**IT - IT Service Monitor**

REC 2017

November 5th, 2017

Jennifer Herasymuik

Jonathon Wells

Quinn Bast

Shawn Clake

[**1 Introduction**](#_dw63wtbmlfk3) **3**

[1.1 Scope](#_uwlc7jdtma99) 3

[1.2 External Requirements](#_1pc8o73ppweb) 3

[1.3 Internal Requirements](#_83pnzi6y6xs) 3

[1.4 Design Issues](#_1rxhlzee8jsb) 3

[**2 Internal Design**](#_jsqd4i5r41jh) **4**

[2.1 Languages and Tools](#_eude9vdhjd80) 4

[2.2 Data Storage](#_ryd6s7ppcuq0) 4

[2.3 Application Pages](#_ak980fii20p) 6

[**3 Additional Information**](#_2f7hhq3azl04) **7**

[3.1 Design Methodology](#_gmxirsxg7bi6) 7

[3.2 Improvements](#_r03e0g76upic) 7

[**4 Bugs**](#_t4udj765c2ty) **7**

## 1 Introduction

IT has a multitude of different platforms and monitoring apps to monitor their systems. With the amount of services they manage, it can get cumbersome to manage multiple different service using multiple different monitoring tools. Instead, using a single app to monitor all of the company's services will make it easier on the IT staff, and should result in better response times, so the customer isn’t kept waiting.

This app will be available to IT staff in the company. They receive an username and password, then are able to login to the dashboard. From here, they are able to monitor the service status, issue issues and resolution, and customer information. This will make it easier for IT staff to keep track of all of the service issues, and prevent mix ups between IT staff.

### 1.1 Scope

The application features a live monitoring tool which polls a set of services using an ICMP ping packet. The application then stores whether the ping is successful or not as well as the time in which the polling occurred. If a ping fails, the system will automatically create an issue and assign an IT staff member.

Employees in IT will be assigned a username and password from their manager. They use this information to login and use the dashboard. From the dashboard, each employee is able to view an overview of their service issues. As issues come in, available staff will be assigned to a specific issue and remain in charge of the issue until it is resolved. Once the issue has been resolved. This document outlines the design, tools, and possible bugs in the web application.

### 1.2 External Requirements

1. Login to web application
2. View the dashboard and all current issues
3. Select an issue to view in more detail
4. Sign off on issue to indicate it is resolved
5. View data in user-friendly UI

### 1.3 Internal Requirements

1. Update the issues as they appear
2. Record when issue is solved and update table
3. Display issues to page in order
4. Assign free employee to issue
5. Poll services at a regular interval to determine status

### 1.4 Design Issues

1. Whether employee is available to take issue or not
2. Whether issue is resolved or not, should only have one unresolved issue per service
3. DIsplaying all of the issues onto the dashboard

## 2 Internal Design

The problem was solved using a web-based application. This application runs off of an Ubuntu server located on campus. The web application is run using the domain: <https://shawnclake.ca> through any web browser. The tools used to create this web application were all tools used to create basic web pages, including the LAMP (Linux Apache MySQL PHP) stack. All specific tools, as well as the internal design specifics for the project are outlined below.

### 

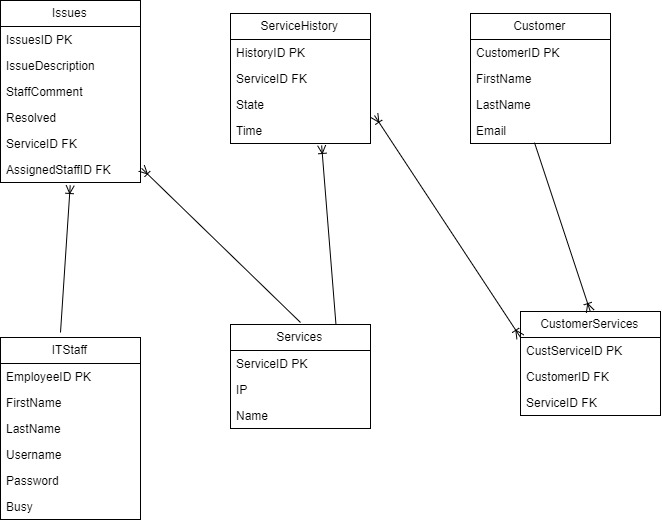
### 2.1 Languages and Tools

For the front end of the of the project, we are using HTML, CSS, and JavaScript. These are the basic languages the team understands and are used to make many web applications. To help with the styling of the application, the application uses Bootstrap. Bootstrap allows the application to automatically adjust the UI depending on the device or window size. This allows it to adjust the navigation bar, tables, and other features to reduced sizes when looking on a mobile device, allowing for simpler navigation and making it easier to read. This is an application of responsive design. The backend was created primarily using PHP.

### 2.2 Data Storage

The data was stored into tables. To create and/or update these tables either PhpMyAdmin or straight MySQL queries We ended up creating six tables along with one view.

