Computer Science Senior Capstone

CS 486C – Senior Capstone Design

ECOders

Project: *Colorado Plateau Cooperative Ecosystem Studies Unit (CPCESU) Project Management System*

Software Design

Overview: The purpose of this document is to list out the requirements and other criteria that will determine if we have successfully delivered a successful product that solves out client's problem

Team Members:

Colton Nunley

Joseph Remy, Jr.

Jasque Saydyk

Ana Paula Chaves Steinmacher - *Mentor*

Northern Arizona University

School of Informatics, Computing, and Cyber Systems

Clients:

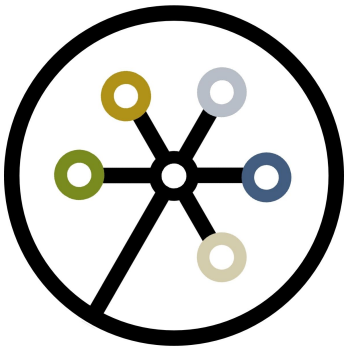
Dr. Todd Chaudhry

Laurie Thom

Cooperative Ecosystem Studies Unit

Colorado Plateau

February 8, 2019 - Rough Draft



This page is left blank intentionally.

Colton Nunley
crn79@nau.edu

Joseph Remy - *Team Lead*
remy@nau.edu

Jasque Saydyk
jrs496@nau.edu

Table of Contents

Table of Contents 3

1. Introduction 5

2. Implementation Overview 7

4. Architectural Overview 8

5. Model and Interface Descriptions 13

6. Implementation Strategy 21

8. Conclusion 25

Colton Nunley
crn79@nau.edu

Joseph Remy - *Team Lead*
remy@nau.edu

Jasque Saydyk
jrs496@nau.edu

This page is left blank intentionally.

Colton Nunley
crn79@nau.edu

Joseph Remy - *Team Lead*
remy@nau.edu

Jasque Saydyk
jrs496@nau.edu

1. Introduction

Preserving nature and our past requires a wide variety of work and information to identify the problems, develop solutions, and gain insight into how our world used to be and how it is changing. This is done with a wide variety of methodologies. From investigating archeology sites to conducting zoological surveys, all of these projects help preserve the nature and history of the American Southwest, allowing us to learn more about the impacts we have on our fragile ecosystem.

The Colorado Plateau Cooperative Ecosystem Studies Unit (CPCESU) is a consortium which organizes federal agencies and Native, state, and local governments with non-governmental organizations and universities to complete a variety of these conservation projects. To do this, the CPCESU gets dozens of project proposals, hundreds of modifications, and millions of dollars each year to setup, organize, track, and archive these projects. Since their founding, they have used an ad-hoc approach to project management using email, Microsoft Excel, a Microsoft Access database, and a shared file system drive.

Due to the make-do nature of the forms and data storage, the CPCESU staff are spending up to half of their time at work to track these projects. If the number of projects were to double, the CPCESU staff would find themselves overwhelmed and have no time to spend on their other work for the organization. In addition to this, there is no defined structure for data entry into the database, leading to partially completed rows and a lack of consistent, vital information the organization needs to renew its charter. There is no way for project leaders and organizations to modify their agreements to extend the timeline and allocate more funds without emailing or calling the CPCESU, and the CPCESU staff need to be able to search their dataset and export statistics they need to report to inquiring organizations, NAU, and the CPCESU Director. Lastly, certain parts of CPCESU projects depend on budget spreadsheets, approval documents, final reports and more, which are not directly linked to projects and need to be found manually in a massive file system.

ECOders is a senior Computer Science capstone group formed under the School of Informatics, Computing, and Cyber Systems at Northern Arizona University. Team members include Joseph Remy, Jr. (Team Lead), Colton Nunley, and Jasque Saydyk and the Team Mentor is Ana Paula Chaves Steinmacher. We were tasked by Dr. Todd

Chaudhry and Laurie Thom with developing a website and database solution for the CPCESU with three essential components: a project management system for the CPCESU; a website for showcasing public projects and information; and a system allowing organizations to develop, submit, and track projects. Our solution, at a glance, shall reduce the time spent tracking and managing the project for CPCESU, allow for organizations to directly submit information into the management system through the use of standardize forms and data import, and must include functionality for quickly and easily exporting data for generating reports.

This software design document begins by outlining our project's overall architectural design and the specific plans for every level and module of our solution. The purpose of this document is to make implementation plans explicit and allow us to catch any problems with our design early on. At the end of the semester, we will revise this document and include it in our final report.

Colton Nunley
crn79@nau.edu

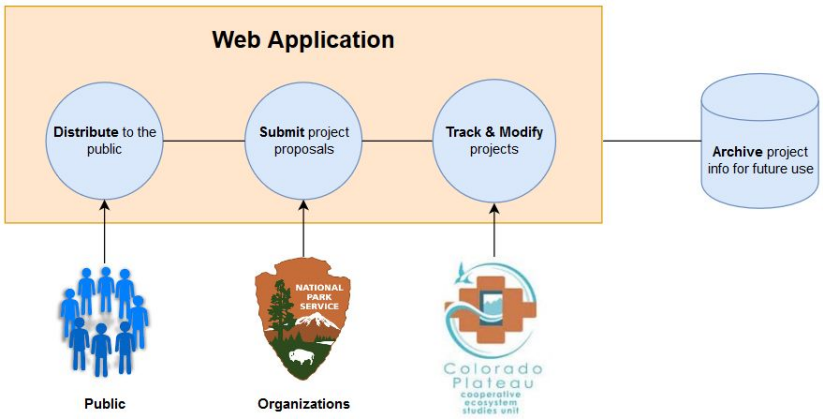
Joseph Remy - *Team Lead*
remy@nau.edu

Jasque Saydyk
jrs496@nau.edu

2. Implementation Overview

Our solution to the CPCESUs problem, nicknamed “Summit”, will be a web application with three primary sections: a public facing front end that distributes projects and informs the public about the CPCESU, a section for organizations to submit project proposals and modifications to projects, and a backend for the CPCESU to manage, track, and modify their projects. All of this is archived into a robust, easy to access, and maintainable database, as shown in Figure 2.1 below.

