2017

# Asset Management Software

Tony Phillips
*Grand Valley State University*

# Asset Management Software

By
Tony L Phillips
April, 2017

# Asset Management Software

By
Tony L Phillips

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University

April, 2017

_____

**Dr Jonathan Engelsma** **Date**

## Table of Contents

# Abstract

A small company was in search for a software that keeps track of the cost of assets through its useful life. This in turn would be used to determine when an asset has reached its economical useful life. They searched for an application that would fit their needs. Through a recommendation, they implemented a big commercial application that is in use in a variety of manufacturing and utility companies. Although this software accomplished their needs, it was not a great match. The goal of this project is to create a slim down version of an asset management software that performs better, more user friendly and is accessible through mobile devices. The user base is not frequent computer users so working with them in an Agile style development was important to obtain that user friendliness.


# Introduction

In 2006, a large farm decided that they wanted to implement a system that would keep track of maintenance of its equipment. The company wanted to know where they are spending their time and money regarding equipment maintenance and have a tool that can help management decide when an asset has reached the end of its economic life. In other words, when would it be better to just buy a new equipment instead of fixing the old one. The farm also wanted to keep track of parts that they have. Too often they would go out and by a new part without even looking to see if they had one in stock. They had parts but no one knew where to look. Worst yet, they had parts to equipment that they no longer had. The company also decided that the system they use must also implement an inventory system.

They looked for suitable systems but none seemed to match their requirements. One of the managers who came from an automotive manufacturing background suggested using Maximo. It met all the requirements they were looking for and decided to go with that.

# Background and Related Work

Maximo [2] is an enterprise asset management software. At the time, they implemented Maximo 6.2, a MRO Software product that was shortly thereafter acquired by IBM. IBM continues the development of Maximo with its latest version 7.6.

Maximo is an intranet web based solution with several options to deploy. They choose to use SQL for the database and WebSphere application server to host the Java enterprise solution. At the time of implementation, the company did not have a server and their computer network was setup as a workgroup. They took this opportunity to buy a server, install Windows Small Business Server 2003 (the latest version at that time) and establish a proper network with security.

They communicated with several companies trying to figure out how many licenses are needed and what hardware was required to support it. A typical Maximo installation requires three separate machines: one to host the SQL server, another for the application core and a third for the reporting module. In discussion

with the consulting company, they realized that it is possible to combine the application core and the SQL server on one machine. They opted to go this route because they were buying 2 concurrent licenses where the standard minimum was 5. They felt this would not hinder the performance in any noticeable way. The choice to keep the reporting part of the application on a separate machine was because it takes a lot of resources to run.

Maximo is used all over the world. Doug Wood [3], an IBM Maximo Architect, has stated the following facts.

- 11 of the 12 largest PHARMACEUTICALS companies in the world are using Maximo
- 7 of the 10 largest AUTOMOTIVE companies in the world are using Maximo
- 8 of the 10 largest OIL & GAS companies in the world are using Maximo
- 11 of the 20 largest diversified UTILTY companies are using Maximo
- 11 of the 12 major AEROSPACE & DEFENSE companies in the world are using Maximo
- 9 of the 15 busiest AIRPORTS in the world are using Maximo
- 6 of the 10 largest ENERGY companies in the world are using Maximo

There is no dispute that it is good at what it does. It does a great job at managing assets, inventory and services performed on those assets to highlight just a few of the key features.

After Maximo was installed, the company hired on another person who also was from an automotive manufacturing background and very familiar with Maximo to help setup and implement the system at the company. This took several months, as every asset and location had to be accounted for and given a unique ID based on grouping of similar assets/locations.

It didn't take long before the company realized that Maximo was more than they needed. They continued to use it because they are already fully invested in it. They have adapted to use it to meet their needs. Some features they bypass altogether and others they use just a portion. One example is the use of lot numbers when purchasing items. The system would keep track of how much inventory each lot had and if there was an expiration date. This feature is helpful when certain lots are recalled. The item and lot can be traced back to where it was used and therefore corrected. But due to the size and nature of the company and items bought, this just added an extra step and with the users not having a good understanding of the process, it only leads to more issues.

