Patrol

by

Sean McLeod, Kramer Hampton, and Ian Geverdt

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Sean McLeod — 4/10/2020
Sean McLeod — Date

Kramer Hampton — 4/10/2020
Kramer Hampton — Date

Ian Geverdt — 4/10/2020
Ian Geverdt — Date

Ryan Moore — 4/26/2020
Ryan Moore, Faculty Advisor — Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services
November 2019

patrol

Prepared By:

Sean McLeod, Project Manager and Database Architect
Kramer Hampton, Lead Software Developer
Ian Geverdt, Network and Security Analyst

Sponsors

Students of
University of Cincinnati
College of Education, Criminal Justice, and Human Services
School of Information Technology

April 2020

Table of Contents

List of Illustrations iii
Tables iii
Figures iii
Acronyms and Abbreviations iv
Abstract 1
Problem Statement 2
Introduction 2
Problem 2
Solution 3
Overview 4
Project Description 4
Project Concept 4
Methodology/Technical Approach 5
Basic User Profile 7
Reporter User Profile 8
Admin User Profile 10
Use Case Diagram 12
Testing Plan 13
Overview 13
Objective 13
Application Testing 13
Testing Procedures 14
Connectivity Test 15
User Interface Test 15
Functionality Test 17
Exploratory Test 20
Pass/Fail Conditions 22
Timeline 24
Project Management 26
Budget 26
Objectives/Deliverables 27
Project Schedule 27
Problems Encountered 31
Future Recommendations for Improvement 32
Technical Elements 34
Network 34
Database 35

Application 35
Security 35
User Interface 36
Logged Out View 36
Main View 37
Report View 38
Officer Management View 38
Incident Customization 39
Conclusion 41
Fall Semester 2019 41
Spring Semester 2020 42
Appendix A 46
References 46
Appendix B 47
Poster 47

List of Illustrations

Tables

No.		Page
Table 1	Basic User Profile	7
Table 2	Reporter User Profile	9
Table 3	Admin User Profile	10
Table 4	Connectivity Test	15
Table 5	Basic User Profile User Interface Test	16
Table 6	Reporter User Profile User Interface Test	17
Table 7	Admin User Profile	17
Table 8	High-Level Functionality Test	18
Table 9	Login Functionality Test	19
Table 10	Create Incident Functionality Test	19
Table 11	Create New User Functionality Test	19
Table 12	Create Incident Type Reports Functionality Test	20
Table 13	Data Saving Functionality Test	20
Table 14	Exploratory Test	22
Table 15	Internal Test Timeline	25
Table 16	Beta Test Timeline	25
Table 17	Work Breakdown Structure	29

Figures

No.		Page
Figure 1	Use Case Diagram	12
Figure 2	Ideal Budget	26
Figure 3	Working Budget	26
Figure 4	Gantt Chart	28
Figure 5	Network Diagram	34
Figure 6	Logged Out View	36
Figure 7	Main View	37
Figure 8	Dynamic Main View	38
Figure 9	Report View	38
Figure 10	Officer Management Index	39
Figure 11	Create Officer View	39
Figure 12	Incident Type Index	39
Figure 13	Incident Type Creation	40
Figure 14	Incident Type Custom Field Listing	40
Figure 15	Incident Type Custom Field Creation	40

Acronyms and Abbreviations

AWS	Amazon Web Service
BJS	Bureau of Justice Statistics
MPD	Mariemont Police Department
PPCS	Police Public Contact Survey
WBS	Work Breakdown Structure

Abstract

Many police departments lack technological resources that enables them to track meaningful data. Beginning in 2020, the State of Ohio has mandated that all departments track and collect police incident data. This data is where most police officers' public contact comes from according to the Bureau of Justice Statistics (Office of Justice Programs). Currently, this data is gathered from notes that the officer makes at the time of the incident and later recorded in an excel document. This process is inefficient and fallible. Patrol is a cross-platform web application that allows officers to easily, accurately and efficiently gather incident data. By creating an application that is cloud-based, officers can easily input information that stores the data at the time of the incident. This data is used to generate reports for the department and state auditors. Patrol's design and development allows officers to save time and difficulty in creating incident reports.

Problem Statement

Introduction

Many police departments, specifically in Hamilton County, are struggling to collect meaningful data. Beginning in 2020, the state of Ohio is mandating that all departments collect traffic stop data. Yet most departments are still managing themselves entirely through Microsoft Excel. Our group had been given Excel sheets that contain payroll data for each week over 20 years. Officers are expected to show up hours before their shift just to enter data into the various excel sheets. Various companies over the years have provided similar solutions but many fizzle out, and some have just shut down their services without warning, taking department data down with it. With this in mind, our goal was to create an easy to use web application that allows police departments to effortlessly collect incident data in an efficient manner.

Problem

The Bureau of Justice Statistics (BJS) and the Police-Public Contact Survey (PPCS) state that the most common reason for contact with the police is being the driver in a traffic stop (Office of Justice Programs,). Whether that's a speeding stop, impaired driving or traffic violation, traffic stops make up most of a police officer's public contact time.

According to Adam Geraci, an officer with the Mariemont Police Department (MPD), beginning in 2020, the state of Ohio is mandating that all departments track and collect traffic stop data. Most departments within Hamilton County struggle to document and

retain significant data due to a restricted budget that doesn't allow much room for upgrading dated technology, no means to conveniently and quickly document data, or some combination of the two. Most of the data collection is sloppily and inefficiently documented because there is no application which can provide them with an effective means to do so. Some police departments have turned to excel or a large leather book to collect data, this is inefficient and time consuming. There was an application that the Cincinnati police departments had previously used called PAMET. However, within the past couple of years, their data storage was lost and they couldn't provide the backups that they had promised, severely tarnishing their reputation (Key, 2018).

Solution

Patrol is a comprehensive web application that provides Mariemont Police Department with the ability to easily and effectively collect, document, and report on traffic stop data. This application allows for collection of all types of stop reasons including auto accidents, robbery, assault, etc. Furthermore, Patrol provides a means to create any new incidents that a police chief wants to track and report on.

Patrol has a target audience in any law enforcement and regional services such as a police department or the fire department. The main focus for Patrol is to provide the department with a single, intuitive platform that is effective for their use case. It provides the right kind of data gathering and retention functions and means to effectively document traffic stop data in order to meet the 2020 Ohio mandate. This application is a cross platform and secure cloud solution, built on AWS, that has an interface that users of differing technical aptitude can use without issue. To ensure that Patrol's data is secure

our group replicates data between two databases to prevent any loss of data in case a database goes down.

Overview

The remainder of this report highlights how Patrol was completed. This report includes the following sections: project description, project concept, user profile, use case diagram, objectives/deliverables, budget, testing, project schedule, technical elements (network, database, and application), user interfaces, problems encountered, what our group learned from the Fall and Spring Semester, and future recommendations.

Project Description

Patrol is a simple web application that allows for police offers to easily and accurately track incident data from a dynamic form no matter the incident. This allows officers' to transition from tracking all incident data from a physical book to a web application. Users that have reporting permissions are able to take the data recorded and report it into the format that they'd like. The Police Chiefs are able to create new users in Patrol whenever new officers are hired and new reports in case there is a new incident classification that needs to be tracked.

Project Concept

The problem that Patrol attempts to solve, was brought to light by Adam Geraci, a police officer at the Mariemont Police Department. He explained to our group that many police departments, specifically in Hamilton County, are struggling to collect meaningful

data. After meeting with Adam about Patrol, the biggest needs he identified were the ability to export data into a printable form, an easy interface for officers to interact with, and dynamic forms based on the incident that officers are reporting on.

Unfortunately, when creating Patrol our group had to abandon one of our goals and one of the features that our team was going to try to fit in. The goal that our goup had to abandon was the ability for officers to export data. When developing Patrol the other goals took precedence over exporting data due to us being able to export data from the database. Although the ability to export data is not available in the UI of Patrol, the database has access to export any data that the Mariemont Police Department wants from the database. This was the only goal that was able to be abandoned while still being able to provide some work-around for. The feature that our group wanted to fit in but didn't was the department sharing option. Currently there has not been discussion with any other police departments about Patrol since it has not been officially used yet, as of now it would be an unused feature and one that could be a project in itself. What was learned when contemplating adding this feature was that it would need multiple servers, which would have most likely taken us over the free tier of AWS.

Methodology/Technical Approach

Through the course of creating Patrol both design requirements and procedures allowed us to meet our goals. In the Fall portion of Senior Project Management a lot of the assignments assigned were getting the requirements of Patrol set in place. These included gantt charts, WBSs, problem statements, and other foundational documents.