Figure 2.1: Solution Diagram



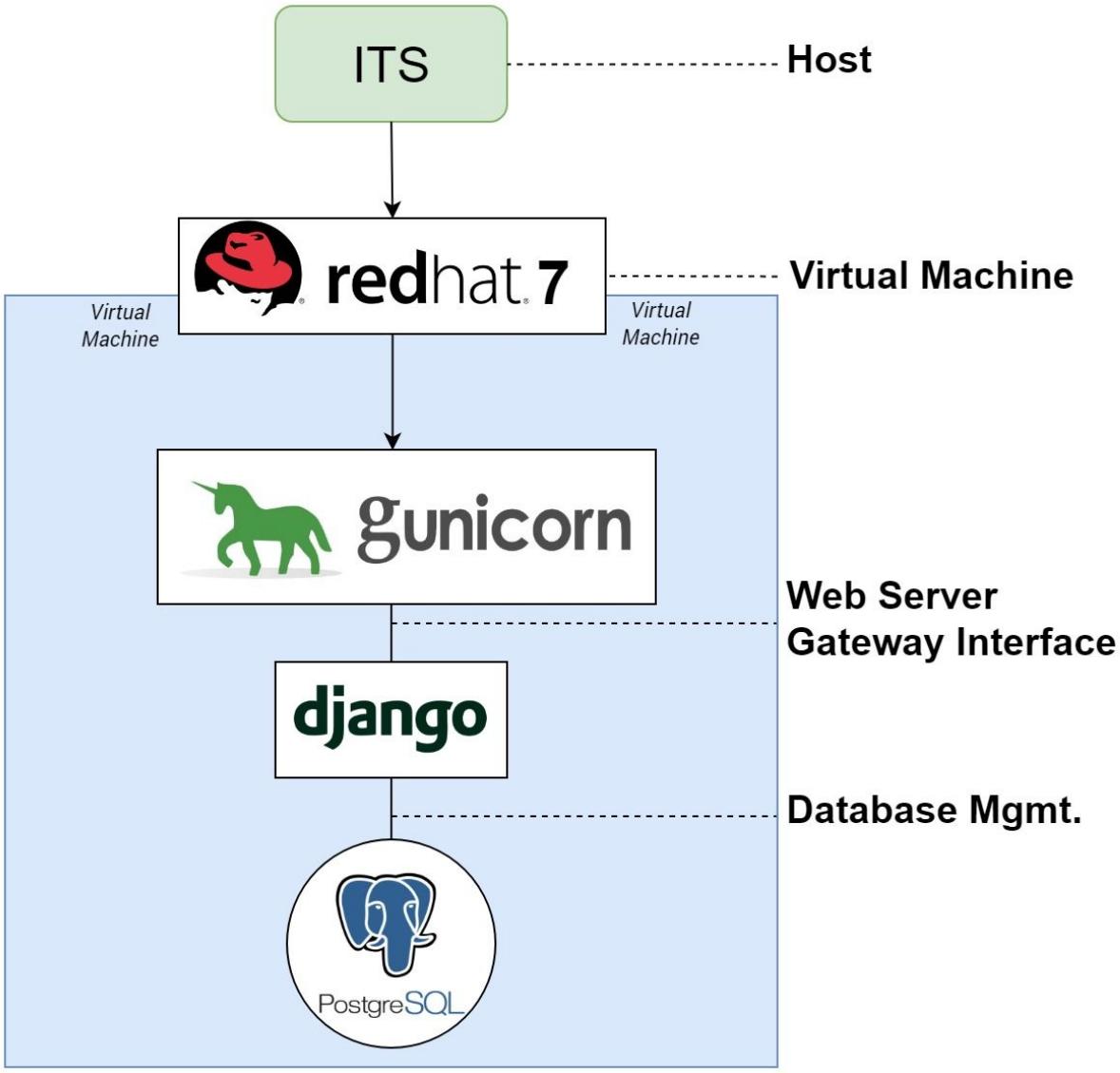
The overall functionality of our solution will have a very heavy focus in supporting our client’s workflow. We want them to be able to hop on our web portal, query a database for project information, and get quick and easy to read results. This also involves creating a seamless transition between project details without losing information. Finally, storing data from previous projects for analysis and statistics that will be used to support future projects is also an important aspect to our vision. The production version of the website will be hosted using NAU ITS, which will be in a Red Hat Enterprise Linux 7 virtual machine. The website itself will be built using Django v1.11, which is the latest long term supported version of the framework till 2020. The database back-end will be PostgreSQL 7 and Gunicorn will be used as the WSGI HTTP server for the website.

Our solution will fix the client’s problem by providing each stakeholder to this website a unique way of accessing the website that addresses their needs. Most importantly, it will decrease the workload with regards to finding and analyzing project data.

4. Architectural Overview

Below is a graphical representation of our system diagram and how each will be communicating with each other respectively.

Figure 4.1: Diagram showing the production hosting for the project



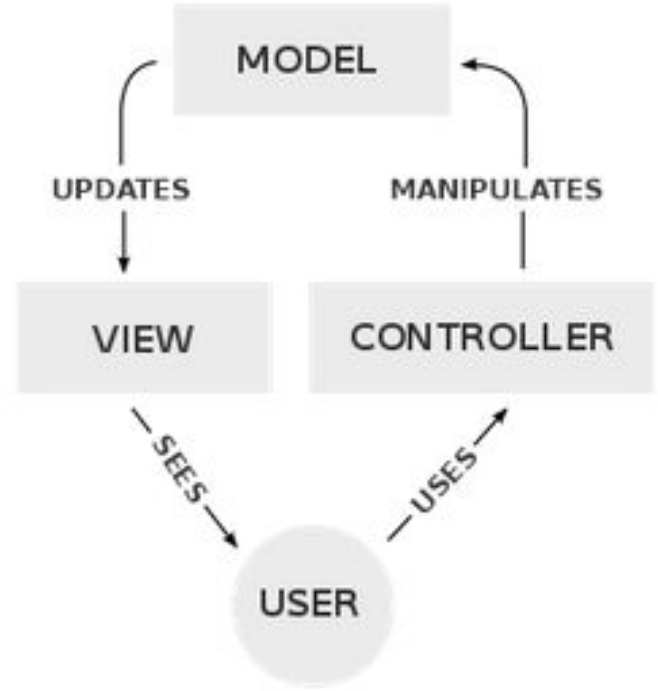
As shown by the diagram above, NAU ITS will be allocating space and resources for our product. The virtual machine that will encapsulate all of the rest of our software is going to be redhat 7. Gunicorn will be handling the WSGI, or Web Server Gateway Interface, and will provide end users with asynchronous calls to our product, which lets our product serve multiple end users at once. PostgreSQL 10.6 will be our chosen back end

