INGENUITY POWERED BY

BIMSHIFT

Final Report

BIMShift Team #1

Josh Wretlind
Jon Weldon
Brandon Rodriguez

June 17, 2014

Table of Contents

Introduction

About BIMShift

Headquartered in Littleton, Colorado, BIMShift's senior management team graduated from Colorado School of Mines. BIMShift is a company that creates custom software for Building Information Modeling (BIM) data. Their clients are primarily active in the Build-Design (BD) phase of building construction and management. The company helps contractors, and building owners, organize project assets such as work orders, photo and video documentation, and building auditing. Additionally, BIMShift solutions allow the aggregation of data from numerous other tools popular in the BD phase of building construction.

Their flagship software, DATAGate, offers users a cloud based service capable of generating reports and pulling data from Revit, Microsoft Access, and SQL files. Adding to this functionality, DATAGate is also able to be accessed on mobile devices with an internet browser, allowing reports to be viewed and downloaded on the go. DATAGate is also integrated with Tap Track™ technology which allows assets to be tagged with QR/NFC codes and easily scanned with a smartphone or other device to bring up detailed asset information.

BIMShift is on the bleeding edge of developing software for the emerging BIM market. Their DATAGate product is already being used as a primary organizational tool for several clients, and they plan on introducing other tools to help manage projects at a higher level, such as Project Track.

Product Vision

Project Track is a software solution delivered as a web application for managing BIM information and maintaining accountability among engineers throughout the life of a project. It acts as an aggregator of data where members of a master contractor can log information about different BIM features, contact other team members, and schedule face-to-face meetings.

An online survey returned Trimble Vico's 5D Virtual Construction Software as a similar software package for managing and scheduling projects throughout the construction phase of a building. While Vico's software appears to allow many of the same features Project Track aims to include (meeting scheduling, management reports, and BIM estimation tools), it does not seem to offer a method for enforcing the use of any part of the program or any specific workflow.

The critical, game changing feature of the Project Track application is its ability to enforce a specific workflow, and to control which team members are able to edit project information. The workflow enforcement is augmented with email notifications, so the entire team is kept up-to-date with the current project state. The end result is a product which is intuitive to use while also being capable of handling complex business logic and rules that can be tailored to any specific business model.

Scheduling meetings in Project Track is designed to integrate well with Outlook and other mail providers with calendar access. A meeting created in Project Track is sent out in a calendar compatible (iCal) format so the event can be easily added to the user's calendar with the click of a button. The RSVP functionality also alerts the meeting organizer of current attendance.

Project Track is built to load as fast as possible on any device. We accomplished this task by relying heavily on asynchronous requests for data, providing the user with only the information they ask for, and none of the information they don't. Using that design mantra, we are able to reduce the amount of information transferred between the client's computer and the BIMShift server, as well as lightning fast page-load times in the browser.

The goal of creating the Project Track software is to fill a void that no existing software provides today. Building contractors and owners are increasingly relying on different suites of software products and different teams of vendors to deliver cost effective solutions. Project Track brings all of these worlds together on all of your devices. BIMShift's dedication to using cloud technologies will also ensure clients are never left with downtime or without program updates.

Requirements

Functional Requirements

Project Admin Page

The "Project Admin" page should allow a user to select a project that he/she is part of, or create a new project. Projects that are not awarded should disappear from this list, and projects the user is not a part of should not be listed on this page. Users should be able to view this page without having a project selected (since they will use this page to select a project).

Project Information Page

The "Project Information" page should allow the user to input information regarding the name, description, billing status, contract type, location, cost, job number, and start/end dates. The page should be viewable with or without a project ID specified. If a project ID is passed in, the page should check whether the user has appropriate permissions to view the page, and if they do, populate the form fields with existing data so the user can update the information. If no project ID is specified, offer the user a blank form to create a new project.

Project Team Page

The "Project Team" page should allow the user who created the project to give other users view or edit access to the project. A project team can be infinitely large, so the user should be able to add additional users past the default number via AJAX[1] requests. The page should also implement an autocomplete feature, so that users can select existing users in the system, if they exist. A project ID must always be specified when viewing this page, as it is impossible to have a team that belongs to no projects. If no project ID is specified, return a error message to the user.

