CSCE 4925: Project Aero
Maintenance Manual

By: Alyssa Thurston, Breuna Riggins, James Sabetti, Travis Goral
Project Manager: Alyssa Thurston
Instructor: Professor Keathly
Client: Dan Minshew and Kyle Tayle, Denton Techmill
Date: 16 February 2018
Status: Final Draft

Table of Contents

Table of Contents **2**

Revisions **4**

1 Introduction **5**

1.1 Purpose 5

1.3 Scope 5

1.4 List of Definitions and Abbreviations 6

1.5 Overview 6

2 System Information **6**

2.1 System Architecture 6

2.2 Security 7

3 Environment **8**

3.1 Equipment 8

3.2 Storage Requirements 8

3.3 Software 8

4 System Maintenance **8**

4.1 Responsibilities 8

4.2 Error Conditions 9

4.3 Maintenance Procedures 9

4.3.1 Backing Up 9

4.3.4 Updates 10

4.3.5 Accessing the Server Remotely 10

5.0 Software Maintenance Procedures **11**

5.1 Consolidated Unit List 11

5.2 PostgreSQL Database 11

5.2.1 Description 11

5.2.2 Functions 11

5.2.3. Input 11

5.2.4 Processing 12

5.2.4.1 Initiation Procedures 12

5.2.4.2 Core Processing Procedures 12

5.2.4.3 Branching Conditions 12

5.2.4.4 Restrictions 12

5.2.4.5 Exit Requirements 12

5.2.4.6 Communications 12
5.2.4.7 Output 12
5.2.4.8 Unique Features 12
5.2.5 Data Structures 13
5.2.6 Verification Procedures 13
5.2.7 Listings 13
5.2.8 Interfaces 13
5.3 XAMPP 13
5.3.1 Description 13
5.3.2 Functions 13
5.3.3. Input 14
5.3.4 Processing 14
5.3.4.1 Initiation Procedures 14
5.3.4.2 Core Processing Procedures 14
5.3.4.3 Branching Conditions 14
5.3.4.4 Restrictions 14
5.3.4.5 Exit Requirements 14
5.3.4.6 Communications 14
5.3.4.7 Output 14
5.3.4.8 Unique Features 14
5.3.5 Data Structures 15
5.3.6 Verification Procedures 15
5.3.7 Listings 15
5.3.8 Interfaces 15

Revisions

Below is a list of any revisions made to this document.

Date	Description of Change Made	Person Making Change
2/12/2018	Document Created	Breuna, Alyssa, Travis, James
3/6/2018	Document Revision	Breuna, Alyssa, Travis, James
3/7/2018	Final Draft Edits	Breuna, Alyssa, Travis, James

1 Introduction

The air quality in the City of Denton is notoriously bad. It's not a surprising fact given that there are two universities and multiple major highways and roadways that run through the city's limits. Worse still, there is only a single air quality sensor within city limits, and the next closest one is almost 15 miles away from the city.

The goal of this project is to create a network of air quality sensors to measure the air quality in the city of Denton so that citizens and city personnel alike can monitor air quality and take appropriate actions. The data obtained from the sensors is to be made public, so that air quality trends can be tracked all around the city. With these requirements met, the network would provide secure, real-time, and reliable data on the air quality in the city of Denton.

With the aim of creating a robust and open source of air quality data for the city of Denton, the Open Denton group has set out to establish a network of air quality sensors to gather and compile that data. In order to help achieve this goal, our group has been tasked with the following:

- Create a web client to display data in a highly readable manner.
- Create an API that records and displays air quality data in a readable manner
- Implement a scalable database to hold and model recorded data
- Integrate sensors and controllers that will read and relay data to the database over the internet
- Create documentation that allows community members to add sensors to the network

This document will outline the maintenance manual with procedures and guidelines on how to maintain the system, software, and database which will be used by the client and the users in the future.

1.1 Purpose

The purpose of this document is to help the client and users have a better understanding of the technical requirements on maintaining the overall product of the air quality system which will help prolong the lifetime of the different operating systems, equipment, and structure.

