



Easy Manage Facilities Management System

Design Document

February 23rd, 2018

Tyler Cockburn
Nathaniel DeSimone
Nicholas Giordano
Christopher Hobson
Eric Lynn
Brennan Ringel
David Yeager

Table of Contents

1	Purpose.....	3
2	Scope.....	3
2.1	Exclusions, Assumptions and Limitations.....	3
3	Solution Design Overview.....	3
4	Technical Architecture.....	3
4.1	Hardware Inventory, Specifications, and Limitations.....	3
4.1.1	Servers.....	3
4.1.2	Input / Output Paths.....	4
4.1.3	Other Devices.....	4
4.1.4	Middleware Specifications.....	4
4.2	Other Hardware Interfaces / External Integration Points.....	4
4.3	Easy Manage High Level / Physical Layout.....	4
4.3.1	High Level Diagram.....	4
4.3.2	Component Diagram.....	4
4.3.3	Modular / Component Description.....	5
4.4	Activity Diagram.....	5
4.5	Use Case Diagrams.....	6
4.6	Web Page Mockup Diagrams / Functionality.....	7
4.7	User Stories.....	16
4.8	EER Diagram / Database Specification.....	17
4.9	Prototype Description.....	17
4.9.1	Login Page.....	17
4.9.2	Google Authentication & Google Calendar.....	17
4.9.3	Main Screen.....	17
4.9.3.1	Navigation Bar.....	17
4.9.3.2	Content Window.....	17
4.9.3.3	Header.....	17
4.9.4	Entries.....	17
4.9.4.1	View Entry.....	18
4.9.4.2	Create Entry.....	18
5	Configuration Specification.....	18
5.1	Web Server Software Configuration.....	18
5.2	Database Backup Configuration.....	18
6	Solution Design Specifications.....	18
6.1	Software Description.....	19
6.2	Coding Standards.....	19
6.3	Solution Data, Information View, Data Requirements.....	20
7	Roles and Responsibility.....	25
8	Terms and Definitions.....	25

1 Purpose

- *The purpose of this document is to outline the technical design of our implementation of a Facility Manager's Web Application. The document will have detailed information regarding development of the application. The proposed peruses of this record are future developers who are to additionally work with this product for continued development.*

2 Scope

2.1 Exclusions, Assumptions and Limitations

- *The development team is designing the web application such that in future, more functionalities can be added to the current design without any difficulties.*
- *There are some features for this project that we want to leave open for future development, even though they may not be completed due to the limited time provided during this semester. Integration to native applications on the google play and iOS store for android and iPhone applications are one example of the features that may not be added this semester to the project. However, the project will be designed keeping in mind that the system should be easy to modify and maintain for future changes.*
- *While we will have users logging in using google authentication, we have also designed the database to allow for users to log in without Google Authentication as something to add if there is time to add that functionality. The password would be hashed with the username and that hash would be stored in the database.*

3 Solution Design Overview

- *In order to allow Facilities Managers to organize data such as contacts, work orders, purchase orders, as well as any other notes they may need to make, our system will store all of this data in our database, and make it accessible from our website. They should be able to search for contacts whether they be employees, or any other kind of contact outside their organization. These will be done by querying our database.*
- *They will also be able to create and view their entries, and filter them by whether or not they are a purchase or work order, or just a generic entry. These can be searched by querying the database to see if one has a certain item, title, or description. Creations and changes will be stored in the database.*
- *Permissions will be stored in the database as an integer, with certain bits turned on for certain permissions. The person who creates the account for an organization will have all permissions for that organization. One of the permission options will be the option to customize other user's permissions. Other ones will be to create and modify various things on the website. There will also potentially be some read permissions changed if the need is seen for them.*
- *Users will be authenticated with Google authentication. If a user gives the website a password, they will have the option to login with that password in the event google is down (or if they prefer that method). Users will be able to login with a username and password, which will be hashed and compared to a stored hashed value on the server's database.*

4 Technical Architecture

4.1 Hardware Inventory, Specifications, and Limitations

Considering this inventory system will be web-based, the development team will be using HTML/CSS/Bootstrap, jQuery, MySQL database, Flask-Python and an AWS server.

4.1.1 Servers

The Development team is using a front-end website based on an AWS server

4.1.2 Input / Output Devices

Devices will be all types of computers which will support web browsers such as

Safari, Google Chrome, Microsoft Edge, and Mozilla Firefox.

4.1.3 Other Devices

Web browsers in smartphones will be able to access the web applications. Web pages use Bootstrap framework which supports mobile view.

4.1.4 Middleware Hosting

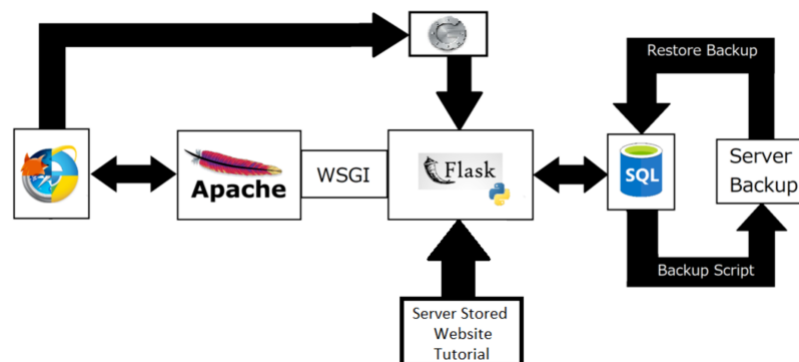
The database used is MySQL and is hosted by the AWS EC2 instance. The development team has used Flask (Python Version 2.7.12) to setup the endpoints and to define the APIs. jQuery is used to send queries to the Flask backend. Git Bash has been used for source control / configuration management during development, with the remote repositories being hosted on GitHub.

4.2 Other Hardware Interfaces / External Integration Points

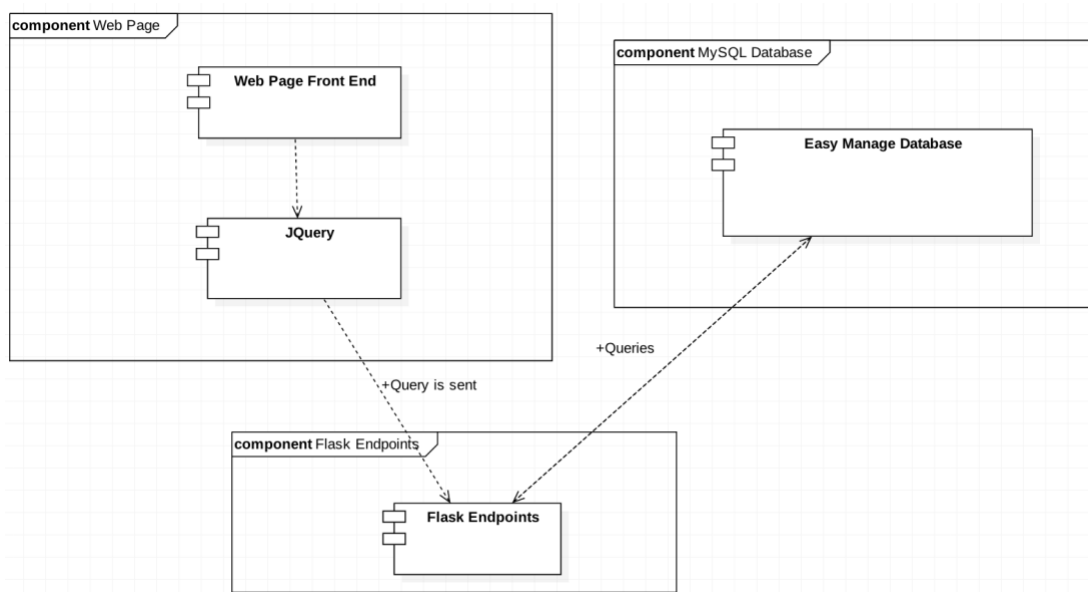
The application will be accessed first by authenticating with Google and integrating with Google Calendar. Users will be able to login with a username and password in the event Google is down, which will be hashed and compared to a stored hashed value on the server's database.

