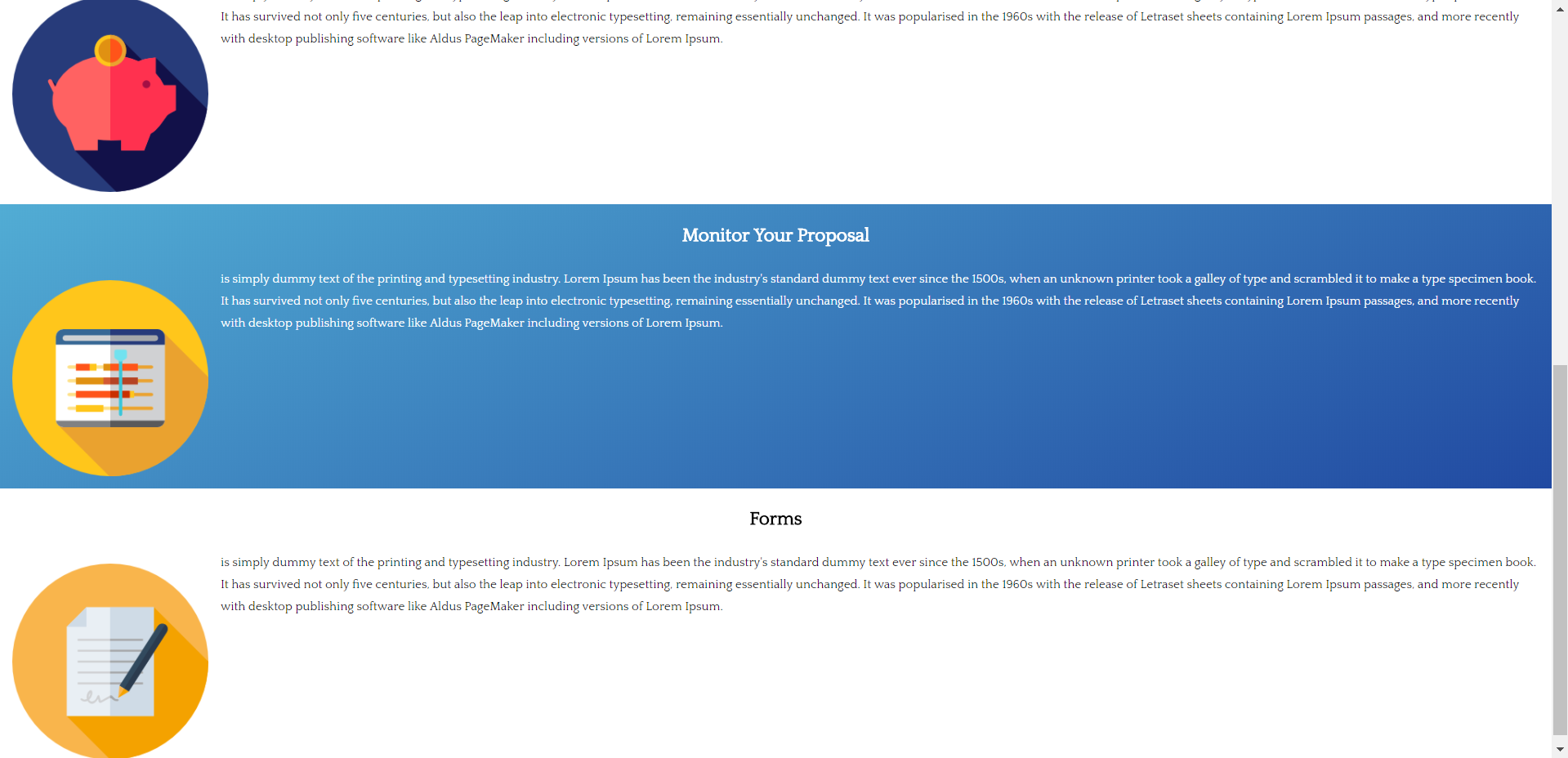
**9.1 Overview of User Interface**

User Interface

* Display a login page
* UAS users can view a list of proposals along with date and status
* UAS user will be able to schedule a meeting with PI user
* Display a timeline for PI user to be able to view their deadlines
* Provide notification emphasizing nearby deadlines
* Display notification of completed forms