1.3 Scope

This manual provides detailed guidelines and information on the maintenance of the overall system to ensure the longevity of air quality sensors , the different operating systems, structure, and equipment. This manual contains the following guidelines to guarantee:

- That the air quality system performance satisfies every requirements of the system

- That the air quality system is maintained properly
- That the performance of the air quality system is optimised

1.4 List of Definitions and Abbreviations

The following is a list of abbreviations and definitions to words that are used in this document;

- D.A.M.N - Denton Air Monitoring Network
- Node- Air Quality Sensor
- AQI- Air Quality Index
- Project Aero: This Project, which encapsulates the D.A.M.N.
- UFW: Uncomplicated Firewall; Explained in section 2.2

1.5 Overview

This manual serves as the general documentation on everything that needs to be done to keep the server functional. It also serves as the documentation on getting software put on the server, in addition to configuring that software.

The various sections of this document can be found in the table of contents.

2 System Information

This section provides an overview of what is involved in the system, and also includes a section regarding security concerns. Additionally, an overview of the communications network is provided as well.

2.1 System Architecture

This system is comprised of a front end, database, web server, and API. The API is responsible for communication information from the Web Server and Database to the front end and vice versa. Additionally, the API will take in information from the Nodes and send it to the server as necessary.

The front-end is on the client's side, and really won't have much to do with us. This front end is simply the internet browser of the user's choice. We are going to support the following browsers:

- Internet Explorer
- Safari
- Mozilla Firefox
- Google Chrome
- Opera
- Microsoft Edge
- Default Browser of 3 Android Phones
- Safari on iPhone
- Google Chrome on iPhone

The database, web server, and API all reside on the server. The server that we are running is Windows Server 2016, with all available updates installed. The server has the following installed on it:

- XAMPP
- Python Library
- MySQL
- PostgreSQL
- JQuery
- Anaconda
- Firefox
- Teamviewer 13

The database is running a PostgreSQL database that presents the data in a time-series format which is ideal for a project such as ours that is designed to take any input and show it in a user friendly matter.

Our web server component is utilizing Apache. Apache is a common, powerful web server. This is what is used to push the HTML based front end out to the internet. This front end is what displays all of the applicable data from the website.

The last component in our system is the API. The API is responsible for communicating with the database and web server, and getting the data to and from the client. Additionally, our API will also be responsible for communicating with the nodes that are put onto our network.

2.2 Security

The main line of security in our system is the firewall. For our firewall, we are utilizing Windows Firewall which comes with the standard Windows Server 2016 installation Some of the technologies that are being utilized require ports to be open and exposed. These include:

- 22
- 25
- 80
- 443
- 5091

3 Environment

This section includes information on the environment that is used on this system. It will include the storage, software, and hardware requirements, in addition to versions of software and other useful information.

3.1 Equipment

There are few requirements for equipment. In order for the system to work properly, all that is needed is a server, and the client side requires a device with an internet browser.

In order to get data, a node has to be connected to the system. This node can be purchased online, or can be one made from a Raspberry Pi.

3.2 Storage Requirements

Client-Side, there are no storage requirements as the system is cloud based, and thus, is not stored locally. Server-side, 50GB is recommended for full system functionality and performance, and additional space is necessary to store data sent from the users.

3.3 Software

For the client side of things, we support the following operating systems or devices:

- Windows 7
- Windows 8 / 8.1
- Windows 10
- MacOS
- iOS
- Android

4 System Maintenance

Detailed below are the maintenance procedures for the system. Included are procedures for updates, back-ups, any encountered errors, and responsibilities of different people.

4.1 Responsibilities

Responsibility for system maintenance may be covered by the development team for a period no longer than 90 days after end of 2018 Spring semester. After this point, responsibility for system maintenance is left to owner's discretion. No more than one person may be needed for routine maintenance issues.