Another issue was reports. The reports for Maximo were very slow, taking five minutes to print a simple report like a work order. Some reports would time out. This was unacceptable. As a work around, reports were created using Microsoft Access due to its quick development of forms and reporting features.

As time has gone by, technologies have changed and browsers have been upgraded. Maximo was designed for Internet Explorer 6 and as users started upgrading their browsers they found out that a good portion of the menus no longer worked. Shortcuts were created to navigate around the system but there were some features that are only accessible through the menus. This was corrected with using a virtual machine with Windows XP installed.

# Program Requirements

The first step was to sit down with the users and management and find out what features they use of the current system. The main goal the company wants to do is keep track of costs related to maintain and service assets and an inventory system. The components of those costs are materials, labor and external services. Work orders are the driving force of the system. Work orders are created two ways, through requested services from users and through scheduled Preventative Maintenance (PM) is manually triggered. Work orders are then assigned to a service tech and they are responsible for seeing the work through. This includes sending the asset out for servicing to a third party or performing the service internally. Laborers time is put to the work order and with the laborer's rate. Materials and third party services are run through purchase orders. These costs are not directly related to work orders until they have been received.

A list of items is to be maintained. These are materials that are used regularly. This allows easy tracking of frequency of purchase and use. A good example of items are PMs that require the same materials every time. When a PM for an oil change comes up, users know that they need a certain filter and oil to perform this work order. This lets the user know what parts to use each time.

They also want to keep a stock of items on hand so that their downtime on assets are minimum. An inventory system is required to keep track of quantity and costs as well as other key bits of information for reordering. A user can run reorders and the system will check its inventory balances, pending work orders that require items and pending purchase orders that have not receive all items. If the calculated balance is at or below the reorder point, it will generate purchase orders automatically.

Users can log into the system and have access only to selected features given to them by administrators and supervisors. Each user logs in with their email and password. Notifications are sent to users via email.

Locations are used to help organize assets. It tells where assets are located. The company also uses locations for buildings instead of assets. Although buildings can be classified as an asset and location they opted in listing them as locations only for simplification. Locations as well as assets and work orders have a hierarchal order to them. This allows a drill down of these things. An example would be a business has a property and that property has several buildings, each building may have several floors, each floor may have several rooms, each room may have several cabinets.

Having access to the systems beyond the confines of the local network was needed. They often run into situation where an employee is not in front of a computer yet needs a purchase order number in a timely manner.

Reports are the key to the application. Management needs to be able to run reports to see what needs to be done and what has been done. Management also needs a detailed report on assets to help decide when an asset has reached the end of its economic life. Reports also must be flexible to match users' needs. The system also needs to be able to print work orders and purchase orders as well as the ability to email POs to vendors.

# Implementation

The first decision was to host this application offsite. This did not take much thought because of the need to be able to access this on mobile devices anywhere. The internet at the business is delivered via Wifi. Although it is a good solution for internet, it does have its limitation. The company gets up to 6Mbps down load and up to 1.25Mbps based on tests performed [24]. Trying to host a website from this location would hinder its performance.

Next was how am I going to implement it. I know I did not want to write it from scratch so I started looking at frameworks. I have only used one framework before which was Ruby on Rails which was also my only exposure to Ruby. I am familiar with PHP as I have been using it for several years creating basic websites. Due to the size of the application I did not want to spend time learning a new language and a framework so I looked at PHP frameworks. In researching them, Laravel was stated to be the most popular [23] with good documentation and a good community base despite being the newest framework. Other frameworks that I was considering were CodeIgniter, Zend, CakePhp and Symfony. Laravel is quickly evolving. When I was researching it, Laravel 5.2 [5] was the current version and of this writing, only a few months later, 5.4 has been released.