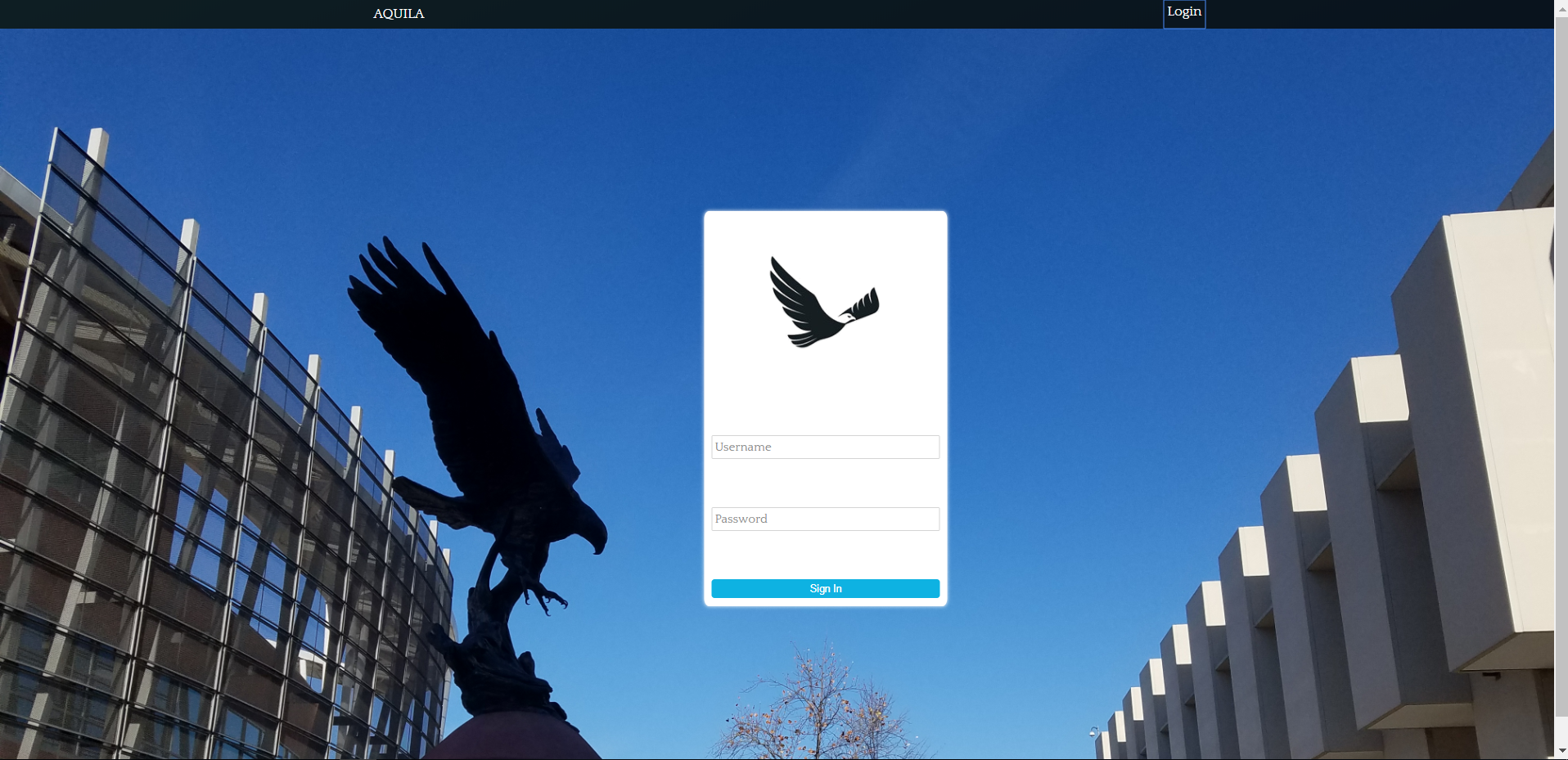
**9.2 Screen Frameworks or Images**





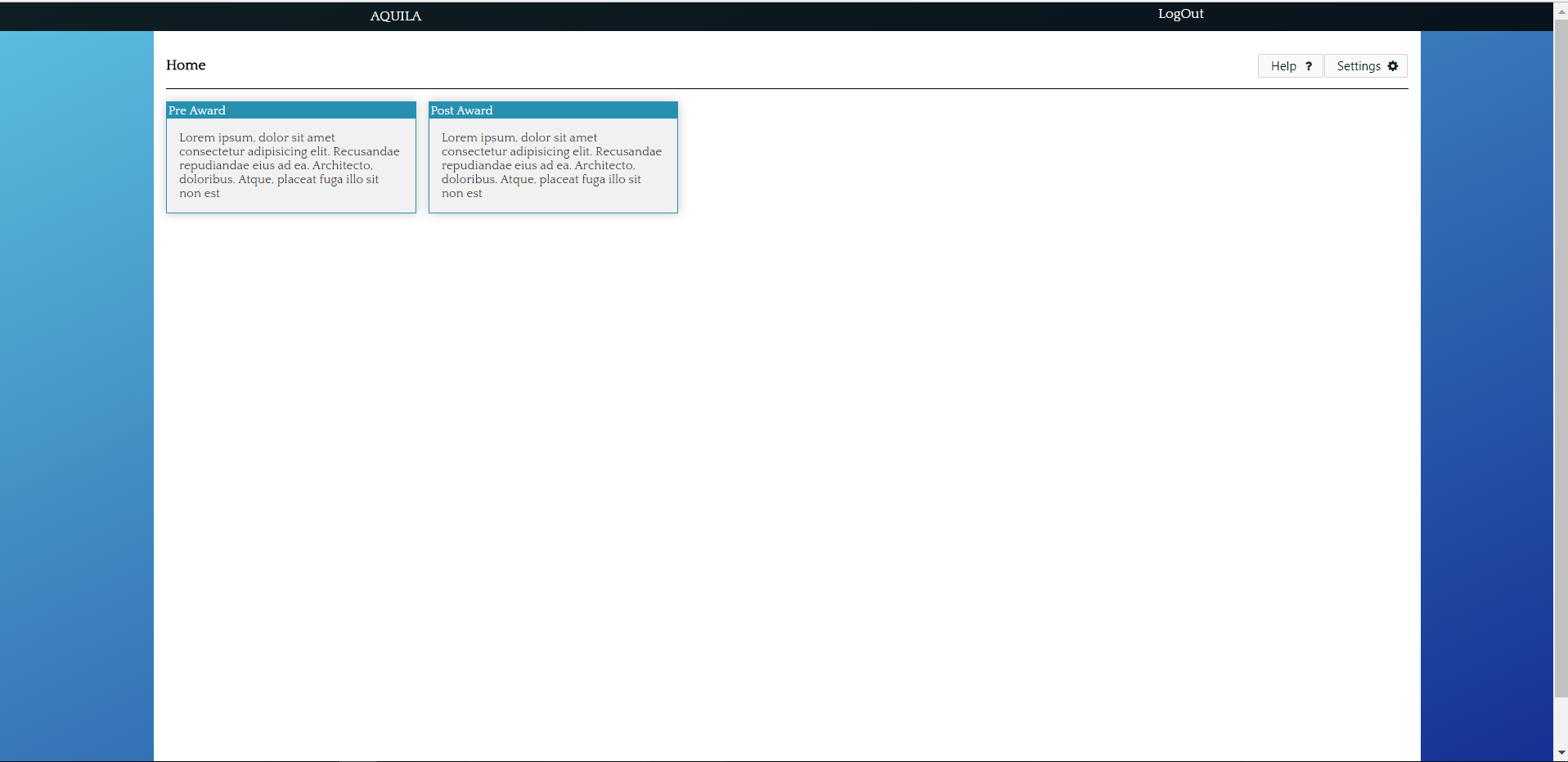
**Landing Page**

This will be the landing page for UAS that will give a brief description on its primary purpose. It will also give a description of the main operations of UAS at Cal State LA. The landing page will show some of the accomplishments of UAS along with statistics. Any general information of UAS will be displayed on the landing page.