4.2 Error Conditions

The most likely error codes one may encounter would be common HTTP Status Messages if the database container or apache server was down. Listed below are common HTTP error codes and their respective explanations:

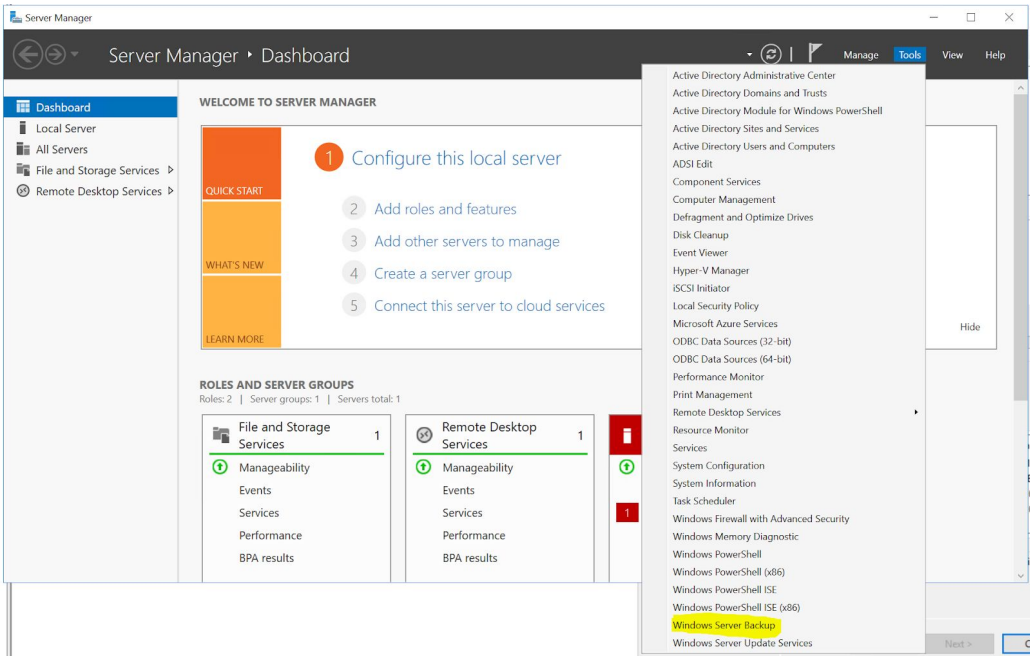
404: The page cannot be found
500: Internal Server Error: Generic error
502: Bad Gateway: Invalid response from upstream server
503: Service Unavailable: Server is overloaded or down
504: Gateway Timeout: Did not receive timely response from upstream server.

4.3 Maintenance Procedures

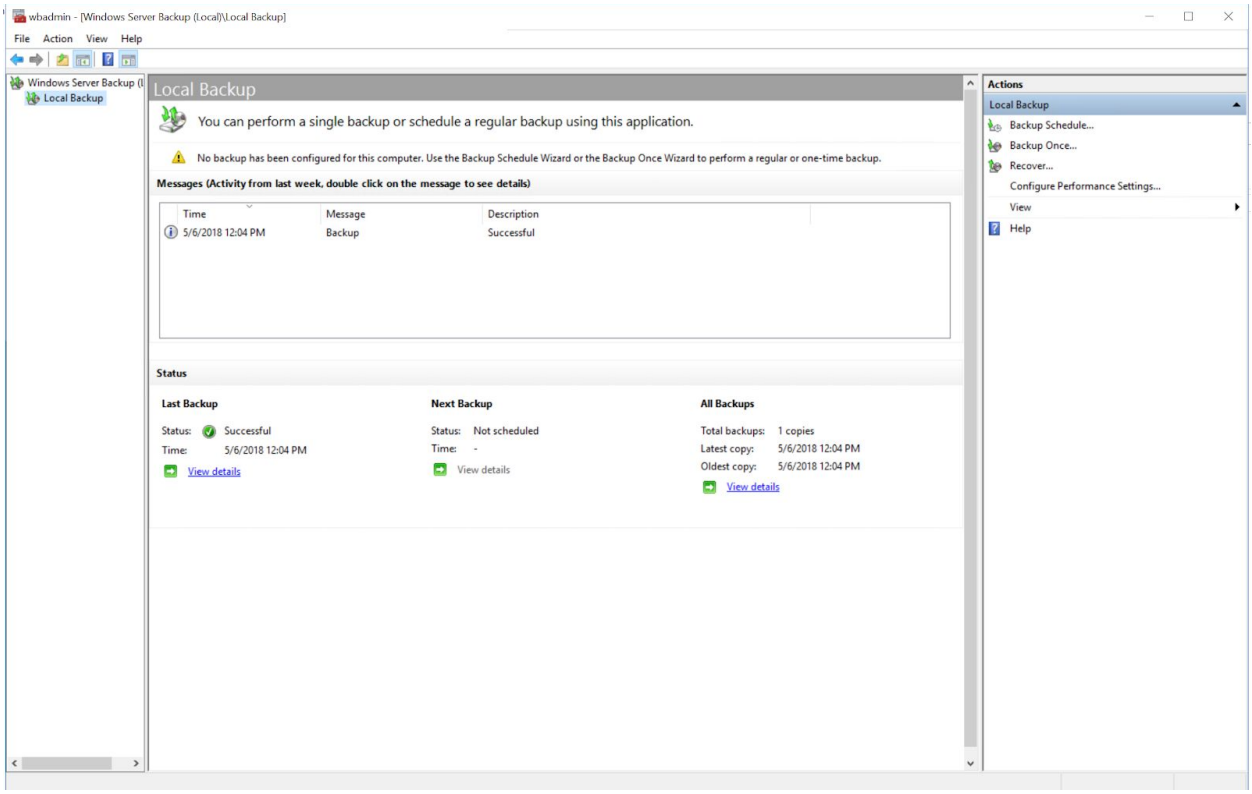
Listed below are procedures for backups, updates, and other miscellaneous maintenance items.

