Christian McDonald

Project Documentation - Kronos MM



# Executive Summary

Let us begin with defining the primary reason for this proposed project. Your company is growing and expanding its materials warehouse infrastructure. With this expansion a new system to maintain inventory is needed. Many of your current tasks in the warehouse are currently being done manually on pen and paper. This has led to errors on the backend for the financial departments and inventory level for the sales department. Now let us take a look at how this proposal can solve the aforementioned issues.

Our system Kronos MM has many capabilities to assist in all aspects of materials management.

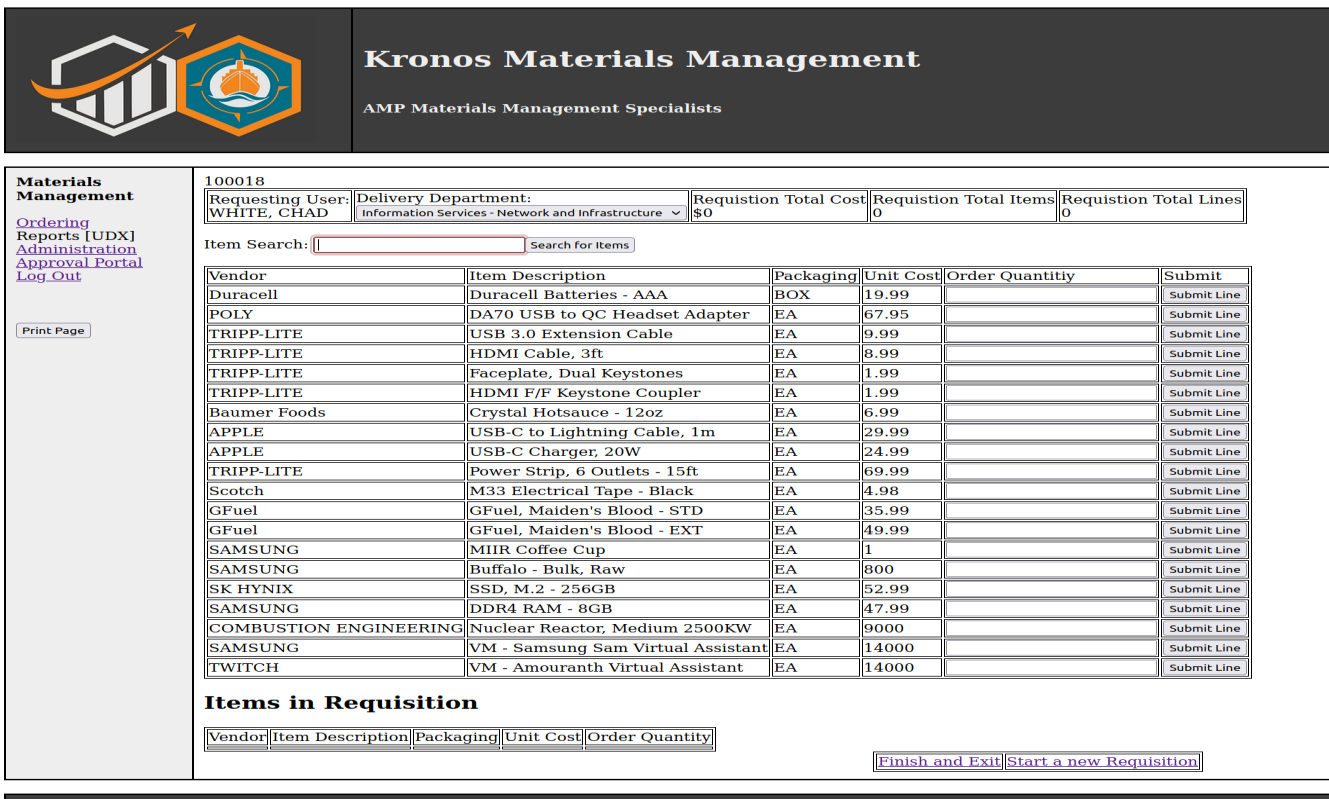
* The ability to ingest new items into the system utilizing a handheld scanner.
* Utilizing pick lists for MM users to be able to run orders to end users
* Access control flags to limit end users ability.
* Approval process for reqs at different approval amounts.

