gusty.bike

Project Plan

David Velasquez
Chris Blankenship
Clay Sweetser

Section I: Introduction

Purpose:

The purpose of this document is to provide a general overview of the *gusty.bike* project, as well as a plan for development of the project.

Scope:

The requested project concerns the development of a modern, maintainable website that the client (a professor at UMW) can use as both a personal website and a teaching aid. Users of the software include the client, Gusty Cooper, and his students.

References:

- CPSC 430 Writing Assignment #2 Project Plan Document
- Group 5 Requirement Documentation
- Software Requirements Specification for Gusty.bike

Overview:

The remainder of this document is organized into the following parts:

- Section 2, "Project Description", in which the project and its client, users, requirements, and constraints are described.
- Section 3, in which the development plan for the project is explained. This includes the general approach towards development of the project, the project's milestones and deliverables, and its tasks.
- Section 4, "Appendix", in which supplemental information regarding this document is listed, including a glossary of terms, author information, and information about additional documents.

Section II: Project Description

System Overview:

The requested project concerns the development of a modern, maintainable website that the client (a professor at UMW) can use as a both a personal blog and teaching aid. Furthermore, the client's students will be able to use the website to submit assignments from the client's classes, and the client will be able to review the submissions.

Client Characteristics:

The client for this project is Gusty Cooper. He is a Computer Science professor at UMW. He has requested this project because he feels that his current collection of websites are unmanageable, and wants a modern website that he may use as both a teaching aid and personal blog.

User Characteristics:

The end-users of this system are as follows:

- Gusty Cooper
    - Client:
      Gusty Cooper will be logging into the website, modifying its contents, and logging into Mooshak to review student submissions.
    - Webmaster:
      Gusty will be able to maintain the website's functional components through a combination of scripts and plugin features..
- Students:
  Students will browse the site and its contents. They will be able to submit their assignments on the Mooshak extension for grading.
- Peers:
  Colleagues and the general public will be able to access the website and consume content.

Functional Characteristics:

The client should be able to log into the website in order to add new content and otherwise administer the site. The client should be able to log into Mooshak in order to manage programming assignments. Students of the client should be able to access the site in order to read the client's blog posts and class material, as well as be able to log into

Mooshak to upload class assignments for grading. Peers, and other visitors to the site, should be able to visit the website in order to read the client's blog posts and view other content.

General Constraints:

The website should easily integrate Mooshak2 for student and client use; this can be done using a reverse proxy (like nginx) to point at multiple web applications. The website should be secure; using a service such as Let's Encrypt to automate the acquisition of SSL/TLS certificates can accomplish this. The website should be hosted on a stable platform somewhere on the cloud; Amazon Web Services would be an ideal platform for this.

Section III: Project Schedule

Approach:

Our constraints will guide our approach. The main concern for the client is the ability for students to access the site and upload their projects to Mooshak. The software is free to use and has now been updated and runs off Java, however it doesn't specify other dependencies like the previous C-based version which used Apache and TCL. Our development will feature a two-pronged approach, developing the website with client-side services for our client and his audience and the Mooshak as standalone systems, and then ensuring their compatibility.

Milestones and Deliverables:

1. Installation on Local Server
    - Manually install components (incl. DjangoCMS, Mooshak, Nginx, etc.)
    - Manually configure components into a local web server
    - Access local web server and test extensions and add-ons

2. Automation on Local Server
    - Write, test, and debug script to automate the installation and configuration of components, including image display, blog posts (without comments).
    - Test functionality of components in real-world scenarios

3. Deployment of Web-Facing Development Server
    - Deploy the completed automation scripts onto a public-facing web server
    - Implement SSL/TLS and write further automation scripts
    - Create a fully-functional live demo of the website

4. Deployment on Production Server
    - Deploy the final automation scripts onto the real server
    - Assist client in adding content to the site

Work Breakdown Structure:

1. Setup Website Server
    - Time Estimate: 1 Day
    - Justification: Virtual Machines can be deployed through a cloud platform like Amazon Web Services in the matter of minutes.