4.3.1 Backing Up

Backups are scheduled and configured via the Windows Server Manager too. As a prerequisite, Windows Server Backup needs to be installed in Server Manager > Add roles and features > Next > Role-based or feature-based installation > Next > Next > Next > Select "Windows Server Backup" and Install.



To create a new backup, go to the Server manager and select Tools > Windows Server Backup From here, you can schedule backups, create a system image, and restore from a system image. To create a system image click Backup Once under the Actions menu and follow the setup. A Full Metal backup will create a system image with all of the current programs and settings preserved. To restore from this image, choose the Recover option under Actions and select your system image file.



4.3.4 Updates

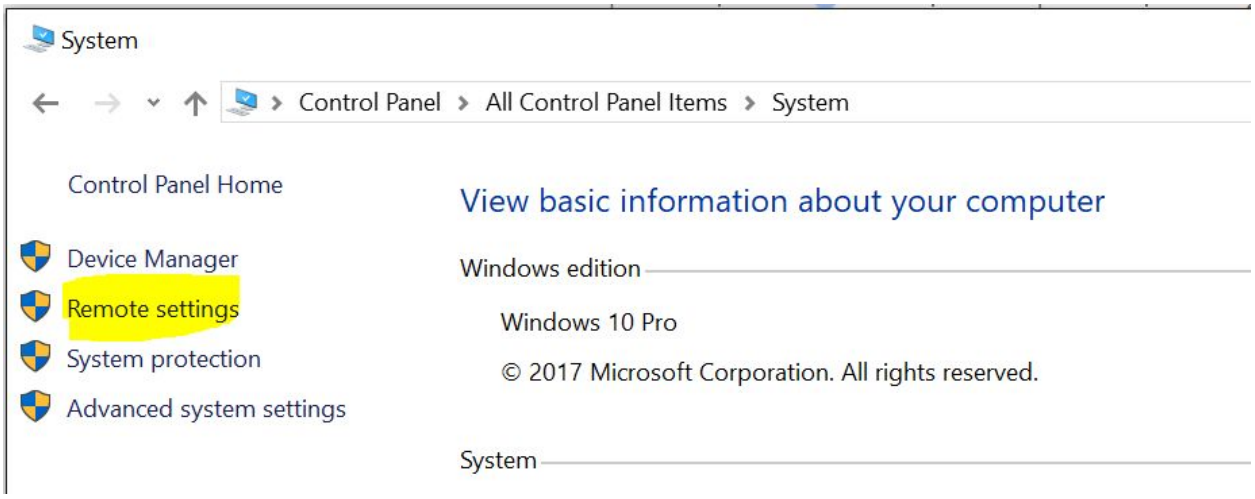
Occasional updates may be desirable to keep packages working and optimal. To do this, log into the server and Start > Settings > Update & Security. Here you can schedule updates or install them manually. It is recommended to install updates at a time when the server is experiencing the least demand.