4.3 Easy Manage High Level / Physical Layout

4.3.2 High Level Diagram



4.3.2 Component Diagram



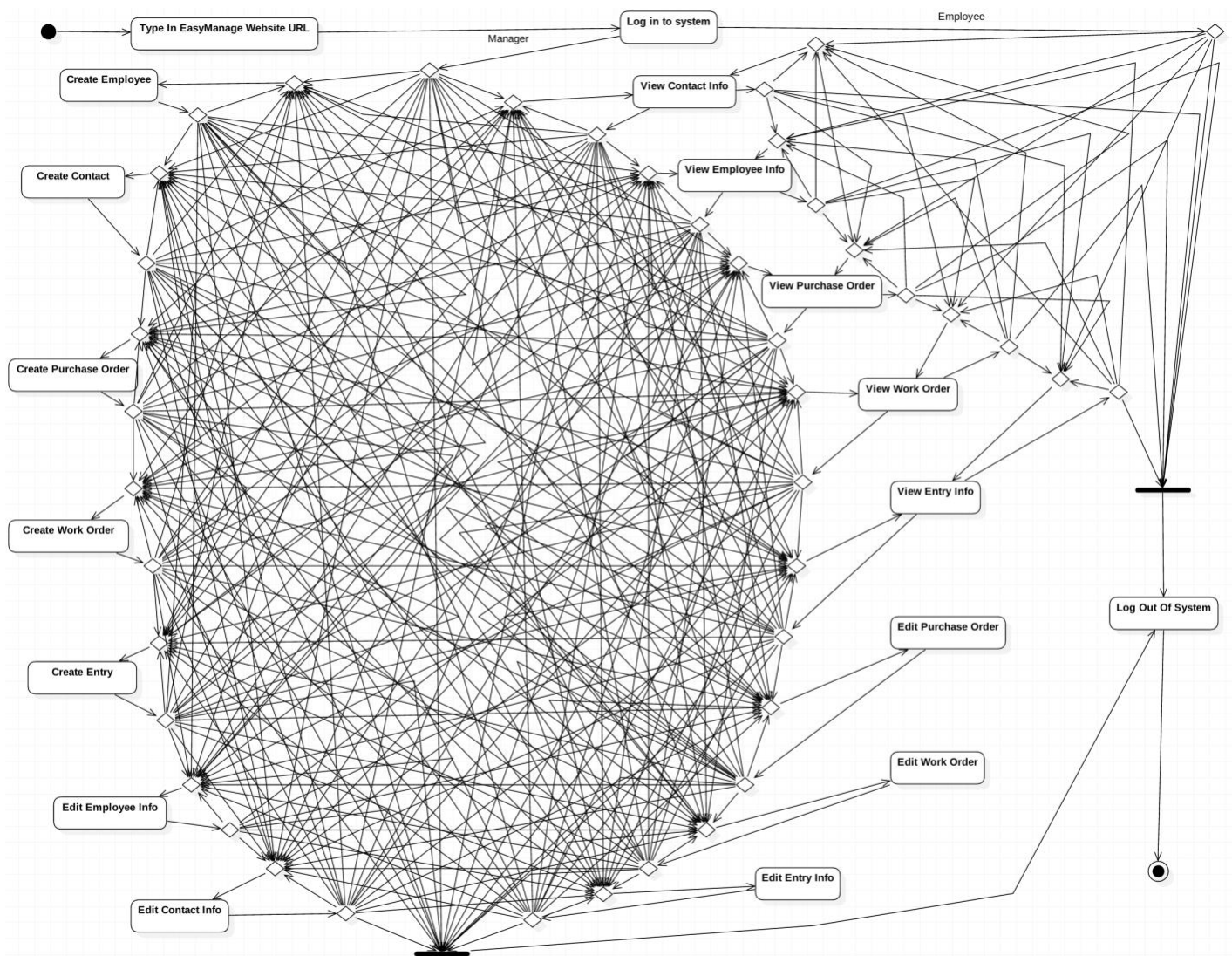
4.3.3 Modular / Component Description

MySQL Database Backend: Our application will be based on a MySQL database. The database will contain many stored procedures to carry out many database operations. These stored procedures will be called by the Python-Flask Backend Framework.

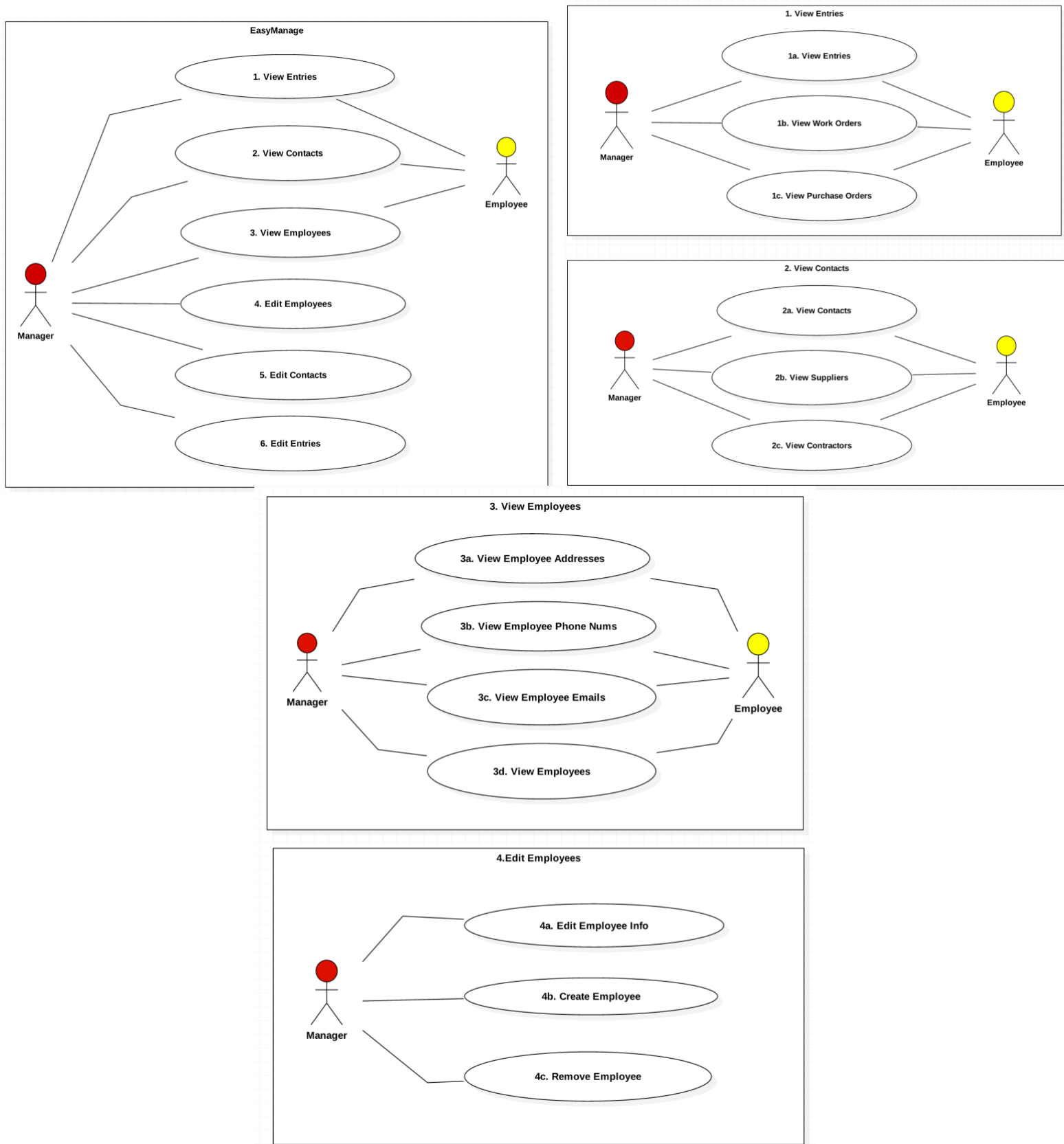
Python-Flask Backend Framework: REST endpoints are defined in Python-Flask. When a certain endpoint is hit by the client-side (via jQuery/AJAX), the corresponding function in Python-Flask will execute. It will call MySQL stored procedures and perform other logical tasks.

