## horizontal line



Children's Home of Poughkeepsie Deployment Plan

December 5th, 2018

**─**

Nick De Paul

Marc Christensen

Michael Gutierrez

Brendan Van Allen

Mark Miller

**Table of Contents**

Project Overview… pg. 2  
Deployment Plan - Server Configuration… pg. 3  
Deployment Plan - Database Initialization… pg. 3  
Deployment Plan - File Automation Script… pg. 5  
Deployment Plan - Hosting the Application… pg. 8  
User Interface - Login Page… pg. 9  
User Interface - Homepage… pg. 9  
User Interface - Record Upload Page… pg. 12  
User Interface - Data Report Page… pg. 13  
User Interface - Chart Page… pg. 18  
User Interface - Help Page… pg. 19  
User Interface - Logout… pg. 20  
User Interface - Admin User Management Page… pg. 20  
User Interface - Admin SQL Entry Page… pg. 21  
Security - Unauthorized Access… pg. 22  
Security - Password Requirements… pg. 22  
Warning… pg. 23  
Appendix… pg. 24

**Project Overview:**

The intent of this project is to create, config, and deploy a working application to provide the Children's Home of Poughkeepsie (CHP) with automation of children data reports with an emphasis on ease of use and data visualization for research and other information specific needs.

The first goal is to aid CHP with a convenient interface to simplify the children data reporting on incident types, programs, placement, and time period. We plan to implement various coding techniques to allow various types of users to easily create, review, and submit incident reports for the children.

The Second goal was to allow for clear and accessible data to be used for research, publications, and papers. Along with easier incident reporting, we are providing a simple and easy to read presentation of children data. This includes but not limited to incidents within a program, incident per child, and overall incidents at CHP. CHP uses the incident extensively to publish papers, conduct research and publish to improve the effectiveness of programs offered at Children homes and to minimize the overall incidents of these children.

**Deployment Plan:**

**Hardware:**

**Our Server Specifications:**

We used a IBM System x3550 M4 Server with 4 Cores, 16 Gigabytes of RAM and 500 Gigabytes of Storage. It is our expectation that whatever hardware is installed at The Children’s Home of Poughkeepsie (CHP) will be compatible with what we used for our development environment at Marist College.

**Server Configuration and Install:**

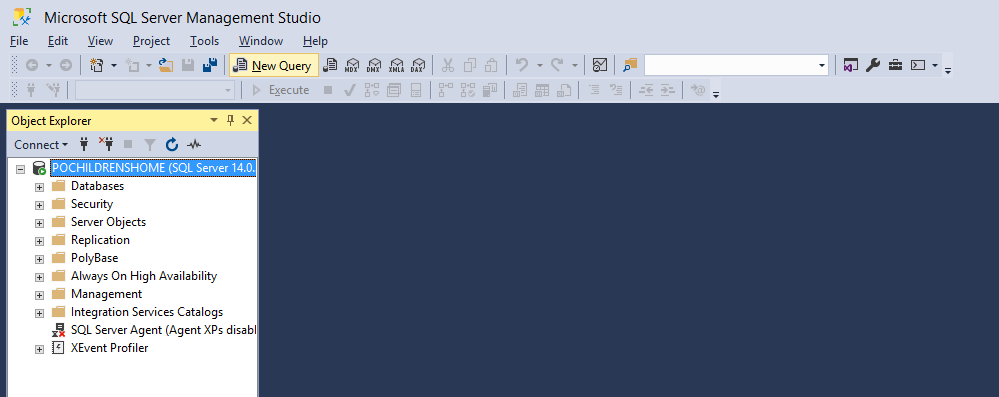
We are assuming that the IT department for CHP has already installed and configured a working server or virtual machine that will have full capability of running the software we have designed for CHP. The requirements for the application include Windows Server 2012 R2 as the server operating system and Microsoft SQL 2017 as the database. These requirements were as per the CHP IT department, and it was our goal to ensure our application ran in this environment so that it was easier for the CHP to implement just our application software instead of having to build and configure new hardware, or install and run a new operating system and middleware.

**Software:**

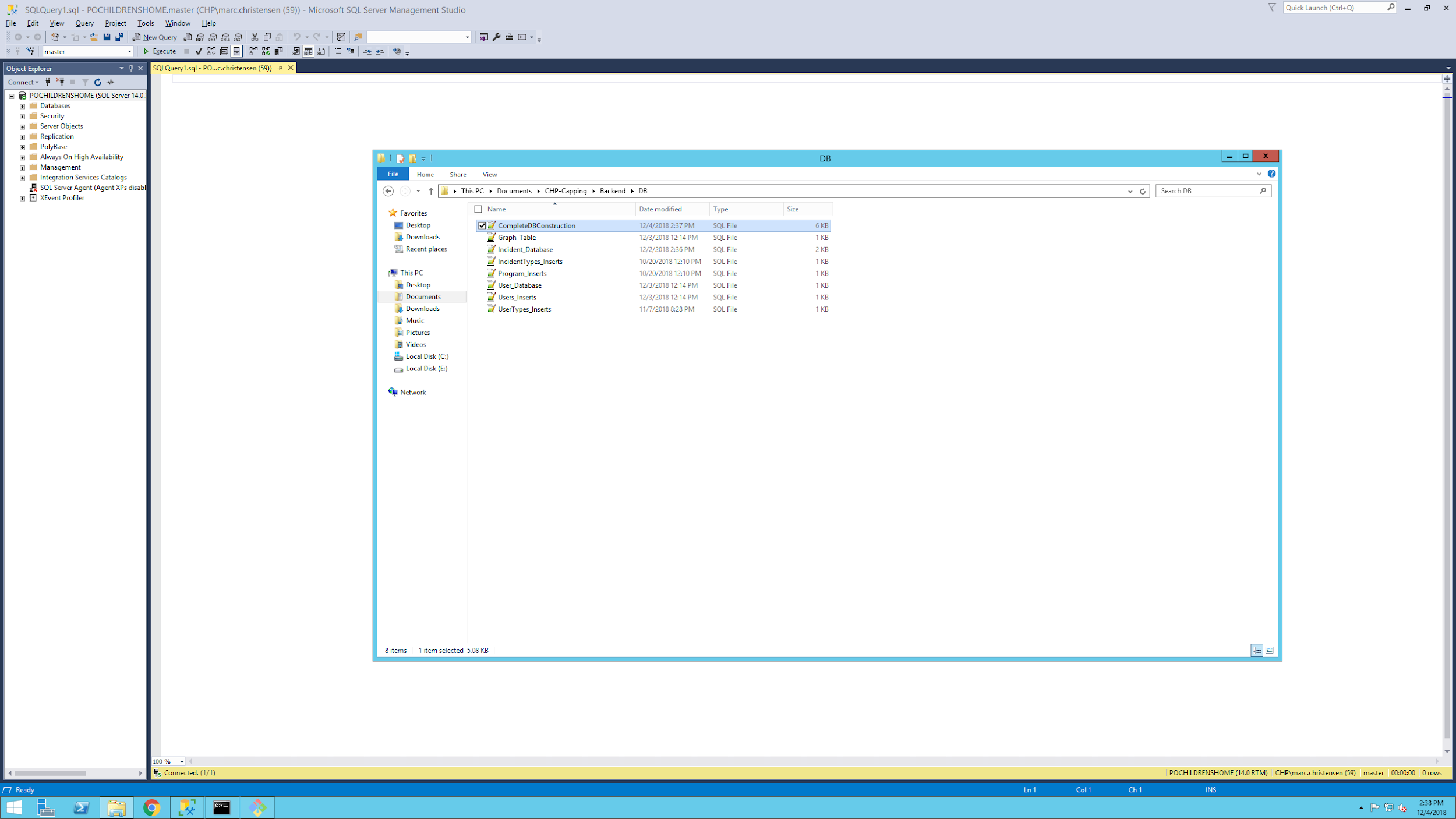
**Database Initialization:**

We also assume that since Philip has installed MS SQL on multiple servers, that he already knows how to instal MS SQL. Using MS SQL 2017, the supporting database for our application can be built by copying and executing the CompleteDBConstruction SQL script found in the Backend folder of the project. The entire database should be built before executing the associated insert SQL commands, or any other inserts/entries via the system, as some tables require others to utilize their primary key as a foreign key. The construction script includes the four insert scripts used to populate specific tables with entries needed for the basic use of the system. These include UserTypes, Users, IncidentTypes, and Programs. Without these basic inserts, the system will be not be assessable/operational. If there are any issues with building/maintaining the database, there are commented out SQL commands at the beginning of the construction script that can be executed to successfully drop all tables from the database, so that they can be rebuilt in the following commands. Every attribute that is serializable in the database is defined to start at 1 and increment by 1 with each entry, using MS SQL specific syntax.

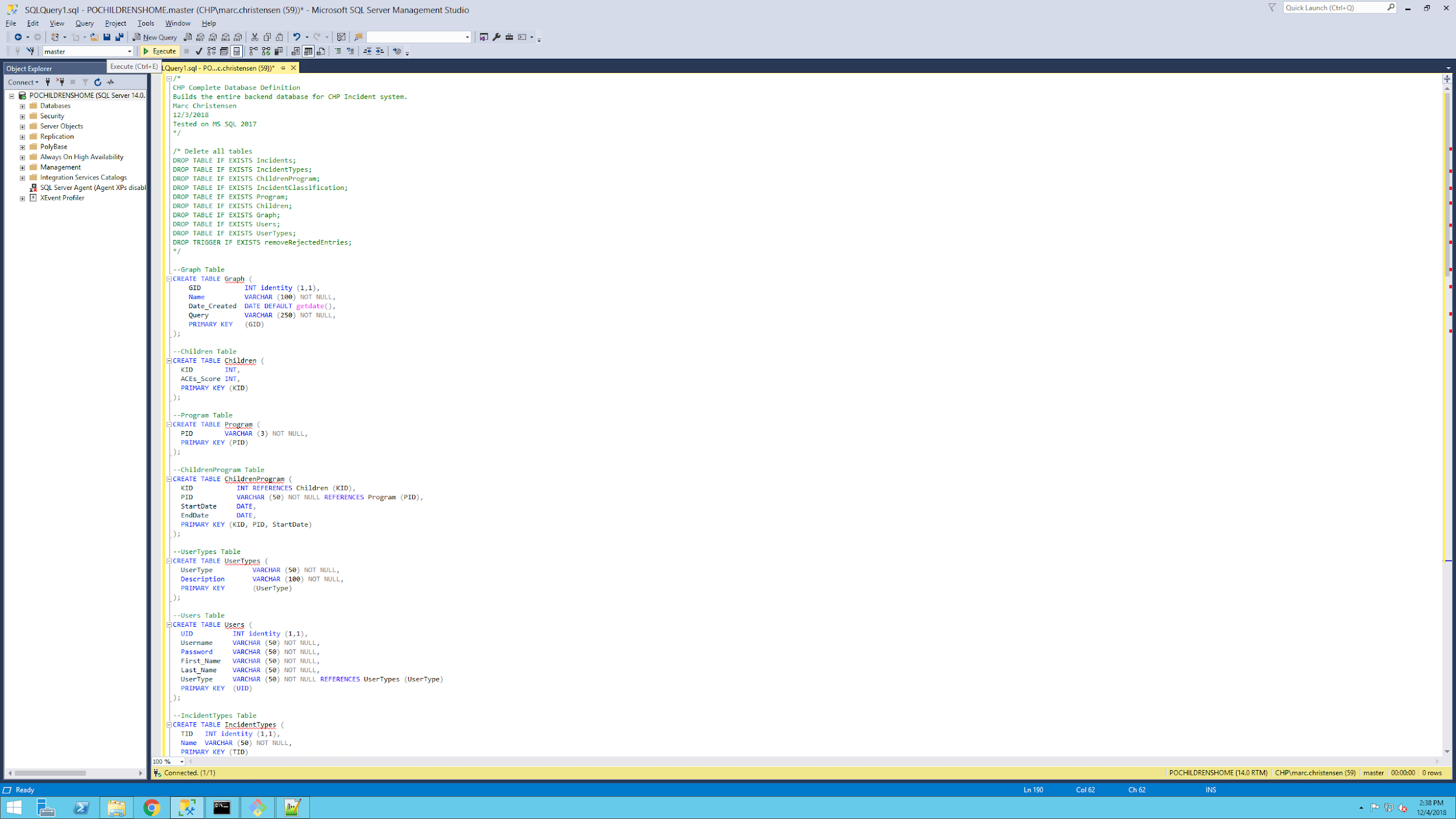
Once in MS SQL, create an empty database, with authentication options that meet the specific security requirements. Then open a new query page using the New Query button.



Afterwards, navigate to CompleteDBConstruction.sql in the Backend folder of the system, open it in a text editor, and copy the entire document.



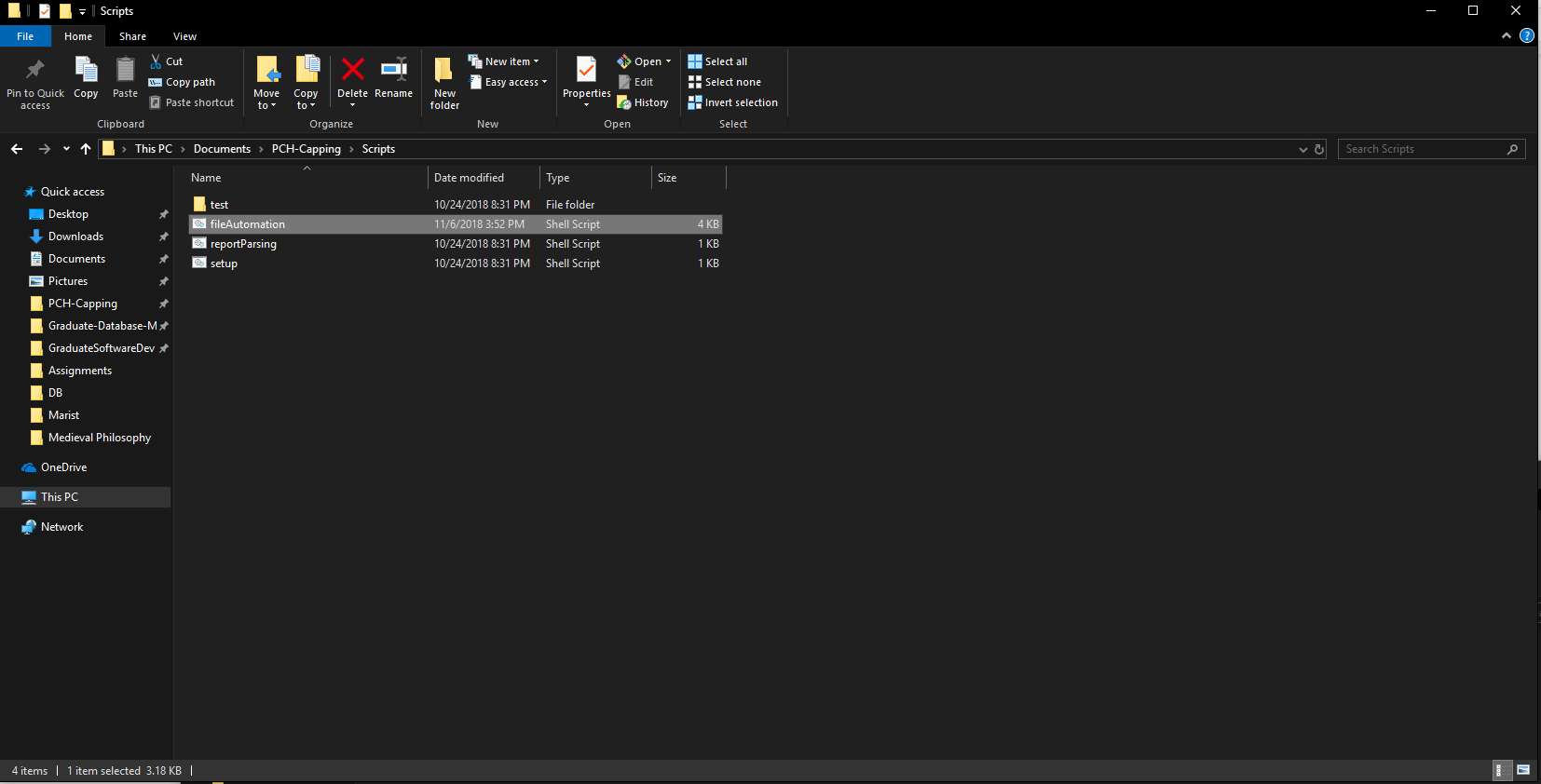
Finally, paste the SQL commands into the query page and click execute. This will build the entire database with tables, inserts, and the required trigger. The commented out section of SQL at the top will drop everything and allow the subsequent commands to recreate the database, if needed.