Design Requirements allowed us to meet our goals in two different ways, the first way being that these different requirements allowed our team to wrap our heads around the aspects of Patrol our group hadn't given thought to yet. Due to the requirements making us figure this information out, our group got a better idea of what Patrol needed to do and how it would be achieved. The second way that these design requirements allowed us to meet our goals was that these documents served as cornerstones throughout both semesters of work. Whenever our team was unsure of a decision regarding Patrol our group would look back to the design requirements and try our best to follow what had been laid out previously. Along with the design requirements, procedures that our team adopted helped us achieve our goals. These procedures included having weekly stand-ups, communication in team-friendly applications (GroupMe, Microsoft Teams, and Zoom), and working sessions. For the entire time of Senior Project Management our Patrol team had done weekly stand-up meetings on Monday, this generally looked like meeting after class to discuss what had been completed and what our group planned on completing in the upcoming week. As the Coronavirus caused us to go remote, this turned into us calling in on Mondays and discussing the same itinerary. Generally once a month after the stand-up meeting our group would do a working session, this allowed us to all tackle a problem our group was having, or simply work together on what needed to be completed. When not meeting via call or in-person, our Patrol team communicated through GroupMe to discuss all other matters or share information needed before/after stand-ups. Through these different procedures it allowed us to get the full feel of a "team" and working environment. These procedures allowed us to communicate when problems occurred or even work through problems to achieve our goals.

Basic User Profile

Table 1: Basic User Profile, describes the basic user profile for the Patrol application. This basic user profile shows what potential users are a part of the user profile, any related experience these user have previously had, what similar applications they may have familiarity with, the tasks users have had prior experience doing, the frequency of use, and key design requirements for the specific user groups.

User Profile (Basic) – Patrol

The Basic user profile is the average police officer. Due to it being mandatory that police officers report their incident data, this is the only access the basic user will have. The interface that the basic user primarily deals with is the simple incident screen that contains textboxes and dropdowns specific to the incident that they're reporting on. All other user profiles inherit the basic user ability to enter traffic stop data.

Basic User Profile Form
Project: Patrol - A police reporting application
Potential Users: • Police officers
Software, Interface, and Related Experience: Most police officers will have had experience with police reporting tools or data collection methods. Patrol will not require officers to have used any similar software in the past as it will be a simple form that they fill out.

Experience with similar applications: ● PAMET ● Dragon Law Enforcement ● Omnigo
Task Experience: Users will have prior experience: ● Using a web application from their preferred browser ● Filling out forms that require specific data ● Submitting incident data
Frequency of Use: This will be used by officers on a daily basis. Ideally police officers are able to easily record all incidents at the end of their shift in a concise manner.
Key Project Design Requirements that the Profile Suggests: ● Simple UI ● Responsive to dropdown selection ● Short response times

Table 1. Basic User Profile

Reporter User Profile

Table 2: Reporter User Profile, describes the reporter user profile for the Patrol application. This reporter user profile shows what potential users are a part of the user profile, any related experience these user have previously had, what similar applications they may have familiarity with, the tasks users have had prior experience doing, the frequency of use, and key design requirements for the specific user groups.

User Profile (Reporter) – Patrol

The Reporter user profile is the police chief, police lieutenant, or any other leader in the police organization. This role deals less with inputting incident data and more with

analyzing and finding trends in the data captured. This role inherits the basic profile and is inherited by the admin user profile.

Reporter User Profile Form
Project: Patrol - A police reporting application
Potential Users: ● Police Chiefs ● Police Lieutenants ● Other Police department leaders
Software, Interface, and Related Experience: Most police chiefs and lieutenants will have had experience with police reporting tools or data collection methods. This profile targets users who have had experience in creating reports off of the incident data collected. Not all users will have experience with data reporting software. To ease this learning curve the UI will allow the user to easily choose what they want to see so that it builds the report they are looking for.
Experience with similar applications: ● PAMET ● Dragon Law Enforcement ● Omnigo ● Excel Pivot Tables
Task Experience: Users will have prior experience: ● Using a web application from their preferred browser ● Analyzing data collected from incident data ● Creating and reviewing reports
Frequency of Use: The frequency of use for the reporter role will be less often than the Basic user profile. Generally this will be used weekly or monthly for end of week/month reporting. This role will not need to be daily, but can be if wanted.

Key Project Design Requirements that the Profile Suggests:
● Simple UI ● Short response times ● Tips if needed ● Ability to export/share reports

Table 2. Reporter User Profile

Admin User Profile

Table 3: Admin User Profile, describes the admin user profile for the Patrol application. This admin user profile shows what potential users are a part of the user profile, any related experience these user have previously had, what similar applications they may have familiarity with, the tasks users have had prior experience doing, the frequency of use, and key design requirements for the specific user groups.

User Profile (Admin) – Patrol

The Admin user profile is the police chief or police lieutenant. This role deals with setting up users into the reporting system and giving certain users permission of the Reporter Role. Along with inputting new users and permissions this profile also handles any new reports that the police department looks to create.

Admin User Profile Form
Project: Patrol - A police reporting application
Potential Users: ● Police Chiefs ● Police Lieutenants

Software, Interface, and Related Experience:

Most police chiefs and lieutenants will have had experience with police reporting tools or data collection methods. This profile targets users who are most likely use to setting up new hires/employees in different systems. For those users who do not have prior experience to this, a tips and tutorial option will be available

Experience with similar applications:

- PAMET
- Dragon Law Enforcement
- Omnigo

Task Experience:

Users will have prior experience:

- Using a web application from their preferred browser
- Onboarding new hires/employees to current systems
- Creating reports that the police department fills out

Frequency of Use:

The frequency of use for the reporter role will be less often than the Basic user profile. Generally this will be used in more of a when needed basis, due to the variability of new hires it is not possible to select a frequency of use.

Key Project Design Requirements that the Profile Suggests:

- Simple UI
- Short response times
- Tips/tutorial if needed

Table 3. Admin User Profile

Use Case Diagram

Figure 1: Use Case Diagram, shows the use case for each user profile of Patrol. This diagram illustrates what specific tasks/permissions are assigned to each respective user profile when using Patrol.

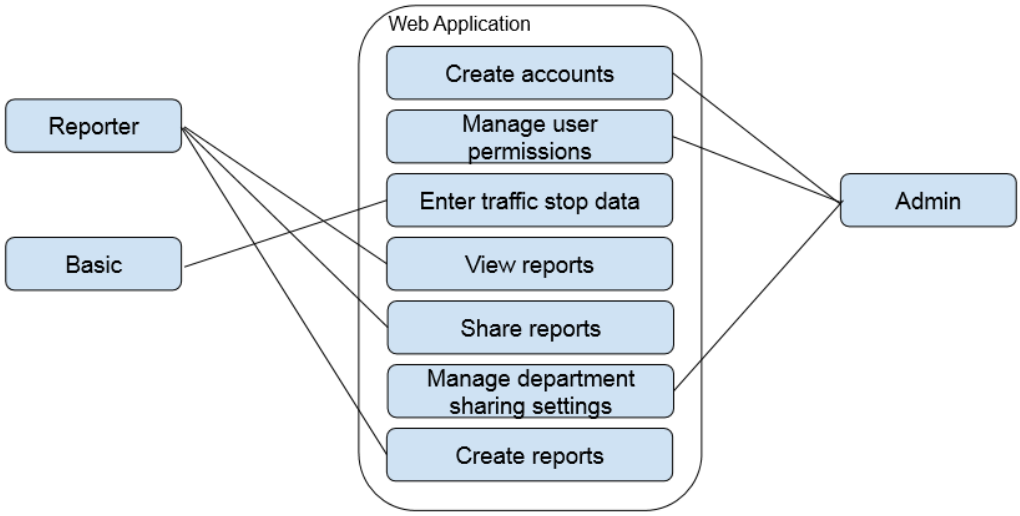

Figure 1. Use Case Diagram

Testing Plan

Overview

This section contains information regarding what tools, exercises, and requirements were used to properly test Patrol for any bugs that may occur when using the web application. Patrol can be accessed on any browser via desktop or mobile/tablet. The following groups were used to test the application:

- Developer
- Other team members
- Beta Users
    - Basic
    - Reporter
    - Admin

Objective

The goal of our testing is two part, the first being internal ensuring that all of our source code and application environment works as is expected without bugs. The second goal of our testing is confirming that the business requirements that were agreed on with the Mariemont Police Department work to their satisfaction, without error or bugs.

Application Testing

Application Testing allowed us to test the entirety of Patrol due to the fact that there are many moving parts that the application consists of. By using system testing our group is

able to test our source code, the AWS EC2 server it is hosted on, and the AWS RDS our database is hosted on. This end-to-end testing allows us to simulate how the end user will most likely interact with the whole application, not just specific parts of it. Any bugs or errors found by the system test allowed us to alleviate these issues before beta testing occurs.

Testing Procedures

The following is necessary in order for us to accurately test Patrol:

- Finish developing the source code for our application
- Create fail/pass conditions
- Keep track of all tests and their results

Tests that were performed:

- Connectivity Test: Captures whether the application can be accessed from the AWS EC2 server, from different web browsers.
- User Interface Test: Testing Patrol in the way that all end users will use the application from end-to-end. The different users tested were:
    - Basic
    - Reporter
    - Admin
- Functionality Test: Interacting with each specific functionality that each end user may use in their normal use of Patrol.
- Exploratory Test: The team members interacted with the application in ways that the app is not meant to be used, essentially trying to “break” the application.