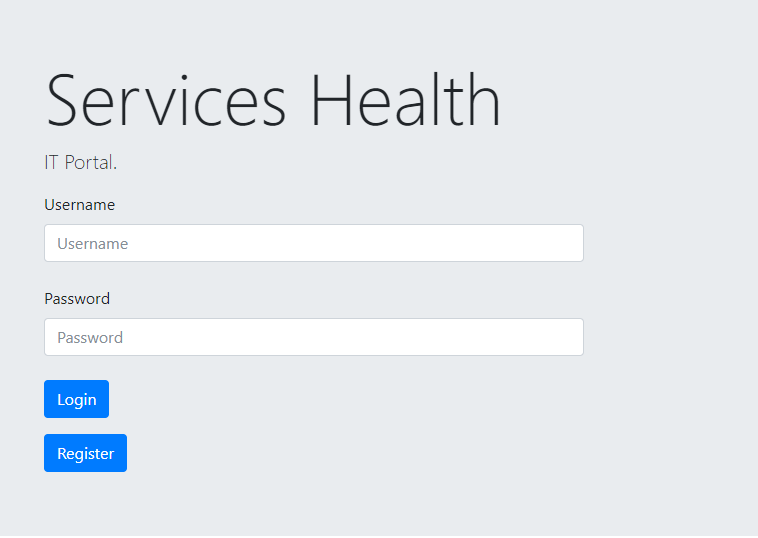
We have an ITStaff table which has the IDs and other information of every employee. This table also contains the ‘Busy’ column to determine whether the employee is available to take an issue or not. There is a Customer table, which holds all information about the IT customers. There is a Services table, which outlines all services available along with their IP address so we have an address to Ping. The CustomerServices table ties together the Customer and Services tables, showing what services are owned by which customer. The last two tables are the Issues table and the Service History table. The Issues table outlines all issues that are either resolved or unresolved. It shows which server has the issue, a description, staff assigned to the issue, possible IT staff comments, and whether that issue has been resolved or not yet. The Service History table shows all instances of the current services and their state at that time. This table is updated every minute currently, which isn’t optimal but met the project requirements. The relationships between the tables can be viewed in Figure 2 below:



**Figure 2: Entity Relationship Diagram**

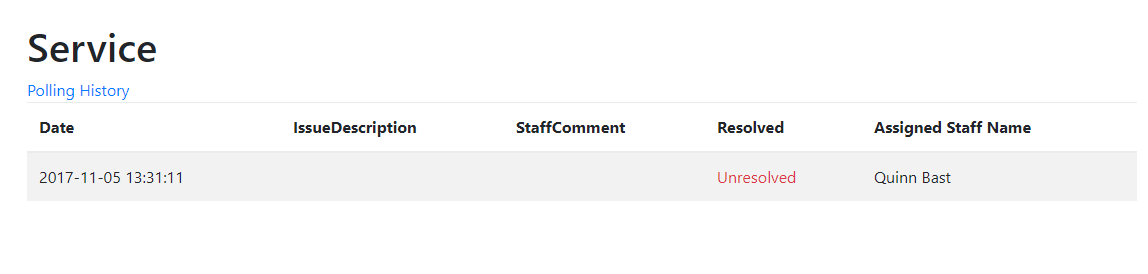
Along with our tables we had a view. This view is called the IP view, which only returns all of the IPs from the services table.

### 2.3 Application Pages



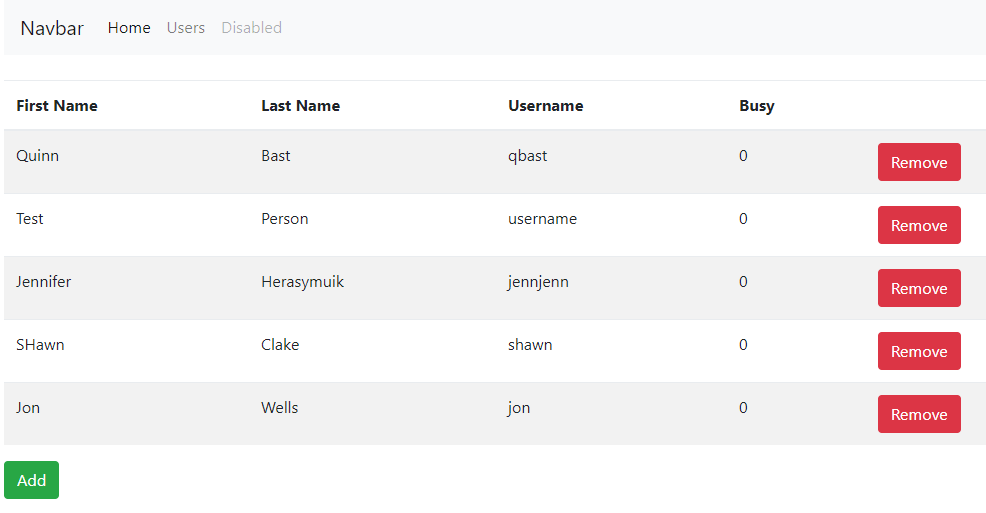
**Figure 3: Login Page**

The login page allows the IT employee to login to the web application using their provided credentials. This page currently does not meet requirements (view section 4 on Bugs).



