Teen Court Database

Software Development Document | Current Version 1.1.7

Prepared By:

Andrew Thompson

Robert Reilly

Revision History

Date	*Author*	*Version*	*Comments*
9/26/12	*Andrew Thompson*	*1.0.0*	*Initial version*
10/4/12	*Andrew Thompson*	*1.0.1*	*Sprint Report 1*
10/30/12	*Andrew Thompson*	*1.0.2*	*Updated User Access and Defendant*
11/1/12	*Andrew Thompson*	*1.0.3*	*Sprint Report 2*
11/20/12	*Andrew Thompson*	*1.0.4*	*Added Court Program and Volunteer*
12/2/12	*Andrew Thompson*	*1.0.5*	*Added SQL schema information*
12/5/12	*Andrew Thompson*	*1.0.6*	*Added database table for each section*
12/6/12	*Robert Reilly*	*1.0.7*	*Added CONOPS*
12/6/12	*Andrew Thompson*	*1.0.8*	*Sprint Report 3*
2/8/12	*Andrew Thompson*	*1.0.9*	*Sprint Report 4*
3/17/12	*Andrew Thompson*	*1.1.0*	*Sprint Report 5*
4/18/13	*Robert Reilly*	*1.1.1*	*Added class documentation*
4/18/12	*Andrew Thompson*	*1.1.2*	*Sprint Report 6*
4/21/13	*Andrew Thompson*	*1.1.3*	*Renamed document, new schema, cleanup*
4/23/13	*Andrew Thompson*	*1.1.4*	*Component Design Documentation*
4/23/13	*Robert Reilly*	*1.1.5*	*Finished CONOPS*
4/24/13	*Robert Reilly*	*1.1.6*	*Merge testing document*
4/24/13	*Andrew Thompson*	*1.1.7*	*Finish document combining*

Table of Contents

1.0 Overview 8
1.1 Scope 8
1.2 Purpose 8
1.2.1 Domain Name and Web Space 8
1.2.2 Server Environment and Database 8
1.2.3 Web Application 8
1.2.4 Security 8
1.3 Account Credentials 9
1.3.1 Web Server 9
1.3.2 CPanel Hosting Access 9
1.3.3 Application Administrator Login 9
1.3.4 Google Analytics Login 9
1.3.5 MySQL Web User 9
1.4 Systems Goals 10
1.5 Concept of Operations (CONOPS) 10
1.5.1 Unregistered Users 10
1.5.2 Program User 10
1.5.3 Program Manager 12
1.5.4 Program Administrator 15
1.5.5 Program-Application Administrator 17
1.5.6 Application Administrator 20
1.6 System Overview 21
1.7 Technologies Overview 22
2.0 Project Overview 23
2.1 Team Members and Roles 23
2.2 Project Management Approach 23
2.3 Phase Overview 23
2.3.1 User Stories and Requirements Gathering 23
2.3.2 Database Schema and Class Design 23
2.3.3 Prototype 23
2.3.4 Application Development 24
2.3.5 Testing 24

2.3.6 Delivery 24
2.4 Terminology and Acronyms 24
3.0 Requirements 25
3.1.1 User Access 25
3.1.2 Teen/Youth Program & Data Access 25
3.1.3 Defendant 25
3.1.4 Volunteer 25
3.1.5 Court 25
3.1.6 Workshop 25
3.1.7 Reports 26
3.1.8 Statistics 26
4.0 Branding 26
4.1 Logo 26
4.2 Color Palette 26
5.0 Design and Implementation 27
5.1.1 JavaScript/JQuery File Location 27
5.1.2 Cascading Style Sheet (CSS) Location 28
5.2 Program 28
5.2.1 Component Overview 28
5.2.2 Application Pages 29
5.2.3 Database Tables 29
5.2.4 Classes 29
5.2.5 Architecture Overview 29
5.2.6 Design Overview 30
5.3 User Account and Access 30
5.3.1 Component Overview 30
5.3.2 Application Pages 31
5.3.3 Database Tables 31
5.3.4 Classes 31
5.3.5 Architecture Overview 31
5.3.6 Design Overview 33
5.4 Defendant 33
5.4.1 Component Overview 33

5.4.2 Application Pages 35
5.4.3 Database Tables 36
5.4.4 Classes 36
5.4.5 Architecture Overview 36
5.4.6 Design Overview 37
5.5 Volunteer 37
5.5.1 Component Overview 37
5.5.2 Application Pages 37
5.5.3 Database Tables 37
5.5.4 Classes 38
5.5.5 Architecture Overview 38
5.5.6 Design Overview 38
5.6 Courts 38
5.6.1 Component Overview 38
5.6.2 Application Pages 39
5.6.3 Database Tables 39
5.6.4 Classes 40
5.6.5 Architecture Overview 40
5.6.6 Design Overview 40
5.7 Workshops 40
5.7.1 Component Overview 40
5.7.2 Application Pages 40
5.7.3 Database Tables 41
5.7.4 Classes 41
5.7.5 Architecture Overview 41
5.7.6 Design Overview 41
5.8 Statistics 41
5.9 Surveys 41
5.10 Program Administration Options 41
5.11 Application Administration Options 41
6.0 Testing 42
6.1 Requirement Testing 42
6.1.1 Log in, recovery, and register 42

6.1.2 Site Admin Functions 43
6.1.3 Program Admin/Manager Functions, Defendant Tab 45
6.1.4 Program Admin/Manager Functions, Volunteer Tab 52
6.1.5 Program Admin/Program Manager, Courts Tab 53
6.1.6 Program Admin/Program Manager, Workshops Tab 55
6.1.7 Program Admin/Program Manager, Users Tab 57
6.1.8 Program User functions 58
6.1.9 User Profile 59
6.1.10 Program Verification 59
6.2 Traceability Matrix 60
7.0 Development Environment 64
7.1 Development IDE and Tools 64
7.2 Source Control 64
7.3 Dependencies 65
7.4 Build Environment 65
7.5 Development Machine Setup 65
8.0 Release | Setup | Deployment 65
8.1 Setup Information 65
8.2 System Versioning Information 65
9.0 End User Documentation 65
Appendix I: List of Figures 66
Appendix II: Supporting Information and Details 67
II.1 Database Schema 67
Appendix III: Sprint Reports 78
III.1 Sprint 1 Progress Report 78
III.2 Sprint 2 Progress Report 79
III.3 Sprint 3 Progress Report 80
III.4 Sprint 4 Progress Report 81
III.5 Sprint 5 Progress Report 83
III.6 Sprint 6 Progress Report 84
Appendix IV: Class Documentation 85
IV.1 Citation 85
IV.2 Core 86

IV.3 Court Location 87
IV.4 Court 87
IV.5 Data 88
IV.6 Defendant 90
IV.7 Guardian 92
IV.8 Location 93
IV.9 Class Program 93
IV.10 School 95
IV.11 Sentence 95
IV.12 User 96
IV.13 Volunteer 97
IV.14 Class Workshop Location 98
IV.15 Workshop 99

1.0 Overview

The Teen Court Database is a web application used to track the progress of juvenile defendants within the local teen, or youth court program. Court administrators will have the ability to generate reports, view statistics and track volunteer information. Since this is intended to be web application, access will be obtained with any internet capable computer or mobile device.

This document contains the necessary components for the design and implementation of the Teen Court Database application. [AT]

1.1 Scope

The scope of this project is to develop a web based system that allows court administrators the ability to track information for juvenile defendants within the teen court programs. [AT]

1.2 Purpose

The purpose of this document is to describe what the various components of the system are and how they will be developed. The major components of the application are described in various subsections:

- *Technologies Used:* Lists the specific languages, plugins, or tools used to develop the component.
- *Component Overview:* Describes the data and database tables within the component.
- *Phase Overview:* Describes what work is done during specific phases of development.
- *Architecture Overview:* Describes how the component will function.
- *Design Overview:* Describes the look and feel of the component.

Any possible development issues or constraints will also be described. [AT]

1.2.1 Domain Name and Web Space

This application will be developed under the domain name teencourtdb.com. Web and database server hosting will be provided by the Baltimore based website design firm, eCoastStudios. Michael Roth is the main contact for this project.

1.2.2 Server Environment and Database

The development server and production server will be on Linux servers with the latest version of PHP and MySQL database. A hosting control solution, cPanel, will allow the development team access to web space management.

1.2.3 Web Application

The main application will provide all the requirements gathered from stakeholders and listed in the Software Requirements Document.

1.2.4 Security

In addition to individual user access based upon email addresses, a security certificate/key has been provided for Secure Socket Layer (SSL) data encryption between the client and server. Also, once the application is live, the defendant table should be encrypted in the database.

1.3 Account Credentials

These are the various accounts used to access the web hosting and database servers. [AT]

1.3.1 Web Server

Address:	teencourtdb.com
Username:	teencour
Password:	1Vz6S(k{o1rx=5>:

1.3.2 CPanel Hosting Access

This login is for the web hosting control panel. A vast array of information can be manipulated here.

Address:	teencourtdb.com/cpanel
Username:	teencour
Password:	1Vz6S(k{o1rx=5>:

1.3.3 Application Administrator Login

Use this account to have just application administrator access – can only add programs and users.

Address:	teencourtdb.com/index.php
Username:	admin@teencourtdb.com
Password:	[T33ncourt]

1.3.4 Google Analytics Login

Google Analytics tracks visitor information.

Address:	http://www.google.com/analytics/
Username:	admin@teencourtdb.com
Password:	1Vz6S(k{o1rx=5>:

1.3.5 MySQL Web User

This is used for the PHP database connections. The user has limited rights and can be viewed in Cpanel.

Host:	localhost
Database Name:	teencour_web
Database User:	1Vz6S(k{o1rx=5>:
Database Password:	t33nc0urtw3b12

1.4 Systems Goals

This project's goals are to give teen/youth court program administrators and their employees an accessible and easy to use solution to track participants in youth court programs. By tracking defendant data, statistics can be generated to prove these types of alternate judicial processes work. This may lead to more funding and more courts taking this approach. [AT]

1.5 Concept of Operations (CONOPS)

This section will cover how the program is to be used from the view point of individuals who will be using it. [RR]

1.5.1 Unregistered Users

This will cover all components that individuals who have yet to log in to the system will interact with.

1.5.1.1 Login

The user will type in their user name and password. If the user name and password match what is in the database, the user will be moved to the home screen. If one or both are wrong, the program will display an error message and ask the user to log in again.

1.5.1.2 Register Account

The user will type in the court program code, their email address, choose a secure password and confirm it, then enter the captcha. If the program code and captcha is valid, the user will be informed that the court administrator for that court has to approve them. If the program code or captcha is invalid, the user will be informed of the error and be asked to enter it again.

1.5.1.3 Forgot Password

The user will type in their email address. After verifying that the email is valid, a random string will be used to set a new password and an email will be sent to inform the user of their new password.

1.5.1.4 Help

The user can view a guide on how to use the system or how to contact the administrator.

1.5.2 Program User

This will cover components that a standard, program user can interact with.

1.5.2.1 Home

After successfully logging in to the court system, the user will be presented with the home screen. The home screen will show all upcoming courts dates. Clicking the view option on a court will allow the user to view that court's information.

1.5.2.2 List Defendants

Clicking on the Defendant tab or List Defendants will take the user to the defendants list. All defendants within the program who have yet to be expunged will be displayed. Defendant information can be viewed by clicking on the View option.

1.5.2.3 View Defendant

Upon opening the page, the user will have all information of the defendant displayed. Program Users are unable to edit any of the information about a defendant, but can print off forms about the defendant.

1.5.2.4 New Defendant

Program users are unable to add new defendants into the program.

1.5.2.5 Search Defendants

Clicking on the Search Defendant button will allow the user to view a list of all defendants within their program. Use of the DataTables search bar allows the user to narrow down who they are looking for.

1.5.2.6 List Volunteers

Clicking on the Volunteer tab or List Volunteers will take the user to the volunteers list. All active volunteers within the program will be displayed. Volunteer information can be viewed by clicking on the Edit option.

1.5.2.7 View Volunteer

Upon opening the page, the user will have all information of the volunteer displayed. Program users are unable to edit any of the information about a volunteer.

1.5.2.8 New Volunteer

Program users are unable to add new volunteers into the program.

1.5.2.9 Search Volunteer

Clicking on the Search Volunteer button will allow the user to view a list of all volunteers within their program. Use of the DataTables search bar allows the user to narrow down who they are looking for.

1.5.2.10 List Courts

Clicking on the Court tab or List Courts will take the user to the courts list. All courts that have yet to be closed within the program will be displayed. Court information can be viewed by clicking on the View option.

1.5.2.11 View Court

Upon opening the page, the user will have all information of the court displayed. Program users are unable to edit any of the information about a court.

1.5.2.12 View Court Hours

Upon opening the page, the user will have all hours that members of the court and the jury pool has worked for this court. Program users are unable to edit any court hours.

1.5.2.13 New Court

Program users are unable to create new courts.

1.5.2.14 Search Courts

Clicking on the Search Court button will allow the user to view a list of all volunteers within their program. Use of the DataTables search bar allows the user to narrow down who they are looking for.

1.5.2.15 List Workshops

Clicking on the Workshop tab or List Workshops will take the user to the workshops list. All active workshops will be displayed. Workshop information can be viewed by clicking on the View option.

1.5.2.16 View Workshop

Upon opening the page, the user will have all information of the workshop displayed. Program users are unable to edit any of the information about a workshop.

1.5.2.17 New Workshop

Program users are unable to create new workshops.

1.5.2.18 Search Workshops

Clicking on the Search Workshop button will allow the user to view a list of all volunteers within their program. Use of the DataTables search bar allows the user to narrow down who they are looking for.

1.5.2.19 Profile

Clicking on the user's name will take them to their profile page. Users are able to change their email, password, personal information, and phone numbers.

1.5.3 Program Manager

This will cover components that a program manager user can interact with.

1.5.3.1 Home

After successfully logging in to the court system, the manager will be presented with the home screen. The home screen will show all upcoming courts dates. Clicking the View option on a court will allow the manager to view that court's information in the Court tab.

1.5.3.2 List Defendants

Clicking on the Defendant tab or List Defendants will take the manager to the Defendants list. All defendants within the program who have yet to be expunged will be displayed. Defendant information can be edited by clicking on the Edit option.

1.5.3.3 View Defendant

Upon opening the page, the manager will have all information of the defendant displayed. Managers will be able to modify the defendant's primary information, personal information, parent information, citation information, intake information, court information, sentence information, workshop information, have the option to expunge the defendant according to the expunge level set by the program administrator, print forms relating to the defendant, and record case notes. Recording any changes made to a defendant will require hitting the update button at top or on the tab.

1.5.3.4 New Defendant

Clicking on the New Defendant button at the top of the page will allow the manager to create a new defendant. The manager can enter the defendant's last name, first name, date of birth, home phone, court case number, and agency case number. After the information has been entered, the manager can click the Add Defendant button at the top of the page. The manager will then be moved to the Edit Defendant page to enter other information relating to the defendant.

1.5.3.5 Search Defendants

Clicking on Search Defendant will allow the manager to view a list of all defendants within their program. Use of the DataTables search bar allows the manager to narrow down what they are looking for.

1.5.3.6 List Volunteers

Clicking on the Volunteer tab or List Volunteers will take the manager to the volunteers list. All active volunteers within the program will be displayed. Volunteer information can be viewed by clicking on the Edit option.

1.5.3.7 View Volunteer

Upon opening the page, the manager will have all information of the volunteer displayed. Managers are able to update volunteer information, volunteer positions, and volunteer hours. Courts the volunteer has been assigned to can be accessed by clicking on the View option, and hours for that court can be assigned by clicking on the Hours option.

1.5.3.8 New Volunteer

Clicking on New Volunteer will allow the manager to enter if the volunteer is active, first name, last name, email, and phone number. Adding the volunteer will take the manager to the view volunteer page to assign positions.

1.5.3.9 Search Volunteers

Clicking on Search Volunteer will allow the manager to view a list of all volunteers within their program. Use of the DataTables search bar allows the manager to narrow down what they are looking for.

1.5.3.10 List Courts

Clicking on the Court tab or List Courts will take the manager to the courts list. All courts that have yet to be closed within the program will be displayed. Court information can be viewed by clicking on the View option.

1.5.3.11 View Court

Upon opening the page, the manager will have all information of the court displayed. Managers are able to change what defendant is assigned to the court, date and time of the court, what type of court it is and if the contract was signed, close the court, choose or enter a court location, and select court members, jury pool, and if guardians will be attending.

1.5.3.12 View Court Hours

Upon opening the page, the manager will have all court members and jury displayed. Managers can assign hours worked individually or to all members at once.

1.5.3.13 New Court

Clicking on New Court will allow the manager to create a new court and enter the defendant, date and time of the court, what type of court it is and if the contract was signed, and choose or enter a court location. Court members, jury pool, and if guardians will be attending will be assigned from the view court page.

1.5.3.14 Search Courts

Clicking on the Search Court button will allow the manager to view a list of all volunteers within their program. Use of the DataTables search bar allows the manager to narrow down what they are looking for.

1.5.3.15 List Workshops

Clicking on the Workshop tab or List Workshops will take the manager to the workshops list. All active workshops will be displayed. Workshop information can be viewed by clicking on the View option.

1.5.3.16 View Workshop

Upon opening the page, the manager will have all information of the workshop displayed. Managers can change workshop information, workshop location, and workshop participants. Participants can be added at the bottom the page, and individual participants can be removed or marked as having completed the workshop from the participant list.

1.5.3.17 New Workshop

Clicking on New Workshop will allow the manager to create a new workshop and enter workshop name, date, time, instructor, officer, workshop description, and workshop location. Workshop participants can be added on the View Workshop page.

1.5.3.18 Search Workshops

Clicking on the Search Workshop button will allow the manager to view a list of all volunteers within their program. Use of the DataTables search bar allows the manager to narrow down who they are looking for.

Profile

Clicking on the manager's name will take them to their profile page. Managers are able to change their email, password, personal information, and phone numbers.

1.5.4 Program Administrator

This will cover components that a program administrator user can interact with.

1.5.4.1 Home

After successfully logging in to the court system, the administrator will be presented with the home screen. The home screen will show all upcoming courts dates. Clicking the View option on a court will allow the administrator to view that court's information in the Court tab.

1.5.4.2 List Defendants

Clicking on the Defendant tab or List Defendants will take the administrator to the Defendants list. All defendants within the program who have yet to be expunged will be displayed. Defendant information can be edited by clicking on the Edit option.

1.5.4.3 View Defendant

Upon opening the page, the administrator will have all information of the defendant displayed. Administrators will be able to modify the defendant's primary information, personal information, parent information, citation information, intake information, court information, sentence information, workshop information, have the option to expunge the defendant according to the expunge level set by the program administrator, print forms relating to the defendant, and record case notes. Recording any changes made to a defendant will require hitting the update button at top or on the tab.

1.5.4.4 New Defendant

Clicking on the New Defendant button at the top of the page will allow the administrator to create a new defendant. The administrator can enter the defendant's last name, first name, date of birth, home phone, court case number, and agency case number. After the information has been entered, the administrator can click the Add Defendant button at the top of the page. The administrator will then be moved to the Edit Defendant page to enter other information relating to the defendant.

1.5.4.5 Search Defendants

Clicking on Search Defendant will allow the administrator to view a list of all defendants within their program. Use of the DataTables search bar allows the administrator to narrow down what they are looking for.

1.5.4.6 List Volunteers

Clicking on the Volunteer tab or List Volunteers will take the administrator to the volunteers list. All active volunteers within the program will be displayed. Volunteer information can be viewed by clicking on the Edit option.

1.5.4.7 View Volunteer

Upon opening the page, the administrator will have all information of the volunteer displayed. Administrators are able to update volunteer information, volunteer positions, and volunteer hours. Courts the volunteer has been assigned to can be accessed by clicking on the View option, and hours for that court can be assigned by clicking on the Hours option.

1.5.4.8 New Volunteer

Clicking on New Volunteer will allow the administrator to enter if the volunteer is active, first name, last name, email, and phone number. Adding the volunteer will take the administrator to the view volunteer page to assign positions.

1.5.4.9 Search Volunteers

Clicking on Search Volunteer will allow the administrator to view a list of all volunteers within their program. Use of the DataTables search bar allows the administrator to narrow down what they are looking for.

1.5.4.10 List Courts

Clicking on the Court tab or List Courts will take the administrator to the courts list. All courts that have yet to be closed within the program will be displayed. Court information can be viewed by clicking on the View option.

1.5.4.11 View Court

Upon opening the page, the administrator will have all information of the court displayed. Administrators are able to change what defendant is assigned to the court, date and time of the court, what type of court it is and if the contract was signed, close the court, choose or enter a court location, and select court members, jury pool, and if guardians will be attending.

1.5.4.12 View Court Hours

Upon opening the page, the administrator will have all court members and jury displayed. Administrators can assign hours worked individually or to all members at once.