and as shown in the diagram above Django is the heart of our system architecture. Django provides us with the ability to manage our front and back end simultaneously using one framework. This allows us and our client to keep our product centralized and simple.

Overview of MVC

Django uses an architecture pattern known as a Model View Controller, or MVC. An MVC pattern will allow the developer complete control over what the user sees by manipulating every step of the process. From receiving the request to sending data back to the user, this architecture allows for ultimate control on how content is served. Overall, this makes our product highly modular without drastically changing the foundation of the functionality.

Figure 4.2: Diagram showing the relationship between the various components of the MVC architecture



In Django the view and controller aspect is handled by the views.py inside of each sub-application. This Python file will handle not only what is shown on a certain page, but also the functionality and specific details of the page. The models.py will handle all the foundations of each model by allowing for code-first database construction and

management. In this file, we will design our model fields as well as design how different models interact with one another.

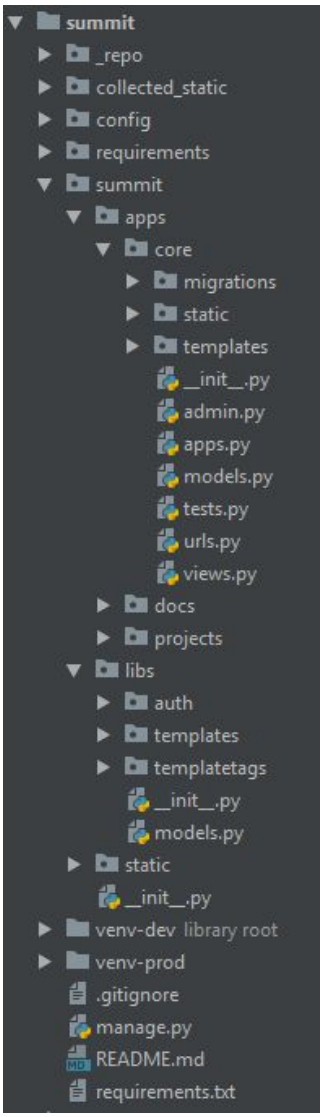


Figure 4.3: File structure of the overall project

File Structure

As shown by the figure below, we can get a sense of how our files are structured. In the root directory, our entire codebase for the project management system resides. Inside the summit/summit/ directory are what Django refers to as "apps." Each of these apps will have the same Python files like urls.py, models.py, views.py, etc. These apps have their own separate functionality and can be work without having to rely on other apps if necessary. There are also apps in the summit/libs folder that allow for us to develop custom, shared applications for features such as notifications and user authentication.

Key Responsibilities and Features

Our project has two overlapping, but distinct areas where the key responsibilities and features can be divided into: global and module-based.

Global

These features affect the entire website as a whole and are not bound to any particular app of the website. This ranges from how various elements may affect the look and feel of the website, who can see what, when on the website, and how the entire project itself is structured.

General theming and sitewide UI

To make a pleasurable user experience, we have elected to use Material Design for Bootstrap for our global theming. With this in place, we can deliver a satisfactory design for the website while also maintaining custom styling and scripts within our project and each application and custom library.

User Groups and Permissions

Our client has expressed the need to separate functionality and access based on user groups. That being said, it is common knowledge and a general security practice to separate user groups based on classification and create a tier permission structure to regulate what each group, and ultimately each user, can do.

Permission-Based Interaction and Views

By separating users into groups and assigning specific permissions, our solution will be able to programmatically and graphically give users different, personalized experiences and also respond in the same manner. For example, the public should be able to query through projects' public data where as administrators will have access to all project data.

Module-based

These features focus on the individual, independent sections of the website and how they interact with one another. These are the areas in which all the the features the essential features the client wants will be delivered.

Project Management

The "projects" app is what will be handling the project workflow. This will include being able to add and edit projects, view project details, gather important information for analysis, and archive old projects for future use.

Modification Process

The CPCESU will also be able to edit and modify projects since agreements are not static. This will grant a much easier workflow than their current modification process. More importantly, there will be a revision history function that will allow the CPCESU to see all the historical changes for each project.

Exporting Data

This function will allow the CPCESU to give inquiring organizations data they need to make informed decisions. This ranges from generating a spreadsheet of all the projects specific to an organization that has worked with CPCESU to generating a spreadsheet of all the funds that have moved through the CPCESU in the past fiscal year.