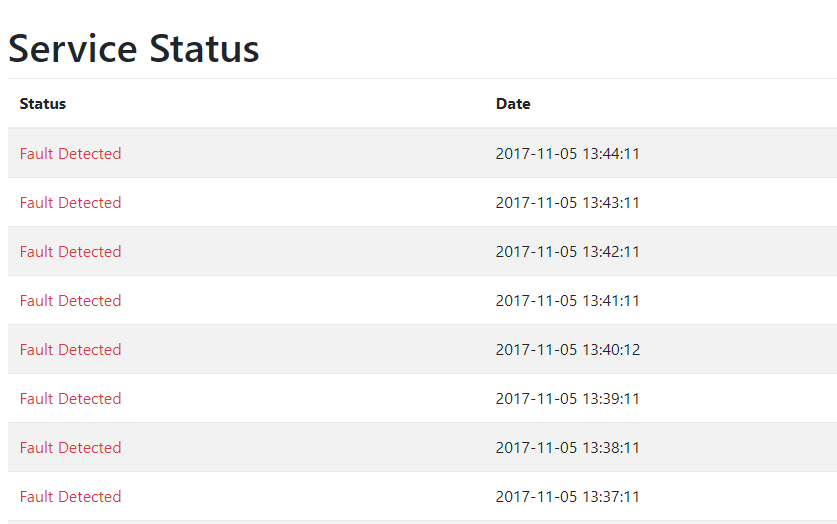
**Figure 4: Details Page**

The details page shows all of the details for a specific server.

****

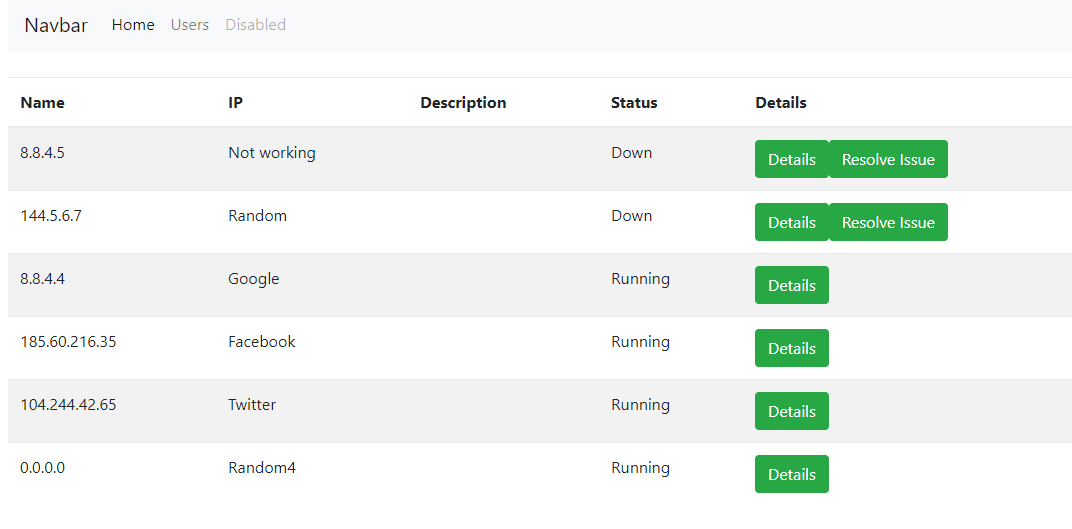
**Figure 5: Users**

The users page lists all IT staff.



**Figure 6: Server Status**

The server status shows the 50 most recent polling status for a service.



**Figure 7: Dashboard Page**

The dashboard page shows information for all of servers.

## 3 Additional Information

### 3.1 Design Methodology

Since there was a short deadline on this project, all tasks were split up between the four team members. One team member was in charge of the creation of the database, all of its tables, and any possible views required. This team member also worked on proper documentation for the web application.

Another team member was in charge of determining how to determine whether there was an issue with a service or not. This was done using a php document that runs every minute. This person also helped with design of the front end as well.

The third team member worked solely on the front end, designing the main pages as well as adjusting the styling required for each page.

The fourth and final team member worked on the front end and helped in the backend.

All of this work was done simultaneously, will collaborating with other team members during the full time period.

### 3.2 Improvements

One improvement we wanted was to add an issues through the application manually, instead of waiting for an issue to come in. Say someone from IT noticed an issue that the app can’t pick up, the staff member can manually add the issue to the site and it will save to the database.

Another goal that was out of scope was letting the user know when he issue is resolved. In the Customer table we can use their email to send them the issue they had, and how it was resolved. This allows for better communication and removes the burden of remembering to notify the customer off the employee.

The last thing that can be improved is the security. With the time allotted to the project, we were not able to ensure proper security measures. For example, our passwords are stored in plain text which is definitely not recommended. With more time, better security features would be added.

## 4 Bugs

* Logging in does not create a session and thus authentication is not verified across pages. (Not enough time, went out of scope)