Untitled Developers

Herbie Duah

John Loudon

Michael Ortega

Luke Sanchez

Capstone Team Project: MSI Web 2.0

Design Document Ver. *1.4*

CS 486c - Spring 2016

2/18/16

Abstract

This document embodies the architecture of the software system that the *Untitled Developers* capstone team is designing for their client, Bjorn Krondorfer, director of the Martin-Springer Institute located at Northern Arizona University. The specific requirements that the software system is scheduled to meet are detailed in a separate requirements document. This document details the implementation design choices that the team has decided to pursue.

Table of Contents

Introduction

The Martin-Springer Institute has come to us requesting an overhaul to their current exhibit website. The system that we are designing will make use of the WordPress content management system which is widely used on the World Wide Web. The key goal of the developers of this system is to make a highly robust, modern website that is subject to modification by the client. We will accomplish this by pursuing the following features:

- Multi-browser compatibility (Firefox, Chrome, Safari, Opera, Internet Explorer)
- Responsive styling
- Maintainability[1]

We will achieve these goals by structuring our design around a solid architectural foundation which will be detailed in the remainder of this document. The design choices made in this document reflect our aims to satisfy each one of our key architectural features.

[1] As a key requirement to the system, user documentation will be included that instructs the client how to change content

Architectural Overview

The most basic design pattern that underpins our entire system structure is the *Client/Server* architecture. *Design patterns* are generally reusable solutions to common problems in the field of software engineering. In the *Client/Server* architectural pattern, our system will receive client browser requests and respond with messages according to HTTP and other network routing protocols. As a web-based system, there are a huge amount of external components that will be interacting with the system that we are developing. The DNS[2]/HTTP connector between the client browser and the host server is represented in figure 1 below as a dotted line. The bi-directional interfaces that bridge this connector with the rest of the system components are represented by the bold, black circles pictured below..

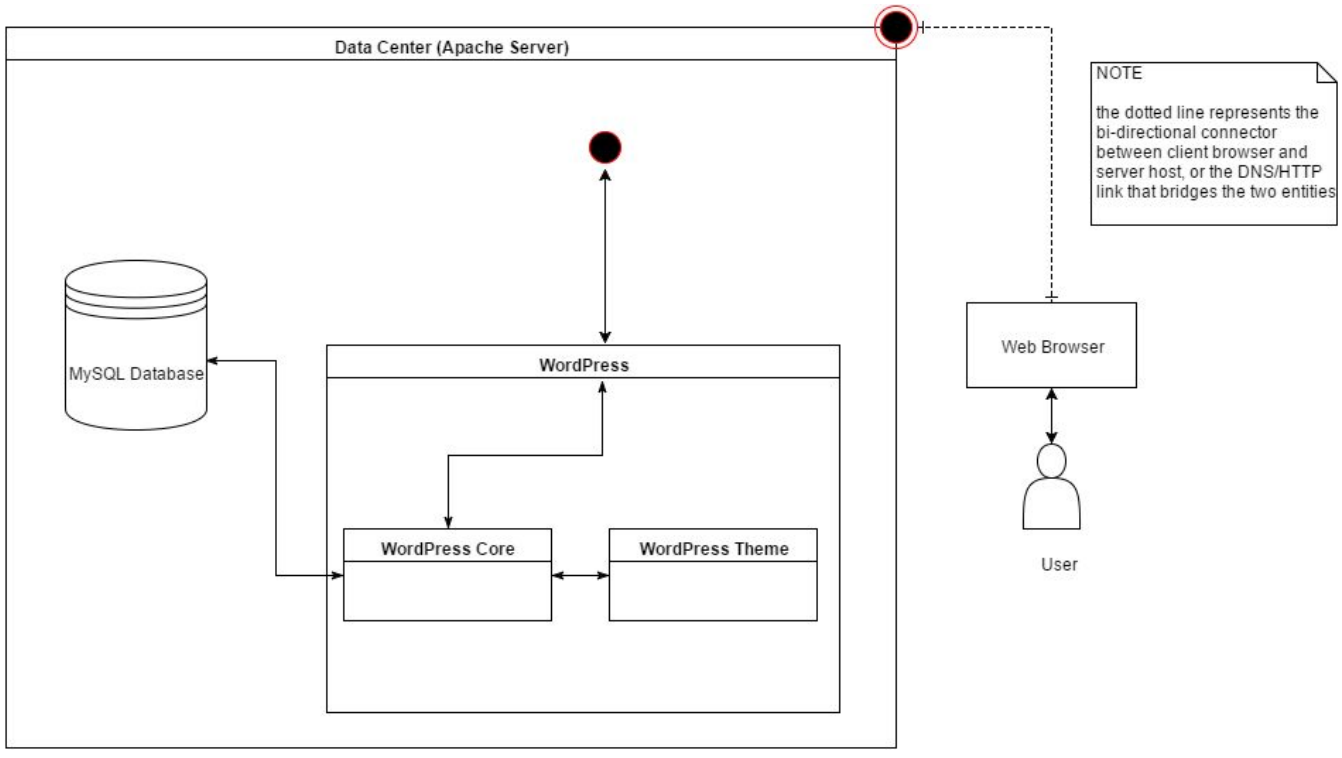

Fig 1. System Environment(s)