Documentation

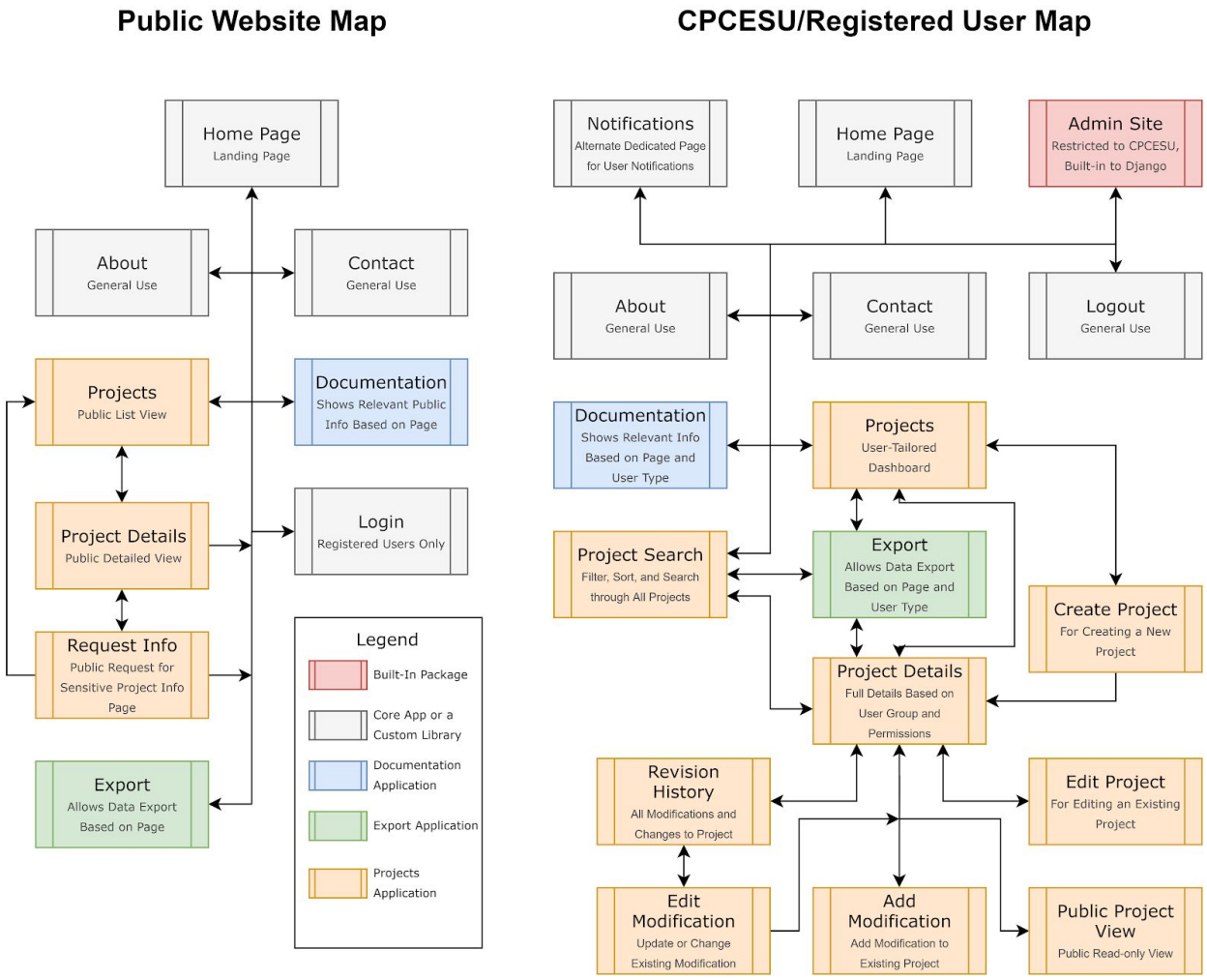
The documentation app is responsible for providing detailed information about the project for the user and the future developers of the project. By building the documentation into the project, we hope to make it easy to maintain and access user and technical documentation.

5. Model and Interface Descriptions

In this section, we have outlined the specific plans for each module. Included with every module is a detailed description and how it interfaces with the rest of the project, any relevant class diagrams and interface mockups, and a technical description of the coding practices in place to get the work done.

Before discussing every module, we have developed the following site map for both the public view (anonymous users) and the map for registered users, namely CPCESU for our deliverable this semester.

Figure 5.1: Sitemaps for both anonymous traffic and registered users



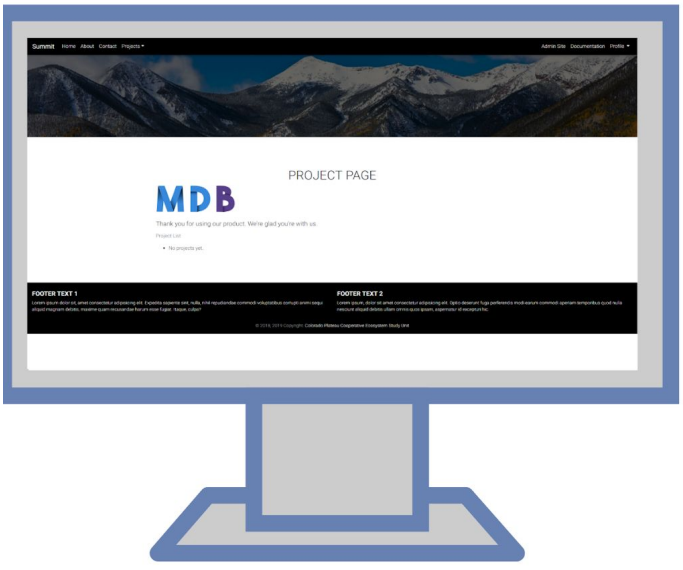
Some pages are the same (notably the “Home Page”, “About”, and “Contact” pages), but some are different experiences (projects list as a public anonymous user versus a registered user with more access) or entirely restricted to one group or the other. All of these pages require different permissions to access.

To account for this, we have split our major features into two groups just like before:
Global - Shared feature libraries between the individual modules
Module-based - Features that are located in specific parts of the project

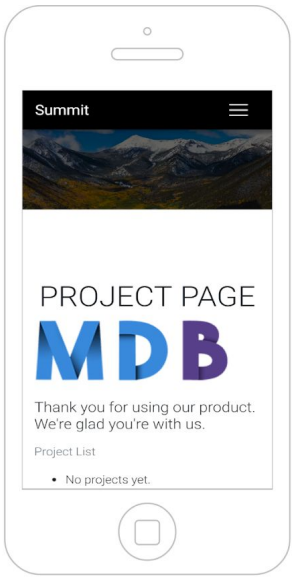
Global

General theming and sitewide UI

Great user interfaces and user experiences are critical for devices and projects to succeed. To facilitate a great user experience with a common theme and design throughout the solution, our team is using Material Design for Bootstrap as the default global stylesheets and JavaScript functionality then layering on our own static files. This allows the project to have one standardized theme easily without having to build the user interface from the ground up, which would not have time for during this capstone.