These are not all of the features that the program has the capability of. During the implantation process changes can be made to better conform the menu and functions to your workflow.

Our system is built to be able to be used by all types of employees. It was designed to be easily learned and operated efficiently with proper training. Warehouse workers will be able to move and see inventory levels from computers on the floor. While inventory/warehouse management will be able to see all movements and inventory levels.

# Scenarios

# Ordering | End user able to put in req for items



Once all the items are added the end user can click finish and exit or start a new requisition.

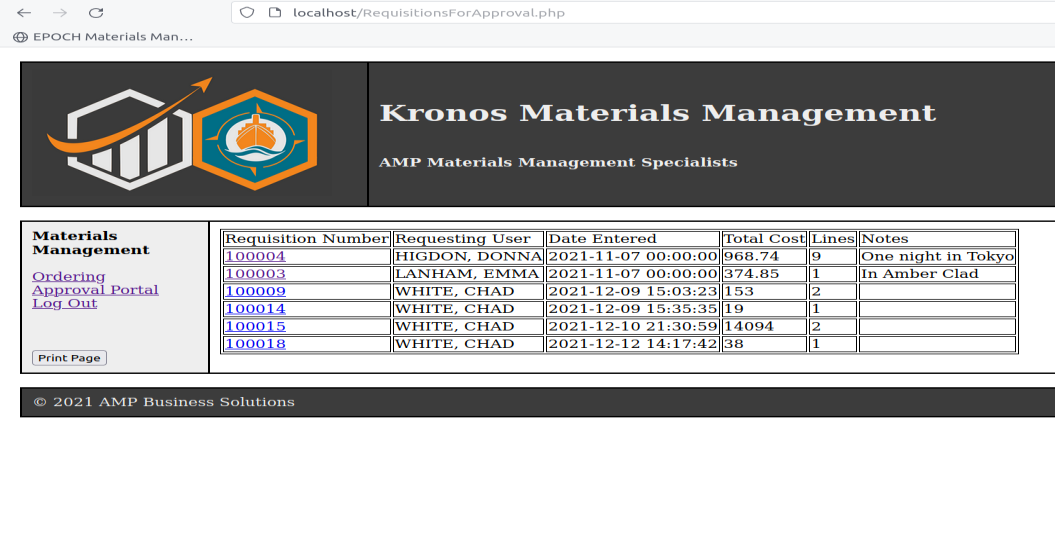
This will publish the req to the SQL table.

# Administration| Administrator permissions page



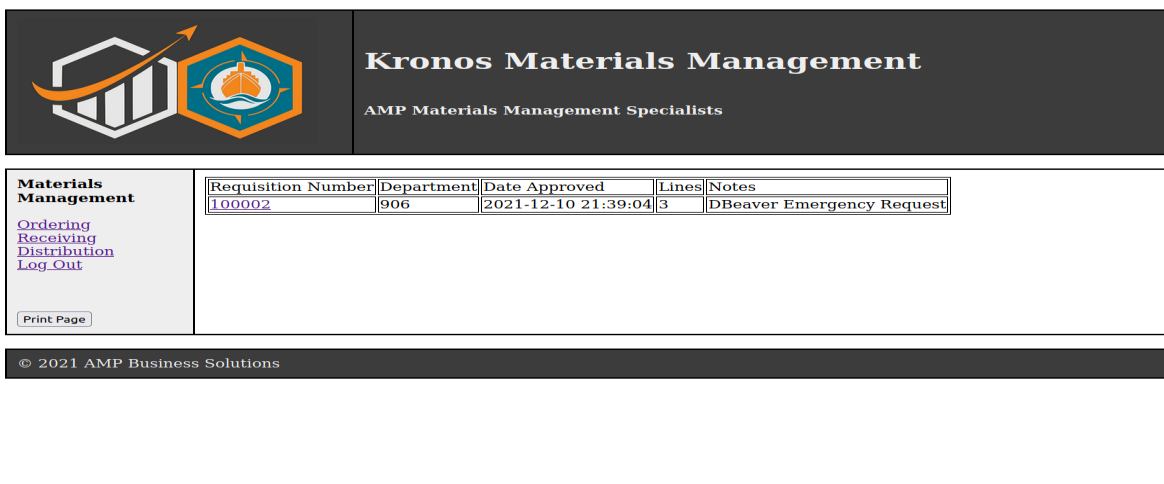
From this screen an admin can look up an end user and change their password.

