Risk Assessment v4.0

by

Paul Wilson

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

Paul Wilson — April 9, 2021
Paul Wilson — Date

Yahya Gilany — April 9, 2021
Yahya Gilany, Faculty Advisor — Date

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

April 2021

Prepared by:

Paul Wilson

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

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

Prisoner recidivism is a growing problem in the United States. Offenders often find themselves rearrested and returned to the system after a few short years. The IT Solution Center's Risk Assessment System was developed as a web-based solution to risk assessment and offers a variety of tools to better manage offenders, conduct risk assessments, and facilitate the reduction of offender recidivism. Since that original development process, more states have adopted use of the Risk Assessment System, and that original mold doesn't always fit the needs of the new users. The Risk Assessment System v4.0 includes a complete front-end framework migration and improves upon the original v3.0 approach by adding some highly requested features. Notable highlights are enhancements to data reporting via the Research Portal, merging/unmerging of offender records, role/permission management, and a chronological offender timeline. This project also focused on creating a more robust database infrastructure through the use of logical replication to an external-server replica. Lastly, a .csv file upload application was developed to take the place of API integrations in the case that clients are unable to build a data upload solution of their own. These features were completed and iterated with a heavy focus on user satisfaction and configurability. Bulletproof functionality, ease of use, and user interface simplicity were all driving factors in achieving positive user acceptance results.

I. INTRODUCTION

i. Problem:

In 2018, the Bureau of Justice Statistics in the U.S. Department of Justice released a special report on prisoner recidivism stating that 83% of all released prisoners in their sample had been rearrested sometime within the study's 9 year follow-up period. (Alper, 2018). "The Ohio Risk Assessment System (ORAS) was developed as a statewide system to assess the risk and needs of Ohio offenders in order to improve consistency and facilitate communication across criminal justice agencies" (Latessa, 2010) as well as "to efficiently allocate supervision resources and structure decision-making in a manner that reduces the likelihood of recidivism." (Latessa, 2010). In the beginning, the assessment tools that enabled this process were completed using paper-based forms. The web-based Risk Assessment System was developed as a more cost-effective, partially-automated alternative to previous collection studies with the added benefit of storing all data in a single database while simultaneously creating convenience for users through the use of web technologies.

The Risk Assessment System v3.0 was released on June 6, 2017. Since then, three years have passed and well over two thousand commits have been published to maintain the version 3 repository. In those three years, technology has taken massive strides forward while large parts of the Risk Assessment System have remained static and, in some cases, forgotten. In short, the Risk Assessment System needs an update.

Most notably, the business logic that the system currently follows was initially developed with only a few, specific clients in mind. With the system having far surpassed those original few, configurability has become a driving force for customers with each having their own specific needs and wants. Users would like a system that appears to be personalized directly for them, even if in reality they know it's multi-tenant from both a business and technical perspective.

In addition to those concerns, many agencies have encountered a problem that occurs when they use a different case management system but would still like to conduct UCCI risk assessments through the Risk Assessment System. This is why the ITSC hosts an external API with documented endpoints, so that users can upload their demographic and user data directly through the same endpoints the front-end application uses. Unfortunately, this process doesn't always proceed as smoothly as would be preferred. The ITSC has stated that it would be convenient for users to have a way to upload .csv data directly into the system by way of a new, external application.

Once that data has been uploaded, users have also shown interest in having a way to analyze it. While they all have their own solutions, the ITSC believes there is an opportunity here for building an in-house solution for visually charting tabular data and allowing that data to be downloaded. This is a combination of a slight pivot from an external solution the ITSC already uses to an internal solution that's more manageable and an entirely new spin on reporting and data analysis for the Risk Assessment System as a whole.

Finally, there have been load and server concerns in recent months, primarily as a result of having a single, monolithic database. There are limitations on how many connections a single database can allow, and those connections are often being used up during high-traffic periods of the day. As the total amount of users continues to expand far past what could have been originally anticipated, the current Risk Assessment infrastructure has begun to struggle in handling such a large amount of users effectively on a day-to-day basis.

ii. Solution:

Risk Assessment v4.0 will target key aspects of the Risk Assessment System that are the most in need of updating. Based on user feedback, Risk Assessment v4.0 will aim to improve on these features and modernize them. The entire technical stack will be evaluated during this process. That includes the front-end layer that serves components to and handles requests from the browser, the API back-end layer that handles requests from externally integrated applications or developer scripts, the back-end microservices which are individualized applications each having their own specific jobs and that handle requests from the API, the database which is where the data is stored, modified, and fetched via queries from the microservices, and the servers themselves–if any configuration changes are deemed necessary during the evaluation process.

The Risk Assessment System is integral to the judicial process in several states. It allows officers to find historical offender data all in one place, automates the completion

and scoring of the risk assessments themselves, and automatically generates caseplans with a focus on the offender's specific needs highlighted by those very same risk assessments. The Risk Assessment API also allows for 3rd-party integration so that clients can build their own tailored interface. Most other case management systems will have one or two of these features but won't offer the complete package. Instead of being yet another simple case management system, the Risk Assessment System is based on time-tested strategies and scholastically-validated research performed by the University of Cincinnati's Corrections Institute.

The sponsor for this project, the IT Solutions Center (ITSC), developed the application originally and continues to support the current in-production version. The ITSC's ties to the University of Cincinnati's School of Information Technology and the fact that the application is developed and monitored primarily by student workers makes the Risk Assessment System an extremely cost-effective option. It's worth mentioning that many 3rd-party integration attempts either fall flat because of funding or simply have issues with the developers they've hired and end up turning to the skilled, flexible student workers at the ITSC.

The Risk Assessment v4.0 will serve to further enhance the entire risk assessment and case management process. Over the years, the ITSC has collected feedback from users and identified core places for improvement through continuous discussion with state administrators and the criminal justice professionals at the UCCI. The features identified for updating by the ITSC include roles/permissions, reports, the Offender

Facesheet, and the Offender Merge feature. A new, lean file upload application will be developed for users wanting an alternative to building their own API integrations. This application will allow users to upload .csv data directly into the Risk Assessment System without having to hire developers to build complex integration scripts of their own. To help ORAS handle the aforementioned load increase, Risk Assessment v4.0 will also focus on bolstering the database system by implementing database replication concepts.

iii. Project Goals & Methodology:

The following section will briefly outline the project goals and methodology used in developing the project. Project goals will consist of a very high-level overview of the project solution, separated into digestible sections. Methodology will contain an explanation of the weekly development process.

Goals:

Several key features of the application require updates including roles/permissions, reports, the Offender Facesheet, and the Offender Merge feature:

- Permissions will be updated to create a more human-readable interface that users can easily use to update user account roles.
- Reports will be moved from the current external solution to something internal.
- Users have requested the addition of an "Offender Timeline" to the Offender Facesheet.

- The Offender Merge UI will be updated for clarity, and the feature itself will allow for the semi-bulletproof merging and unmerging of offenders.

A Risk Assessment file upload application will be developed to provide users with a way to upload data without having to hire developers to build an API integration.

The Risk Assessment System will be transformed into a high-availability environment through the use of logical database replication.

Methodology:

This project will be developed using the Agile methodology. There will be weekly sprint meetings and daily standups with the Risk Assessment development team. Each requirement will be developed, presented to the sponsor, and tested internally before being released to the production environment.

iv. Overview:

The remainder of this final report outlines in detail how the project was completed. The report includes the following sections: project concept, design objectives, methodology & technical approach, user profile, use case diagram, technical architecture, testing, budget, project timeline, problems encountered & analysis of problems solved, and future recommendations.

II. DISCUSSION

i. Project Concept:

In 2018, the Bureau of Justice Statistics in the U.S. Department of Justice released a special report on prisoner recidivism stating that 83% of all released prisoners in their sample had been rearrested sometime within the study's 9 year follow-up period. (Alper, 2018). The Risk Assessment System was developed as a web-based solution to offender risk assessment and offers a variety of tools allowing officers to better manage offenders, conduct assessments, and facilitate the reduction of recidivism for offenders nationwide. The Risk Assessment v4.0 will serve as an update to that original idea.