1.5.4.13 New Court

Clicking on New Court will allow the administrator to create a new court and enter the defendant, date and time of the court, what type of court it is and if the contract was signed, and choose or enter a court location. Court members, jury pool, and if guardians will be attending will be assigned from the view court page.

1.5.4.14 Search Courts

Clicking on the Search Court button will allow the administrator to view a list of all volunteers within their program. Use of the DataTables search bar allows the administrator to narrow down what they are looking for.

1.5.4.15 List Workshops

Clicking on the Workshop tab or List Workshops will take the administrator to the workshops list. All active workshops will be displayed. Workshop information can be viewed by clicking on the View option.

1.5.4.16 View Workshop

Upon opening the page, the administrator will have all information of the workshop displayed. Administrators can change workshop information, workshop location, and workshop participants. Participants can be added at the bottom the page, and individual participants can be removed or marked as having completed the workshop from the participant list.

1.5.4.17 New Workshop

Clicking on New Workshop will allow the administrator to create a new workshop and enter workshop name, date, time, instructor, officer, workshop description, and workshop location. Workshop participants can be added on the View Workshop page.

1.5.4.18 Search Workshops

Clicking on the Search Workshop button will allow the administrator to view a list of all volunteers within their program. Use of the DataTables search bar allows the administrator to narrow down who they are looking for.

1.5.4.19 List Users

Clicking on the Users tab will display all users in the administrator's program. Clicking on View will allow you to edit that user.

1.5.4.20 View User

Upon opening the page, the administrator will have all of the user's information displayed. Administrators can change if the user is active, their first name, their last name, their email, and password. Administrators can also change the user's access level and time zone.

1.5.4.21 New User

Clicking on New User will allow the administrator to create a new user and enter the user's first name, last name, email, password, access level, and time zone. User program is set to the administrator's.

1.5.4.22 Profile

Clicking on the administrator's name will take them to their profile page. Administrators are able to change their email, password, personal information, and phone numbers.

1.5.5 Program-Application Administrator

This will cover components that a program- application user can interact with.

1.5.5.1 Home

After successfully logging in to the court system, the administrator will be presented with the home screen. The home screen will show all upcoming courts dates. Clicking the View option on a court will allow the administrator to view that court's information in the Court tab.

1.5.5.2 List Defendant

Clicking on the Defendant tab or List Defendants will take the administrator to the Defendants list. All defendants within the program who have yet to be expunged will be displayed. Defendant information can be edited by clicking on the Edit option.

1.5.5.3 View Defendant

Upon opening the page, the administrator will have all information of the defendant displayed. Administrators will be able to modify the defendant's primary information, personal information, parent information, citation information, intake information, court information, sentence information,

workshop information, have the option to expunge the defendant according to the expunge level set by the program administrator, print forms relating to the defendant, and record case notes. Recording any changes made to a defendant will require hitting the update button at top or on the tab.

1.5.5.4 New Defendant

Clicking on the New Defendant button at the top of the page will allow the administrator to create a new defendant. The administrator can enter the defendant's last name, first name, date of birth, home phone, court case number, and agency case number. After the information has been entered, the administrator can click the Add Defendant button at the top of the page. The administrator will then be moved to the Edit Defendant page to enter other information relating to the defendant.

1.5.5.5 Search Defendant

Clicking on Search Defendant will allow the administrator to view a list of all defendants within their program. Use of the DataTables search bar allows the administrator to narrow down what they are looking for.

1.5.5.6 List Volunteer

Clicking on the Volunteer tab or List Volunteers will take the administrator to the volunteers list. All active volunteers within the program will be displayed. Volunteer information can be viewed by clicking on the Edit option.

1.5.5.7 View Volunteer

Upon opening the page, the administrator will have all information of the volunteer displayed. Administrators are able to update volunteer information, volunteer positions, and volunteer hours. Courts the volunteer has been assigned to can be accessed by clicking on the View option, and hours for that court can be assigned by clicking on the Hours option.

1.5.5.8 New Volunteer

Clicking on New Volunteer will allow the administrator to enter if the volunteer is active, first name, last name, email, and phone number. Adding the volunteer will take the administrator to the view volunteer page to assign positions.

1.5.5.9 Search Volunteer

Clicking on Search Volunteer will allow the administrator to view a list of all volunteers within their program. Use of the DataTables search bar allows the administrator to narrow down what they are looking for.

1.5.5.10 List Court

Clicking on the Court tab or List Courts will take the administrator to the courts list. All courts that have yet to be closed within the program will be displayed. Court information can be viewed by clicking on the View option.

1.5.5.11 View Court

Upon opening the page, the administrator will have all information of the court displayed. Administrators are able to change what defendant is assigned to the court, date and time of the court,

what type of court it is and if the contract was signed, close the court, choose or enter a court location, and select court members, jury pool, and if guardians will be attending.

1.5.5.12 View Court Hours

Upon opening the page, the administrator will have all court members and jury displayed. Administrators can assign hours worked individually or to all members at once.

1.5.5.13 New Court

Clicking on New Court will allow the administrator to create a new court and enter the defendant, date and time of the court, what type of court it is and if the contract was signed, and choose or enter a court location. Court members, jury pool, and if guardians will be attending will be assigned from the view court page.

1.5.5.14 Search Court

Clicking on the Search Court button will allow the administrator to view a list of all volunteers within their program. Use of the DataTables search bar allows the administrator to narrow down what they are looking for.

1.5.5.15 List Workshop

Clicking on the Workshop tab or List Workshops will take the administrator to the workshops list. All active workshops will be displayed. Workshop information can be viewed by clicking on the View option.

1.5.5.16 View Workshop

Upon opening the page, the administrator will have all information of the workshop displayed. Administrators can change workshop information, workshop location, and workshop participants. Participants can be added at the bottom the page, and individual participants can be removed or marked as having completed the workshop from the participant list.

1.5.5.17 New Workshop

Clicking on New Workshop will allow the administrator to create a new workshop and enter workshop name, date, time, instructor, officer, workshop description, and workshop location. Workshop participants can be added on the View Workshop page.

1.5.5.18 Search Workshop

Clicking on the Search Workshop button will allow the administrator to view a list of all volunteers within their program. Use of the DataTables search bar allows the administrator to narrow down who they are looking for.

1.5.5.19 List Programs

Clicking on the Programs tab will display all programs in the system. Clicking on View will allow you to edit that program.

1.5.5.20 View Program

Upon opening the page, the administrator will have all of that program's information displayed. Administrators can change if the program is active, name, code, phone number, time zone, expunge type, physical address, and mailing address.

1.5.5.21 New Program

Clicking on New Program will allow the administrator to create a new program and enter the program's name, code, phone number, time zone, expunge type, physical address, and mailing address.

1.5.5.22 List Users

Clicking on the Users tab will display all users in the system. Clicking on View will allow you to edit that user.

1.5.5.23 View User

Upon opening the page, the administrator will have all of the user's information displayed. Administrators can change if the user is active, their first name, their last name, their email, and password. Administrators can also change the user's program they are assigned to, access level, and time zone.

1.5.5.24 New User

Clicking on New User will allow the administrator to create a new user and enter the user's first name, last name, email, password, program, access level, and time zone.

1.5.5.25 Profile

Clicking on the administrator's name will take them to their profile page. Administrators are able to change their email, password, personal information, and phone numbers.

1.5.6 Application Administrator

1.5.6.1 List Programs

Clicking on the Programs tab will display all programs in the system. Clicking on View will allow you to edit that program.

1.5.6.2 View Program

Upon opening the page, the administrator will have all of that program's information displayed. Administrators can change if the program is active, name, code, phone number, time zone, expunge type, physical address, and mailing address.

1.5.6.3 New Program

Clicking on New Program will allow the administrator to create a new program and enter the program's name, code, phone number, time zone, expunge type, physical address, and mailing address.

1.5.6.4 List Users

Clicking on the Users tab will display all users in the system. Clicking on View will allow you to edit that user.

1.5.6.5 View User

Upon opening the page, the administrator will have all of the user's information displayed. Administrators can change if the user is active, their first name, their last name, their email, and password. Administrators can also change the user's program they are assigned to, access level, and time zone.

1.5.6.6 New User

Clicking on New User will allow the administrator to create a new user and enter the user's first name, last name, email, password, program, access level, and time zone.

1.5.6.7 Profile

Clicking on the administrator's name will take them to their profile page. Administrators are able to change their email, password, personal information, and phone numbers..

1.6 System Overview

Users will access this application through qualified web browsers offered on most major internet capable devices, such as personal computers, tablets and smart phones. The application will offer cross-platform and cross-browser support. When a user's browser makes a connection to the website, the pages and data will be served to the client. The diagram in figure 1 shows the system's web architecture overview.

There are five major components to the actual web application. The top level object, Program, contains information pertaining to an individual teen/youth court program. Users belong to a specific program and are able to add defendants and volunteers. The defendants and volunteers are used to build workshop and court objects. Figure 2 shows the relationship between these components. The design and implementation of each component is described in full detail in section 4.0 Design and Implementation. [AT]

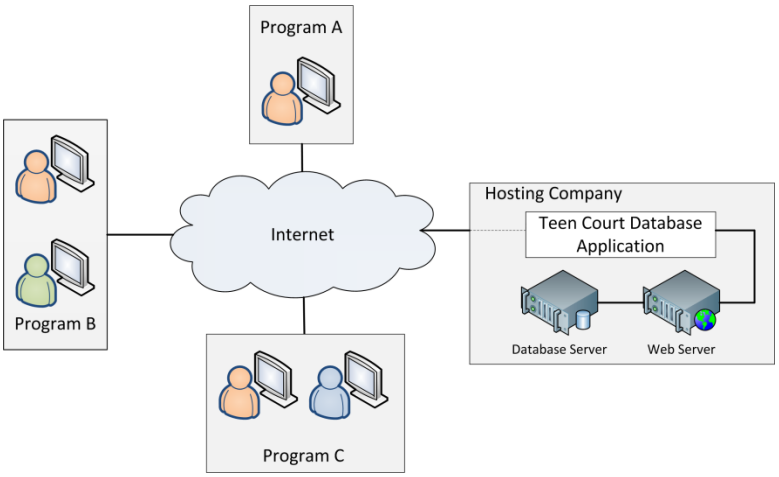

Figure 1: System Diagram

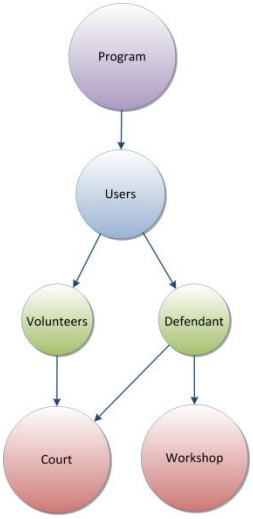

Figure 2: Major Component Diagram

1.7 Technologies Overview

This application will be developed using the latest version of PHP and take an Object-Oriented approach. PHP is popular and widely used HTML-embedded scripting language that allows developers to quickly build dynamically generated web pages. More information: can be found at http://us3.php.net/.

A data access layer will be implemented to help secure information between the web and database server. This will most likely be PHP's built in PHP Data Object (PDO). For more information, visit http://www.php.net/manual/en/book.pdo.php.

For data storage, the latest version of MySQL will be used with this application. This is a popular open-source relational database management system (RDBMS). More information can be found at http://www.mysql.com/.

JSON, JavaScript Object Notation, will be used for formatting the data of SQL results. http://www.json.org/

A JavaScript Framework, JQuery, will be used to ensure cross-browser JavaScript support. To manipulate client webpages without making additional requests to the server, AJAX, or Auto-synchronous JavaScript and XML, will be used and is built into the framework. More information can be found at http://www.jquery.com.

JQueryUI , a JavaScript Framework that generates client-side widgets will be used to provide a more functional user interface. More information about this framework is available at http://www.jqueryui.com.

Several JQuery plugins were used, these include:

DataTables, a plugin that reads JSON formatted data generated from the database and displays highly functional listing of tabular data. Sorting, searching and paging make data easy to view and manipulate. http://www.datatables.net/

JQuery Validate, a plugin to allow easy client-side validation and error reporting for web forms. http://docs.jquery.com/Plugins/Validation

JQuery Mask, a plugin to allow form field input masks. Useful to force phone number styles, dates, etc. http://plugins.jquery.com/mask/

PwStrength, a plugin to check user password strengths based on weights of characters, numbers, symbols, etc. Not a password policy enforcement, just provides a general strength. http://plugins.jquery.com/pwstrength/

JQuery ptTimeSelect, a plugin that mimics JQueryUI's DatePicker but does it for time.

http://pttimeselect.sourceforge.net/example/index.html

Google RECAPTCHA, this is used on the publically accessible registration page to prove human interactions. [AT]

2.0 Project Overview

This section will describe team member roles and how the project will be managed. [AT]

2.1 Team Members and Roles

Andrew Thompson will be the Team Leader and Lead Developer. Robert Reilly will be Lead Tester and Developer. Author notation will be identified by [AT] for Andrew and [RR] for Robert at the start of major sections (*ex: after heading 2.0 above*). Subsections within the main sections assumed to be by the same author.

2.2 Project Management Approach

The Scrum Team is scheduled to have Sprint Meetings every Tuesday and Thursday at 10:00 AM. These meetings will not last longer than 15 minutes. Additional Meetings will be scheduled as needed. The Project Lead, Andrew Thompson, is the primary contact with the Product Owner, Marlene Todd.

The project backlog will be tracked within Trello – an online collaboration and organization tool. Scrum Masters and Scrum Team Members are required access to this site. The Product Owner will not need to access this site. The Scrum Team will manage Sprint Backlog priorities based upon the Product Owners input. Sprints will last between two and three weeks. Sprint Reports will be sent to the Scrum Master and Product Owner at the end of scheduled Sprints.

The online repository, GitHub, will provide source control, source-code browser, and a project wiki. Scrum Team Members and the Scrum Master are required to have access to this site. [AT]

2.3 Phase Overview

This system will be developed in phases, they are as follow: [AT]

2.3.1 User Stories and Requirements Gathering

During this phase, the requirements for the system will be gathered from user stories. The product owner informs the team of any requirements, limitations or constraints. This information can be found in the Software Requirements Document and 3.0 Requirements.

2.3.2 Database Schema and Class Design

The database schema, or design, will be developed during the second phase. The team needs to mimic the existing Access Database currently being used by Lawrence County Teen Court, while making sure the design is as efficient as possible. The major class objects will also be designed during this phase; this includes defendants, volunteers, court listings, workshops and any other objects that can be derived from the requirements.

2.3.3 Prototype

The prototype will provide limited functionality of the final application and act as a demonstration for the client. This will allow the team to make changes without major code rework. All major areas of the application should be built for the client review.

2.3.4 Application Development

This phase will be the actual development phase. All aspects of the application will be implemented to the client's specifications and requirements. Some unit testing will take place during this phase.

2.3.5 Testing

Some aspects of testing will take place during the Application Development phase. This is to ensure proper data entry for the application. Individual domain constraints will be tested along with data accuracy and program functionality.

2.3.6 Delivery

This phase is where the application will be turned over to the client. The web files and database will already be on the production server. User manuals for each user level will also be delivered in PDF format and hard copy. The user accounts used to access the website and database will also be turned over.

2.4 Terminology and Acronyms

Listed here is common terms used throughout the document. [AT]

- AJAX: Asynchronous JavaScript and XML is a set of development tools to provide the client side to communicate with the server without refreshing the page.
- CAPTCHA: A CAPTCHA is a program that protects websites against bots by generating and grading tests that humans can pass but current computer programs cannot.
- Client: An application or system that accesses a service provided by a server.
- Court Program: A local city, county or community youth or teen court that uses this web application.
- Database: An organized set of data and associated data structures.
- DBMS: A Database Management System allows an interface to manipulate a database.
- Domain Name: An identification string that identifies represents an Internet Protocol (IP) resource. A Domain Name Server (DNS) handles the translation from Domain Name to IP address.
- JSON: JavaScript Object Notation is a lightweight data-interchange format.
- JQuery: JavaScript Framework that allows client manipulation.
- JQueryUI: JavaScript Framework that allows various client widgets to provide a more functional user interface.
- MySQL: A popular and reliable Database Management System (DBMS).
- PHP: PHP Hypertext Preprocessor is an HTML-embedded scripting language capable of Object Orientated Programming (OOP) principles.
- Server: A computer hardware system and software that serves as a dedicated host for one or more services.
- SSL: Secure Socket Layer is a cryptographic protocol that provides communication security over the Internet.
- Web Application: A software application that is developed in a browser-supported programming language and is accessed by users over an Intranet or Internet.
- Web Hosting: A hosting service on the Internet that allows website owners to access their site and typically provides web space management tools or control panels.

3.0 Requirements

The purpose of this web application is to allow youth and teen courts to track participants within their programs. In order to achieve this, several requirements are specified from the stakeholders. [AT]

3.1.1 User Access

Only individuals working for a particular court program may have access to the application. Defendants, their families, and volunteers will **not** be allowed to view **any** information contained in the application. The different levels are described in section 4.1 User Account and Access.

3.1.2 Teen/Youth Program & Data Access

The authorized users shall only have access to information pertaining to their particular court program. Court program administrators are able to adjust options pertaining to their individual programs. These options are covered in further detail in section 4.2 Court Program. Application Administrators will have the additional access to site-wide statistics and options to add new court programs. These levels of access are described in section 4.1 User Account and Access.

3.1.3 Defendant

The defendant is the primary data gathering point for this application. A defendant can be added, updated, deleted, and expunged. Each court program and their users can only access defendant information they enter. The different information fields are described in section 4.3 Defendant.

3.1.3.1 Defendant Expunging

Each court program will have one of four options to expunge defendant information: none, sealed, partial expunge and full expunge. The sealed option retains defendant data within the database but restricts the data to be displayed in results other than demographic statistics. Partial expunging removes the defendant's personal information from the database but leaves citation information for statistics gathering. Full expunge removes all traces of the defendant within the system.

3.1.4 Volunteer

Volunteers are those individuals who assist with the teen court program. Generally they are assigned as court members or jury participants. The ability view status and volunteer history will be implemented. More information can be found in 4.4 Volunteer.

3.1.5 Court

A user is able to create a court session at a certain time and location and attach defendant cases, court members and jury pool. Volunteer and defendant hours will be updateable. More information can be found in 4.5 Court.

3.1.6 Workshop

Various workshops can be created and populated from open defendants. More information can be found in 4.6 Workshops.

3.1.7 Reports

All necessary and legal documents will be generated for the defendants.

3.1.8 Statistics

Statistics reporting will include discoverable data, demographics data, and both local and regional statistics. Statistics have not been included at this time, rather they need to be produced once data has been gathered and area of statistics narrowed. They will not be implemented during this phase.

4.0 Branding

To keep a consistent look and feel of the application, several branding guidelines were implemented. This section will list the logo graphic and font and also the primary color used throughout the application.

4.1 Logo

The font used in the logo is TrajanPro. A photoshop and pdf file of the gavel logo will be provided to the client.

4.2 Color Palette

The following colors are used extensively throughout the application.

Primary Palette

#464646
R: 70 G: 70 B: 70

#858479
R: 133 G: 132 B: 121

#BBBAAB
R: 187 G: 186 B: 171

#D9D4CA
R: 217 G: 212 B: 202

Secondary Palette

#2C6079
R: 44 G: 96 B: 121

#FFFFFF
R: 255 G: 255 B: 255

#BF5A3F
R: 191 G: 90 B: 63

#72AECB
R: 114 G: 174 B: 203

5.0 Design and Implementation

The web application will be developed as several different areas or components. Components may interact with others depending upon their type or usage. Each component is described in detail below. The five major areas are the teen/youth program, user accounts and access, defendant, volunteers, courts and workshops.

Throughout the site, several areas will remember previously entered data and allow the user to select this data instead of re-entering it. These are displayed as small buttons next to the input field and looks like a window icon. Some of these areas are locations (city, state, zip), school information, statutes, sentencing requirements and common locations.

Whenever a date or time field is required, JQueryUI allows a stylized calendar or time selection window to appear. These fields also have client-side masking to ensure the date is formatted properly. The current format is MM/DD/YYYY HH:MM A. Dates get converted from user time zones to Central Standard Time (function in the core class) prior to database insertion. When the date is retrieved, it is then converted from Central Standard Time to the time zone set for the user. This ensures that proper times are displayed for users.

Phone number fields are also provided with a client-side validation mask. This ensures all phone numbers are entered in the same format. The current format is (123) 456-7890.

Some areas require a large input of data or have data that can be segmented to make the user interface less cluttered. When this is the case, the JQuery Tab elements are used. A client cookie is used to keep track of the current tab in case the user submits a form and gets redirected to the last page.

In some areas, such as defendant, the JQuery functions have been implemented with their own file. These are named jquery.js and control any JQuery elements on the component. If a jquery.js file is absent, the functions are at the top of the application pages themselves.