# Approval | Approval users are able to approve reqs

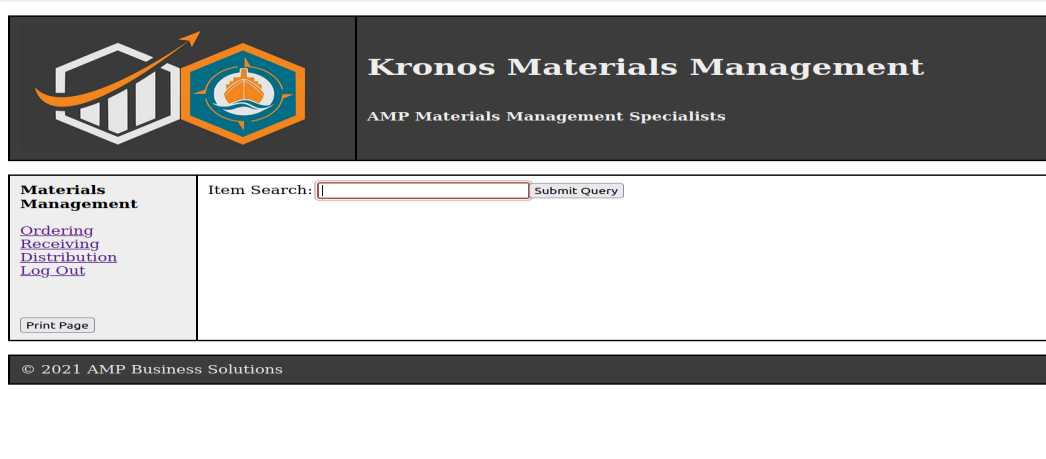


From this screen an end user with approval access will see the reqs they can approve. Once you click on the req number you can approve or deny the req.