Initially I tried to setup my development environment using Vagrant [6], VirtualBox [15] and Homestead [1, 14]. Vagrant helps create development environment quickly and easily using VirtualBox or VMware. Homestead is a Vagrant environment that is already setup with a Laravel environment. After several attempts of using Vagrant I decided to setup the development environment from scratch. I installed Ubuntu 16.04 on a VirtualBox followed by the rest of LAMP. Next I installed Composer [13], a PHP dependency manager. Composer installs the Laravel installer with just one command line. One more command line and the beginning of a Laravel project is started.

One decision I had to make was to use PHP 5.6 or 7. The biggest advantage of using PHP 7 was the performance. Per Zend.com Laravel performs 70.2% more requests per second [12]. The down side is that they are releasing updates to fix bugs on a regular basis. Another issue to consider was hosting options.

Who can host Laravel and what version of PHP do they use. I chose to use a virtual server where I have full control of every aspect of the server. For performance reasons, I chose to use PHP 7 for my application.

Laravel has user and authorization features built in but I chose to use Sentinel by Cartalyst [7] for a few additional features it has. Since the application will have several different types of users, I used the users and roles features to limit who has access to certain parts. It takes care of all the user registration, authentication and authorization. Sentinel also allows multiple sessions. This is helpful when a user goes in and out of the office where they use a computer and mobile device without having to login each time. Sentinel was then installed using Composer.

There are several options for data storage. I used MySQL for several reasons. First, because it can handle large amounts of data. The current system has over 180,000 work orders to date plus all the supporting data that goes with it. I need a system that supports high volumes of write operations. Secondly, I didn't need all the advance features of PostgreSQL or SQL Server. I am familiar with MySQL and SQL Server and both are capable for this application but MySQL is free and easier to install on a linux system.

Interaction with data in Laravel is handled with Eloquent ORM (Object Relational Mapper) [8]. Eloquent models are created to interact directly with the data and can be used for all CRUD operations. Models can have functions and special getters and setters that perform a function every time a property is set or get. This is extremely helpful when dealing with date and time, translating it between human form and computer form. Relationships with other models are created and accessible within the model. This simplifies passing related data to the view. Instead of creating complex queries or multiple queries and passing those as variables you can get the instance of the model and all related models. Eloquent also supports SQL in part and in whole in case you need to create more complex queries. I had to use some SQL for calculating the various inventory balances. Transactions are used on some actions that require multiple writes and I needed all or none to be saved.

The application has many supporting components where users can add, edit and view these components. These include things like general ledger accounts, unit of measures and tax rates just to name a few. In general, deleting of records are not available to keep historical data. Instead users can change the active state or status of that record to hide it from most views.

Middleware has been implemented to make sure that selected views require login or it will redirect them to a login screen. Another form of security is to check if the user is assigned the correct role to view or edit the requested view. For each view, there are selected roles that can view it, roles that can add or edit and then there might be other roles for things like who can change the status of work orders.

For the User Interface, I am using jQuery [9], jQuery UI and Bootstrap [10]. I have some experience with these and see the benefit they perform. One of the features of jQuery UI that I am using for the first time is Autocomplete [11]. It is very helpful for the user. As they type it will give suggestions based on what information should be in that field. It is setup to start after three characters have been entered by the user. Then an ajax call is performed to retrieve matching data and then render the data with custom format. The custom format includes the code and name or description of the data so the user can make the correct choice.

Another UI function I used was a search function. I used Bootstrap's modal as the basis of the popup. I have one modal for all the search fields and just reuse it. For each of the searches the modal is customized with a title, size and filled in with a view that was retrieved by an ajax call. The user enters a search term and clicks search. Then another ajax call is performed. It looks for a match in the code or name/description and populates a list for the user to select from. Once selected, the value will be inserted in to the corresponding field.

The last UI function to mention is flatpickr [17]. The application has several fields with dates, date & time and time. Flatpickr is very easy to use and customize. It didn't take long to get this in use. A highlight for this add-in is that it uses mobile device's default date and time picker. On computers, selecting the time and date is very easy as it uses scroll to help pick the correct month, year and time quickly.