Desktop



Mobile

Figure 5.2: Same theme and user interface - just responsive to the platform

Another aspect of this sitewide UI is allowing developers to make use of a custom dynamic navigation menu. Using the same structure that already exists in Django, URL routes can now also be published as links on the navigation bar and include additional features and restrictions. This is in the /config/links.py file and example code is shown below in Figure 5.3.

Figure 5.3: Translating two of the Project App URLs into a dropdown menu with user restrctions (authentication). User is already authenticated to see this.

With this system, the user can have a harmonious experience while having the correct navigation and content served to them. The tiered stylesheets and JavaScript (global, then app-specific then page specific) allow for certain pages to be tailored to the user in different ways while still maintaining the current global theme. With the dynamic navigation bar, users have the same styled navigation, but are served the right links depending on their user permissions and group.

User Groups and Permissions

In order to facilitate unique user experiences without compromising on functionality or security, we have worked with the client to develop a general user group and permission feature. With this module, we can set users and profiles, assignment groups and permissions, and create barriers between information and access when necessary. To have the customization we need while maintaining the already existing authentication module in Django, we have wrapped around the internal library with our own and building off of the existing architecture. This is mostly creating the correct user groups for client and making new permissions than the built-in authentication would allow. This is graphically described on the next page (Figure 5.4).

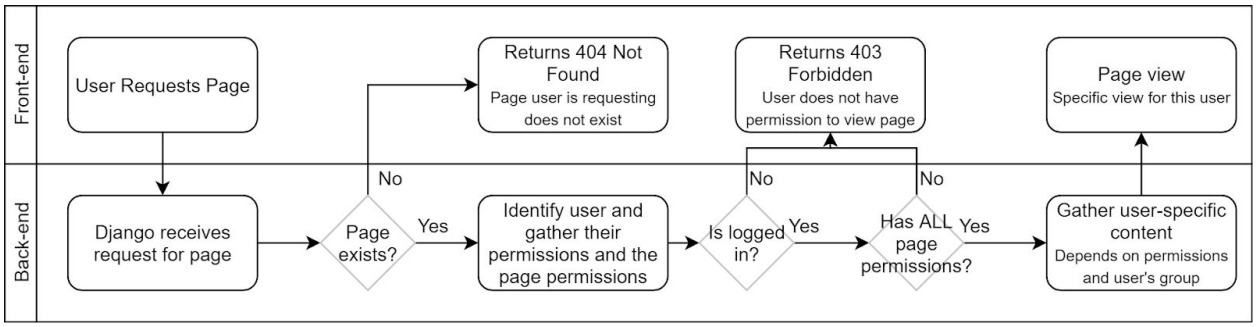
Figure 5.4: The three types of users for this app and their associated permissions

With this custom authentication library, the rest of the libraries and applications can access the new user model, the different group models, and the new permissions. These can help make it easier to check if certain users have permissions or not, sending the correct view depending on the user group, and allowing for custom actions in future applications and features.

Permission-Based Interaction and Views

Part of maintaining security and an enjoyable user experience is making sure that the user is getting the information that he or she desires and allowed by established user groups and permissions. This includes getting, updating, or posting data to the database; reading user documentation which may change based on the user's permissions; having the correct navigation based on user group; and so on. It is

Figure 5.5: Page request including validation for page request including user's authentication status and permissions



important that this feature helps keep the user out of areas they should not be by a means of graphical and stylistic design.

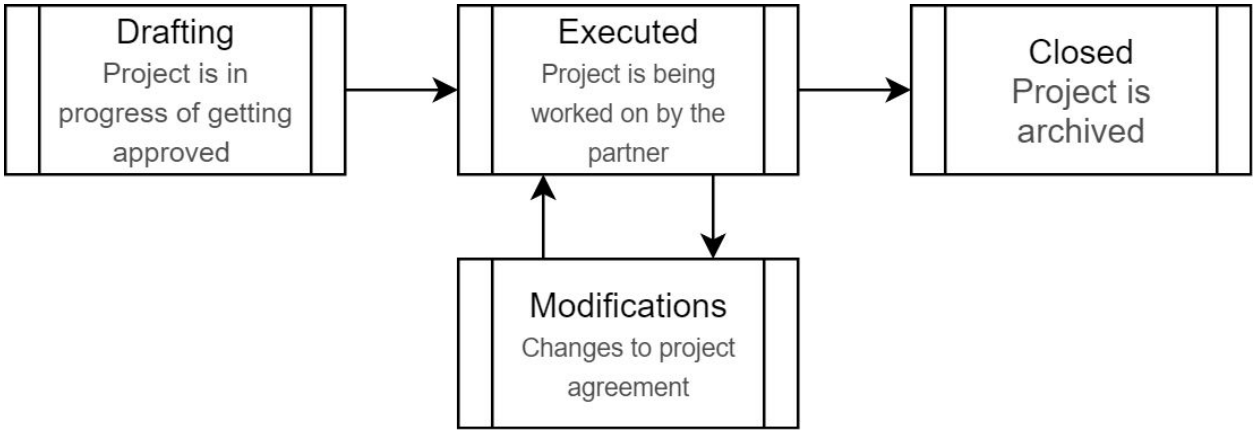
This is a site-wide feature that requires both our custom permissions and user groups with the applications tailoring certain users to specific activities or views. This way, the user is not given more information than necessary, which limits the risk of sensitive information leaving the system, while also preventing confusion and being responsive with what the user can and cannot do.

Module-based

Project Management

The projects app is what will be handling the project workflow. This will include adding a project, viewing project details and gathering important information for analysis, and archiving old projects for future use.