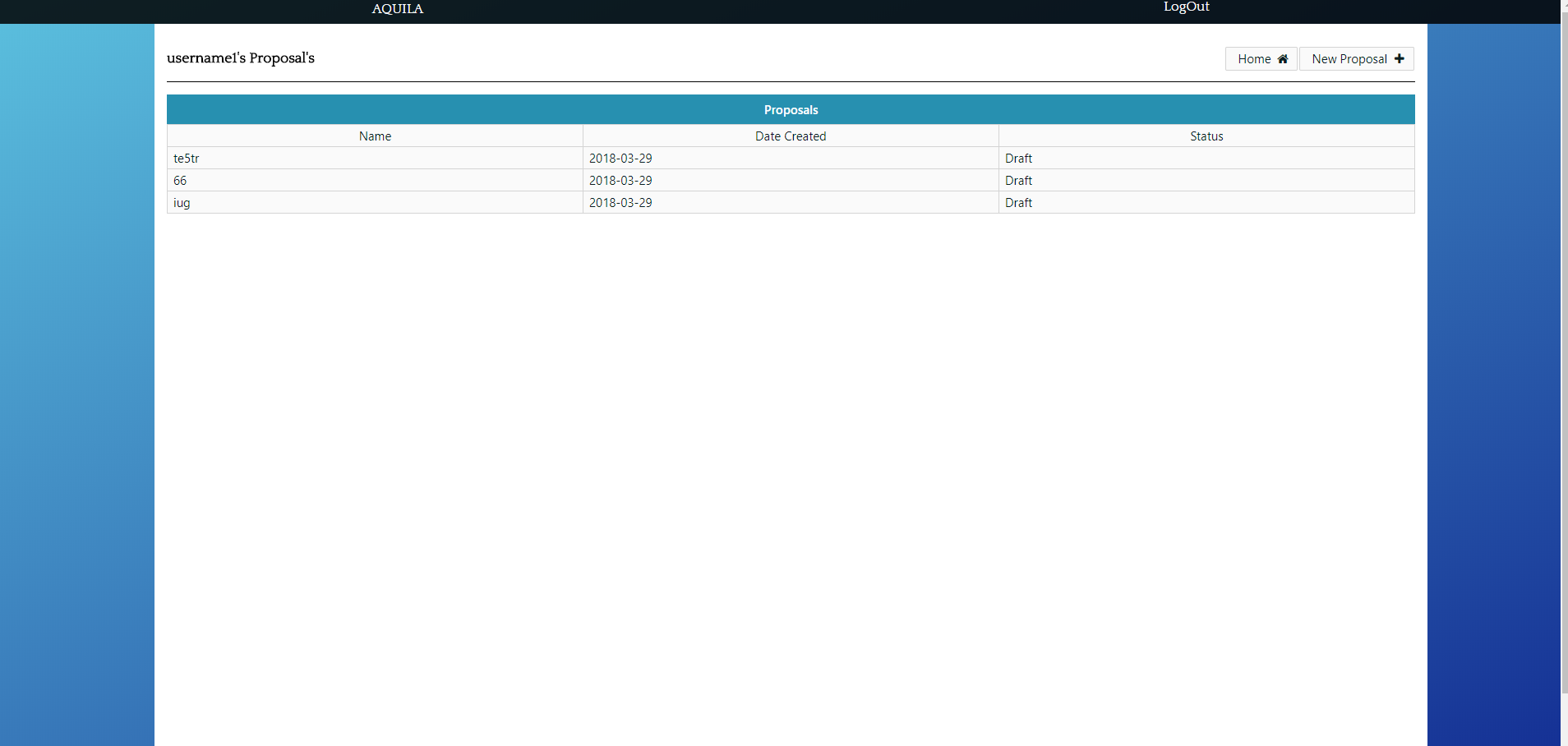
**Login Page**

The login page will allow existing users to input their credentials such as their username and password. Once the user enters their credentials and if authorized the user will be able to view the user homepage. If the user does not want continue to login they are can back to the homepage.



