Computer Science Senior Capstone

CS 486C – Senior Capstone Design

ECOders

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

# Software Testing Plan

Overview: The purpose of this document is to describe how ECOders will ensure expectations and requirements by means of unit tests, integration tests, and usability tests.

***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

April 3, 2019 - Version 1.0

ECOders

This page is left blank intentionally.

# Table of Contents

This page is left blank intentionally.

# 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

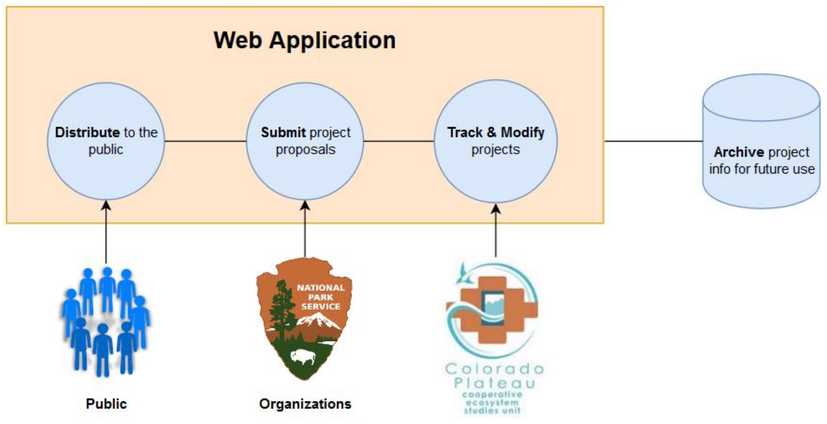
Chaudhry and Laurie Thom with developing a website and database solution for the CPCESU with three essential components: a project management system for the CPCESU; a website for showcasing public projects and information; and a system allowing organizations to develop, submit, and track projects. Our 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 testing document begins by outlining our project's overall implementation of our product, summit, followed by our plan for testing summit. Ultimately, the purpose of this document is to create various testing environments (unit, integration, and usabilities tests). Perhaps the most important aspect is to give the product to our clients early for hands-on experience and gather feedback. This will ensure that we have opportunities to make changes before the final product delivery at the end of the semester. This will also give us invaluable set of information that will improve our product for the end user, narrow the client expectation gap further, and assure requirements are being fulfilled.

# 2. Implementation Overview

Our solution to the CPCESUs problem, codenamed “Summit”, will be a web application with three primary sections: a public facing front end that displays projects and informs the public about the CPCESU, a section for organizations to view 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 clients’ workflows. 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 10 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.

# 3. Unit Testing

Unit testing is the lowest level of software testing to ensure that the components of a software product function they way they are suppose to. By validating each individual piece works as intended, we can ensure that the system does what it needs to do correctly, but that does not necessarily mean that each of the pieces will work together in harmony, which is under the purview of integration testing.

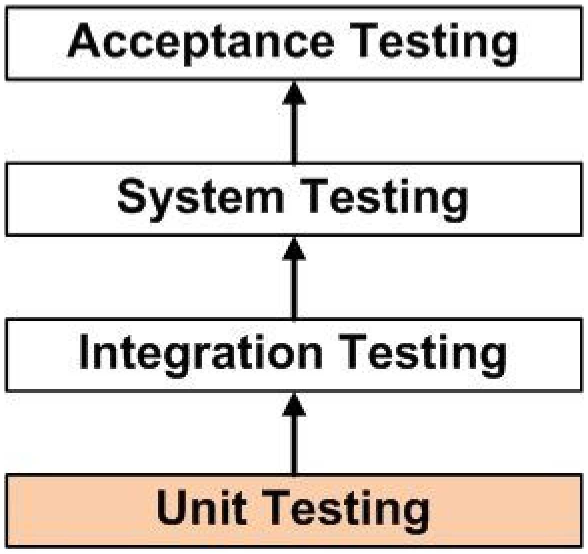


*Figure 1.1: The entire process of software testing*

Before we move forward, we must come to an understanding of what a unit to unit test is with regards to the CPCESU Project Management System. A unit is the smallest chunk of code that can be isolated and tested, which for most object oriented languages is going to be a method, which will be the case for us as we are working in Python

Our unit tests will use the Python Unit testing framework, which is an extensive and modern testing framework provided by the Python language. We will also be using functions provided by Django to help facilitate ease of testing with the Python Unit testing framework. For low level Unit testing, there isn't anymore technologies required due to the rudimentary nature of the tests.

As for our goals with the unit testing, we aim to have a code coverage score of 50%, as determined by the Django tool "coverage run". The reason why this is so low is because we have determined that many sections of our code doesn't require any tests, as those tests would be meaninglessly simple or not able to be done. One example of this is forms.py, which following Django's DRY model, don't have much in the way of testing. The forms are automatically created by Django which looks at the model the form inherits. If there was a custom Form class created to handle some special scenario in our code, that would warrant a test, but in our case, testing this file isn't needed.