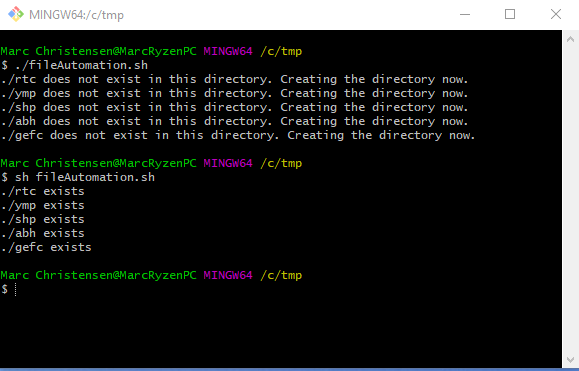
**File Automation Script:**

The fileAutomation.sh is a bash shell script found in the Scripts sub-directory of the system. This script automates the creation of supporting directories, to contain uploaded behavioral reports, and Archives, to contain reports that have been entered into the database. The sub-directories created by the script reflect the several programs being tracked by the home, which allows their associated information to be entered into the database correctly. Each directory has an Archive directory, which ensures the information is not added to the database twice and serves as a last-means backup to restore the database. This script calls the parsing script to get the pertinent information from each report in all the programs. As the parsing script completes successfully and generates a temporary database entry for each uploaded record, the script then moves each record into the programs respective Archive directory.

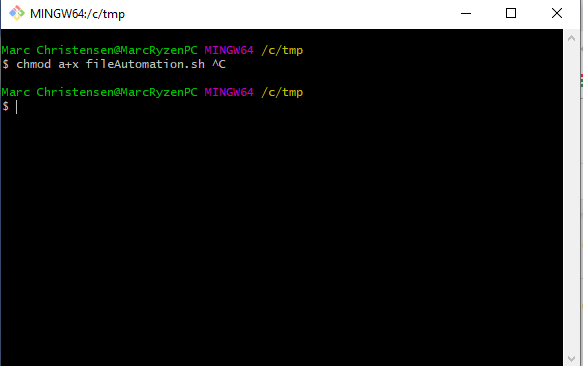
The fileAutomation script can be found in the Scripts folder of the project and can be copied anywhere to create the necessary folders.



To run the script simply type “./fileAutomation.sh” or “sh fileAutomation.sh” in the Git shell, when your working directory contains the script. If the script has already been run, it informs the user that the file structure has already been created, as seen in the second running of the script.

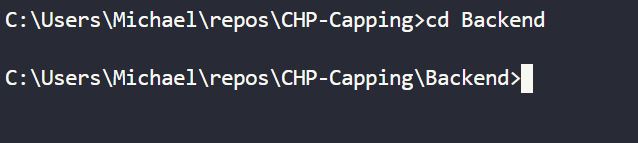


If you run into issues where the script will not run, add execute permissions for all users to the script using “chmod a+x fileAutomation.sh” and it will be allowed to run.

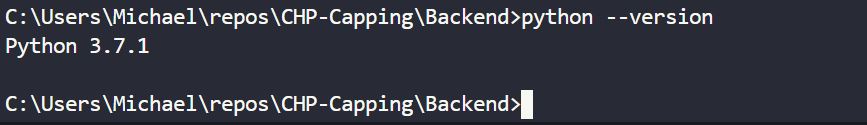


**Hosting the application using Server.py:**

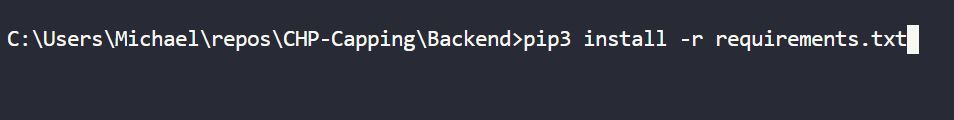
The server.py script is the script used to actually run the entire server hosting the app. It is located under the Backend directory which is a folder inside the main project folder. Open a command prompt and navigate to the Backend folder.

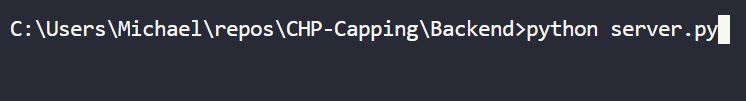


Before turning on the web server, check if Python3 is installed by running the command ‘python3 --version’, and if it throws an error, it means it is not installed.



Once installed, run the command ‘pip3 install -r requirements.txt’ to install all the necessary dependencies needed .

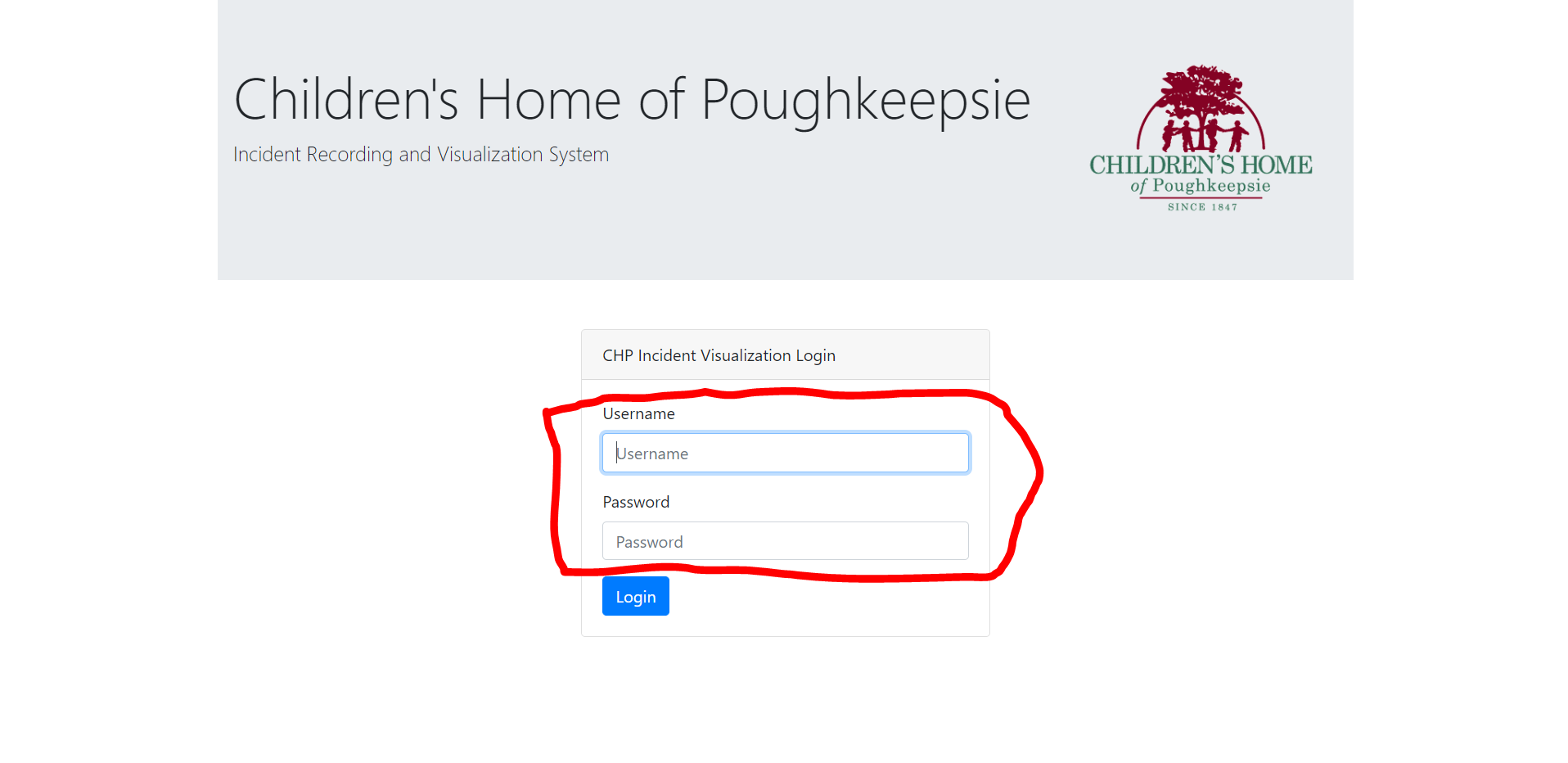


Finally run the command ‘’python3 server.py” command and the web server will be turned on. 

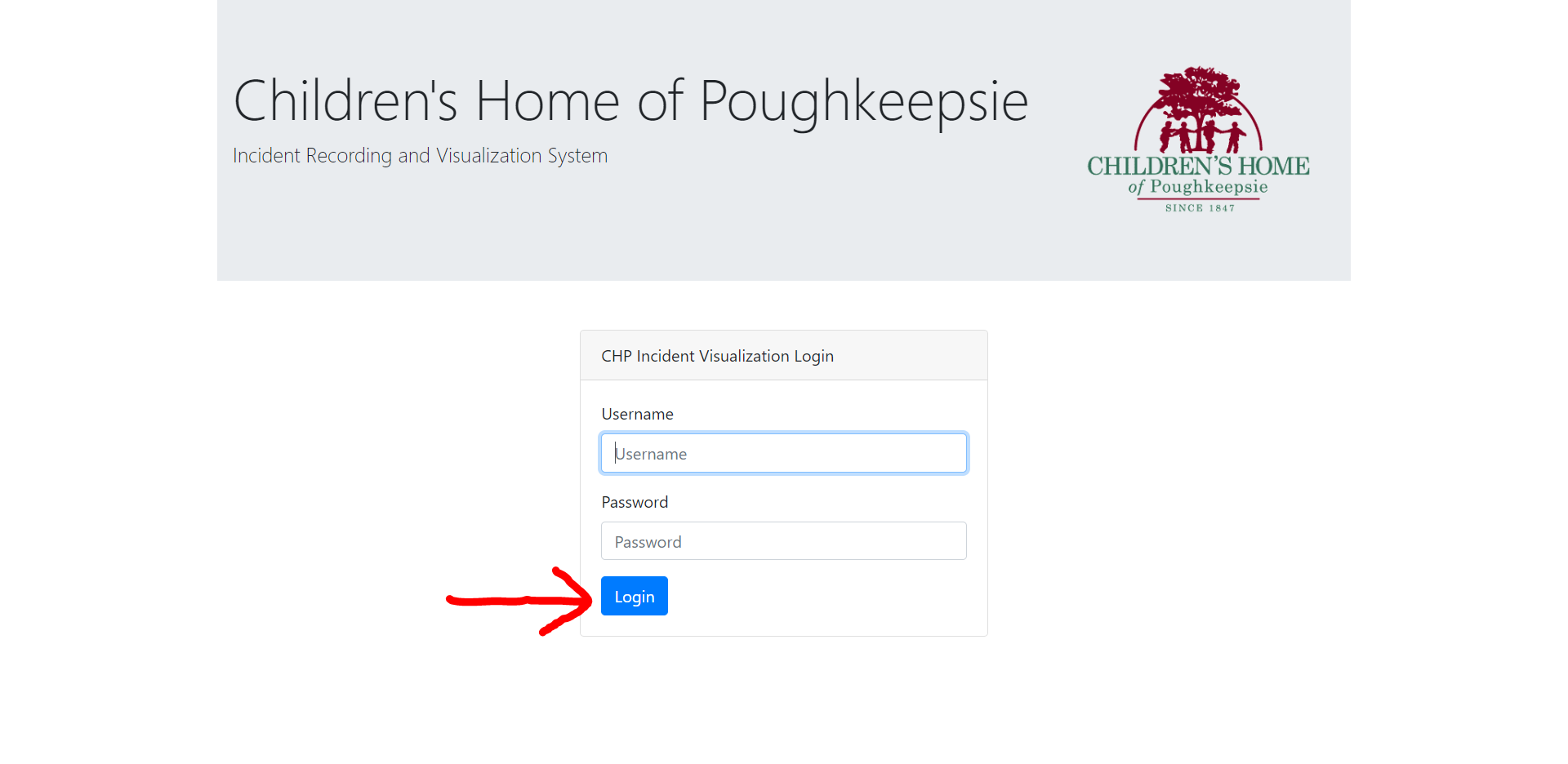
**User Interface**

**Login Page**

When you first access the application, it will bring you to the Login screen. This is where you will type in your valid credentials. First type in your username or password.

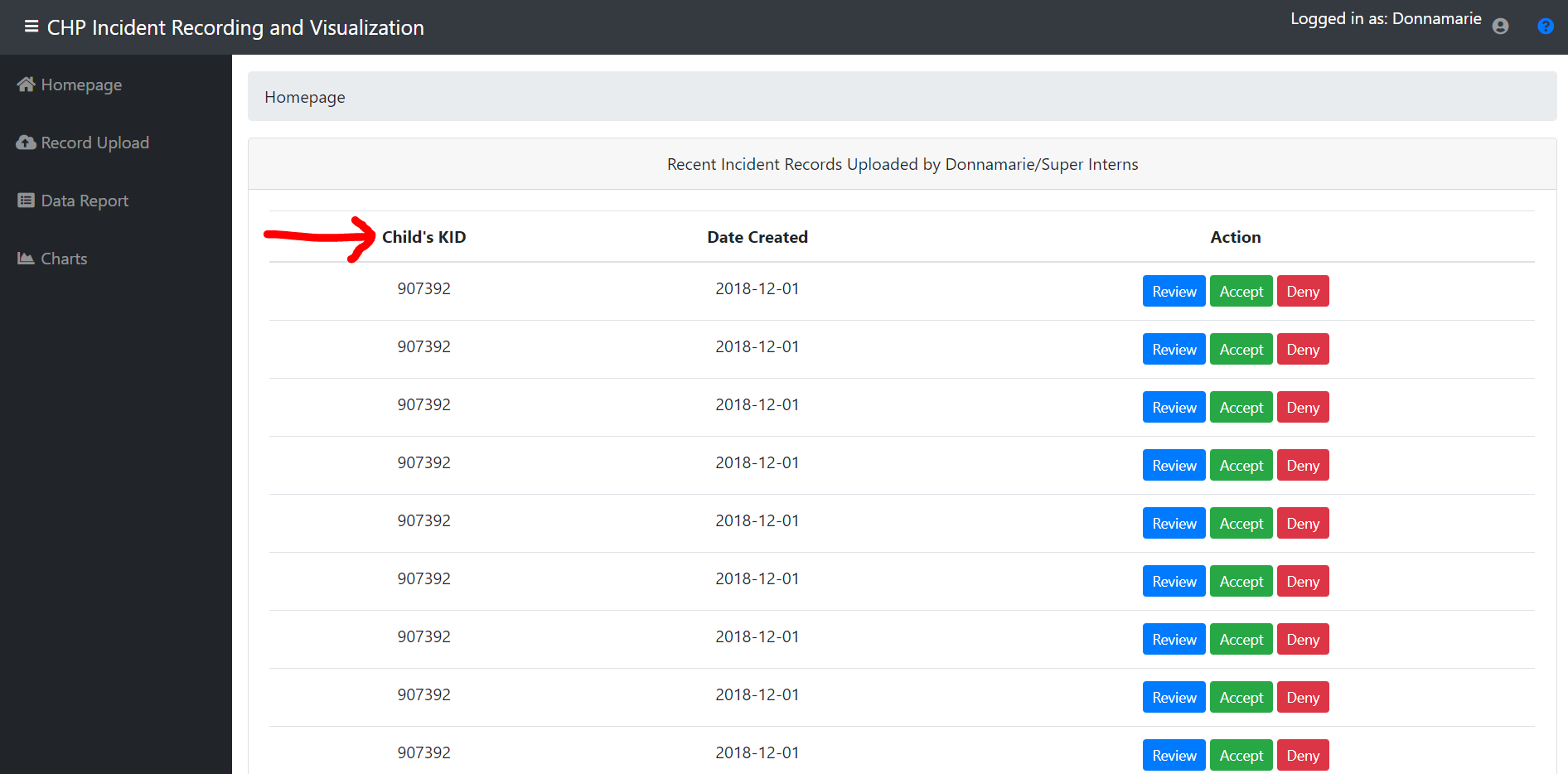


Once you type in your username and password either press the “Enter” key on your keyboard or press the Login button to Log into the application.