Connectivity Test

Due to our end users potentially having many different preferred web browsers our connectivity test consisted of testing on Microsoft Edge, Google Chrome, and Mozilla Firefox. This test consisted of repetitively attempting to reach Patrol's web domain from each web browser. Our group attempted to connect from each web browser 20 times. The information that the launch tests allowed us to learn was:

1. Can Patrol be accessed repetitively from the same device?
2. Is there a browser that Patrol works best on?
3. Are there any issues accessing Patrol?
4. Is there any problem with the server that Patrol is hosted on?

Table 4: Connectivity Test, details the results of our connectivity tests from different web browsers and any notes that provide additional information.

Below is a table that shows the results and any necessary notes of the connectivity test.

Launch Test	Times Tested	Times Passed	Times Failed	Notes
Google Chrome	20	20		
Microsoft Edge	20	20		
Mozilla Firefox	20	20		

Table 4. Connectivity Test

User Interface Test

Our user interface test mimicked how each user profile interacts with our application. Our group did 10 full end-to-end tests that go over all the expected uses each profile either have access to or would use the most. For the Basic user profile our group attempted to login and then navigate to each possible incident type to verify fields change depending on the selected incident. The Reporter user profile consisted of logging in, navigating to

each possible incident type to verify field change depending on the selected incident, view the reports tab, view the reports for each incident type. Lastly for the Admin user profile our group attempted to login, add additional users to the department, select new departments to share data with, and create 5 new custom report types. The information that the User Interface test allowed us to learn was:

1. Does each user profile have access to what they need?
2. Are there any bugs with a functionality that a user profile would interact with regularly?
3. Does each user profile work as intended and expected?

Table 5: Basic User Profile User Interface Test, details the results of our basic user profile user interface test. This tested the use cases of the basic user profile.

Basic User Profile

Test	Times Tested	Times Passed	Times Failed	Notes
Login	10	10		
Selection of each incident type	10	10		
Verification of field change when incident type change	10	10		

Table 5. Basic User Profile user interface test
Above is the table that shows the tests of the Basic user profile user interface test and the corresponding results and notes.

Table 6: Reporter User Profile User Interface Test, details the results of our reporter user profile user interface test. This tested the use cases of the reporter user profile.

Reporter User Profile

Test	Times Tested	Times Passed	Times Failed	Notes
Login	10	10		
Selection of each incident type	10	10		
Verification of field change when incident type change	10	10		
Access to Reports Menu option	10	10		
Selection of each incident type report viewing	10	10		

Table 6. Reporter User Profile user interface test
Above is the table that shows the tests of the Reporter user profile user interface test and the corresponding results and notes.

Table 7: Admin User Profile User Interface Test, details the results of our admin user profile user interface test. This tested the use cases of the admin user profile.

Admin User Profile

Test	Times Tested	Times Passed	Times Failed	Notes
Login	10	10		
Add additional users	10	10		Didn’t allow me to make users with same username. This is good.
Create 5 Custom Report Types	10	10		Didn’t allow me to make same code for multiples. This is good.

Table 7. Admin User Profile user interface test
Above is the table that shows the tests of the Admin user profile user interface test and the corresponding results and notes.

Functionality Test

The Functionality test assessed the main functions of Patrol that end users interact with the most and on a typical usage. Functionalities that were tested are the login, creating

incident reports, exporting data, adding new users, creating custom reports, and saving data to the database. All of these functionalities were tested 20 times to ensure accuracy and that application is consistent in how it works. The information that the functionality test allowed us to learn was:

- Are users able to login?
- Can officers create incident reports?
    - Do any reports not work?
- Can the reporter user profile export data to State specifications?
- Is the admin user profile able to create new users?
- Can the reporter user profile create custom reports?
    - Do any fields not work in the custom report builder?
- Is data correctly saved from the application to the database?

Table 8: High-level Functionality Test, details the results of our main functionalities in the Patrol application.

Below is the table that contains the high-level tests of each functionality within the Patrol web application.

Functionality	Times Tested	Times Passed	Times Failed	Notes
Login	20	20		
Create Incident	20	20		
Create New Users	20	20		
Create Incident Type Reporting	20	20		
Data Saving	20	20		

Table 8. High-level Functionality Test

Table 9: Login Functionality Test, details the results of our login functionality test in the Patrol application.

Login Functionality Testing

Test	Pass/Fail?
Do text fields for the login work correctly?	Pass
Does the “submit” button work correctly for the login?	Pass

Table 9. Login Functionality Test
Above is the table that shows the detailed tests of the login functionality and if the respective test passed or failed.

Table 10: Create Incident Functionality Test, details the results of our create incident functionality test in the Patrol application.

Create Incident Functionality Testing

Test	Pass/Fail?
Does the dropdown for incident report selection work correctly?	Pass
Do the text fields for the incident report creation work correctly?	Pass
Is every incident type able to be created?	Pass
Are users restricted from entering a future date in the incident time?	Pass

Table 10. Create Incident Functionality Test
Above is the table that shows the detailed tests of the create incident functionality and if the respective test passed or failed.

Table 11: Create New Users Functionality Test, details the results of our create new users functionality test in the Patrol application.

Create New Users Functionality Testing

Test	Pass/Fail?
Do the text fields in the create new user page work correctly?	Pass
Does the “create” button work for creating new users?	Pass
Is only the admin user profile able to create new users?	Pass

Table 11. Create New Users Functionality Test

Above is the table that shows the detailed tests of the create new users functionality and if the respective test passed or failed.

Table 12: Create Incident Type Functionality Test, details the results of our create incident type reports functionality test in the Patrol application.

Create Incident Type Reports Functionality Testing

Test	Pass/Fail?
Are all data fields able to be reported on?	Pass
Do the fields filter when clicked on?	Pass
Are only the reporter and admin user profiles able to create an incident type report?	Pass

Table 12. Create Incident Type Reports Functionality Test
Above is the table that shows the detailed tests of the create incident type reports functionality and if the respective test passed or failed.

Table 13: Data Saving Functionality Test, details the results of our data saving functionality test in the Patrol application.

Data Saving Functionality Testing

Test	Pass/Fail
Is data saved when officers create a new incident type?	Pass
Is all the data saved to the correct data fields within RDS?	Pass

Table 13. Data Saving Functionality Test
Above is the table that shows the detailed tests of the data saving functionality and if the respective test passed or failed.

Exploratory Test

For the Exploratory tests our team set aside time on two separate occasions to sit down together and test Patrol. The hope for the exploratory tests were to find bugs that may have not come up in any of the previous test cases. When doing these tests our group did not focus on one functionality over another, but rather the entirety of the application. Any

bugs that are found through this were put in the table below. Bugs were categorized by priority (1 being the lowest priority, 5 being the highest priority):

1. Minor inconvenience to the developers – This is something that does not affect the end user at all. These would be bugs that only the developers may notice.
2. Minor inconvenience to the end user – This is something that would not affect the functionality of the application, but would give the end user doubts about the application. These would be not well refined UI displays, or misspellings.
3. Major inconvenience to the end user – This is something that would not affect the functionality of the application, but may affect how well the end user could do their job using the application. These would be missing incident types in a dropdown, or other missing information that could hinder information the police department are trying to enter.
4. Functionality not working (low priority) – This is something that is not working as intended, but is not a major business requirement requested by the Mariemont police department. This would be an inability to share data amongst different departments or an inability to create incident type reports by the Reporter user profile.
5. Functionality not working (high priority) – This is something that is not working as intended and is a major business requirement requested by the Mariemont police department. These would be an incident report not saving data, users not being able to login, or new users not being able to be added, etc.

The information that the exploratory test allowed us to know was:

- Are there any bugs not found by the other tests?

- What additional bugs need to be alleviated before our application is presentable?

Dates Met:

1. 3/23/2020, 6:00 – 7:00 pm
2. 3/30/2020, 6:00 – 7:00pm

Table 14: Exploratory Test, details the results of our exploratory test in the Patrol application.

Functionality	Bug Description	Priority (1-5)	Notes
Incident Creation	Currently dropdown has “Alaska Native”, think it should be “Alaskan Native”.	2	Fixed
Incident Creation	Currently dropdown has “African America”, think it should be “African American”.	2	Fixed
Officer Creation	Currently there is a maximum length for username. Should this be same for first and last name?	1	Fixed
Incident Type Creation	Currently there is no maximum length for incident type code	2	Fixed
Incident Type Creation	Would like to be able to delete custom fields created for an incident type	2	Would delete incidents with that field, should not make change.
Incident Type Creation	Currently have Recent Incidents along with “placeholder” underneath it. Should populate this or remove it.	3	Fixed
Incident Report	Day shown is one day ahead of when created.	2	Time is dictated by the time zone that our RDS instance is running in. Ensure that our RDS is running on the same time zone as the officers.
Incident Creation	The incident “Telephone Harassment” is spelled incorrectly	2	Fixed

Table 14. Exploratory Test

Above is the table that shows any bugs found in the exploratory tests and the priority that the bug has.