2. Install Mooshak
    - Time Estimate: 2 Days
    - Justification: Once the server has been set up, the installation and configuration of Mooshak using automated scripts will require only the amount of time to run the scripts, change the administrator password, and make any other adjustments necessary.
    - Prerequisites: Step 1

3. Install DjangoCMS
    - Time Estimate: 2 Days
    - Justification: Once the server has been set up, the installation and configuration of DjangoCMS using automated scripts will require only the amount of time to run the scripts, change the administrator password, and make any other desired adjustments to the platform including the installation of extensions.
    - Prerequisites: Step 1

4. Create Initial CMS Content
    - Time Estimate: 2 Days
    - Justification: Once the installation has completed the initial content can be uploaded to the site and placed on appropriate pages.
    - Prerequisites: Step 3

5. Change DjangoCMS Appearance
    - Time Estimate: 4 Days
    - Justification: We will apply sleek and modern templates and basic color schemes to avoid a dated look. The color scheme will revolve around the color Orange, which is the client's favorite.
    - Prerequisites: Step 3

6. Setup Virtual Contest in Mooshak
    - Time Estimate: 1 Day
    - Justification: Once Mooshak has been installed, a test of its basic functionalities, including its virtual contest capabilities, will need to be tested.
    - Prerequisites: Step 2

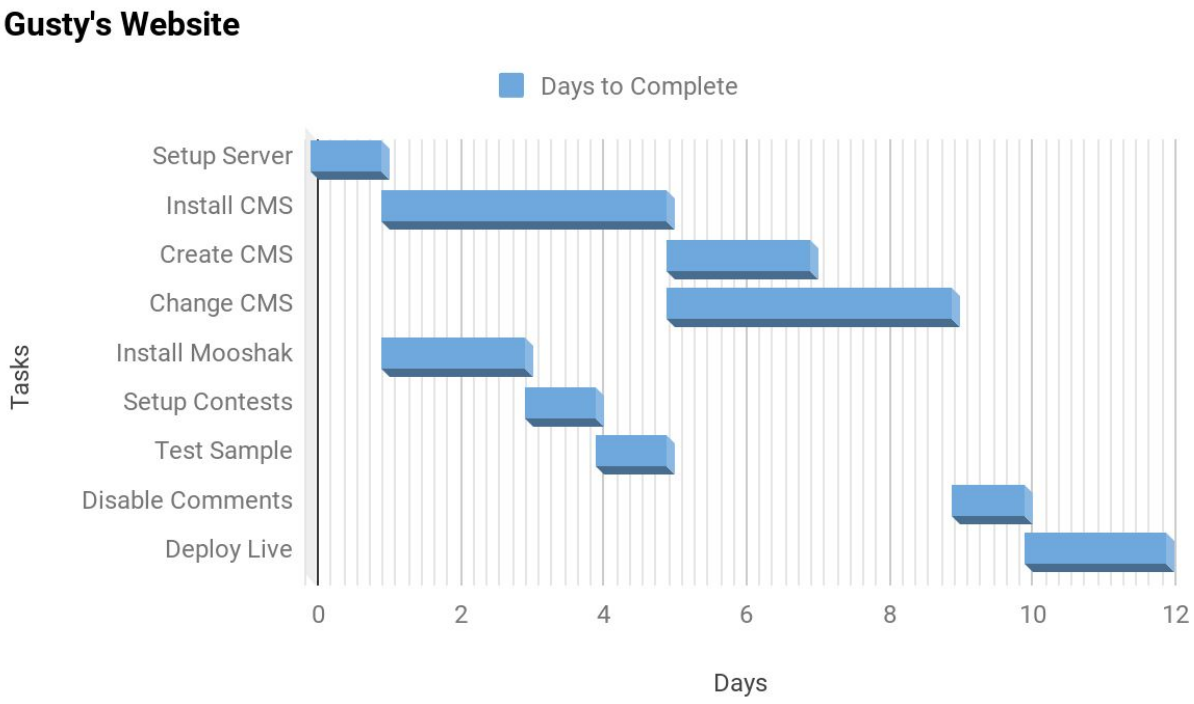
7. Test Sample Submissions in Mooshak
    - Time Estimate: 1 Day
    - Justification: After setting up the sample test we will know what we want the desired outputs to be, so testing will be quite easy. We can create a couple sample submissions which will create valid and invalid outputs to confirm that Mooshak is functioning as desired.

