Certn Support Resolution Tool

Project Report

https://github.com/Certn/support-tool

University of Victoria

SENG 499 - Team 3

Reed McIlwain V00862212		Mahum Azeem V00886660
Kelvin Filyk V00810568	Amandeep Johal V00856653	Eric Dahl V00870685
Keegan Kavanagh V00900571	Rafay Chaudhry V00797951	Conor Butte-Landsfried V00876461
Jonathon Squire V00863838	Alexander Schell V00769833	Paramvir Randhawa V00848357

Executive Summary

As part of their capstone project, a team of students from the University of Victoria chose to work with Certn to help them develop a system to help improve their workflow. Through consultation, the team and Certn set out to create the Support Resolution Tool. This document goes over the project and its development in its entirety, from the initial goals, through to the work that was done and the work that can be done.

The goal of the Support Resolution Tool was to help support staff at Certn search for applications as well as send consent documents and onboarding/report links to the person in the selected application. This tool was also to be built as a web application using Certn's existing tech stack, including using Typescript, and following existing Certn UI design guidelines by harnessing AntD styling components.

After three months of development, the team was able to accomplish these goals. All features requested by Certn (the searching of applications, sending of documents and links) have been implemented, as well as other features that were required due to the nature of the app itself, like the ability to log in to Certn. Some extra goals for functionality raised after the start of development were not completed however, and there remain some minor areas for improvement. Examples of these areas include the app currently using mock endpoints to simulate functionality and some glitches due to AntD styling.

With the features and shortcomings covered, this document then lays out the work that remains to be done to fully bring this application to full working condition within the Certn ecosystem. Work going forward will include replacing the aforementioned mock endpoints with real ones, as well as possibly implementing the missed stretch goals. To assist with this, the document and Readme of the repo will provide info on how to implement new features.

Overall, the team is confident that we have produced a quality product that, with some work done in respect to integration, will provide Certn with a high-quality component to enhance their workflow and simplify tasks for their support staff. Furthermore, the team has found this to have been an excellent learning experience, both for the technologies used, and as an experience in taking a product from concept to delivery.

Introduction

In the fast-paced world we live in today, the desire for highly talented individuals has resulted in a need to expedite the hiring process to avoid losing these individuals to other positions. For businesses that require background checks, the hiring process becomes strenuous, as receiving the appropriate checks can take upwards of several weeks. This delay is arduous for employers and employees alike. Certn saw this as an opportunity for change.

Certn is a data acquisition company that provides the average individual the ability to perform background checks on potential hires, tenants, and more. These checks include ID verifications, criminal record checks, education verifications, etc. from over 200,000 sources and databases across the globe including the RCMP, academic institutes, United States criminal records, among others. Although the acquisition of data is relatively fast from a technical view, manual interaction from the Certn support team is often required in the form of responding to emails to provide permissions, forms of identifications, or clarifying aspects of the search.

Business Opportunity

The most common delay in the delivery of Certn's reports is the need to resend consent documents, onboarding material, and/or links to the report itself. To do so, a member of Certn's support team is required to manually dive through the database to acquire and verify the information in question before re-sending them to the client. Processing these common requests by hand can take upwards of 30 minutes per request, creating a hassle for the support team.

Team 3 views these delays as unnecessary and therefore have offered to develop the Support Resolution Tool in response to Certn's grievances. Although the support tool does not generate revenue itself, minimizing additional delay from when a user requests a report to when said report is delivered accentuates Certn's business model while increasing customer satisfaction.

Business Objectives

By developing the support tool, Team 3 hopes to expedite a number of Certn's time-consuming, manual processes. To do so, the tool must be capable of navigating Certn's wide range of API calls to perform the same tasks done by hand in a timely, user-friendly manner. This includes being able to:

1. Search for an application via a keyword, first name, last name, date, etc.
2. View and email any number of consent documents associated to a given application
3. View and send both onboarding and report links via email to affected parties associated to a given application

The support tool responsible for completing these actions should exist in a modular, web application format such that Certn can either adopt the features into their existing tools, or can be built into a new, superior support tool that better fits Certn's needs.

Scope and Limitations

This section will lay out the major features and design aspects present in the Certn support tool as well as any which were excluded.