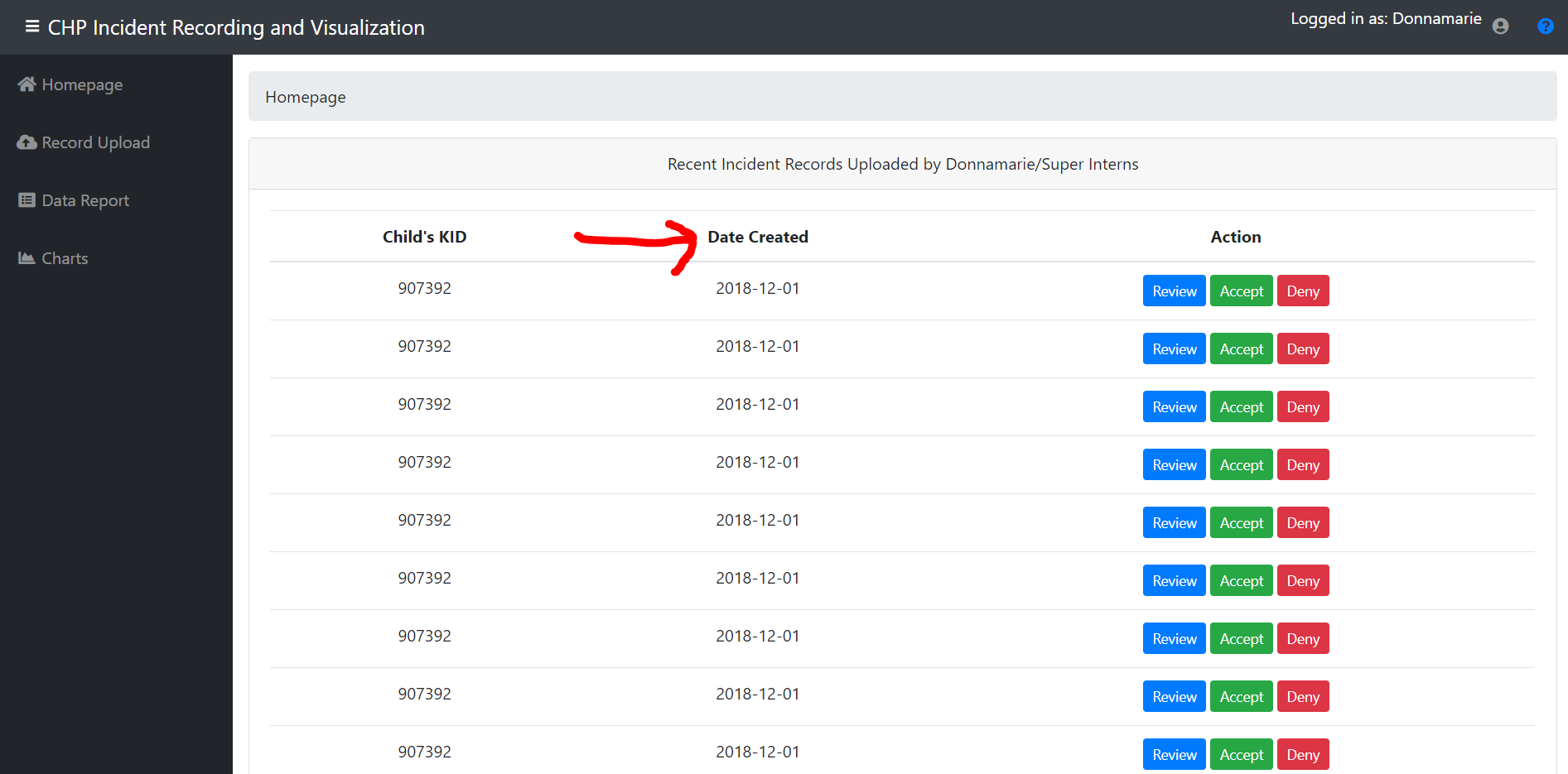
**Homepage**

When you first Log into the application it will bring you to the Homepage. This is where you can view the incident reports that the interns have submitted for review. Also, what the full user and super interns have submitted will be here as well.

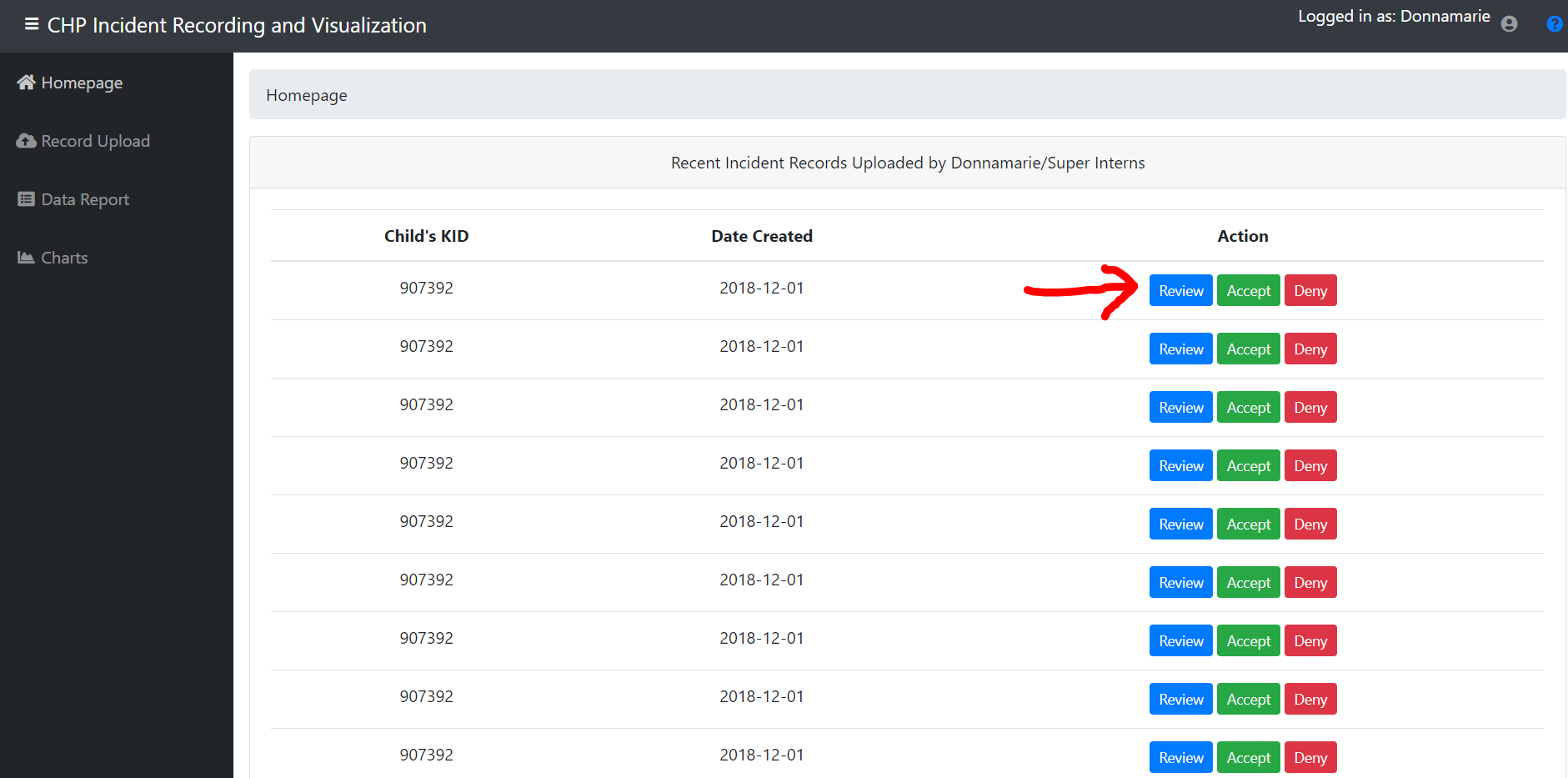
You will see the child’s KID



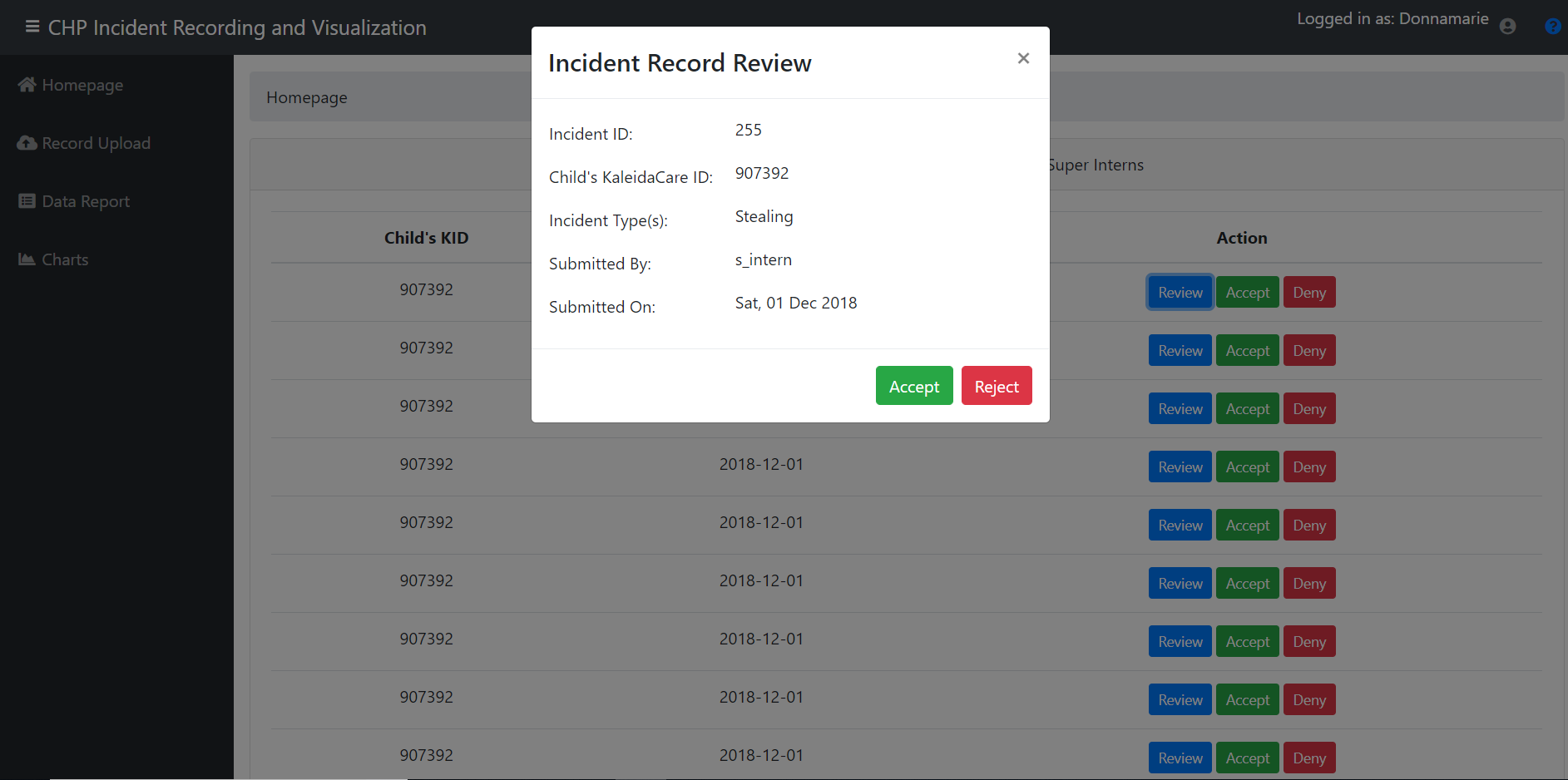
The Date that the incident report was submitted.



This is where you can press any of these three buttons. Click Review and it will show you the Incident Report. Click Accept and it will add the Incident Report to the application. Click Deny and it will remove the Incident Report.

