JumboCode: Jeanne Geiger Crisis Center and Domestic Violence High Risk Team

# Team

Lead: Barry Eom

Designer:

Engineers:

* Client Side (3, 4 people)
* Server Side (5 - 6 people)

# Client Background

Jeanne Geiger Crisis Center (JGCC) is a non-profit organization that works to empower domestic violence survivors and end domestic violence. The staff of JGCC provides advocacy, clinical, and legal services to survivors of domestic violence and their families. In 2005, after the homicide of one of JGCC’s client, Domestic Violence High Risk Team (DVHRT) was created to work with communities across the country to identify the most dangerous domestic violence cases.

DVHRT Model consists of four strategies with the goal of creating mechanisms for the transfer and identify of critical information in high risk cases, closing systemic gaps where homicides can occur. DVHRT works with sites for nine to twelve months to implement the model, providing assessment of their current response system and training to advocates, law enforcement, and prosecutors.

In the past, after communities working with DVHRT finish their training, they enter an evaluation phase, where JGCC collects data from sites to be able to assess how successful their homicide prevention efforts are. To do so, they have what they call *data repositories*, spreadsheets that allow them to collect more information on the cases they accept onto their high risk team. They also send JGCC their data each month of aggregate rundown.

DVHRT’s problem is they want to move from static, insecure spreadsheets to a more dynamic, collaborative, and secure database. Our goal is to empower DVHRT and the communities working with DVHRT to more effectively validate and successfully execute their homicide prevention efforts.

# Project Overview

We need to create an end-to-end website, a fully secure database that enables JGCC/DVHRT and the communities working with DVHRT to keep track of all their cases. Authentication is a must. JGCC will have admin access, enabling them to:

* Create and grant access to new implementation communities
* Run reports on each sites data based on dates entered
* Make customizations to certain data fields depending on the community when creating new grant access (referral source is a field that must be editable)
* Communities will be able to run reports on their own data

Each community, on the other hand, will have their own log-in and access to their own database. The database must capture all of the information in the format of DVHRT’s [data repository](#_Database_Schema). Some of the fields in the data repository are dropdown menus. The victim and abuser names, however, must be anonymized or not visible to anyone besides the communities themselves. More details can be found below.

# Objective and Key Results

The scope of this project with OKRs is outlined in milestones.

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Objective** | **Key Results** |
| 1. Technical Validation | Ensure that our technology is capable of supporting the critical features. | * We are able to work with authentication technology * We can create database creation for each account in a contained environment * We are able to validate that the database is secure * We are able to anonymize certain aspects of database (name of victim) * We are able to have customize capabilities for certain fields in the database |
| 1. Minimum Viable Product (v1) | Create an end-to-end system that have the following minimum functionalities:   * Functional backend * Admin and community accounts * Low fidelity user interface | * DVHRT can create a new account and database for a new community, customizing certain fields * DVHRT can look at database of a community * A community can securely log-in to their database * A community can view, create, edit, and delete cases |
| 1. V2 Product | Create a v2 product with features focused on automation | * DVHRT can run reports on each community’s database * Communities will be reminded to receive notifications to update certain fields (pretrial hearing field) * Communities and DVHRT can post on a message forum to cooperate more, share success stories, and access more resources * Communities can run their own monthly reports |

# Technology

**Client Side**

* React.js
  + Frontrunner for web app deployment, currently. Uses the idea of components well

**Server Side**

* Server running on Django
  + Written using Python, really good for quick web development, pairs well with databases
* Database: PostgreSQL
  + Supported by Heroku and works well with Django

**Hosting the Server**

* Heroku
  + Integrates well with Github. Any push to master branch on github automatically deploys that newest codebase

# User Flows

**Admin (DVHRT and JGCC) Flow**

|  |  |  |
| --- | --- | --- |
| **Story** | **User flow** | **Technology / Clarifications** |
| 1. Getting to the website | A user, Jeanne Geiger/DVHRT staff member, goes to the landing page, via a link. | * React.js for front end * Heroku |
| 1. Log-in | User is welcomed with a log-in information page. User logs in. | * [TBD] Log-in authentication tool |
| 1. Homepage | User is prompted with three choices on the home page. *View communities, Add community*, *edit* *community*, or *delete* community. |  |
| 1. Add landing page | User clicks on add. User is taken to the add page. |  |
| 1. Add Fields | User creates username and password. User also adds options for the field *referral source*. |  |
| 1. Add Submit | User submits with a confirmation screen then taken to the home page. | * Post to database |
| 1. Edit landing page | User clicks on edit. User is taken to the edit page. User is prompted to edit referrals. User goes back to the home page afterwards. | * Retrieve all from database |
| 1. View | User chooses to view. Browsing through the different communities, user selects a which communities to view. |  |
| 1. View search | User looks through the list of cases in a community. [User clicks on *Run Report.*] |  |
| 1. View filter | User also selects to filter by case and sees everything. | Should past versions be kept—i.e. how important is version control? |
| 1. Return after view | User is additionally prompted with option to *View Other Community* or *Return to Home*. User goes home. |  |
| 1. Delete landing page | User clicks on *delete a community.* User is taken to the delete landing page. | Question for JG: *distinguish between archive and delete*. *Deleting can be important for privacy reasons, but archiving can be good if they are not trying to delete data permanently.* |
| 1. Delete verification | User chooses to delete option. User is prompted with—*are you sure you want to delete?*—pop-up. User clicks yes. | Will we not immediately remove / delete cases? Will we ever permanently delete cases? |