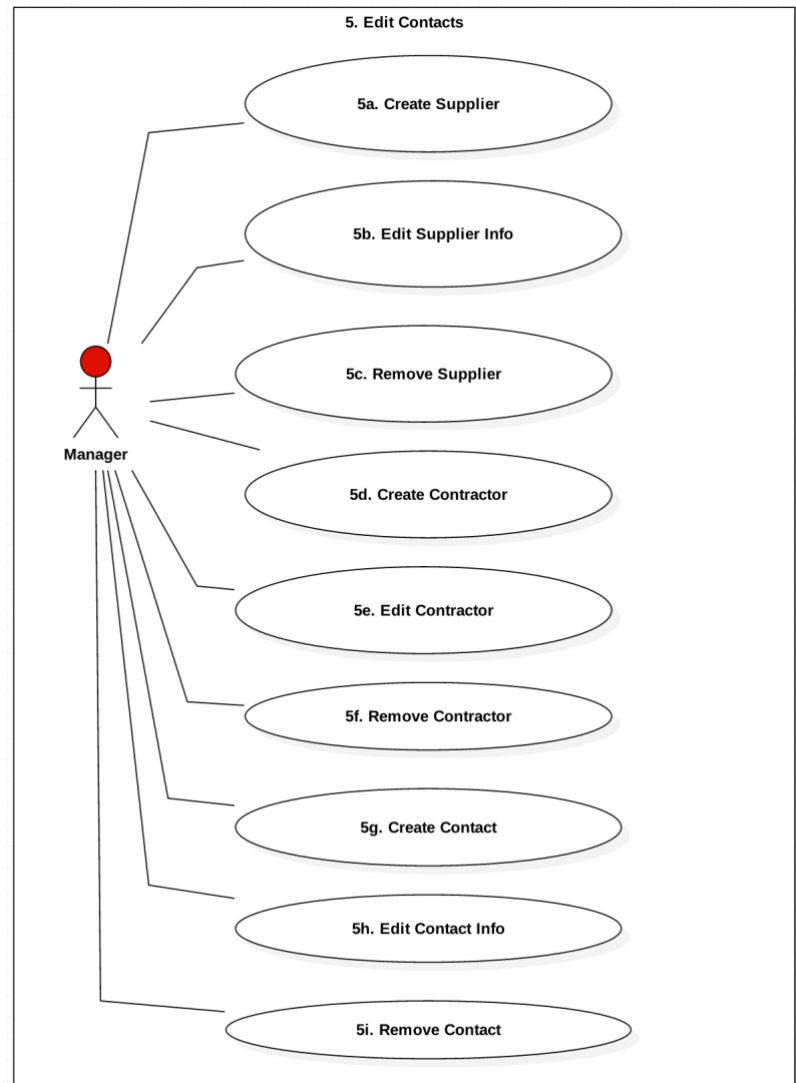
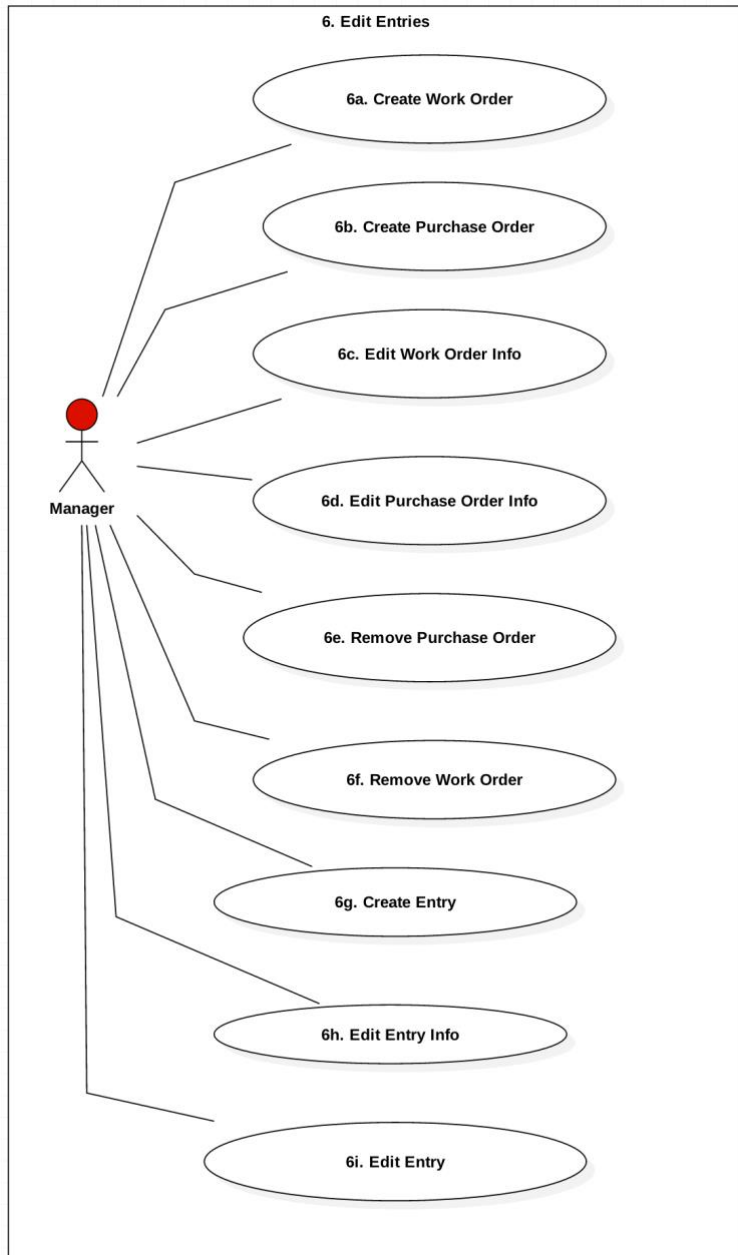
JavaScript / jQuery Front End: JavaScript with jQuery will be included in each HTML document. The role of these are to send REST calls to our Python-Flask backend framework when the user interacts with certain elements of the page.

4.4 Activity Diagram



4.5 Use Case Diagram





4.6 Web Page Mockup Diagrams / Functionality

Login Screen: The login page will be displayed to anyone that goes to the web address.

Endpoints: /login

Login Button: This button takes the user to a Google Authentication Page where he or she will have to sign in. If credentials are valid, it will take him or her to the Main Screen.

The Main Screen: The main screen is where all users, Managers or Employees will be taken to. The screen will have our logo at the top, across from it will be a message displaying who is logged in and the sign out button.

Endpoints: /logout, /home, /entries, /entries/new, /entries/search?=<filter>, /entries/<entryID>/remove

Navigation Bar: The Navigation bar contains all the ways to navigate the application. A user can get to any part of the application from click on the links from this tab. This bar

never moves while going through application. It also always displays the same options for the user. Users will know where they are currently in the application by the Navigation Bar.

- **Entries:** Main Page where entries are
- **Work Orders:** Shows all work orders
- **Purchase Order:** Shows all purchase orders
- **Contacts:** Shows contacts related to current or past work orders
- **Calendar:** Google Calendar associated with account
- **General Info:** About the Program, the development team, etc.
- **Tutorials:** Tutorial Screen (Stretch Goals)
- **Settings:** Settings for User account

The Content Window: The box to the right of the navigation bar will be where the content is displayed to this box only. Content should not overflow past this box. This box will take up a large portion of the screen to fit all information within the application.

Sign Out Button: Button is used to sign users out, taking them back to the login page.

Entries: Entries page will contain all the entries currently in the system. This includes both Work Orders and Purchase Orders.

Endpoints: /logout, /home, /entries, /entries/new, /entries/search?=<filter>, /entries/<entryID>/remove, /entries/<entryID>,/entries/<entryID>/modify, /contacts/new, /contacts/search?=<filter>

Entry Number: Each Entry will have a corresponding number associated with it. This number is self-generated and goes in consecutive order. An Example would be E#00001, the next would be E#00002, and so on. This information will be displayed to the left of the table displaying the Title and Description.

Entry Title: Each Entry has a title. This title will be displayed in the middle of the table, in between Entry Number and Description.

Entry Description: The first part of the Entry Description will be displayed all the way to the right. Some descriptions will be too long so just the first part of them will be displayed.