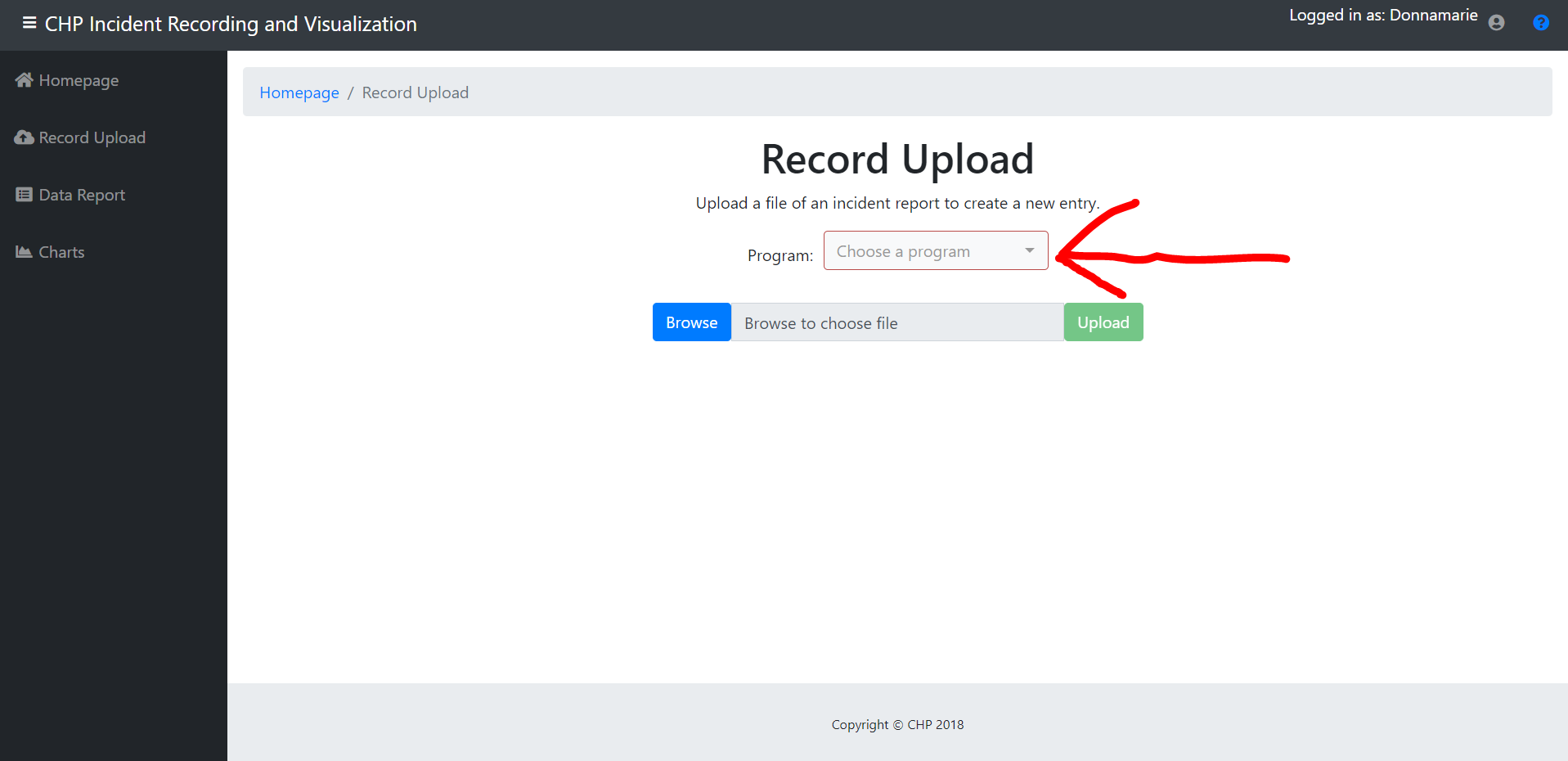


Once you click review it will show you the information in the incident. From here you can either Accept or Reject the incident report.

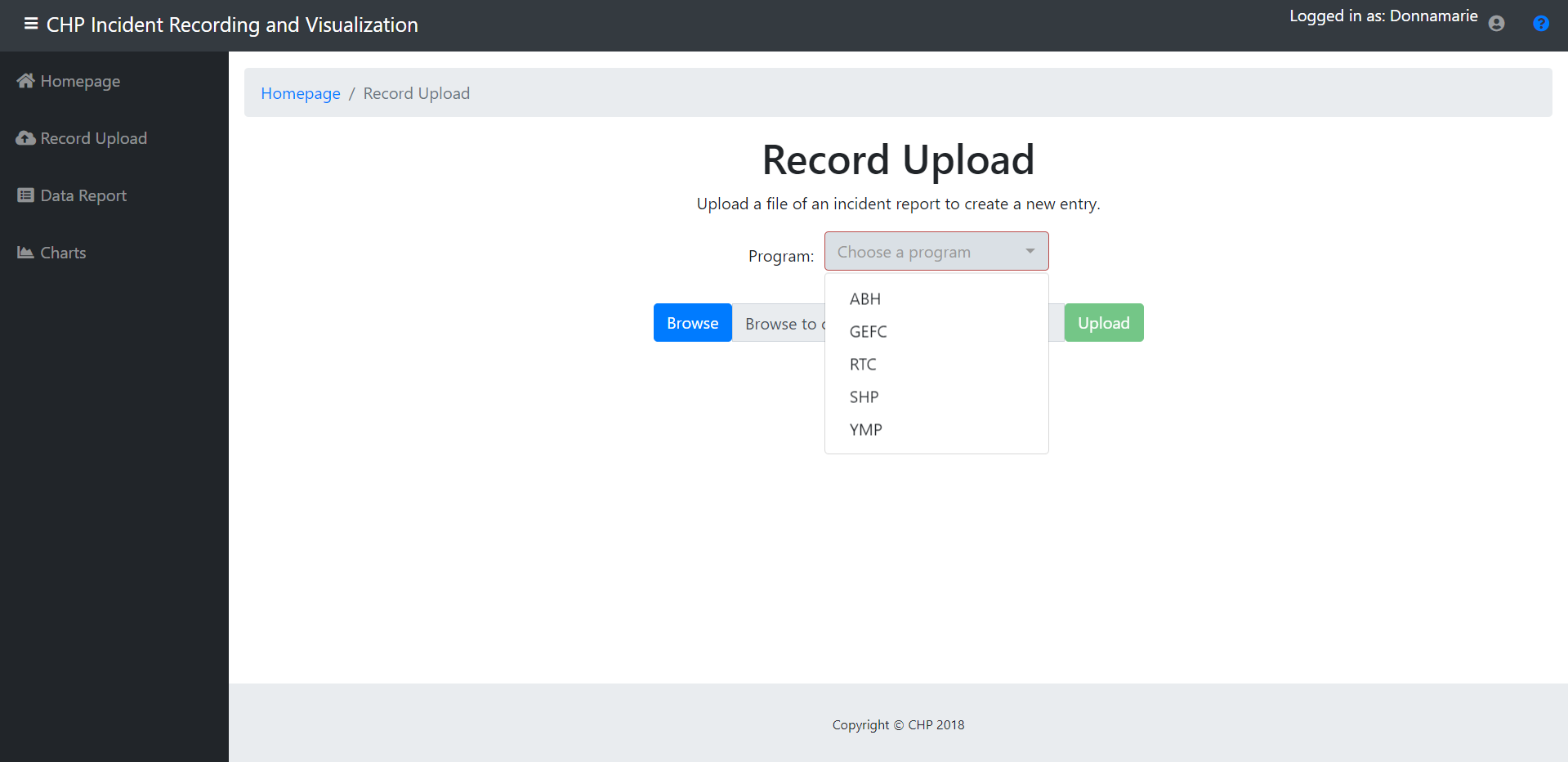


**Record Upload Page**

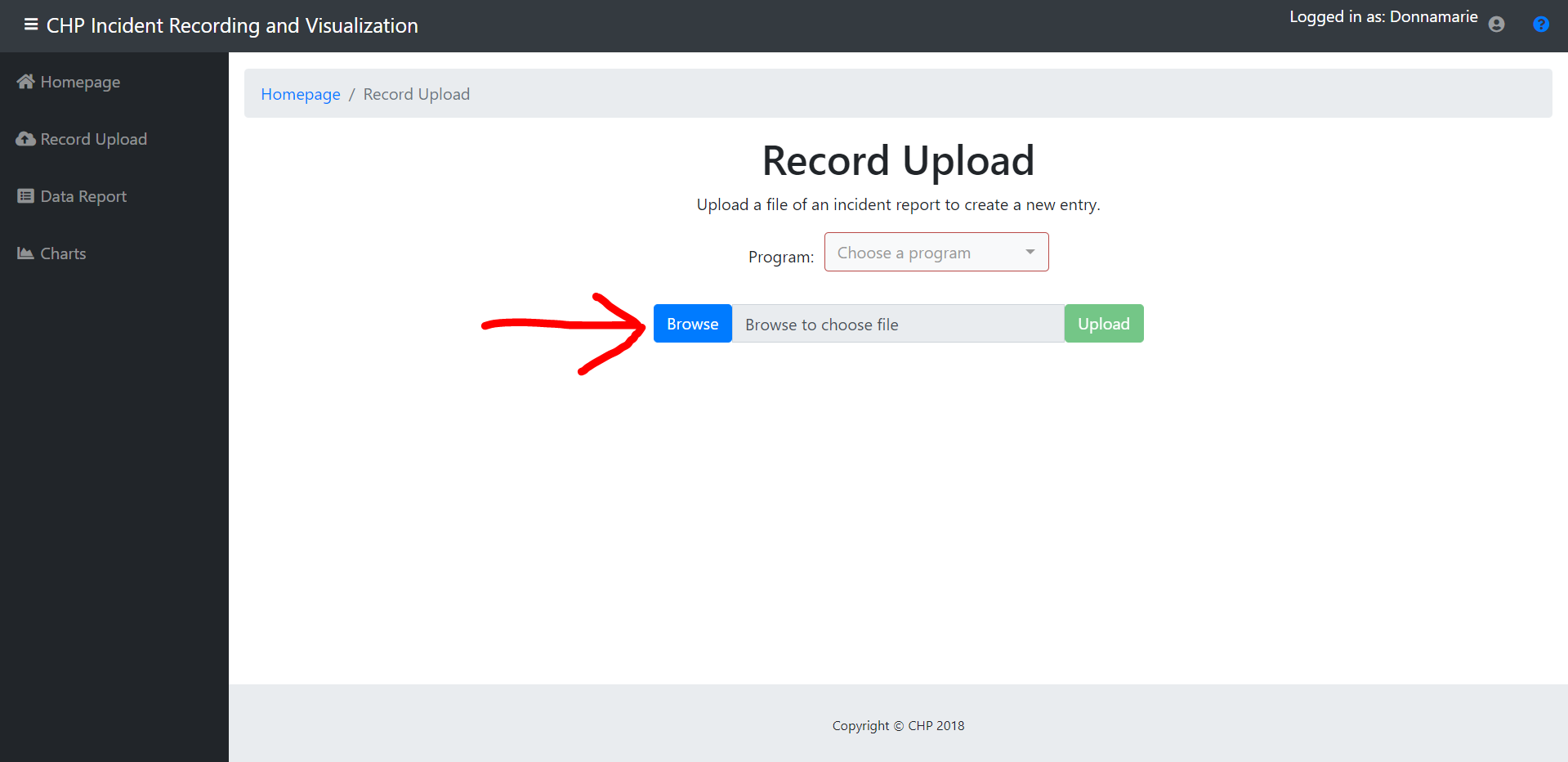
The Record Upload page will allow you to upload incident reports. First click on “Choose a program”.

****

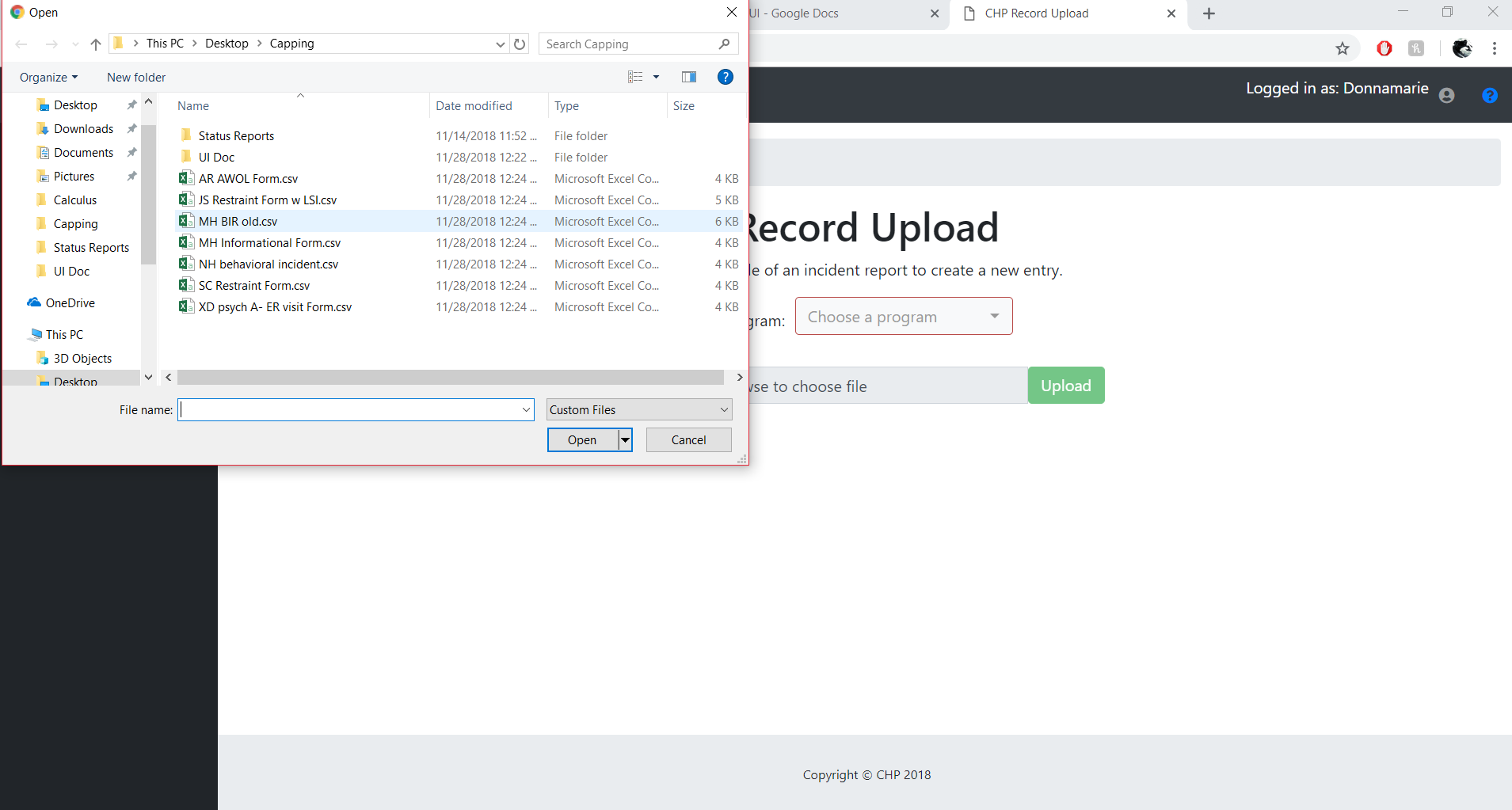
Then pick the program you are uploading the incident report from.