Major Features

The application can be split into three broad categories: the login functionality, the search page and the application page. A brief introduction on all categories is given in the following sections.

Login Page

The Login Page allows for secure access to the application to authorized users in addition to generating tokens for API calls. Any attempt to access the application by a user that is currently not logged in will result in a redirect to the Login Page. The main element on this page is the login form, prompting users to enter the email and password associated with their account. The Login Page also has links out of the application to create an account and to recover a forgotten password. Upon successful login, the user is provided with a session key, granting access to the rest of the app, and is redirected to the main search page of the application. If the user enters an incorrect password or email, they are simply redirected to the login page with an error. To assist users, forms on this page will error out if not given valid input (e.g. password form will show a "Password not entered"). The mechanisms used in the process of logging in, and determining if a user is logged in are discussed more thoroughly in the "Security" section.

In case of a user trying to access a nonexistent page on the application, the routing system will simply direct a user to a customized 404 error page allowing the user to click back to the main search page. If a user is not logged in, they will not reach this 404 page, and instead will be redirected to the login page.

Search Page

The Search Page is the default view a user will be directed to once logged in. The Search Page serves a specific function: the ability to search for applications. To aid users of the system, searches can be conducted in one of two ways: using the basic search option or the advanced search option. To search for applications, it is possible to search using the following identifiers: first name, last name, or email associated with an application. It is also possible to search for applications using any of the identifiers individually or in aggregate. If identifiers are used in aggregate, only applications that contain all identifiers will be returned in the search result. The basic search option lets a user search for applications using identifiers separated by a space. However, the system also offers an alternative user flow for searching by toggling the advanced search options. Once the advanced search options are toggled, the search bar expands to provide specific search fields for each of the identifiers. It is then possible to search for search identifiers individually or in aggregate as it was done with the basic search bar. As searches can return hundreds if not more applications, it was decided to add pagination to ease navigation of the returned applications.

Each page of the search results table shows ten applications at a time. A user can then jump between pages using the page numbers at the bottom of the table or use the arrows key to move one page forward or one page back. Application search results are sorted by the “Created By” column in the search results table. Although the UI does not support this yet, it is possible to specify what column to order the search results by; this can be done by passing the parameter ‘ordering’ with an appropriate value to the ‘getApplication’ used by the search bar.

Application Page

The Application Page consists of three major features, of increasing complexity: Advanced Application Info, Critical Checks Status, and Application Actions. The Advanced Application Info is a table at the top of the page giving the pertinent information related to an application; namely, the email, first name, last name, phone number, creation date, last modified date, name of who ordered the check, team of who ordered the check, and the overall status associated with the application. This is the same information provided on the Search Page.

Next is the Critical Checks Status dropdown on the right-hand side of the page. Though Certn performs many types of background checks, there are nine that are considered “Critical Checks”: US Criminal Record Check, Criminal Record Search, SSN Verification, References Check, Motor Vehicles Records (USA), Equifax Credit Report, Employment Verification, Education Verification, and Credential Verification. The status of these nine checks can be a contributing factor to a stalled or erroneous application, thus it is necessary to display their status. The “Complete” category is for checks which have been processed and require no further action from a Certn employee. This complete status is assigned regardless of whether the check itself passed or failed, as this determination is typically beyond the employee’s control; thus, the status labels corresponding to this category are ERROR, RETURNED, VERIFIED, SYSTEM UNABLE TO VERIFY, PARTIALLY VERIFIED, and UNVERIFIED. The “In Progress" category is for any checks which have not been completed yet and corresponds to the status labels ANALYZING, PARTIAL, PENDING, and VERIFICATION PENDING. Finally, the “Failure” category is for checks which will require an employee’s attention. Currently, there are no status labels corresponding to this category, thus it is treated as a default switch case in the code. This determination is discussed further in the following “Limitations and Exclusions" section. It is possible that a critical check was not requested for a particular application, thus the status labels

NONE, UPGRADE, and UPGRADE TO VERIFY correspond to critical checks which should not be displayed in any of the three dropdown categories.