[1] Asynchronous JavaScript And XML. Although our backend is *technically* AJAJ, as we use JSON instead of XML, we will follow industry standard and refer to it as AJAX anyways.

BIM/VC Overview Page

The “BIM/VC Overview” page should allow the Project Executive team member to log high level plans (i.e., 3D Modeling, 4D Scheduling, Integrated Work Plan) in the database. A project ID must always be provided to view this page, or an error should be returned.

Award Project Page

The “Award Project” page should allow the project creator to input whether or not the contract for the specified project was won or lost. Awarded projects will continue to go through a predetermined workflow, whereas non-awarded projects are archived (the data does not get deleted, but is marked as archived, so it will not be shown to users). A project ID must be specified on this page, or an error should be returned.

Choose Project Kickoff Meeting Page

The “Choose Project Kickoff Meeting” page should allow the Project Executive team member to select a date and time for a face-to-face meeting to occur. Upon selection of this meeting, an email should be sent out to team managers with an iCal calendar invitation attached. A project ID must be specified on this page, or an error should be returned.

BIM/VC Extended Page

The “BIM/VC Extended” page replaces the “BIM/VC Overview” page. Once a project is awarded, the “BIM/VC Overview” page is no longer displayed to users (the data is archived) and the extended form is shown. The extended form should offer the project creator an interface for assigning engineers to specific BIM/VC features, specify start and end dates for these features, indicate whether or not this specific feature is requested, and include additional notes unique to the project. A project ID must be specified on this page, or an error should be returned.

BIM/VC Team Page

The “BIM/VC Team” page should allow the project creator to log whether the organization he/she belongs to, or a third party vendor, is in charge of implementing BIM/VC features. A project ID must be specified on this page, or an error should be returned.

Non-Functional Requirements

The entire site should be usable from mobile browsers (specifically Chrome on Android 4.4 and Safari on iOS 7). The site will be hosted using an Apache web server with PHP 5.5. The server will host a Git repository that we can push to and pull from to publish the application. jQuery 2.0.2 is the JavaScript framework that will be used. All JavaScript code must use the jQuery object, as there may be future conflicts with the “$” alias. AJAX requests should be used to retrieve all content on the site.

Every page that requires a project ID should use existing authentication functions to confirm the user has appropriate permissions to view or edit the page. These authentication functions will continually validate logins, check for session timeout, and prevent multiple logins.

System Architecture

Project Track runs on a standard LAMP[2] stack paired with jQuery. It relies heavily on AJAX to load pages, forms, and users. Project Track is completely dynamic, except for main.php, which acts as a template file for PHP files to be loaded in to. Most PHP files initially loaded into main.php are simple containers that setup empty divs. The meat of the content generation and display comes from JavaScript and AJAX responses.

The full path of the URL defines the page to load, in addition to the selected project and any other variable: http://server.com/main.php#script.php/project=1/var2=two

Main.php can be cached, so, on a link click, the only network action is loading the script after the hash. Many pages require a project to be specified, which, along with any other variables, is included in the URL. Additionally, main.php loads a JavaScript file of the same name (i.e. “script.js”) and runs its “start” function[3]. If the page has specific AJAX requests, they are handled in a companion AJAX page with the same prefix, script_ajax.php.

Because form loading and handling is so central to Project Track, most form requests are served by a single PHP script. At around 1,500 lines, project_detail_ajax.php is responsible for filling out forms, determining who can edit them, and processing new users.

Figure 1 shows how a final display page is built for the user. The server starts with a container php and a loaded JS file, which may then load other JS files. By AJAX request, handler PHP scripts pull from the database and iteratively spawn and fill the relevant fields. We then use jQuery to modify the DOM with the new elements, the browser styles them with CSS, and the user sees their requested content.

[2] Linux, Apache, MySQL, and PHP – A tried, true, and trite tech stack
[3] See Appendix A for further information on JavaScript file formatting.