4.3.5 Accessing the Server Remotely

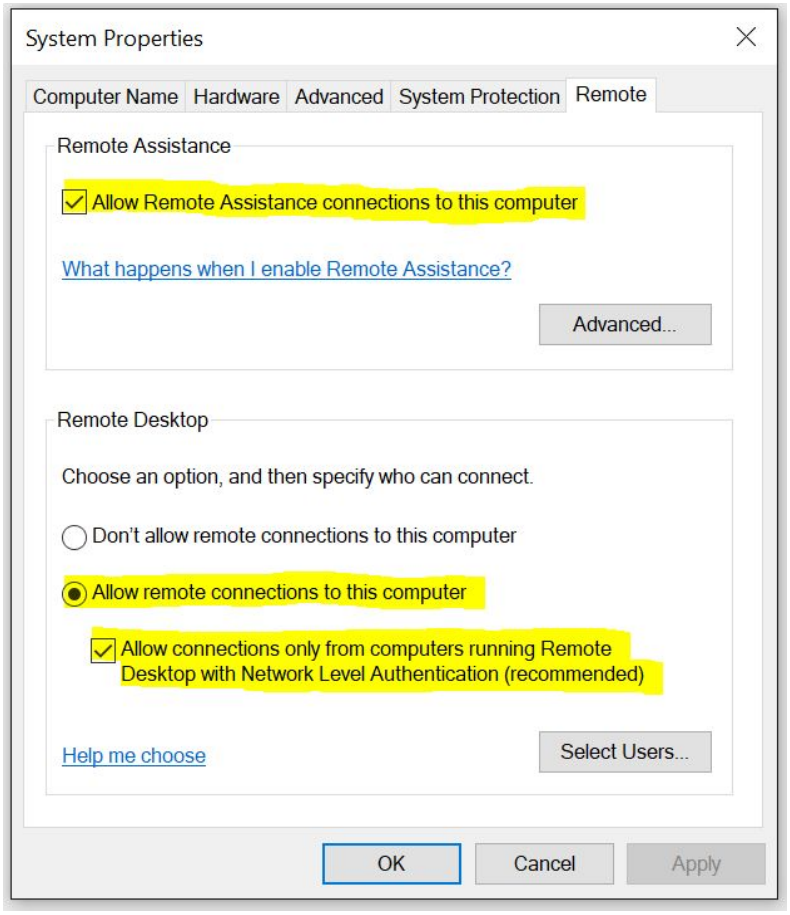
Windows Remote Desktop Connection is required to connect to the server. To run RDP from a Windows computer, press the Win + R key and type “mstsc.exe” and click Run.

To find the right IP address, go to the browser on the server and type “What is my IP Address,” it should spit it out to you. Type this address in and click connect. You will be asked to provide credentials upon login.

If Remote Desktop is not already configured on both the Server and Client computers, it will need to be enabled in order to connect to the server from a remote client. To enable Remote Desktop on Windows, go to the Control Panel, select Large icons under the View by: menu, then select System. From here, select Remote settings from the top-left.



Make sure all the following options are selected on both the client and server computers so that you can use Remote Desktop connections:



Depending on your router settings, you may also need to allow the default RDP port 3389 in order to allow Remote Desktop connections. If problems still persist, check your firewall settings for any deny rules that would forbid connections on port 3389.

5.0 Software Maintenance Procedures

Listed below are detailed descriptions for any maintenance procedures the software units may need to go under

5.1 Consolidated Unit List

5.2 PostgresQL Database
5.3 XAMPP

5.2 PostgreSQL Database

This is the database we are using to house all of the recorded information sent to us by the sensors. This will be available on a remote server via a virtual machine.