Another goal of ours with unit testing is to have a test passing percentage of 90%. This is to allow use some leeway for solving various bugs we might come across while writing our tests that we might not be able to fix right at that moment.

Now we will go through each app and file of the project to specify whether or not tests will be required, and if they are, some example cases of boundary values that must be addressed

**Projects app**

This is the main app of the project, where most of the work is taking place. Most of the unit tests created for this project will be made and testing this part of the project.

- models.py
    - This file mostly doesn't require any software testing. The reason for this is because the file consists of primarily database table and column declarations disguised as classes and fields respectively, which is how Django abstracts database operations. These declarations do not need to be tested due to how simple their construction is. There are some custom methods in these classes, but they are currently only helper methods for manual operations, like getting the file path of a file. These simple helper methods don't require tests either, but there maybe the case where there is a method that declares some business logic, like some custom validation for what can go into a field. These methods containing business logic, which there aren't any yet, will need tests written for them.
- forms.py
    - Repeating from above, the forms file follows the Django DRY model and doesn't have much in the way of testing. The forms are automatically created by Django which looks at the model the form inherits. However, if

there was a custom Form class created to handle some special scenario in our code, that would warrant a test, but in our case, testing this file isn't needed.

- views.py
  - This is the core file that directs and manipulates web traffic the way it needs to go and how it is going to look. Every class and method will need to be tested to ensure that every view takes you to the web pages that you are suppose to go to and you 404 out of any pages you are not allowed to enter. There isn't any special boundary values to report here as we are only checking if the web traffic is being directed around correctly.
- urls.py
  - This file doesn't need to be tested as it is just a bunch of regular expressions dictating which view functions web traffic should be directed to render their webpage.
- tasks.py
  - This file consists of various celery tasks that can be run. Every function in this file will require a test to verify that it meets requirements and functions properly.
  - notifications() will require tests that will check that the same notification isn't added twice, and that the function will error if the date and time for the notification to occur on is negative.
  - read_pdf() function will be tested to ensure that non-pdf files are rejected, pdf files that are not Grant Cooperative Agreements are rejected, that the data from the pdf is saved into a model in the database with the correct values, and that the function doesn't fail when it's given a Grant Cooperative Agreement with garbage values, like "abc" in the budget columns.
- ocr.py
  - This file contains the method that actually deconstructs the PDF and interprets the values in it, storing the results in a dictionary for any function to handle the data.
  - option_interpeter() function determines if a checkbox in the PDF has been selected or not by checking if there are any non-white color values in the space. This will be tested by verifying it's functionality works correctly, which will be determined by feeding it a white image, a white image with a

    - single grey pixel, a white image with a single black pixel, and lastly a black image. This will handle all the various cases this function will go up against.
    - collect_data() function deconstructs the PDF into images and feeds those images into Tesseract. This function will be tested by feeding it non-pdf files, pdf files that are not Grant Cooperative Agreements, that the data from the pdf is saved into a dictionary with the correct values, and that the function doesn't fail when it's given a Grant Cooperative Agreement with garbage values, like "abc" in the budget columns.
- Template Javascript
    - Any javascript located in html files or separate files will not be tested as a unit test, but as an integration and usability tests

**Documentation app**

- views.py
    - This is the core file that directs and manipulates web traffic the way it needs to go and how it is going to look. Every class and method will need to be tested to ensure that every view takes you to the web pages that you are suppose to go to and you 404 out of any pages you are not allowed to enter.
- The models.py, forms.py, and urls.py will not need to be tested for same reasons as stated in the Projects apps for those files

**Auth app**

- Only the views.py file will need to be tested for the same reason as stated in the Documentation app and Project app, however the core logging in, logging out functionality will be determined by an Integration test.

**Core app**

- Files in this app will not be tested due to the lack of anything of note within it

# 4. Integration Testing

Another important factor of assuring product quality and integrity is to conduct integration testing. This makes sure that all of the systems - the web framework, the database, and the Web Server Gateway Interface (WSGI) - are all working properly.

**Goal and Overall Approach**

Our goal for integration testing is to make sure that all the essential technologies and dependencies are holding together nicely in development. Our overall approach for this style of testing will include pushin all available features to our development server. This will allow us to have an idea of how are product will be served to the end user and provide us with valuable feedback.

**Focus**

The main focus of the integration testing will be making sure our web framework and database are connected properly. Since these two components are the largest and most significant portions of our end product, there will be a big focus on how well they integrate with each other. To do this we will be testing separate features that specifically involve the web framework and/or the database.

**Features that will be tested:**

- Log in
    - A CPCESU employee/staff member should have the option to log in and have extended privileges compared to a normal user.
- Log out
    - The User should have the option to log out and lose their prescribed privileges.
- Project Creation
    - Initialize a project with correctly formatted data and confirm that the project was in fact created.
    - Intentionally initialize a project with bad data to confirm that there is proper control in place to prevent project creation.