Finally, the largest feature of the Application Page is the Application Actions. This is the sidebar and the Recipient box in the middle of the page corresponding to the three currently implemented actions: Send Onboarding Link, Send Report Link, and Send Consent Documents. All three actions display the email associated with the application and display an error message if no email was found. This email can be edited if it was incorrectly filled out, and an email can be sent from the Recipient box with the corresponding action's information. Currently, these emails and email edits are sent to a mock API, which is explained further in the "Future Work" section. The Send Onboarding Link action will send an email to the recipient with the onboarding link allowing them to provide their information to start the background checks, the Send Report Link will send an email with a link to their Certn report, and Send Consent Documents will send an email with the selected consent documents as attachments. Both the Send Onboarding Link and Send Report Link actions have the URL displayed in the Recipient box for one-click (using the Copy icon) or manual (by highlighting) copy-and-paste and will display an error message if the corresponding URL was not found. The Send Consent Documents Action has a scrollable list of consent documents that may need to be sent to a recipient. A Certn employee can preview any of the consent documents with a PDF Viewer modal before marking the necessary consent documents for the email.

Design

One feature affecting all elements of the app is the UI, and the fact it was built to Certn design specifications. These specifications were laid out in a Figma document providing general colour, layout, and text guidelines, along with basic designs for elements like buttons and forms. From these elements, a number of prototype pages were built out for the app UI. With approval given to these prototypes, the UI was then built to them as closely as possible.

As with the rest of the Certn UI, AntD was used as the basis for most elements in the UI, with elements being customized with Certn brand colours, and fonts, along with some small modifications like button radiuses. This reduced the amount of time taken to develop the elements themselves, and allowed the team to focus on element layout within the app. The usage of AntD also allowed the inclusion of more enhancements that may have not been included if they were made from scratch, like the aforementioned form errors and input validation, as well as loading bars and pagination.

After the completion of the UI, a final pass was made of the entire app to ensure consistent colours and text throughout the app. This included the centralization of commonly used Certn UI specs like colour, breakpoints and fonts to a central theme tsx file, and then referencing these specs when required. This allows for the easy modification of these elements if needed, and helps with style consistency. Furthermore, throughout the app, all styling of elements has been contained to __SC.tsx files, to make editing of styling smoother.

Limitations and Exclusions

The following section describes the features which were not implemented along with the reasoning for their exclusion. This section does not include items for future work, known bugs, or stretch goals, which are rather covered in the "Future Work" section.

On the Search Page, the option of sorting search results by column was proposed. However, due to the fact that the Search Page currently only pulled ten results at a time (thus complicating how to pull the results after a sorting method was chosen) and the requirements elicitation indicated that this was not a desired feature, this was not implemented. Similarly, adding a search field for phone numbers was proposed for the Advanced Search filters, but since Certn's API did not support searching phone numbers the way it did for name and email, this feature was also not implemented.

On the Application Page, the "Failure" category for the Critical Checks Status is not a valid use case for how Certn currently applies statuses. There is a project following the completion of this Support Tool which will add the appropriate error-checking to sort Critical Checks into the respective categories according to their status and result. For now, however, only "Complete", "In Progress", and undisplayed are valid use cases.

Technical Details

This section explains how to set up the support tool and go over the project layout and architecture, in addition to any other considerations that are needed when further developing the tool.

Project Setup and Testing

This project requires an installation of Nodejs with a version of at least 14+. The current version, if it exists, can be checked with node -v. If needed, download links to newer versions can be found at https://nodejs.org/en/download/

Version 7 of npm or greater is also required, and can be checked with npm -v. The following page contains set-up instructions: https://docs.npmjs.com/try-the-latest-stable-version-of-npm. If using windows, the tool at https://github.com/felixrieseberg/npm-windows-upgrade might also be useful.

Once the above prerequisites are installed, and the repo has been cloned, the tool can be prepped for development by running the following commands in the root directory:

- npm ci - Installs all necessary dependencies
- npm start - Starts the app on port 3000

A full list of the npm scripts which can be run on this project can be found in the package.json file of the project.

This project uses Cypress to test the front end operations. If testing just the service, without a locally running instance, use: npm test. This will start the local service, run all tests against it, and then shut the local service down. The results of the tests will be placed on the cli.

If there is already a service running (e.g. npm start was run beforehand) the following commands can be entered:

- npm run cy:run - Runs tests without the Cypress UI, returns results to the cli.
- npm run cy:open - Runs the Cypress app itself with the UI showing performed tests.