This project was conceived and developed based on research from the UC Corrections Institute. The IT Solutions Center at the University of Cincinnati currently handles all the development work on the Risk Assessment System. As an ITSC software developer currently working on the project, the opportunity to work on the application in a much larger capacity was offered through senior design. The ITSC laid out their expectations for this process, and the team conducted the requirements analysis from there.

ii. Design Objectives:

Objectives:

The team developed functional enhancements for the Risk Assessment System in addition to two entirely new, external applications. These enhancements aimed at

updating the current version of the system while simultaneously adding brand new features for users to enjoy. The full extent of the objectives are referenced below (see Table 1).

Offender Timeline	• Outlines key events within an offender record chronologically • Located on the Offender Facesheet
Internal Reporting Service	• Replaced use of the JasperReports server with an in-house Reporting Service • Formatted as a paginated list with options for export to .csv and .xlsx as well as printing
Role Permissions	• Implemented "related" and "unrelated" permissions • Created a user-friendly action/subject paradigm • Enhanced the front-end permissions UI to make it easier for users to modify role permissions • Implemented an implicit/explicit permissions model
Merge/Unmerge Offender Feature	• Allow for duplicate offenders to be merged together into one record • Allow for mistakenly merged offenders to be unmerged into separate records • Create a front-end merge UI where users can: ○ View historical merges

	○ Monitor merge progress for each subset of offender data ▪ Assessments, caseplans, programs, offenses
Creatable/Customizable Forms	• Highly configurable form UI that allows admin users to create forms for use throughout their agency • Allows for flexibility and scalability of the system with regard to new agencies and the specific data they have available
Database Replication	• Develop & implement logical replication for the ORAS production database • Develop & implement a strategy to facilitate load balancing and automated failover
RA-Mobile	• A mobile application through which users can login, access their caseload, view offenders, and enter assessments in the following ways: ○ Through a functional mobile form ○ Using OCR to translate camera images into assessment data to be stored in the Risk Assessment database

Uploader	• An external application that should provide an alternative to developing scripts / interfaces with the Risk Assessment API o Users can upload data without the technical complexity of working with a development team and managing bugs on both sides

Research Portal	• An external application that should eventually replace use of the JasperReports server
	  o Introduced interactive, helpful visuals of the report content
	  o Data table formatted as a paginated list with options for export to .csv

Table 1 Project Objectives

Table 1: Project Objectives

Stretch Goals:

- Finalize the front-end migration from Angular.js to React.js
- Update the Risk Assessment's push notification feature
- Use database replication to "refresh" the dev and staging databases monthly
- Setup a pre-production environment for trainings and constrain testing to a single, development environment

- Implement a "User merge" feature that allows users to merge mistakenly duplicated user accounts

Abandoned Goals:

- The "Internal Reporting Service" turned into what is now call the "Research Portal"
- "RA Mobile" and the "creatable / customizable forms" were abandoned in favor of the "Uploader" application which the ITSC saw as a more pressing concern

iii. Methodology & Technical Approach:

The following section will briefly outline the technical methodology and approach used in developing the project. Design Requirements will consist of a very high-level overview of the project requirements analysis. Procedures will contain an explanation of the iterative development process. Technologies Used will list and briefly define the technologies utilized in this project.

Design Requirements:

Having well described design requirements helped immensely with contract estimates. The fact that this is an existing project and that the team had prior experience with the system allowed more exact requirement descriptions. This led to meeting both

the initial timeline constraints as well as fostering a trusting relationship between the team, the sponsor, and the clients.

Procedures:

Developing vertically, focusing on one feature at a time, has allowed the team to follow the Agile iterative process for each piece of this project. The technical approach to each iteration consists of developing locally, testing locally, sending the newly written code in for peer review, and then eventually having Quality Assurance test the changes on not one but two testing environments before User Acceptance testing was conducted.

iv. User Profile:

The User Profile has been separated into two sections. The first are general, high-level insights that didn't fit a specific user profile but that were notable enough to necessitate being mentioned. The second section is comprised of individual tables for each user type (see Figures 1.1-1.4).

General Insights

- Technological Expertise
    - The primary expertise of the users is not generally associated with technical savvy. In fact, it seems that the average user of the Risk Assessment System will need a highly intuitive, extremely straightforward interface with helpful hints throughout.

- Variability of Agency Needs & Wants
    - The capability, needs, and wants of users may vary by agency and/or subagency in many different configurations that cannot be predicted.
    - In some cases, entire features are agency specific.

User (Figure 1.1)
Understanding of Agency Processes: Low
Technological Expertise: Low
Software and Interface Experience: - User should have basic experience with using a web browser and navigating through a site.
Experience with Similar Applications: - Google Chrome, Firefox, Survey or Form-based Web Applications, and RA version 3
Task Experience: - Basic web application interactions such as using the mouse to click specific areas, using the keyboard for text entry, answering questions in a form, navigating using a basic navbar, moving through paginated lists, and searching using filters
Frequency of Use: - This is highly dependent on the user's needs. While it's most likely to be daily, it could also be weekly or monthly in some cases
Key Interface Design Requirements that the Profile Suggests: - Secure & highly available - Easily accessible & easily navigated - Highly intuitive with a straightforward design

Figure 1.1 User Profile

Supervisor (Figure 1.2)
Understanding of Agency Processes: Moderate
Technological Expertise: Low
Software and Interface Experience: - User should have basic experience with using a web browser and navigating through a site.
Experience with Similar Applications: - Google Chrome, Firefox, Survey or Form-based Web Applications, and RA version 3
Task Experience: - Basic web application interactions such as using the mouse to click specific areas, using the keyboard for text entry, using the overview to review assessments, navigating using a basic navbar, moving through paginated lists, searching using filters, and viewing reports.
Frequency of Use: - This is highly dependent on the supervisor's needs. While it's most likely to be daily, it could also be weekly or monthly in some cases
Key Interface Design Requirements that the Profile Suggests: - Secure & highly available - Easily accessible & easily navigated - Highly intuitive with a straightforward design

Figure 2.2 Supervisor Profile

Administrator (Figure 1.3)
Understanding of Agency Processes: High
Technological Expertise: Low
Software and Interface Experience: - User should have basic experience with using a web browser and navigating through a site.
Experience with Similar Applications: - Google Chrome, Firefox, Survey or Form-based Web Applications, and RA version 3
Task Experience: - Basic web application interactions such as using the mouse to click specific areas, using the keyboard for text entry, managing agency-wide configurations, navigating using a basic navbar, moving through paginated lists, and searching using filters.
Frequency of Use: - This is highly dependent on the administrator's needs. This is likely to be less frequent, weekly or even monthly depending on the admin.
Key Interface Design Requirements that the Profile Suggests: - Secure & highly available - Easily accessible & easily navigated - Highly intuitive with a straightforward design

Figure 3.3 Administrator Profile

Developer (Figure 1.4)
Understanding of Agency Processes: Extremely Low
Technological Expertise: High
Software and Interface Experience: - User should have advanced experience with software integration through an API including interpreting API documentation and sending/receiving JSON data.
Experience with Similar Applications: - Google Chrome, Firefox, Postman (or any other API development platform)
Task Experience: - Basic web application integration interactions including following guides to send and receive JSON data - Basic documentation interpretation, reading through necessary endpoints and using that to develop requests to be used in the integration
Frequency of Use: - Daily during the development process. This will change once the integration goes live, likely being monthly or weekly depending on when the integration needs to update for deprecated or modified endpoints
Key Interface Design Requirements that the Profile Suggests: - Secure & highly available - Well documented - Flexible so that integrations aren't crippled by new releases and can be warned in the case that an endpoint is to be deprecated or modified

Figure 4.4 Developer Profile

Figure 1: User Profiles

v. Use Case Diagram

The Use Case Diagram below (see Figure 2) is representative of the different user roles and their unique permission paradigms. It shows both the interaction from each type of user as well as the daily responsibilities of the full-stack developer.