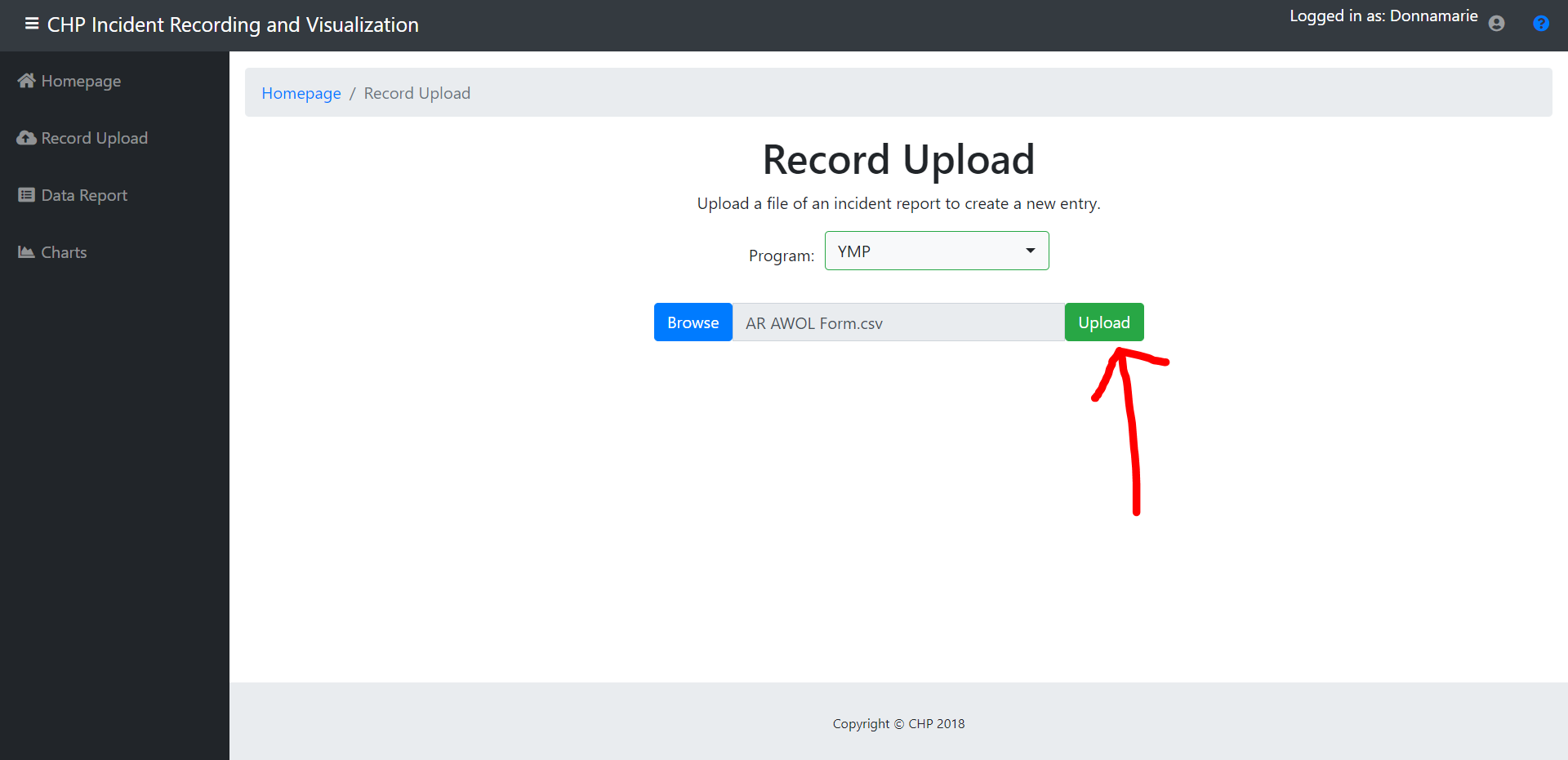
Click Browse.



Here you will pick an incident report file. Either pick a .csv or .xlsx file.

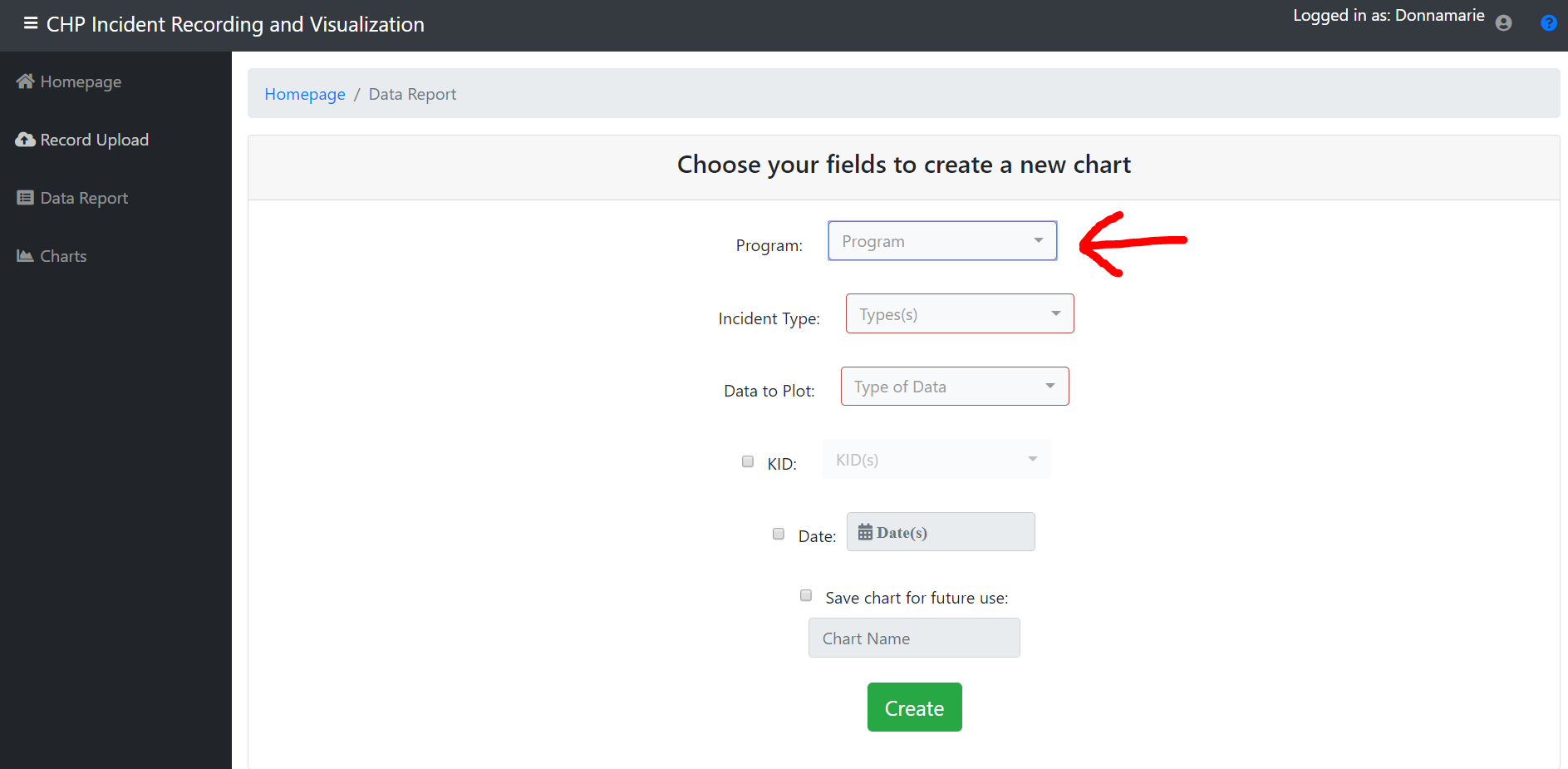


Then click Upload and it will upload your incident report to the application.

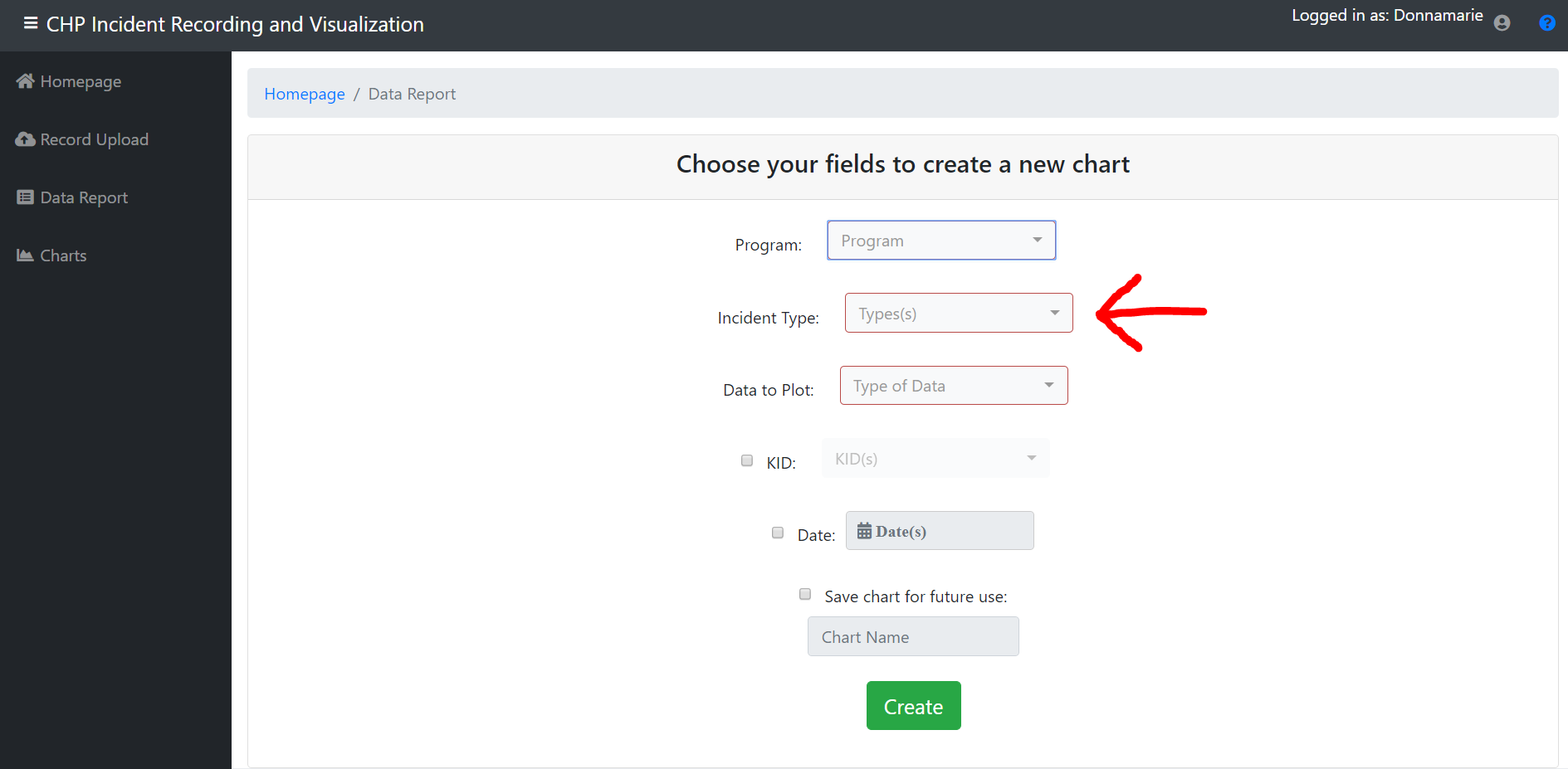


**Data Report Page**

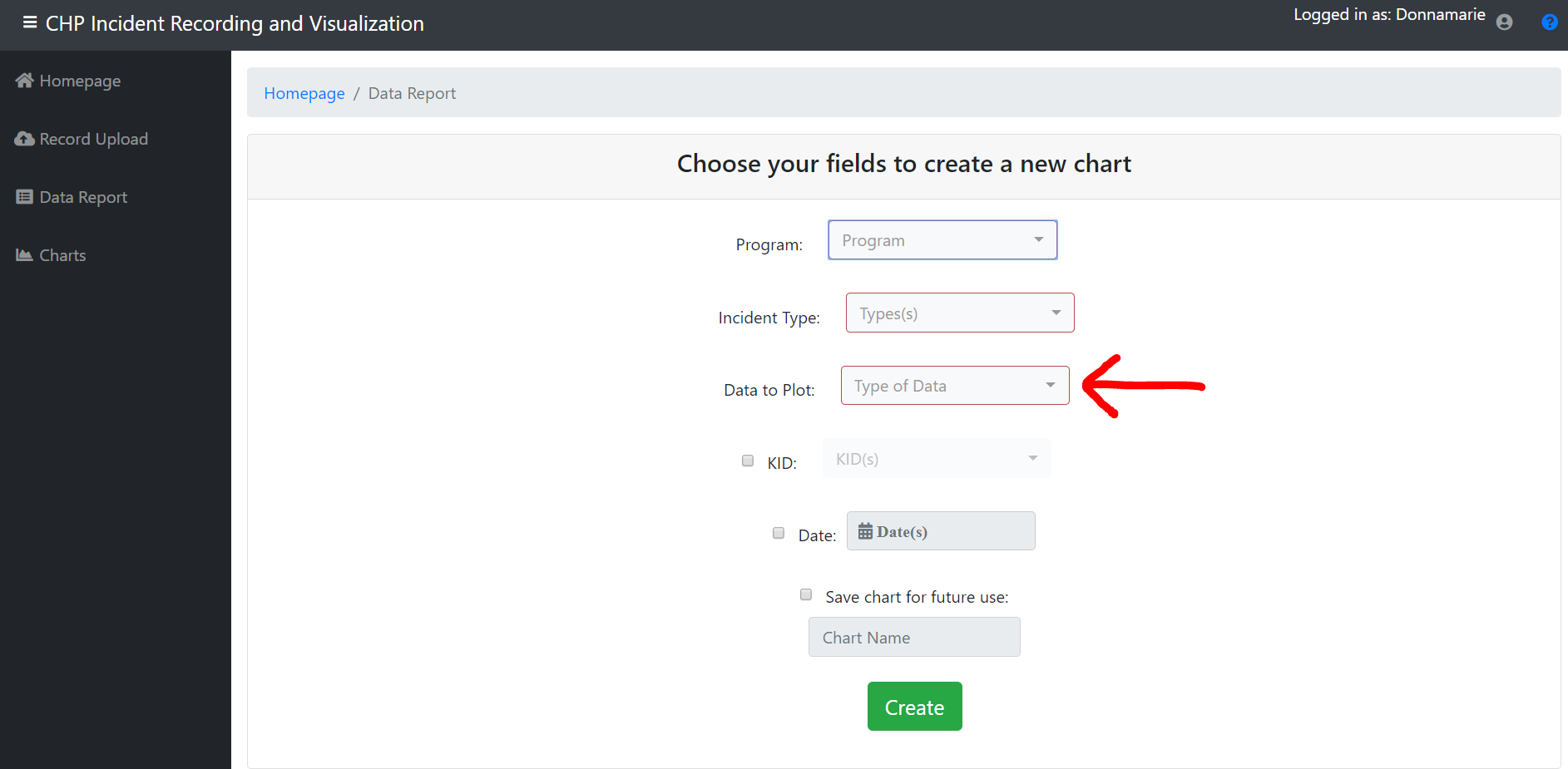
The Data Report page allows you to view charts from the various incident types. First click Program and choose a program that you would like to look at.