- Prerequisites: Step 6

8. Disable Comments:
    - Time Estimate: 1 Day
    - Justification: Once the CMS is in working order, comments can easily be disabled; that is if the chosen extensions even include them.
    - Prerequisites: Steps 3, 4, and 5

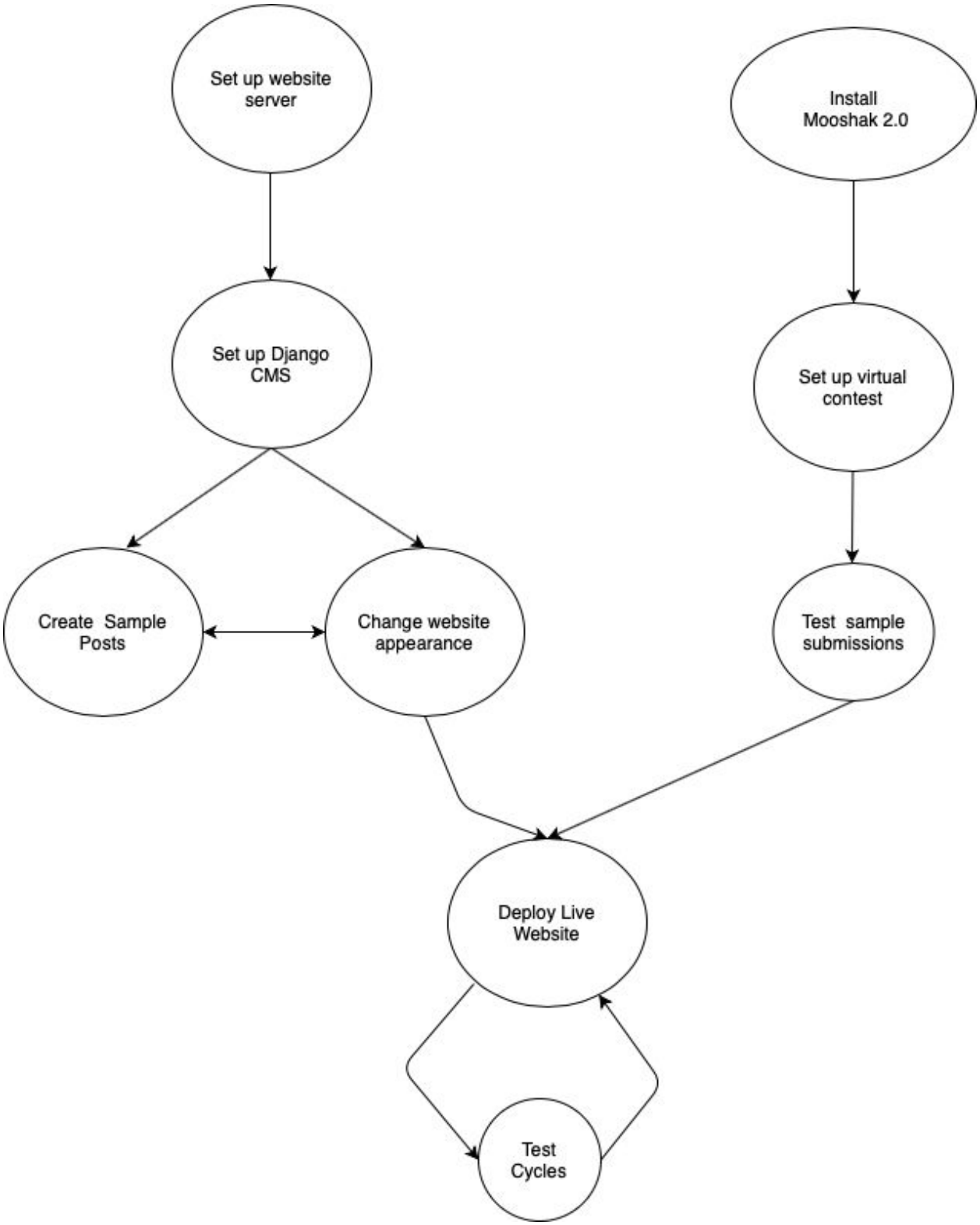
9. Deploy Live Website
    - Time Estimate: 2 Days
    - Justification: Once all software components have been installed, configured, and tested, deployment should just require the running of an automated script and minor adjustments on the hosting platform.
    - Prerequisites: Steps 5, 7, 8