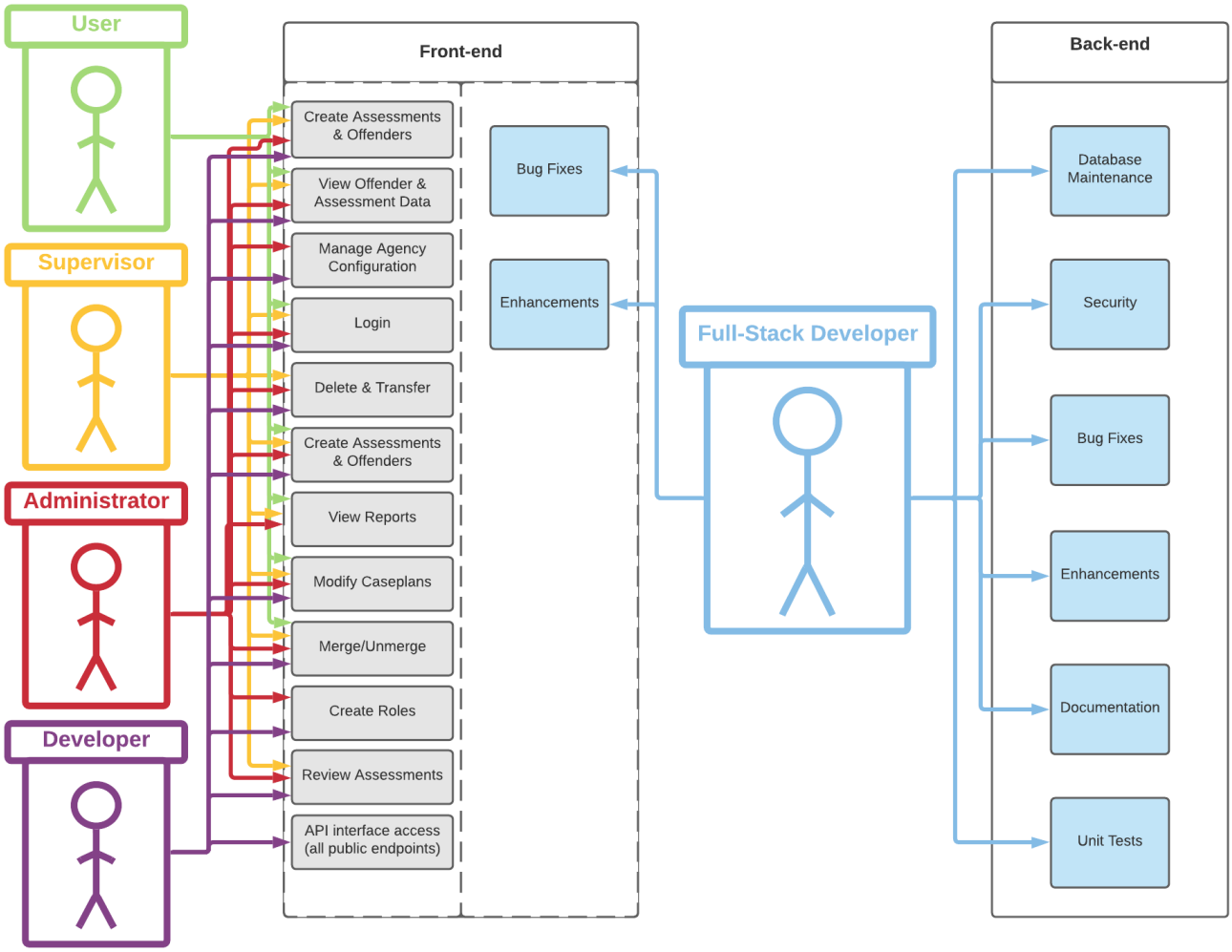

Figure 2 Use Case Diagram

Figure 2: Use Case Diagram

vi. Technical Architecture

Architecture Diagram:

The Architecture Diagram below (see Figure 3) is representative of the entire technical infrastructure of the Risk Assessment System and the new, external applications developed during this project. Please note that there are 19 total microservices in the Risk Assessment ecosystem instead of the 4 symbolically represented in the diagram here.

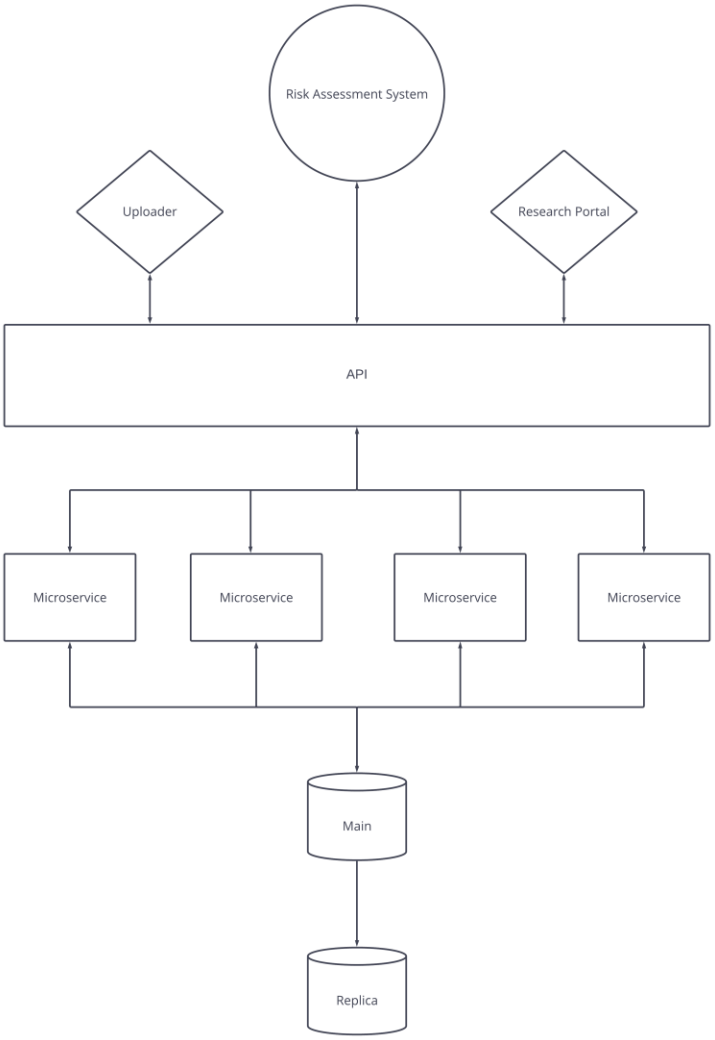

Figure 3 Architecture Diagram

Figure 3: Architecture Diagram

Technologies Used:

The technologies listed below (see Table 3) were used in some part of the process whether that be the front-end, back-end, database, or external applications. The table below is separated into four major sections for each feature category and each technology has its own section with a brief description and explanation of how it has been used in the project.

ORAS Updates and Features (Merge, Permissions, Reports, Forms, Timeline)	React.js	• Front-end JavaScript framework ideal for building interactive web applications and responsive UI's • The Risk Assessment System has been migrated to use React.js • All front-end UI updates will have to be made using React.js
	Node.js	• JavaScript runtime environment • Node.js is used in each layer of the Risk Assessment architecture for conducting server-side operations
	Express	• Node.js web application framework • This is used on the Risk Assessment front-end layer to

		build an API for use by the front-end components
	Restify	• Node.js web service framework optimized for building RESTful APIs • This is used extensively in the Risk Assessment API and Microservice layers
	PostgreSQL	• Open-source relational database • This is the database system that is used by the Risk Assessment System
	Docker	• Docker allows for containerizing apps so that they can be easily run, restarted, and/or reconfigured in a very quick, efficient way • Docker is used extensively throughout the Risk Assessment System. Most notably, the PostgreSQL database instance is spun up as a docker container on the database server

Research Portal	dc.js	• Library for generating interactive, dimensional charts and data tables
	Node.js	• Allows for building mobile applications using HTML, CSS, and JavaScript to target multiple platforms with one code base
	Restify	• Node.js web service framework optimized for building RESTful APIs • The Research Portal uses Restify as its back-end server framework

	Typescript	• Open-source language that builds on JavaScript by adding static type definitions
		• This is especially helpful for two things:
		  ○ Reducing time spent fixing bugs
		  ○ Increasing speed and efficiency of writing code

Uploader	Socketcluster.io	• Library for connecting, subscribing, and receiving WebSocket push notifications • This is used to allow real-time updates to the client front-end without forcing them to wait for async responses from the server
	Node.js	• Allows for building mobile applications using HTML, CSS, and JavaScript to target multiple platforms with one code base
	Restify	• Node.js web service framework optimized for building RESTful APIs • The Uploader uses Restify as its back-end server framework
	Typescript	• Open-source language that builds on JavaScript by adding static type definitions • This is especially helpful for two things: ○ Reducing time spent fixing bugs • Increasing speed and efficiency of writing code

Database Replication	PostgreSQL	• Open-source relational database • This is the database system that is used by the Risk Assessment System
	pglogical	• A logical replication system implemented as a PostgreSQL extension • I plan to use pglogical to facilitate logical replication between the database servers

Table 2 Technologies Used

Table 2: Technologies Used

vii. Testing

The following section will briefly outline the testing methodology and approach used in testing the project. Test Plan will consist of a very high-level overview of the project testing procedures. Test Results will contain an explanation of the outcomes of the tests described in the Test Plan.