Throughout the project, I used Git and GitHub [16]. GitHub was very valuable in the process as I could put in all the features and requests as issues. From there I could keep track of the progress of each issue. I used the ZenHub [18]plugin for Chrome Browser to further assist in the agile development. I used the Milestones feature as my sprints. I gave each issue an estimated story point which can be translated into time to complete and assigned it a sprint. As time went by and I completed issues, I use the burndown report. This is very helpful in seeing how the sprint is doing in comparison to story points completed vs time. This GitHub project was shared with my professor so we both could see how I was doing on each of the sprints. This also helped me to better judge how much time it would take to complete issues.

For deployment to the web for testing and production, I used Forge [4] which took care of setting up the server for me. Forge is a service provided by Laravel.com. It takes care of installing and setting up Nginx, PHP, MySQL and more. It works on several cloud providers. I chose to use Digital Ocean [19] and they have choices of server hardware configurations. To start out I opted for the smallest server and I can always upgrade as needed. Once Forge had setup the server, it took only a few minutes to pull the project from GitHub and configure a few files for production environment.

After the initial deploy and each update, I had users review it and give me feedback on what improvements or changes are needed. Their input helped keep the application focused on what was important to them.

My sprints were setup for three week intervals but I pushed updates more frequently so I could get feedback from users quicker.

Initially when users register for an account for this application, I used Mailgun [20] to send the user an authentication email to verify their email is valid. For testing purposes, I have bypassed authentication only to make it easier for users to log in and test it. Each time I update it I rollback all the migrations and rerun them followed by seeding the database with some sample data for them to use.

The cost to host the application has been very reasonable. My implementation would average less than $24 a month. This includes a domain name from Godaddy [21] at $19.99 per year, virtual server from DigitalOcean at $6.00 per month or $72.00 per year, setup and administer server with Forge at $140.00 per year and email service from mailgun.com at no cost. Each service has varying cost based on what you select. I for see upgrading the server to suite the performance needs of the actual production deployment. Mailgun has a tiered price plan and I believe the usage would stay under 10,000 emails per month which is free. Over that the cost is $0.0005 per email up to 500,000 where the next price tier starts.

# Results, Evaluation, and Reflection

Initially I was planning on having the basic features completed and deployed for actual use. I feel the project was larger then what time allowed for a capstone project. I have most of the features done and ready for production. There was more of a learning curve in learning the new framework than expected which slowed me down in the beginning. Then later in the project I ran into issues with javascript/jQuery. Most notable was the autocomplete feature with custom display and trying to get it to display results within a modal. I later found out that it was working but the modal was displayed on top of the results [22].

I feel the usability of the application is very high. It keeps the core functions of the existing application and removes the excess functions what were not used. Therefore, users can navigate the forms quicker and easier. I am very happy with the data input methods which add to the efficiency of the forms. The most notable ones are the autocomplete and date/time selection. The current application has a search function which I also implemented in this application but the autocomplete will speed up data entry.

Over all I am pleased with the work done so far. It was a fun challenge despite having to struggle through a few things. I learned a lot and look forward to continuing my learning as I complete and beyond.

# Conclusions and Future Work

The foundation of the application is completed but before it can be used in production it needs some basic reports. Besides individual printing of purchase orders and work orders it will also need a list of current work orders and a report that shows the detail of individual assets. This detail report will include total

expense over the life of the asset and what work has been done to it. This will give management the tool it needs to base decisions on.

Before I deploy for production use I will be considering all aspects of security. One that I will be doing is adding a SSL certificate that can be obtained free through Forge. Because the application will have some important data, I want to keep it safe. At that time, I will re-enable the email verification.

The next feature to be added is the ability to create invoices from purchase orders. With a few simple clicks and a couple key strokes, an invoice can be created. This feature will be handy as the next feature is added. But the invoice entry screen is another way to enter expenses for work orders at point of sale. This will alleviate users from having to run everything through the system saving time and money.

After that an invoice export function will be added. This will allow data to be entered once instead of once here and again into the company's financial software. This must be flexible as the other software will be changed or replaced in the future.

# Bibliography

Credit others for their work. Cite location of all tools used.