Create Entry: Given the correct permissions, users will be able to create an Entry. This button will fill the content window up with the “create entry” screen. On this screen, the user will be promoted to fill out the Title, what type of order, which contact is associated with the entry, a cost, and a description. The Date Created field will be locked to the current date.

Attach Button: Users will be able to attach documents to the entries. Clicking this will bring up a file explorer to attach document.

Cancel Button: Users will be able to cancel the entry if needed. This will bring them back to the screen with the entries on it.

Save Button: As long as the appropriate fields are filled out, users will be able to save the entry and store it in the program.

View Entry: Given the correct permissions, users will be able to view entries. On either Entries, Work Orders, or Purchase Orders, users will be able to click on any entry cell and the corresponding entry will appear in the context box. If a user clicks off the screen, the entry will go away, and the user will be brought back the previous screen.

Edit Button: Users will have to click the edit button and, given the appropriate permissions, will be able to edit the current entry.

Cancel Button: Users will be able to cancel the edits made on the entry.

Save Button: User can save edits made with this button.

Search Entries: Users will be able to search through entries with the search bar at the top. When a user types a word into the search bar, the related entries will appear on the screen. This also works in searching through All Entries, Work Orders, and Purchase Orders.

Filter Entries: Users will be able to set filters on the search so find entries easier. (Stretch Goal)

Work Orders: This has all the same functionality as the previous “Entries” page, however, it will only contain Work Order entries.

Endpoints: /logout, /home, /entries, /entries/new, /entries/search?=<filter>, /entries/<entryID>/remove, /entries/work/<entryID>./entries/<entryID>/modify, /contacts/new, /contacts/search?=<filter>

Purchase Orders: This has all the same functionality as the previous “Entries” page, however, it will only contain Purchase Order entries.

Endpoints: /logout, /home, /entries/new, /entries/search?=<filter>, /entries/<entryID>/remove, /entries/purchase/<entryID>, /entries/<entryID>/modify, /contacts/new, /contacts/search?=<filter>

Contacts: Users will be able to view all the contacts that are stored in the system from this page. It will fix the content window with a list of all the available contacts. Users will be able to click on a contact from the list and bring up a window in the content box to view all information regarding this contact

Endpoints: /logout, /home, /contacts, /contacts/new, /contacts/search?=<filter>, /contacts/supplier, /contacts/supplier/search?=<filter>, /contacts/contractor, /contacts/contractor/search?=<filter>, /contacts/<contactID>, /contacts/<contactID>/modify, /contacts/<contactID>/remove, /contacts/<contactID>/personnel, /contacts/<contactID>/personnel/search?=<filter>

Contact Name: The name of the person will populate on the very left side. If it is an organization, that corresponding name will be in that field.

Contact Company: The company associated with the contact will be listed. The number of this company will also be paired with this information. The company will be displayed to the right of their name.

Contact Phone and Alt Phone: The numbers and alternative numbers for each contact will be show. It is possible for some of the numbers not to fix in the screen, so the user will have to open contact to view all the information more neatly.

Contact Address: The contact’s address or company address will appear here. This is only acquired by clicking on the contact.

View Contact: By clicking on one of the cells, the user will be able to view the contact and all the additional information not displayed in the list. Clicking away from the screen will make this window go away.

Edit Contact: Given the right permissions, a user will be able to edit the existing contact by pressing the edit button. The user will have to press save after or cancel if they don’t want the changes to be made.

Create Contact: Given the right permissions, the user will be able to create a contact. The same above fields are all to be filled out on the create contact page. User either hits save or cancel if they wish to not make the contact anymore.

Search Contact: User will be able to search through the contacts list by name, number, address, or company with the search bar above the content box.

Calendar: Users will be able to see their Google Calendar. All functionality here will be related to how google calendar works. No additional functionality needed.

General Info: Users will be able to collect general information about our development team and program. This is NOT a tutorial page, just some paragraphs in the content box that displays this information.

Endpoints: /logout, /home, /about, /about/axolDev, /about/ezManage

Tutorials (STRETCH GOAL): Users will be able to view tutorial of our application. These tutorials will be simple walkthroughs of each page.

Endpoints: /logout, /home, /help, /help/search?=<filter>, /help/<tutorialID>

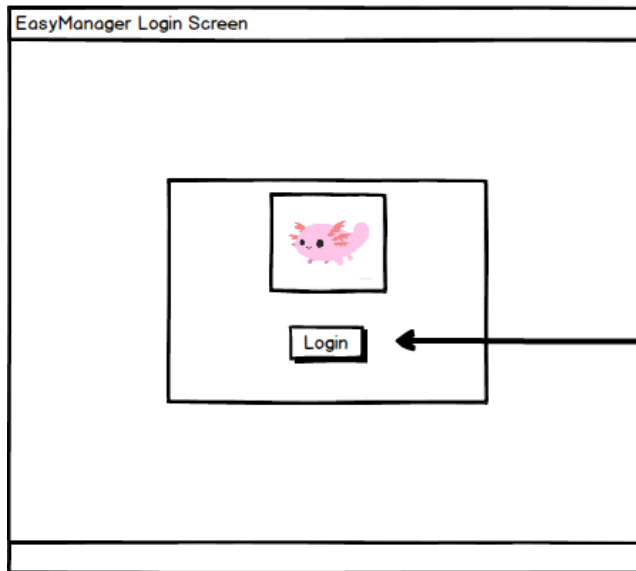
Settings: Manager will be able to adjust settings for the users on the application.

Edit permissions: The user will have permissions to perform certain tasks with the entries and contacts. Users that do not have permission will lose access to these certain buttons on screen. The Manager will have to select a user from a group of its Employees. Once selected, the Manager will be able to toggle the certain permissions

Endpoints: /logout, /home, /account, /account/<userID>, /account/settings, /account/settings/permissions, /account/settings/permissions/<userID>, /account/settings/permissions/<userID>/modify, /account/<userID>/settings,

/account/settings/modify, /account/<userID>/settings/modify, /account/personnel,
 /account/personnel/new, /account/personnel/search?=<filter>,
 /account/personnel/<employeeID>, /account/personnel/<employeeID>/modify,
 /account/personnel/<employeeID>/remove

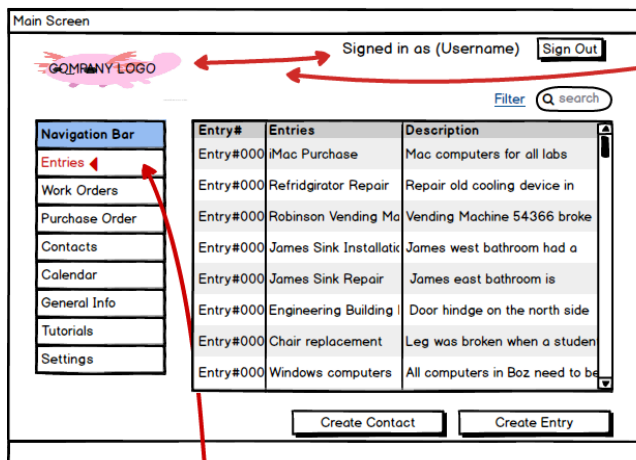
Login Screen



THERE WILL NOT BE A USERNAME/
 PASSWORD TEXTFIELD

LOGIN BUTTON WILL DIRECT USER TO
 GOOGLE LOGIN PAGE
 AFTER PROPER AUTHENTICATION, IT WILL
 REDIRECT USERS TO MAIN SCREEN

Main Screen



The Company Logo and "Signed in/Sign out"
 will appear on top of every page except the
 login.

When a user is on a certain page, like Entires
 currently, it will be highlighted so it is clear to
 the user which tab they're on

Create New Entry

Main Screen

Signed in as (Username) [Sign Out](#)

[Filter](#)

Navigation Bar

- Entries
- Work Orders
- Purchase Order
- Contacts
- Calendar
- General Info
- Tutorials
- Settings

Create New Entry

Title of Entry

Work Order Type:

Contact:

Cost:

Created:

Description:

Additional comments

Clicking away from the entry will close it without saving.

When the contact is clicked, it will bring up a list of stored contacts. New contacts can also be made from here. This screen will bring on top of the edit page. The current contact will be highlighted.