Almost all areas have a process.php page. This is where form POST and GET data gets sent, database interaction handled and redirection occurs. [AT]

5.1.1 JavaScript/JQuery File Location

Aside from the JavaScript and JQuery files that may reside within each components directory, several of these types of files are includes in the internal_header.php and external_header.php files. The location and description of these files are:

/includes/js/jquery.dataTables.min.js	Source file that provides the DataTable plugin for JQuery
/includes/js/jquery.validate.min.js	Source file for client-side form validation and error display
/includes/js/jquery.ui.timepicker.js	Source file for the JQueryUI-styled time selection window
/includes/js/jquery.pwstrength.min	Source file for the password strength checker
/includes/js/jquery.mask	Source file for client-side form input field masking
/includes/js/jquery.cookie	Source file for plugin that provides easier cookie control for JQuery

/includes/js/dataTables.ajaxReload	Source file for DataTables plugin to allow Asynchronous requests

5.1.2 Cascading Style Sheet (CSS) Location

The following files contain the styling elements used throughout the site and on various plugins and are contained in header_internal.php, header_external.php and the login page, index.php:

/includes/css/site.css	Main Style Sheet for custom elements used on the site
/includes/css/jquery-ui-teencourt.css	Style Sheet for JQueryUI elements
/includes/css/jquery.ui.timepicker	Style Sheet for the JQuery TimePicker plugin
/includes/css/jquery.dataTables	Style Sheet for the JQuery DataTables plugin

5.2 Program

Every youth or teen court program or association who uses this application will be assigned their own data space. This is the highest level component and almost all data is reliant upon it. Once all the necessary data has been entered, users can be added to it and they can start using the application. [AT]

5.2.1 Component Overview

This is the primary, top-level component in the application.

5.2.1.1 Program Listing

This section uses DataTables for data display. JSON Source file is located at: /data/programs.php

5.2.1.2 Program View

Only Application Administrators and the hybrid Application Administrator / Program Administrator can add or edit programs. The Program Administrators access level can only update their information.

Every program will have the following information available:

- Court Name
- Code
- Phone Number
- Time Zone
- Expunge Type
- Physical Address
- Mailing Address

Once the locations (city, state, zip) get entered, it can be selected throughout the site wherever a location is prompted. This is done to reduce data entry and possible user error.

Time zone is used as the default for users added to this program. This is used to adjust date and timestamp data from the database.

When a new program is added, a list of court positions is automatically added to the *court_position* table where the program id is a foreign key. This is done by calling the stored procedure

addCourtPositions. Program administrators can add their own custom positions in the My Program menu item. Volunteers use these for availability as court members. View the courts section for default member values.

5.2.2 Application Pages

Not all of these pages are used at the same time. These are mainly areas that use some aspect of the program component.

- /admin/programs.php
- /admin/program.php
- /admin/view_program.php
- /admin/process.php – *all form POST/GET processing*
- /includes/header_internal.php
- /includes/secure.php
- /data/programs.php – *for JSON data*
- /data/programs_common_locations.php – *for JSON data*
- /data/programs_locations.php – *for JSON data*
- /data/programs_schools.php – *for JSON data*
- /data/programs_sentences.php – *for JSON data*
- /data/programs_statutes.php – *for JSON data*

5.2.3 Database Tables

The following MySQL tables are primarily used for this component with program id used throughout the application as foreign key for many tables:

- teencour.program
- teencour.program_common_location
- teencour.program_locations
- teencour.program_officers
- teencour.program_schools
- teencour.program_sentences
- teencour.program_statutes

5.2.4 Classes

The following classes are primarily used or dependent for this component:

- /includes/class_data.php
- /includes/class_program.php
- /includes/class_location.php
- /includes/class_school.php
- /includes/class_sentence.php

5.2.5 Architecture Overview

Being the primary component, program data is used in a variety of ways in almost every section of the application. Mostly the program id is used as foreign key constraints on many tables. The initial physical address is added to the program locations table and can be selected from anywhere a location modal selection window exists on the application.

5.2.6 Design Overview

The only area where the program information is actually accessible is by the Application/Program Administrator or the My Program area where the Program Administrator can update or add various parts that rely on the program. These are the collected data such as locations, schools, etc.

5.3 User Account and Access

This section describes the User Account and User Access development process. Users are validated by email address and a password. There are five different levels of access and each provides different functionality. Refer to figure 2 on the next page. The abilities of each user are described in greater detail in the CONOPS section. [AT]

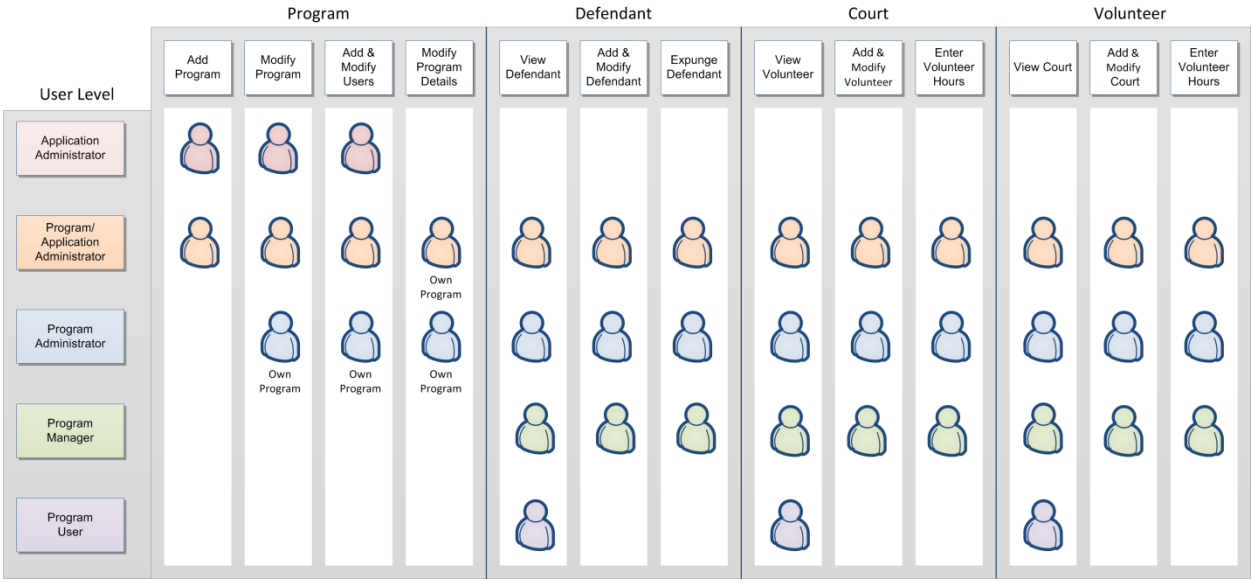

Figure 3: User Access Levels

5.3.1 Component Overview

The following areas are for the users can be accessed from the user menu option.

5.3.1.1 User Listing

There are two places to view users. One is the Application Administrator access level which can only view program and user information site-wide. The other is the Program Administrator which only has access to their specific users.

5.3.1.2 User View

Every user accessing the system will be required to have the following information in the database:

- Active or not
- ID of the Court Program they belong to
- Name (First and Last)
- Email Address
- Password (not actually stored in the database)
- Phone numbers

- Account Type

There are five account types: Application Administrator, a hybrid of both Program Administrator and Application Administrator (ex: Marlene's positions), Program Administrator, Program Manager and Program User.

5.3.2 Application Pages

This component is used widely throughout the site because of user access, but the following pages are where the primary user information is entered or accessed.

- /index.php – *the main login page*
- /register.php
- /profile.php
- /admin/users.php
- /admin/view_users.php
- /admin/process.php – *all form POST/GET processing*
- /includes/secure.php
- /data/users.php – *for JSON data*
- /data/user_history.php – *for JSON data*

5.3.3 Database Tables

The following MySQL tables are primarily used for this component:

- teencour.program
- teencour.user
- teencour.user_log
- teencour.user_phone

5.3.4 Classes

The following classes are associated with this component:

- /includes/class_program.php
- /includes/class_user.php
- /includes/class_data.php – *for JSON data*

5.3.5 Architecture Overview

The primary concern with user access is ensuring the user passwords cannot be accessed through unauthorized means. Prevention against hacking and brute force access scripts must be high priority. ~~*There will be a maximum of 3 login attempts before the user is locked out of the system. They must then wait a certain amount of time before trying again. This prevents brute force scripted hacking attempts.*~~ *(Not Yet Implemented)*

The user's password is not actually stored in the database; rather it is used in conjunction with a random value, a salt string and the user's email address. This is hashed using the SHA-256 cryptographic hash function to create a 128-character string which is stored in the user table. To gain access, the first 64-characters from the previously hashed string are used as the salt string along with the password entered

to create a new SHA-256 hashed string. The existing string is compared to the generated string to ensure user authentication. A new hash is then generated upon successful login.

New users will register their account from the registration page. The users will be given a court access code by their individual administrators and will need this to register with the application. The access code will be matched with the database to ensure it is a valid code. As the user enters a password, it will be checked against a strength algorithm, using AJAX, to make sure they are using a somewhat secure.

To avoid bots from spamming the registration page, a CAPTCHA system will be implemented to prove a human is registering for the application.

5.3.5.1 User Registration

Users are allowed to register for the application by using a code given to them by their program administrator. This code is generated when the program is created. After the user enters their credentials and passes the Google RECAPTCHA, the program administrator must approve or deny and assign a level to the user.

Figure 2 shows the architecture diagram of the registration process.

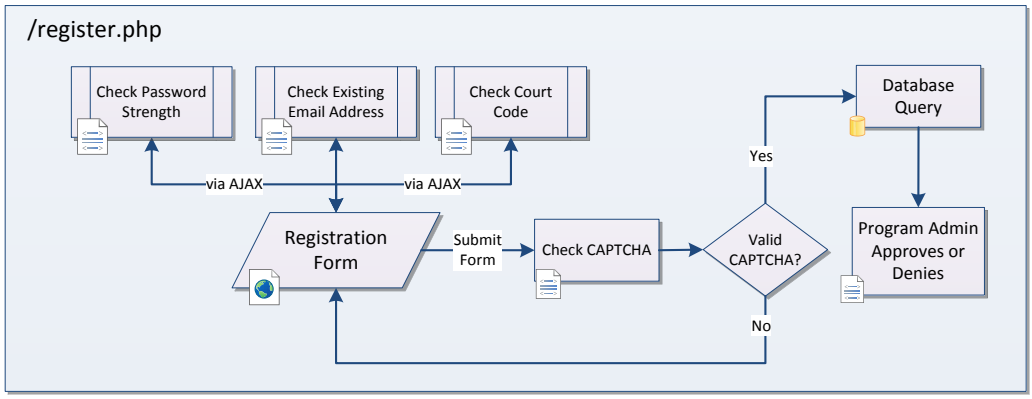

Figure 4: User Registration Architecture Diagram

5.3.5.2 User Access

A user gains access to the web application by entering their email address and password. If they are validated, their access level is set, login is recorded and session gets approved. A session will stay valid up to one hour of inactivity at which point the user will be logged out and the session destroyed.

Figure 3 is the architecture diagram of the user access implementation.

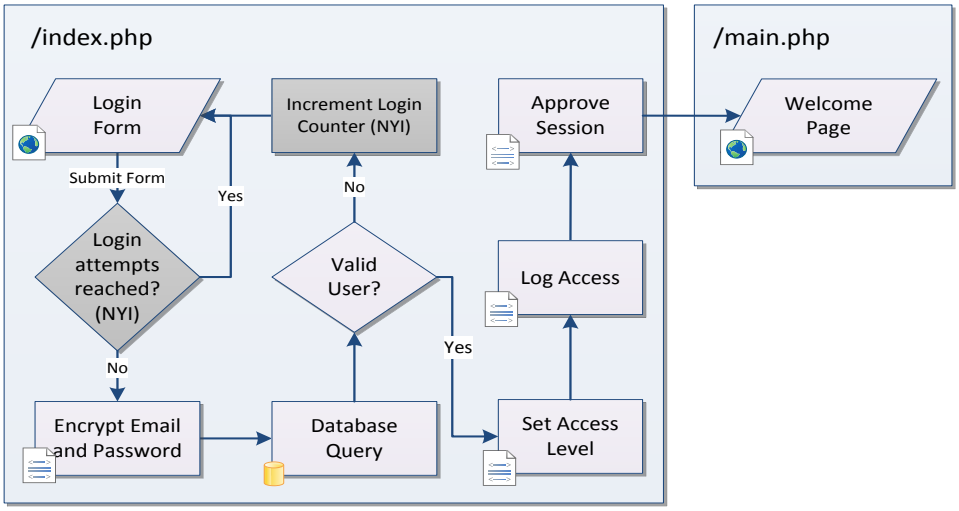

Figure 5: User Access Architecture Diagram

5.3.6 Design Overview

The user profile and registration page use the JQuery PwStrength plugin to access the users password strength. The user is available to change their password, email, time zone or phone numbers at any time by clicking on their name in the upper right.

5.4 Defendant

This section describes the Defendant development process. The Defendant is the main data object for this application. Several fields will be propagated via dropdown boxes or modal selection windows that the court administrator has access to. [AT]

5.4.1 Component Overview

The following information is available for all defendants and available from the defendant menu option.

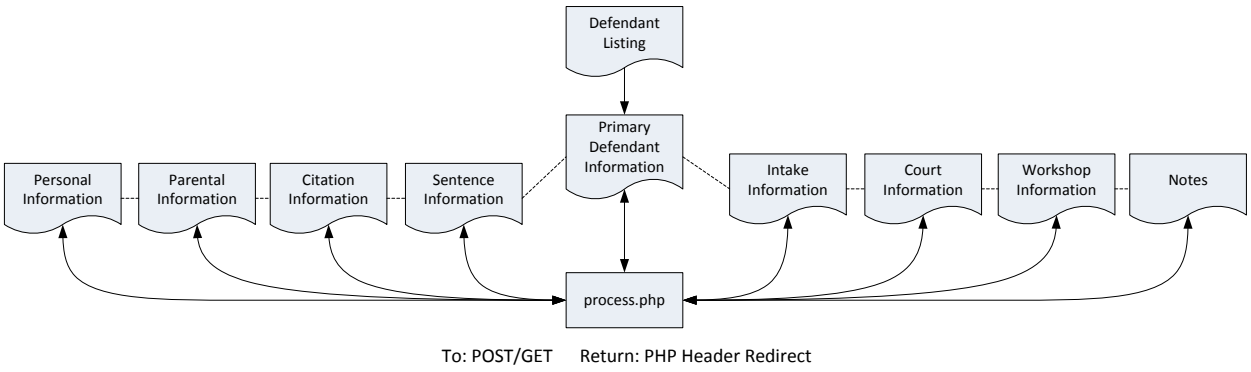

Figure 6: Defendant Architecture Diagram

5.4.1.1 Defendant Listing

This section uses DataTables for data display. JSON Source file is located at: /data/defendants.php

5.4.1.2 Defendant View

Primary Defendant Information

- Name
  - First Name
  - Middle Initial
  - Last Name
- Date of Birth
- Home phone number
- Added
- Court Case number
- Agency Case number
- Closed date
- Expunged date

The added field is an automatic timestamp.

Personal Information (Tab)

- Physical Address & Location
- Mailing Address & Location
- School Data
  - School Name & Location
  - Grade level
  - Contact Name
  - Contact Phone Number
- Driver's License Data
  - Number
  - State
- Physical Description
  - Height
  - Weight
  - Sex
  - Eye Color
  - Hair Color
  - Ethnicity

Once a location or school is added, that information can be selected using the select button. This is done to avoid duplication and to prevent user data errors.

Parent/Guardian Information (Tab)

- Relationship
- First Name
- Last Name
- Home Number
- Work Number
- Employer
- Email Address
- Physical Address & Location
- Mailing Address & Location
- Living with that parent/guardian

There is no limit on the number of guardian or parents.

Citation Information (Tab)

- Citation Date
- Citation Time
- Location
- Common Place
- Citing Officer Data
- Mirandized Checkbox
- Drugs or Alcohol Involved
- Offense Data
  - Statue
  - Title
  - Type
- Stolen/Vandalized Data
  - Description of items
  - Value or Amount
- Vehicle Data
  - Year
  - Make
  - Model
  - Color
  - License Number
  - License State
  - Comments

Intake Information (Tab)

- Intake Date
- Intake Time
- Reschedule Date
- Reschedule Time
- Intake Interviewer
- Referred to Juvenile – Not Qualified
- Dismissed / No Complaint

Court Information (Tab)

- Court Assigned
  - Type
  - Venue
  - Location
  - Closed
- Jury Assigned
  - Venue
  - Location
  - Closed
  - Hours Assigned

Sentence Information (Tab)

- Type
- Requirement
- Complete

Multiple sentence information can be entered and once a sentence is added, it can be selected and used again.

Workshop (Tab)

- Workshops Attending
  - Date
  - Name
  - Venue
  - Location
  - Completed

Expunge (Tab)

- Order Signed
- Letter Completed
- Case Completed
- Defendant Evaluation Completed
- Parent Evaluation Completed
- Workshop Completed
- Referred to Juvenile Date

5.4.2 Application Pages

The application pages containing this component are:

- /defendant/index.php
- /defendant/view.php

  These files are part of view.php by using php's include function:

  - /defendant/tab_personal.php
  - /defendant/tab_guardian.php
  - /defendant/tab_citation.php
  - /defendant/tab_intake.php
  - /defendant/tab_court.php
  - /defendant/tab_sentence.php
  - /defendant/tab_workshop.php

    - /defendant/tab_expunge.php
    - /defendant/tab_forms.php
    - /defendant/tab_notes.php
- /defendant/process.php – *all form POST/GET processing*
- /defendant/search.php
- /defendant/jquery.js – *JQuery functions*

5.4.3 Database Tables

Since the defendant is one of the primary components, it is used in a majority of areas via foreign key relations. These are the primary tables that defendant information is contained:

- teencour.defendant
- teencour.guardian
- teencour.citation
- teencour.citation_offense
- teencour.citation_stolen_items
- teencour.citation_vehicle
- teencour.defendant_sentence
- teencour.intake_information

5.4.4 Classes

The following classes are associated with this component:

- /includes/class_defendant.php
- /includes/class_location.php
- /includes/class_school.php
- /includes/class_guardian.php
- /includes/class_citation.php
- /includes/class_workshop.php
- /includes/class_workshop_location.php
- /includes/class_court.php
- /includes/class_court_location.php
- /includes/class_sentence.php
- /includes/class_data.php – *for JSON data*

5.4.5 Architecture Overview

Again, once some specific areas have been added, like location, schools, common locations, etc. – they will be available from the modal selection windows throughout the section.

Under the citation tab, the user can select existing offenses or immediately add a new one. Also, stolen items or vehicles involved can be added. There is no limit on how many can be added.

The court and workshop tabs just list what the defendant is currently enrolled in or assigned to. Adding the defendant needs to be done in those particular components.

When a defendant is expunged, it is done so based on the programs expunge type. Depending on this value, different data will be removed from the defendant record or not at all.

5.4.6 Design Overview

This section will be compromised of two main areas, the top defendant information and a multiple-tab area containing additional information described above. Once the primary information is added, the user will have access to the additional tabbed areas.

5.5 Volunteer

Volunteers are used to fulfill court positions or be a member of a jury. They need to be assigned to a court and have their hours tracked. [AT]

5.5.1 Component Overview

The following areas are accessible from the volunteer menu option.

5.5.1.1 Volunteer Listing

This section uses DataTables for data display. JSON Source file is located at: /data/volunteers.php

5.5.1.2 Volunteer View

Primary Information

- Active or not
- First Name
- Last Name
- Phone #
- Email

Tabbed Information

- Court Positions
- Hours volunteered

The court positions are generated with a default list when a program is added. This is done by the stored procedure call *addCourtPositions.* Program Administrators can add custom positions. View the court section for default member values. The hours volunteered display a list of courts they have been a member of or a part of a jury for.

5.5.2 Application Pages

The following application pages are where the main information is for the volunteer component:

- /volunteer/index.php
- /volunteer/view.php
- /volunteer/search.php
- /volunteer/process.php *(all form actions go here)*

5.5.3 Database Tables

The following MySQL tables are primarily used for this component:

- volunteer
- volunteer_position

5.5.4 Classes

The following classes are associated with this component:

- /includes/class_volunteer.php
- /includes/class_data.php – *for JSON data*

5.5.5 Architecture Overview

The volunteer is a pretty straight forward component. A person gets added to the program and court positions are selected. They can then be added to a court.

5.5.6 Design Overview

Once primary volunteer is added, the tabbed area comes available to select court positions. This is just a list of positions and checkboxes.