- Public Project Table
  - With the privileges of a normal public user, verify that there is a public project view available.

**Confirming Results**

- HTTP responses, redirects, and user interfaces
  - Logged in CPCESU employee vs. anonymous public user
- Check database for results
  - Addition/Deletion of projects and their corresponding modifications.

**Conclusion**

Based on the information that will be gathered from the integration testing, our team will be able to confidently analyze the viability of our product and how well the technologies fit together. This will give our end user more assurance knowing our product will not only satisfy their expected outcome, but also assurance that summit will not fail because of the integration of our technologies.

# 5. Usability Testing

The last form of testing for our product is usability testing. This means that we will have ourselves and our clients individually go through the motions of user stories, such as creating projects, viewing our personalized dashboard, searching for projects and people, and so on. The goals of usability testing are to gather user feedback, make graphical and functional changes where necessary, and confirm requirement completion.

**What is Usability Testing**

Simply, usability tests allow end users to try out the product through guided by documentation, actively through demonstration by developers, etc., and provide positive and negative feedback that can help improve the product while development can continue. From passive observation of users and no guidance to active, hands-off demonstrations only by developers, there are varying degrees of control to collect data and make qualitative decisions. For the scope of this plan, we will balance both the freedom of our clients and creating user stories (with guides or walkthroughs) to ensure that we are able to collect valuable feedback for making changes.

**Considerations**

To conduct usability tests, we have outlined the nature of the project, our clients' knowledge such as technical capabilities, consequences of our current design, and other considerations. First and foremost, our project is a web application and the end users may have completely different experiences depending on the type of device and browser they use and what is their current browser's version. Even with the use of MD Bootstrap, our front-end framework for CSS and JavaScript, we should still take into account that the user interfaces may vary.

Next, we have two clients, one that is adept with technology and one that is capable, but may need more guidance. Regardless, we will need to write thorough documentation to help aid our usability tests before allowing a more hands-off approach. Lastly, there may be a hurdle for both of our clients to transition from their old system (Excel and Access primarily) to a centralized web application. We want our product to be the end-all, be all, however, we do need to consider some positive and negative transition effects between them that may result in changes in the product post-usability testing.

As for making alterations to the solution based on feedback, it could vary. From something minor like changing a button's text to larger changes including database changes, we will do our best to accommodate our clients' needs within reason. As long as changes are within the latest version of our Software Requirements Document and feedback warrants positive changes within rational guidelines, we will do our best to implement revisions.

**Detailed Plan**

Our current plan is as follows.

- First, we will create user stories to gather qualitative feedback. This mainly includes making sure that all of the various aspects of the project are covered such as creating and editing projects, viewing project details, and more. These will be written from the end user's perspective and will guide our usability tests.
- From the user stories, we will set milestones for the feedback we want to gather. This includes client satisfaction, ease of use, and so on. These metrics will be used to determine problem areas that need to be addressed.
- There will be at least four opportunities to conduct usability testing, two of which will be indirect (giving a manual and letting the clients work the product without any additional guidance from us) and two direct (where we will walkthrough the product with our clients and interview for feedback). Indirect testing will allow our end users to use the system on their own time and take notes of things that work or don't. Direct testing, which will be the week after indirect testing, will allow us to all collectively evaluate the product. These two methods will help give us time to address changes while also keeping constant communication with our clients.
- Since our project is a web application, we will include both text and images in our manual to assure that the user can still find a way to use the product. Also, our clients can let us know if the user interface is different than the figures used.

## Timeline

Since there are approximately five weeks left till the final product must be turned in, we will plan our user testing using this timeline.

- Week of April 8 - Indirect Testing
    - Give both Todd and Laurie a manual and a checklist of things to try out. The checklist will be based on our user stories and milestones for this round of usability tests.
- Week of April 15 - Direct Testing
    - During an in-person meeting, we will all collectively go over the results of the previous week. This gives our clients the opportunity to show us where works need to be done, where any confusion exists, and so on.
- Week of April 22 - Final Indirect Testing
    - In accordance with the capstone class, this is when we do our final round of demos and usability testing. This may have different user stories, depending on the results of the previous tests. This is crucial to help make sure that the hand off at the end of the semester goes well. This will have the near-final documentation and product. The goal of this round is to only have minor cosmetic changes and nearly all requirements fulfilled.
- Week of April 29 - Final Direct Testing
    - This is the last opportunity for our clients to show us in person what things need to be changed. Again, these should only be minor graphical changes.

Overall, with our usability tests nearly ready, we hope to gain further insight into what our clients' expectations are and making sure that we not only complete our functional and non-functional requirements, but that we are ultimately delivering a product that Todd and Laurie would be happy to use.

# 6. 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 clients. 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, this software testing document will give our team the opportunity to get valuable feedback from our clients. This, in turn, will improve our product, summit, as a whole and confirm functional and non-functional requirements along the way for final product delivery.