[2] DNS is an acronym for *Domain Name System* which is a hierarchical naming system for distributed computers and other components on the internet

Component Overview:

Listed below are the key components that make up the body of our system. These components can be seen in a graphical chart by referring to Fig. 1 on the previous page. We will now explore the components in further detail:

Web Browser - The users of the system will be accessing it through a web browser with an internet connection. The browsers that we have targeted for development are *Firefox, Google Chrome, Safari, Opera* and *Internet Explorer.*[3] Our architecture will include functional interfaces between these browsers and our final web system.

Data Center (Server) - The website will be hosted on LAMP(Linux, Apache, MySQL, PHP) stack server courtesy of NAU. This component will house all the data related to the website and acts as the interface between user requests and resource deliveries.

MySQL Database - The data center houses a MySQL software suite which implements basic database functionality. This MySQL database will be configured to work with the WordPress Core component.

WordPress (Core) - The WordPress core is the set of core functionalities that the WordPress CMS system comes bundled with. These core functionalities are responsible for generating the content in response to client requests, and are also responsible for aiding the MSI Director in

[3] These browsers were chosen as our target platforms based on browser usage statistics available here:https://www.w3.org/WAI/intro/wcag.php

updating the website with new content. This component will be the ‘switchboard’ through which all other components operate.

WordPress Theme - The ‘Theme’ component is the name for the directory in which most development work will be done. In this component are style definitions and *template-tag*[4] definitions which will play a key role in building our custom theme.

JavaScript Visualizations (not pictured) - This component will house the various scripts that are required to load and display our custom JavaScript visualizations. JavaScript is a client side programming language that is executed in the browser - these scripts will define and display the map and timeline visualizations that enable the website to be more engaging.

[4] Template tags are described in detail in the *Module and Interface* section of the document

Module and Interface Descriptions

Data Center (Server) Modules

Module 1 - Apache Server Software

The hosting server that we will launch our system on will be bootstrapped by the Apache software suite. This software comes pre-bundled with application layer networking protocols to route our payloads to clients over the internet via DNS/HTTP.

Module 2 - The MySQL Database

The server will host a MySQL database which will be used to store all textual content related to the website. This database will be used to organize dynamically updatable content through the content management system *WordPress*. This content management system is a complete package of predefined php files (as pictured in Fig. 2 below) which allows the client to modify and update content through an intuitive front-end interface.