1. Stauffer, Matt. Laravel Up & Running, Sebastopol, O'Reilly, 2017
2. Maximo – An IBM Enterprise Asset Management Solution
   https://www.ibm.com/us-en/marketplace/maximo
3. Doug Wood, (11/7/2014). BIM for Operations, p.5.
   https://www.ibm.com/developerworks/community/files/basic/anonymous/api/library/75dbdf46-1a08-429c-9742-bd340d7d1fd3/document/c0c764b3-f861-47f6-8878-fae48d99c5e9/media
   doug.wood@us.ibm.com
4. Forge – Deploy and manage PHP servers
   https://forge.laravel.com
5. Laravel – PHP Framework
   https://laravel.com
6. Vagrant by HashiCorp
   https://www.vagrantup.com
7. Sentinel– Authorization and Authentication Package by Cartalyst
   https://cartalyst.com/manual/sentinel/2.0
8. Eloquent – A simple Active Record implementation
   https://laravel.com/docs/5.3/eloquent
9. jQuery – Javascript library
   https://jquery.com
10. Bootstrap – Responsive Frontend Framework
    http://getbootstrap.com
11. jQuery UI Autocomplete
    https://jqueryui.com/autocomplete/#custom-data
12. Zend.com – Statistics on PHP 7
    http://www.zend.com/en/resources/php7_infographic
13. Composer – Dependency Manager for PHP
    https://getcomposer.org
14. Homestead
    https://laravel.com/docs/5.3/homestead
15. VirtualBox – Oracle Virtual Machine
    https://www.virtualbox.org
16. GitHub – Web based version control repository
    https://github.com
17. Flatpickr – Date & time picker
    https://github.com/chmln/flatpickr

https://chmln.github.io/flatpickr

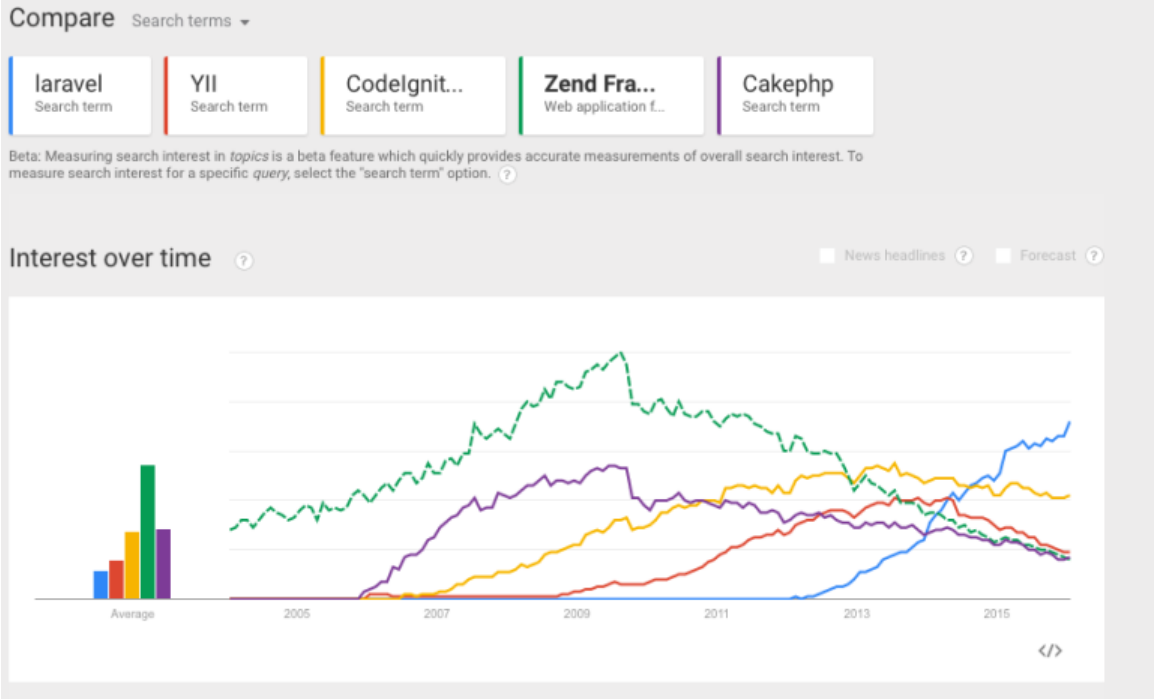
18. ZenHub – Agile GitHub Project Management
    https://www.zenhub.com
19. DigitalOcean – Cloud Computing Platform
    https://www.digitalocean.com
20. Mailgun – Email Service for Developers
    https://www.mailgun.com
21. GoDaddy.com – Domain Name Registrar
    https://www.godaddy.com
22. http://stackoverflow.com/questions/3217134/jquery-ui-autocomplete-inside-a-modal-ui-dialog-suggestions-not-showing
23. Monus, Anna, 10 PHP Frameworks For Developers – Best of
    http://www.hongkiat.com/blog/best-php-frameworks/
24. Speakeasy Speed Test
    https://www.speakeasy.net/speedtest