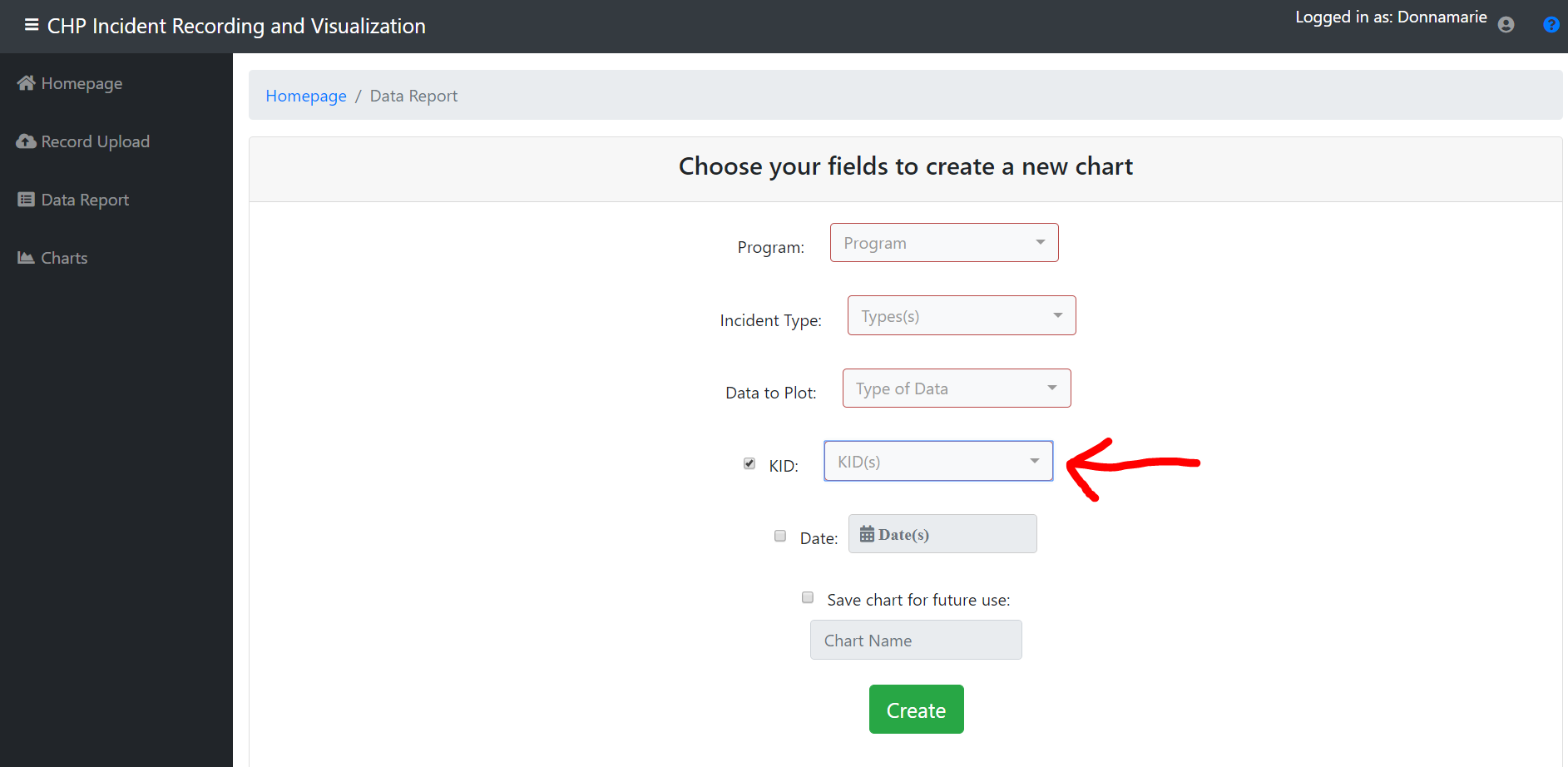
Then click on Incident Type and choose the type of incident.



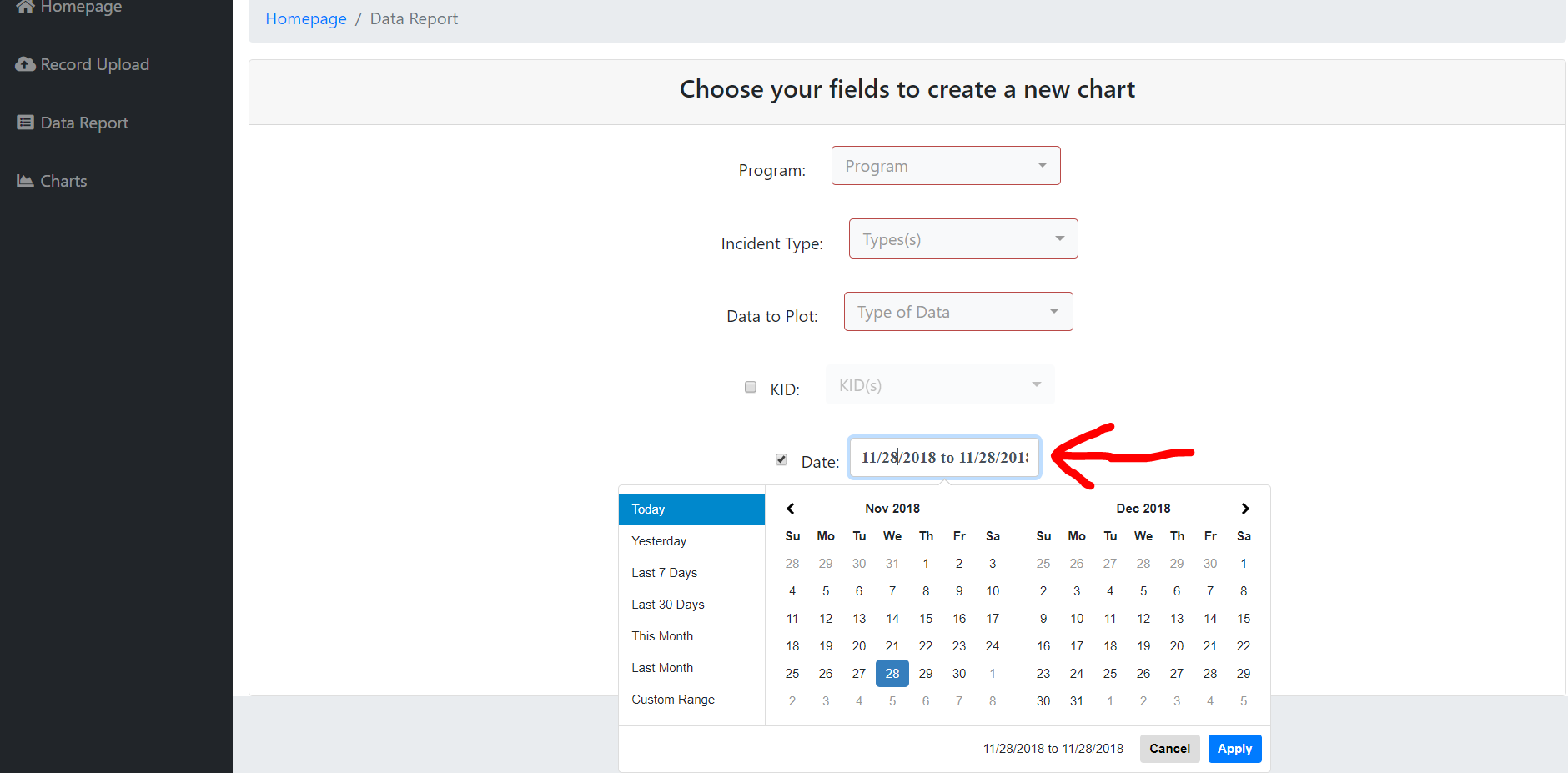
Then click on Data to Plot and pick Means.



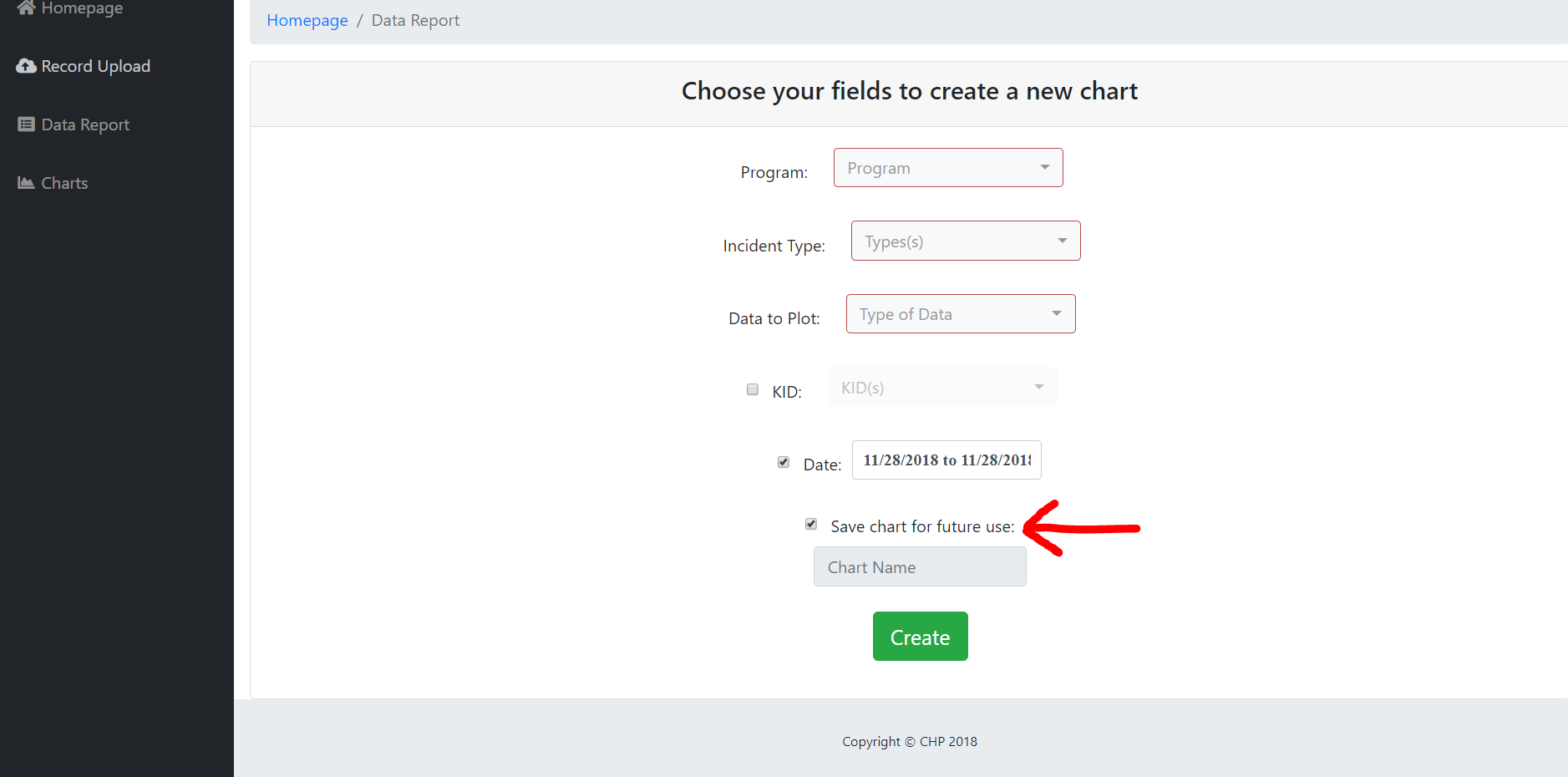
This is optional. You can click on the KID checkbox and type in the KID of the child you would like to look at.



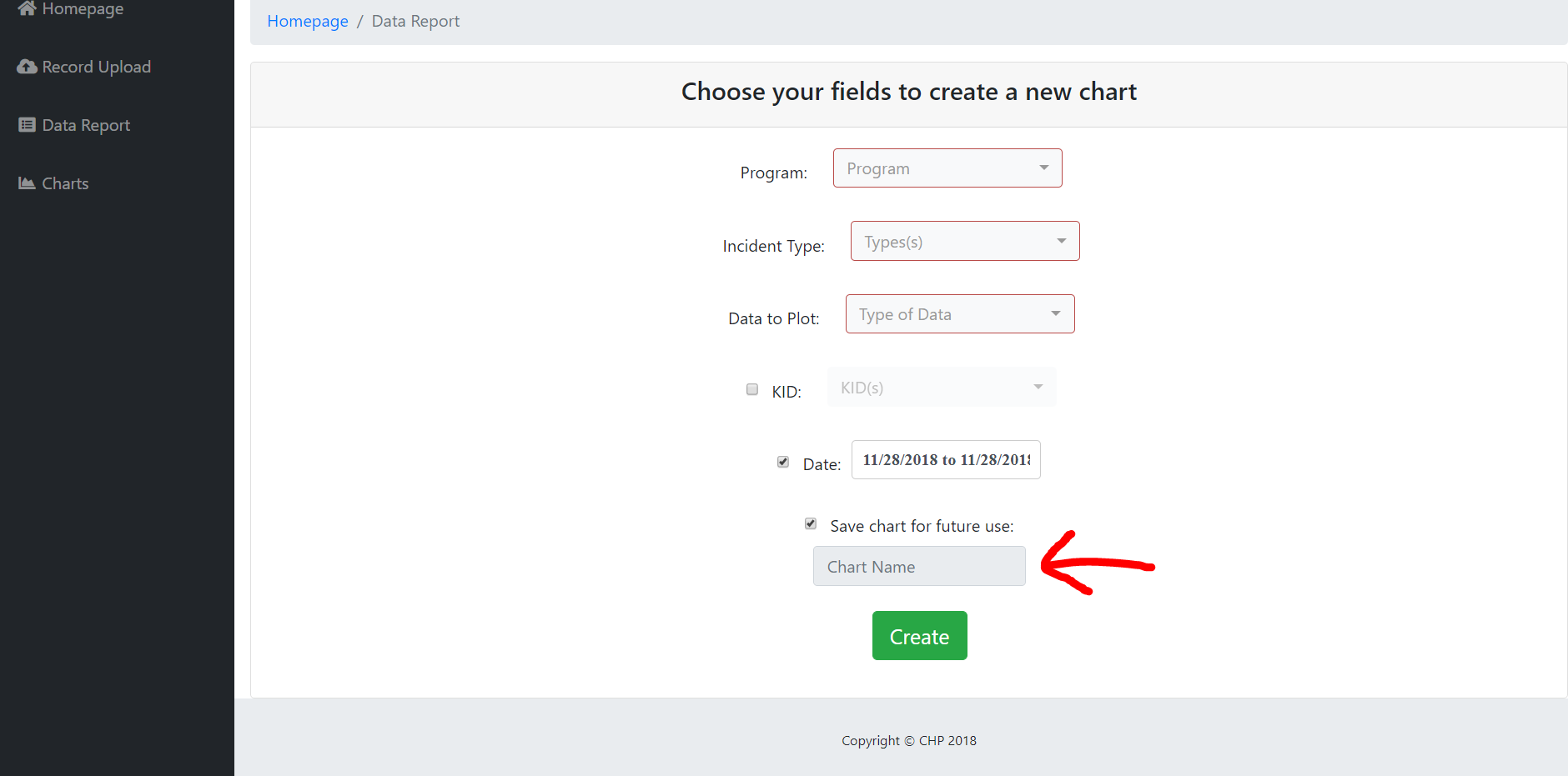
This is also optional. Click the Date checkbox and pick the dates you would like to look at.