|  |  |  |
| --- | --- | --- |
| **Story** | **User flow** | **Technology / Clarifications** |
| 1. Getting to the website | A staff member goes to the landing page, via a link. | * React.js for front end * Heroku |
| 1. Log-in | User is welcomed with a log-in information page. User logs in. | * [TBD] Log-in authentication tool |
| 1. Homepage | User is prompted with three choices on the home page. Add, edit, or delete. |  |
| 1. Add landing page | User clicks on add. User is taken to the add page. |  |
| 1. Add Fields | User is prompted to fill out all the information pertaining to a case. |  |
| 1. Add Submit | User submits. User is prompted with the option to *add another*, which returns to Add Fields page, or *go home*, which takes them to the homepage. User clicks on *go home*, taking them to the home page. | * Post to database |
| 1. Edit landing page | User clicks on edit. User is taken to the edit page. User sees a page with series of cases, with a search-bar at the top right. User has the option to filter or search for a case (by case ID or name). | * Retrieve all from database |
| 1. Edit – filter | User chooses to filter. User can filter by name, date, birthdate, or case ID. User sees the case the user likes. User clicks on the case. |  |
| 1. Edit – search | User searches for a case using the case ID. User finds the case. User selects to edit the case. |  |
| 1. Edit case | User sees that al the fields pertaining to the case is shown editable on a page. User edits some pages. | Should past versions be kept—i.e. how important is version control? |
| 1. Edit submit | User submits the case. User chooses to go back to the homepage. |  |
| 1. Delete landing page | User clicks on *delete a case.* User is taken to the delete landing page. Similar to the edit page, user can filter or search for a case to delete. User finds the case they want to remove. | Question for JG: *distinguish between archive and delete*. *Deleting can be important for privacy reasons, but archiving can be good if they are not trying to delete data permanently.* |
| 1. Delete verification | User chooses to delete option. User is prompted with—*are you sure you want to delete?*—pop-up. User clicks yes. | Will we not immediately remove / delete cases? Will we ever permanently delete cases? |

# Database Schema

Four Tables, all pointed to by a case ID

**Table *Victim\_Demographics***

|  |  |
| --- | --- |
| **Field Name** | **Type** |
| Case # | *int* |
| Victim Name (Last, First) | *string* |
| DoB | *date* |
| Gender | Female  Male  Other |
| Race/Ethnicity | American Indian/Alaska Native  Asian  Black/African American  Hispanic or Latino  Native Hawaiian/Pacific Islander  White  Other Unknown |
| Age @ Case Acceptance | 13 or younger  14-17  18-19  20-29  30-39  40-49  50-59  60+  Unknown |
| Primary Language | English  Spanish/Spanish Creole  Arabic  Cambodian/Khmer  Chinese  French/French Creole  German  Greek  Italian  Polish  Portuguese/Portuguese Creole  Russian  Vietnamese  Unknown |
| Town | *str* |
| Relationship Type | Current Spouse/Intimate Partner  Former Spoud/Intimate Partner  Current Dating Relationship  Former Dating Relationship  Other |
| Relationship Length | <1 year  1-5 years  6-9 years  10-14 years  15-19 years  20-29 years  30+ years |
| Minor Children in Home | Yes  no |
| Referral Source | Editable dropdown |

**Table *Abuser\_Demographics***

|  |  |
| --- | --- |
| **Field Name** | **Type** |
| Case # | *int* |
| Victim Name (Last, First) | *string* |
| DoB | *date* |
| Gender | Female  Male  Other |
| Race/Ethnicity | American Indian/Alaska Native  Asian  Black/African American  Hispanic or Latino  Native Hawaiian/Pacific Islander  White  Other Unknown |
| Age @ Case Acceptance | 13 or younger  14-17  18-19  20-29  30-39  40-49  50-59  60+  Unknown |
| Primary Language | English  Spanish/Spanish Creole  Arabic  Cambodian/Khmer  Chinese  French/French Creole  German  Greek  Italian  Polish  Portuguese/Portuguese Creole  Russian  Vietnamese  Unknown |
| Town | *str* |