Other notable mentions

www.laracast.com – Forums and videos for Laravel

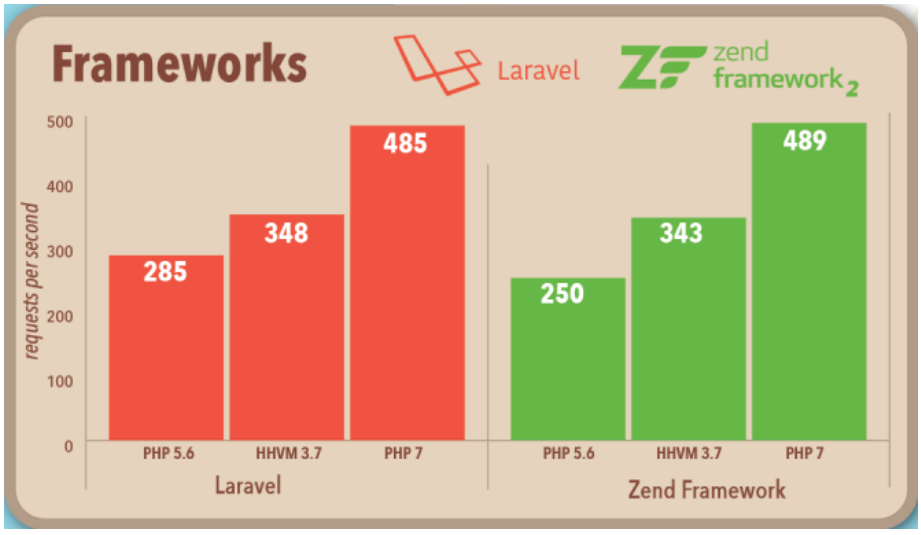
# Appendices

PHP Frameworks Interest Over Time



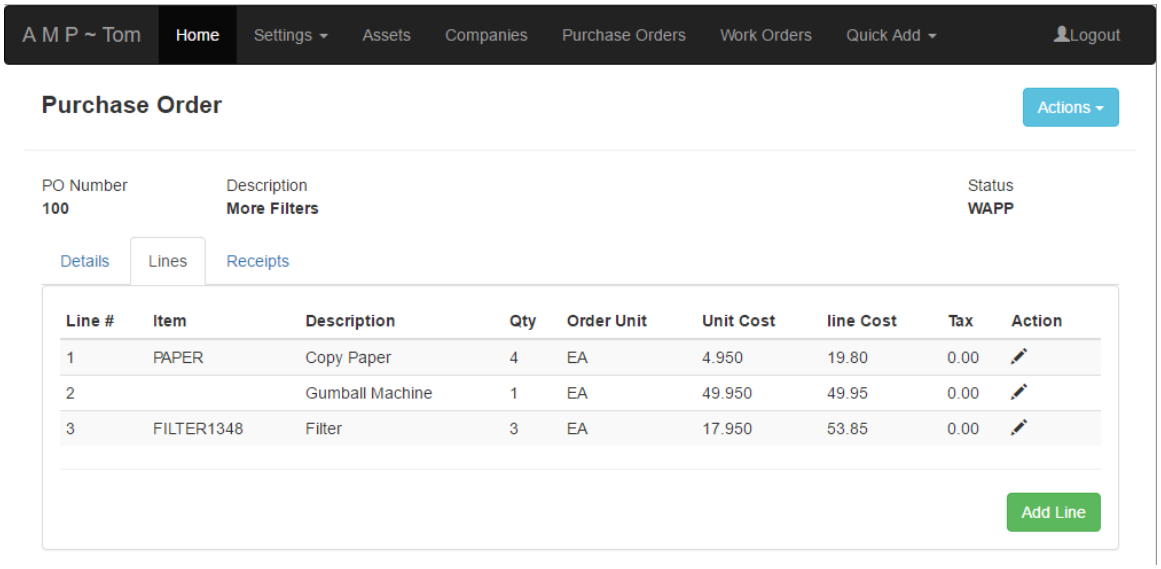

Source: https://www.linkedin.com/pulse/why-laravel-best-php-framework-2016-ramoliya-hitesh