If running the tests before committing new changes, the command `npm run cy:run` should be sufficient to ensure the changes did not break anything. If something is broken however, or new tests are being developed, then the command `npm run cy:open` would be the better option. Either option, however, will require multiple terminal instances.

Layout and Architecture

The application was built in Typescript using the React library, for the purpose of developing a modular, sophisticated user interface while also incorporating industry-leading technologies. Wherever possible, functional components were implemented in place of class-based components. Prioritizing the use of functional components enabled us to practice concise, code-specific testing, class simplification, and helped us better ensure component separability.

Since project inception, our goal has been to research the most reliable internal use development patterns established in industry. Because the premise of our application is to function as an IT support tool for internal usage, we chose to adopt a tabular GUI layout for efficient listing and interaction with multiple client records. The second goal was to then incorporate tools for reading and manipulating client records in a way that complemented the needs of support staff.

Record objects were designed to be focusable, allowing for independent editing at a reduced scope. After a client record is selected, a tool bar containing multiple interaction options appears. So far, interaction functions have been limited to emailing application requests and onboarding links, due to time constraints of completing the project within a semester. That being said, our infrastructure is easily scalable in the case that Certn developments wish to incorporate new functionality later on.

Using the react-router library, our application was built with a mindset of providing rapid, simple access to functionality. The application home page therefore incorporates the bulk of our functionality, containing the general/specific search bars and client records display table. Relatedly, all functional path lengths of our application are reached within 4 clicks of initial login.

Security

During implementation, Team 3 was required to adhere to strict security guidelines set out by Certn. In accordance with their policy of maintaining data privacy for their userbase, our team was not allowed full access to their API. As an alternative, Certn granted us administrative access to their demo application API, which is typically reserved for use by prospective Certn clients. Full access to the demo API enabled us to send more detailed API requests, including making use of search functionality and requesting application-specific details. Full admin access also allowed our entire team to work within a single shared demo database and use the full suite of application types, thus allowing us to mimic how a full support team would share access to a large database of application records.

Accessing Certn database servers still requires logging into their application as a Certn staff member, therefore, signing in had to be implemented thoughtfully. Login functionality is handled by passing a base-64 encoded username/password combination directly to the Certn login server. On a successful login, the client is granted a time-limited access token for read/write access to the Certn API. Our application stores this access token locally and uses it to populate data within our application. Functionality was built to automatically detect token expiry on the client-side (logging the user out accordingly).

Other Considerations

The support tool file system is arranged such that page-specific components are stored within mutually shared mid-level directories. Storing component code files in this way allows for our code base to be easily parsable, and was done in the interest of keeping on-boarding straight-forward. This was also done to fit with the preferences of the Certn team, who use this exact file schema. The top level of the support tool consists of six directories and eleven source files.

The six directories are broken down as follows:

- The “husky” directory contains the necessary files for husky to run and lint the commit messages.
- The “cypress” directory contains our suite of integration tests.
- The “docs” directory contains all documents we created to help upkeep project standards (at the moment it only contains the pull request template).
- The “node” directory stores all necessary node modules required by the application.
- The “public” directory contains our index.html file.
- The “src” directory contains our project code and components tree.

The src directory is organized very specifically, and includes an API directory, a “components” directory, a “themes” directory and six standard React application files. The API directory consists of two directories, one pertaining to the Certn API and the other to a mock API (for application testing and usage while endpoints haven’t yet been connected to). The component directory contains a subdirectory pertaining to each unique web page included in the support tool. Each webpage directory contains all the necessary components used by that webpage. The themes directory holds all CSS standards to be used project-wide. Lastly, The files in the “src” directory include the app, index, interfaces, and context files, which pertain to a number crucial base-level code structures which account for communication, structure, and state transfer within our application. App.tsx is the main file where the router is held and index.tsx is where our app gets called. Lastly, Interface.tsx and ApplicationInterfaces.tsx hold all interfaces needed for the app. Note that .tsx denotes the standard Typescript file format.

The eleven files in the root directory are configuration files for the support tool and the README to help new developers understand the project.

Future Work

This section goes over some of the leftover/future work that can be done on the support tool, such as fixing remaining bugs, replacing our mock endpoints, adding new application actions, and completing any of the stretch goals we did not.