**Table *Risk\_Factors***

|  |  |
| --- | --- |
| **Field Name** | **Type** |
| Case # | *int* |
| Has the physical violence increased in severity or frequency over the past year? | *bool* |
| Have you left him/her after living together in the past year? | *bool* |
| Does he/she control most or all of your daily activities? | *bool* |
| Has he/she tried to kill you? | *bool* |
| Has he/she ever threatened to kill you? | *bool* |
| Has he/she used a weapon against you or threatened you with a lethal weapon? | *bool* |
| Has he/she ever tried to choke (strangle) you? | *bool* |
| Has he/she choked (strangled) you multiple times? | *bool* |
| Do you believe he/she is capable of killing you? | *bool* |
| Does he/she own a gun? | *bool* |
| Has he/she ever threatened or tried to commit suicide? | *bool* |
| Is he/she unemployed? | *bool* |
| Has he/she avoided being arrested for domestic violence? | *bool* |
| Do you have a child that is not his/hers? | *bool* |
| Does he/she use illegal drugs? | *bool* |
| Is he/she an alcoholic or problem drinker? | *bool* |
| Has he/she ever forced you to have sex when you did not wish to do so? | *bool* |
| Is he/she violently or constantly jealous? | *Bool* |
| Has he/she ever beaten you while you were pregnant? | *Bool* |
| Threatens to harm victim’s children? | *Bool* |
| Does he/she spy on you, leaving threatening notes or messages, destroy your property or call you hwne you don’t want him/her to? | *Bool* |

**Table *Outcomes***

|  |  |
| --- | --- |
| **Field Name** | **Type** |
| Case # | *int* |
| Connection to Domestic Violence Services | *bool* |
| Engagement in ongoing domestic violence services | *bool* |
| Charges Filed At or After Case Acceptance | Police Response: Charges filed  Police Response: No Charges Filed  No Police Response: Not Applicable |
| Pretrial hearing outcome | Released on Bail  Released on Personal Recognizance  Detained/Pretrial Detention Statute  Detained/Bail Unmet  Detained/Other  Pending Pretrial Hearing |
| Sentencing Outcomes: Disposition | Charges Dismissed  Not Guilty  Deferred Adjudication  Plead/Found Guilty  Pending Disposition |
| Sentencing Outcomes: Sentence | Incarceration  Probation  Incarceration Followed by Probation |

# Project Deadlines

* September 6th: v0 spec
* Mid September: finalize spec
* End of September: team members finalized
* End of November: client testing with low fidelity user prototype for UI
* End of December: implementation of backend complete
* February: connect frontend and backend 🡪 launch minimum viable prototype
* March:

# Resources

**Version Control: Github/Git**

* Cloning a github repository:
  + `git clone https://github.com/JumboCode/TEN.git`
  + <https://help.github.com/articles/cloning-a-repository/>
* Branches - creating and managing:
  + `git checkout -b <BRANCH\_NAME>
  + <https://github.com/Kunena/Kunena-Forum/wiki/Create-a-new-branch-with-git-and-manage-branches>

**Django Documentation**

* Introduction to the framework: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>
* General website: <https://www.djangoproject.com/>
* Abstract out settings.py file to a folder: <https://simpleisbetterthancomplex.com/tips/2017/07/03/django-tip-20-working-with-multiple-settings-modules.html>
  + TODO: make sure python manage settings.py works

**PostgreSQL Documentation**

* New to PostgreSQL: <https://www.postgresql.org/about/>
* Connecting heroku and PostgreSQL: <https://devcenter.heroku.com/articles/heroku-postgresql#understanding-heroku-postgres-plans>
* Pairing up with Django: <https://docs.djangoproject.com/en/2.1/ref/databases/>

**Heroku - Server**

* <https://devcenter.heroku.com/articles/github-integration>
* Hosting on server: <https://devcenter.heroku.com/articles/heroku-postgres-plans>
* Server: <https://dashboard.heroku.com/apps/textbook-exchange-network/deploy/github>
* Database plans for $$: <https://devcenter.heroku.com/articles/heroku-postgres-plans>

**React Documentation**

* Intro to React documentation: https://reactjs.org/tutorial/tutorial.html#lifting-state-up
* Documentation on how React works! Make your own **tic tac toe game**: https://reactjs.org/tutorial/tutorial.html#lifting-state-up
* Thorough rundown of how to make your own React boilerplate (explains basic setup to get a base website/server running): <https://medium.freecodecamp.org/how-to-build-your-own-react-boilerplate-2f8cbbeb9b3f>
* Redux: <https://redux.js.org/>
  + Why use Redux: <https://www.fullstackreact.com/articles/redux-with-mark-erikson/>

**Javascript Documentation**

* Hello, World example: <https://javascript.info/hello-world>
* HTML/CSS Tutorials: <https://www.w3schools.com/html/>