# Materials | Materials Management



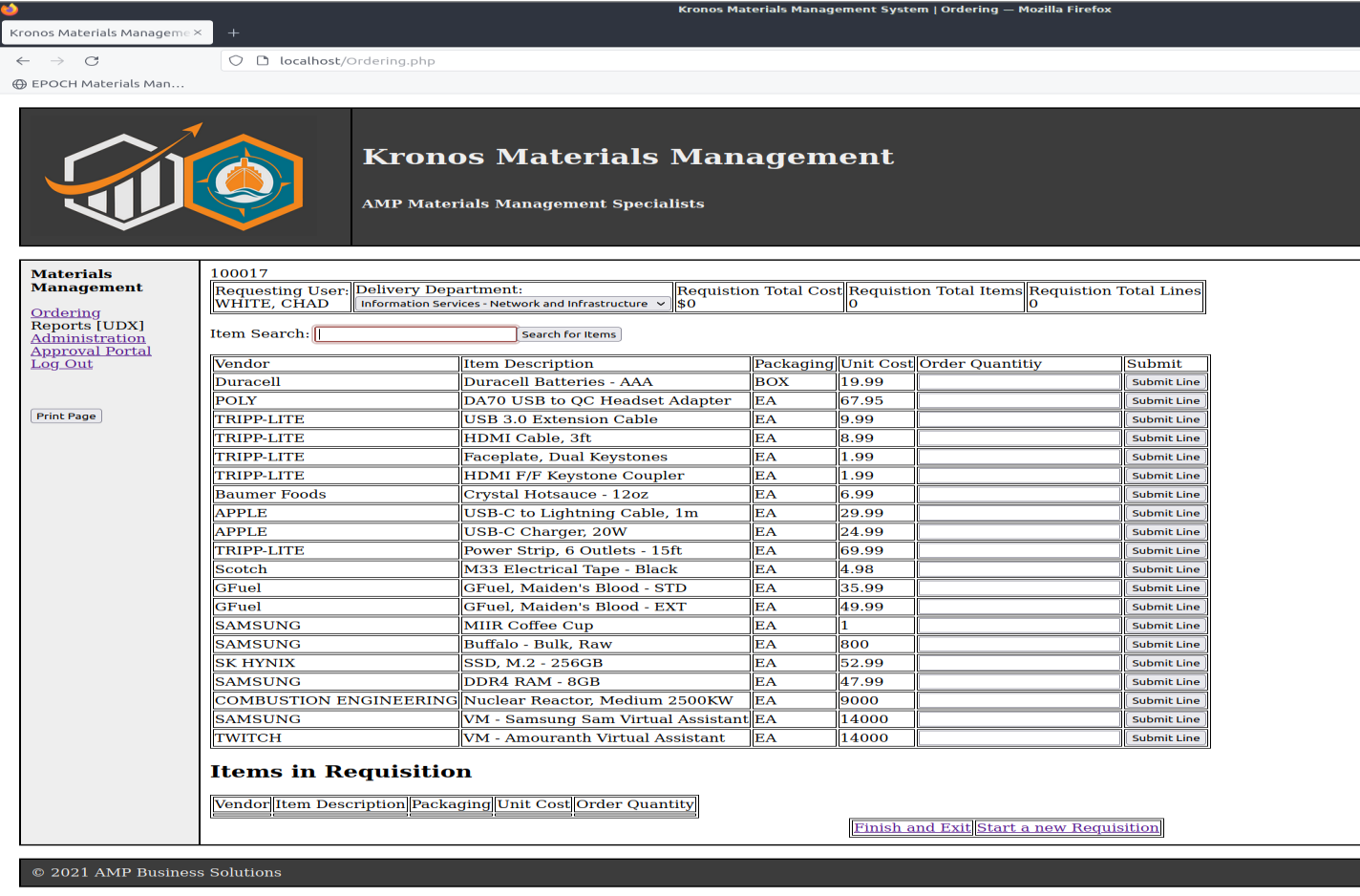
This screen is for the material management users to see reqs that have been approved and need to be pulled. When they click on the req it will transfer them to the picklist page to begin pulling the items for the requester.



This page is for the receiving portion in materials management. When an item SKU is inputed either by scanner or by hand it will look into the database to see if it exists. If it does then it will ask how many you would like to add. If it does not show it will ask for more information to add it to the database.