Pass/Fail Conditions

In order for Patrol to meet the satisfaction of the Mariemont Police Department and our team's expectations the web application must successfully pass every test mentioned above. The Pass/Fail conditions for each test are:

- Connectivity Test – Patrol must load the login screen and be fully responsive when the web domain is reached.
- User Interface Test
    - Basic User Profile – The user must be able to login every time attempted and Patrol must have responsive form building for each incident type selected.
    - Reporter User Profile – The user must be able to login every time attempted, Patrol must have responsive form building for each incident type selected, the reports menu option must take a user to the reports page in less than 3 seconds, and each incident type must be able to be reported on.
    - Admin User Profile – The user must be able to login every time attempted, the added user must save to the database each time it is created, the new department shared to must save to the database each time a new sharing is created, and all 5 custom report types must be available once created by the admin.
- Functionality Test – The specific functionality must be able to be used how the user profile intended for that functionality expects it to. If there is a text field it must be able to be filled with text, dropdowns must be complete and able to be

chosen, all buttons must be clickable and perform some action if necessary fields are completed, etc.

- Exploratory Test – No bugs must be found higher than a level 1 which is considered just a minor inconvenience to the developer.

Timeline

For the testing of Patrol our group had two different timelines that briefly intersect. The first timeline was our internal testing that begins with unit testing of source code and network architecture. This is done every time that new source code is written, or a network implementation is added. At the minimum both of these are checked weekly when our group meets. Once the source code is finished the group moved from unit testing to connectivity testing to ensure that the web application can be accessed. From then on user interface testing, functionality testing, and exploratory testing was run in parallel until the Tech Expo. After our first exploratory testing is completed by the group, Patrol will then roll out beta testing to the Reporter user profile as it is who our group has been working closest to in the Mariemont Police department. Once the reporter user profile has had some time to test the web application it will then move to the police chief who is considered the admin user profile, and then finally to the police officers that make up the basic user profile.

Table 15: Internal Test Timeline, details the timeline of internal testing in the Patrol application.

Internal Testing

Test	Person(s)	Date Range	Frequency
Unit Testing (source code)	Kramer Hampton	11/1/2019 – 4/14/2020	Weekly
Unit Testing (Network Architecture)	Sean McLeod Ian Geverdt	1/12/2020 – 4/14/2020	Weekly
Connectivity	All Group Members	3/9/2020 – 4/14/2020	Weekly
User Interface	All Group Members	3/9/2020 – 4/14/2020	Weekly
Functionality	All Group Members	3/9/2020 – 4/14/2020	Weekly
Exploratory	All Group Members	3/23/2020	One Time
Exploratory	All Group Members	3/30/2020	One Time

Table 15. Internal Test Timeline

Above is the table that shows Patrol's internal testing timeline.

Table 16: Beta Test Timeline, details the timeline of beta testing in the Patrol application.

Beta Testing

Who	When
Group Members	11/1/2019 – 4/14/2020
Reporter User Profile (Adam Geraci)	4/20/2020 – 6/30/2020
Admin User Profile (Police Chief)	5/4/2020 – 6/30/2020
Basic User Profile (Police Officers)	5/11/2020 – 6/30/2020

Table 16. Beta Test Timeline

Above is the table that shows Patrol's beta testing timeline.

Project Management

Budget

Figure 2: Ideal Budget, this would be the budget if our group created Patrol on AWS to the specifications our team wanted. Our team would use this budget to implement AWS services that are above the free tier, allowing for scalability if needed.

No.	Item	Unit Hours	Unit Price	Line Item Total
1	Labor Hours	200	$ 70.00	$ 14,000.00
2	t2.Medium EC2 Server (monthly)	1	$ 23.95	$ 574.80
1	db.t3.Medium RDS (monthly)	1	$ 42.14	$ 505.68
			1st year Total Cost	$ 15,080.48
			5th year Total Cost	$ 19,402.40

Figure 2. Ideal Budget

Figure 3: Working Budget, this is the budget our team used to create Patrol. To thank the Mariemont Police Department for letting us do this project our group wanted to donate, not sell this project to them.

No.	Item	Unit Hours	Unit Price	Line Item Total
1	Labor Hours	200	$ -	$ -
1	t2.Micro EC2 Server (monthly)	1	$ 0.69	$ 8.26
1	db.t3.Micro RDS (monthly)	1	$ 1.07	$ 12.87
			1st year Total Cost	$ 21.13
			5th year Total Cost	$ 105.65

Figure 3. Working Budget

Objectives/Deliverables

Develop a mobile-friendly web application to assist Police Officers in the collection and reporting of crime data. Patrol gives officers key insights into the work they are doing every day.

Patrol provides officers with a modern data entry system with the help of modern technologies. Our back-end is high-throughput and highly scalable server written in Elixir and built on AWS technologies. The front-end has a modern look and feel with the use of React and modern design systems.

Our Features Include:

- Officers have the ability to quickly enter traffic stop data securely on the go or from any computer.
- Higher ranking officers have the ability to report on and summarize various data points within the traffic stop data.
- Higher ranking officers have the ability to create new incidents that need to be reported on.
- High ranking officers have the ability to add and delete new officers that get hired at the Mariemont Police Department.

Project Schedule

Figure 4: Gantt Chart, this visualizes our project plan and deliverables of the Fall 2019 semester and the Spring 2020 semester.

Table 17: WBS (Work Breakdown Structure), lays out our group's project plan to complete Patrol in the scheduled deadline.