5.2.1 Description

The PostgreSQL Database will serve as one of the major focal points for our system. It will house all recorded information, sent to us by the sensors in the field, and will relay that information, via an API, to our front end website. Our database runs with the standard SQL language in place and is designed for ease of use.

In terms of its relation to other softwares, PostgreSQL will run in conjunction with the API in order to deliver the necessary data to the front end, which will be hosted by the Apache2 Server and displayed on the front end.

5.2.2 Functions

The SQL database will be housing the records for all of the sensors and then sending this information, as needed, to the front-end website.

5.2.3. Input

Data input into the SQL database will come as a series of numbers, all with predetermined attributes attached to them, from the sensors in the field. Depending on the sensor, there will be a different magnitude of numbers coming into the database, and the order in which they arrive, as well as their attributes, will determine where they go within the database.

In review, as previously stated, the SQL database will take in a series of numbers sent by the sensors within the field. These numbers will be stored in the database and organized there for use by the LocalHost via the API.

5.2.4 Processing

5.2.4.1 Initiation Procedures

The PostgreSQL database should be up and running upon startup and requires no initiation on the user's part. Configuration and administration is performed via the pgAdmin application located on the desktop

5.2.4.2 Core Processing Procedures

The SQL database is designed to intake numerical and character values and assign them to certain categorical attributes based on the design of the sensor inputting the information to the system.

5.2.4.3 Branching Conditions

The SQL database has no branching conditions.

5.2.4.4 Restrictions

As of writing this, the SQL database has the ability to intake any kind of value, whether it be numerical, character, timestamp, string, etc. Values and be dependent on the sensor reporting the information in order to attribute the categories for the values. The SQL database also does not have a filter gap in place to prevent incorrect on inaccurate numbers from being reported.

5.2.4.5 Exit Requirements

If the SQL database ever needs to be stopped, for whatever reason, open the task manager and locate the PostgreSQL process and end it.

Do not forget to start the database up again whenever the issue has been resolved.

5.2.4.6 Communications

The SQL database will communicate with the front-end via queries run by php files that should output the information to a chart. As of today, there is no chart functionality however data can be displayed correctly on front-end.

5.2.4.7 Output

The SQL database will simply output the information requested from it via a query from the php code.

5.2.4.8 Unique Features

As of the writing of this manual, the SQL has no unique features.

5.2.5 Data Structures

As per the EPA guidelines, our sensors will report any of the following information back to the database and categorize them based on the attributes applicable:

- Carbon Monoxide
- Lead
- Nitrogen Dioxide
- Ozone
- Particle Pollution
- Sulfur Dioxide
- Location (address or longitude or latitude coordinates)
- Time
- Identifying number/name for the sensor

5.2.6 Verification Procedures

A user will need to have an account with our front end in order to have their information reported to the database in order to ensure responsibility and accuracy for all information reported.

5.2.7 Listings

SQL utilizes the standard software listings.

5.2.8 Interfaces

The SQL database will not necessarily graphically interface with any other software, but it will utilize an API connection in order to communicate back and forth with the front end and display the relevant information.

5.3 XAMPP

This is what we will be using to design out front end and make sure that everything is working properly.

5.3.1 Description

XAMPP is the platform that the front end of the website will be broadcasted from.

5.3.2 Functions

XAMPP is the Apache distribution that our website will be hosted from. It is run locally on the server itself and is currently using the projectaerodenton.com domain

5.3.3. Input

All inputs will come from the html, php, or javascript files stored within the Public Documents folder. These files can be altered and changes will be reflected on the projectaerodenton website.

5.3.4 Processing

5.3.4.1 Initiation Procedures

Once an html or javascript file is ready to be a webpage, simply upload or save it to the Public Documents folder and if necessary, restart the Apache service from the XAMPP control panel.

5.3.4.2 Core Processing Procedures

All information input into the website will be via an html page of a javascript page.

5.3.4.3 Branching Conditions

The webpage has no branching conditions.

5.3.4.4 Restrictions

As of writing this, the webpage is restricted based on its handling capabilities of both html and javascript.

5.3.4.5 Exit Requirements

If the webpage ever needs to be stopped, for whatever reason, simply stop the Apache process from the XAMPP control panel

Do not forget to start Apache again whenever the issue has been resolved.