# Reports

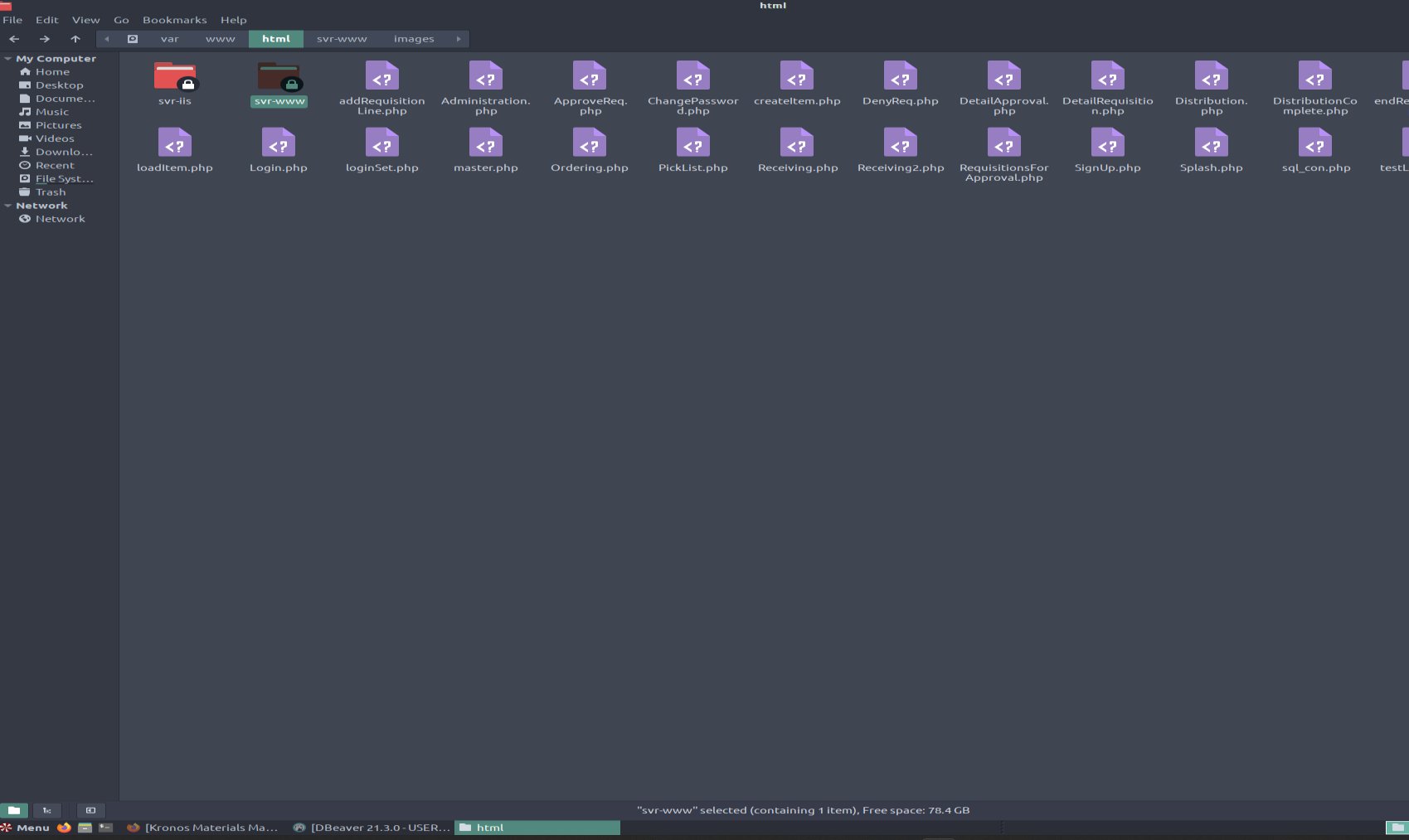
At the moment our report section is lacking. We have focused on functionality of the program over the reporting as it will be built on the infrastructure of the rest of the program. For now the main reports fall under ordering to show what items can be searched for.



# System Architecture

For the architecture of the system as it stands today in the test environment and not in production. All of the functionality and data is on one VM.

* The stand-in Web server is located in the /var/www/html/svr-www directory. For the moment it is holding the company images.
* The stand in HR data is located in the /var/www/html/svr-iis directory. This is holding all of the pictures of end users.
* The VM itself is the application server. All of the php pages are located in the /var/www/html/ directory. These are the pages you will see inside the application.



## Source Code Structure

Source code structure introduction. The following is a summary of the source code directories and their contents:

| **Code Directory** | |
| --- | --- |
| **Directory** | **Usage** |
|  |  |
| Administration.php | Source code for Administration functions and page. |
| ApproveReq.php | Source code for approving requisitions. |
| ChangePassword.php | Source code for changing passwords. |
| createitem.php | Source code for item ingestion. |
| DenyReq.php | Source code for setting the deny flag for requisition. |
| DetailApproval.php | Source code to show reqs to be approved. |
| DetailRequisition.php | Source code to pull information on a requisition. |
| Distribution.php | Source code to see what reqs need to be pulled. |
| DistibutionComplete.php | Source code to change the status flag for requisitions. |
|  |  |
| itemPicked.php | Source code to update sql for item pulls. |
| loaditem.php | Source code to to ingest a item |
| Login.php | Source code to run login verification. |
| Master.php | Source code for the master page that is used across the project. |
| Ordering.php | Source code for the ordering functionality and page. |
| PickList.php | Source code for picklist functionality and page. |
| Recieving.php | Source code for Materials Management page. |
|  |  |
| Requisitiongforapproval.php | Source code to run sql to see what reqs can be approved for the logged in user. |
| Splash.php | Source code for splash page to pull image. |
| sql\_con.php | Primary source code to connect to sql database. |
| usersummary.php | Source code to see end user data. |
|  | Highlighted rows indicate directories containing source code. |

# 

# Executables

There are no exe files being used inside of the project as it is all HTML based interactions.