Choose Contact

Name	Company	Phone	Alt Phone
John Apple	Axolotls	(555)-234-7345	N/A
Jacky Myers	Rowan	(524)-232-7745	NA
Henry Harris	Axolotls	(555)-234-7345	N/A
David Smith	Axolotls	(555)-234-7345	N/A

for adding and editing File attachments

File

cuteAxolotl.png	<input type="checkbox"/>
managerReport.txt	<input type="checkbox"/>

Work Orders

Main Screen

Signed in as (Username) [Sign Out](#)

[Filter](#)

Navigation Bar

- Entries
- Work Orders
- Purchase Order
- Contacts
- Calendar
- General Info
- Tutorials
- Settings

Entry#	Entries	Description
Entry#0002	Refridgirator Repair	Repair old cooling device in
Entry#0003	Robinson Vending Machir	Vending Machine 54366 br
Entry#0004	James Sink Installation	James west bathroom had
Entry#0005	James Sink Repair	James east bathroom is

Only Current Work Orders appear here

Search Feature

Main Screen

COMPANY LOGO

Signed in as

Q James

Entry#	Entries	Description
Entry#000	James Sink Insta	James west bathroom
Entry#000	James Sink Repc	James east bathroom

Navigation

When searching for an Entry, users can type the name in the top which will result in all of the entries disappearing except for possible matches

Purchase Orders

Main Screen

COMPANY LOGO

Signed in as (Username)

Filter Q search

Entry#	Entries	Description
Entry#000	iMac Purchase	Mac computers for all labs
Entry#000	Engineering Building	Door hindge on the north side
Entry#000	Chair replacement	Leg was broken when a student
Entry#000	Windows computers	All computers in Boz need to be

Navigation Bar

Entries

Work Orders

Purchase Order ◀

Contacts

Calendar

General Info

Tutorials

Settings


Only Current Purchase Orders appear here

Create Contact

Create Entry

Contacts

Main Screen

 Signed in as (Username) [Sign Out](#)

[Filter](#)

Navigation Bar

- Entries
- Work Orders
- Purchase Order
- Contacts**
- Calendar
- General Info
- Tutorials
- Settings

Choose Contact


Name	Company	Phone	Alt Phone
John Apple	Axolotls	(555)-234-7345	N/A
Jacky Myers	Rowan	(524)-232-7745	NA
Henry Harris	Axolotls	(555)-234-7345	N/A
David Smith	Axolotls	(555)-234-7345	N/A
Alex Bates	Rowan	(524)-232-7745	NA

[New Contact](#)

[Create Contact](#) [Create Entry](#)

View Contact

Main Screen

 Signed in as (Username) [Sign Out](#)

[Filter](#)

Navigation Bar

- Entries
- Work Orders
- Purchase Order
- Contacts**
- Calendar
- General Info
- Tutorials
- Settings

Viewing Contact

NAME:

COMPANY:

EMAIL:

PHONE:

ALT PHONE:


ADDRESS:

[Remove](#) [Edit](#) [Cancel](#) [Okay](#)

[Create Contact](#) [Create Entry](#)

Create New Contacts

Main Screen

 Signed in as (Username) [Sign Out](#)

[Filter](#)

Navigation Bar

Entries

Work Orders

Purchase Order

Contacts

Calendar

General Info

Tutorials

Settings

Create New Contact

NAME:

COMPANY:

EMAIL:

PHONE:

ALT PHONE:


ADDRESS:

[Edit](#) [Cancel](#) [Ok](#)

[Create Contact](#) [Create Entry](#)

Calendar

Main Screen

 Signed in as (Username) [Sign Out](#)

[Filter](#)

Navigation Bar

Entries

Work Orders

Purchase Order

Contacts

Calendar

General Info

Tutorials

Settings

FEBRUARY 2018


S	M	T	W	T	F	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	1	2	3
4	5	6	7	8	9	10

Google calendar will handle most of this functionality

General Info

Main Screen

Signed in as (Username) [Sign Out](#)

 COMPANY LOGO

Navigation Bar

Entries

Work Orders

Purchase Order

Contacts

Calendar

General Info ◀

Tutorials

Settings


ABOUT EasyManager:

ABOUT US:

Tutorials

Main Screen

Signed in as (Username) [Sign Out](#)

 COMPANY LOGO

Navigation Bar

Entries

Work Orders

Purchase Order

Contacts

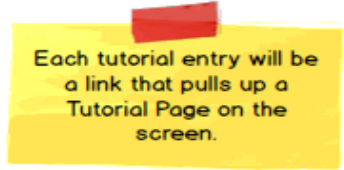
Calendar

General Info ◀

Tutorials

Settings

Tutorial:
Navigation Bar
Opening Entries
Editing Entries
Settings



Each tutorial entry will be a link that pulls up a Tutorial Page on the screen.

STRETCH GOAL

Settings

Main Screen

Signed in as (Username) [Sign Out](#)

COMPANY LOGO

Navigation Bar

- Entries
- Work Orders
- Purchase Order
- Contacts
- Calendar
- General Info
- Tutorials
- Settings

Permissions

User

View Entries ☒

Create New Entry ☒

View Contact ☒

Create Contact ☒

[Cancel](#) [SAVE](#)

Click "user" box to select user.Permissions subject to change

4.7 User Stories

- As a manager of a facility using the Easy Manage system, I would like to be able to view all entries made into the database system by myself and employees under me for the sake of inventory management. This includes being able to view purchase orders for supplies and work orders for services.
- As a manager, I would also like to be able to view general information about my contacts. Contacts should be split up into contacts, suppliers of products, and contractors for out-of-house work.
- As a manager, I would like to be able to pull up my employees' name, address, email, or phone number. I'd like to be able to edit my employees' information as well as being able to add or delete employees.
- As a manager, I'd like to be able to add and remove different suppliers. I'd like to be able to edit suppliers' information in the system. I'd also like to be able to add, remove, and edit contractors in the database.
- As a manager, I'd like to be able to create purchase orders and create work orders. I'd like to be able to remove purchase orders and remove work orders. And I'd like to be able to edit purchase orders and work orders.
- As an employee using the Easy Manage system, I would like to be able to view entries I've made into the inventory system. I'd like to be able to see work orders I've submitted and purchase orders as well.
- As an employee, I'd also like to be able to see contacts, suppliers, and contractors' information so that I can contact people to complete necessary work. I'd also like to be

able to use Easy Manager to see my personal information in the system, including my listed address, email, name, and phone number.

- As a general end user of the Easy Manager system, I'd like to be able to see a tutorial on how to navigate the software.
- As an end user, I'd like the software to have a navigation bar on the side to organize where I can find everything like my work orders, entries, purchase orders, calendar, contacts, general info about the program, and settings.
- As an end user, I'd like access to a calendar to see my schedule for the day, and I'd like to be able to add or remove events from this calendar.
- As a manager, I'd like to be able to add events to my employees' calendars.
- As a general end user of the system, I'd like there to be a settings tab (actual settings are to be announced.)
- As a user, I'd like there to be a search bar to search the current page for whatever text I want. This way, I can search for people/organizations by name, phone number, or company number.
- I'd like to be able to sign in with Gmail as a user because it is easy. I would also like the option as a user of Easy Manager to not sign in with Gmail, in case I do not have Gmail.
- I'd like the Easy Manager application to have a "Sign Out" button for obvious security purposes.
- As an end user of the system, I would like data about me that is stored on the server to be encrypted using my password as a key.
- As an end user, I would also not like my password stored in plaintext. Rather, a secure, pseudo-unique hash of the password should be stored in the system.

4.8 EER Diagram / Database Specification

See 6.3 Solution Data, Information View and Data Requirements

4.9 Prototype Specification

4.9.1 Login Page

The login page will be very minimum. It will display our logo, name, and one "login" button. This button will take users to Goggle Authentication Page where they will have to log in.

4.9.2 Google Authentication & Google Calendar

Our prototype will implement the Google Authentication System for Login. A successful login with Google Authentication will take users to the "Entries" Page. Failed login will not take user in. The prototype will also feature an open source Calendar widget with Google Calendar API built in. The widget can be found in the following link: <https://fullcalendar.io/download>

4.9.3 Main Screen

4.9.3.1 Navigation Bar

The Navigation Bar on the left will be displayed. It will contain one item, "Entries". This will take users to the "Entries" page.