Figure 5.6: Summarized project lifecycle for this project management system with modification loop



This is the core of the project, the single system that all other systems feed into. As such, it is the pillar that will determine the ultimate success or failure of this project as a whole. The center of the project management system is the Projects Dashboard, which will show the user the status of each project in an intuitive way and allow them to modify the status of projects easily. Interlinked into that will be the core functions of the system, like showing the Project Details when a particular project is focused on, Create

Projects when they arrive, deriving most of the information automatically with a pdf scanner, and Revision History tracking all the while.

Modification Process

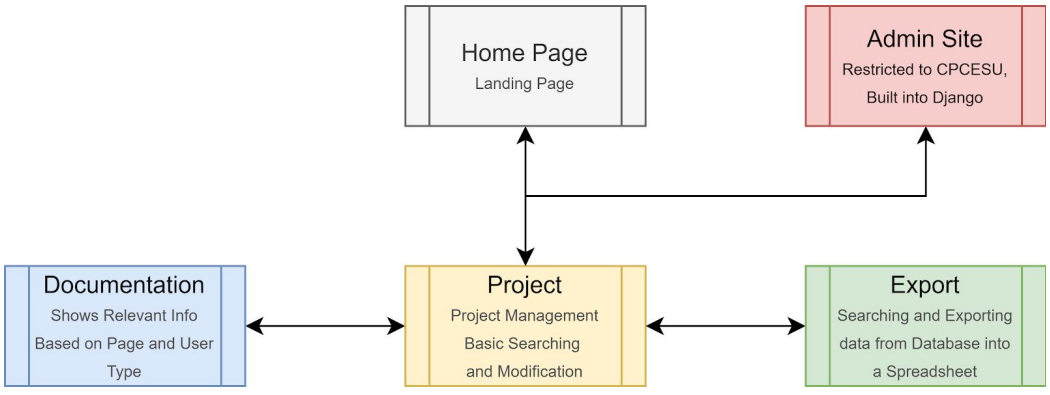
The CPCESU will also be able to edit and modify a project if need be. This will grant a much easier workflow then their current modification process. More importantly, there will be a revision history function that will allow the CPCESU to see all the changes on a project.

The modification process is a critical subsection of the project management system, accessible through the individual projects on it. The entire modification history of the project will be listed out for users to be able to determine any mistakes or the general history of the project. Although the scope is limited for this project, we want to design it so that it can easily be extended into a process for organizations to use to modify their own projects.

Exporting Data

This function is responsible for creating spreadsheets from various searches the user is able to perform on the database. This works as a separate app with a redirection to it in the project management page and the project list page for registered users of the system. The user will be able to search through projects with a general search bar or have the option to use several predesignated categories or options, like selecting from a list of organization names or not including funds in the generated data. The page will

Figure 5.7: Simplified Registered User Site Map focusing on the relationship of the apps



the give a look as to how the spreadsheet will look and what it will contain, with an option to download it.

This app is akin to the project search, but the function is to be used inside of the projects app, whereas this search with more options for the user tailored around generating a spread of the data in the database. This is also to future-proof the design of the website, incase the client wants to include more powerful report generating options, we will have an app set to do that.

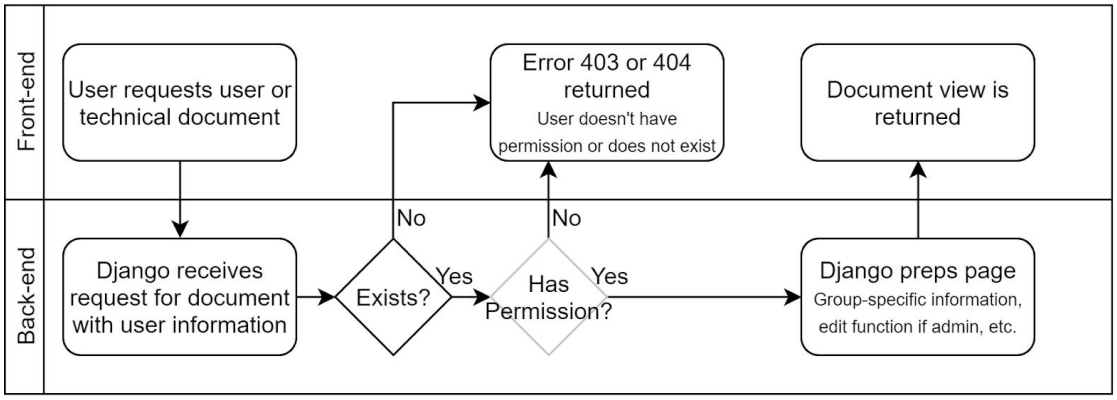
Documentation

This app will have two different faces to it, the first of which is an easily viewable help page for any registered user of the website that is focused on documenting the processes of the CPCESU and the purpose of any given feature. There will also a slightly obfuscated half, accessible by any registered user, containing the project documentation for a future developer of the website. This documentation will be included in the project to ensure that the documation of the project is always transported with the project.

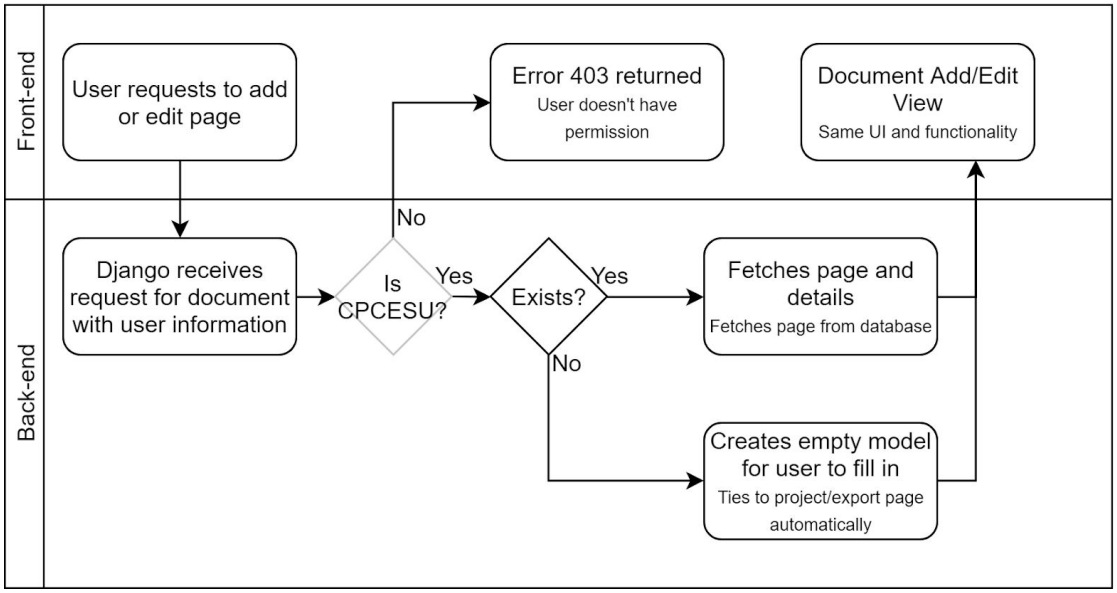
This app will have a link to it on every page with a feature specific to the project, viewable only by registered users of the project. This app will also have a link that will only be accessible through the settings menu of a given user that will take them to the developers section of the documentation. This can all be summarized as two request flows: one for all users reading documents and one for being able to add and edit documents.