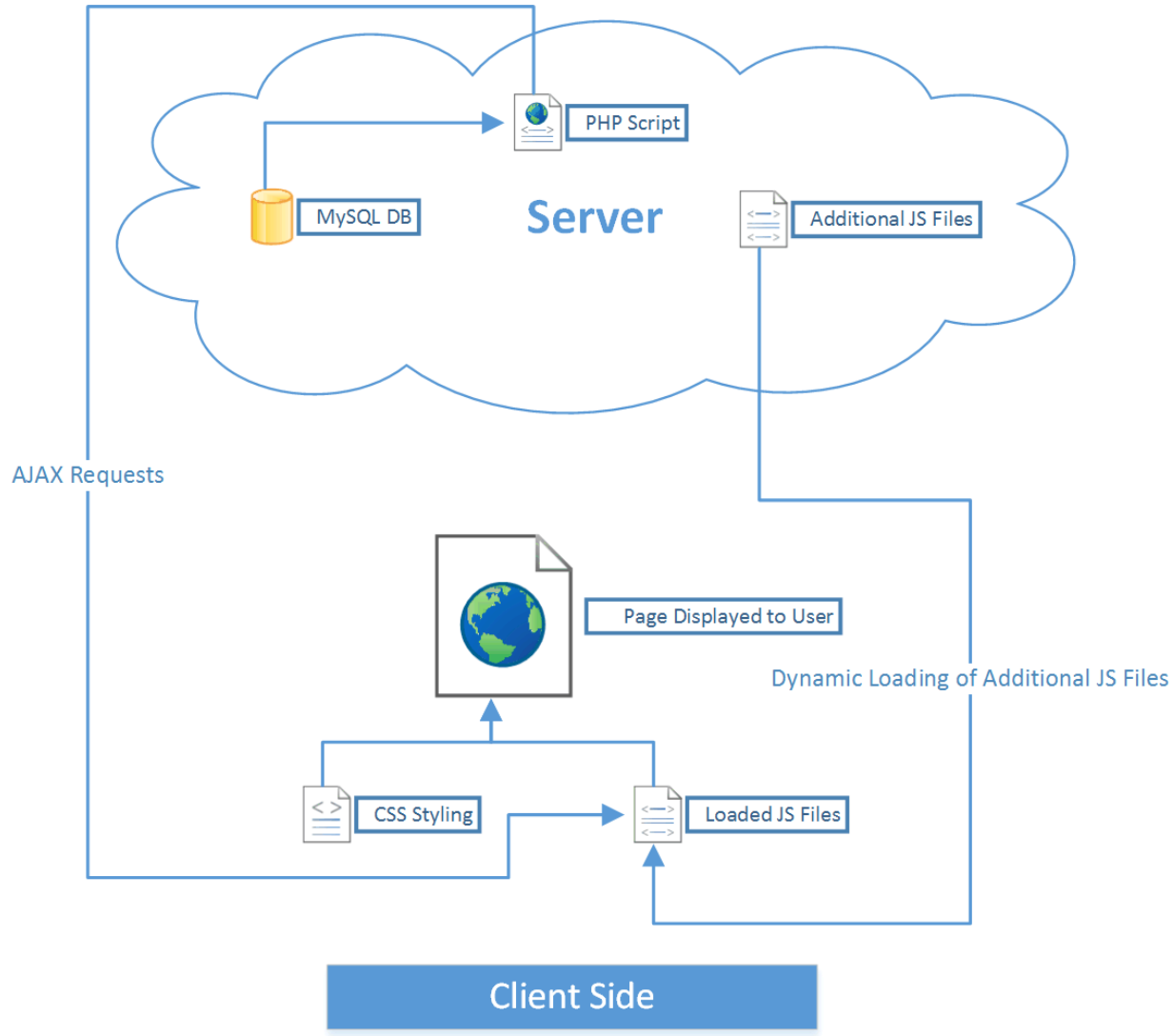


Figure 1: Graphical Representation of our Site Architecture

Technical Design

Creating new users is perhaps the most crucial subsystem of any project. There are an abundance of well established methods to do such a task, however, Project Track needs to be closed and secure. Existing users should be the only ones with the ability to create or invite new users, but they should be able to set or know any other users' passwords. Therefore, authorized users will provide the email address of a new user, along with some business information, which is sent server side via AJAX. We insert a new user row, set the email to be the username, store the extra data, and place 'NULL'[4] into the password column. Next, we generate and email a token

[4] An actual string literal, 'NULL'

to the provided email address, and include a link with the token for their user ID. The email has a link containing a GET variable that, when clicked, will allow them to provide a password and finalize their creation.