5.6 Courts

Courts are where the defendants will have their cases judged by a jury of their peers and/or volunteers. Volunteers can act as various court positions. Court positions are automatically generated when a program is created, but the Program Administrator has the ability to add more. The initial positions available are:

- Judge
- Prosecuting Attorney
- Defense Attorney
- Clerk
- Bailiff
- Exit Interviewer
- Advisor
- Jury

Volunteer hours are also recorded in this component. [AT]

5.6.1 Component Overview

The following areas are accessible from the court menu option or the initial page upon login.

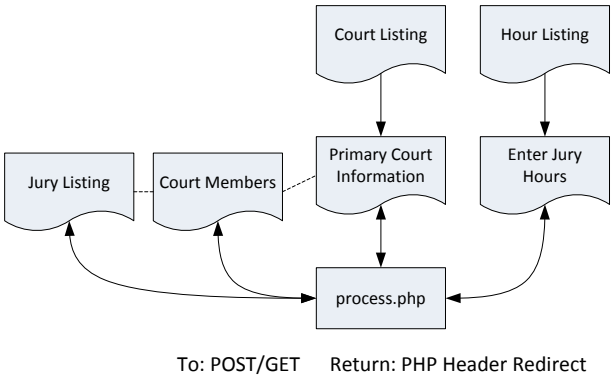

Figure 7: Court Architecture Diagram

5.6.1.1 Court Listing

This section uses DataTables for data display. JSON Source file is located at: /data/courts.php. These data tables are under the court listing area in the menu and the home menu item. The initial age upon login, or /main.php page contains a list of upcoming court dates.

5.6.1.2 Court View

- Court Date
- Court Time
- Court Type
- Contract Signed
- Closed
- Court Address
  - Name
  - Location
- Court Members
- Jury Members

The court name and location are recorded and can be selected from a list by the modal select windows.

5.6.2 Application Pages

The application pages containing this component are:

- /court/index.php
- /court/view.php

  These files are part of view.php by using php's include function:

  - /court/tab_members.php
  - /court/tab_jury.php

- /court/hour_entry.php
- /court/hours.php
- /court/process.php – *all form POST/GET processing*
- /court/search.php
- /court/jquery.js – *JQuery functions*

5.6.3 Database Tables

The following MySQL tables are primarily used for this component:

- court
- court_guardian
- court_defendant
- court_jury_defendant
- court_jury_volunteer
- court_location
- court_member
- court_position

5.6.4 Classes

The following classes are associated with this component:

- /includes/class_court.php
- /includes/c lass_court_location.php

5.6.5 Architecture Overview

When a court location is entered the first time, it can be selected using the modal selection window for future courts. This prevents re-entering common areas and data errors. The court member area is a series of dropdown input fields that are populated from the volunteer's available positions. Right now, a volunteer can be assigned multiple areas. Jury members are a listing of volunteers who have been marked for a jury position and any active defendants.

5.6.6 Design Overview

Once the initial court information is entered, the lower tab area will be visible. In this area, the court members, jury member and defendant tabs will be available.

5.7 Workshops

Workshops are primarily used as sentencing requirements for defendants. Once a workshop location is entered once, it can be selected with the modal selection button. Once a workshop is created, defendants can be added to it. If the defendant attended the workshop, they can be marked as having it completed. [AT]

5.7.1 Component Overview

The following areas are accessible from the workshop menu option.

5.7.1.1 Workshop Listing

This section uses DataTables for data display. JSON Source file is located at: /data/workshops.php.

5.7.1.2 Workshop View

- Title
- Date
- Time
- Instructor
- Officer
- Description
- Workshop Address
    - Venue Name
    - Address
    - Location

5.7.2 Application Pages

The application pages containing this component are:

- /workshop/index.php
- /workshop/view.php
- /workshop/search.php
- /workshop/process.php – *all form POST/GET processing*
- /workshop/jquery.js – *JQuery functions*

5.7.3 Database Tables

The following MySQL tables are primarily used for this component:

- teencour.workshop
- teencour.workshop_location
- teencour.workshop_roster

5.7.4 Classes

The following classes are associated with this component:

- /includes/class_workshop.php
- /includes/class_workshop_location.php

5.7.5 Architecture Overview

The workshop listing is fairly straight forward. Once the initial information is entered, the user will be able to add participants to it. Once a workshop location is added, it can be selected next time with the modal selection window. When a defendant completes the workshop, clicking on the completed link will assign the current timestamp to it.

5.7.6 Design Overview

The participant listing will not show up until the initial information has been entered.

5.8 Statistics

Statistics will not be implemented in this version.

5.9 Surveys

Surveys will not be implemented in this version.

5.10 Program Administration Options

Not yet implemented – will be developed during summer 2013.

5.11 Application Administration Options

Not yet implemented – will be developed during summer 2013.

6.0 Testing

Testing will cover all of the requirements within 3.0 Requirements. A traceability matrix is included to track results. [RR]

6.1 Requirement Testing

6.1.1 Log in, recovery, and register

6.1.1.1 Log in with valid email - Requirements 1, 2

Expected: Login to program main screen.

Input: Email Address = robert.reilly@mines.sdsmt.edu

Password = test

Result: Logged in, presented with main screen.

6.1.1.2 Log in with invalid email - Requirement 1

Expected: Error message stating wrong name or password.

Input: Email Address = blah@blah.blah

Password = blah

Result: Error message to contact admin.

6.1.1.3 Recover password with valid email - Requirement 1

Expected: Typing in current email sends recovery email sent with new password.

Input: Email Address = robert.reilly@mines.sdsmt.edu

Result: No email sent, password not changed.

BUG Need to send email

6.1.1.4 Register new user via register page - Requirements 1, 2

Expected: Typing in valid code and filling out form registers a new user.

Input: Code = TEST123

First name = Joe

Email = Joe@joe.com

Password = teencourt

Result: New user registered to system, set to inactive.

6.1.1.5 Register new user via register page with invalid code, Requirement 1

Expected: Error message saying the code is invalid.

Input: Teen/Youth Court Program Code = test

Result: Validation prevents registration, requires valid code.

6.1.1.6 Help menu - Requirement 1

Expected: Presented with a help menu to provide information about the website.

Result: Sent to index if I had yet to log in, sent to empty page if previously logged in.

BUG Need to get help pages, make it external

6.1.2 Site Admin Functions

6.1.2.1 Create New Program - Requirement 2

Expected: Filling out required fields allows creation of new program.

Input: Court Name = Test Two

Code = 123TEST

Result: New program created with required information.

6.1.2.2 Edit Existing Program fields - Requirement 2

Expected: Setting the program to inactive will mark program as inactive.

Input: Active = No (dropdown option)

Result: Program remains active, update has no effect.

Expected: Changing program name updates recorded court name.

Input: Court Name = Test

Result: Program name updated.

Expected: Changing program code updates recorded code.

Input: Code = 123TEST456

Result: Program code updated.

Expected: Changing phone number update recorded phone number.

Input: Phone Number = 123-456-7890

Result: Phone number updated.

Expected: Changing timezone updates corded timezone.

Input: Timezone = Pacific Standard Time (PST) (dropdown option)

Result: Timezone updated.

Expected: Changing expunge type updates recorded expunge type.

Input: Expunge Type = Partial (dropdown option)

Result: Expunge type updated.

Expected: Changing physical address information updates recorded physical address.

Input: Street = 123 Example St.

City = Test Town

State = CA

Zip = 97732

Result: Physical address updated.

Expected: Changing mailing address information updates recorded mailing address.

Input: Street = 456 Example St.

City = Test Town

State = CA

Zip = 97732

Result: Mailing address updated.

6.1.2.3 Delete Existing Program, -Requirement 2

Expected: Deleting the program removed the program from the database.

Input: Delete program button

Result: No effect on clicking Delete Program.

BUG Delete Program not implemented

6.1.2.4 Add New User - Requirement 1, 2

Expected: Filling out required fields allows creation of new user.

Input: First Name = Joe

Email Address = joe@joe.com

Force Password = teencourt

Result: New user created with required information

6.1.2.5 Edit Existing User fields - Requirements 1, 2

Expected: Changing active status updates recorded active.

Input: Active = Yes (dropdown option)

Result: Active status updated.

Expected: Changing first name updates recorded first name.

Input: First Name = Joey

Result: First name updated.

Expected: Changing last name updates recorded last name.

Input: Last Name = Smith

Result: Last name updated.

Expected: Changing email address updates recorded email address.

Input: joey@joe.com

Result: Email address updated.

Expected: Changing password sets new password.

Input: teen

Result: Password updated.

Expected: Changing program updates user program.

Input: Test (dropdown option)

Result: User program updated.

Expected: Changing access level updates user access level.

Input: Program Manager (dropdown option)

Result: User access level updated.

Expected: Changing timezone updates user timezone.

Input: Central Standard Time (CST) (dropdown option)

Result: User timezone updated.

6.1.2.6 Delete Existing User - Requirements 1, 2

Expected: Deleting the user removes them from the database.

Result: User was deleted from the database.

6.1.3 Program Admin/Manager Functions, Defendant Tab

6.1.3.1 Add new Defendant - Requirement 3

Expected: Filling out required fields allowed addition of new defendant

Input: First Name = Tommy

Last Name = Lee

Date of Birth = 06/09/1997

Court Case # = 1845

Result: Defendant added with required fields

6.1.3.2 Update Defendant Information fields - Requirement 3

Expected: Changing first name updates recorded first name.

Input: First Name = Thomas

Result: First name updated.

Expected: Changing last name updates recorded last name.

Input: Last Name = Wilson

Result: Last name updated.

Expected: Changing middle name updates recorded middle name.

Input: Middle Name = Lee

Result: Middle name updated.

Expected: Changing date of birth updates recorded date of birth.

Input: Date of Birth = 07/08/1998

Result: Date of birth updated.

Expected: Word in date of birth will give error.

Result: Letters not able to be typed in field.

Expected: Non-date value in date of birth will give error.

Input:Date of Birth = 123456

Result: Date does not change for incorrectly formatted date.

Expected: Changing phone number updates recorded phone number.

Input:Phone Number = 123-4567

Result: Phone Number updated.

Expected: Changing court case # updates recorded court case #.

Input:Court Case # = 038495

Result: Court Case # updated.

Expected: Changing agency case # updates recorded agency case #.

Input:Agency Case # = 482

Result: Agency Case # updated.

6.1.3.3 Update Defendant Personal Information fields - Requirement 3

Expected: Changing Physical Address information updates recorded physical address information.

Input: Address = 123 Test St.

City = Deadwood (Select Existing Location option)

State = SD (Select Existing Location option)

Zip = 57732 (Select Existing Location option)

Result: Physical Address information updated.

Expected: Changing Mailing Address information updates recorded mailing address information.

Input: Address = 321 Test St.

City = Deadwood (Select Existing Location option)

State = SD (Select Existing Location option)

Zip = 57732 (Select Existing Location option)

Result: Mailing Address information updated.

Expected: Using an address not in the existing address system will added it to recorded locations.

Input: City = Sturgis

State = SD

Zip = 57785

Result: Location was added to listing.

Expected: Changing school information updates recorded school information.

Input: School Name = Central High School (Select Existing Location option)

Grade = 12

Address = 1246 Main Blvd.(Select Existing Location option)

City = Deadwood (Select Existing Location option)

State = SD (Select Existing Location option)

Zip = 57732 (Select Existing Location option)

School Contact = Mr. Jones

Phone = 123-4567

Result: School Information updated.

Expected: Using a school not in the existing school selection will add it to recorded schools.

Input: School Name = Lead-Deadwood Middle School

Address = 234 South Main

City = Deadwood

State = SD

Zip = 57732

Result: School was added to listing.

Expected: Changing description updates recorded description

Input: Height = 6'0"

Weight = 165

Sex = Male (dropdown option)

Eye Color = Brown (dropdown option)

Hair Color = Black (dropdown option)

Ethnicity = Caucasian (dropdown option)

Result: Description updated.

Expected: Changing driver’s license updates recorded driver’s license.

Input: Number = 1348679

State = SD

Result: Driver's License updated.

6.1.3.4 Update Defendant Guardian fields - Requirement 3

Expected: Adding New Parent/Guardian updates recorded information.

Input: Relationship = Mother (dropdown option)

First Name = Sarah

Last Name = Wilson

Home Phone = 987-6543

Work Phone = 234-5678

Employer = Regional Hospital

Email Address = SarahWilson@yahoo.com

Defendant lives with this person? = No (dropdown option)

Result: Parent/Guardian updated.

Expected: Changing Physical Address updates recorded physical address

Input: Address = 321 Test St.

City = Deadwood (Select Existing Location option)

State = SD (Select Existing Location option)

Zip = 57732 (Select Existing Location option)

Result: Physical Address updated.

Expected: Changing Mailing Address updates recorded mailing address

Input: Address = 321 Test St.

City = Deadwood (Select Existing Location option)

State = SD (Select Existing Location option)

Zip = 57732 (Select Existing Location option)

Result: Physical Address updated.

Expected: Delete Guardian to remove the guardian.

Result: Guardian deleted from defendant information.

6.1.3.5 Update Defendant Citation fields - Requirement 3

Expected: Changing Citation Information updates recorded citation information.

Input: Date = 04/01/2013

Time = 3:50 PM

Address: 789 Test St.

Common Place = Highway 44 exit (Select Existing Location option)

City = Deadwood (Select Existing Location option)

State = SD (Select Existing Location option)

Zip = 57732 (Select Existing Location option)

Officer = Johnson, Randall (dropdown option)

Miranda Given? = Yes (dropdown option)

Drugs/Alcohol? = Yes (dropdown option)

Result: Citation Information updated.

Expected: Using a common place not in the existing common place selection will add it to recorded common places

Input: Common Place = Interstate 90 exit

Result: Common Place added.

Expected: Adding a new officer will include the officer in the dropdown.

Input: Identification = 18492

Last Name = Aker

First Name = Larry

Phone Number = 567-7654

Result: Officer added to dropdown list.

Expected: Adding an existing offense updates recorded offenses.

Input: 22-21-1 Invasions of Privacy (Add Existing Offense option)

Result: Offenses updated.

Expected: Adding a new statute will include it in the Add Existing Offense options.

Input: Code = 22-30A-1

Title = Theft

Description = Any person who takes, or exercises unauthorized control over, property of another, with intent to deprive that person of the property

Result: Statute added to defendant and Add Existing Offense options.

Expected: Deleting an offense will remove it from the defendant's citation.

Result: Offense was removed.

Expected: Adding stolen/vandalized item will add it to the defendant's citation.

Input: Item = Car Stereo

Value = 150

Result: Item added.

Expected: Deleting an item will remove it from the defendant's citation.

Result: Item was removed.

Expected: Adding a vehicle involved will add it to the defendant's citation.

Input: Year = 1998

Make = Toyota

Model = Corolla

Color = White

License = 49BA78

State = SD

Comment = Vehicle was wrecked.

Result: Vehicle added.

Expected: Deleting a vehicle will remove it from the defendant's citation.

Result: Vehicle was removed.

6.1.3.6 Update Defendant Intake fields - Requirement 3

Expected: Updating Intake Information will updated the recorded intake information.

Input: Intake Date = 04/11/2013

Intake Time = 12:10 AM

Intake Reschedule Date = 04/18/2013

Intake Reschedule Time = 4:15 PM

Intake Interviewer = Account, Test (dropdown option)

Referred to Juvenile - Not Qualified = Checked

Dismissed - No Complaint = Checked

Result: Intake Information updated.

Expected: Word in Intake Date field to give error.

Result: Letters unable to be typed in field.

Expected: Non-date number in Intake Date field to give error.

Result: Format forces date

Expected: Word in Intake Time field to give error.

Result: Letters able to be typed in field, crashes update.

BUG Need to force AM/PM

Expected: Non-time number in Intake Time field to give error.

Result: Format forces time

Expected: Word in Reschedule Date field to give error.

Result: Letters unable to be typed in field.

Expected: Non-date number in Reschedule Date field to give error.

Result: Format forces date

Expected: Word in Reschedule Time field to give error.

Result: Letters able to be typed in field, crashes update.

BUG Need to force AM/PM

Expected: Non-date number in Reschedule Time field to give error.

Result: Format forces time

6.1.3.7 Update Defendant Court fields - Requirement 3

Expected: Defendants not in a court will be able to be assigned to one.

Result: Option to assign to court available.

Expected: Defendants assigned to court will have court information available.

Result: Court information available.

Expected: Defendants assigned to multiple courts will have all court information available.

Result: All courts displayed.

Expected: Defendants not on a jury will have information saying they are not on one.

Result: Not assigned anywhere information available.

Expected: Defendants assigned to one jury pool will have court information available.

Result: Court jury information available.

Expected: Defendants assigned to multiple jury pools will have court information for all courts available.

Result: All courts displayed.

6.1.3.8 Update Defendant Workshops fields - Requirement 3

Expected: Option to create workshops from tab.

Result: Create Workshop moves user to 'New Workshop' page.

Expected: Defendant in one or more workshops will have information for all workshops displayed.

Result: All workshop information displayed.

6.1.3.9 Update Defendant Expunge fields - Requirement 3, 3.1

Expected: Marking all checkboxes or entering date in expunge field will expunge defendant from program.

Result: Expunge Defendant does nothing.

BUG Need to get Expunge defendant working

6.1.3.10 Update Defendant Forms - Requirement 3, 6

Not implemented

6.1.3.11 Update Defendant Case Notes - Requirement 3

Expected: Updating case notes will update recorded case notes

Input: Test

Result: Case Notes updated.

6.1.4 Program Admin/Manager Functions, Volunteer Tab

6.1.4.1 Add Volunteer - Requirement 4

Expected: Filling out all required fields would add the volunteer to the database.

Input: Last Name = Miller

Phone # = 605-787-5431

Result: New volunteer added.

6.1.4.2 Edit Existing Volunteer Fields - Requirement 4

Expected: Changing active status updates recorded active status.

Input: Active? = No (dropdown option).

Result: Active updated.

Expected: Changing first name updates recorded first name.

Input: First Name = Seth

Result: First Name updated.

Expected: Changing last name updates recorded last name.

Input: Last Name = King

Result: Last Name updated.

Expected: Changing phone number updates recorded phone number.

Input: Phone # = 605-134-5787

Result: Phone # updated.

Expected: Changing email updates recorded email.

Input: Email = seth_king@gmail.com

Result: Email updated.

Expected: Changing volunteer positions updates recorded volunteer positions.

Input: Advisor = Checked

Clerk = Checked

Exit Interviewer = Checked

Result: Volunteer positions updated.

6.1.4.3 Delete Volunteer - Requirement 4

Expected: Clicking delete volunteer removes them from the database.

Result: Volunteer was set to inactive status.

6.1.5 Program Admin/Program Manager, Courts Tab

6.1.5.1 Add new court - Requirement 3

Expected: Filling out all required fields will add the court to the database.

Input: Defendant = Wilson, Thomas (select defendant option)

Court Date = 04/16/2013

Court Time = 12:30 PM

Result: New Court added

6.1.5.2 Edit existing court - Requirement 3, 4

Expected: Changing defendant updates recorded defendant.

Input: Defendant = Smith, Timothy (select defendant option)

Result: Defendant updated.

Expected: Changing court date updates recorded court date.

Input:4/22/13

Result: Court Date updated.

Expected: Error when trying to type in letters to court date field.

Result: Unable to type letters.

Expected: Error when trying to type non-date number to court date field.

Result: Field forces date format, gives requirement error if field is not a date and prevents longer dates.

Expected: Changing court time updates recorded court time.

Input:02:45 PM

Result: Court Time updated.

Expected: Error when trying to type in letters to court time field.

Result: Unable to type letters.

BUG Need to force AM/PM

Expected: Error when trying to type non-time number into court time field.

Result: Field forces time format, gives requirement error if field is not a time and prevents longer times.

Expected: Changing court type updates recorded court type.

Input: Court Type = Hearing (dropdown option)

Result: Court Type updated

Expected: Changing contract signed updates recorded answer for contract signed.

Input: Contract Signed? = No (dropdown option)

Result: Contract Signed updated

Expected: Marking court as closed will close the court.

Input: Closed = Checked

Result: Court was closed.

Expected: Selecting a court location will update court location.

Input: Name = Deadwood Courthouse (Select Existing location option)

Address = 78 Sherman Street (Select Existing location option)

City = Deadwood (Select Existing location option)

State = SD (Select Existing location option)

Zip = 57732 (Select Existing location option)

Result: Court Location updated.

Expected: Using a new court location will update court location and record it as an existing court location.

Input: Name = Spearfish Middle School

Address = 1600 N Canyon St.

City = Spearfish (Select Existing location option)

State = SD (Select Existing location option)

Zip = 57783 (Select Existing location option)

Result: Court Location updated and location added to program existing locations.

Expected: Changing court members updates recorded court members.