# Code Architecture

The main structure for the project is below.

Peppermint Linux OS

Apache 2 - Provides web services

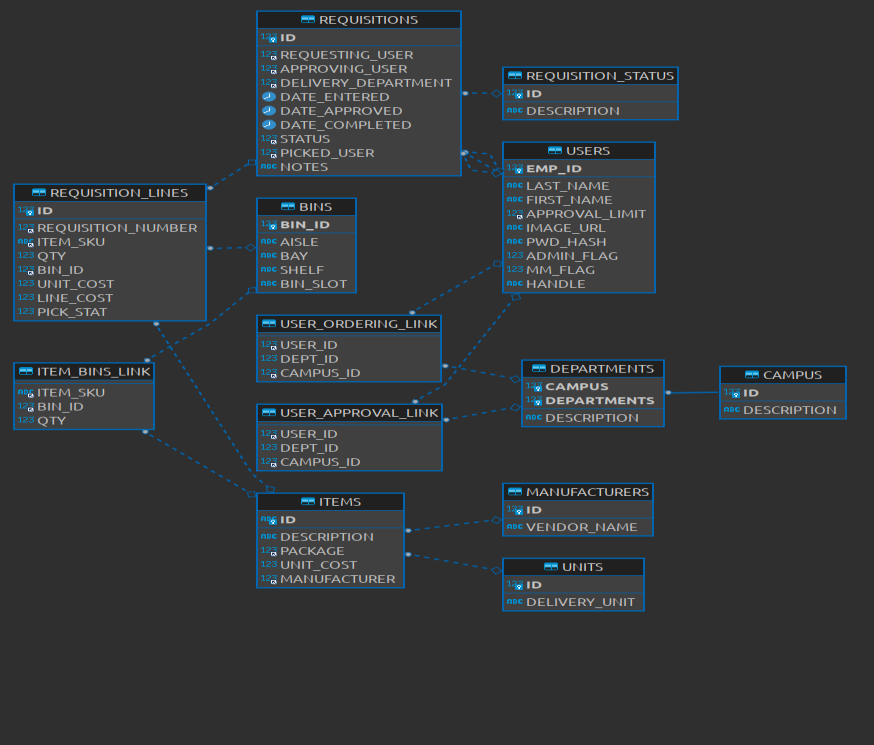
Maria DB instance - SQL server

PHP - Preprocessor , which allows us to have dynamic content.

The whole of the project is based on HTML pages. Then is backed up by PHP to run the forms and sql transactions. The source code is located in the /var/www/html/ in different php files. The descriptions are above in the project. When you navigate inside the website these are the pages you are pulling from. Not all of them are their own pages but may just be code to complete a functionality. The SQL database is loaded onto the VM and is communicated by a piece of code in the sql\_con.php file. It is used throughout the project to make it simpler to run sql updates and queries.

## Database or Data Store

Introduction to the database or data store.



Information on the sql data structure.

* Users table - This table holds all end users data and permission flags
* Requisition table - This holds the primary data for all req put into the system except for the req lines.
* Requisition\_lines - This table has a primary key to the Requisition table, but this table holds all req lines so it is easy to alter.
* Bins table - This is a static table to hold the bin structure for where items are.
* Items\_Bin\_Link- This holds the current inventory and where it is in the bins.
* User\_Ordering\_Link - this table contains data that restricts what department an end user can order for.
* User\_Approval\_Link - This table holds the departments and approvers can approve for.
* Items - This table holds SKUs and what items the system knows about.
* Manufacturers- Holds manufacturers names so it can be changed easily.
* Units- Holds unit quantities names to be easily changed.
* Departments - Holds what departments and what campuses are in the system.
* Campus - Holds campus descriptions.
* Requisition\_Status - Branched out table to make it easier to add a req status in the future.