4.9.3.2 Content Window

The Content window located to the right of navigation will only display entries and when users can create entry.

4.9.3.3 Header

A simple header will be displayed on top of Navigation Bar and Content Window. This will have a logo and our product name.

4.9.4 Entries

On the “Entries” page, users will be able to view entries already put in on the Content window. The entries will be listed one-by-one. Entries will contain an Entry number, Title, Description, and the date it was created.

4.9.4.1 View Entry

When a user clicks on the entry on the page, it will bring up all the previously mentioned information in Content Window.

4.9.4.2 Create Entry

A user can create a new entry. They will fill out a Title and Description. The date and entry number created will be made as well. This created entry should show up on the “Entries” page.

5 Configuration Specification

5.1 Web Server Software Configuration

The web application running in our development environment is currently residing on an Amazon proprietary version of Ubuntu 16.04. Some software dependencies include:

- Python 2.7.12
- Pip Python Package Manager 9.0.1
- Flask 0.12.2
- Flask-MySQL 1.4.0
- MySQL 5.5.57
- Bootstrap 3.3.5
- jQuery 3.2.0
- Gspread 0.6.2 (Google Client Library for Python)
- Apache 2.4.27, mod_ssl 2.2.34, mod_wsgi 4.5.24

The configuration files needed are hosted in the GitHub repository. The only ports needed for our application are the standard HTTP/S ports 80 and 443. Apache is configured to proxy all Python requests to a local socket, so the Flask port 5000 is not used. Apache is also configured to serve the static content without invoking Python to maximize efficiency.

5.1 Database Backup Configuration

The system is set to back up the database every night at midnight. There will be a retention of 3 daily backups, 4 weekly backups and 2 monthly backups on any given day. The backups are compressed using the bzip2 algorithm to save space.

6 Solution Design Specifications

6.1 Software Description

Easy Manage is a web-based application designed to assist facilities Facility Managers in their day to day tasks. Managers can manage permissions of employees to have the ability to view and edit contacts, entries, work-orders, and purchase-orders. Access to this information helps users easily track and view information that is pertinent to performing their roles effectively and efficiently. Managers also have the ability to assign work orders to employees as well as contract out work orders to other contractors. Managers also have the ability to purchase items from various suppliers and keep track of it through purchase orders. Also with Google Authentication integration, users can easily create an account to have access to a multitude of services.

6.2 Coding Standards

Some coding standards include:

- Using useless primary keys in database tables. Refactoring to use integers as unique keys/index values in database tables wherever possible.

- To keep track of versions of this web application, as well as the progress of the code, GitHub is used extensively. There will be a master branch with a working version, and a dev branch off of master. Individual functionalities will be branched off of dev, and merged back in when finished. Dev will be merged into master occasionally when it is stable, and has no branches off of itself.
- Code will be commented wherever possible, especially if the functionality is not clear just by looking at it. Things like how permissions will be stored will especially be commented since they will be stored as integers, with certain bits turned on for certain permissions.

6.3 Solution Data, Information View and Data Requirements

Database is formed by using MySQL consists of the following tables:

- organization, employee, contact, user, entry, work_order, purchase_order, supplier, contractor, pdf, address, phone_number, email, employee_address, employee_phone_number, employee_email, contact_phone_number, contact_email, contact_address, out_house_performs, in_house_performs, type_of_supplier, supplier_to_type, contractor_to_type, type_of_contractor

Table Name	Column Names / Data Types *Foreign Keys in BOLD*
organization	<ul style="list-style-type: none"> • organization_id – INT *NOT NULL *AUTO_INCREMENT • user_id – INT *NOT NULL • organization_name - VARCHAR(256) <u>Primary Key:</u> organization_id
employee	<ul style="list-style-type: none"> • employee_id – INT *NOT NULL *AUTO_INCREMENT • organization_id - INT *NOT NULL • first_name – VARCHAR(64) • last_name – VARCHAR(64) • position – VARCHAR(256) <u>Primary Key:</u> employee_id
contact	<ul style="list-style-type: none"> • contact_id – INT *NOT NULL *AUTO_INCREMENT • organization_id - INT *NOT NULL • name – VARCHAR (256) • company_name – VARCHAR(256) *NULLABLE • d_type – VARCHAR(16) <u>Primary Key:</u> contact_id
supplier	<ul style="list-style-type: none"> • supplier_id – INT *NOT NULL *AUTO_INCREMENT • contact_id – INT *NOT NULL <u>Primary Key:</u> supplier_id
contractor	<ul style="list-style-type: none"> • contractor_id – INT *NOT NULL *AUTO_INCREMENT • contact_id – INT *NOT NULL <u>Primary Key:</u> employee_id
user	<ul style="list-style-type: none"> • user_id – INT *NOT NULL *AUTO_INCREMENT • username – VARCHAR(64) • password_hash_value – CHAR(64) • access – INT NOT NULL • google_client_id – VARCHAR(45) • d_type - VARCHAR(10) <u>Primary Key:</u> user_id
contact_user	<ul style="list-style-type: none"> • contact_user_id - INT *NOT NULL *AUTO_INCREMENT • user_id - INT *NOT NULL • contact_id - INT *NOT NULL <u>Primary Key:</u> contact_user_id

<i>employee_user</i>	<ul style="list-style-type: none"> • <i>employee_user_id</i> - INT *NOT NULL *AUTO_INCREMENT • user_id - INT NOT NULL • employee_id - INT NOT NULL <u>Primary Key:</u> <i>employee_user_id</i>
<i>address</i>	<ul style="list-style-type: none"> • <i>address_id</i> – INT *NOT NULL *AUTO_INCREMENT • <i>address</i> – VARCHAR (80) • <i>zipcode</i> - VARCHAR(20) • <i>city</i> - VARCHAR(64)\ <u>Primary Key:</u> <i>address_id</i>
<i>phone_number</i>	<ul style="list-style-type: none"> • <i>phone_number_id</i> – INT *NOT NULL *AUTO_INCREMENT • <i>phone_number</i> – VARCHAR(20) • <i>type</i> – VARCHAR(15) <u>Primary Key:</u> <i>phone_number_id</i>
<i>email</i>	<ul style="list-style-type: none"> • <i>email_id</i> – INT *NOT NULL *AUTO_INCREMENT • <i>email</i> – VARCHAR(255) <u>Primary Key:</u> <i>email_id</i>
<i>employee_address</i>	<ul style="list-style-type: none"> • <i>employee_address_id</i> – INT *NOT NULL *AUTO_INCREMENT • employee_id – INT *NOT NULL • address_id – INT *NOT NULL • <i>Priority</i> – INT(10) <u>Primary Key:</u> <i>employee_address_id</i>
<i>employee_phone_number</i>	<ul style="list-style-type: none"> • <i>employee_phone_number_id</i> – INT *NOT NULL *AUTO_INCREMENT • employee_id – INT *NOT NULL • phone_number_id – INT *NOT NULL • <i>priority</i> – INT (10) <u>Primary Key:</u> <i>employee_phone_number_id</i>
<i>employee_email</i>	<ul style="list-style-type: none"> • <i>employee_email_id</i> – INT *NOT NULL *AUTO_INCREMENT • employee_id – INT *NOT NULL • email_id – INT *NOT NULL • <i>priority</i> – INT(10) <u>Primary Key:</u> <i>employee_email_id</i>
<i>contact_address</i>	<ul style="list-style-type: none"> • <i>contact_address_id</i> – INT *NOT NULL *AUTO_INCREMENT • contact_id – INT *NOT NULL • address_id – INT *NOT NULL • <i>priority</i> – INT(10) <u>Primary Key:</u> <i>contact_address_id</i>
<i>contact_phone_number</i>	<ul style="list-style-type: none"> • <i>contact_phone_number_id</i> – INT *NOT NULL *AUTO_INCREMENT • contact_id – INT *NOT NULL • phone_number_id – INT *NOT NULL • <i>priority</i> – INT(10) <u>Primary Key:</u> <i>contact_phone_number_id</i>
<i>contact_email</i>	<ul style="list-style-type: none"> • <i>contact_email_id</i> – INT *NOT NULL *AUTO_INCREMENT • contact_id – INT *NOT NULL • email_id – INT *NOT NULL • <i>priority</i> – INT(10) <u>Primary Key:</u> <i>contact_email_id</i>