Input: Advisor = Carter, Alan (dropdown option)

Bailiff = Crawford, Cindy (dropdown option)

Clerk = Dawson, Roger (dropdown option)

Defense Attorney = Smith, Joe (dropdown option)

Exit Interviewer = Carter, Alan (dropdown option)

Judge = Buck, Joe (dropdown option)

Prosecuting Attorney = Jones, Tom (dropdown option)

Result: Court members updated

Expected: Changing Jury members updates recorded jury members.

Input: Volunteer, Murray, Sarah (Add Jury Members option)

Defendant, Doe, Dave (Add Jury Members option)

Result: Jury pool updated.

Expected: Deleting Jury members removes them from this court's jury.

Result: Selected jury member deleted.

Expected: Changing parents/guardians updates parents/guardians to be present at the court.

Input: Smith, Bob = Yes (dropdown option, name is defendant's father)

Result: Present parents/guardians updated.

6.1.5.3 Delete Court - Requirement 3

Expected: Deleting a court will remove it from the database.

Result: Court was deleted from database.

6.1.5.4 Add Hours - Requirement 3, 4

Expected: Adding hours to a court will mark court members and jury pool as having worked for those hours.

Input: Globally set all hours = 2

Result: All court members and jury pool was set as having worked two hours in database.

6.1.6 Program Admin/Program Manager, Workshops Tab

6.1.6.1 Add Workshop - Requirement 5

Expected: Filling out all required fields will add the workshop to the database.

Input: Title = Alcohol Abuse

Date = 04/19/2013

Time = 07:00 PM

Instructor = Marlene Todd

Result: Workshop was added.

6.1.6.2 Edit Workshop - Requirement 3, 5

Expected: Changing title will updated recorded title.

Input: Title = Alcohol

Result: Title updated.

Expected: Changing date will updated recorded date.

Input: Date = 04/18/2013

Result: Date updated.

Expected: Error when trying to type in letters to date field.

Result: Unable to type letters.

Expected: Error when trying to type non-date number to date field.

Result: Field forces date format, gives requirement error if field is not a date and prevents longer dates.

Expected: Changing time updates recorded court time.

Input:06:30 PM

Result: Time updated.

Expected: Error when trying to type in letters to time field.

Result: Unable to type letters.

BUG Need to force AM/PM

Expected: Error when trying to type non-date number into time field.

Result: Field forces time format, gives requirement error if field is not a time and prevents longer times.

Expected: Changing officer updates recorded officer.

Input: Johnson, Randall (dropdown option)

Result: Officer updated

Expected: Changing description updates recorded description.

Input: Description = Discussion on effects of alcohol abuse.

Result: Description updated.

Expected: Changing workshop location will updated recorded location.

Input: Name = Lead Courthouse (Select Existing Location option)

Address = 425 Main St (Select Existing Location option)

City = Lead (Select Existing Location option)

State = SD (Select Existing Location option)

Zip = 57725 (Select Existing Location option)

Result: Workshop Location updated.

Expected: Adding a new workshop location will add it to the database and update recorded location.

Input: Name = Spearfish Middle School

Address = 1600 N Canyon St.

City = Spearfish

State = SD

Zip = 57783

Result: Location added to existing locations and workshop location updated.

Expected: Selecting a workshop participant will add them to the workshop.

Result: Defendant was added to the workshop.

Expected: Marking participant as completed will record the time in database.

Result: Current timestamp is recorded for workshop completion.

Expected: Removing a participant will remove them from the workshop.

Result: Selected participant was removed.

6.1.6.3 Delete Workshop – Requirement 5

Expected: Deleting workshop will remove it from the database.

Result: Workshop was deleted from the database.

6.1.7 Program Admin/Program Manager, Users Tab

6.1.7.1 Add New User - Requirement 1, 2

Expected: Filling out required fields allows creation of new user.

Input: First Name = Joe

Email Address = joe@joe.com

Force Password = teencourt

Result: New user created with required information

6.1.7.2 Edit Existing User fields - Requirements 1, 2

Expected: Changing active status updates recorded active.

Input: Active = Yes (dropdown option)

Result: Active status updated.

Expected: Changing first name updates recorded first name.

Input: First Name = Joey

Result: First name updated.

Expected: Changing last name updates recorded last name.

Input: Last Name = Smith

Result: Last name updated.

Expected: Changing email address updates recorded email address.

Input:joey@joe.com

Result: Email address updated.

Expected: Changing password sets new password.

Input: teen

Result: Password updated.

Expected: Changing access level updates user access level.

Input: Program Manager (dropdown option)

Result: User access level updated.

Expected: Changing time zone updates user time zone.

Input: Central Standard Time (CST) (dropdown option)

Result: User time zone updated.

6.1.7.3 Delete Existing User - Requirements 1, 2

Expected: Deleting the user removes them from the database.

Result: User was deleted from the database.

6.1.8 Program User functions

6.1.8.1 View Defendant – Requirement 1

Expected: Defendant information would be listed in a read-only format.

Result: Defendant information could be read, but could not be altered.

6.1.8.2 View Volunteer – Requirement 1

Expected: Volunteer information would be listed in a read-only format.

Result: Volunteer information could be read, but could not be altered.

6.1.8.3 View Court – Requirement 1

Expected: Court information would be listed in a read-only format.

Result: Court information could be read, but could not be altered.

6.1.8.4 View Court Hours – Requirement 1

Expected: Court hours would be listed in a read-only format.

Result: Court hours could be read, but could not be altered.

6.1.8.5 View Workshops – Requirement 1

Expected: Workshops would be listed in a read-only format.

Result: Workshops could be read, but could not be altered.

6.1.9 User Profile

6.1.9.1 Edit Profile – Requirement 1

Expected: Changing email updates recorded email.

Input: Email Address = robert@mines.sdsmt.edu

Result: Email updated.

Expected: Changing password updates recorded password.

Input: Reset Password = teencourt

Result: Password updated.

Expected: Changing first name updated recorded first name.

Input: First Name = Robert

Result: First Name updated.

Expected: Changing last name updates recorded last name.

Input: Last Name = R

Result: Last name updated.

Expected: Changing time zone updates recorded time zone.

Input: Time zone = Central Standard Time (CST) (dropdown option)

Result: Time zone updated.

Expected: Adding a phone number updates recorded phone numbers.

Input: Type = Cell

Number = 605-863-0896

Extension = 0

Result: Phone number added

Expected: Removing a phone number removes it from recorded phone numbers.

Result: Phone number removed.

6.1.10 Program Verification

6.1.10.1 View other program defendants – Requirement 2

Expected: Trying to view a defendant outside of your own program will cause an error.

Result: Page prevents defendant data from being loaded.

6.1.10.2 View other program volunteers – Requirement 2

Expected: Trying to view a volunteer outside of your own program will cause an error.

Result: Page loads to enter a new volunteer, adding the volunteer inserts a new one in the correct program.

6.1.10.3 View other program courts – Requirement 2

Expected: Trying to view a court outside of your own program will cause an error.

Result: Page loads to enter a new court, adding the court inserts a new one in the correct program.

6.1.10.4 View other program workshops – Requirement 2

Expected: Trying to view a workshop outside of your own program will cause an error.

Result: Page loads to enter a new volunteer, adding the volunteer inserts a new one in the correct program.

6.2 Traceability Matrix

Test Cases	Testing for:	Req. 1	Req. 2	Req. 3	Req. 3.1	Req. 4	Req. 5	Req. 6	Test Pass/Fail	Comments
1.1	Valid email	X	X						Pass	
1.2	Invalid email	X							Pass	
1.3	Password recovery	X							Fail	No email sent
1.4	Register user	X	X						Pass	
1.5	Register invalid	X							Pass	
1.6	Help menu	X							Fail	No help menu, can not reach help page if not logged in
2.1	Create new program		X						Pass	
2.2	Edit program									
2.2.1	Active		X						Pass	
2.2.2	Court name		X						Pass	
2.2.3	Court code		X						Pass	
2.2.4	Phone		X						Pass	
2.2.5	Timezone		X						Pass	
2.2.6	Expunge Type		X						Pass	
2.2.7	Physical Address		X						Pass	
2.2.8	Mailing Address		X						Pass	
2.3	Delete Program		X						Fail	Delete program not done yet
2.4	Add new user	X	X						Pass	
2.5	Edit user									
2.5.1	Active	X	X						Pass	
2.5.2	First name	X	X						Pass	
2.5.3	Last name	X	X						Pass	
2.5.4	Email	X	X						Pass	
2.5.5	Password	X	X						Pass	

2.5.6	Program	X	X		Pass
2.5.7	Access level	X	X		Pass
2.5.8	Timezone	X	X		Pass
2.6	Delete User	X	X		Pass
3.1	Add Defendant			X	Pass
3.2	Edit Defendant				
3.2.1	First name			X	Pass
3.2.2	Last name			X	Pass
3.2.3	Middle name			X	Pass
3.2.4	Date of birth			X	Pass
3.2.5	Word for DoB			X	Pass
3.2.6	Invalid number DoB			X	Pass
3.2.7	Phone			X	Pass
3.2.8	Court case number			X	Pass
3.2.9	Agency number			X	Pass
3.3	Defendant Personal				
3.3.1	Physical Address			X	Pass
3.3.2	Mailing Address			X	Pass
3.3.3	New city/state/zip			X	Pass
3.3.4	School			X	Pass
3.3.5	New school			X	Pass
3.3.6	Description			X	Pass
3.3.7	Drivers license			X	Pass
3.4	Defendant Guardian				
3.4.1	Add new guardian			X	Pass
3.4.2	Physical Address			X	Pass
3.4.3	Mailing Address			X	Pass
3.4.4	Delete Guardian			X	Pass
3.5	Defendant Citation				
3.5.1	Citation information			X	Pass
3.5.2	New common place			X	Pass
3.5.3	New officer			X	Pass
3.5.4	Add existing offense			X	Pass
3.5.5	Add new offense			X	Pass
3.5.6	Delete offense			X	Pass
3.5.7	Add item			X	Pass
3.5.8	Delete item			X	Pass
3.5.9	Add vehicle			X	Pass
3.5.10	Delete vehicle			X	Pass
3.6	Defendant Intake				
3.6.1	Intake Information			X	Pass
3.6.2	Word for date			X	Pass

3.6.3	Non-date number	X				Pass	
3.6.4	Word in time	X				Fail	Need to force AM/PM
3.6.5	Non-time number	X				Pass	
3.6.6	Word in time	X				Pass	
3.6.7	Non-date number	X				Pass	
3.6.8	Word in reschedule	X				Fail	Need to force AM/PM
3.6.9	Non-time number	X				Pass	
3.7	Defendant Court						
3.7.1	Assign to court	X				Pass	
3.7.2	Court Information	X				Pass	
3.7.3	Multiple courts	X				Pass	
3.7.4	Not on jury	X				Pass	
3.7.5	Assign to jury	X				Pass	
3.7.6	Multiple juries	X				Pass	
3.8	Defendant Workshop						
3.8.1	Assign to workshop	X			X	Pass	
3.8.2	Workshop(s) Information	X			X	Pass	
3.9	Defendant Expunge						
3.9.1	Expunge Defendant	X	X			Fail	Not implemented
3.10	Defendant Forms	X			X	Fail	Not implemented
3.11	Defendant Case Notes	X				Pass	
4.1	Add Volunteer			X		Pass	
4.2	Edit Volunteer						
4.2.1	Active			X		Pass	
4.2.2	First name			X		Pass	
4.2.3	Last name			X		Pass	
4.2.4	Phone number			X		Pass	
4.2.5	Email			X		Pass	
4.2.6	Positions			X		Pass	
4.3	Delete Volunteer			X		Pass	
5.1	Add Court	X				Pass	
5.2	Edit Court						
5.2.1	Defendant	X				Pass	
5.2.2	Court Date	X				Pass	
5.2.3	Word in date	X				Pass	
5.2.4	Non-date number	X				Pass	
5.2.5	Court Time	X				Pass	

5.2.6	Word in time			X		Fail	Need to force AM/PM
5.2.7	Non-time number			X		Pass	
5.2.8	Court type			X		Pass	
5.2.9	Contract Signed			X		Pass	
5.2.10	Close			X		Pass	
5.2.11	Court Location			X		Pass	
5.2.12	New court location			X		Pass	
5.2.13	Court Members			X	X	Pass	
5.2.14	Jury			X	X	Pass	
5.2.15	Delete jury			X	X	Pass	
5.2.16	Parents/Guardians			X		Pass	
5.3	Delete Court			X		Pass	
5.4	Add Hours			X	X	Pass	
6.1	Add Workshop				X	Pass	
6.2	Edit Workshop						
6.2.1	Title				X	Pass	
6.2.2	Workshop Date				X	Pass	
6.2.3	Word in Date				X	Pass	
6.2.4	Non-date number				X	Pass	
6.2.5	Workshop time				X	Pass	
6.2.6	Word in time				X	Fail	Need to force AM/PM
6.2.7	Non-time number				X	Pass	
6.2.8	Officer				X	Pass	
6.2.9	Description				X	Pass	
6.2.10	Workshop Location				X	Pass	
6.2.11	New location				X	Pass	
6.2.12	Add Participant			X	X	Pass	
6.2.13	Participant Complete			X	X	Pass	
6.2.14	Remove Participant			X	X	Pass	
6.3	Delete Workshop				X	Pass	
7.1	Add New User	X	X			Pass	
7.2	Edit User						
7.2.1	Active	X	X			Pass	
7.2.2	First Name	X	X			Pass	
7.2.3	Last Name	X	X			Pass	
7.2.4	Email	X	X			Pass	
7.2.5	Password	X	X			Pass	

7.2.6	Access level	X	X	Pass	
7.2.7	Timezone	X	X	Pass	
7.3	Delete User	X	X	Pass	
8.1	View Defendant	X		Pass	
8.2	View Volunteer	X		Pass	
8.3	View Court	X		Pass	
8.4	View Court Hours	X		Pass	
8.5	View Workshops	X		Pass	
9.1	Edit Profile	X			
9.1.1	Email	X		Pass	
9.1.2	Password	X		Pass	
9.1.3	First Name	X		Pass	
9.1.4	Last Name	X		Pass	
9.1.5	Timezone	X		Pass	
9.1.6	Add phone number	X		Pass	
9.1.7	Delete phone number	X		Pass	
	View other defendants		X	Pass	
10.1	View other volunteers		X	Pass	
10.2	View other courts		X	Pass	
10.3	View other workshops		X	Pass	

7.0 Development Environment

This section describes programs used to develop the application, where it will be hosted and what kind of environment the server is.

7.1 Development IDE and Tools

For MySQL schema model and database management, MySQL Workbench will be used: http://dev.mysql.com/downloads/workbench/. Aptana Studio 3 will be used as the IDE for development of the web application and the prototype: http://www.aptana.com/products/studio3. The Aptana IDE has integrated GitHub support.

7.2 Source Control

GitHub will be used for the primary file repository and source control. Builds will be pushed to GitHub after every editing session. The link is: https://github.com/SDSMT-CSC/TCD.

7.3 Dependencies

PHP 5.2.17, MySQL 5.5, MySQL PDO, JQuery and JQueryUI using Google Hosted Library.

7.4 Build Environment

The build environment is a Linux server running Apache HTTP Web Server and MySQL Database Management System. The servers are located in Dallas, TX. The login credentials are located in Account Credentials. The dedicated IP address: 50.22.71.90

7.5 Development Machine Setup

The application will be built on the existing web space. In the event of any hosting or connection issues that may arise during development, the team has access to a private, temporary server with the same server environment. This way progress on the application will not be hindered while the issue is resolved. [AT]

8.0 Release | Setup | Deployment

The site is already released and live on http://teencourtdb.com. [AT]

8.1 Setup Information

An initial program with id set to 0 is the default program needed to add others. It is listed as Teen Court DB.

8.2 System Versioning Information

Prototypes were developed and numbered 1 to 4 and are available on GitHub. The most current version is always located at http://teencourtdb.com.

9.0 End User Documentation

A user guide will be made available in PDF format on GitHub and delivered to the client. [RR]

Appendix I: List of Figures

Figure 1: System Diagram 21
Figure 2: Major Component Diagram 21
Figure 3: User Access Levels 30
Figure 4: User Registration Architecture Diagram 32
Figure 5: User Access Architecture Diagram 33
Figure 6: Defendant Tab Diagram 33