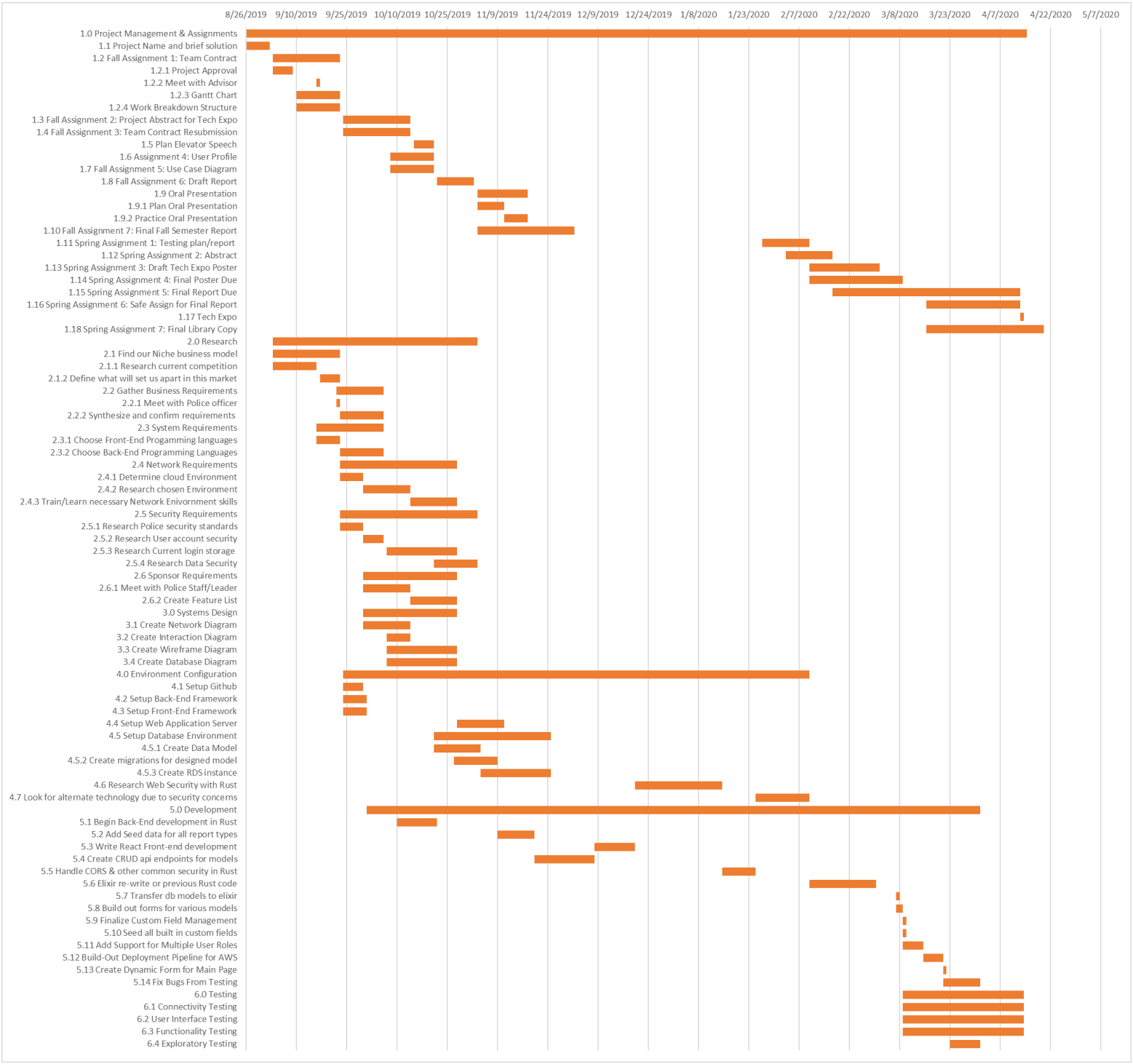

Figure 4. Gantt Chart

Task	Start Date	End Date	Duration
1.0 Project Management & Assignments	8/26/2019	4/15/2020	233
1.1 Project Name and brief solution	8/26/2019	9/2/2019	7
1.2 Fall Assignment 1: Team Contract	9/3/2019	9/23/2019	20
1.2.1 Project Approval	9/3/2019	9/9/2019	6
1.2.2 Meet with Advisor	9/16/2019	9/17/2019	1
1.2.3 Gantt Chart	9/10/2019	9/23/2019	13
1.2.4 Work Breakdown Structure	9/10/2019	9/23/2019	13
1.3 Fall Assignment 2: Project Abstract for Tech Expo	9/24/2019	10/14/2019	20
1.4 Fall Assignment 3: Team Contract Resubmission	9/24/2019	10/14/2019	20
1.5 Plan Elevator Speech	10/15/2019	10/21/2019	6
1.6 Assignment 4: User Profile	10/8/2019	10/21/2019	13
1.7 Fall Assignment 5: Use Case Diagram	10/8/2019	10/21/2019	13
1.8 Fall Assignment 6: Draft Report	10/22/2019	11/2/2019	11
1.9 Oral Presentation	11/3/2019	11/18/2019	15
1.9.1 Plan Oral Presentation	11/3/2019	11/11/2019	8
1.9.2 Practice Oral Presentation	11/11/2019	11/18/2019	7
1.10 Fall Assignment 7: Final Fall Semester Report	11/3/2019	12/2/2019	29
1.11 Spring Assignment 1: Testing plan/report	1/27/2020	2/10/2020	14
1.12 Spring Assignment 2: Abstract	2/3/2020	2/17/2020	14
1.13 Spring Assignment 3: Draft Tech Expo Poster	2/10/2020	3/2/2020	21
1.14 Spring Assignment 4: Final Poster Due	2/10/2020	3/9/2020	28
1.15 Spring Assignment 5: Final Report Due	2/17/2020	4/13/2020	56
1.16 Spring Assignment 6: Safe Assign for Final Report	3/16/2020	4/13/2020	28
1.17 Tech Expo	4/13/2020	4/14/2020	1
1.18 Spring Assignment 7: Final Library Copy	3/16/2020	4/20/2020	35
2.0 Research	9/3/2019	11/3/2019	61
2.1 Find our Niche business model	9/3/2019	9/23/2019	20
2.1.1 Research current competition	9/3/2019	9/16/2019	13
2.1.2 Define what will set us apart in this market	9/17/2019	9/23/2019	6
2.2 Gather Business Requirements	9/22/2019	10/6/2019	14
2.2.1 Meet with Police officer	9/22/2019	9/23/2019	1
2.2.2 Synthesize and confirm requirements	9/23/2019	10/6/2019	13
2.3 System Requirements	9/16/2019	10/6/2019	20
2.3.1 Choose Front-End Progamming languages	9/16/2019	9/23/2019	7
2.3.2 Choose Back-End Programming Languages	9/23/2019	10/6/2019	13
2.4 Network Requirements	9/23/2019	10/28/2019	35
2.4.1 Determine cloud Environment	9/23/2019	9/30/2019	7
2.4.2 Research chosen Environment	9/30/2019	10/14/2019	14

2.4.3 Train/Learn necessary Network Enivornment skills	10/14/2019	10/28/2019	14
2.5 Security Requirements	9/23/2019	11/3/2019	41
2.5.1 Research Police security standards	9/23/2019	9/30/2019	7
2.5.2 Research User account security	9/30/2019	10/6/2019	6
2.5.3 Research Current login storage	10/7/2019	10/28/2019	21
2.5.4 Research Data Security	10/21/2019	11/3/2019	13
2.6 Sponsor Requirements	9/30/2019	10/28/2019	28
2.6.1 Meet with Police Staff/Leader	9/30/2019	10/14/2019	14
2.6.2 Create Feature List	10/14/2019	10/28/2019	14
3.0 Systems Design	9/30/2019	10/28/2019	28
3.1 Create Network Diagram	9/30/2019	10/14/2019	14
3.2 Create Interaction Diagram	10/7/2019	10/14/2019	7
3.3 Create Wireframe Diagram	10/7/2019	10/28/2019	21
3.4 Create Database Diagram	10/7/2019	10/28/2019	21
4.0 Environment Configuration	9/24/2019	2/10/2020	139
4.1 Setup Github	9/24/2019	9/30/2019	6
4.2 Setup Back-End Framework	9/24/2019	10/1/2019	7
4.3 Setup Front-End Framework	9/24/2019	10/1/2019	7
4.4 Setup Web Application Server	10/28/2019	11/11/2019	14
4.5 Setup Database Environment	10/21/2019	11/25/2019	35
4.5.1 Create Data Model	10/21/2019	11/4/2019	14
4.5.2 Create migrations for designed model	10/27/2019	11/9/2019	13
4.5.3 Create RDS instance	11/4/2019	11/25/2019	21
4.6 Research Web Security with Rust	12/20/2019	1/15/2020	26
4.7 Look for alternate technology due to security concerns	1/25/2020	2/10/2020	16
5.0 Development	10/1/2019	4/1/2020	183
5.1 Begin Back-End development in Rust	10/10/2019	10/22/2019	12
5.2 Add Seed data for all report types	11/9/2019	11/20/2019	11
5.3 Write React Front-end development	12/8/2019	12/20/2019	12
5.4 Create CRUD api endpoints for models	11/20/2019	12/8/2019	18
5.5 Handle CORS & other common security in Rust	1/15/2020	1/25/2020	10
5.6 Elixir re-write or previous Rust code	2/10/2020	3/1/2020	20
5.7 Transfer db models to elixir	3/7/2020	3/7/2020	1
5.8 Build out forms for various models	3/7/2020	3/9/2020	2
5.9 Finalize Custom Field Management	3/9/2020	3/9/2020	1
5.10 Seed all built in custom fields	3/9/2020	3/9/2020	1
5.11 Add Support for Multiple User Roles	3/9/2020	3/15/2020	6
5.12 Build-Out Deployment Pipeline for AWS	3/15/2020	3/21/2020	6
5.13 Create Dynamic Form for Main Page	3/21/2020	3/21/2020	1
5.14 Fix Bugs From Testing	3/21/2020	4/1/2020	11
6.0 Testing	3/9/2020	4/14/2020	36

6.1 Connectivity Testing	3/9/2020	4/14/2020	36
6.2 User Interface Testing	3/9/2020	4/14/2020	36
6.3 Functionality Testing	3/9/2020	4/14/2020	36
6.4 Exploratory Testing	3/23/2020	4/1/2020	9
6.4.1 Exploratory Test 1	3/23/2020	3/23/2020	1
6.4.2 Exploratory Test 2	3/30/2020	3/30/2020	1

Table 17. WBS (Work Breakdown Structure)

Problems Encountered

1. Random people attempted to access Patorl's IP address: When looking at the logs of when our IP domain was being accessed our team realized that there were harmful attempts to access our site. To combat this our group set up a VPC (Virtual Private Cloud) and other rules on who could access our site.
2. NGINX unnecessary use: At the beginning of our group's project our group expected to use NGINX as a reverse proxy for our web application. Once our group began deploying the application to allow for testing, it was unnecessary and Patrol could do everything our group needed without it. To avoid this technical debt our group took NGINX off of our EC2 server.
3. Card Expired: There was an issue where our group's Amazon Web Services account was nearly terminated due to the card on file expiring. This was just a few weeks before expo and caused a little bit of panic because Patrol still had some testing and development needed. Fortunately, a few hours after adding a new card, the account opened back up and our group was able to continue working.
4. Lack prior AWS experience: Although all of our team had heard and knew what AWS was, there was a learning period needed before our group felt truly comfortable to make important modifications/configurations on our server,

database, or security. This was solved by creating test instances that our group could practice on and reading through documentation on what modifications/configurations had the ability to do.

5. Phoenix LiveView requires a websocket connection and getting that to work properly with our AWS load balancer and container services was a challenge. Our group eventually found documentation for Phoenix sockets that explained how to add the hostname to “known hosts”.

Future Recommendations for Improvement

If our group had to do Patrol all over again, our group would have never used NGINX, as mentioned earlier Patrol planned on using NGINX as a reverse proxy. After meeting with Professor Moore, he had told us that it was not needed. It wasn’t totally a waste because our group got to learn about what NGINX had to offer, but it did take up a whole working meeting that was held after one of our classes.

Mid way through our project our group also rewrote Patrol in Elixir. Initially the Rust version was fast and had a tiny memory footprint however it was lacking many security features. Our group decided it would be better to write the application in a much more mature framework. That framework was Phoenix. Phoenix is a battle-tested web framework written in Elixir, built on Erlang. It comes with many security features out of the box and the best part is a developer don’t have to explicitly handle many of them. Phoenix allows users to focus on the application while providing users with the confidence that the app is secure.