WordPress Core Modules

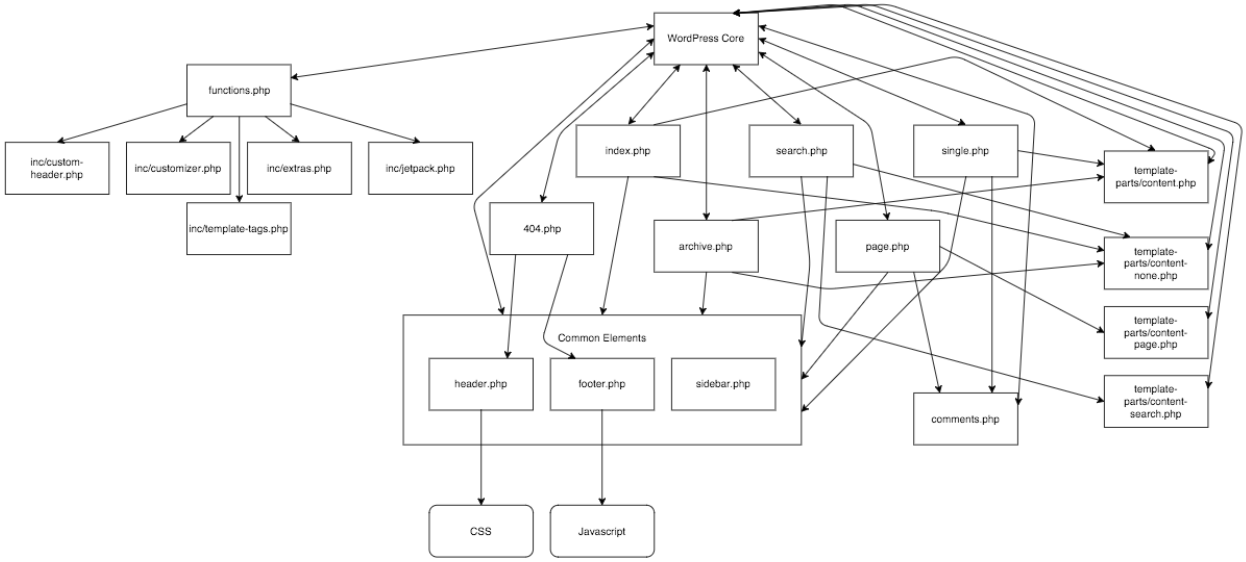

Fig. 2 - WordPress Core - key file dependencies distilled

Module 1 - The WordPress 'Loop'

The 'Loop' is a key module inside of the WordPress CMS component that facilitates the retrieval and display of the content of the website. The structure of the 'Loop' is a basic PHP script which calls an array of different 'template tags' and other built-in WordPress function calls. The 'Loop' is the point of entry into the core of the website. The loop begins execution in the file 'index.php' which is seen directly below the *WordPress Core* box of Figure 2 which is seen on the previous page

Module 2 - Template Tags

Template tags are an abstract data type defined by the *WordPress* core library. These template tags used in tandem with the loop allow pages to be built dynamically through PHP calls within the *WordPress* core and the MySQL database. Template tags are composed mainly of HTML

container elements and PHP conditional scripts. The template tags can be seen on the right side of Figure 2 pictured on the previous page.

Module 3 - CSS Style Specifications

The CSS design choices are vital to the project's success and are abstracted into a single file named *style.css* in the main theme directory. The file is divided into multiple sub-sections that detail specific, element-style choices which are separated by block comments. These styles will be used by template tags to create stylized template pages that will be used by the client to display important information across the website in an aesthetically pleasing container[5].

Module 4 - JavaScript Visualizations

The system will deploy two unique javascript visualization modules, making use of prewritten javascript libraries. These libraries, *Leaflet* and *Vis* will be used to create an interactive map and timeline respectively. These visualizations will be hand crafted to accompany the content currently found on the MSI website. The modules will be housed within the *WordPress* core directory under a folder called named ‘js’.

[5] Containers are an abstraction element that encapsulates the borders of information displayed on a web page independent of screen dimensions

Implementation Plan

The implementation of the system is following the AGILE software development methodology. Team members meet every week to review the project's current status and set sprint goals for the next weekly meeting. The team is also meeting bi-weekly with the client and weekly with our mentor to ensure that our architectural decisions are aligned with the client’s wants and needs.

The essential nature of a website system is multidimensional in its components, and is difficult to encapsulate in a typical work-breakdown structure. Pictured below (Fig 3) is our projected development phase lifetimes and artifact delivery dates (Fig 4).