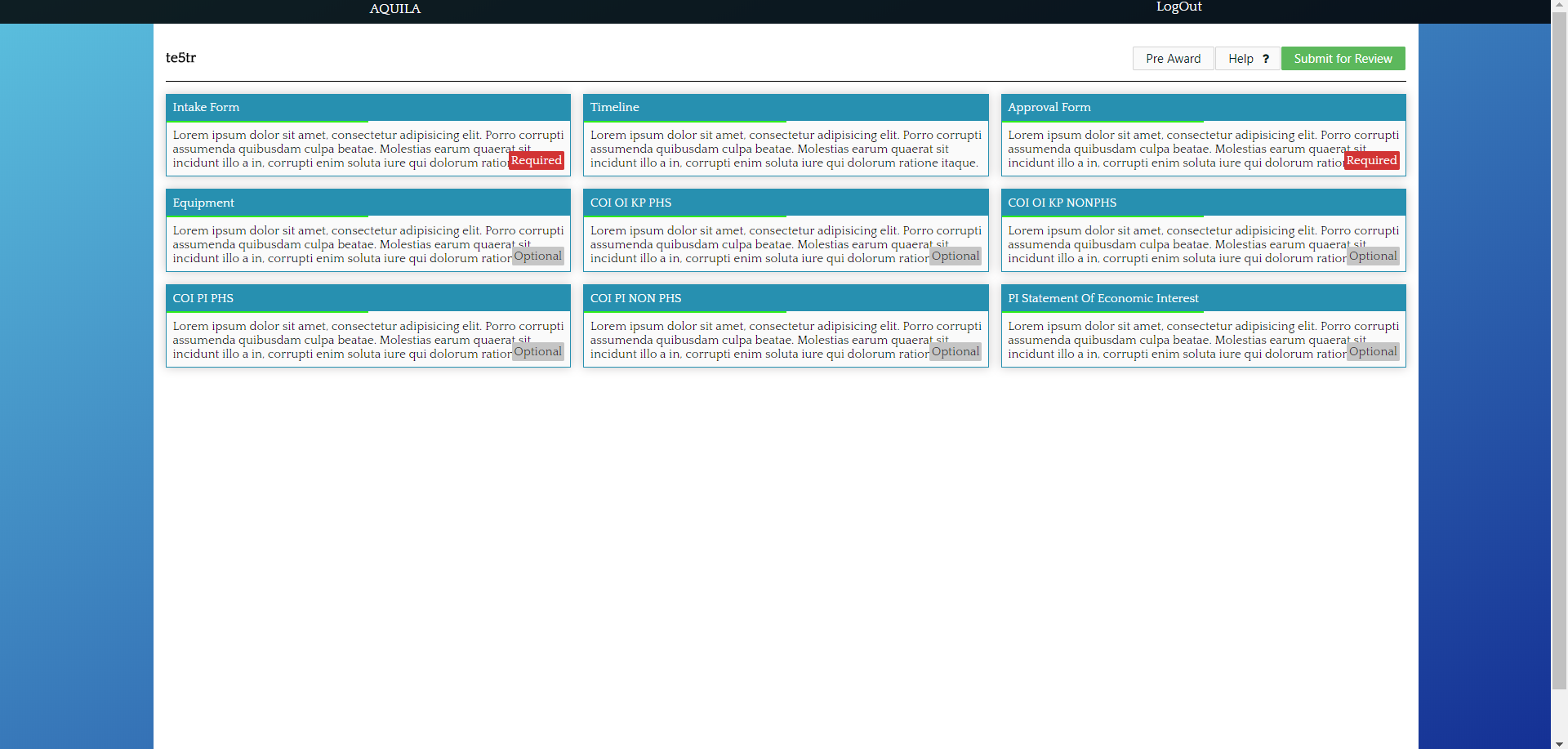
**User Home Page**

After logging in, the user will be allowed to pick which process they want to enter. The user will be allowed to view their information or edit their information.