If our group had more time to work on Patrol there are 2 functions that our group would have done. The first function is one that had to be abandoned, department sharing,

this would have enabled the Mariemont Police Department to share data with a neighboring police department. The idea of Patrol could be used as a software as a service for multiple police departments, but the time allotted only allowed us to do it for the Mariemont Police Department. Without other departments using the application, department sharing would be a complex unused feature initially. Another function that our group would have liked to do if there was more time would be a live map of incidents in any given time frame. This would allow officers to view where the highest volume of incidents reported on took place most likely using the Google Maps API. As noted earlier our group got a lot of great advice from Professor Moore in our meetings with him, his suggestions of taking out NGINX and network security helped guide our project to completion. Our group removed NGINX as it was a barrier for us and included a Virtual Private Cloud for network security. Professor Moore's recommendation of putting the database on its own network and only letting inbound requests come from the VPC was very helpful in implementing security for the data held by the police deparment. Our Patrol team's plans for this project are on providing the Mariemont Police Department with the Patrol web application. Our team agreed that Kramer Hampton would help with any support that the Police Department requests.

Technical Elements

Network

As indicated in Figure 5, Network Diagram, Patrol uses Amazon Web Services to host our databases and application servers. To ensure reliability of our application, our group included load balancers to help equalize the incoming traffic and network loads across multiple servers. Additionally, the gathered Patrol data spans more than one database that replicates between each other in case of emergency. Since Patrol is dealing with real personal information, Patrol cannot risk having all of the information that's gathered be stored in a single database.

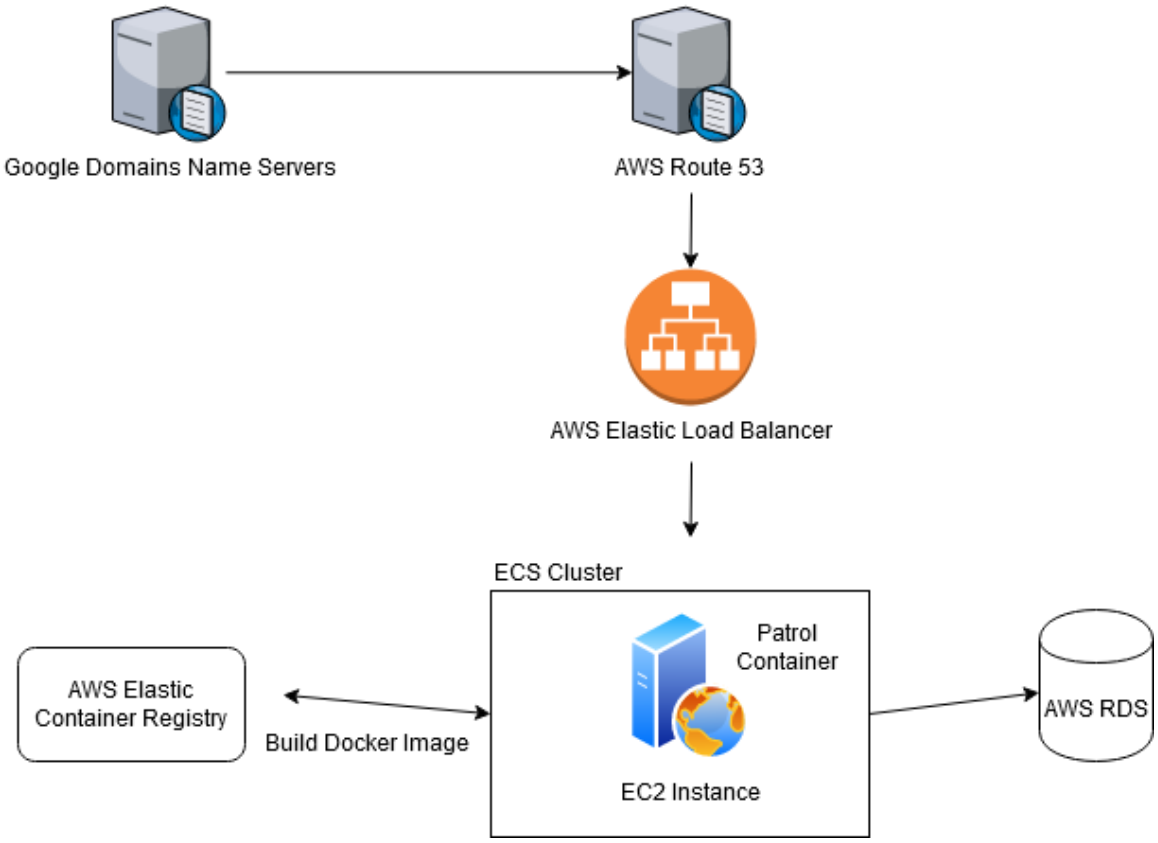

Figure 5. Network Diagram

Database

Patrol uses a PostgreSQL database for data storage. PostgreSQL is a free open-source technology that allows us to easily connect to Amazon's Relational Database Service. Our data model is designed with speed and efficiency in mind. If Patrol is used long-term it has to store data in an efficient way for quick reporting and easy data visualization. To ensure there is fault tolerance with the data Patrol is storing our group had set up the RDS database to have daily backups along with data replication with a lag of just 100 seconds. These were major problems in the previous tools that the Mariemont Police Department had used.

Application

Patrol is written in Elixir and uses the Phoenix web framework. This server interfaces with the Postgres database through an ORM called Ecto. Patrol takes advantage of Tailwind CSS to rapidly build out responsive web pages. Patrol utilizes cutting edge features such as Phoenix LiveView to provide users with a truly dynamic interface. Patrol focuses on data entry as many government agencies have their own reporting systems. A CSV dump of department data will be readily available to state governments.

Security

Patrol has a few different security features in place. The first of these is scheduling regular backups of the database. Patrol has daily backups in place, so in the case of a blip in the database, Patrol is able to roll back easily. The second measure Patrol

has in place is data replication. The data isn't stored on one single database, there are multiple set in place that are always replicating between each other, so in the case that one of the database servers goes down, there is another in place to carry the load. And lastly, the database is on its own, private network. In AWS, these are known as VPC's. The database and webservers are isolated to their own VPC's. The database VPC is set up to only allow inbound traffic from the webserver VPC and the webserver VPC is blacklisted to all addresses besides the members of the Mariemont Police Departments network. This would be the best approach as only officers of the MPD need to input incident data. Additionally, outside of the scope of database security, the website itself requires credentials to connect anyway.

User Interface

Logged Out View

As indicated in Figure 6, officers will use a simple username password combination to access patrol at their respective sub-domain of patrolling.app.

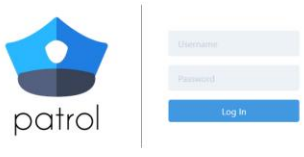

Figure 6. Logged Out View

Main View

The main view after logging in is where 90% of officers spend their time within Patrol. It offers a dynamic form for officers to report on almost any kind of incident as seen in figure 8. If it doesn't exist out of the box, Admin's can create the new Incident Type. This form updates with the available custom types. General information needed for all incidents is located at the top indicated in figure 7, while any custom fields specific to that incident are added below. Any number of people may be added to an incident.