entry	<ul style="list-style-type: none"> • entry_id – INT *NOT NULL *AUTO_INCREMENT • employee_id - INT *NOT NULL • organization_id - INT *NOT NULL • title – VARCHAR(75) • date_created – DATE • description – VARCHAR(4096) • d_type – VARCHAR(20) <u>Primary Key:</u> entry_id
entry_modification	<ul style="list-style-type: none"> • entry_modification_id: – INT *NOT NULL *AUTO_INCREMENT • employee_user_id – INT *NOT NULL • entry_id – INT *NOT NULL • time_stamp – DATETIME <u>Primary Key:</u> entry_modification_id
work_order	<ul style="list-style-type: none"> • work_order_id – INT *NOT NULL *AUTO_INCREMENT • entry_id – INT *NOT NULL • status – VARCHAR(64) • completion_date – Date() *NULLABLE <u>Primary Key:</u> work_order_id
purchase_order	<ul style="list-style-type: none"> • purchase_order_id – INT *NOT NULL *AUTO_INCREMENT • entry_id – INT *NOT NULL • status – VARCHAR(64) <u>Primary Key:</u> purchase_order_id
pdf	<ul style="list-style-type: none"> • pdf_id – INT *NOT NULL *AUTO_INCREMENT • entry_id – INT *NOT NULL • pdf_data – VARCHAR(34) <u>Primary Key:</u> pdf_id
supplies	<ul style="list-style-type: none"> • supplies_id – INT *NOT NULL *AUTO_INCREMENT • supplier_id – INT *NOT NULL • purchase_order_id – INT *NOT NULL • arrival_date – DATE <u>Primary Key:</u> supplies_id
items_supplied	<ul style="list-style-type: none"> • items_supplied_id – INT *NOT NULL *AUTO_INCREMENT • supplies_id – INT *NOT NULL • item_name – VARCHAR(256) • price – INT • currency_type – VARCHAR(64) • number_of_items_sold INT <u>Primary Key:</u> items_supplied_id
in_house_performs	<ul style="list-style-type: none"> • in_house_performs_id – INT *NOT NULL *AUTO_INCREMENT • employee_id - INT *NOT NULL • work_order_id - INT *NOT NULL • hours_worked – INT • pay_rate – INT • currency_type – VARCHAR(20) <u>Primary Key:</u> in_house_performs_id
out_house_performs	<ul style="list-style-type: none"> • out_house_performs_id – INT *NOT NULL *AUTO_INCREMENT • contractor_id - INT *NOT NULL • work_order_id - INT *NOT NULL • cost – INT • currency_type – VARCHAR(20) <u>Primary Key:</u> out_house_performs_id

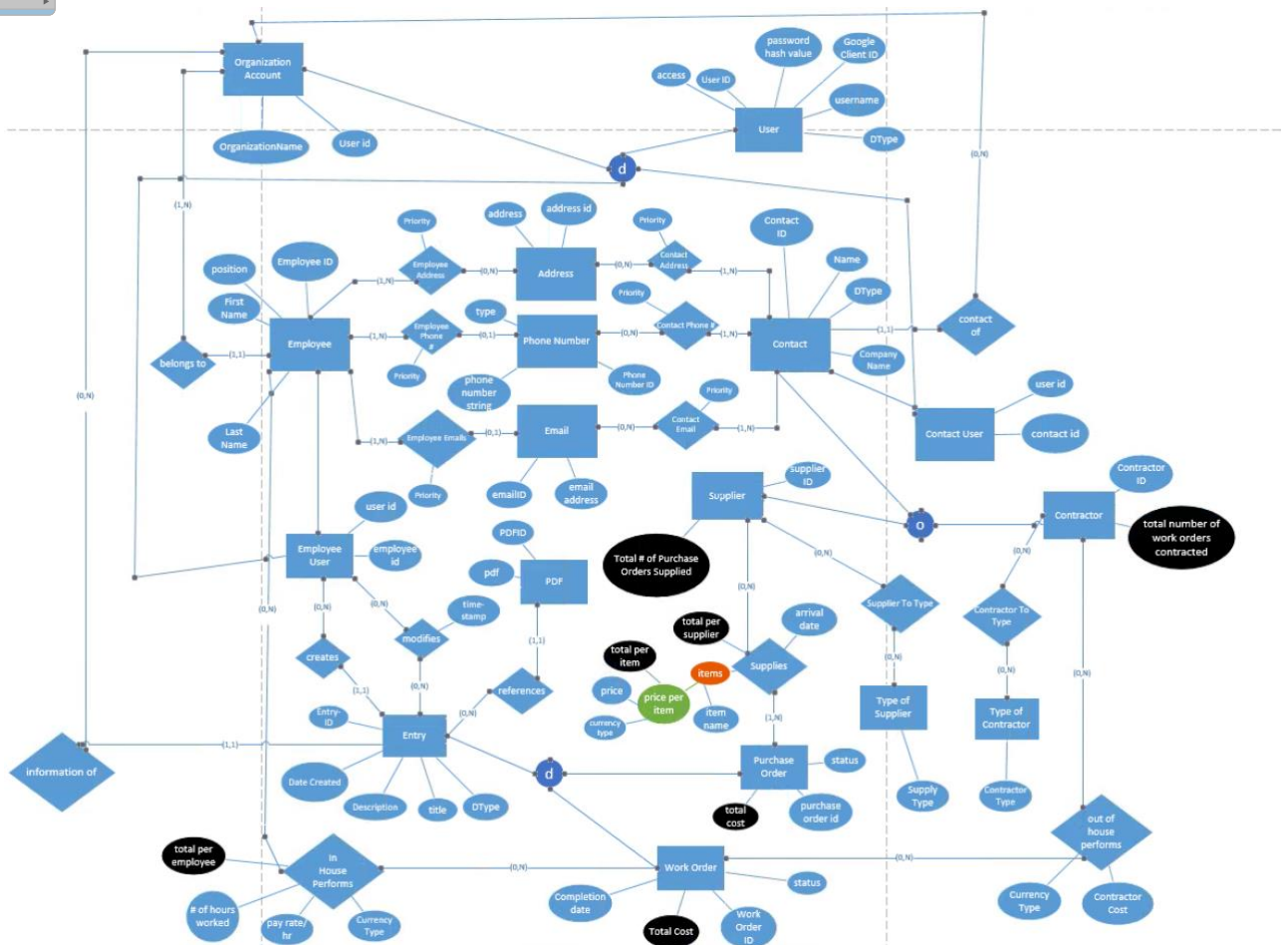
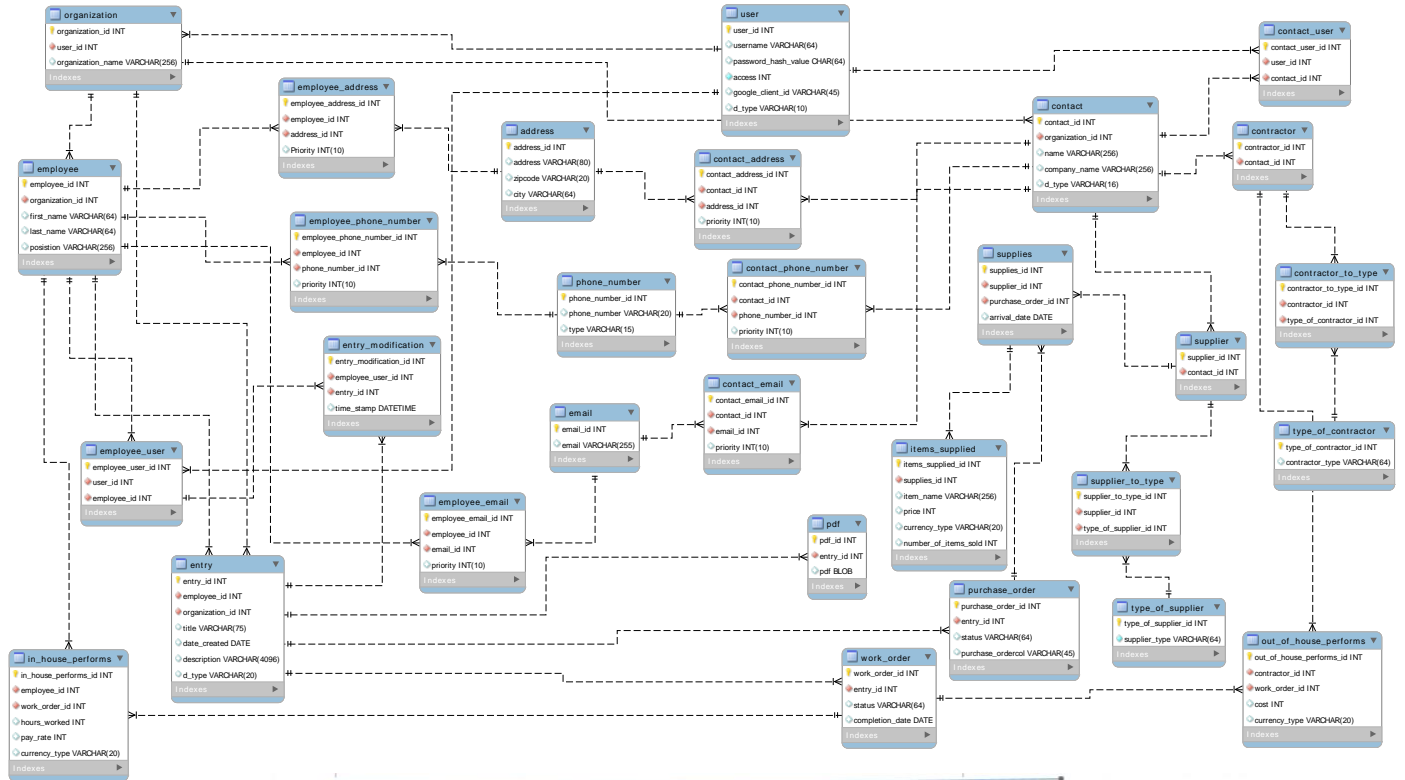
<i>type_of_supplier</i>	<ul style="list-style-type: none"> • <i>type_of_supplier_id</i> - INT *NOT NULL *AUTO_INCREMENT • <i>supplier_type</i> - VARCHAR(64) <i>Primary Key: type_of_supplier_id</i>
<i>supplier_to_type</i>	<ul style="list-style-type: none"> • <i>supplier_to_type_id</i> - INT *NOT NULL *AUTO_INCREMENT • <i>supplier_id</i> - INT *NOT NULL • <i>type_of_supplier_id</i> - INT *NOT NULL <i>Primary Key: supplier_to_type_id</i>
<i>type_of_contractor</i>	<ul style="list-style-type: none"> • <i>type_of_contractor_id</i> - INT *NOT NULL *AUTO_INCREMENT • <i>contractor_type</i> - VARCHAR(64) <i>Primary Key: type_of_contractor_id</i>
<i>contractor_to_type</i>	<ul style="list-style-type: none"> • <i>contractor_to_type_id</i> - INT *NOT NULL *AUTO_INCREMENT • <i>contractor_id</i> - INT *NOT NULL • <i>type_of_contractor_id</i> - INT *NOT NULL <i>Primary Key: contractor_to_type_id</i>

