CS165 Database Systems
Machine Project - Criminal Records Management

Overview:
This project is intended as an avenue for students to apply both concepts learned in lectures and tools and technologies learned in lab. You will form groups of three, and write a **web application**. If any of you wish to work alone or in a pair instead, please contact us through the class e-mail at least three days before the deadline for submitting group names.

Technologies:
Since this is a machine project, you will have to implement and submit a working software prototype. To do this, you will need to use existing technology, some of which we have or will introduce in lab sessions. This is an opportunity for you to be exposed to many technologies essential in both an academic career conducting research, or a career writing or managing the production of software.

- *RDBMS*

  You are required to use **Postgres** for this project. The minimum acceptable version is **Postgres 9.1**. We have already covered basic PostgreSQL in class. With what we have covered, you know enough to construct tables and write queries to achieve reasonably high marks for this project. However, doing more research on your own can make the work of building your application easier, or allow you to write optional or custom features for even higher marks.

- *Web Framework/CMS*

  The database will only be a component of your application, even though it may be the most important one. It is strongly recommended that you use a web framework instead of building your application from scratch. You may choose any web framework, in any language, with any plugins or third-party components. You may also use a content management system like Drupal. This will make it easier to write simple features, but you may have a harder time writing custom or specific features. The recommended frameworks are **Django** and **Ruby on Rails**, because they are compatible with Postgres, well documented, have large communities and use languages that should not be too difficult or time-consuming to learn.

- *Source Control Management*

  For submission, you will be required to use either the **git** or **mercurial** distributed code versioning systems, and you must host a remote repository at **github** or **bitbucket**. It is also strongly recommended that you use the repository for development, since this is a group project, and you will need to coordinate work done by multiple members. Learning the basics of using a DCVS is an investment that will pay off in your other classes and at work in the future. You are not required to master these tools, learning basic usage will be enough for a long time.

Basic Requirements:
You will design and implement a records system for managing the following elements:

- *Reported Crimes*

  Crimes fall under a category. There should be at least five categories of crime. You may choose what those categories are. The time and date when a crime occurs should also be recorded. Assume this is always already known when the data for a specific crime is inputted. Crimes also happen in a particular location. When a crime is being investigated, it can be assigned an investigating officer. When a crime is solved, it is assigned a suspect as the perpetrator. Assume a crime is always committed by only one person, but can be investigated by many officers.

- *Suspects*

  Suspects are individuals who may have committed one or more crimes. At the very least, their name and location should be recorded. For simplicity, assume that suspects are typically active in only one area. In the real world, names are not unique, so the system should be able to accommodate individuals with the same name.

- *Agents*

  Agents are officers who investigate crimes and patrol locations. At minimum, their name and current location should be recorded. While suspects do not change locations, agents should be able to.

- *Locations*

  Locations are places that suspects can be active in, agents can investigate, and crimes can happen. You may choose the level of granularity and organization for your locations. For example, you may want your locations to be planets and moons in a solar system, countries in the world map, different regions in the Philippines, or divide a fictional city like Gotham into a 2D grid of locations. Choose something that will be easy and consistent. Have human-readable names for each of the locations. Like human names, the names of places are not unique. For example, there are 50 places in the United States called "Greenville". You should use a minimum of twenty different locations.

The above are the minimum required data that should be represented in your system. You may add or modify any attributes or tables if they will be useful for you in implementing certain features. For example, if having a Category table for crimes will help you assign a category to a crime, or help filter out a list of crimes by category, you may create one. You may also customize the data to fit a specific theme. A superhero-themed system may have a superpower attribute recorded for each criminal and each agent, for example. An alien or fantasy theme might merit recording the race of the suspects and criminals, and any special properties of locations.

Evaluation will be based on what **features** have been completely and properly implemented. Specific features and their point allocation are given in the next section.

Features:
The total number of points across all features is **200/100**. The maximum number of points you can acquire for your project is **150/100**. Since the weight of this component of your total grade 20%, you can get a weighted maximum of **30%**.

Basic CRUD:
The four fundamentals things that persistent-state applications are expected to do are **create**, **read**, **update**, and **delete** data.