Appendix II: Supporting Information and Details

II.1 Database Schema

```
CREATE TABLE IF NOT EXISTS `citation`
  (
    `citationid`     INT(10) UNSIGNED NOT NULL auto_increment,
    `defendantid`    INT(10) UNSIGNED NOT NULL,
    `officerid`      INT(10) UNSIGNED DEFAULT NULL,
    `date`           TIMESTAMP NULL DEFAULT NULL,
    `address`        VARCHAR(45) DEFAULT NULL,
    `locationid`     INT(11) DEFAULT NULL,
    `mirandized`     TINYINT(1) DEFAULT NULL,
    `drugsoralcohol` TINYINT(1) DEFAULT NULL,
    `commonplaceid`  INT(10) UNSIGNED DEFAULT NULL,
    `added`          TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`citationid`),
    KEY `fk_citation_court_common_place1_idx` (`commonplaceid`),
    KEY `fk_citation_defendant1_idx` (`defendantid`),
    KEY `fk_citation_citation_officer1_idx` (`officerid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `citation_offense`
  (
    `offenseid`   INT(10) UNSIGNED NOT NULL auto_increment,
    `defendantid` INT(10) UNSIGNED NOT NULL,
    `statuteid`   INT(10) UNSIGNED NOT NULL,
    PRIMARY KEY (`offenseid`)
  ) engine= innodb;

CREATE TABLE IF NOT EXISTS `citation_stolen_items`
  (
    `itemid`     INT(10) UNSIGNED NOT NULL auto_increment,
    `citationid` INT(10) UNSIGNED NOT NULL,
    `name`       VARCHAR(45) DEFAULT NULL,
    `value`      DECIMAL(18, 2) DEFAULT NULL,
    PRIMARY KEY (`itemid`),
    KEY `fk_citation_stolen_items_citation1_idx` (`citationid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `citation_vehicle`
  (
    `vehicleid`     INT(10) UNSIGNED NOT NULL auto_increment,
    `citationid`    INT(10) UNSIGNED NOT NULL,
    `licensenumber` VARCHAR(50) DEFAULT NULL,
    `licensestate`  VARCHAR(50) DEFAULT NULL,
    `make`          VARCHAR(50) DEFAULT NULL,
    `model`         VARCHAR(50) DEFAULT NULL,
    `year`          VARCHAR(50) DEFAULT NULL,
    `color`         VARCHAR(50) DEFAULT NULL,
    `comment`       TEXT,
    PRIMARY KEY (`vehicleid`),
    KEY `fk_citation_vehicle_citation1_idx` (`citationid`)
  ) engine=innodb;
```

```
CREATE TABLE IF NOT EXISTS `court`
 (
   `courtid`        INT(10) UNSIGNED NOT NULL auto_increment,
   `programid`      INT(10) UNSIGNED NOT NULL,
   `defendantid`    INT(10) UNSIGNED NOT NULL,
   `courtlocationid` INT(10) UNSIGNED DEFAULT NULL,
   `type`           VARCHAR(45) NOT NULL,
   `contract`       TINYINT(1) DEFAULT NULL,
   `date`           DATETIME NOT NULL,
   `timeentered`    TINYINT(1) NOT NULL DEFAULT '0',
   `closed`         DATETIME DEFAULT NULL,
   PRIMARY KEY (`courtid`, `programid`),
   KEY `fk_jury_court1_idx` (`programid`),
   KEY `fk_jury_defendant1_idx` (`defendantid`),
   KEY `fk_trial_trial_location1_idx` (`courtlocationid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `court_guardian`
 (
   `courtid`    INT(10) UNSIGNED NOT NULL,
   `guardianid` INT(10) UNSIGNED NOT NULL,
   PRIMARY KEY (`courtid`, `guardianid`),
   KEY `fk_trial_guardian_guardian1_idx` (`guardianid`),
   KEY `fk_trial_guardian_trial1_idx` (`courtid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `court_jury_defendant`
 (
   `courtid`    INT(10) UNSIGNED NOT NULL,
   `defendantid` INT(10) UNSIGNED NOT NULL,
   `hours`      DECIMAL(8, 2) DEFAULT NULL,
   PRIMARY KEY (`courtid`, `defendantid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `court_jury_volunteer`
 (
   `courtid`    INT(10) UNSIGNED NOT NULL,
   `volunteerid` INT(10) UNSIGNED NOT NULL,
   `hours`      DECIMAL(8, 2) DEFAULT NULL,
   PRIMARY KEY (`courtid`, `volunteerid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `court_location`
 (
   `courtlocationid` INT(11) NOT NULL auto_increment,
   `locationid`     INT(10) UNSIGNED NOT NULL,
   `programid`      INT(10) UNSIGNED NOT NULL,
   `name`           VARCHAR(45) DEFAULT NULL,
   `address`        VARCHAR(45) DEFAULT NULL,
   PRIMARY KEY (`courtlocationid`),
   KEY `locationid_unique` (`locationid`)
 ) engine=innodb;
```

```
CREATE TABLE IF NOT EXISTS `court_member`
 (
   `courtid`     INT(10) UNSIGNED NOT NULL,
   `volunteerid` INT(10) UNSIGNED NOT NULL,
   `positionid`  INT(10) UNSIGNED NOT NULL,
   `hours`       DECIMAL(8, 2) DEFAULT NULL,
   PRIMARY KEY (`courtid`, `positionid`, `volunteerid`),
   KEY `fk_jury_pool_jury1_idx` (`courtid`),
   KEY `fk_trial_members_trial_position1_idx` (`positionid`),
   KEY `fk_jury_pool_volunteer1` (`volunteerid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `court_position`
 (
   `positionid` INT(10) UNSIGNED NOT NULL auto_increment,
   `programid`  INT(10) UNSIGNED NOT NULL,
   `position`   VARCHAR(45) NOT NULL,
   PRIMARY KEY (`positionid`),
   KEY `fk_jury_position_court1` (`programid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `custom_data`
 (
   `customid`    INT(10) UNSIGNED NOT NULL,
   `defendantid` INT(10) UNSIGNED NOT NULL,
   `value`       VARCHAR(45) DEFAULT NULL,
   PRIMARY KEY (`customid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `custom_fields`
 (
   `fieldid`   INT(10) UNSIGNED NOT NULL auto_increment,
   `programid` INT(10) UNSIGNED NOT NULL,
   `fieldname` VARCHAR(45) DEFAULT NULL,
   `required`  TINYINT(1) DEFAULT NULL,
   PRIMARY KEY (`fieldid`, `programid`),
   UNIQUE KEY `fieldid_unique` (`fieldid`),
   KEY `fk_custom_fields_court1_idx` (`programid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `defendant`
 (
   `defendantid`       INT(10) UNSIGNED NOT NULL auto_increment,
   `programid`         INT(10) UNSIGNED NOT NULL,
   `firstname`         VARCHAR(45) NOT NULL,
   `lastname`          VARCHAR(45) NOT NULL,
   `middlename`        VARCHAR(45) DEFAULT NULL,
   `homephone`         VARCHAR(45) DEFAULT NULL,
   `dob`               VARCHAR(25) NOT NULL,
   `paddress`          VARCHAR(150) DEFAULT NULL,
   `plocationid`       INT(11) DEFAULT NULL,
   `maddress`          VARCHAR(150) DEFAULT NULL,
   `mlocationid`       INT(11) DEFAULT NULL,
   `schoolid`          INT(11) DEFAULT NULL,
   `schoolcontactname` VARCHAR(150) DEFAULT NULL,
   `schoolcontactphone` VARCHAR(50) DEFAULT NULL,
   `schoolgrade`       VARCHAR(15) DEFAULT NULL,
   `height`            VARCHAR(25) DEFAULT NULL,
   `weight`            VARCHAR(25) DEFAULT NULL,
```

```
    `eyecolor`          VARCHAR(25) DEFAULT NULL,
    `haircolor`         VARCHAR(25) DEFAULT NULL,
    `sex`              VARCHAR(25) DEFAULT NULL,
    `ethnicity`         VARCHAR(25) DEFAULT NULL,
    `licensenum`          VARCHAR(25) DEFAULT NULL,
    `licensestate`       VARCHAR(25) DEFAULT NULL,
    `notes`             TEXT,
    `courtcasenumber`    VARCHAR(45) DEFAULT NULL,
    `agencycasenumber`   VARCHAR(45) DEFAULT NULL,
    `expungedate`        DATETIME DEFAULT NULL,
    `closedate`          DATE DEFAULT NULL,
    `added`             TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`defendantid`, `programid`),
    KEY `fk_defendant_program1_idx` (`programid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `defendant_sentence`
  (
    `defsentid`   INT(10) UNSIGNED NOT NULL auto_increment,
    `sentenceid`  INT(11) NOT NULL,
    `defendantid` INT(11) NOT NULL,
    `value`       VARCHAR(255) DEFAULT NULL,
    `additional`  VARCHAR(255) DEFAULT NULL,
    `complete`    DATE DEFAULT NULL,
    PRIMARY KEY (`defsentid`),
    KEY `index` (`sentenceid`),
    KEY `defendant` (`defendantid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `guardian`
  (
    `guardianid`  INT(10) UNSIGNED NOT NULL auto_increment,
    `defendantid` INT(10) UNSIGNED NOT NULL,
    `relation`    VARCHAR(25) DEFAULT NULL,
    `firstname`   VARCHAR(45) DEFAULT NULL,
    `lastname`    VARCHAR(45) DEFAULT NULL,
    `paddress`    VARCHAR(150) DEFAULT NULL,
    `plocationid` INT(11) DEFAULT NULL,
    `maddress`    VARCHAR(45) DEFAULT NULL,
    `mlocationid` INT(11) DEFAULT NULL,
    `homephone`   VARCHAR(45) DEFAULT NULL,
    `employer`    VARCHAR(45) DEFAULT NULL,
    `workphone`   VARCHAR(45) DEFAULT NULL,
    `email`       VARCHAR(100) DEFAULT NULL,
    `liveswith`   TINYINT(1) DEFAULT NULL,
    PRIMARY KEY (`guardianid`),
    KEY `fk_parents_defendant1_idx` (`defendantid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `intake_information`
  (
    `defendantid` INT(11) NOT NULL,
    `intake`      DATETIME NOT NULL,
    `reschedule`  DATETIME DEFAULT NULL,
    `inteviewer`  INT(11) NOT NULL,
    `referred`    DATETIME DEFAULT NULL,
    `dismissed`   DATETIME DEFAULT NULL,
    PRIMARY KEY (`defendantid`)
  ) engine=innodb;
```

```
CREATE TABLE IF NOT EXISTS `program`
 (
   `programid`  INT(10) UNSIGNED NOT NULL auto_increment,
   `code`       VARCHAR(75) NOT NULL,
   `name`       VARCHAR(150) NOT NULL,
   `paddress`   VARCHAR(150) DEFAULT NULL,
   `pcity`      VARCHAR(45) DEFAULT NULL,
   `pstate`     CHAR(2) DEFAULT NULL,
   `pzip`       VARCHAR(25) DEFAULT NULL,
   `maddress`   VARCHAR(150) DEFAULT NULL,
   `mcity`      VARCHAR(45) DEFAULT NULL,
   `mstate`     CHAR(2) DEFAULT NULL,
   `mzip`       VARCHAR(25) DEFAULT NULL,
   `phone`      VARCHAR(25) NOT NULL,
   `expunge`    TINYINT(4) NOT NULL,
   `timezoneid` INT(10) NOT NULL,
   `active`     TINYINT(1) NOT NULL DEFAULT '1',
   `added`      TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (`programid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `program_common_location`
 (
   `commonplaceid` INT(10) UNSIGNED NOT NULL auto_increment,
   `programid`     INT(10) UNSIGNED NOT NULL,
   `commonplace`   VARCHAR(45) DEFAULT NULL,
   PRIMARY KEY (`commonplaceid`, `programid`),
   KEY `fk_court_common_place_court1_idx` (`programid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `program_locations`
 (
   `locationid` INT(11) NOT NULL auto_increment,
   `programid`  INT(11) NOT NULL,
   `city`       VARCHAR(50) NOT NULL,
   `state`      CHAR(2) NOT NULL,
   `zip`        VARCHAR(25) NOT NULL,
   PRIMARY KEY (`locationid`),
   KEY `fk_programid_program` (`programid`)
 ) engine=myisam;

CREATE TABLE IF NOT EXISTS `program_officers`
 (
   `officerid` INT(10) UNSIGNED NOT NULL auto_increment,
   `programid` INT(10) UNSIGNED NOT NULL,
   `firstname` VARCHAR(45) DEFAULT NULL,
   `lastname`  VARCHAR(50) DEFAULT NULL,
   `idnumber`  VARCHAR(50) NOT NULL,
   `phone`     VARCHAR(45) DEFAULT NULL,
   PRIMARY KEY (`officerid`),
   KEY `fk_citation_officer_court1_idx` (`programid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `program_schools`
 (
   `schoolid`   INT(10) UNSIGNED NOT NULL auto_increment,
   `programid`  INT(10) UNSIGNED NOT NULL,
   `schoolname` VARCHAR(45) DEFAULT NULL,
   `address`    VARCHAR(45) DEFAULT NULL,
```

```
    `city`       VARCHAR(45) DEFAULT NULL,
    `state`      VARCHAR(45) DEFAULT NULL,
    `zip`        VARCHAR(45) DEFAULT NULL,
    PRIMARY KEY (`schoolid`, `programid`),
    KEY `fk_school_information_program1_idx` (`programid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `program_sentences`
  (
    `sentenceid`  INT(10) UNSIGNED NOT NULL auto_increment,
    `programid`   INT(10) UNSIGNED NOT NULL,
    `name`        VARCHAR(150) DEFAULT NULL,
    `description` VARCHAR(150) DEFAULT NULL,
    `additional`  VARCHAR(255) DEFAULT NULL,
    PRIMARY KEY (`sentenceid`),
    KEY `programid` (`programid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `program_statutes`
  (
    `statuteid`   INT(10) UNSIGNED NOT NULL auto_increment,
    `programid`   INT(10) UNSIGNED NOT NULL,
    `statute`     VARCHAR(50) DEFAULT NULL,
    `title`       VARCHAR(50) DEFAULT NULL,
    `description` TEXT,
    PRIMARY KEY (`statuteid`, `programid`),
    UNIQUE KEY `statuteid_unique` (`statuteid`),
    KEY `fk_citation_court1_idx` (`programid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `survey`
  (
    `surveyid`    INT(10) UNSIGNED NOT NULL auto_increment,
    `description` MEDIUMTEXT,
    `created`     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `sent`        DATETIME DEFAULT NULL,
    PRIMARY KEY (`surveyid`),
    UNIQUE KEY `surveyid_unique` (`surveyid`)
  ) engine=innodb;

CREATE TABLE IF NOT EXISTS `survey_answers`
  (
    `surveyid`   INT(10) UNSIGNED NOT NULL auto_increment,
    `userid`     INT(10) UNSIGNED NOT NULL,
    `questionid` INT(10) UNSIGNED NOT NULL,
    `answer`     VARCHAR(45) DEFAULT NULL,
    PRIMARY KEY (`surveyid`, `userid`, `questionid`),
    KEY `fk_survey_answers_survey1_idx` (`surveyid`),
    KEY `fk_survey_answers_survey_questions1_idx` (`questionid`),
    KEY `fk_survey_answers_user1_idx` (`userid`)
  ) engine=innodb;
```

```
CREATE TABLE IF NOT EXISTS `survey_options`
 (
   `optionid`    VARCHAR(45) NOT NULL,
   `questionid` INT(10) UNSIGNED NOT NULL,
   `option`     VARCHAR(45) DEFAULT NULL,
   PRIMARY KEY (`optionid`),
   KEY `fk_survey_options_survey_questions1_idx` (`questionid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `survey_questions`
 (
   `questionid` INT(10) UNSIGNED NOT NULL auto_increment,
   `surveyid`   INT(10) UNSIGNED NOT NULL,
   `question`   VARCHAR(255) DEFAULT NULL,
   `type`       INT(10) UNSIGNED DEFAULT NULL,
   PRIMARY KEY (`questionid`, `surveyid`),
   UNIQUE KEY `questionid_unique` (`questionid`),
   KEY `fk_survey_questions_survey1_idx` (`surveyid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `timezone`
 (
   `timezoneid` INT(11) NOT NULL auto_increment,
   `display`    VARCHAR(100) NOT NULL,
   `timezone`   VARCHAR(100) NOT NULL,
   PRIMARY KEY (`timezoneid`)
 ) engine= innodb;

CREATE TABLE IF NOT EXISTS `user`
 (
   `userid`     INT(10) UNSIGNED NOT NULL auto_increment,
   `programid`  INT(10) UNSIGNED NOT NULL,
   `typeid`     INT(10) UNSIGNED NOT NULL,
   `firstname`  VARCHAR(75) NOT NULL,
   `lastname`   VARCHAR(75) NOT NULL,
   `email`      VARCHAR(255) NOT NULL,
   `timezoneid` INT(10) NOT NULL,
   `hash`       CHAR(128) NOT NULL,
   `createdate` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   `lastlogin`  DATETIME DEFAULT NULL,
   `active`     TINYINT(1) NOT NULL DEFAULT '0',
   `deleted`    TINYINT(1) NOT NULL DEFAULT '0',
   PRIMARY KEY (`userid`),
   KEY `programid` (`programid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `user_log`
 (
   `logid`      INT(10) UNSIGNED NOT NULL auto_increment,
   `userid`     INT(10) UNSIGNED NOT NULL,
   `action`     VARCHAR(45) NOT NULL,
   `recordid`   INT(11) DEFAULT NULL,
   `ip_address` VARCHAR(45) NOT NULL,
   `date`       TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (`logid`),
   KEY `fk_user_log_user1` (`userid`)
 ) engine=innodb;
```

```
CREATE TABLE IF NOT EXISTS `user_phone`
 (
   `phoneid` INT(10) UNSIGNED NOT NULL auto_increment,
   `userid` INT(10) UNSIGNED NOT NULL,
   `phonenum` VARCHAR(45) NOT NULL,
   `ext` VARCHAR(5) DEFAULT NULL,
   `type` VARCHAR(25) DEFAULT NULL,
   PRIMARY KEY (`phoneid`),
   KEY `fk_user_phone_user1` (`userid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `user_type`
 (
   `typeid` INT(10) UNSIGNED NOT NULL auto_increment,
   `type` VARCHAR(50) NOT NULL,
   `active` TINYINT(1) NOT NULL,
   PRIMARY KEY (`typeid`)
 ) engine=myisam;

CREATE TABLE IF NOT EXISTS `volunteer`
 (
   `volunteerid` INT(10) UNSIGNED NOT NULL auto_increment,
   `programid` INT(10) UNSIGNED NOT NULL,
   `firstname` VARCHAR(45) DEFAULT NULL,
   `lastname` VARCHAR(45) DEFAULT NULL,
   `phone` VARCHAR(45) DEFAULT NULL,
   `email` VARCHAR(255) DEFAULT NULL,
   `active` TINYINT(1) DEFAULT '1',
   PRIMARY KEY (`volunteerid`, `programid`),
   UNIQUE KEY `volid_unique` (`volunteerid`),
   KEY `fk_volunteer_court1_idx` (`programid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `volunteer_position`
 (
   `volunteerid` INT(10) UNSIGNED NOT NULL,
   `positionid` INT(10) UNSIGNED NOT NULL,
   PRIMARY KEY (`volunteerid`, `positionid`),
   KEY `fk_volunteer_position_court_position1_idx` (`positionid`),
   KEY `fk_volunteer_position_volunteer1_idx` (`volunteerid`)
 ) engine=innodb;

CREATE TABLE IF NOT EXISTS `workshop`
 (
   `workshopid` INT(10) UNSIGNED NOT NULL auto_increment,
   `programid` INT(10) UNSIGNED NOT NULL,
   `date` TIMESTAMP NULL DEFAULT NULL,
   `title` VARCHAR(45) DEFAULT NULL,
   `description` TEXT,
   `instructor` VARCHAR(45) DEFAULT NULL,
   `officerid` INT(11) DEFAULT NULL,
   `workshoplocationid` INT(10) DEFAULT NULL,
   PRIMARY KEY (`workshopid`, `programid`)
 ) engine=innodb;
```

```
CREATE TABLE IF NOT EXISTS `workshop_location`
 (
   `workshoplocationid` INT(11) NOT NULL auto_increment,
   `locationid`        INT(11) NOT NULL,
   `programid`          INT(11) NOT NULL,
   `name`              VARCHAR(45) DEFAULT NULL,
   `address`           VARCHAR(45) DEFAULT NULL,
   PRIMARY KEY (`workshoplocationid`)
 ) engine= innodb;

CREATE TABLE IF NOT EXISTS `workshop_roster`
 (
   `workshopid`  INT(10) UNSIGNED NOT NULL,
   `defendantid` INT(10) UNSIGNED NOT NULL,
   `completed`   DATETIME DEFAULT NULL,
   PRIMARY KEY (`workshopid`, `defendantid`),
   KEY `fk_workshop_roster_workshop1_idx` (`workshopid`),
   KEY `fk_workshop_roster_defendant1_idx` (`defendantid`)
 ) engine=innodb;

ALTER TABLE `citation`
 ADD CONSTRAINT `fk_citation_court_common_place1` FOREIGN KEY (`commonplaceid`)
 REFERENCES `program_common_location` (`commonplaceid`) ON DELETE no action ON
 UPDATE no action,
 ADD CONSTRAINT `fk_citation_defendant1` FOREIGN KEY (`defendantid`) REFERENCES
 `defendant` (`defendantid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `citation_stolen_items`
 ADD CONSTRAINT `fk_citation_stolen_items_citation1` FOREIGN KEY (`citationid`)
 REFERENCES `citation` (`citationid`) ON DELETE no action ON UPDATE no action,
 ADD CONSTRAINT `fk_stolen_items_citation` FOREIGN KEY (`citationid`)
 REFERENCES `citation` (`citationid`);

ALTER TABLE `citation_vehicle`
 ADD CONSTRAINT `fk_citation_vehicle_citation1` FOREIGN KEY (`citationid`)
 REFERENCES `citation` (`citationid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `court`
 ADD CONSTRAINT `fk_jury_court1` FOREIGN KEY (`programid`) REFERENCES `program`
 (`programid`) ON DELETE no action ON UPDATE no action,
 ADD CONSTRAINT `fk_jury_defendant1` FOREIGN KEY (`defendantid`) REFERENCES
 `defendant` (`defendantid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `court_guardian`
 ADD CONSTRAINT `fk_trial_guardian_guardian1` FOREIGN KEY (`guardianid`)
 REFERENCES `guardian` (`guardianid`) ON DELETE no action ON UPDATE no action,
 ADD CONSTRAINT `fk_trial_guardian_trial1` FOREIGN KEY (`courtid`) REFERENCES
 `court` (`courtid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `court_jury_defendant`
 ADD CONSTRAINT `fk_jury_member_def_trial1` FOREIGN KEY (`courtid`) REFERENCES
 `court` (`courtid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `court_member`
 ADD CONSTRAINT `fk_jury_pool_jury1` FOREIGN KEY (`courtid`) REFERENCES `court`
 (`courtid`) ON DELETE no action ON UPDATE no action,
 ADD CONSTRAINT `fk_jury_pool_volunteer1` FOREIGN KEY (`volunteerid`)
 REFERENCES `volunteer` (`volunteerid`) ON DELETE no action ON UPDATE no action
,
```

```
  ADD CONSTRAINT `fk_trial_members_trial_position1` FOREIGN KEY (`positionid`)
  REFERENCES `court_position` (`positionid`) ON DELETE no action ON UPDATE no
  action;

ALTER TABLE `court_position`
  ADD CONSTRAINT `fk_jury_position_court1` FOREIGN KEY (`programid`) REFERENCES
  `program` (`programid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `custom_data`
  ADD CONSTRAINT `fk_custom_data_custom_fields1` FOREIGN KEY (`customid`)
  REFERENCES `custom_fields` (`fieldid`) ON DELETE no action ON UPDATE no action
;

ALTER TABLE `custom_fields`
  ADD CONSTRAINT `fk_custom_fields_court1` FOREIGN KEY (`programid`) REFERENCES
  `program` (`programid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `defendant`
  ADD CONSTRAINT `fk_defendant_program1` FOREIGN KEY (`programid`) REFERENCES
  `program` (`programid`);

ALTER TABLE `guardian`
  ADD CONSTRAINT `fk_parents_defendant1` FOREIGN KEY (`defendantid`) REFERENCES
  `defendant` (`defendantid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `program_common_location`
  ADD CONSTRAINT `fk_court_common_place_court1` FOREIGN KEY (`programid`)
  REFERENCES `program` (`programid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `program_officers`
  ADD CONSTRAINT `fk_citation_officer_court1` FOREIGN KEY (`programid`)
  REFERENCES `program` (`programid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `program_sentences`
  ADD CONSTRAINT `program_sentences_ibfk_1` FOREIGN KEY (`programid`) REFERENCES
  `program` (`programid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `program_statutes`
  ADD CONSTRAINT `fk_citation_court1` FOREIGN KEY (`programid`) REFERENCES
  `program` (`programid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `survey_answers`
  ADD CONSTRAINT `fk_survey_answers_survey1` FOREIGN KEY (`surveyid`) REFERENCES
  `survey` (`surveyid`) ON DELETE no action ON UPDATE no action,
  ADD CONSTRAINT `fk_survey_answers_survey_questions1` FOREIGN KEY (`questionid`
  ) REFERENCES `survey_questions` (`questionid`) ON DELETE no action ON UPDATE
  no action,
  ADD CONSTRAINT `fk_survey_answers_user1` FOREIGN KEY (`userid`) REFERENCES
  `user` (`userid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `survey_options`
  ADD CONSTRAINT `fk_survey_options_survey_questions1` FOREIGN KEY (`questionid`
  ) REFERENCES `survey_questions` (`questionid`) ON DELETE no action ON UPDATE
  no action;

ALTER TABLE `survey_questions`
  ADD CONSTRAINT `fk_survey_questions_survey1` FOREIGN KEY (`surveyid`)
  REFERENCES `survey` (`surveyid`) ON DELETE no action ON UPDATE no action;
```

```
ALTER TABLE `user`
  ADD CONSTRAINT `fk_user_court` FOREIGN KEY (`programid`) REFERENCES `program`
  (`programid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `user_log`
  ADD CONSTRAINT `fk_user_log_user1` FOREIGN KEY (`userid`) REFERENCES `user` (
  `userid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `user_phone`
  ADD CONSTRAINT `fk_user_phone_user1` FOREIGN KEY (`userid`) REFERENCES `user`
  (`userid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `volunteer`
  ADD CONSTRAINT `fk_volunteer_court1` FOREIGN KEY (`programid`) REFERENCES
  `program` (`programid`) ON DELETE no action ON UPDATE no action;

ALTER TABLE `volunteer_position`
  ADD CONSTRAINT `fk_volunteer_position_court_position1` FOREIGN KEY (
  `positionid`) REFERENCES `court_position` (`positionid`) ON DELETE no action
  ON UPDATE no action,
  ADD CONSTRAINT `fk_volunteer_position_volunteer1` FOREIGN KEY (`volunteerid`)
  REFERENCES `volunteer` (`volunteerid`) ON DELETE no action ON UPDATE no action
;

ALTER TABLE `workshop_roster`
  ADD CONSTRAINT `fk_workshop_roster_defendant1` FOREIGN KEY (`defendantid`)
  REFERENCES `defendant` (`defendantid`) ON DELETE no action ON UPDATE no action
,
  ADD CONSTRAINT `fk_workshop_roster_workshop1` FOREIGN KEY (`workshopid`)
  REFERENCES `workshop` (`workshopid`) ON DELETE no action ON UPDATE no action;


DELIMITER $$

CREATE DEFINER=`teencour`@`localhost` PROCEDURE `addCourtPositions`( IN pID INT )

BEGIN

        INSERT INTO court_position (programid,position) VALUES ( pid,'Judge');
        INSERT INTO court_position (programid,position) VALUES ( pid,'Prosecuting Attorney');
        INSERT INTO court_position (programid,position) VALUES ( pid,'Defense Attorney');
        INSERT INTO court_position (programid,position) VALUES ( pid,'Clerk');
        INSERT INTO court_position (programid,position) VALUES ( pid,'Bailiff');
        INSERT INTO court_position (programid,position) VALUES ( pid,'Exit Interviewer');
        INSERT INTO court_position (programid,position) VALUES ( pid,'Advisor');
        INSERT INTO court_position (programid,position) VALUES ( pid,'Jury');

END$$

DELIMITER;
```