Relationships between entities outlined in this database include:

- *Employee Address*: (N:M) Every employee has at least 1 and at most 'N' addresses and every address is associated with at least 0 and at most 'N' employees. Relationship is accomplished through a bridging table and includes the primary keys from EMPLOYEE and ADDRESS tables as foreign keys in the new table EMPLOYEE ADDRESS.
- *Employee Phone Number*: (1:N) Every employee has at least 1 and at most 'N' phone numbers and every phone number is associated with at least 0 and at most '1' employee. Relationship is accomplished through a bridging table and includes the primary keys from EMPLOYEE and PHONE NUMBER tables as foreign keys in the new table EMPLOYEE PHONE NUMBER.
- *Employee Email*: (1:N) Every employee has at least 1 and at most 'N' emails and every email is associated with at least 0 and at most '1' employee. Relationship is accomplished through a bridging table and includes the primary keys from EMPLOYEE and EMAIL tables as foreign keys in the new table EMPLOYEE EMAIL.
- *Contact Address*: (N:M) Every contact has at least 1 and at most 'N' addresses and every address is associated with at least 0 and at most 'N' contacts. Relationship is accomplished through a bridging table and includes the primary keys from CONTACT and ADDRESS tables as foreign keys in the new table CONTACT ADDRESS.
- *Contact Phone Number*: (N:M) Every contact has at least 1 and at most 'N' phone numbers and every phone number is associated with at least 0 and at most 'N' contacts. Relationship is accomplished through a bridging table and includes the primary keys from CONTACT and PHONE NUMBER as foreign keys in the new table CONTACT PHONE NUMBER.
- *Contact Email*: (N:M) Every contact has at least 1 and at most 'N' emails and every email is associated with at least 0 and at most 'N' contacts. Relationship is accomplished through a bridging table and includes the primary keys from CONTACT and EMAIL tables as foreign keys in the new table CONTACT EMAILS.
- *Entry Modification*: (N:M). A user can modify at least 0 and at most 'N' entries and an entry can be modified by at least 0 and at most 'N' employees. Relationship is accomplished through a bridging table and includes the primary keys from USER and ENTRY tables as foreign keys in the new table ENTRY MODIFICATION.

- Entry Creation: (1:N) A user can create at least 0 or at most 'N' entries and an entry can be only be created by 1 and only 1 user. Relationship is accomplished by using the primary key of USER (UserID) as a foreign key in ENTRY table.
- Belongs To: (1:N) An employee must belong to one and only one organization but an organization must have at least one and up to 'N' employees that belong to it.
- Contact Of: (1:N) A contact must be a contact of one and only one organization but an organization can have at least zero and up to 'N' contacts associated with it.
- Information Of: (1:N) An entry must have one and only one organization associated with its creation and an organization can have at least zero and up to 'N' entries associated with them
- References: (1:N) An entry can reference at least 0 and at most 'N' PDFs but a PDF must be associated with 1 and only 1 entry. This relationship is accomplished by using the primary key of ENTRY (EntryID) as a foreign key in the PDF table.
- Supplies: (N:M) A purchase order can be supplied by at least 1 or at most 'N' suppliers and a supplier can supply at least 0 or at most 'N' purchase orders. The relationship is accomplished through a bridging table by using the primary keys from the SUPPLIER and PURCHASE ORDER tables as foreign keys in the new table SUPPLIES.
- In-House Performs: (N:M) An employee can perform on at least 0 or at most 'N' work orders and a work order can be performed by at least 0 or at most 'N' employees. The relationship is accomplished through a bridging table by using the primary keys of the EMPLOYEE and WORK ORDER tables as foreign keys in the new table IN-HOUSE PERFORMS.
- Out-House Performs: (N:M) A contractor can perform on at least 0 or at most 'N' work orders and a work order can be performed by at least 0 or at most 'N' contractors. The relationship is accomplished through a bridging table by using the primary keys of the CONTRACTOR and WORK ORDER tables as foreign keys in the new table OUT-HOUSE PERFORMS.
- Supplier to Type: (N:M) A supplier can have at least zero or up to 'N' supply types that better distinguish and filter what the suppliers actually sell and a type of supplier can be associated with at least zero or up to 'N' suppliers.
- Contractor to Type: (N:M) A contractor can have at least zero or up to 'N' contractor types that better distinguish and filter what type of work the contractors actually do and a type of contractor can be associated with at least zero or up to 'N' contractors.

A further look into the design of the database can be seen through the following EER diagram. Described on the next page.



7 Roles and Responsibilities

******All members were actively involved in all three parts but specialization is as follows

Front End : Nathaniel DeSimone, Tyler Cockburn

Middle End: Christopher Hobson, Eric Lynn

Back End: Brennan Ringel, David Yeager, Nicholas Giordano

8 Terms and Definitions

AWS - Amazon Web Services

MySQL - My Structured Query Language

HTML - Hypertext Markup language

Bootstrap - Framework for webpages

jQuery - JavaScript library for simpler webpage interaction and REST calls

REST - Representational state transfer

Pip - "Pip installs packages" - A package installer/management tool for Python