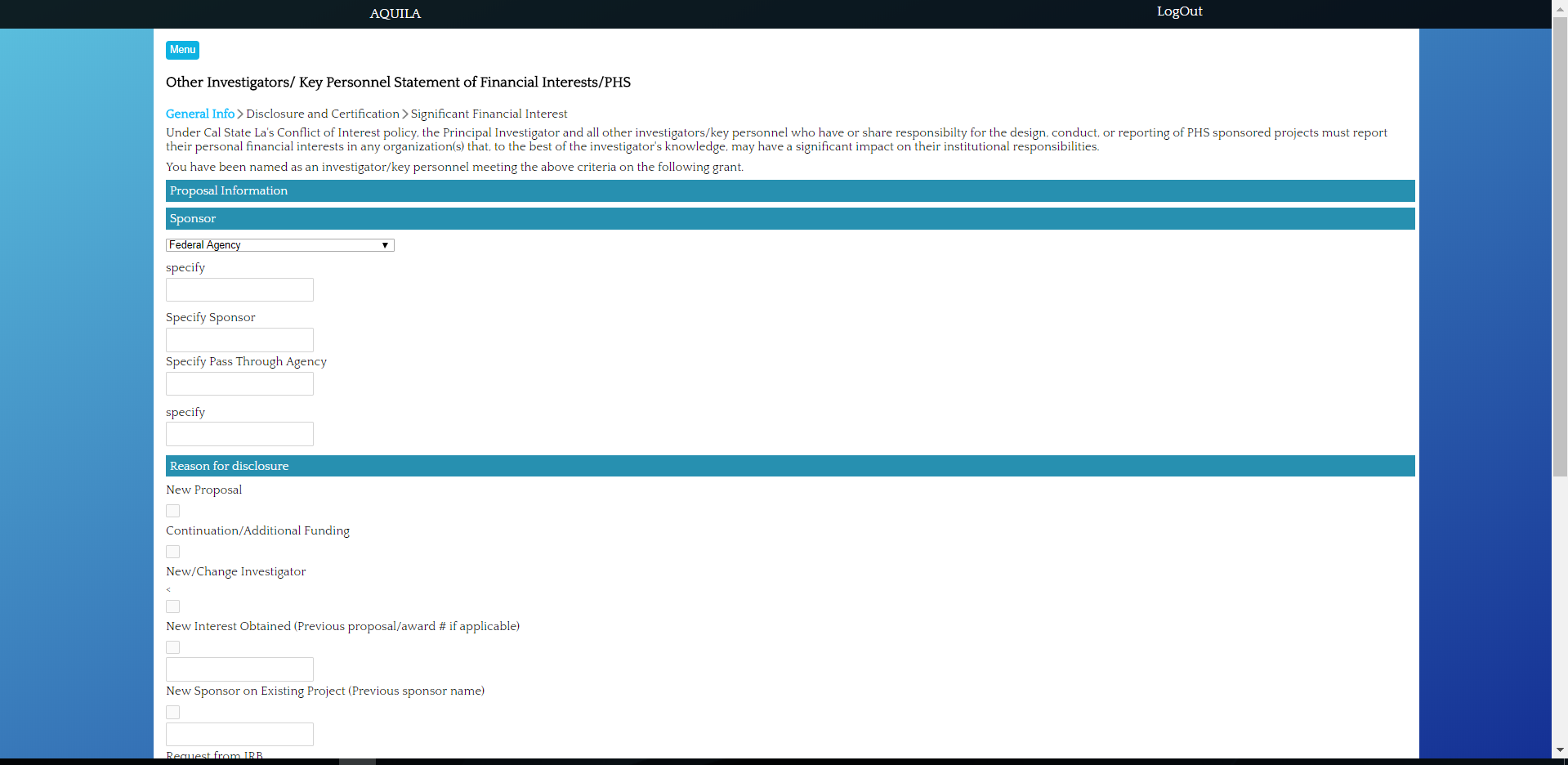
**Pre Award Home Page**

This page will be appear after the user click on the Pre Award process. Here they will be able to make new projects and they will be able to view their previous or current projects. If they click on a project that they worked on previously and it has been completed they will be able to view everything that composed that project. If a project was denied they will also be able to view when it got denied. If they are currently working on a project it will prompt them to view or edit their project. Along with their current project they will be able to view a edit proposal page where they will view multiple forms and a timeline/deadlines that the user needs to complete.