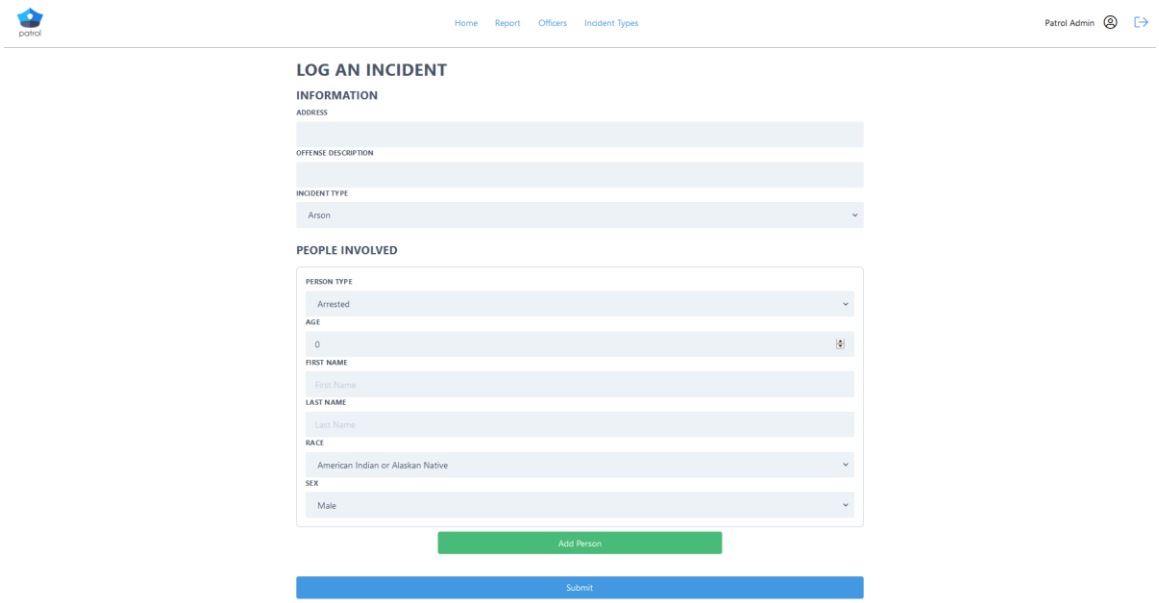

Figure 7. Main View

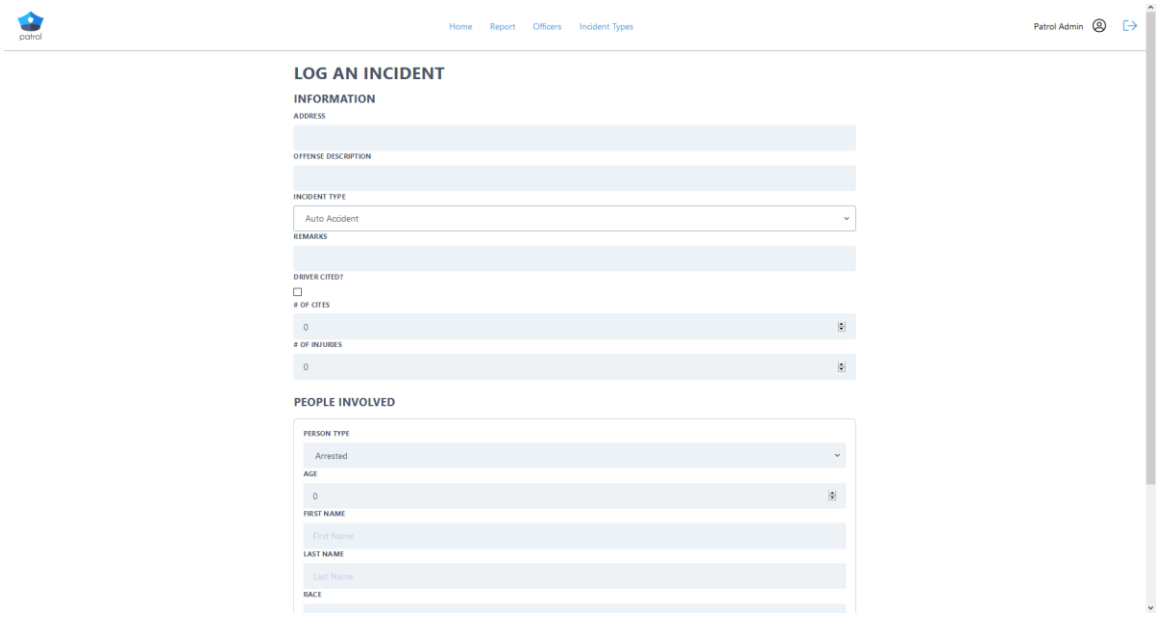

Figure 8. Dynamic Main View

Report View

As shown in figure 9, report view allows officers to view submitted incidents filtered by incident type.

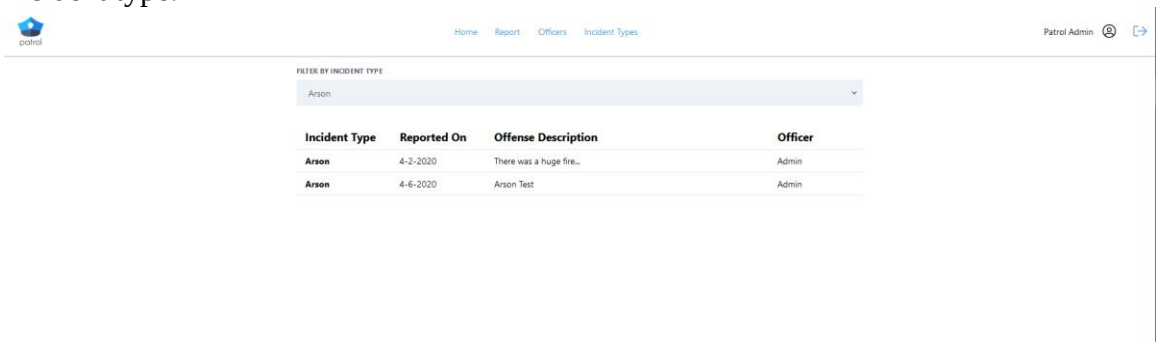

Figure 9. Report View

Officer Management View

The officer management view (figure 10) allows Admins to manage the access Officers have to the application. This includes adding additional officers (figure 11), editing existing ones, and deleting officers.

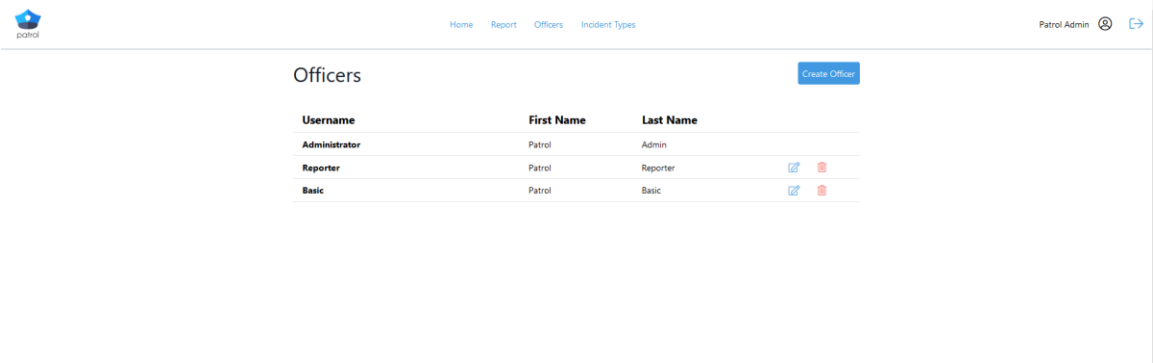

Figure 10. Officer Management Index

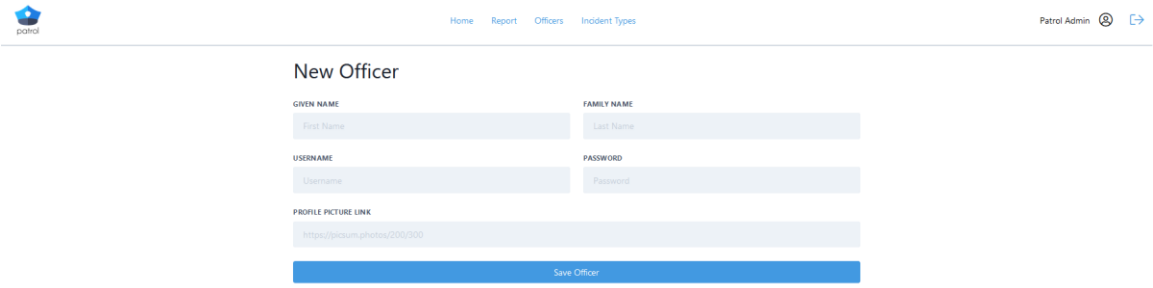

Figure 11. Create Officer View

Incident Customization

Admins can customize anything incident related within Patrol as shown in figure 14. This includes adding new Incident Types (see figure 13) and creating new Custom Fields (see figure 15) for an incident type. Editing incident type information immediately reflects in the dynamic form on the home page. All Incident Types can be seen as indicated in Figure 12.