Methodology & Approach:

Each project was subjected to rigorous manual-testing to make sure that each facet of the product worked from a user standpoint while hosted on a local machine. That was followed by internal acceptance testing with the project sponsors. After that, the ITSC's Quality Assurance team also verified that these features were working as intended on two separate testing

environments. Once passing Quality Assurance on those two environments, the features/enhancements were sent to users for User Acceptance Testing.

Test Plan:

Scope of Testing

- Roles & Permissions
  - Role Templates
    - Create
    - Retrieve
    - Update
    - Delete
  - Role
    - Create
    - Retrieve
    - Update
    - Delete
- Offender Timeline
  - View with varying number of events
- Offender Merge/Unmerge
  - Search functionality
    - Merge
    - Unmerge
  - Merge
    - “Normal” circumstances
    - Additional offenders (3, 4, 5)
  - Unmerge
    - “Normal” circumstances
    - After adding additional data to the merged record
- Research Portal
  - View reports
    - Interactive graphs
    - Filter by clicking
    - Filter by modifying date interval
    - Download .csv
- Uploader
  - Upload file
  - Validation of file
    - Columns
    - Data
  - Update jobs
    - Check for a few offenders

Objectives

a) All major features and use cases of those features need to be accounted for
b) All use cases must account for all user roles
c) All use cases must account for all agencies within all hierarchies
d) Testing must be conducted locally by the developer before being deployed to the testing servers
e) Testing must also be conducted remotely by the ITSC's Quality Assurance team before being moved further along the development pipeline
f) Final testing must be done by a select group of Risk Assessment users before the product is shipped live to the production environment
g) All bugs need to be resolved *before* the IT Expo

Test Results:

Test Logs and Procedures

- Role & Permissions

Role Templates

Record #	Test case #	Type	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	Create	New role template	New role template	P	-	11/1/20
2	1B	Retrieve	2 role templates	2 role templates	P	-	11/1/20
3	1C	Update	Update permissions	Update permissions	P	-	11/1/20
4	1D	Delete	Role template deleted	Role template deleted	P	-	11/1/20

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date
dev	Rishabh	all	P	-	11/6/20
staging	Rishabh	all	P	-	11/8/20

Table 3 Role Templates Test Logs

Role

Record #	Test case #	Type	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	Create	New role template	New role template	P	-	11/1/20
2	1B	Retrieve	2 role templates	2 role templates	P	-	11/1/20
3	1C	Update	Update permissions	Update permissions	P	-	11/1/20
4	1D	Delete	Role template deleted	Role template deleted	P	-	11/1/20

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date
dev	Rishabh	all	P	-	11/6/20
staging	Rishabh	all	P	-	11/8/20

Table 4 Role Test Logs

- Offender Timeline

Record #	Test case #	Type	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	View	Correctly styled timeline	Correctly styled timeline	P	-	10/4/20
2	1B	View	Correctly styled timeline	Correctly styled timeline	P	-	10/4/20
3	1C	View	Correctly styled timeline	Correctly styled timeline	P	-	10/4/20

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date
dev	Rishabh	View	P	-	10/10/20
staging	Rishabh	View	P	-	10/12/20

Table 5 Offender Timeline Test Logs

- Offender Merge/Unmerge

Search functionality

Record #	Test case #	Type	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	Search Merge	Test Offender	Correctly styled timeline	P	-	12/6/20
2	1B	Search Unmerge	Correctly styled timeline	Correctly styled timeline	P	-	12/6/20

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date
dev	Rishabh	Search	P	-	12/11/20
staging	Rishabh	Search	P	-	12/12/20

Table 6 Merge/Unmerge Search Test Logs

Merge/Unmerge

Record #	Test case #	Type	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	Merge	Test Offender	Test Offender	P	-	12/6/20
2	1B	Unmerge	Test Offender & Test2 Offender2	Test Offender & Test2 Offender2	P	-	12/6/20

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date
dev	Rishabh	Merge/ Unmerge	P	-	12/11/20
staging	Rishabh	Merge/ Unmerge	P	-	12/12/20

Table 7 Merge/Unmerge Test Logs

- Research Portal

View Reports

Record #	Test case #	Test	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	Click on each interactive graph	Each graph filters	Each graph but one filters	F	Score graph doesn't filter	3/10/21
2	1B	Filter by modifying date	Graphs should filter	Graphs should filter	P	-	3/10/21
3	1C	Download .csv	Data should download as .csv	Data should download as .csv	P	-	3/10/21

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date
dev	Ryan	all	Pending	-	-
staging	Ryan	all	Pending	-	-

Table 8 Research Portal Test Logs

- Uploader

Upload File

Record #	Test case #	Test	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	Upload file with correct columns	Successfully mapped columns	Each graph but one filters	P	-	3/14/21
2	1B	Upload wrong file type	Error. – wrong file type	Error. – wrong file type	P	-	3/14/21
3	1C	Upload file with wrong columns	Column alerts populate red with errors	Column alerts populate red with errors	P	-	3/14/21

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date

dev	Ryan	all	Pending	-	-
staging	Ryan	all	Pending	-	-

Table 9 Upload File Test Logs

Validation Job

Record #	Test case #	Test	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	Validate file with errors	Errors visible, no run job option	Errors visible, no run job option	P	-	3/14/21
2	1B	Validate file with no errors	Progress 100%, run job option	Progress 100%, run job option	P	-	3/14/21
3	1C	Reupload	Data should download as .csv	Data should download as .csv	P	-	3/14/21

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date
dev	Ryan	all	Pending	-	-
staging	Ryan	all	Pending	-	-

Table 10 Validation Job Test Logs

Update Job

Record #	Test case #	Test	Expected output	Actual output	Pass/ Fail	Reason for failure/success	Date
1	1A	Run update job	Offender correctly updated	Offender correctly updated	P	-	3/14/21

Environment	tester	Type	Pass/ Fail	Reason for failure/success	Date
dev	Ryan	all	Pending	-	-
staging	Ryan	all	Pending	-	-

Table 11 Update Job Test Logs

Review

The smallest bug can destroy a simple use case. Testing is important because it catches the things that cannot possibly or reasonably be anticipated. Often, there may not have been a bug, per se, but the perspective of the user isn't always clear when developing the feature. In some cases, the inconvenience of having to hit several buttons or the confusion that might occur due to a complicated user interface doesn't become obvious until QA or Client testing. Going through the manual tests personally helped with the understanding of this part in the development process.

No major reworks have been required, but that's mostly because there was constant self-testing and internal feedback from both the ITSC Quality Assurance team and the ITSC sponsors as each feature/enhancement moved through the project lifecycle. The main issues encountered were with the new Roles & Permissions enhancement. The Operations Lead noted that—even though the current workflow *was* improved—that workflow was almost never used. Because users don't really understand how the Roles & Permissions work, a lot of that work falls to the Operations team. They mentioned that a more simplistic, agency-wide approach might be a good addend to what had already been improved.

In the future, the team should focus on implementing unit and integration testing on a large scale. It would make deployments much less terrifying and would likely speed up the testing process overall. Manual testing is great and works fairly well when managed correctly, but automated tests are more consistent as long as they're written well.

viii. Budget

The two Risk Assessment budgets below (see Table 12 and Table 13) are separated into original and final budgets, respectively. Each is broken down into three sub-categories including software, hardware, and labor. Much of the software is open-source, all of the hardware was supplied by ITSC, and the only additional cost is team salary.