Zend.com – Request per Second

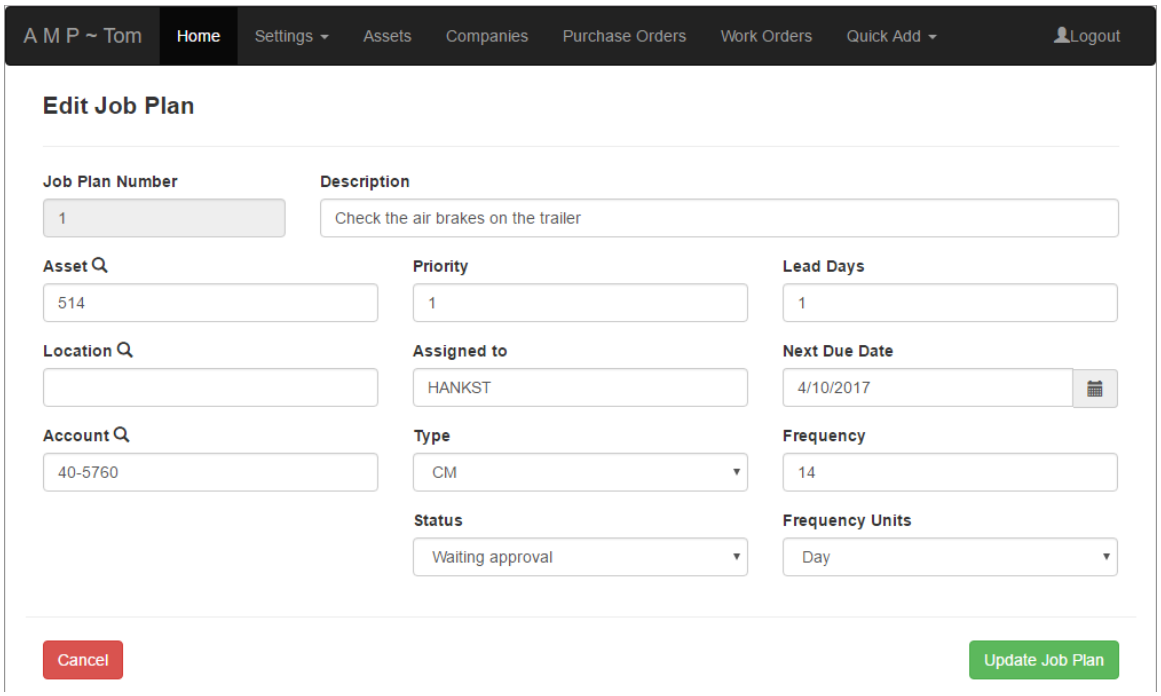



Source: http://www.zend.com/en/resources/php7_infographic

Screen shot – Purchase Order




Screen shot – Job Plan Edit



Maximo screen shot – Work order