The tokens are the string hex digit representations of a SHA256 hash output. This method provides both a uniform length token, and a high standard of security. We did not feel the need to use HMAC, or more expensive hashing, because we generate tokens with a twice hashed seed. The first hash is the concatenation of the users email behind the unix timestamp in seconds; the second hash uses the unix timestamp in microseconds concatenated with the first hash, which is concatenated again with the unix timestamp in seconds. The string output of the second hash is the users token.

```
$token = hash("sha256", microtime().hash("sha256",time().$email).time());
```

e.g. `$token='64bc9f495190ca43dacf9c22492416f2d36e47a5babdb1fe8cf5f0643bd08475';`

After a new user sets their password, they are automatically logged in, and redirected to a list of projects on which they are assigned. As this is their first time logging in, the user should only see the project which created their user. The user may then update the parts they are responsible for, though many user roles are unable to modify projects -- these "restricted" user roles exist so that all team members are kept in the loop. Nonetheless, all users assigned to a project may always view the details of the project.

The core of the front end is displaying and modifying projects; one jQuery accordion presents selected projects and handles any edits. This centralization is easy to navigate, with drop down panes, and is persistent across multi pane usage. It performs AJAX requests as necessary to stay up to date and incrementally change the project. Our system allows a single loading of the form structure, with on the fly autocomplete and modification.

Project Track is driven by flexibility, on both the frontend and the backend systems. In order to keep our code lightweight and extensible, we avoided hard coding HTML elements and layouts, as much as possible. Instead, we often query the database to pick which forms and fields to display, along with their names. Case in point, the generation of the "Project Team" pane: the server runs a query to return all position fields that are displayable, and iterates over the response to echo labels and input elements with the correct IDs. The drop down menus, too, are built from the corresponding enumerations in the database. For example, the district drop down:

```
SHOW COLUMNS FROM `table_name` WHERE field LIKE 'district';
```
[5]

Since our forms are so fluid, we never have to look at code to rework them. To create a new team member position, we simply insert a new row in the database. The entire system would

[5] See Appendix C: Response of Enumeration Query

therefore be able to display, insert, and update team members with that new position. A back end this simple and dynamic is easy to debug, and allows anyone with *phpMyAdmin* access to adapt forms without knowledge of HTML, or needing to know how the PHP layer processes queries. As a result, Project Track is easy to maintain and has longevity.

Project Track also utilizes a table named “project_state” where the current status of each project is stored. This table is responsible for enforcing a specific workflow set forth by BIMShift. Project workflows can be broken down into two categories: new projects, and awarded projects. New projects contain basic directory data, such as its name, description, and cost. These projects may also have an active team and high level BIM/VC options entered. The “New Project” workflow is visually represented in **Figure 2**.