- *Create - 20pts*
  Your application should provide a user interface for inputting new data. This includes crimes, suspects, agents, and any other custom data you have created. The only exception are locations, which may be pre-loaded and fixed in quantity. Software systems are typically created for the benefit of people who may not be able to write code. Your application should not require the user to directly write SQL or use an external tool like phpPgAdmin to add new data into the system.
  - *Create Crimes - 5pts*
  - *Create Suspects - 5pts*
  - *Create Agents - 5pts*
  - *Create Other - 5pts*
- *Update - 20pts*
  It should be possible to change or update entries for crimes, suspects, agents and locations without having to delete and recreate the entry. Most importantly, it should be possible to assign, remove and change the culprit and investigators for a crime. For other tables you may have created, this is optional.
  - *Update Crimes - 5pts*
  - *Update Suspects - 5pts*
  - *Update Agents - 5pts*
  - *Update Locations - 5pts*
- *Delete - 20pts*
  It should be possible to delete crimes, suspects, and agents. The application should ask the user for confirmation before doing so. If a suspect or agent is removed, crimes they have committed or are investigating should not be removed from the database. For other tables you may have created, this is optional.
  - *Delete Crimes - 5pts*
  - *Delete Suspects - 5pts*
  - *Delete Agents - 5pts*
  - *Confirmation & No Cascade - 5pts*

- *Read - 40pts*
  Users should be able to retrieve paginated lists of crimes, suspects, and agents. You may choose how many items are shown in one page. Crimes should be ordered by date and time. It should be possible to filter crimes by category, location, and whether they have been solved or not. It should be possible to search for a specific crime. Suspects and agents should be ordered alphabetically. It should be possible to filter them by location, and search by name. For each suspect, there should be a way to view their criminal history. This is a list of one or more crimes that they have committed in the past. For each unsolved crime, there should be a way to view a list of investigating officers. Both should be paginated.

  - *Paginated List of Crimes - 5pts*
  - *Searching and Filtering the List of Crimes - 5pts*
  - *Paginated List of Suspects - 5pts*
  - *Searching and Filtering the List of Suspects - 5pts*
  - *Paginated List of Agents - 5pts*
  - *Searching and Filtering the List of Agents - 5pts*
  - *Paginated Criminal History - 5pts*
  - *Paginated List of Investigating Officers - 5pts*

Intermediate Features:
For many applications, having CRUD features is sufficient. Very often, we want to know more interesting things that can be found out by running queries on the data.

- *Visualize - 45pts*
  People have an easier time processing and making conclusions from visual representations of data. This is why charts are such a common way to present statistical data.

  - *Average Crime Per Hour Histogram - 20%*
    Display a histogram that shows the average amount of crime across all locations that happens at each hour of the day, from 0:00-1:00, to 23:00-24:00.
  - *Histogram by Location - 10%*
    Improve the above feature above by parameterizing by location. If a location is supplied, the histogram should only use data from the selected location.
  - *Pie Chart of Crimes Categories 10%*
    Display a pie-chart that shows the breakdown of crimes by category.
  - *Pie Chart by Location 5%*
    Improve the above feature by parameterizing by location. If a location is supplied, the chart should only use data from the selected location.

- *Accounts, Deployment and Integration - 35pts*

  - *User Authentication and Authorization - 15*
    Unregistered users should not be allowed access to the application, but given the option to apply for access, which must be approved by an existing user. Users can log in to access your application and superuser account can manage users. This feature also requires a way to view existing users, and create and delete accounts, and an initial default account.

  - *Deploy to a Test Server - 5*
    Run your application on the Internet (hint: use Heroku).
  - *Provide a Web API - 10*
    Third-party applications have access to your data.
  - *Send Alerts through Email - 5*
    If you're able to deploy your application, certain e-mail addresses can receive notifications when an interesting event happens (e.g. crime reaches a certain threshold; crime occurs in some location)

Miscellaneous:
There are a few other miscellaneous features that should be considered.

- *Discretionary - 20pts*
  This is not a design class. Regardless, creative or attractive designs, interesting project themes, innovative features not listed above, or even humor are valuable and should be rewarded. Each instructor will be given free points to allocate to anything interesting for this purpose.
  - *Mike - 10pts*
  - *Nestor - 10pts*
- *Prerequisites*

  There are also some features that are required but ungraded. You will need to prepare a README that will provide basic instructions on how to build, install and use your application. You will also need to prepare sufficient sample data to demonstrate your features, and a script that will populate your database with your sample data. Without these, we will not be able to check the functionality of your work. Both github and bitbucket have features that will allow you to create instructions, and the web framework you choose should give you a way to pre-load data into your application.

Submission:
You must send a list of your group members and your group name to the class e-mail by **January 25, 2014**. By this time, you are also expected to have created a project page on your project hosting site. You will need to have provided read-privilege access to your repository for the **cs165-13-14-2** account in github, or **cs165_13_14_2** account in bitbucket.

Development should cease on **March 30, 2014**. Each of your remote repositories should contain a branch named **submit**. Your instructors will pull from this remote branch.