Leftover Bugs

We only had a couple known bugs that did not get fixed, both of which are minor visual issues.

One relates to the email and phone number fields on the search page. When entering invalid input, the border should be red along with the rest of the error indicators, but instead it shows as green. This is likely due to how AntD's styling was overridden, as borders were set in general to highlight as green when in focus to match with Certn's colors. So, there needs to exist a way to keep the green color over AntD's default blue, while also allowing the border for errored fields to be red when in focus

The other bug involves the page numbers on the search page. When navigating through the pages, the current page number turns white while the new applications are being fetched, making it unreadable. Instead it should be green.

Replacing Mock Endpoints

One of the larger hurdles we encountered with the project was that some of the required endpoints were not accessible to us. To complete the project and produce a functioning application, the team had to create several mock endpoints. To create the mock endpoints, team members had several thorough discussions with the Certn team to develop accurate and meaningful data models. It was important to make sure that the models were accurate and in line with Certn's other models so that the project handoff and development could progress seamlessly after the end of the term. In total, three mock endpoints were developed for use within the project. The first endpoint was built to fetch consent documents related to a specific application. The other two mock endpoints were centered around the email attached to a specific application, with one allowing an email to be sent to the attached address and one allowing the attached address to be changed. All mock endpoints can be found in the index-mock.tsx file.

Fetching Consent Documents

The first mock endpoint built was designed to simulate fetching consent documents from the Certn API. It is named "getListOfPdfs" and does not take any parameters. The mock implementation utilizes a JSON blob provided to the team by Certn as the main data source. When the endpoint is hit, a document from the JSON blob that is labelled with the type "consent" is returned. This implementation relies purely on static data and will return the same consent document regardless of the application.

There are several ways that fetching consent document endpoints could be implemented server-side, however, it is important to note the data model that the client is expecting, which was provided by Certn and can be found in the Certn-Mock-API directory. Looking forward and considering the requirements provided by Certn, our recommendation is to have the endpoint accept an application id, or the region attached to the application and then return a list of consent documents relevant to the specific application.

Sending Emails

The second mock endpoint is named "sendEmail" and was built to facilitate sending emails containing documents or links to the email attached to the application. This endpoint is entirely non-functional and will simply print a message out to the console when hit. This endpoint is currently triggered by completing one of the three actions provided by the application actions. The endpoint requires that details about the calling action be provided such as the action type and any links that should be sent. These details are delivered through a passed EmailInfo object, which is an interface defined in Interfaces.tsx.

Creating the endpoint to send emails will require Postmark configuration, in addition to creating the endpoint itself. To facilitate a smooth transition, the team recommends that the server endpoint be built to accept an action type along with the relevant email contents such as document ids or relevant links. Based on the action type, the endpoint would proceed to send the appropriate email via Postmark.

Updating Application Email

The final mock endpoint was built to simulate changing the email associated with a selected application and is named “updateEmail”. This endpoint accepts the updated email and will return the updated email if it is successful. The functionality of this endpoint is spread across several helper functions as it changes the client-side email associated with the application. When the email is changed via the Edit Email modal, the email address on the application page reflects this. However, this is not a persistent change, and the email will revert when the page is refreshed.

This endpoint could be the most complex endpoint to create as it involves changing the email associated with the application everywhere. This most likely means that there are many touch points, leading to an increase of complexity. Without having more domain knowledge of Certn data management, it is difficult to make any recommendations.

Adding New Actions

The ability to perform actions on a selected application is the focus of the project. Currently, there are three actions that have been implemented: sending an onboarding link, sending a report link, and sending consent documents. All existing actions will send the above to the email associated with the specific application. As a natural progression to the application, it is anticipated that more actions will be added to the Application Page and in turn the Application Action list and Action tabs. In this section the process for adding a new action, along with the specific touch points will be discussed.