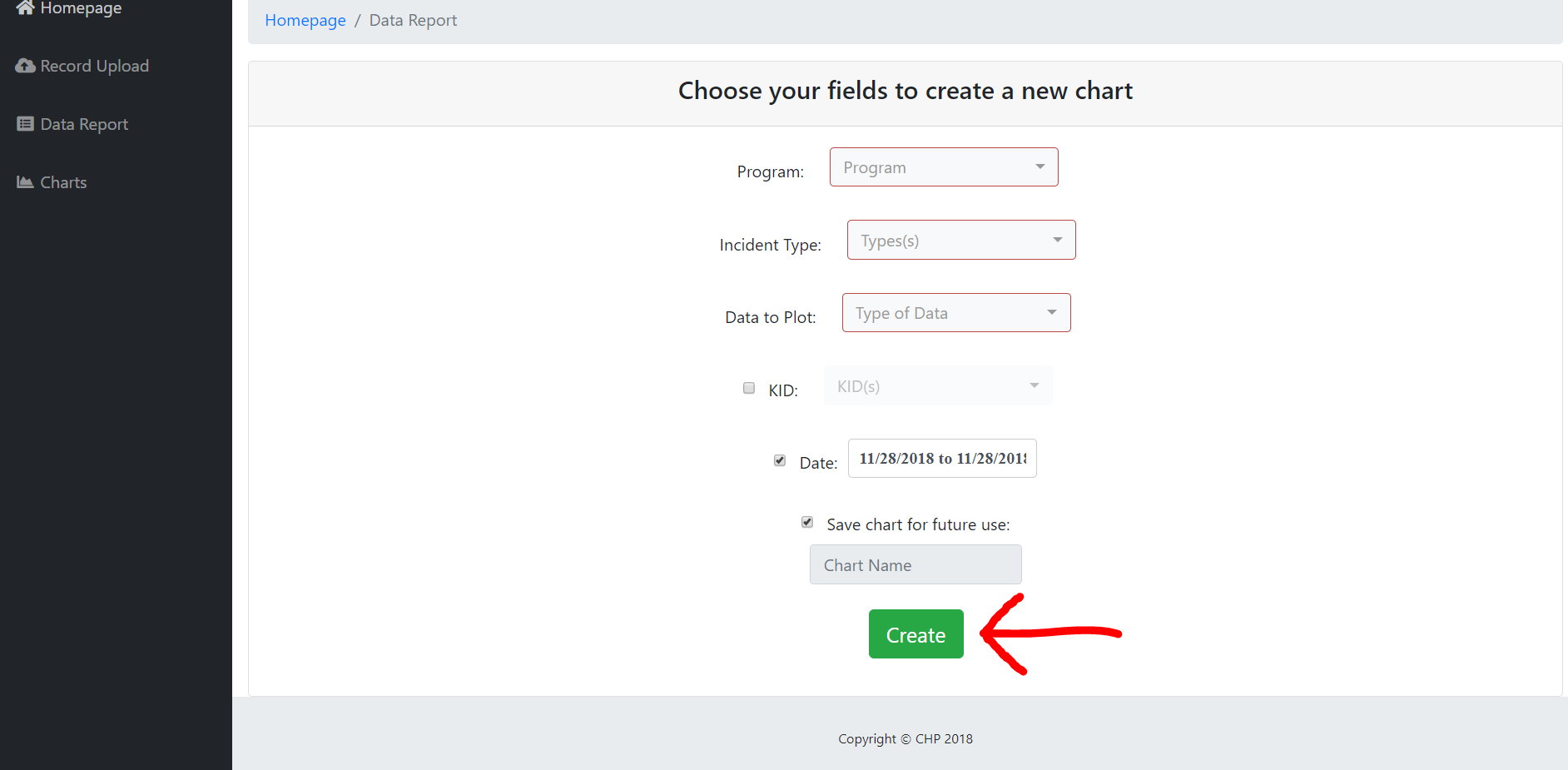
This is optional. Click the “Save chart for future use” checkbox. It will save the charts on the Chart Page.



Type in a name that you would like the chart to be called.

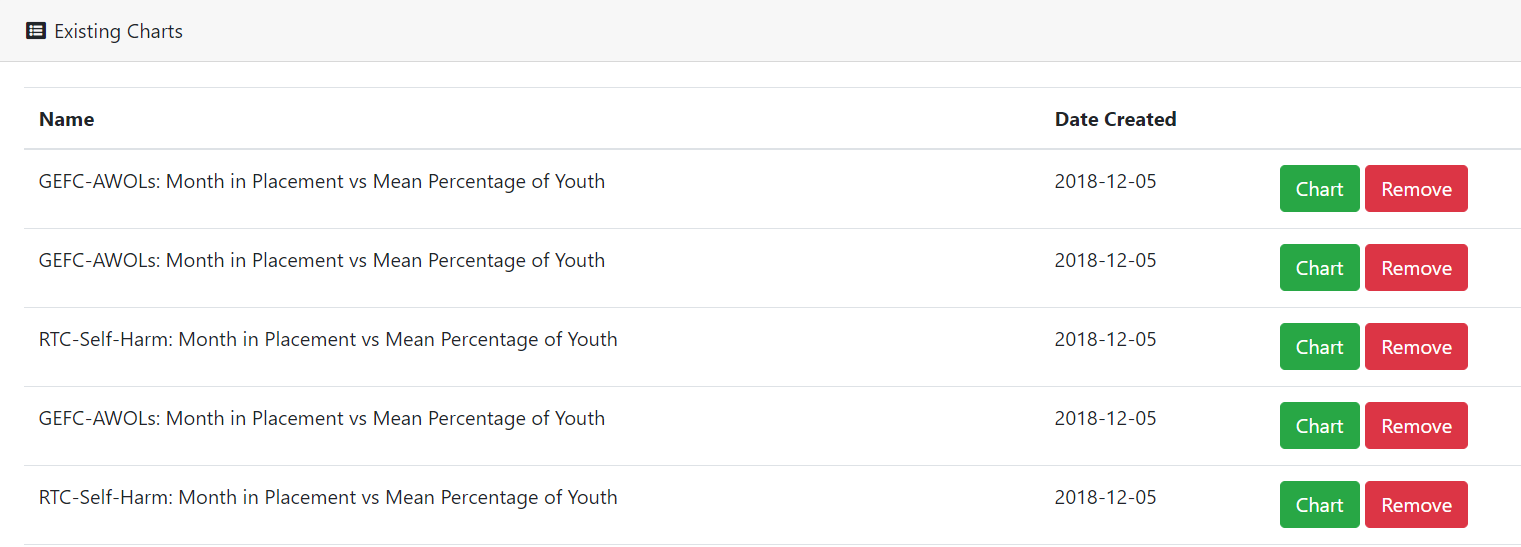


Click Create to make and display the chart.



**Charts Page**

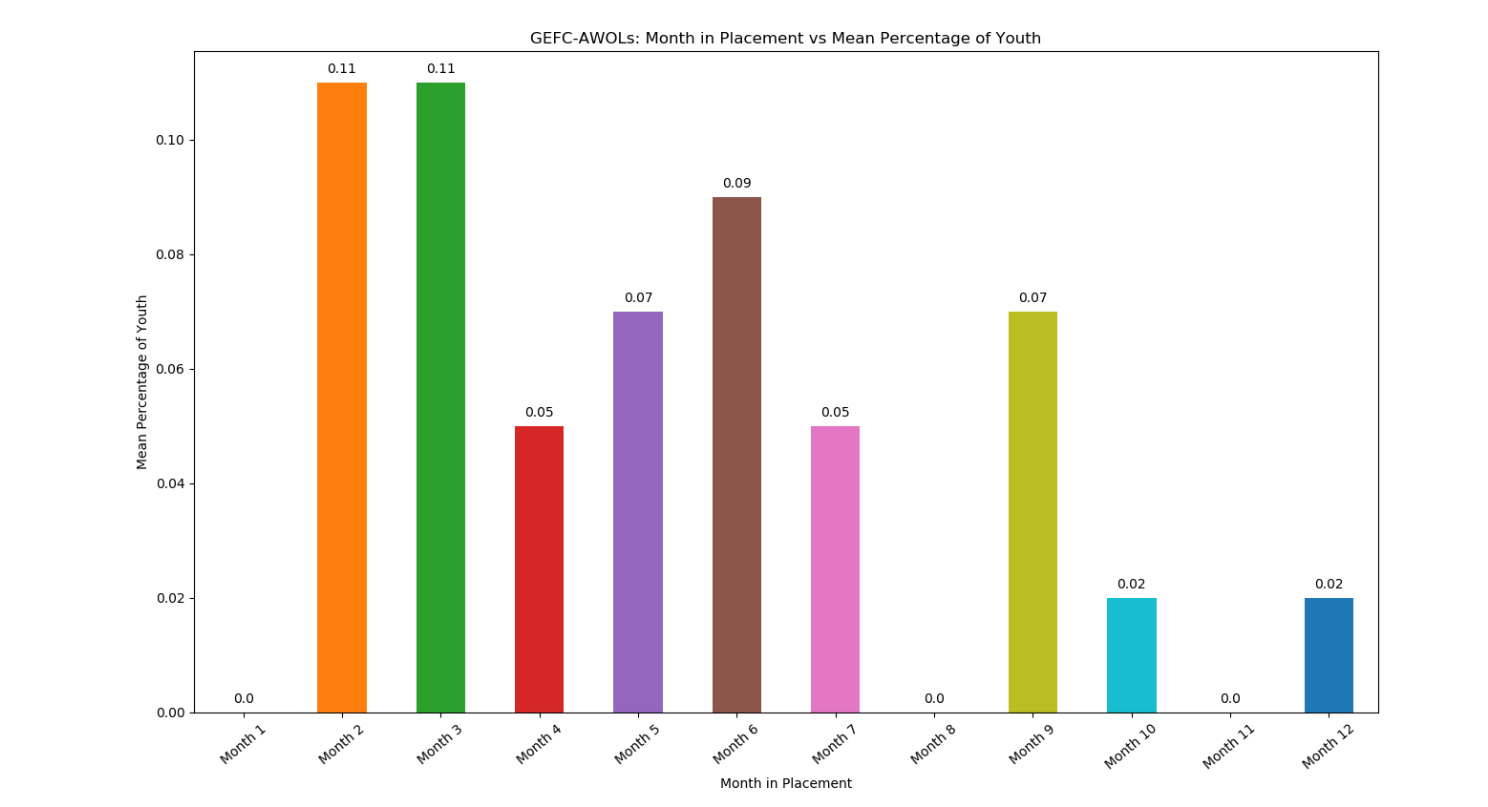
On the Charts page you can view existing charts that have been saved.



You can either click on Chart to view the chart or Remove to delete the chart.

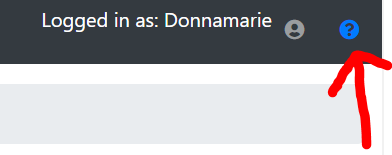
****

This is what the chart will look like.

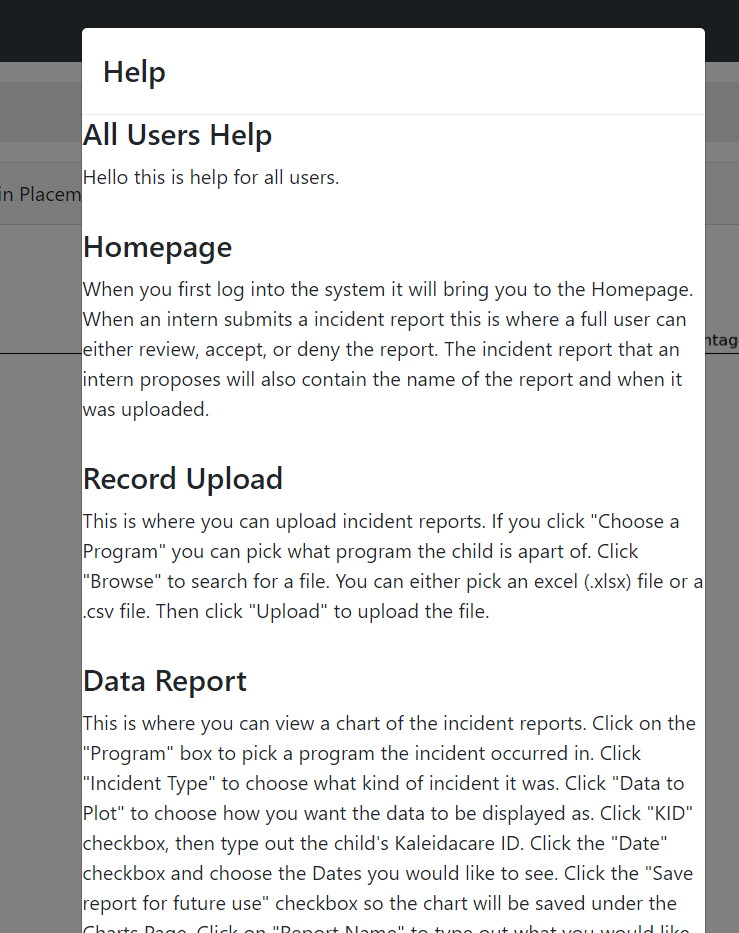


**Help Page**

If you ever need help navigating through the application, in the top right corner of the screen you will see a blue circle with a black question mark in it. Click on it.

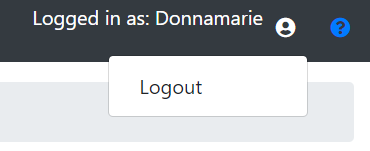


Once you click on it the Help Page will pop up.



**Logout**

When you are done using the application. In the top right corner there is a person icon. Click on the person icon.



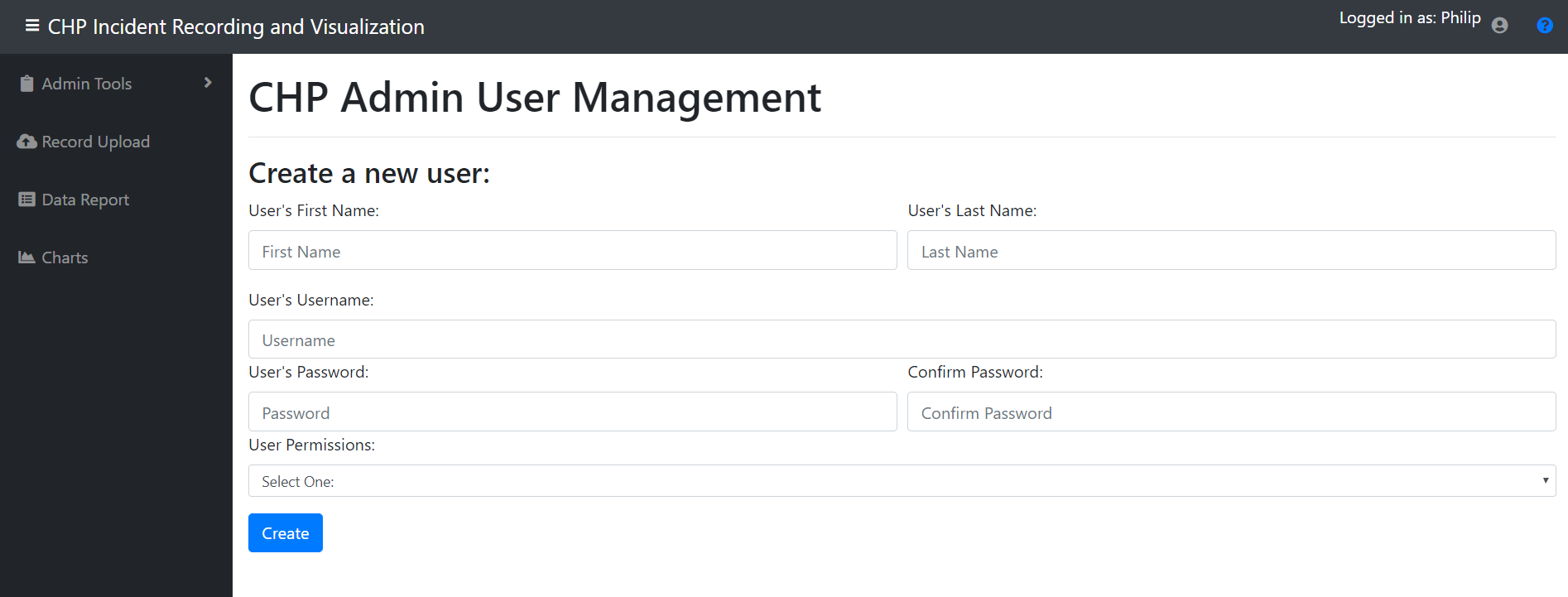
When you click on Logout it will redirect you back to the Login Page.