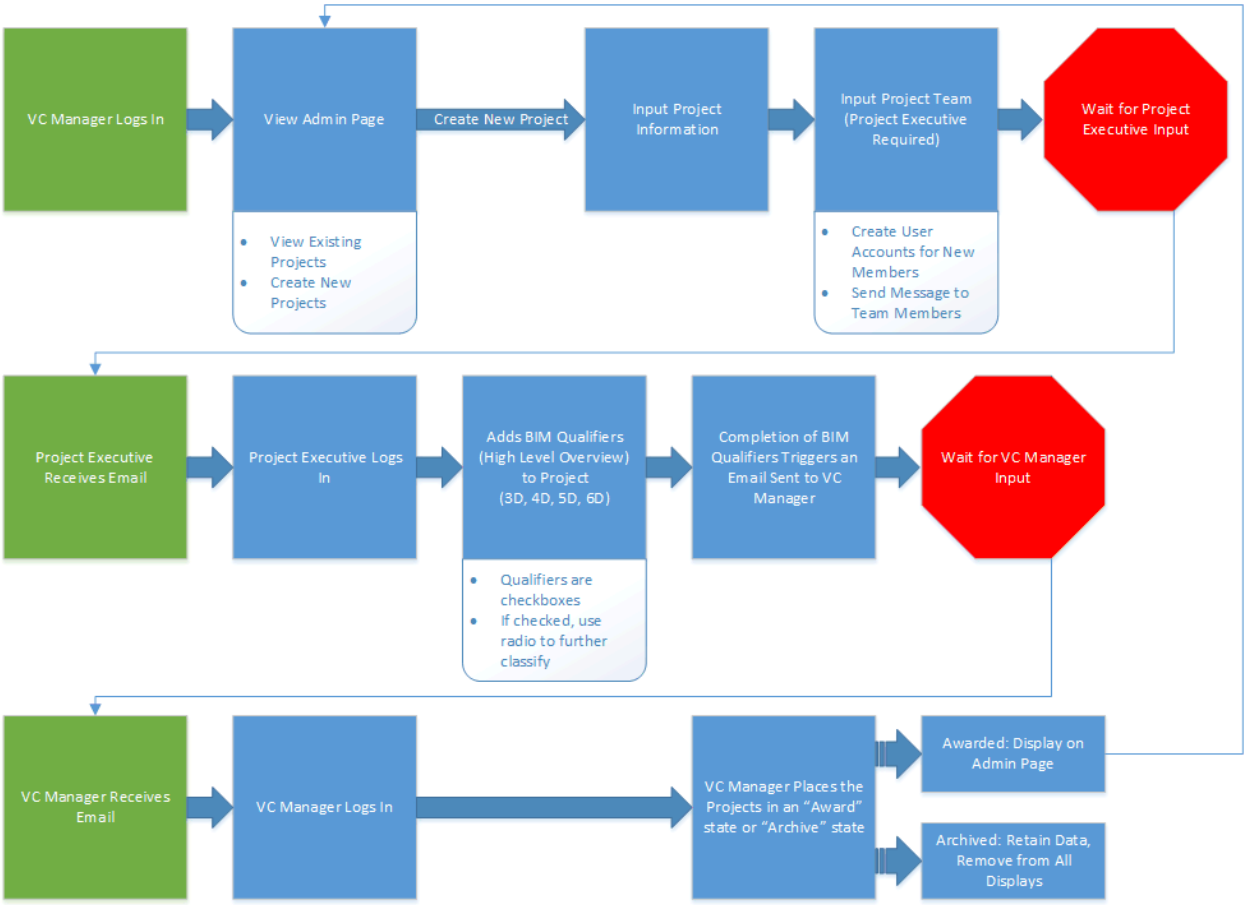


Figure 2: Workflow for Creating a Project

After BIM/VC options are entered, the project creator (VC Manager) inputs whether the project was awarded to the company. If the project was lost, there is no need to continue documenting it, and so all the data is archived. If the project is awarded, team members are able

to continue viewing project data, as well as “unlocking” new forms as the project state advances. Refer to **Figure 3** to see this workflow.