Application Page

The Application Page is the first point of contact and the source of truth for all of the current actions. The Application Page's core functionality can be found in ApplicationPage.tsx. Within this file a call is made to the Certn API to fetch a specific application based on an ID provided in the URL. This data is then parsed and passed to various child components via props. Currently, three props are passed to the ApplicationActions component. The first prop is "data " which contains high level information about the application and is of type AdvApplicationInfo which can be found in interfaces.tsx. The second prop is "links" which currently contains the onboarding link and the report link associated with the specific application. It is of type LinkInfo which can be found in interfaces.tsx. The final props is "updateEmailMOCK " and passes a function. This prop can be removed when the update email API endpoint is implemented server-side. When adding a new action, it may be necessary to add a new prop to pass relevant application information to the ApplicationActions component or the types AdvApplicationInfo and/or LinkInfo may need to be updated to account for more data.

Application Actions

The Application Actions component is the child of the Application Page and can be found in ApplicationActions.tsx. This component is responsible for displaying the menu of actions on the application page and also is the parent of the ActionTabs. When adding a new item, a new menu item will need to be added to the action menu. The new menu item will need to have a key, title and any handlers such as onClick defined. Examples of menu items can be found in ApplicationActions.tsx. The Application Actions component keeps track of which action tab is selected and passes the active action key as a prop to the Action Tab component. Several other props are passed to the Action Tabs component such as the email associated to the application, the LinkInfo object, any consent documents relevant to the application (only applicable for the send consent documents action), a loading boolean and a updateEmailMOCK function passed from the Application Page. The loading props should be toggled any time there is an async call made within the Application Actions component. Depending on the new action, it may be necessary to add additional props to manage other relevant data.

Action Tabs

The final component that is relevant to adding a new application action is the Action Tabs component. This component is responsible for displaying the selected application action and displays the appropriate forms and text. Currently, the "action" prop controls which tab is being displayed via a ternary operator. When adding a new action, it is possible to add another conditional to account for the new application. This method would be adequate for adding one or two new actions, however, scaling past this it is recommended to make use of tabs or another method conditionally rendering actions based on the active action.

Missed Stretch Goals

This section explains some of the stretch goals our team didn't get to during the duration of the project.

Ability to Generate New Report Links

This stretch goal was listed by Certn as an extension of the ability to send a candidate the report link or code in the event that they do not know how to access their report. It was decided early on that a report link would be sent to a listed email instead of a report code, thus putting the possibility that this stretch goal would be completed in doubt. It was briefly discussed that the ability to create a randomized report link could be implemented, but such discussion was tabled early on for the fourth sprint. When the stretch goal that gives the user the ability to change the email associated with an application was chosen instead, this one was deemed to not be a priority.

Ability to Search Consent Documents by Name

This stretch goal was listed by Certn as an extension of the ability to send a candidate consent documents so that they would know exactly what they are agreeing to. However, one team of developers were in charge of all functionality regarding the consent documents and another for the searching functionality of the application. Thus, it posed a bit of difficulty in regard to which developers would handle it. Due to the nature of the stretch goal, it was decided that if a developer team would take on the work, it would be the one working on the searching functionality. The stretch goal was not brought up in the fourth sprint as it was deemed to not be a priority when considering other work to be done.

International Dashboard

This stretch goal was requested of us by Certn in the middle of the third sprint for development during the fourth sprint. This dashboard would list all applications which have an international criminal check requested. The intention was to integrate the International Dashboard into the support tool but to restrict access so that only those with a certain level of clearance could access it.

Ultimately, it was acknowledged by Certn that the request to add in the international dashboard was last minute and any work put towards it would be appreciated. The actual code provided by Certn to help incorporate the dashboard was provided a few days later. It was decided that all discussion about this stretch goal would be tabled until the planning of sprint four.

For sprint four, it was decided that work would only be done on the international dashboard once all already agreed upon work for the main support tool had been finished. At that point, if a group found it had the time to implement the dashboard, even partially, it would be done. Ultimately, with work freeing up with less than a week to go in the sprint, it was decided that the international dashboard would not be implemented.

However, with documentation on the support tool included as well as comments in the code explaining files and functions, enough resources have been provided to Certn if they wish to implement the dashboard.

Closing Summary

Overall this project has provided team 3 with an awesome opportunity to experience software development within an agile team. It would not have been possible without guidance and communication from our excellent contacts at Certn, and we hope this report provides enough information to get the support tool up and running for actual use. We've all grown, learned new skills, and made new friends as a result of this project, and want to thank Certn, the SENG 499 teaching team, and everyone else involved.