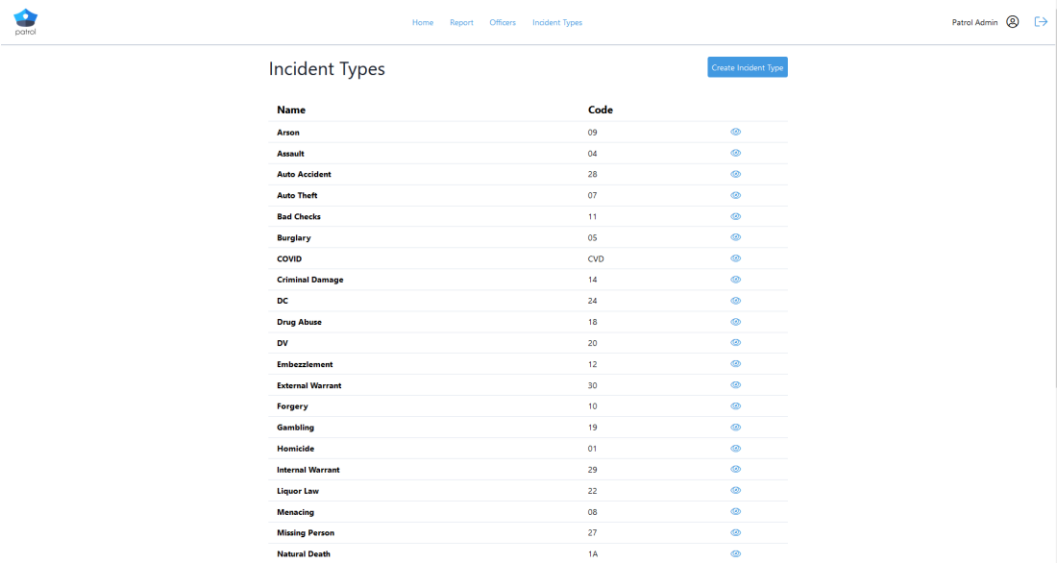

Figure 12. Incident Type Index

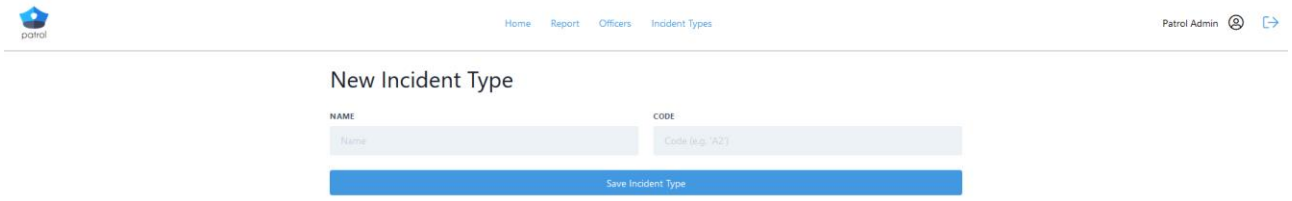

Figure 13. Incident Type Creation

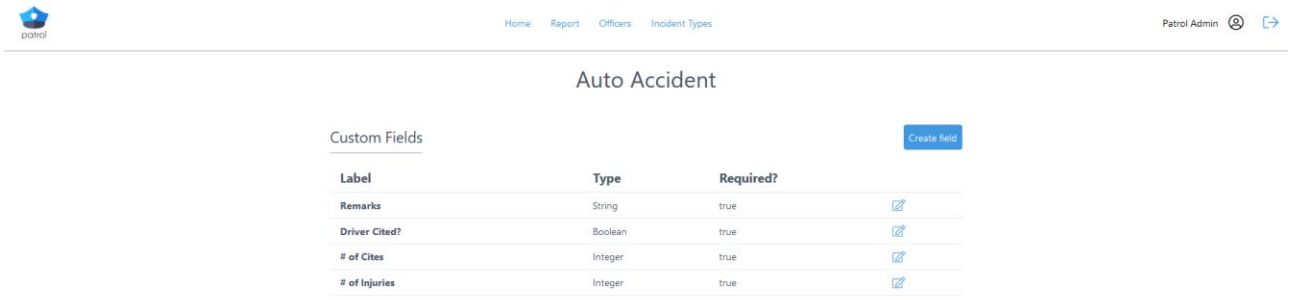

Figure 14. Incident Type Custom Field Listing

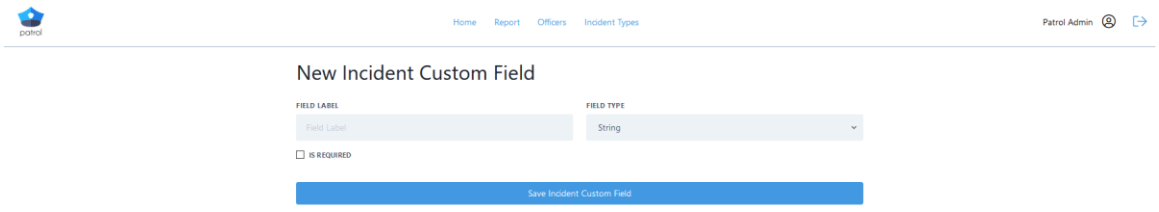

Figure 15. Incident Type Custom Field Creation

Conclusion

Fall Semester 2019

This Fall Semester has definitely had its ups and downs with creating a product/project for Senior Design. When our group first started the class our group wanted to create a mobile application that could track food trucks from city to city. Upon first research this was something that was needed, but not available. Unfortunately, after our group did more research some applications that seemed incomplete or merely in certain cities turned out to actually be what our group was looking to create. This lack of research on our part at the beginning changed the way our group researched the rest of the semester. Our group learned that when researching, our team needed to see what other applications are truly out that may compete with our product, what requirements are specifically needed, and what are the right technologies to build this new product. In the Fall Semester a lot was learned, researched, and discussed within our Patrol team.

From the networking aspect of Patrol, Ian has researched a large breadth of AWS and how Patrol would work in a cloud and serverless environment. With this research Ian has learned a lot more of the capabilities that AWS has to offer and how these different systems work with each other.

Sean has been working with the database of Patrol and previously had only worked with SQL Server Management Studio, through working on Patrol he has gotten a chance to explore other database options. Patrol is using PostgreSQL which is a new technology for him. With creating the data model Sean also learned how to store data in an efficient manner that is easy to maintain moving forward.

Kramer previously has had a lot of experience with the technologies that Patrol is using but gets further exposure to them. Kramer and Sean working together to implement Diesel which is an ORM and query builder are new technologies for both of them.

As a team Patrol has learned to get a head start on assignments and deadlines so that quality work will be published. Quality is much more important than getting a project done quickly. A problem that our group encountered when trying to make this a quality project was trying to determine what exactly our goals should be regarding the requirements of Patrol. To address this, our group was able to schedule a meeting with an Officer at the Mariemont Police Department to get a firsthand explanation of their goals and requirements that fir their daily use case.

Spring Semester 2020

The Patrol team learned that communication helps define expectations, the importance of testing, how to setup a secure network via AWS services and code used, and how to give a good presentation virtually. Our team relied on communication in the Spring Semester largely in part to COVID-19 affecting meeting in person. This allowed our team to understand how to be clear in our goals, punctual in the timelines set, and to reduce assumption. Through learning these aspects of communication, it turned our team from discussing high-level details to being specific in our meetings, meeting more often, and having more accountability in our share of the project. Once development was finished, our team took to testing Patrol as mentioned earlier in our test plan. Through testing Patrol from the officers' point of view our group was able to understand what points of emphasis for them were not satisfactory in our first rollout. After iterations of

testing and additional development our group is delivering a product to them that highlights the main use cases that they are looking for. Through additional research our Patrol team learned how to create a secure application. With a mix of the framework Patrol usew, the AWS configurations created, and the testing, our group can confidently say that Patrol is a secure application. In the future when creating networks, software, or handling data our team now knows how to put into place security methods in a plethora of ways. After presenting Patrol to the advisors our team got valuable feedback from our presentation especially dealing with the demo. Due to the demonstration of our application having a higher priority in this semester our group figured to make that a large part of our focus. Kramer did a great job of walking through the demonstration and explaining functionality. From the feedback our group received from the advisors our team learned to minimize “dead space” in our demonstration and to keep a consistent rhythm when presenting.

Ian learned a lot about Amazon Web Services and the different solutions they offer and how to take advantage of them such as creating a VPC. He was able to practice trust in his teammates that they were able to successfully complete the work that they were assigned so our group could complete this project—it takes a village after all.

Skills that Sean has learned while working on Patrol are how to formulate a testing plan and consistently test to ensure applications are working properly and how to provide data reliability. Prior to Patrol Sean had worked at places that had testers separate from developers, leaving Sean with just the development skill. While working on this project Kramer handled most of the development while Sean took did a majority of the testing, being on the other side of the fence has taught Sean the importance and skill of

testing. Walking through the testing plan gave Sean the skill of viewing the application from an outsider's perspective, although him and his group may know what the web application is supposed to do, testing is put in place so that every function is held accountable. On the data side Sean got to work with the AWS RDS tool which held our PostgreSQL database. Through working with this tool Sean learned how to setup both data backups and data replication within AWS. This was crucial for the Mariemont Police Department because their prior system had lost their backup data.

After working on the development of Patrol, Kramer got to work with web sockets that were new to him. Through reading documentation, he was able to develop Patrol in the way he wanted end users to interact with it. A lot of the web sockets that Kramer had to learn were for Phoenix LiveView which was also a new technology that he had learned for this project. Phoenix LiveView allowed Patrol to have the completely dynamic form and views.

Since the Fall Semester our Patrol team has done four major accomplishments these are finish development, create and modify database settings, implement security for Patrol, and test our finished product. Kramer worked on most of the development for Patrol and has added measures in case the web applications needs to be scaled or deployed elsewhere. All of the team members have worked in modifying the settings of the AWS architecture, some of the specific database updates our group created were data backups and data replication of our RDS service. Ian has primarily worked on the security by setting up a VPC, making sure that only people who Patrol intends to use it have access to it. Along with creating a VPC, Ian made sure to only allow requests from the VPC to hit the database. Sean has done a majority of the testing for Patrol in both

creating the test plan and carrying out the tests. The testing plan covers connectivity, end-user, functionality, and exploration testing. Due to our testing being done in iterations, our group was able to test and then go back to fix any bugs that were found. These bug corrections were coded by Kramer and then re-checked by Sean in the next iteration of testing.

Appendix A

References

Key, Jennie. “Years of Police Records Gone: 'No. 1 Concern Is That This Would Affect a Criminal Case'.” *Cincinnati.com*, Cincinnati Enquirer, 7 Dec. 2018, www.cincinnati.com/story/news/crime/crime-and-courts/2018/12/07/could-compute-failure-jeopardize-suburban-cincy-police-cases/2224939002/.

“Office of Justice Programs.” *Bureau of Justice Statistics (BJS)*, https://www.bjs.gov/index.cfm?ty=tp&tid=702.

Appendix B

Poster