Risk Assessment v4.0 Original Budget			
Resource	Unit, Hours	Cost	Total
SOFTWARE			
Docker	1	$0	$0
JasperReports	1	$0	$0
pm2	1	$0	$0
drone	1	$0	$0
nginx	1	$0	$0
PostgreSQL	1	$0	$0
Lucidchart	1	$9	$9
Postico	1	$40	$40
Subtotal			$49
HARDWARE			
Mac Mini 8,1 Pro	1	$0	$0
Monitors	2	$0	$0
Wireless Mouse	1	$0	$0
Keyboard	1	$0	$0
Subtotal			$0
LABOR			
Salary	640	$15,360	$15,360
Assistance from ITSC personnel	minimal	$0	$0
Subtotal			$15,360
Total			**$15,458.00**

Table 12 Original Project Budget

Table 12: Original Project Budget

Risk Assessment v4.0 Final Budget			
Resource	Unit, Hours	Cost	Total
SOFTWARE			
Docker	1	$0	$0
JasperReports	1	$0	$0
pm2	1	$0	$0
drone	1	$0	$0
nginx	1	$0	$0
PostgreSQL	1	$0	$0
Lucidchart	1	$9	$9
Postico	1	$40	$40
Subtotal			$49
HARDWARE			
Mac Mini 8,1 Pro	1	$0	$0
Monitors	2	$0	$0
Wireless Mouse	1	$0	$0
Keyboard	1	$0	$0
Subtotal			$0
LABOR			
Salary	1280	$30,720	$30,720
Assistance from ITSC personnel	minimal	$0	$0
Subtotal			$30,720
Total			**$30,818.00**

Table 13 Final Project Budget

Table 13: Final Project Budget

ix. Project Timeline

The Project Timeline below (see Table 14) is split into two sections: Project Management Deliverables and Technical Features. Each section has a total day count in red and a list of individual tasks with a day count for each and how those days lineup on the calendar.

1.0 Project Management Deliverables: 120 days			
Task	Days	Start Date	End Date
1.1 Team Members and Project Name	1	8/24/2020	8/24/2020
1.2 First Draft of Team Contract	7	8/24/2020	8/31/2020
1.3 Project Abstract for Tech Expo	7	8/31/2020	9/6/2020
1.4 Team Contract Resubmission	7	9/6/2020	9/13/2020
1.5 Three Minute Elevator Speech	7	9/13/2020	9/20/2020
1.6 User Profile	14	9/20/2020	10/4/2020
1.7 Use Case Diagram	14	10/4/2020	10/18/2020
1.8 Draft Report	14	10/18/2020	11/9/2020
1.9 Fall Oral Presentation	7	11/9/2020	11/16/2020
1.10 Final Fall Semester Report	14	11/16/2020	11/30/2020
1.11 Testing Plan	7	2/8/2021	2/14/2021
1.12 Final IT Expo Abstract	7	2/15/2021	2/21/2021
1.13 Draft IT Expo Poster	7	3/1/2021	3/7/2021
1.14 Final IT Expo Poster	7	3/15/2021	3/21/2021
1.15 Fall Presentation	7	3/29/2021	4/4/2021
1.16 Final Spring Semester Report	14	4/3/2021	4/12/2021
1.17 Prepare for IT Expo	14	4/4/2021	4/13/2021
2.0 Technical Features: 209 days			
Feature	Task	Days	Start Date	End Date
Redesigning Permissions Feature: 34 days				
	2.1.1 Research	**(3)**	8/31/2020	
	2.1.1.1 Determine how the Risk Assessment permission system currently functions	1		
	2.1.1.2 Look into other existing permission management systems	1		

	2.1.1.3 Investigate the difference between implicit and explicit permissions	1		9/3/2020
	2.1.2 Design	**(5)**	9/3/2020	
	2.1.2.1 Draw sketch of the new Permissions UI	1		
	2.1.2.2 Flesh out action/subject paradigm	1		
	2.1.2.3 Plan database modifications	3		9/8/2020
	2.1.3 Development	**(17)**	9/8/2020	
	2.1.3.1 Create front-end UI	3		
	2.1.3.2 Hook up new UI to back-end	3		
	2.1.3.3 Make database changes	3		
	2.1.3.4 Implement selection of related permissions	3		
	2.1.3.5 Implement implicit/explicit permissions	3		
	2.1.3.6 Finish adding any remaining action/subject and related permission data	2		9/25/2020
	2.1.4 Testing	**(2)**	9/25/2020	
	2.1.4.1 Self-test	1		
	2.1.4.2 QA	1		9/27/2020
	2.1.5 Rework (if necessary)	**(7)**	9/27/2020	10/4/2020
Redesigning Merge/Unmerge Offender Feature: 21 days				
	2.4.1 Research	**(3)**	10/4/2020	
	2.4.1.1 Determine how the current merge feature works	1		
	2.4.1.2 Decide on the data model for an offender merge	1		
	2.4.1.3 Decide on the microservice execution responsibility for an offender merge	1		10/7/2020
	2.4.2 Design	**(2)**	10/7/2020	
	2.4.2.1 Draw a sketch of the new offender merge UI	1		
	2.4.2.2 Diagram how the process will work between the API and microservices	1		10/9/2020
	2.4.3 Development	**(7)**	10/9/2020	
	2.4.3.1 Make database changes	1		
	2.4.3.2 Create the new front-end UI	2		

	2.4.3.3 Hook up the new UI to back-end	1		
	2.4.3.4 Implement merge history and merge category success/error monitoring	1		
	2.4.3.5 Allow for merge retries on category error	1		
	2.4.3.6 Finalize styling of the UI	1		10/16/2020
	2.4.4 Testing	**(2)**	10/16/2020	
	2.4.4.1 Self-test	1		
	2.4.4.2 QA	1		10/18/2020
	2.4.5 Rework (if necessary)	**(7)**		10/25/2020
Offender Timeline: 21 days				
	2.2.1 Research	**(3)**	10/25/2020	
	2.2.1.1 Collect list of all offender data that has a date attached to it	1		
	2.2.1.2 Determine if necessary API endpoints exist already or if a new endpoint must be developed	1		
	2.2.1.3 Locate a suitable library to generate a timeline	1		10/28/2020
	2.2.2 Design	**(2)**	10/28/2020	
	2.2.2.1 Draw a sketch of the timeline	1		
	2.2.2.2 Determine location on the facesheet	1		10/30/2020
	2.2.3 Development	**(7)**	10/30/2020	
	2.2.3.1 Modify the database where needed to add date information	2		
	2.2.3.2 Modify the front-end facesheet UI to include the offender timeline	4		11/6/2020
	2.2.4 Testing	**(2)**	11/6/2020	
	2.2.4.1 Self-test	1		
	2.2.4.2 QA	1		11/8/2020
	2.2.5 Rework (if necessary)	**(7)**	11/8/2020	11/15/2020
Research Portal: 49 days				
	2.6.1 Research	**(3)**	11/15/2020	
	2.6.1.1 Determine how the Risk Assessment System currently interacts with the external JasperReports server	**2**		
	2.6.1.2 Decide on best process for retrieving data	1		11/18/2020

	2.6.2 Design	**(2)**	11/18/2020	
	2.6.2.1 Determine application workflow	1		
	2.6.2.2 Diagram the API process of fetching data	1		11/20/2020
	2.6.3 Development	**(30)**	11/20/2020	
	2.6.3.1 Create application	2		
	2.6.3.2 Create login page	4		
	2.6.3.3 Back-end work to fetch data via existing report query	4		
	2.6.3.4 Create report view			
	2.6.3.4.1 Create dc.js charts	8		
	2.6.3.4.2 Create dc.js data table	14		12/20/2020
	2.6.4 Testing	**(7)**	12/20/2020	
	2.6.4.1 Self-test	7		
	2.6.4.2 QA	2		12/27/2020
	2.6.5 Rework (if necessary)	**(7)**	12/27/2020	1/3/2021
Uploader: 56 days				
	2.3.1 Research	**(3)**	1/3/2021	
	2.3.1.1 Determine workflow for all types of validation and the final update job	1		
	2.3.1.2 Map out the technical architecture of the application	1		
	2.3.1.3 Research the current push notification process being used in Risk Assessment	1		1/6/2021
	2.3.2 Design	**(2)**	1/6/2021	
	2.3.2.1 Draw a diagram of the Uploader workflow from start to finish	1		
	2.3.2.2 Sketch each page involved in the workflow and include the responsibilities of each	1		1/8/2021
	2.3.3 Development	**(42)**	1/8/2021	
	2.3.3.1 Make any necessary script changes	2		
	2.3.3.2 Develop the react-dropzone upload page	2		
	2.3.3.3 Hook up the new front-end view to the back-end server	2		
	2.3.3.4 Develop an “object-mapper” JSON object to use for agency configuration	2		