Figure 5.8: Top - Reading document request flow | Bottom - Adding new or editing existing document

Document Get Request Flow



Document Add/Edit Request Flow



6. Implementation Strategy

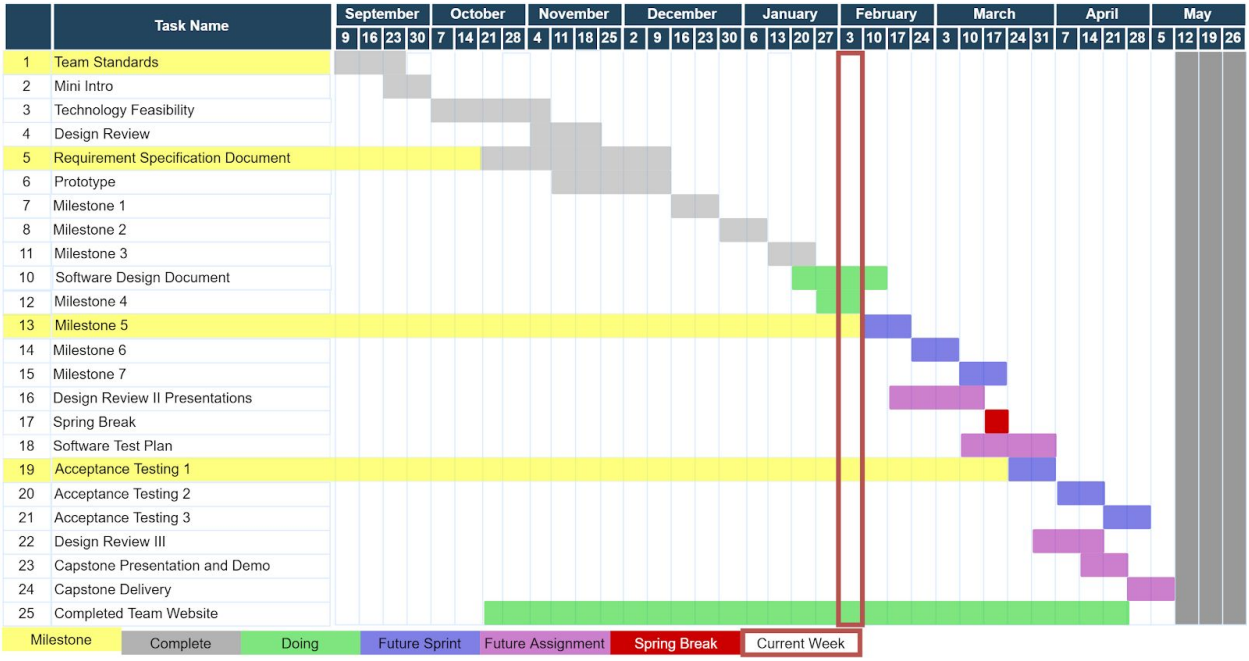
The project plan is an essential part to every software project as its primary purpose is to notice when the project is veering off-course as early on as possible, which allows for ample time to readjust the project and the final product.

To develop the CPCESU Project Management System named Summit, we plan to work in software sprints of two weeks, implementing at least a core feature every two weeks for the website. Throughout the sprint, the development server will host the website in its most current form and, at the end of the software sprint, a production release of the software is made to the production server. This will allow the client ample time to review the website and request any changes and submit and bugs during development. We will have a total of seven milestones throughout the semester up until spring break.

When Spring break is over (March 24th, 2019), we should have all major features of the website completed and the product will enter into a production environment as much as possible for the client to help determine the existing flaws with the product. These last three sprints will be called the Acceptance Testing sprints, and their purpose is the get the product fully ready to be adopted by the client. At the end of the final Acceptance Testing sprint, the product is delivered to the client and they will determine if they will accept the solution or not.

This is all visually outlined for the year in Figure 6.1 on the next page.

Figure 6.1: Gantt chart of our timeline for Summit development as of February 5th, 2019