Fig 3. Development Phases

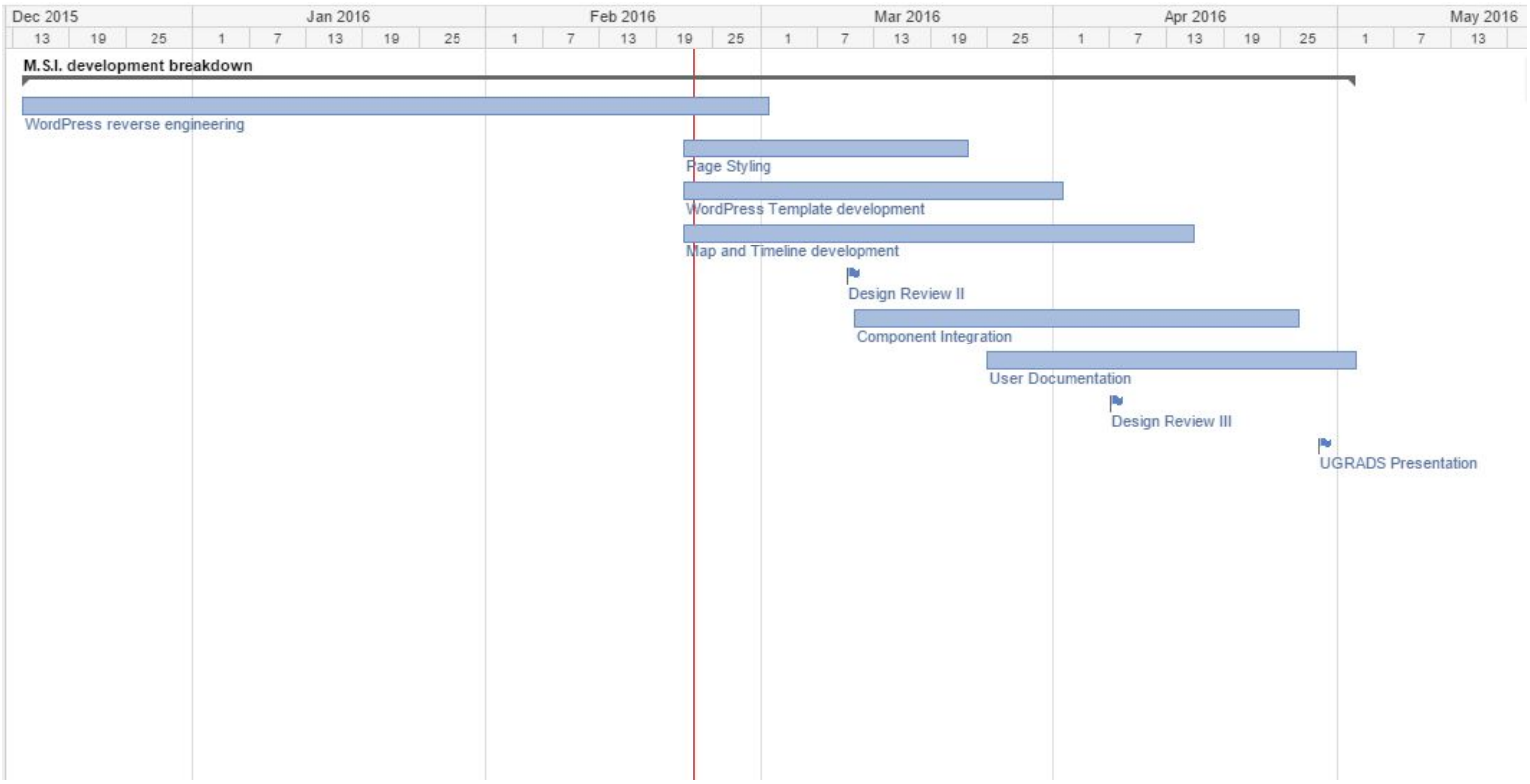

Fig 4. Hard deadlines for development phases and artifact deliveries

Task name	Start time	End time	Duration month
Total estimate	08/02/16 12:50	05/05/16 12:57	2.88
Not assigned	08/02/16 12:50	05/05/16 12:57	2.88
WordPress de-architecting	08/02/16 12:50	10/03/16 13:50	1.03
CSS Style drafting	08/02/16 12:50	10/03/16 13:50	1.03
WordPress theme construction	08/02/16 12:50	09/04/16 10:50	2.01
Map and Timeline development	08/02/16 12:53	09/04/16 10:53	2.01
Component integration	08/04/16 12:54	30/04/16 05:54	0.75
User documentation	08/04/16 12:55	01/05/16 18:55	0.79
Architecture specs 1.0	11/02/16 12:55	11/02/16 12:55	
System fully deployed	05/05/16 12:57	05/05/16 12:57	
Architecture Documents updated	11/02/16 13:02	13/03/16 14:02	1.03
Architecture specs 2.0	12/03/16 13:03	12/03/16 13:03	

Team Roles

An essential component to our implementation strategy is the assigning of specific tasks to specific group members. These basic roles within the team are organized as follows:

Luke - Responsible for team assignments, ensuring the project moves forward with regularly scheduled meetings and development and testing of web pages integrated with WordPress.

John - Responsible for the organization and maintenance of the system source code repository. Also responsible for development and testing of of web pages integrated with WordPress.

Herbie - Responsible for rolling out new layout designs and collecting feedback from client to satisfy client needs and coding style decisions in CSS.

Michael - Responsible for researching and implementing highly engaging and interactive map and timeline technologies.

References

Current MSI Website - http://bedzinexhibit.org/exhibit_home/

Mobile Friendly Testing - https://www.google.com/webmasters/tools/mobile-friendly/

WordPress CMS - https://codex.wordpress.org/Developer_Documentation

Writing Effective WordPress Documentation -

https://www.smashingmagazine.com/2012/07/writing-effective-wordpress-documentation/

Design Patterns - Taylor, Richard N., Nenad Medvidović, and Eric M. Dashofy. Software Architecture: Foundations, Theory, and Practice. Hoboken, NJ: Wiley, 2010. Print.

Gantt Chart maker - wrike.com