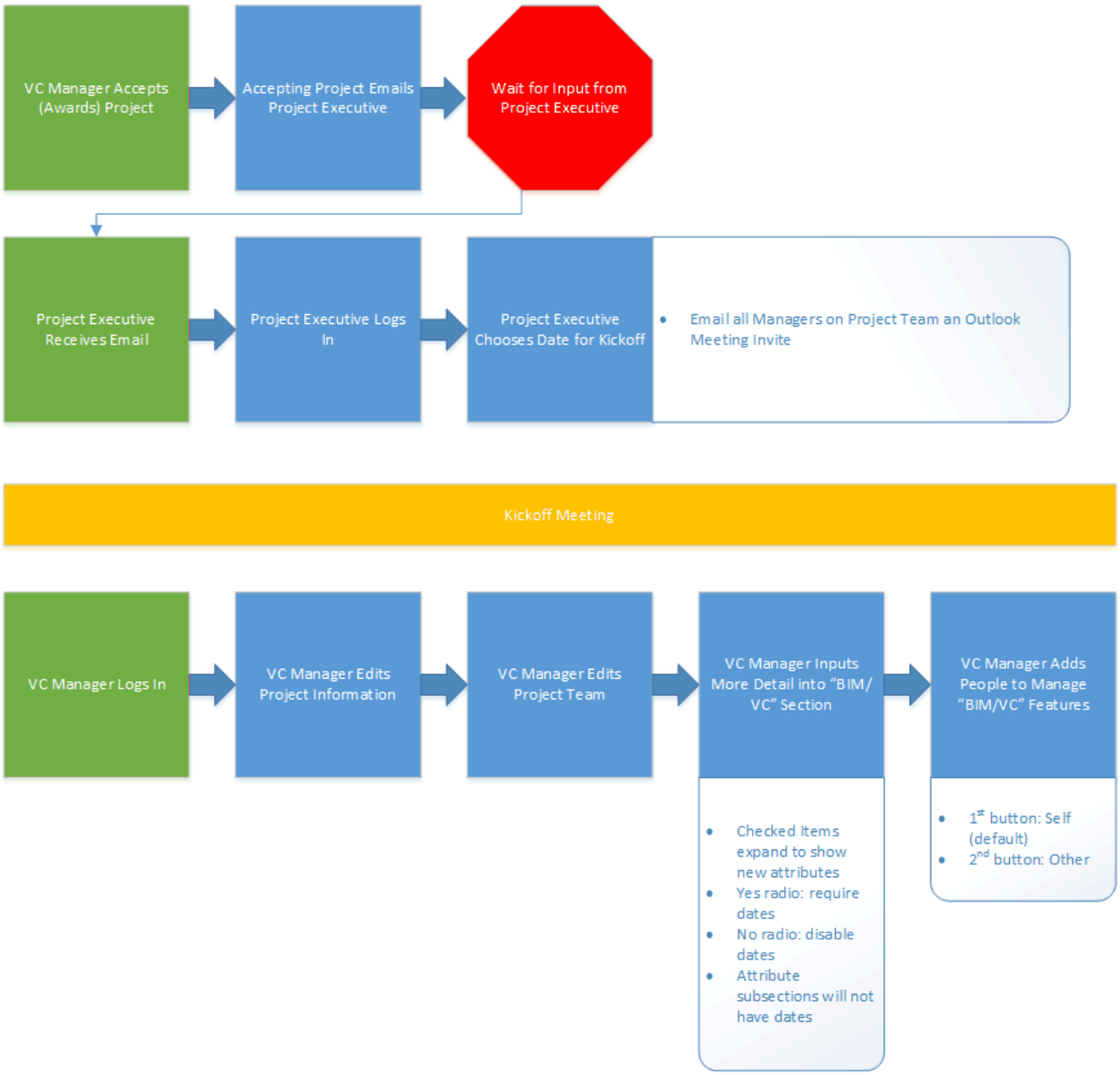


Figure 3: Extended Workflow for Awarded Projects

Design and Implementation Decisions

Our team already had a toolset defined by BIMShift, which included HTML, JavaScript, AJAX (via jQuery), PHP and a MySQL database. Along with language and platform decisions,

BIMShift also specified a generic structure of our files (e.g. checking if a user is logged in, scoping JavaScript). We made several other choices, as a team, to improve the consistency and maintainability of our code base.

Project Track relies heavily on form data. One early issue was determining if the form fields needed to be updated or inserted into the database table. While the problem could be resolved using an excess of server logic, we opted to associate every form control with a hidden form element. The hidden element had an identical HTML ID (with _dbid appended to the end). On the server, we then only needed to check whether the ID was valid. If it was, the script would update that ID. If it was not, the script would insert data into a new ID

.