Appendix III: Sprint Reports

This section will contain a complete list of all of the period progress and/or sprint reports which are deliverables for the phases and versions of the system. [AT]

III.1 Sprint 1 Progress Report

This report is to inform you of the completion of Sprint 1 and will list the team members, give a brief overview of the project and list any deliverables submitted to the GitHub repository. The team members for this senior design project are Andrew Thompson and Robert Reilly.

PROBLEM

Teen, or youth court, programs provide an alternative solution for first-time offenders by offering a peer-driven sentencing program instead of regular court prosecutions.

These types of programs results in reduced costs per case versus traditional justice system and lower recidivism rates. The problem comes with the ability to track defendants and gather statistics to reinforce the benefits of youth court programs. With a better solution to track and generate statistics, additional funding or budget increases could be an option for these programs.

SPONSOR

The Teen Court Database project is sponsored by Marlene Todd, Program Director for Lawrence County Teen Court. Working with the National Association of Youth Courts, this application could be marketed to any of the 1050+ youth courts across the nation. The statistics gathered could be used by local and state judicial systems.

SOLUTION

Marlene would like an application developed that would offer similar data tracking of their current Access Database. Other youth courts could be setup and allowed access to the application to track their own data. This information would be only accessible by the individual courts. The application administrators will be able to generate some statistics from data in the entire system. This would give real-time statistics for youth court programs on a national level.

PROJECT

This application will be developed on a server running Linux operating system and Apache HTTP web server. The language used to develop the application will be PHP. PHP Data Object will be used as a data access layer between the application and the database server.

MySQL will be used as the Relational Database Management System. The only cost associated with this project will be a hosting fee from eCoastDesign and the registration of a domain name.

Both of these fees will be paid by the product owner. The name of this project is still yet to be determined and Marlene was doing research on it. Possible names may be something neutral – like youthcourtdb.com.

WORK COMPLETED

During sprint 1, we worked on the initial documentation for the project requirements and design. We met with Marlene Todd on September 18th and discussed her vision of the project and what sort of problems they are having now. We also obtained a copy of the documentation for the Access Database

application they are currently using. From this we were able to make a list of requirements needed for the project. Robert Reilly worked on the Software Requirements Document and Andrew worked on the Software Design Document.

Andrew also was in contact with Michael Roth from eCoastStudios to discuss hosting options and server environments. This is the web design and hosting company that the National Association of Youth Courts uses. Marlene and Michael are currently working on the details for web hosting and domain registration.

DELIVERABLES

Uploaded to the GitHub repository are:

- Mission Statement
- Initial Software Requirements Document (SRD)
- Initial Software Design Document (SDD)
- Approved Software Agreement – will be signed at client presentation on October 11th.

NEXT SPRINT EXPECTATIONS

The next sprint is scheduled to start on October 10th and will be the design phase. Some of the items coming up in this sprint are:

- Design the database schema
- Develop class data and methods
- Initial design of the user interface
- Start prototype

BACKLOG ITEMS RETAINED

Items from last sprint not completed due to client action needed:

- Determine Domain Name
- Obtain Web Space

III.2 Sprint 2 Progress Report

This report is to inform you of the backlog completed during Sprint 2. During this timeframe, the team met briefly five times to discuss the current backlog and organize responsibilities. This was the design phase of the project, so Bobby Reilly was responsible for the PHP Class Object listing and Andrew Thompson was responsible for the MySQL Database Schema and initial prototype layout. We also decided on an IDE package, Aptana Studio 3, for development. This program allows GitHub access to make uploading our web files to the repository easier.

The domain name, teenourtdb.com, and web hosting has been decided upon. The team is just waiting for eEcoast Studios to send the account credentials. In the meantime, Andrew started working on the initial prototype layout on a local webserver that should mimic what the development environment should be.

Andrew visited with the client on October 26th in Deadwood, SD. The Software Agreement was signed and several items were discussed. We talked at length about the expunge defendant functionality. Since each youth court program handles expunging different, we decided to have three different expunging options available. This option will be set when the court program is created and it will allow a partial

expunge, full expunge and sealed record. We also discussed the user registration process. This information should be updated in the design document.

These documents are initial versions and will change and grow once we start the actual prototype and development of the application.

DELIVERABLES

Uploaded to the GitHub repository are files for Prototype 1:

- PHP Class Listing (data members and methods)
- MySQL Database Schema
- Updated Software Development Document (SDD)

NEXT SPRINT EXPECTATIONS

The next sprint is scheduled to start on November 7th and will be the prototype phase. Some of the items coming up in this sprint are:

- Implement the database schema
- Build HTML/CSS pages for the prototype

BACKLOG ITEMS RETAINED

We were unable to obtain the web space before the last spring ended.

Update: Website hosting obtained 11/4.

MEETING NOTES

10/4: Discuss potential classes needed for design.

10/9: Help Robert with PHP/class functionality and discuss database design.

10/18: Discuss results of client meeting and changes/additions to design.

10/23: Discuss layout options for defendant data and more database changes.

10/30: Final run through of database schema and potential class list.

III.3 Sprint 3 Progress Report

Sprint 3 was the client prototype build phase. During sprint 2, the initial look and feel of the application was decided upon. At that time, the login page and the preliminary defendant page were developed. We had some issues with finding time to meet and work on the application, so the majority of the work was done during the last couple weeks of the sprint. The prototype has no database functionality and no error checking.

Several externally-accessed pages were developed or improved upon. This includes the application login, user registration and password recovery pages. The rest of the files are the internal, or pages that will be secure. Most of the defendant area was designed but there still needs to be some organization/changes to the data. Other main areas that were designed include volunteers, court rosters, and workshops. These areas contain listing of items, ability to add, modify and search items. A user profile page was also designed. For the tabular data listings, we decided to use a JQuery plug-in called DataTables (http://datatables.net/). This uses JSON data to obtain a higher level of user control for the data view including searches and column sorting.

One of the issues we have is a display issue with the Internet Explorer browser. This is scheduled to be fixed over Christmas break – along with other minor code cleanup. Another issue was that the bandwidth limit for the web space was set incorrect. We used a backup web server to continue development until the admin at eCoastStudio was able to fix it.

Bobby was learning PHP/JQuery while doing this, but he was able to complete the Volunteer and Workshop sections and organize the majority of the presentation. Andrew completed the remainder of the prototype sections and made modifications to the MySQL database.

We have a client meeting with Marlene Tuesday, December 18th to present the prototype and finalize a few questions we have about data placement and functionality.

During Christmas break, we plan to start implementing functionality to the site starting with the user accounts and access.

Prototype Location: http://teencourtdb.com/

Database Access (via Cpanel):

- URL: http://teencourtdb.com/cpanel
- Username: teencour
- Password: ifHQS14zvcN2

DELIVERABLES

Uploaded to the GitHub repository are files for Prototype 2:

- Sprint 3 Report
- Web files

BACKLOG ITEMS RETAINED

- Prototype: Reports
- Prototype: Statistics
- Prototype: Administration areas

We are just going to implement these in the next sprints. Some of the items, like statistics, need to be defined and have some information in the database to ensure proper data generation.

MEETING NOTES

11/13: Discuss ways to implement data displays, including the use of JSON

11/22: Discussion on who will work on what sections of the prototype

11/27: Short coding session to teach fundamental layout of files/structure

12/2: Long coding session finishing up prototype

III.4 Sprint 4 Progress Report

Most of sprint 4 was spent on the user access portion of the project. This included the database interaction for looking up users, granting access, and session information. The hashing process for the user password was completed. This uses a 128 character string that stores a salted hash for their email address and password. This is regenerated every time a user logs in. Session security and auto-logout times were completed. One last part of user access that needs to be implemented is limiting the number of failed login attempts.

We decided to add time zone information for users and programs. Users will have their time zone set to the default one the program uses, but will be able to change it within their control panel. This time zone gets set when the user logs in and will display any database times and dates consistent with the user's local time zone. Modifications to the database included adding fields to the program and user tables and a new table for available time zones. These time zones are in PHP format and all six major time in the United States are listed – with a special zone for non-DST in Arizona. The following is an example of the display name and PHP value:

timezoneID	Display	timezone
1	Eastern Standard Time (EST)	America/New_York
2	Central Standard Time (CST)	America/Chicago

The next portion that was worked on was the registration page and admin section for managing programs and users. The PHP/HTML pages were completed and work was started on the data listing class. This class will be used to generate JSON formatted database information that will be used to list users, programs, courts, etc. The registration page is mostly complete and just needs the add user functionality in the user class. These areas should be completed by the start of the next sprint (Tuesday, February 12th at the latest).

Lastly, form validation was implemented via a JQuery plugin. This plugin allows interaction with forms to validate data entry. For example, on the registration page, the email is checked asynchronously to verify it is not already in use, passwords match, and required fields are checked.

BACKLOG ITEMS RETAINED

We selected more backlog items than we could have accomplished this sprint. We wanted to get the entire administrator areas for programs and users completed, but fell short. The remaining backlog items include the user and program implementation for administrators, the program class code and individual user control panel. We should have a good start on the data listing class and the entire user access and registration should be done by Tuesday, February 12th.

NEXT SPRINT EXPECTATIONS

During sprint 5, we expect to have the administration sections completed and start on the defendant management. This is the major part of the site and will require a significant amount of work. Some of it may need to be spread out over the final sprint.

MEETING NOTES

1/10/13: Discuss current state of prototype. Andy covered IE fixes and brought up the idea to change to RECAPTCHA, a free captcha service.

1/15/13: IE Fixes done and captcha changed. Discuss work for court program/user admin areas. Robert was assigned the layout for the HTML portion of these pages while Andy will work on the user login and password encryption.

1/22/13: We discussed User Access and Registration functionality. Also, decided to add Time zone information for users so anything with timestamps in the database stay local.

1/29/13: Bobby was assigned the data class. Discussed how it should function (PHP class -> MYSQL interaction -> JSON encoding). Also, continuing work on User Registration / Admin areas… admin gets separate menu, etc.

2/7/13: We talked about why the data class is behind schedule. Bobby is still learning the interaction between PHP Classes and MYSQL. Assigned a due date for Tuesday, February 12th.

III.5 Sprint 5 Progress Report

Sprint 5 was comprised of finishing the user and program admin areas left over from the last sprint and beginning work on the main area of the site. This included the defendant, workshop, volunteer, and court management portions. We wanted the main functionality to be complete but several items were left in the backlog. We also knew there would be some final design/architecture work to do in the final sprint, such as the reports section. This was going to be worked on while Bobby was testing. Unfortunately, we are still behind schedule by a few weeks.

WORK COMPLETED

As Andrew was finishing the program and user admin areas, Bobby started work on the Volunteer section. Andrew also finished some user areas. These areas were the user action/history logging functionality and profile page. This last section is where the user can manage their information, including password change and phone number list.

The defendant area is the largest and most complex of the entire site. Andrew worked on this section. The defendant is broken up into several areas – or tabs. Each tab required its own functionality and architecture. Along with the primary defendant information, the completed tabs were personal information, guardian/parent information, citation information and defendant intake. This was a majority of JQuery, datatables, and PHP class implementation. Several classes were created to handle citations, schools and locations. The way users select and add locations were redesigned to be used wherever a city/state/address is needed. This was done to reduce the amount of malformed, incorrect or duplicate information in the database. Some AJAX functionality was added in order to insert data into a table without refreshing the page. Originally, much more of the site was supposed to use this but due to time constraints, URL redirection is used in some cases.

Quite a few global functions were completed during this time. Bobby was having an issue where a user generated time string based on their time zone would get entered into the database. When this date-time field would be read from the database, it would be converted and adjusted to the user's time zone. The problem is the user generated time need to be converted to the server time before inserting into the tables otherwise the date-time would be off. A global function was created to do that.

Bobby worked on the volunteer and workshop areas. This included adding functionality to the classes and the PHP/HTML implementation. These areas are much smaller than the defendant, so it was good place for him to practice the JQuery/PHP scripting.

MEETING SUMMARY

Andrew has the flu from 2/12/2013 to 2/14/2013 so no meetings were held during that time.

2/19/2013: Discuss which areas would be worked on and by whom and various global design concepts. Tasks were assigned via Trello.

2/21/2013: Discussed JQuery/AJAX/Datatable interaction concepts.

2/26/2013: Progress check, still behind schedule.

2/28/2013: No meeting held.

03/04/2013 - 03/09/2013, Spring Break: Several emails conversations took place between Andrew and Robert and from Andrew and Marlene. This was mainly to give the client an update and get some clarification on a few items like defendant sentencing.

3/12/2013: Discussed user/database time issue. Andrew worked up a solution.

REMAINING BACKLOG

I know we were supposed to have the main functionality completed at this point, but several items remain from sprint 5's backlog. This includes an important portion of the site, the court administration area. This is where a court/trial is setup with defendants and volunteers. Other areas include defendant expungement and defendant sentencing. Some clarification was needed on the court sentencing so Marlene and Andrew had an email discussion regarding the functionality of this area. The expungement area closes or seals the dependent on the programs expunge type. Another main area we haven't gotten to is the Program Administrator's page. This is where the program admin can modify any data collected by the managers and other admins. This includes items such as locations, common areas, officer data, state statutes, schools, etc.

III.6 Sprint 6 Progress Report

Sprint 6 was spent finishing core functionality of the application before the design fair. Bobby also started side-wide testing and documentation. The poster for the design fair was also completed and presented to the class. Suggestions made during the peer review were made before the poster was printed and mounted. I wanted to wait until after the design fair to write the sprint report since we had a review meeting with our client during it. She had brought up a few issues described below.

WORK COMPLETED

Andrew worked on finishing up some of the main areas of the application. This included some cleanup to the workshop and volunteer area, the primary court and court location components, the court and sentencing areas in the defendant view. Bobby began side-wide testing, various bug fixes, the volunteer hours tab and class/method documentation.

We also ran into a problem where unauthorized access was gained to the site and JavaScript code was injected into index.php pages and any file with the .js extension. We both ran malware detection on our systems but think it came from the host server and not us. The files were cleaned up and the FTP password was changed.

During our presentation to Marlene at the design fair there seemed to be some confusion as to how the court section worked. It was initially designed that a court was bound to a defendant, but Marlene wants the court to be separate and stand-alone. Multiple defendants can then be added to a court after it is created. The changes should be easy enough to implement and will be added as a task for the summer sprint.

MEETING SUMMARY

3/26/2013: Discussed layout for poster.

3/28/2013: Decided to go with logoed polo shirts for the design fair and extend the branding for the site this way. We also decided on lanyards with name badges. Bobby was also sent a checklist for documentation and testing.

4/4/2013: Discuss electronic TV poster vs. printed poster and what how we wanted the table setup for the design fair.

4/16/2013: Technically this happened after Sprint 6 ended, but I wanted to include the application demo we had with our client and a change brought up. Marlene noted she has some money to finish development of the application.

REMAINING BACKLOG

Several key functionalities of the application did not get implemented. This includes the defendants expungement, the program administration and the ability to delete (set flag) a defendant.

NEXT SPRINT EXPECTATIONS

Dr. McGough and Robert decided he could continue throughout the summer to replace an incomplete grade. This will be considered Sprint 7 and will be tracked via Trello and GitHub as usual. He will be responsible for the sprint report and any client or manager interactions. The following areas need to be implemented:

- Program Administration
- Court Modification per Marlene
- Defendant Expunge
- Detailed Searches
  - Defendant
  - Courts
  - Volunteer
  - Workshops
- Contact page
- Reports / PDF generation

Depending on what get completed over the summer, Marlene has some money to finish development and that will be discussed next fall.

Appendix IV: Class Documentation

This section will contain a complete list of all PHP classes used covering class purpose, class members, and class methods.

IV.1 Citation

Description: The citation class stores information about a defendant's citation information, what offenses they are charged with, what items were stolen, and if vehicles were used. [RR]

Class Members

citationID: Private, Unique identifier for this citation.

defendantID: Private, Unique identifier for the defendant.

officerID: Public, Unique identifier for the arresting officer.

citationDate: Public, Date of citation.

address: Public, Address of citation location.

locationID: Public, Unique identifier for location. Holds city, state, and zip fields.

commonLocationID: Public, Unique identifier for common places. Holds a short description of the location.

mirandized: Public, Whether the defendant was mirandized or not.

drugsOrAlcohol: Public, Whether drugs or alcohol was present at the time of citation.

Class Methods

__construct($ defendantID): Public, Loads citation information for the given defendantID into citation object, otherwise returns with an empty object.

updateCitation(): Public, Updates the citation information in the database if citationID is filled, otherwise it creates a new row and inserts the information.

getOffenseList($user_type): Public, Gets the offenses for the defendant based off of defendantID and citationID. Option to remove offense is visible depending on user type.

addOffense($statuteID): Public, Adds new offense to record.

removeOffense($offenseID): Public, Removes offense from record.

getStolenItems(): Public, Gets stolen items based off of citationID.

addStolenItem($name, $value): Public, Records the name and value of an item and adds to record.

removeStolenItem($itemID): Public, Removes an item from record.

getVehicles():Public, Gets vehicles used in offense.

addVehicle($year, $make, $model, $color, $license, $state, $comment): Public, Adds a vehicle to record.

removeVehicle($vehicleID): Public, Removes a vehicle from record.

getDefendantID():Public, Returns DefendantID.

getCitationID():Public, Returns CitationID.

IV.2 Core

Description: Core class used to handle connections to the database. [RR]

Class Members

dbh: Public, PDO for connecting to the database.

instance: Private static, object holding instance of open database connection

Class Methods

__construct(): Private, Sets up PDO object.

dbOpen(): Public static, Create a new instance of a database connection.

dbClose(): Public static, Closes the database connection.

covertToServerDate($originalTS, $userTimeZone): Public, Converts time stamps to Chicago time zone to correctly store within the database.