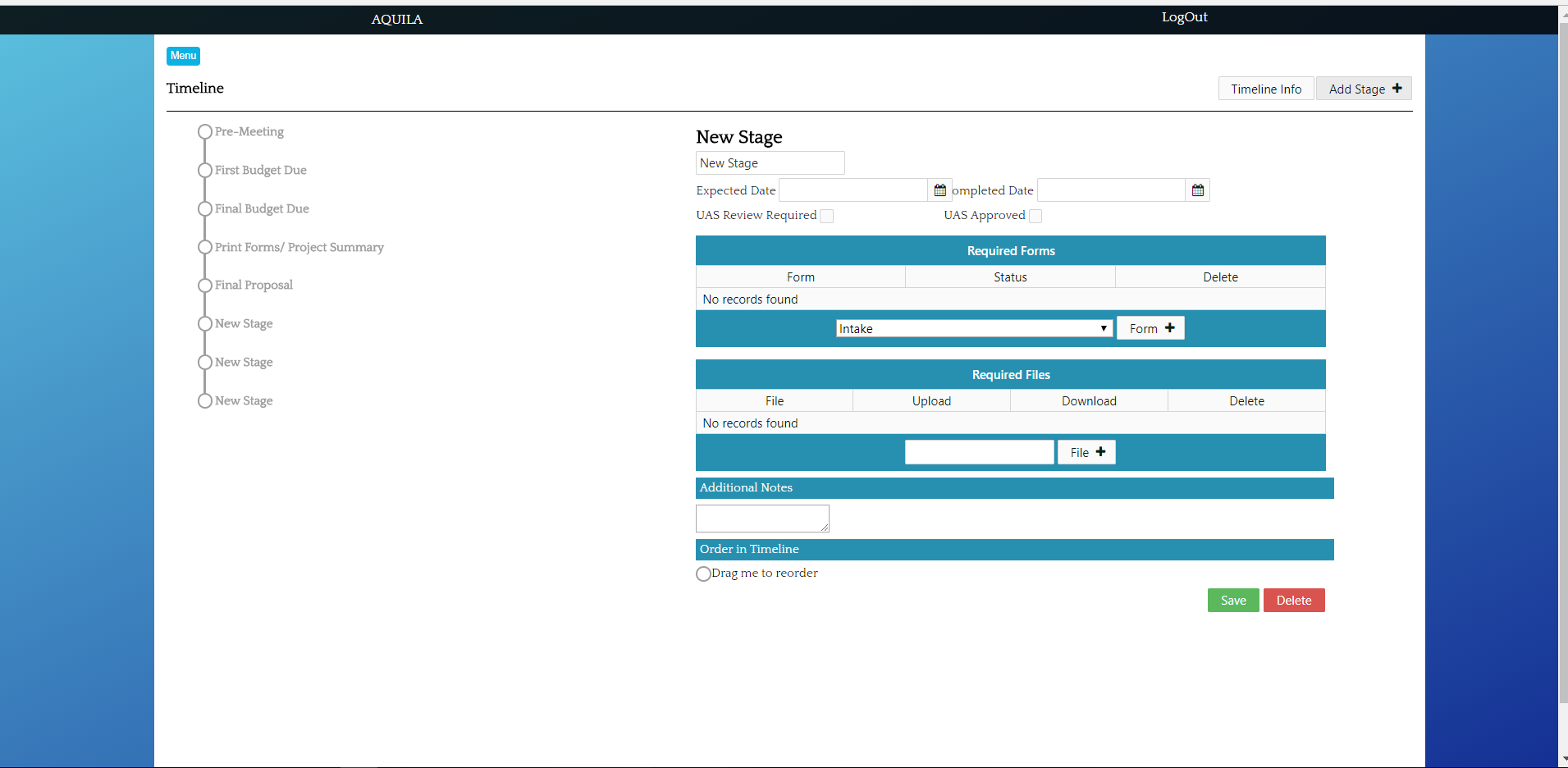
**Edit Proposal**

Once they click on their project, they will see the forms view a edit proposal.A status bar will be displayed to view the progress on a specific form. Only required forms will be displayed. A overall timeline of all the project will be on edit proposal but it will be locked until the user has scheduled a meeting with a UAS analyst. Their will also be an approval form that will be locked the approval stage in the timeline has been reached.



**Form Page (Example)**

This is an example of a form. The form will have a status bar on the top showing the progress of that form. They will enter the information and save and then proceed to the next page.

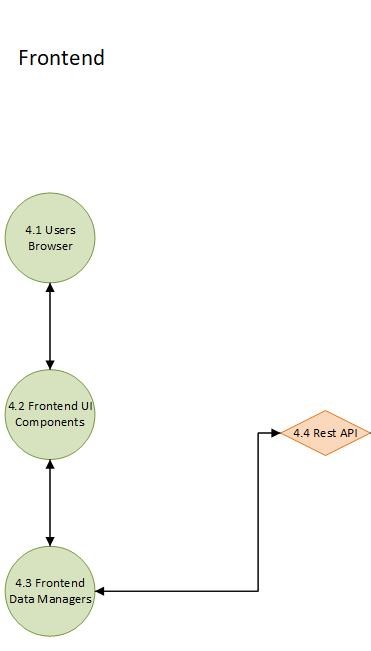


**Timeline**

This will be the timeline page that is locked until the user has met up with UAS analyst. The timeline page has control of the forms and files. Through the timeline the analyst will be able to give access to the forms required from the PI. The analyst will be able to give the deadline of the forms through the timeline. UAS Analysts will be able to reorder the stages of the timeline but once the approval form has been completed all the forms will be locked and no user will able to edit any forms. The analyst has privileges that the PI does not such as being able to modify the due dates, delete and add forms.

**9.3 User Interface Flow Model**

// left hand side of dfd, get user input, communicate with server, change display based on success or failure



**10. Requirements Validation and Verification**

For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.

For each entry describe the method for testing that the requirement has been met.

|  |  |  |
| --- | --- | --- |
| 4.1 | **User Interface Module (UI)** Requirements |  |
| Requirement No. | Requirement Description | Components |
| 4.1.1 | UI shall provide a login page | Frontend: 4.2 Frontend UI Components  Backend: 4.6 Authentication Login |
| 4.1.2 | UI shall display a current status bar | Frontend: 4.2 Frontend UI Components |
| 4.1.3 | UI shall allow the PI to upload documents | Frontend: 4.2 Frontend UI Components  Backend: 4.5 Data Module, 4.9 File Module |
| 4.1.4 | UI shall allow the PI to access their archive | Frontend: 4.2 Frontend UI Components  Backend: 4.5 Data Module, 4.7 Timeline |
| 4.1.5 | UI shall allow UAS oversee all projects | Frontend: 4.2 Frontend UI Components  Backend: 4.5 Data Module, 4.7 Timeline |
| 4.1.6 | UI shall allow the applicant to logout | Frontend: 4.2 Frontend UI Components |