Gantt Chart:



The above chart provides a rough estimate on the amount of time and order the various tasks required in the project will take.
Setting up the server the website will run on must be done before any other tasks. After server setup, both DjangoCMS and Mooshak may be installed, developed, and tested in parallel. Then, after some final configuration, the website may be fully deployed.

Task Dependency Diagram:



As described in the *Approach* section of this document, since the system has two distinct goals, we will tackle them as parallel systems.

First we will inspect our client's current cloud hosting environment. In that cloud environment we will install Mooshak, install dependencies and plugins necessary to launch the backend of the website like Django, Nginx, Python, and Java. Some of the development stages will happen concurrently, creating the website server and design as well as Mooshak's server-like service. On the website server we'll install Django and DjangoCMS as our framework for our client's content creation system.

When our client begins creating sample content to test the CMS, we will begin changing the cosmetics of the site. Our client's favorite color is orange and we will incorporate that into the color scheme, as well as his existing logo. As we go through that small cycle of content creation and cosmetic changes with client feedback, we will also begin ensuring Mooshak and our system are working compatably. This will lead us into the final rounds of testing and potentially implementing a few more supplementary features.

Section IV: Appendix

Glossary:

Apache:
Web server software commonly used to host web applications
(https://httpd.apache.org/)

CMS:
Content Management System
A website/web application primarily used to organize and display textual and visual content and media.
(https://en.wikipedia.org/wiki/Content_management_system)

Django:
A Python-based web framework
(https://www.djangoproject.com/)

DjangoCMS:
A CMS built with Django
(https://www.django-cms.org/)

Nginx:
Web server and reverse proxy software. Also commonly used to host web application.
(https://www.nginx.com/)

TCL:
High level programming language used to build GUIs.
Often embedded in C.
(https://www.tcl.tk/)

TLS/SSL:
Encryption protocol for web communication
(https://en.wikipedia.org/wiki/Transport_Layer_Security)

UMW:
University of Mary Washington
(https://www.umw.edu/)

Author Information:

Clay Sweetser

- Section One
- Section Two (System Overview, Client Characteristics)

- Section Three (Work Breakdown Structure)
- Section Four (Glossary)
- Gantt Chart and Gantt Chart Write Up

Chris Blankenship

- Section Two (Functional Characteristics, General Constraints)
- Section Three (Milestones and Deliverables, Work Breakdown Structure)
- Section Four (Glossary, Additional Documents)

David Velasquez

- Section Two (Client Characteristics)
- Section Three (Approach, Work Breakdown Structure)
- Dependencies Diagram and Dependencies Diagram Write Up

Additional Documents:

Existing Websites:

- http://gusty.bike/
- http://gustycooper.org/
- http://gustycooper.github.io/

Client's Logo