5.3.4.6 Communications

The webpage will communicate with the PostgreSQL database via queries run by javascript and php files.

5.3.4.7 Output

The webpage will display the information within the html files in the Public Documents folder.

5.3.4.8 Unique Features

As of the writing of this manual, the webpage no unique features.

5.3.5 Data Structures

All data stored within the webpage can be found in the Public Documents files and folders of html, conf, or javascript file extensions.

5.3.6 Verification Procedures

The only way to edit any of the information within the webpage is to go into the server's Public Documents and edit the files from there.

5.3.7 Listings

The webpage will follow the basic software listings.

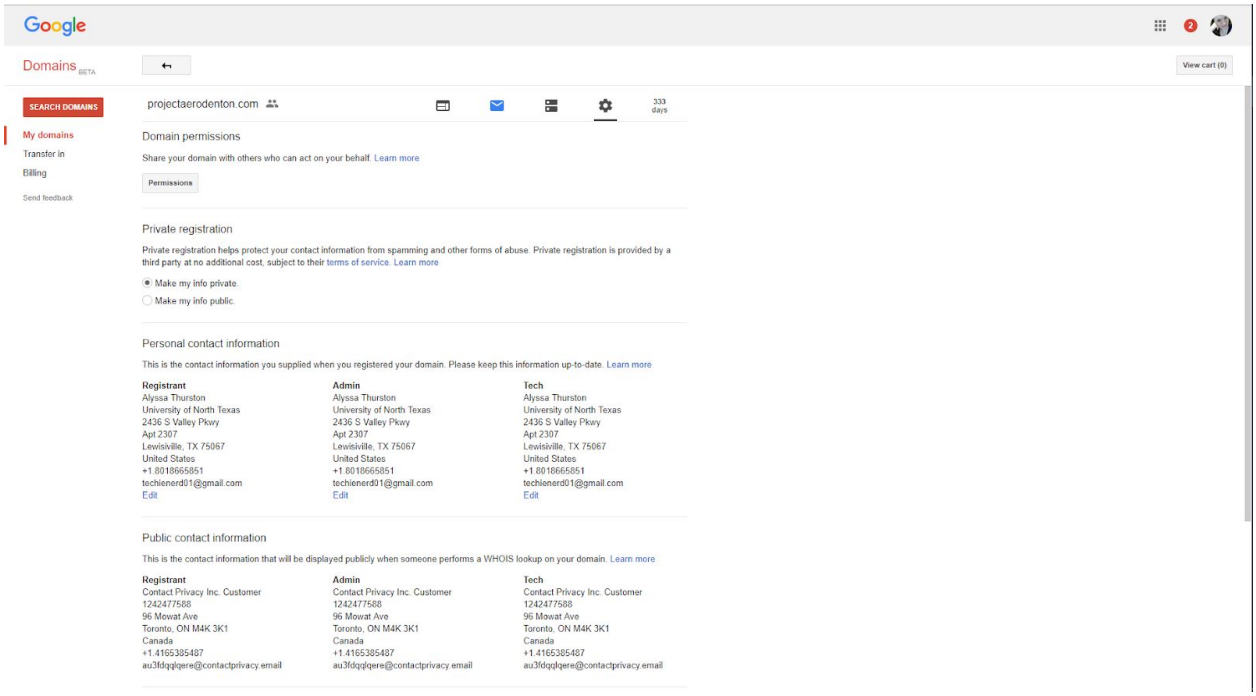
5.3.8 Interfaces

The webpage files can be edited via a text editor such as Notepad++ within the server desktop itself.

5.4 Google DNS Information

Before delving into the depths that you need to know about Google DNS, it's important to first understand what DNS is. DNS, Domain Name Service, is a service that says "this IP address (192.168.1.1 for example) belongs to this name (www.google.com, or google.com, for example)." We have purchased the name "projectaerodenton.com" for a year. In order for this to work, it is important that DNS service is set up properly.