|  |  |  |
| --- | --- | --- |
| 4.2 | **Pre Award (PR)** Requirements |  |
| Requirement No. | Requirement Description | Components |
| 4.2.1 | PR shall allow PI to create a new project | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal |
| 4.2.2 | PR shall allow the PI to fill out the Intake Form | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal, 4.7 Timeline, 4.8 Form Module |
| 4.2.3 | PR shall allow the PI to fill out the Budget Form | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal, 4.7 Timeline, 4.9 File Module |
| 4.2.4 | PR shall allow the PI to fill out the Approval Form | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal, 4.7 Timeline, 4.8 Form Module |
| 4.2.5 | PR shall allow the PI to fill out optional Equipment Form | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal, 4.7 Timeline, 4.8 Form Module |
| 4.2.6 | PR shall allow the PI to fill out optional Conflict of Interest Forms | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal, 4.7 Timeline, 4.8 Form Module |
| 4.2.7 | PR shall have access to a Timeline | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal, 4.7 Timeline |
| 4.2.8 | PR shall allow the PI to communicate with a UAS staff | Frontend: 4.2 Frontend UI Components  Backend: 4.11 Email Notification, 4.12 User Module |
| 4.2.9 | PR shall allow the PI to terminate the project at any moment | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.10 Proposal, 4.7 Timeline |
| 4.2.10 | PR shall allow the PI to save a project at any moment | Frontend: 4.2 Frontend UI Components  Backend: 4.5 Data Module |
| 4.2.11 | PR shall allow user to edit information in the forms | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal, 4.7 Timeline, 4.8 Form Module |
| 4.2.12 | PR shall allow UAS to oversee any current projects | Frontend: 4.2 Frontend UI Components  Backend: 4.10 Proposal |
| 4.2.13 | PR shall allow UAS to approve a proposal | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.10 Proposal, 4.7 Timeline |
| 4.2.14 | PR shall allow UAS to submit a proposal | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.10 Proposal, 4.7 Timeline |

Post Award Requirements are in progress.

|  |  |
| --- | --- |
| 4.3 | **Post Award (PO)** Requirements |
| Requirement No. | Requirement Description |

|  |  |  |
| --- | --- | --- |
| 4.4 | **User Management (UM)** Requirements |  |
| Requirement No. | Requirement Description | Components |
| 4.4.1 | UM shall put user information into database | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |
| 4.4.2 | UM shall allow only one account per applicant | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |
| 4.4.3 | UM shall require a distinct username | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |
| 4.4.4 | UM shall require a phone number | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |
| 4.4.5 | UM shall require an email address | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |

|  |  |  |
| --- | --- | --- |
| 4.5 | **Proposal Archive (PA)** Requirements |  |
| Requirement No. | Requirement Description | Components |
| 4.5.1 | PA shall save all completed projects from a PI | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.5 Data Module, 4.10 Proposal |
| 4.5.2 | PA shall allow PI to search through their previous projects | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.5 Data Module, 4.10 Proposal |
| 4.5.3 | PA shall allow UAS to view all projects | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 44.5 Data Module, 4.10 Proposal |
| 4.5.4 | PA shall save different versions of the same project | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.5 Data Module, :4.10 Proposal |

|  |  |  |
| --- | --- | --- |
| 4.6 | **Security (SEC)** Requirements |  |
| Requirement No. | Requirement Description | Components |
| 4.6.1 | SEC shall verify the user | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.5 Authentication Login, 4.12 User Module |
| 4.6.2 | SEC shall verify a UAS faculty | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.5 Authentication Login, 4.12 User Module |
| 4.6.3 | SEC shall only allow the PI to view their projects | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |
| 4.6.4 | SEC shall allow UAS to view project | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |
| 4.6.5 | SEC shall not allow UAS to revise completed projects | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |
| 4.6.6 | SEC shall allow UAS to download any project | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.12 User Module |
| 4.6.7 | SEC shall allow the applicant to logout of their account | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.6 AUthentication Login, 4.12 User Module |
| 4.6.8 | SEC shall allow the UAS faculty to logout of their account | Frontend: 4.2 Frontend UI Components, 4.3 Frontend Data Managers  Backend: 4.6 AUthentication Login, 4.12 User Module |

**11. Glossary**

Does not apply at this moment.

**12. References**

Does not apply at this moment.