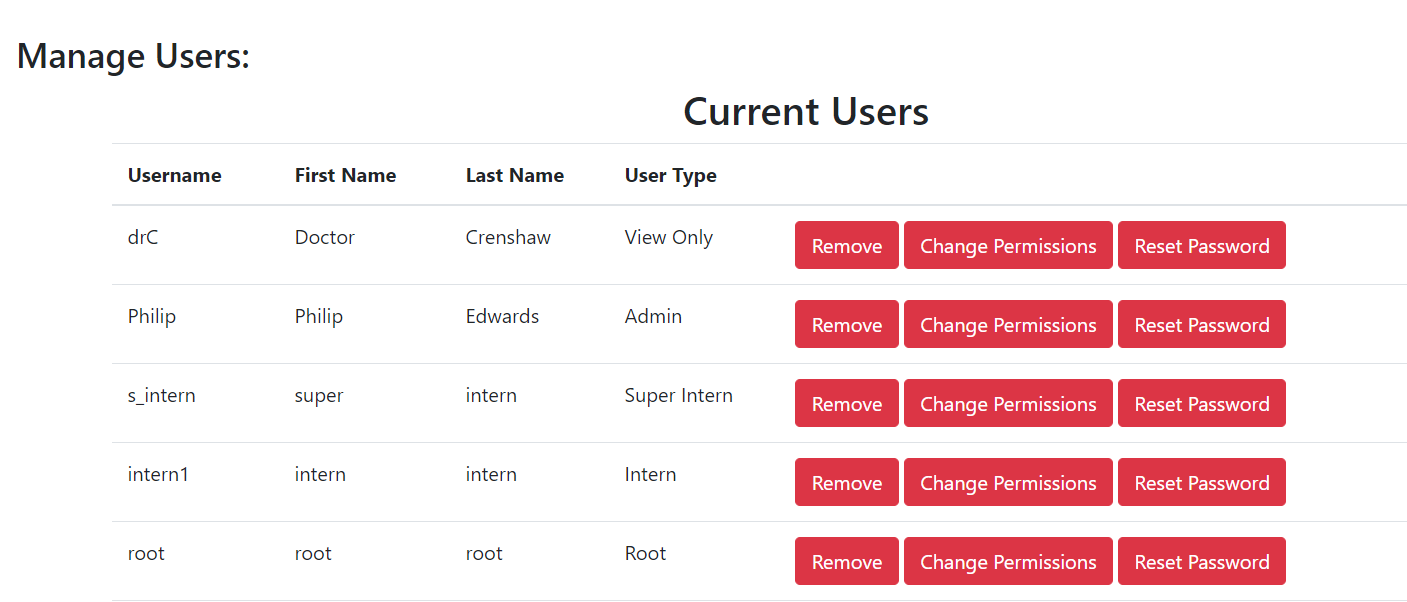
**Admin**

This is for Admin’s only.

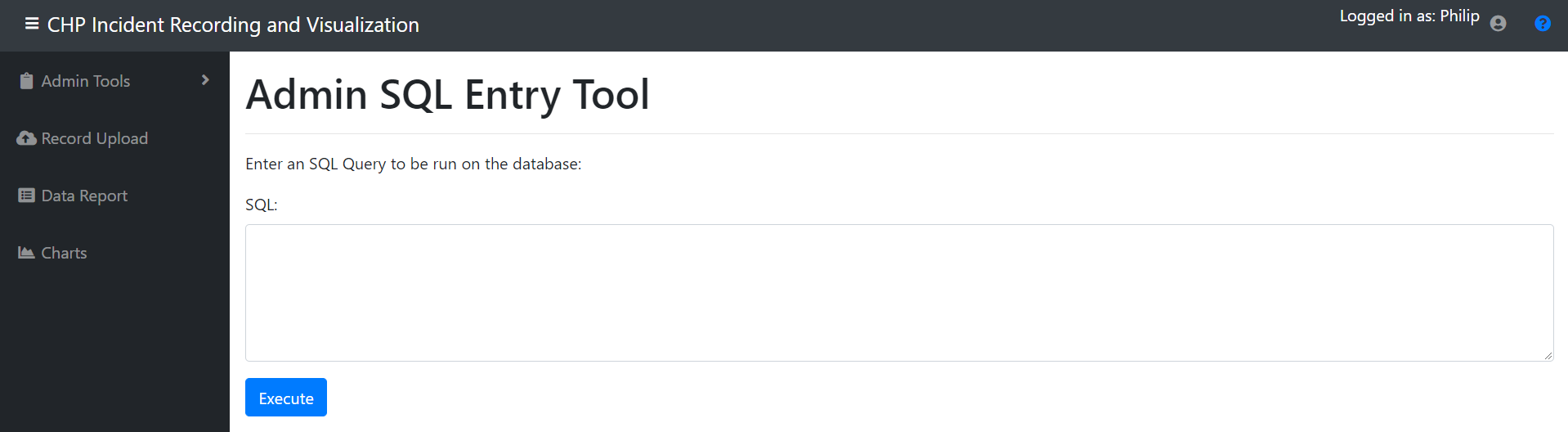
**User Management Page**

The admin has the ability to create, remove, change permissions, and change the password of the users.

On the User Management Page at the top of the page you can create a user.

Scroll down a little and you will see a list of all of the users. There you can change their password, change their permissions, and remove the user.

**SQL Entry Page**

The admin also has access to a SQL Entry page. That will allow the admin to execute queries into the database if needed.

## **Security:**

Since this application is being deployed on the internal network at CHP and will not be accessible from the Internet, we felt that rigorous security measures were unnecessary. However, we did include some basic features that are designed to protect against unauthorized access to the internal pages of the application. Password requirements enforcement was also implemented as per the specs of the project.

**Unauthorized Access:**

When a user successfully logs in, their UserType is saved as a Flask session variable. Whenever a new page is loaded, either from a click redirect or direct URL entry, the Flask route method for that page runs a short python script that first checks if the user is logged in (i.e. if the UserType session variable is set), and then checks the UserType session variable to verify that it matches one of the UserTypes that has access to the page. If the user does not have access to the page, they are redirected to their landing page, as if they just logged in.

Each UserType has a specific set of pages they have access to, and they should not be able to access any pages outside of that set. The pages that each UserType has access to is as follows (Index not included since all users need access to login):

Root: Full access to all pages

Admin: UserMgt, SQLEntry, RecordUpload, DataReport, Charts

Full User: Homepage, RecordUpload, DataReport, Charts

Super Intern: RecordUpload, DataReport, Charts

Intern: RecordUpload, DataReport, Charts

View Only: DataReport, Charts

### **Password Requirements:**

The password requirements we received from CHP are as follows:

* Password must be between 8 and 25 characters in length.
* Password must include letters, at least one number, and at least one symbol.
* Password cannot contain spaces.

These requirements are NOT implemented in the database, but only in the code for the application. This means that a user password can be set to anything through a direct SQL query to the database. The requirements are enforced when creating a new user through the application, or when an admin or root user attempt to change a user password. Error messages are displayed if any of the requirements fail, detailing exactly what requirements the password did not meet.

Note: The requirement stating that the password must include at least one symbol did not specify if there are any illegal symbols. In our implementation, the following symbols are accepted as legal symbols for a password: ! @ # $ % ^ & \* ( ) ` ~ - \_ + = { } [ ] : ; ' | " < , > . ? /

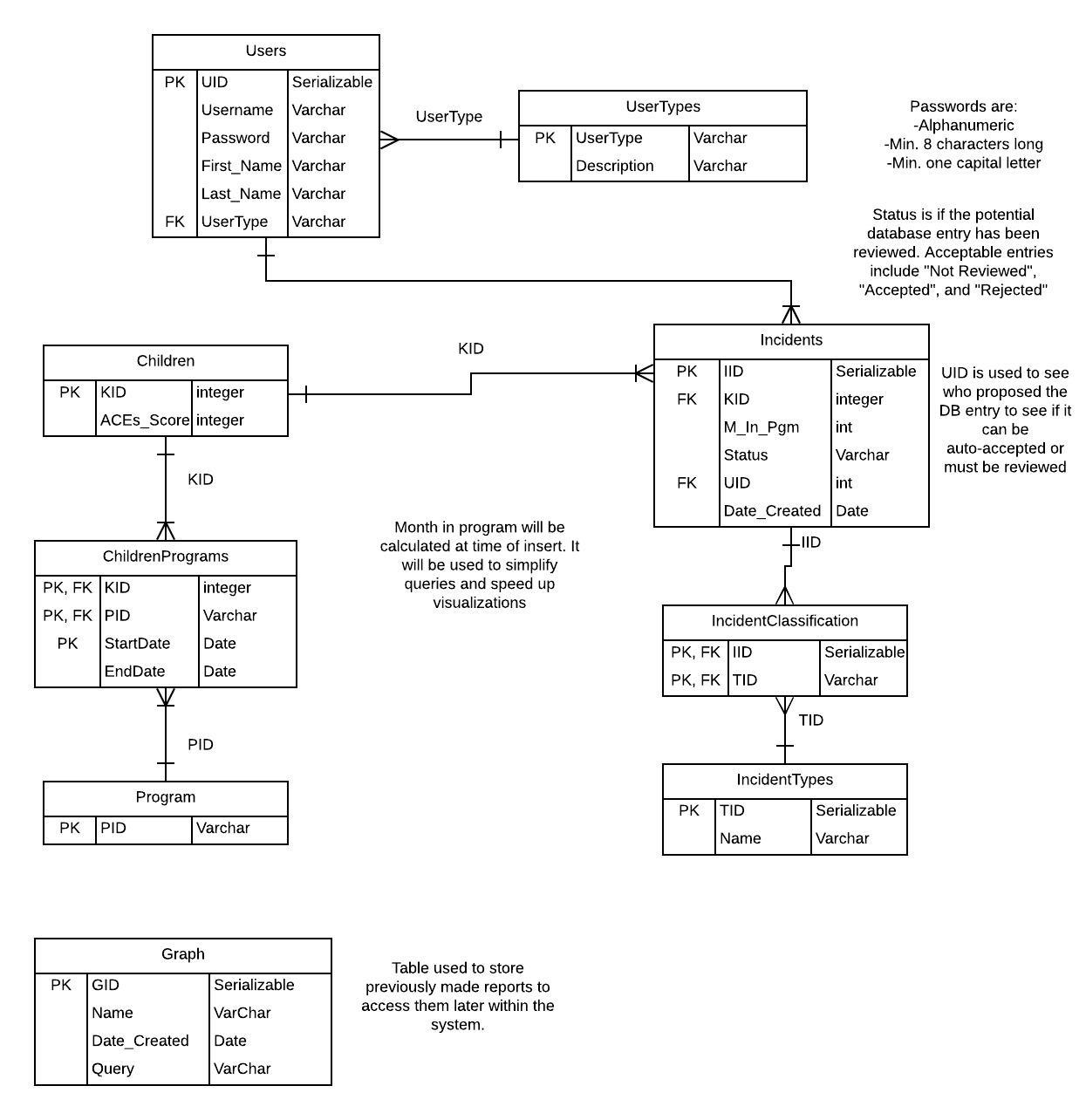
**Warning:**

We have made a page for the admin that will allow them to execute sql queries directly into the database. We have allowed our database to be expandable so if needed in the future you have the option to edit the database. Be careful when executing queries into the database because if you accidentally delete something or change the wrong table you could break the whole entire database rendering the application useless. If that happens then the database will need to be built again from scratch. There is documentation in the deployment plan on how to build the database if needed in the future.

**Appendix:**

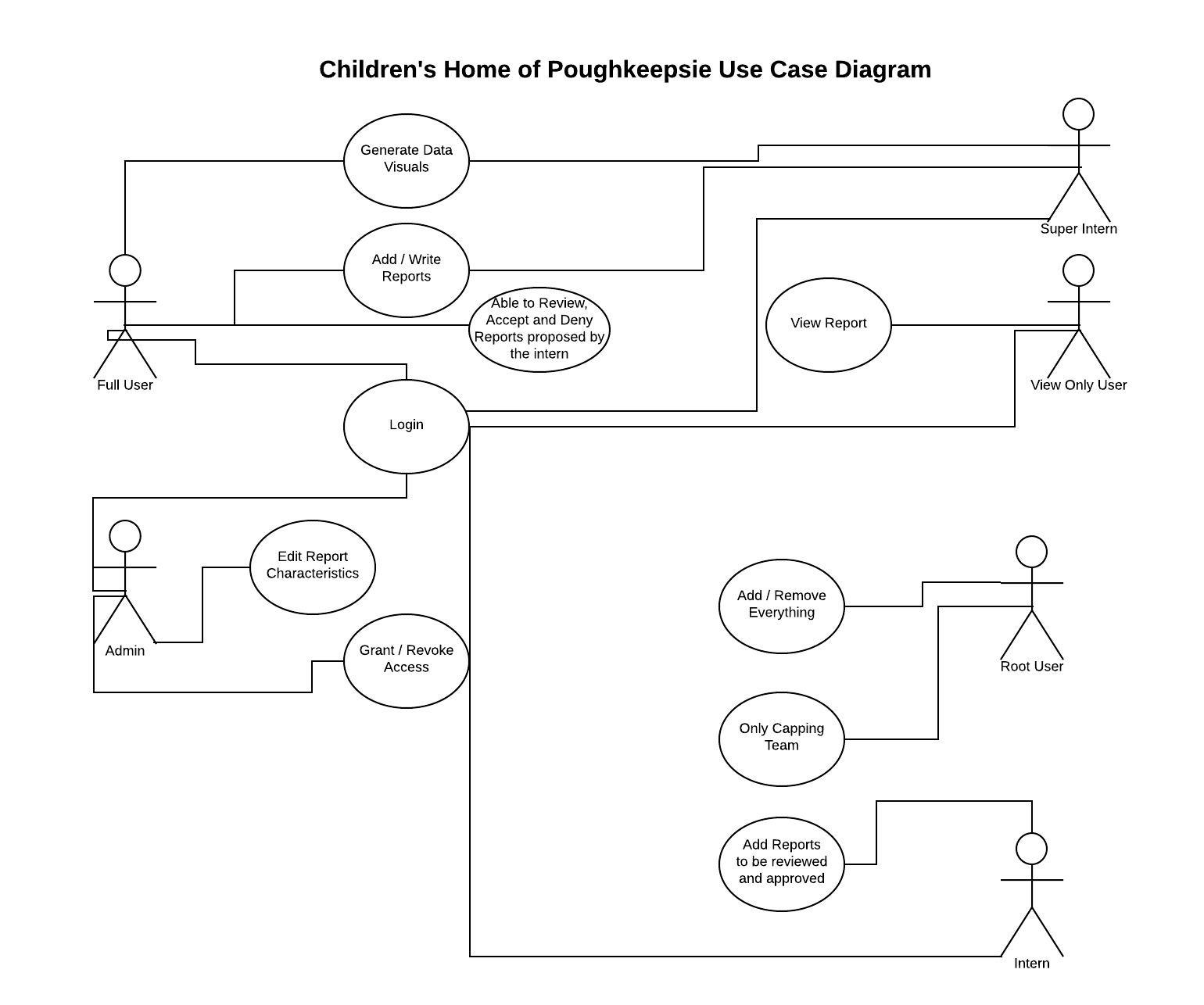
**ER-Diagram:**

**ER Diagram**



* Strong Entities
  + Users - Table containing all user account to login
  + UserTypes - Table containing the various types of user accounts
  + Children - Table containing minimal information about children of CHP
  + Incidents - Table to store all incidents
  + Program - Table storing the various CHP programs
  + Incident Types - Table storing the various types of incidents
  + Graph - Table to store previously created reports
* Weak Entities
  + ChildrenPrograms - Table to link many children to many programs
  + IncidentClassification - Table to link many incidents to many incident types

**Use Case Diagram:**



We envisioned there to be multiple users using this application; full user, admin, super intern, view only user, intern, and root user. The root user is essentially the capping team. So that user is temporary. The root user’s capabilities are to add and remove everything. A full user can generate charts, add and write reports, and login. The full user can also review, accept and deny reports that have been proposed by the intern. A super intern has all of the same capabilities as a full user. A intern can login can propose a report that gets reviewed and approved by the full user. The view only user can login and only view the reports. They cannot make or change anything. An admin is able to login, edit report characteristics, and grant and revoke access.