First, go to domains.google.com/registrar, and click "settings" next to projectaerodenton.com. This should bring you to a screen similar to the following:



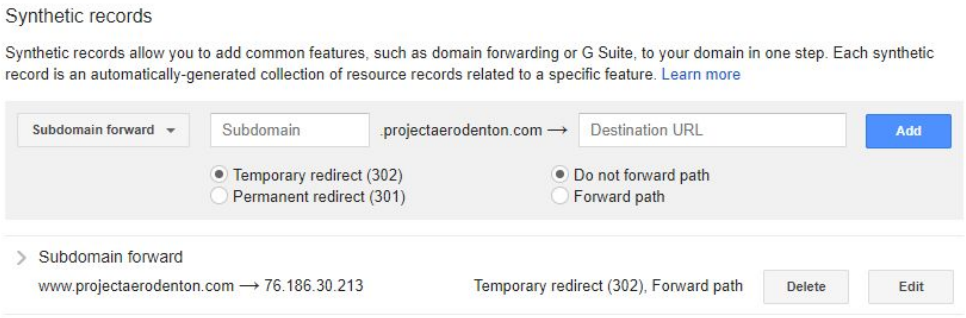
As a basic walk-through of the most important things here:

- If you click the Permissions button, you can share the domain administration with other people. At the time of this writing, we have shared them with Dan Minshew and Kyle Taylor. No other people will have access to these domain settings.
- We recommend keeping the “Make my info private” setting turned on. This way, you don’t have the people of the internet showing up to your doorstep.
- Change the personal contact information to match your own information. Alyssa has included hers there as a placeholder.
- The public information is only ever displayed if someone performs a WHOIS lookup on your domain. To get your information, they’d have to go through the company listed there.

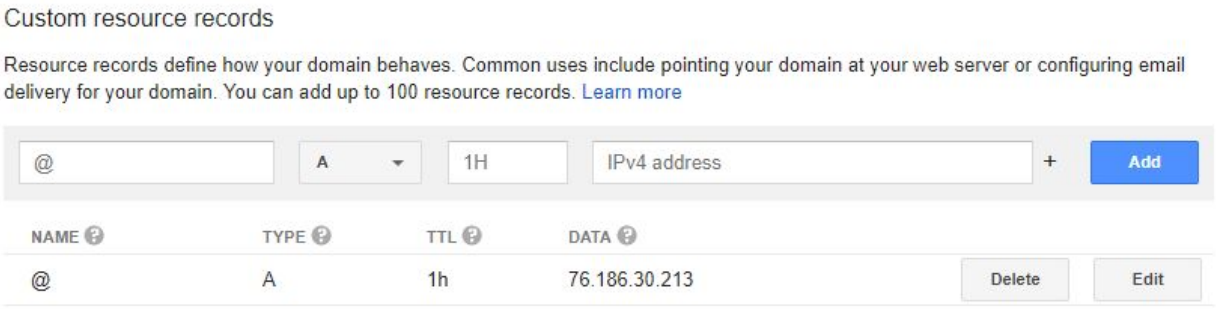
The next essential screen is the “Configure DNS” page, from the settings screen, you can access that by clicking the icon to the left of the settings gear, it looks like two rectangles stacked on top of each other. There is only two settings that you’ll want to edit or change here:

- Synthetic records: These allow for you to put the “www” in front of “projectaerodenton.com” and will still direct you to the right page. We have configured “www.projectaerodenton.com” to forward to our server. When your own server is up and

running, you'll need to change the IP Address shown here to match your own.



- Custom resouce records: Simply put, these point your domain to the address that they need to map to. Here, we have set the domain to point to the IP Address of our server. Ensure that you change this when you get your own server setup.



One last cool feature that you can set up is your own custom e-mail forwarding, which is done by clicking the envelope to the left of the "configure DNS icon." If you wanted to create an email such as "admin@projectaerodenton.com" you would type "admin" in the first box, then type the email address of where you would want emails sent to "admin@projectaerodenton.com" to go. This feature doesn't create an email account, per se, but will forward all emails sent to a specified address to the email listed here.

Billing information is provided on the last icon, which shows the number of days before registration of your domain expires. You can turn on auto-renewal here, or add years. Transfer out is the process you go through to transfer ownership of your domain to someone else.