BIMShift included the jQuery framework for our use in developing Project Track. We tried to use jQuery's API as much as possible to avoid writing our own functions that accomplished the same task. jQuery includes a function called `serializeArray()`, which creates a JSON object containing the values of all the form controls on the page. We initially utilized this function to send the form data from the client's computer to the server. Unfortunately, serializeArray() did not grab hidden form fields, or disabled form fields, and broke our workflow when we added hidden IDs to each element. To fix the problem, we wrote a function called `serializeArray(string form_name)` that serialized all form controls on the page. To make it more flexible for this project, we also allowed for a way to specify a specific form to serialize, in the event that we had multiple forms on one page. Our serializeArray function produced code in the same format as its jQuery counterpart, reducing the amount of refactoring needed when we updated method calls.

Project Track uses AJAX requests to send data between the browser and the server. To reduce the load on the server, we wanted to implement client-side validation, so data was only sent to the server when we believed it to be correct. We wrote custom JavaScript validation functions instead of using a framework. There didn't seem to be any dominant jQuery plugin, and by writing our own tests we avoided using licenses that conflicted with BIMShift.

Our application dynamically generated a large amount of HTML tags that are populated with data from the database. To accomplish this task, we had two options. We could either use JavaScript to create new DOM elements, position them in the correct location on the page, and use JavaScript again to populate these new elements, or use PHP to generate the HTML with the data already embedded, and simply replace the page with the PHP output. We chose to use PHP to generate the HTML, since the server scripts were also responsible for inserting and updating data in the database. Changes we made to the queries often required changes to be made in the HTML as well. Having all of the code in one file simplified communication needed in our source code.

Lastly, we had a number of different pages that all contained forms. While the forms collected different data, the process for submitting, validating, and responding to a form was very similar. Adhering to DRY (Do not Repeat Yourself), we created a shared JavaScript file to be used in all the form pages that acted as an interface for form submission. The other side of this decision was a mandatory set of functions that had to be defined in the individual form scripts (much like Java interfaces). An added benefit of this change was aligning all of our form scripts with common function names that offered well-documented pre and post conditions.

Results

The Project Track web application is currently visible from the URL https://pt.bimshift.com. The application is both functional, and fulfills the obligatory items set forth in the Requirements section of this document. Notably, the site offers an interface for users to access their existing projects, and a method for users to create new projects. Projects are controlled by a set of business rules and workflow handed down to us from BIMShift, and use a roles based system to verify only the intended user is able to edit a certain aspect of the project.

Specifically, the person who creates the project is placed on the project team as the Virtual Construction (VC) Manager. The VC Manager is responsible for adding users to the team, as well as interacting with the Project Executive (PE) to advance the project state. The project team has autocomplete functionality, so the VC Manager can easily select existing users. New names prompt the VC Manager to input additional information, and creates a user profile. After adding team members, Project Track sends an email notification to all the team members, alerting them of their addition to the team. Any new users are also prompted with a link to setup a password for their account.

The PE is then responsible for inputting high level BIM/VC features into the project. After this data is entered, the VC Manager has the power to award or archive the project. Archived projects are not deleted, but instead are hidden from the front-end user interface. If the VC Manager awards the project, the BIM/VC page expands to show more options, and allow engineers on the project team to be assigned to BIM/VC attributes.

Finally, an awarded project also gives the project team access to a BIM/VC Team page where the VC Manager can declare whether their organization is handling BIM/VC features, or whether a third-party contractor is in charge.

Project Track operates independently of other web services, and includes, in their entirety, any external frameworks (like jQuery). These factors made development of the application very easy, as we were operating in a virtually isolated environment interacting with only the tools we needed.