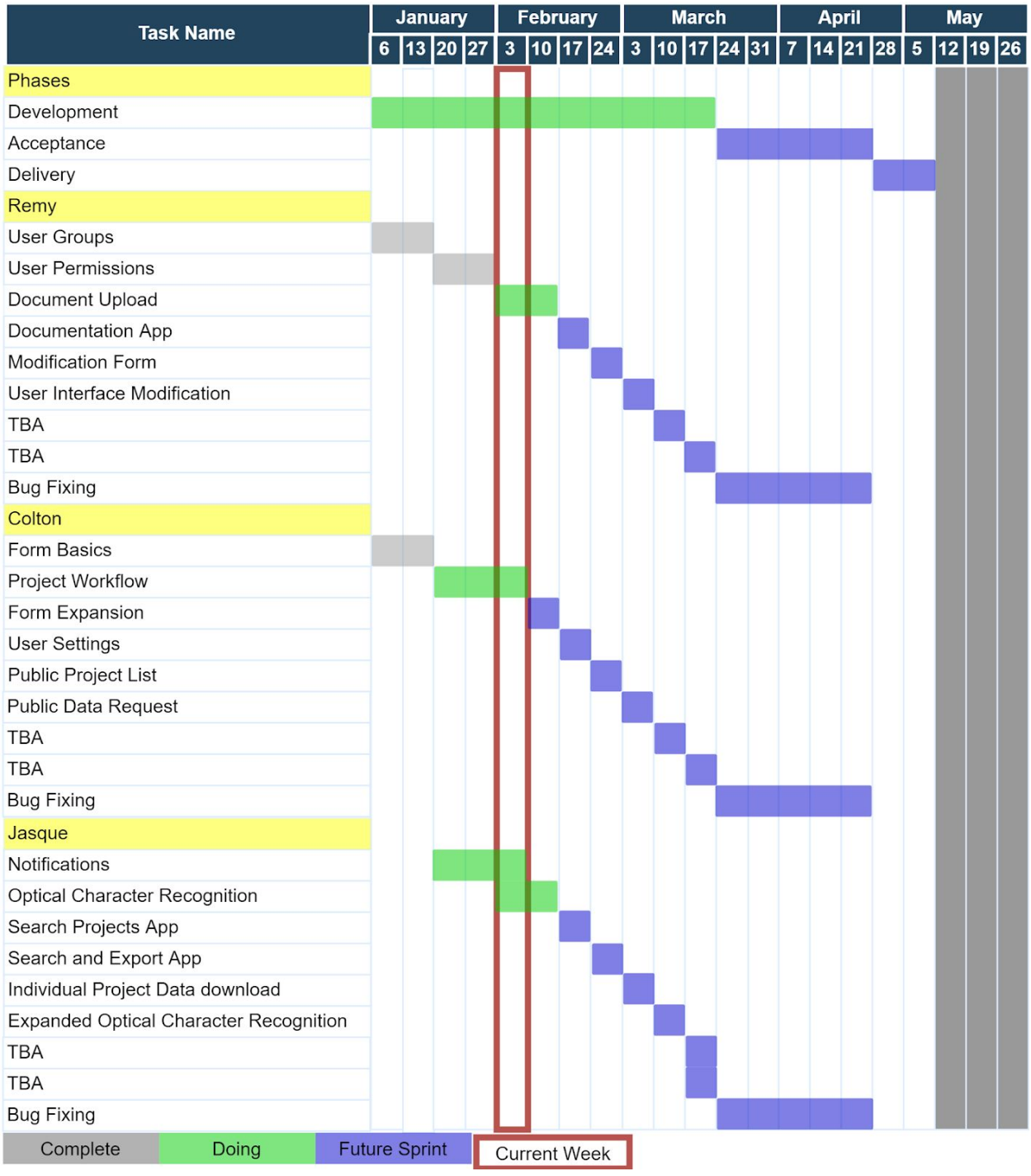
A break down of the goals of each milestone are given down below. In general, each major feature of the website has a setup, complete, and verified phase. The setup phase consists of a beta or partial implementation of the feature, complete phase consists of the feature completed, and verified phase consists of getting the feature looked at by the client and adjusting it to their comments if needed.

Sprint Name	Goal	Requirement IDs
Milestone 1	- CI/CD setup - Overall sitemap setup - Login/accounts setup - Forms setup	**F6.00**
Milestone 2	- Implement core project management ideas - Notifications barebones complete	**F1.00** **F13.00**
Milestone 3	- Additional Project management features and testing - Modification system setup - Forms completed	**F4.00** **F12.00**
Milestone 4	- Autofill forms setup	**F2.00**

	-	Project management verified	**F3.00**
	-	Modification system complete	**F10.00**
	-	Data export setup	
	-	Search setup	
Milestone 5	-	Data export complete	**F8.00**
	-	Search complete	**F11.00**
	-	Modification system verified	
	-	Autofill forms complete	
	-	Public facing website complete	
Milestone 6	-	Data export verified	**F5.00**
	-	Search verified	**F9.00**
	-	Autofill forms verified	
Milestone 7		*Buffer - Used for extra requested features and lagged schedules*	

However, no plan survives first contact, so the above Gantt chart has been broken down and re-calibrated for the rest of the semester. It shows our current status with the project and what major features we will have to tackle in the upcoming weeks to Spring Break (Figure 6.2, next page).

Figure 6.2: Gantt chart of each team member's timeline for Summit development as of February 5th, 2019



8. Conclusion

Our environment is an important part of who we are as human beings. As Carl Sagan once said, “it underscores our responsibility to deal more kindly with one another, and to preserve and cherish the pale blue dot, the only home we've ever known.” It is our innate responsibility to protect and preserve the Earth. Solving CPCESU’s problem with their current workflow and project management system would give them the ability to focus their time and effort on more pertinent matters instead of trying to find lost data. By building Summit for the CPCESU, we can support their passion for the preservation and protection of the Southwestern United States.

To review the problem, the CPCESU currently does not have a solid project management system that will allow them to store and retrieve data properly. Our clients would benefit greatly from an improved workflow and environment that is easy to use and maintain. The solution to this problem will be our project management system, Summit. Our web-based solution will provide our clients with a simple to use interface and streamlined database access, ultimately giving them the ability to effectively manage past, current, and potentially future projects. By providing data security, easy database queries, as well as a user management system with privileges, our clients can focus more of their time on important projects with quality trusted support from our project management system.

As ECOders, we have come up with a plan that will allot enough time for our team to bring a good and working product to our clients in May. By creating this document, we have laid out the coding and programmatic foundation for the Summit solution. This document will be our professional guide to executing this software plan and implementing the project management system for our client. The additional benefit is that this can be passed down to the next team or future developers so that our design methodology can be quickly and easily understood. The only caveat presently is that this document is subject to change as implementation may be different by the end of the semester - when the deliverable is complete.

Ultimately, with this software design document, we expect to easily complete our requirements and the various milestones along the way on the software-side of this capstone experience.