	2.3.3.5 Develop a validation list page (errors, reupload)	2		
	2.3.3.6 Develop an upload list page	1		
	2.3.3.7 Connect lists to job progress/errors via WebSocket technology	1		
	2.3.3.8 Implement detached child processes so that each job has its own pid	1		
	2.3.3.9 Add "delete job" button	1		2/19/2021
	2.3.4 Testing	**(2)**	2/19/2021	
	2.3.4.1 Self-test	1		
	2.3.4.2 QA	1		2/21/2021
	2.5.5 Rework (if necessary)	**(7)**	2/21/2021	2/28/2021
Database Replication: 35 days				
	2.5.1 Research	**(5)**	2/28/2021	
	2.5.1.1 Research database replication	1		
	2.5.1.2 Look into database replication options for PostgreSQL	1		
	2.5.1.3 Research examples for implementing database replication	1		
	2.5.1.4 Outline the risks associated with database replication and make a plan for managing each	1		
	2.5.1.5 Research load balancing / failover and consider the available options for PostgreSQL	1		3/5/2021
	2.5.2 Design	**(2)**	3/5/2021	
	2.5.2.1 Diagram the new technical architecture of the system	1		
	2.5.2.2 Consider triggers/functions that may need to be added to automate the replication management and monitoring	1		3/7/2021
	2.5.3 Development	**(13)**	3/7/2021	
	2.5.3.1 Implement database replication	5		
	2.5.3.2 Add any database triggers or functions that may be needed	7		3/20/2021
	2.5.4 Testing	**(15)**	3/20/2021	
	2.5.4.1 Replication Test	15		4/4/2021

Table 14 Project Timeline

Table 14: Project Timeline

x. Problems Encountered & Analysis of Solutions:

The Problems Encountered below are the highest-level problems encountered during this project. Each problem is followed directly by the solution that eventually resolved it.

- *Problem:* HTML input elements don't have a built-in attribute for creating indeterminate checkboxes
    - *Solution:* A new React.js component was created to handle the requirement for an indeterminate checkbox that dynamically interprets the type of check that should be shown on the input element

- *Problem:* The merge/unmerge feature is fairly complex and was confusing to end users
    - *Solution:* Simple visualizations were added to this feature along with help text to make the process more straightforward.

- *Problem:* npm didn't have a library built specifically for creating React.js timelines
    - *Solution:* A tangentially-relevant library was used to create a timeline. A horizontal scrolling menu was set up and then basic css was utilized to "draw" the timeline itself.

- *Problem:* The chosen graphical/charting solution, dc.js, didn't work out-of-the-box with React.js. There were a lot of issues trying to get that library to work
    - *Solution:* An npm library was located that restructured dc.js elements so that they worked properly with React.js

- *Problem:* User's Uploader front-end client can't wait for a synchronous response from the back-end server while it might be busy running ~28,000 row, multi-day jobs
    - *Solution:* The node child processes for these jobs were detached from the main node.js server thread and queues were used in tandem with WebSocket technology to push periodic updates to the front-end via push notifications through a hashed channel

- *Problem:* Shifting requirements from the sponsor
    - *Solution:* Changed gears to reroute toward more immediate requirements and altered the contract timeline to reflect this

xi. Recommendations for Future Improvement:

- *Recommendation:* Track process ids of detached jobs
    - The application already has the uploader scripts running as detached child processes, but it should save the process ids in the database so that the front-end can track whether a job has failed or is still processing.

- *Recommendation:* Look into using React Context with Chart.js instead of using dc.js
    - dc.js is a bit temperamental when used with React.js because both want to control the DOM. Because of this, it was pretty difficult to implement. Though a workable solution was eventually found, there are likely better ways to implement the same functionality with less headache.

- *Recommendation:* Add additional Role-Permission control within the application
    - The roles and permissions can be further enhanced. It might even be worth separating this into an external application entirely. The first thought that comes to mind would be adding a way to bulk add/remove permissions to/from a list of selected roles within an agency.

Future Plans:

In the future, the team plans to continue working on all of these features, specifically the Research Portal and the Uploader. There are configuration things baked into the architecture that didn't get completed because the team ran out of time. More reports and scripts will be added as both of these applications travel through the testing pipeline on their way toward being deployed to production.

III. CONCLUSION

This project was filled with lessons. The project management experience gained during this time has shown exactly how to take prior development expertise and transform it into a marketable package. Even if that's only getting a first crack at procuring these types of project reports, the benefits are obvious to someone on a path to an eventual career as an IT professional. Working within a team means collaboration and cooperation which, in turn, necessitates a high level of understanding for both the industry itself and the professional responsibilities of one's colleagues. Basically, knowing how the system works helps give some insight into how to work most effectively within that system.

This project started as nice-to-have features that the Risk Assessment development team had always intended to bake into the new version along with a wide sweeping front-end migration from Angular.js to React.js. This senior design project afforded the opportunity to spend class time implementing those ideas and growing as a software developer.

Despite somewhat significant prior experience in software development and working with the Risk Assessment technologies specifically, there had never been much opportunity to develop a full feature from start to finish, let alone a full application (or two). The team had no TypeScript experience at all. Surprisingly, developing the new applications actually served to strengthen understanding of the Risk Assessment System itself. Most of the team's experience prior to this project involved fixing minor bugs and developing operational patches or scripts while the primary focus was on on managing

emergent client concerns and bringing new agencies onboard. So it was a great experience to be able to reach a little further.

This project coincided with one member taking a more active role in leadership of the Risk Assessment application. This project was completed amidst a hectic gauntlet of learning how to be a leader and how to manage a team, especially in COVID times. It's almost like COVID put training weights on the entire IT industry. The team was forced to deal with project timelines and the forced adjustment of those timelines. Estimates were required based on complexity of a potential feature *and* perceived skill of the developer assigned to the task. React.js was basically new to the team at the start of this project and it has become something the team is extremely comfortable with now. Between code reviewing for the developers, managing branches and environments, deploying releases, communicating with clients, being invited to internal ITSC meetings, leading sprint and daily standups, and managing the database administration for the application, this project has granted a multitude of valuable lessons.

All of that has culminated in this wonderful project. As an unexpected blessing, the sweeping front-end changes empowered the team to find small bugs that neither the development team nor the users ever knew were there. The team was able to fully design and build features and applications from the ground up, finally being able to give the end users some of the tools they'd been requesting from the ITSC for years. Before this project, the team was severely lacking in front-end development experience. Not only was the team weak in terms of understanding how to use a front-end framework, but they also had extremely limited experience with advanced CSS concepts and actually making a web application look presentable. More importantly, it became obvious that design at

its most basic is delivering something that meets both the expectations of your team as well as the needs of the end users. Sleek and modern isn't always correct when the client wants something basic and functional. It's more about balancing expectations than delivering a masterpiece.

Work will continue on the Risk Assessment System going forward. Both the Research Portal and the Uploader are basically prototypes, with each having only one report and one script available, respectively. Much of the following semester will consist of tuning replication settings to make the logical replication as synchronous as possible and rolling out the initial phase of version 4. It is an exciting time at the ITSC, one described mainly by hard work and scrambling to meet deadlines. Despite the challenges ahead, however, this project has confirmed that the team will be much greater after overcoming them.

IV. REFERENCES

i. Alper, M., Durose, M. R., Markman, J. (2018). 2018 update on prisoner recidivism: A 9-year follow-up period (2005-2014). Retrieved from https://www.bjs.gov/content/pub/pdf/18upr9yfup0514.pdf

ii. Lovins, Brian and Latessa, Edward J. "Creation and Validation of the Ohio Youth Assessment System (OYAS) and Strategies for Successful Implementation." *Justice Research and Policy*, vol. 15, no. 1, 2013, pp. 67–93., doi:10.3818/jrp.15.1.2013.67.

iii. Latessa, Edward J., et al. "The Creation and Validation of the Ohio Risk Assessment System (ORAS)." *Federal Probation Journal*, vol. 74, no. 1, June 2010.

iv. Latessa, E. J., Lovins, B., & Lux, J. (2017). The Ohio Risk Assessment System. *Handbook of Recidivism Risk/Needs Assessment Tools,* 147-163. doi:10.1002/9781119184256.ch7

v. APPENDICES

Appendix A:

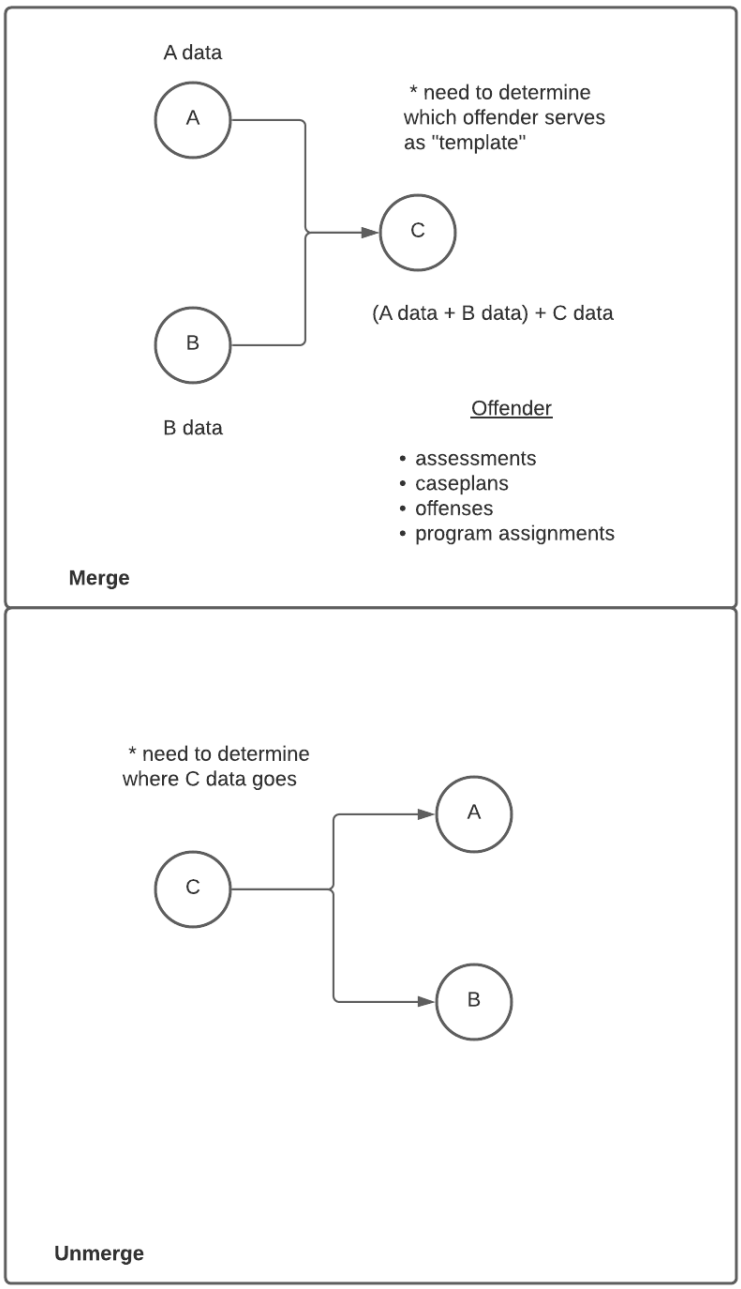

Figure 4 Merge/Unmerge Technical Diagram

Figure 4: Merge/Unmerge Technical Diagram

Appendix B:

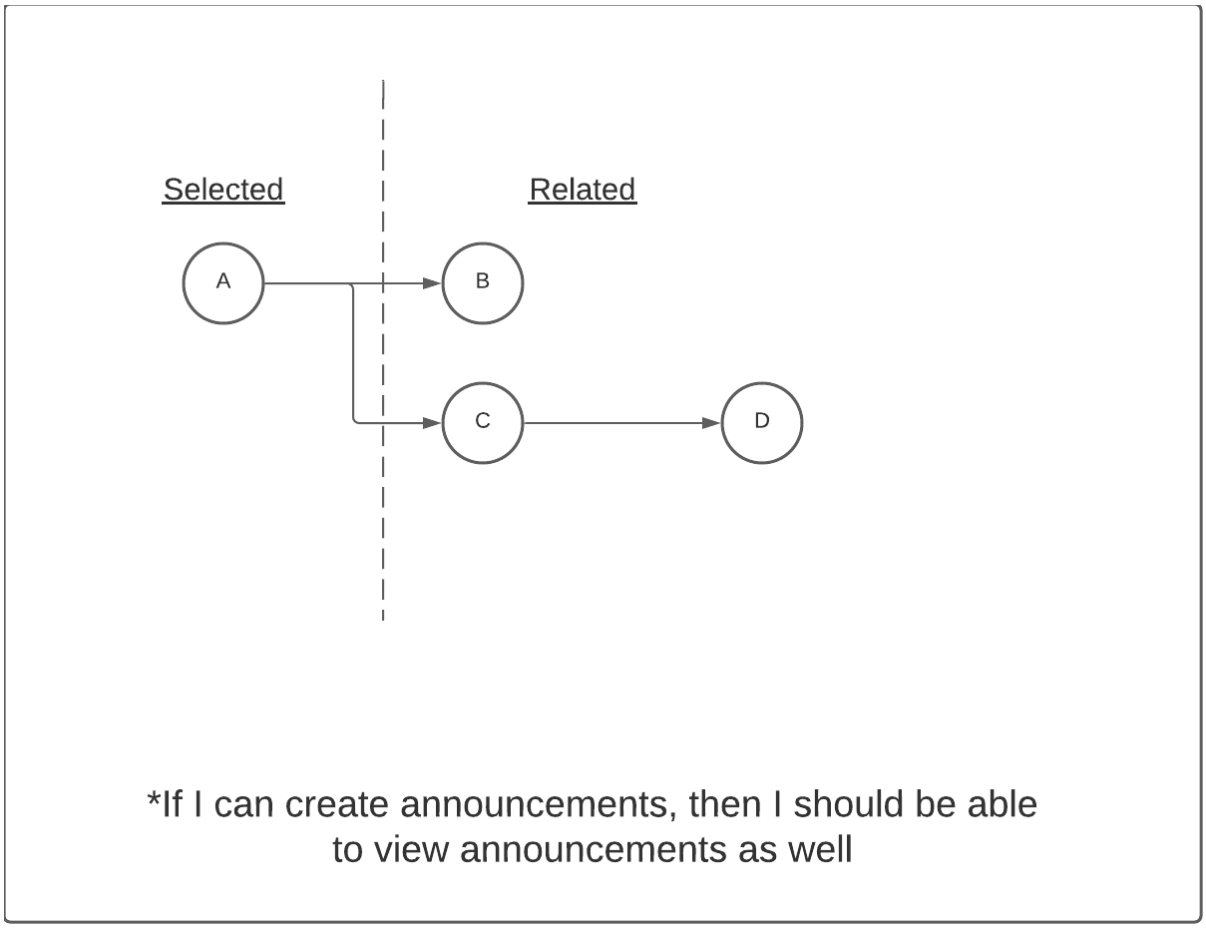

Figure 5 Role Permissions Selected/Related Technical Diagram

Figure 5: Role Permissions Selected/Related Technical Diagram

Appendix C:

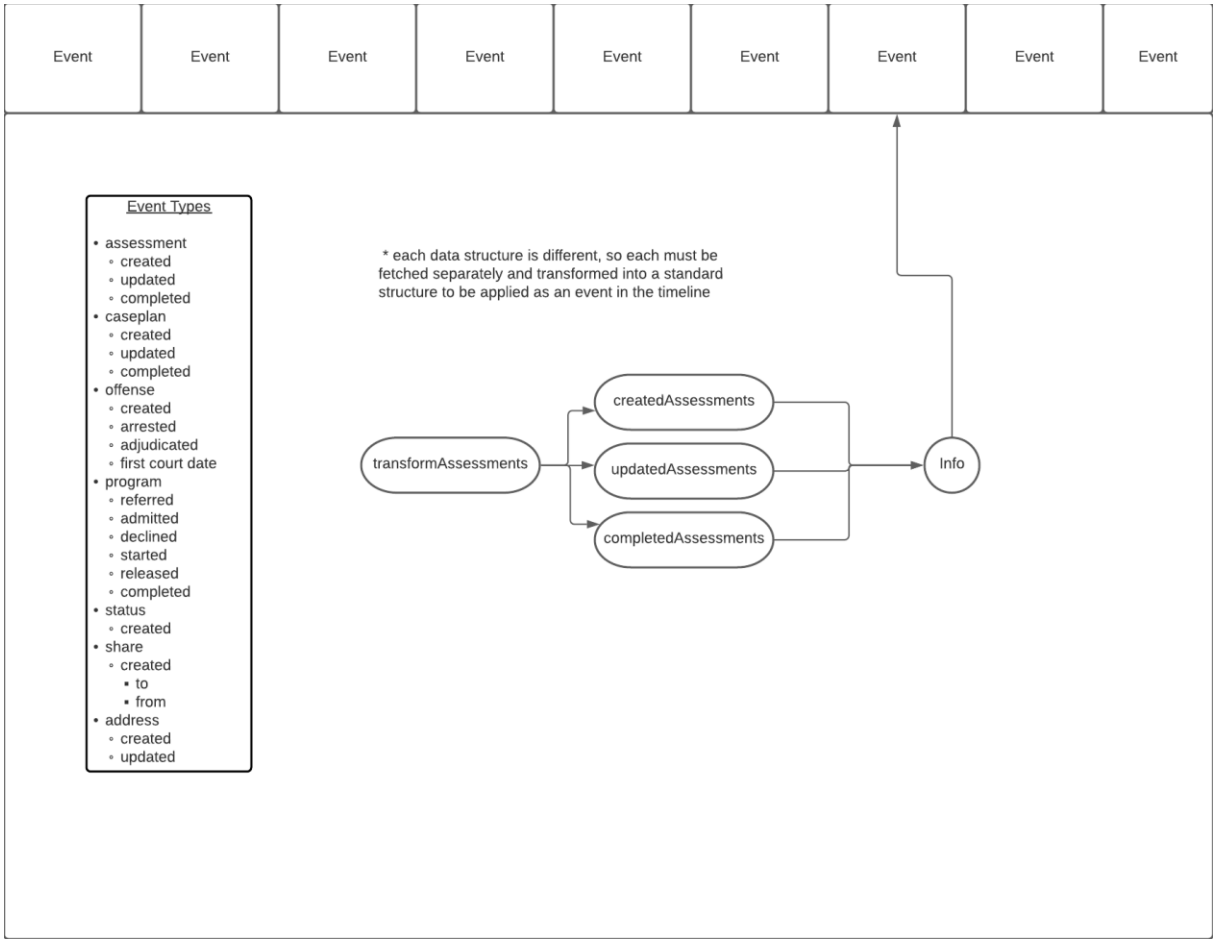

Figure 6 Offender Timeline Technical Diagram

Figure 6: Offender Timeline Technical Diagram

Appendix D:

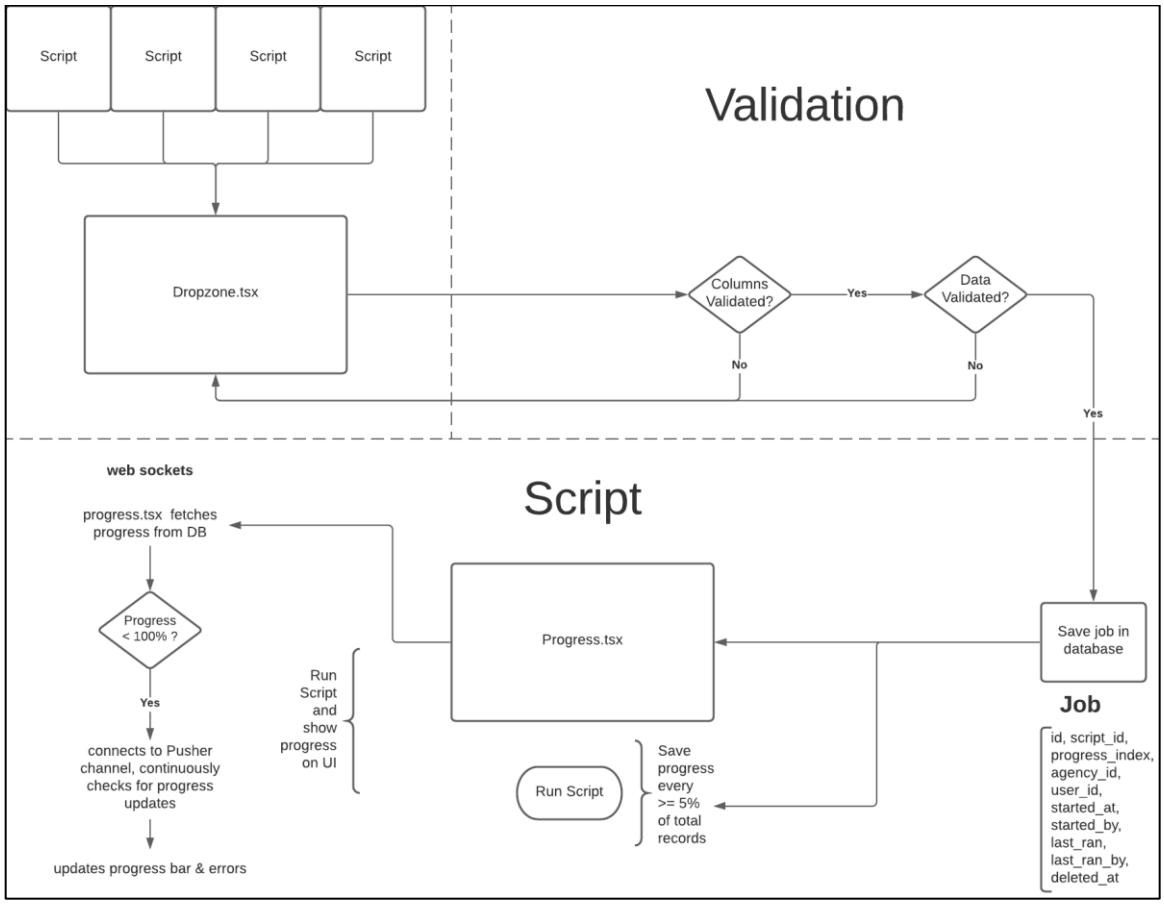

Figure 7 Uploader Technical Diagram

Figure 7: Uploader Technical Diagram

Appendix E:

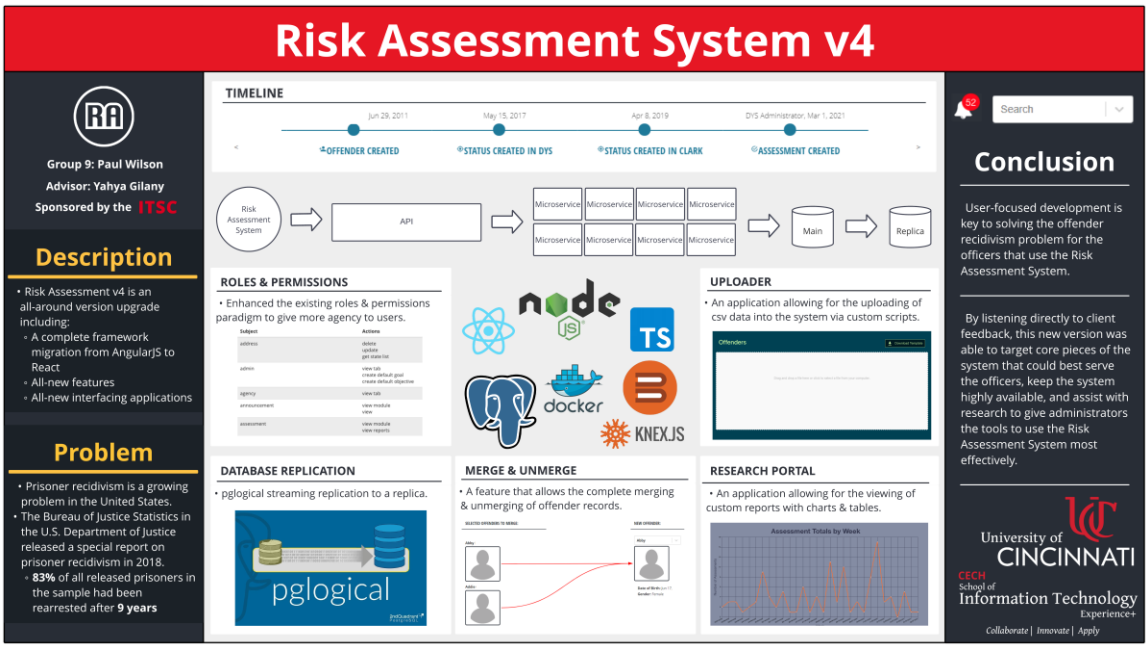

Figure 8 IT Expo Poster

Figure 8: IT Expo Poster