Appendix

Appendix A: JavaScript Scoping Style

As was defined by our client, how we handled scoping for the JavaScript files was to implement a global object with the same name as the file, and then to create sub functions underneath that object to avoid any namespace type issues. Then, when these files are loaded, the “start” method of the file is called. The “start” method contained code that needed to be executed each time a user visited a page. For example, if we need to make an AJAX request in order to get the form for a given page, we would call a function that completes that AJAX request from the “start” method. An example of how this works follows:

```
var test_script = {};

(function() {

	testScript.displayForm = function() {
		jQuery.ajax({
			url:"test_script_ajax.php",
			data:{
				type:’someType’,
				name: ’someName’
				jsVars: JSON.stringify(scriptVars),
				load_source: 'AC'
			},
			dataType: 'json',
			type: 'POST',
			cache: false
		 }).done(function(data){
			// Do something here.
		 });
	};

	test_script.start = function() {
		test_script.displayForm();
	};

})();
```

Appendix B: Dynamic HTML Generation Using SQL Enumerations

Project Track uses table columns to dynamically generate some HTML form elements. One example in our code is generating a drop down of geographic districts. Here, we would execute a SQL statement such as:

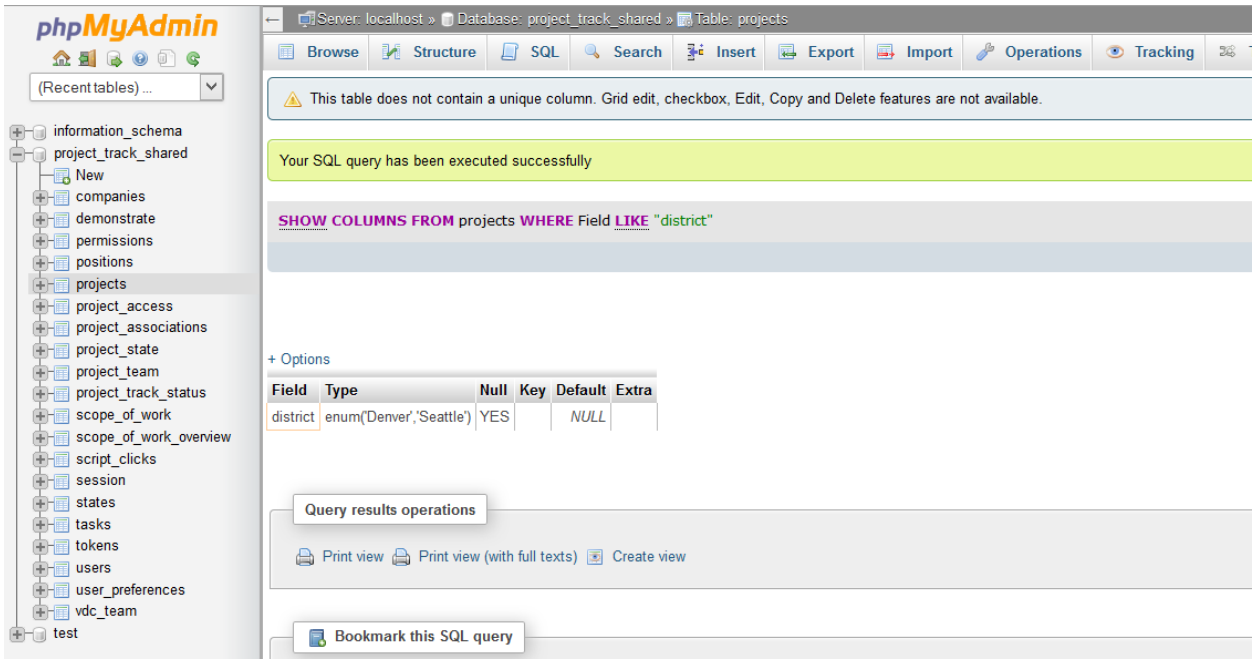
```
SHOW COLUMNS FROM `table_name` WHERE field LIKE ‘district’;
```

After fetching the result of the previous statement, we then extract the enumerated options from the “Type” column. This is handled in PHP, and can be accomplished using code similar to the following:

```
$enumList = explode(',',
   str_replace("'", '',
```

```
    substr($EnumResponseContract['Type'], 5, (strlen($EnumResponseContract['Type'])-6))
    )
);
```

Appendix C: Response of Enumeration Query



Appendix D: Database Schema