External Files & Data

There are no external files being used in the project. The system is fully contained in the VM or in production inside the server required.

Programming Language | C#.NET

The project is written in HTML, SQL, and PHP. The HTML code is the primary viewing concept and background. The SQL code is to be able to request data and update the sql databases with information from forms inside the PHP. PHP is used to hold variables and process SQL communication statements.

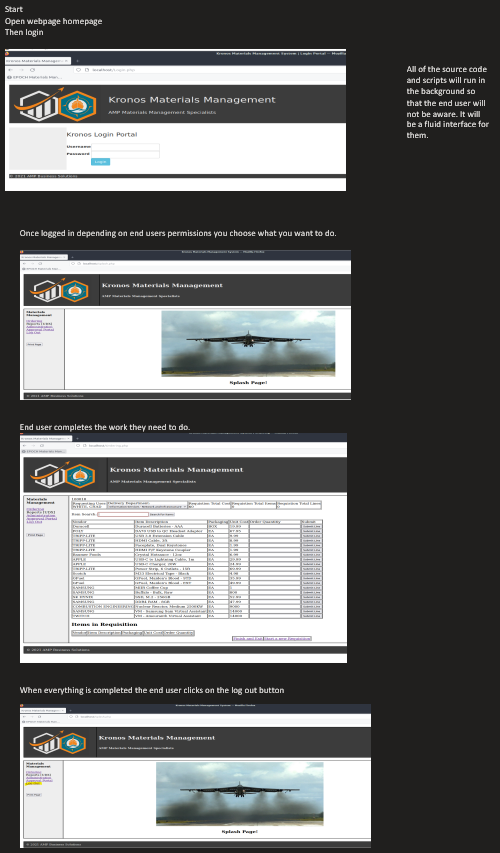
The closest thing to a built function is the sql\_con.php code. It is pulled into almost all of the pages for SQL communication.

Project Classes and Project Modules

Classes and Modules are not utilized in the source code. Much of the data manipulation and outputs are done using PHP therefore classes were not needed.

Program Start and End Flow

Describe and then diagram the program flow. Here is an example of program flow from a fat-client based PC application.

Summary

In this document we have looked at the breadth of the project set forth. There is more that the project can do and it is also adaptable to the ever changing future. We are combining HTML,SQL, and PHP to create a coherent GUI for an end user to be able to order equipment for their company. While also keeping track of users permissions and inventory for the company.

# APPENDIX B (BUILD AND RELEASE PROCESS)

When it comes to updating this project it can be done somewhat easily. Since all of the actions are based around php files. Adding a new php file and pointing to it inside another file which has already been done can be implemented. When it comes to updating already built pages or code. Running a QA version of the project making changes and testing out the bugs. Then you can take the changed file and overwrite it to the production server and the change is in place. I would schedule a downtime just in case something goes wrong with production outside of your control.

# APPENDIX C (CLIENT INSTALLATION INSTRUCTIONS)

For an end user who is wishing to access the project. If the web service is set up correctly and all connections are up and running. The only thing the end user would have to do is access the website that is bound to the project. This can be accomplished by the IIS process of the web server. Other than that the end user would have to have a web browser to access the site. Nothing further is needed to use the project for an end user.

# APPENDIX D (DEVELOPER SETUP INSTRUCTIONS)

To set up this project for production you will need 3 servers. One for the HR information that is restricted to specific personnel. The next is a Web server to broadcast the website for your end user. Then the application server which will hold the source files. Finally there will need to be a SQL server to hold all of the MM information for the application. With all of the servers up then you would need to broadcast the website via the web server with IIS. This allows the end users to access the website. You will also need to point the website to the source files on the application server otherwise it is just a blank site.

In the VM the QA is running on it is built like this.

Peppermint Linux OS

Apache 2 - Provides web services

Maria DB instance - SQL server

PHP - Preprocessor , which allows us to have dynamic content.

If this is to move to production it may need to be set up for a Microsoft OS which may cause some communication issues.