IV.3 Court Location

Description: The court location handles information for where courts will be held. [RR]

Class Members

courtLocationID: Private, Unique identifier for this court location

programID: Private, Unique identifier for the program.

locationID: Public, Unique identifier for location. Holds city, state, and zip fields.

name: Public, Location name.

address: Public, Location address.

city: Public, Location city, brought in with locationID.

state: Public, Location state, brought in with locationID.

zip: Public, Location zip, brought in with locationID.

Class Methods

__construct():Public, create empty court location object.

updateCourtLocation(): Public, adds a new court location to database if it doesn't exist in the database, otherwise updates the location if they don't match.

getCourtLocation($workshopLocID): Public, loads court location object with data based on courtLocID.

getCourtLocationID(): Public, returns courtLocationID.

getProgramID(): Public, returns programID.

setLocationID($val): Public, sets locationID.

setProgramID($val): Public, sets programID.

IV.4 Court

Description: The court class is used to assign defendants to court and manage court positions filled by volunteers.

Class Members

courtID: Private, unique identifier for court.

programID: Private, unique identifier for program.

defendantID: Private, unique identifier for defendant.

courtDate: Public, date and time court is to be held.

type: Public, what kind of court it is (hearing, trial)

contractSigned: Public, if the defendant signed the contract to go through the Teen Court program.

closed: Public, if the court was completed.

courtLocationID: Public, unique identifier for court location.

timeEntered: Public, mark if the time has been set for members of this court

Class Methods

__construct($user_programID): Public, create an empty court object.

updateCourt(): Public, adds court if courtID is 0, otherwise updates court.

getFromID($id): Public, gets court information from courtID.

compareProgramID($id, $user_program): Public, returns true if user's programID matches court's programID.

deleteCourt(): Public, removes everything from this court.

getCourtMembers(): Public, get court members for this program.

updateCourtMembers($members): Public, update all assigned court members.

existingCourtMembers(): Public, Gets a list of existing court members for a particular court, used to make volunteer active in the court member dropdown lists.

getJuryMembers(): Public, Gets a list of existing court members for a particular court, used to make volunteer active in the court member dropdown lists.

updateJuryMember(): Public, update all assigned jury members.

deleteJuryMember($id, $type): Public, deletes a jury member from the assigned jury pool.

updateCourtGuardians($guardians): Public, updates guardians attending a particular court.

checkGuardianAttending(): Public, returns an array of guardians attending a particular court, used to check dropdowns.

getMembers for Time($type): Public, returns an array of members for a particular court and their time spent.

setMembersTime($globalHrs, $members, $jury): Public, sets time spent for members/jury on a particular court.

setDefendantID($val): Public, sets defendantID.

setCourtID($val): Public, sets courtID.

getDefendantID(): Public, return defendantID.

getCourtID(): Public, return courtID.

IV.5 Data

Description: The Data class is used to fetch information that is displayed with the dataTables and dropdown menu options. [RR]

Class Members

None

Class Methods

fetchUserListing($user_programID, $user_type): Public, fetches all users or only users in the program depending on user type into JSON encoding.

fetchProgramListing():Public, fetches all programs into JSON encoding.

fetchProgramLocations($user_programID): Public, fetches locations for the given program into a JSON object.

fetchProgramSchools($user_programID): Public, fetches schools for the given program into a JSON object.

fetchProgramStatutes($user_programID): Public, fetches statutes for the given program into a JSON object.

fetchProgramSentences($user_programID): Public, fetches sentence for the given program into a JSON object.

fetchProgramDropdown($id): Public, generates a dropdown list of programs that are active, marks a program as selected depending on $id.

fetchUserTypeDropdown($id, $utype): Public, fetches the possible user types that a user could be assigned into a JSON object.

fetchTimezoneDropdown($id): Public, generates a dropdown list of possible timezones.

fetchCourtListing($user_programID): Public, fetches all courts for the given program into a JSON object.

fetchCourtLocation($user_programID): Public, fetches court locations for the given program into a JSON object.

fetchCourtJuryPool($user_programID): Public, fetches possible jury pool members for the given program into a JSON object.

checkJuror($courtID, $jurorID, $type): Public, checks a juror to see if they are available for the court listing or not.

fetchDefendantListing($user_programID): Public, fetches defendants for the given program who are still going through the Teen Court program into a JSON object.

fetchVolunteerListing($user_programID): Public, fetches active volunteers for the given program into a JSON object.

fetchWorkshopListing($user_programID): Public, fetches workshops for the given program into a JSON object.

fetchWorkshopDefendantsListing($user_programID): Public, fetches defendants from the given program to be workshop participants into a JSON object.

fetchWorkshopLocation($user_programID): Public, fetches workshop locations for the given program into a JSON object.

fetchProgramCommonLocation($user_programID): Public, fetches common places for the given program into a JSON object.

IV.6 Defendant

Description: The defendant class is used to handle and protect information about defendants within the Teen Court program.

Class Members

defendantID: Private, unique identifier for the defendant.

programID: Private, unique identifier for the program.

firstName: Private, defendant's first name.

lastName: Private, defendant's last name.

middleName: Private, defendant's middle name.

phoneNumber: Private, defendant's phone number.

dateOfBirth: Private, defendant's date of birth.

courtCaseNumber: Private, court case defendant is assigned to.

agencyNumber: Private, agency defendant is assigned to.

expungeDate: Private, date defendant was expunged from Teen Court program.

closeDate: Private, date defendant's trial was closed.

pID: Public, unique identifier for physical city, state, and zip.

pAddress: Public, address defendant lives at.

pCity: Public, city defendant lives in, brought in through pID.

pState: Public, state defendant lives in, brought in through pID.

pZip: Public, zip code defendant lives in, brought in through pID.

mID: Public, unique identifier for mailing city, state, and zip.

mAddress: Public, address defendant's mail goes to.

mCity: Public, city defendant's mail goes to, brought in through mID.

mState: Public, state defendant's mail goes to, brought in through mID.

mZip: Public, zip code defendant's mail goes to, brought in through mID.

schoolID: Public, unique identifier for defendant's school.

schoolContactName: Public, staff member to contact at school.

schoolContactPhone: Public, staff member's phone.

schoolGrade: Public, defendant's grade level.

height: Public, defendant's height.

weight: Public, defendant's weight.

eyecolor: Public, defendant's eye color.

haircolor: Public, defendant's hair color.

sex: Public, defendant's gender.

ethnicity: Public, defendant's ethnicity.

licenseNum: Public, defendant's driver's license number.

licenseState: Public, state that issued defendant's driver's license.

notes: Public, notes about the defendant.

intake: Public, date and time for intake interview.

reschedule: Public, date and time for rescheduled interview.

interviewer: Public, unique identifier for person interviewing defendant.

referred: Public, date and time defendant was referred to juvenile system.

dismissed; Public, date and time defendant was dismissed from Teen Court.

added: Public, date and time defendant was added to program.

Class Methods

__construct(): Public, create empty defendant object.

getFromID($id): Public, gets defendant information from a defendant id.

updateDefendant(): Public, adds the defendant if userid is 0, otherwise updates the defendant record.

updatePersonal(): Public, updates the defendant's personal information. This is done after initially adding a defendant to the database or when editing one.

updateIntake(): Public, updates the defendant's intake information. This is done after initially adding a defendant to the database.

getGuardianList(): Public, returns an array of guardianIDs for the defendant.

totalGuardians(): Public, returns a count of guardians for the defendant.

checkWorkshop(): Public, returns if the defendant is in a workshop and if they have completed it.

checkCourt(): Public, returns if the defendant has been assigned to a court and if they have completed it.

checkJury(): Public, returns if the defendant has been assigned to any courts as a jury member and hours assigned so far.

checkSentence(): Public, returns list of sentences that have been assigned to the defendant .

updateNotes(): Public, updates defendant notes.

setDefendantID($str): Public, sets defendantID.

setProgramID($str): Public, sets programID.

setFirstName($str): Public, sets firstName.

setLastName($str): Public, sets lastName.

setMiddleName($str): Public, sets middleName.

setPhoneNumber($str): Public, sets phoneNumber.

setDateOfBirth($str): Public, sets dateOfBirth.

setCourtCaseNumber($str): Public, sets courtCaseNumber.

setAgencyNumber($str): Public, sets agencyNumber.

getDefendantID(): Public, returns defendantID.

getProgramID(): Public, returns programID.

getFirstName(): Public, returns firstName.

getLastName(): Public, returns lastName.

getMiddleName(): Public, returns middleName.

getPhoneNumber(): Public, returns phoneNumber.

getDateOfBirth(): Public, returns dateOfBirth.

getCourtCaseNumber(): Public, returns courtCaseNumber.

getAgencyNumber(): Public, returns agencyNumber.

getExpungeDate(): Public, returns expungeDate or N/A if expungeDate is null.

getCloseDate(): Public, returns closeDate or N/A if closeDate is null.

IV.7 Guardian

Description: The guardian class is used to handle information about a defendant's parent or guardian. [RR]

Class Members

defendantID: Private, Unique identifier for the defendant.

guardianID: Private, Unique identifier for the guardian.

relation: Public, Guardian's relation to the defendant.

firstName: Public, Guardian's first name.

lastName: Public, Guardian's last name.

homePhone: Public, Guardian's home phone number.

workPhone: Public, Guardian's work phone number.

employer: Public, Guardian's employer.

email: Public, Guardian's email.

pAddress: Public, Guardian's physical address.

pID: Public, Unique identifier for guardian's physical city, state, and zip.

mAddress: Public, Guardian's mailing address.

mID: Public, Unique identifier for guardian's mailing city, state, and zip.

liveswith: Public, If the defendant lives with the guardian.

Class Methods

__construct($defendantID) Public, creates a guardian object with the given defendantID.

getFromID($id): Public, loads the guardian object with information based off the given ID.

updateGuardian(): Public, inserts or updates the guardian object into the database depending on if guardianID is set.

removeGuardian():Public, deletes guardian from database.

setGuardianID($str): Public, set guardianID.

getGuardianID():Public, return guardianID.

getDefendantID():Public, return defendantID.

IV.8 Location

Description: The Location class stores city, state, and zip information for use in other classes without needing to store the same information in different places. [RR]

Class Members

programID: Private, Unique identifier for the program.

locationID: Public, Unique identifier for the location.

city: Public, City name.

state: Public, State city is in.

zip: Public, City's zip code.

Class Methods

__construct($programID): Public, create an empty location object with the given programID.

getFromID($id): Public, load the location object with information based off the given ID.

findLocation($city, $state, $zip): Public, return location object based off programID, city, state, and zip.

addLocation($city, $state, $zip): Public, if the location city, state, and zip is not in the database, adds the location to the database. Otherwise retrieves the locationID.

IV.9 Class Program

Description: The Program class is used to set up and manage the individual Teen Court programs. [RR]

Class Members

programID: Private, Unique identifier for the program.

code: Private, Used to allow other users to register to the program.

name: Private, Program name.

phys_address: Public, Program's physical address.

phys_city: Public, Program's physical city.

phys_state: Public, Program's physical state.

phys_zip: Public, Program's physical zip code.

mail_address: Public, Program's mailing address.

mail_city: Public, Program's mailing city.

mail_state: Public, Program's mailing state.

mail_zip: Public, Program's mailing zip code.

phone: Public, Program's phone number.

expunge: Public, Program's expunge method.

timezoneID: Public, Program's timezone.

active: Public, If the program is active or not.

Class Methods

__construct(): Public, creates an empty program object.

programExists($code): Public, checks the database to see if a program exists.

getFromCode($code): Public, gets program information from an existing program code.

getFromID($id): Public, gets program information from id.

updateProgram(): Public, updates or adds the program depending on if programID is set.

fetchUserDropdown($userID): Public, returns dropdown options of users in the program. Will return with a user selected if userID matches.

fetchOfficerDropdown($officerID): Public, returns dropdown options of officers in the program. Will return with an officer selected if officerID matches.

addCommonLocation($location): Public, inserts a new common place if location is new and returns the commonplaceID, otherwise just returns the commonplaceID.

getCommonLocation($location): Public, returns the common place name.

addOfficer($firstname, $lastname, $idNumber, $phone): Public, inserts the officer into the database.

addStatute($programID, $code, $title, $description): Public, inserts the statute into the database.

getProgramPositions(): Public, returns key and ID of court positions.

addSentence($name, $description, $additional): Public, inserts the sentence into the database.

getProgramID(): Public, gets ProgramID.

getName(): Public, gets program name.

getCode(): Public, Gets program code.

getFullAddress(): Public, Gets the program's physical address.

setProgramID($str): Public, Sets ProgramID.

setName($str): Public, Sets program name.

setCode($str): Public, Sets program code.

IV.10 School

Description: The School class is used to manage information about the schools that defendants attend. [RR]

Class Members

programID: Private, unique identifier for the program.

schoolID: Public, unique identifier for the school.

name: Public, school's name.

address: Public, school's address.

city: Public, school's city.

state: Public, school's state.

zip: Public, school's zip code.

Class Methods

__construct($programID): Public, creates an empty school object with the given programID.

getFromID($id): Public, loads the school information that matches.

findSchool($name, $address, $city, $state, $zip): Public, return a school object that matches all fields.

addSchool($name, $address, $city, $state, $zip): Public, this function looks up a school based on name, address, city, state and zip code. if it is found, the id is returned. If not, it is added to the school table. This prevents any duplicate information in the database.

IV.11 Sentence

Description: The Sentence class is used to connect sentences a court gives to a defendant. [RR]

Class Members

defsentID: Private, unique identifier for the sentence that a defendant was given.

sentenceID: Private, unique identifier for the sentence that a program has.

defendantID: Private, unique identifier for the defendant.

name: Public, name of sentence.

description: Public, description of what the sentence will require to be completed.

additional: Public, optional extra field for sentence.

additionalValue: Public, value of additional field.

completeDate: Public, date for when sentence was completed.

Class Methods

__construct($defendantID): Public, create empty sentence object with the given ID.

addSentenceFunction($defendantID, $sentenceID): Private, assigns a sentence to a defendant.

addSentence($sentences): Public, add new sentence for defendant.

getFromID($defsentID): Public, gets sentence information.

removeSentence($defsentID): Public, removes sentence requirement.

updateSentence(): Public, updates sentence requirements

setDefendantID($var): Public, sets defendantID.

getDefendantID(): Public, returns defendantID.

getSentenceID(): Public, returns sentenceID.

IV.12 User

Description: The user class is used to add and edit users, verify and update user password, and ensure only one email is used within the system.

Class Members

userID: Private, unique identifier for the user.

programID: Private, unique identifier for the program.

typeID: Private, determines user access levels.

firstName: Private, user first name.

lastName: Private, user last name.

email: Private, user email.

password: Private, hashed and salted version of user password.

lastLogin: Private, last time user last logged into account.

timezoneID: Private, user local timezoneID.

timezone: Private, user local timezone as string.

active: Private, if user is active or ‘deleted’.

Class Methods

__construct(): Public, create empty user object.

getFromLogin($email, $password): Public, checks the email and password the user entered at login with record in the database. If the email address exists, the password is checked with the current hash. If the hash is the same, proceed with user access and login.

getFromID ($id): Public, gets user information from a user id.

checkHash($hash): Private, checks a current hash against the algorithm to verify integrity.

newHash(): Private, generate a new hash based on email and password.

updateUser(): Public, adds the user if userid is 0, otherwise updates the user record.

removeUser($id): Public, marks the user as deleted and inactive in the database, doesn't actually remove the user

emailExists($email): Public, checks the database to see if an email address exists for a user.

fetchPhoneNumbers(): Public, gets the users phone numbers.

addPhone($type, $number, $ext): Public, to add a phone number to the user.

removePhone($id): Public, to remove a phone number from the user.

addEvent($event, $id = NULL): Public, logs a user action.

fetchHistory($id): gets a list of user's actions.

getUserID(): Public, return userID.

getFirstName(): Public, return firstName.

getLastName(): Public, return lastName.

getName(): Public, returns firstName and lastName.

getProgramID(): Public, returns programID.

getType(): Public, returns typeID.

getTimezone(): Public, returns timezone.

getTimezoneID(): Public, returns timezoneID.

getLastLogin(): Public, returns lastLogin.

getEmail(): Public, returns email.

isActive(): Public, returns active.

setUserID($val): Public, sets userID.

setProgramID($val): Public, sets programID.

setType($val): Public, sets typeID.

setFirstName($val): Public, sets firstName.

setLastName($val): Public, sets lastName.

setEmail($val): Public, sets email.

setPassword($val): Public, sets password.

setTimezoneID($val): Public, sets timezoneID.

setActive($val): Public, sets active.

display(): Public, for testing, shows all fields for user.

IV.13 Volunteer

Description: The volunteer class is for handling volunteer information, positions, and hours assisting the court. [RR]

Class Members

volunteerID: Private, unique identifier for the volunteer.

programID: Private, unique identifier for the program.

firstName: Private, volunteer's first name.

lastName: Private, volunteer's last name.

phone: Private, volunteer's phone number.

email: Private, volunteer's email.

positions: Private, array of volunteer positions.

active: Private, whether volunteer is active or not.

Class Methods

__construct($programID): Public, creates empty volunteer object.

getVolunteer($id): Public, returns volunteer information based on id.

updateVolunteer(): Public, if volunteerID is empty, inserts volunteer into database. Otherwise updates volunteer information. After insert/update, clears all positions the volunteer holds in the database and inserts new positions.

clearPositions(): Public, deletes all positions that the volunteer is currently assigned.

deleteVolunteer(): Public, sets volunteer to inactive.

editVolunteerHours(): Public,

printVolunteerHours(): Public,

getVolunteerID(): Public, returns volunteerID.

getProgramID(): Public, returns programID.

getFirstName(): Public, returns firstName.

getLastName(): Public, returns lastName.

getPhone(): Public, returns phone.

getEmail(): Public, returns email.

getPositions(): Public, returns positions.

getActive(): Public, returns active.

setVolunteerID($val): Public, sets volunteerID.

setProgramID($val): Public, sets programID.

setFirstName($val): Public, sets firstName.

setLastname($val): Public, sets lastName.

setPhone($val): Public, sets phone.

setEmail($val): Public, sets email.

setPositions($val): Public, sets positions.

setActive($val): Public, sets active.

IV.14 Class Workshop Location

Description: The workshop location class handles information for where workshops will be held. [RR]

Class Members

workshopLocationID: Public, unique identifier for this court location

locationID: Public, unique identifier for location. Holds city, state, and zip fields.

programID: Public, unique identifier for the program.

name: Public, location name.

address: Public, location address.

city: Public, location city, brought in with locationID.

state: Public, location state, brought in with locationID.

zip: Public, location zip, brought in with locationID.

Class Methods

__construct(): Public, reates an empty workshop location object.

updateWorkshopLocation(): Public, adds a new workshop or edits an existing workshop depending on workshopLocationID.

getWorkshopLocation($workshopLocID): Public, loads workshop location object with data based on workshopLocID.

getWorkshopLocationID():Public, return locationID.

getProgramID():Public, returns programID.

setLocationID($val): Public, set locationID.

setProgramID($val): Public, set programID.

IV.15 Workshop

Description: The workshop class is used to create, modify and remove workshops and workshop participants. [RR]

Class Members

workshopID: Private, Unique identifier for the workshop.

programID: Private, Unique identifier for the program.

date: Private, date and time of the workshop.

title: Private, title of workshop.

description: Private, description of workshop.

instructor: Private, workshop's instructor.

officerID: Private, unique identifier for the officer if one is assisting the workshop.

workshopLocationID: Private, unique identifier for the workshop location.

Class Methods

__construct($user_programID): Public, creates a new workshop object.

updateWorkshop(): Public, inserts a new workshop into the database if workshopID is empty, otherwise updates the information for that workshop.

deleteWorkshop(): Public, deletes workshop roster and workshop.

getWorkshop($id): Public, returns workshop information retrieved based on id.

addWorkshopParticipant($workshopID, $defendantID): Public, adds defendant and program to roster.

removeWorkshopParticipant($workshopID, $defendantID): Public, deletes defendant from workshop.

completedWorkshopParticipant($workshopID, $defendantID): Public, marks defendant as having completed the workshop at current time.

listWorkshopParticipants($id): Public, returns list of workshop participants for the given workshop.

getWorkshopID(): Public, returns workshopID.

getProgramID(): Public, returns programID.

getDate(): Public, returns date.

getTitle(): Public, returns title.

getDescription(): Public, returns description.

getInstructor(): Public, returns instructor.

getOfficerID(): Public, returns officerID.

getworkshopLocationID(): Public, returns workshopLocationID.

setWorkshopID($val): Public, sets workshopID.

setProgramID($val): Public, sets programID.

setDate($val): Public, sets date.

setTitle($val): Public, sets title.

setDescription($val): Public, sets description.

setInstructor($val): Public, sets instructor.

setOfficerID($val): Public, sets officerID.

setworkshopLocationID($val): Public, sets workshopLocationID.