CAPSTONE
FINAL PROJECT REPORT

TEAM: AACE

Ash Phillips,
Alex Warton,
Chap Callanan,
Elizabeth Robinson

1.0. Team Chapter

1.1. Introduction

1.1.1. Project Context, Goals, Scope

The Domain and Industry Sector

The domain for this project was an automatic web scraping application capable of scraping court databases and returning results to a web server and new result alerts to a given government sector email address. The desired outcome of this project was to search from a list of auditor names who work for the Department of Environment and Science and return an output of open cases they are involved in and their previous case history.

The Client's Organization and their Business Goals

The client for this project was the Department of Environment and Science (DES), specifically the contaminated land management team, who are required to oversee several third-party environmental auditing and consulting companies. As part of its supervisorial role, the department would like to know if an auditor becomes involved in a civil action relating to their consultation of a contaminated land site. Given there are currently 28 approved auditors in Queensland, it was not feasible for the contaminated land management team to manually search each auditor's details on the court's website to check if the auditor was being sued or not – especially considering this should be checked daily. Therefore, their main goal for this project was to receive an easy-to-use and self-explanatory web-scraping tool to automatically conduct searches of the court's web resources so the department could be notified when an auditor was involved in a civil action. This tool was constructed in the form of a web application that impacts how the department searches for auditors; using this application should save them time and resources that would have otherwise been used for this manual searching task.

Project Scope

The scope of the minimum viable product (MVP) included manual and automatic searches using the Queensland scraper and a functional auditor list where auditors could be added and deleted. These features were of the highest priority to the final artefact. The scope of the enhanced system included the addition of scrapers for other state's court databases, login functionality, and the ability to edit an auditor's details, disable automatic searches, and review cases from the automatic searches. These features were of lower priority than those in the MVP and were classed as "Should Haves" for the final artefact. The scope of the stretch system included the ability to view the list of auditors currently being searched, view an auditor's case history, filter manual search results based on states, and save the reviewer user's name to the case details when reviewed. These features were of the lowest priority to the final artefact and were classed as "Could Haves".

The Technical Context of the Application and its Inputs

This application relies on many technical aspects such as scraping data from websites using coding language attributes that allows this. Users are able to input certain items in which the program will react and insert certain aspects of this input into the application to be searched and data to be obtained. It also relies on organization infrastructure such as servers to be able to achieve this search periodically.

1.1.2. Project Outcomes and Success

Project Outcomes

The outcomes for this project were to deliver a fully functional automatic court website scraper that could notify the DES when new cases entered the system. The artefact would be produced in the form of a webserver that the DES would host on their systems. A DES user would need to be able to enter the details of currently employed auditors for the automatic search to find cases to display on the Dashboard page. These auditors would be inputted into the system database by the team so it would be setup for the DES. The auditors could be edited or removed whenever required, new auditors could also be added to the system.

The automatic search functionality was expanded on by the team to also include manual search functionality so a DES user could search, present on the Search page of the webserver. The name, organisation, or case number for a person of their choosing without having to input their information into the system for the automatic search to query on.

The enhanced outcomes for this webserver were to also include scrapers for the other states for each search to run on. These scrapers would provide the DES with a larger range to search for cases on their employed auditors, as most also operate in states outside Queensland. This enhanced system would also include the ability to disable the automatic search in the setting page if a user wished to. Review functionality would be implemented to the Dashboard page for the cases found by the automatic search. When cases were found they would be displayed under the Pending section. A DES user could then view the details of the case and mark it was reviewed, where the case would then be moved to the Reviewed section. Login functionality would be implemented to protect the server.

The stretch outcomes included a method to view the list of auditors currently in the system that were being searched through with the automatic search. These auditors would be displayed in a table on an Auditors page, where a user could also perform all the editing, removal, and addition actions on the database. An auditor's case history would also be viewable by viewing an auditor's details. Here, cases that had been located on an auditor would be displayed; pending or reviewed. User functionality would also be used to track activity on the webserver; if a user were to review a case their name would be recorded in the case details as being the one to review it. A filter would also be implemented on the Search page so users could search within select states to refine searches.

Project Success

The team was able to deliver the minimum viable product of the automatic court website scraper that notifies when new cases are found. We were also able to deliver on most of the enhanced and stretch goals present.

The automatic and manual searches were able to be made fully functional. They were able to scrape most court websites, except for Tasmania and Western Australia as their related court websites were unable to be scrapped due to Captcha blocks. While it was a goal to have all states covered by the searches, this goal was presented as an enhanced goal and therefore did not affect the outcomes of the expected artefact.

Small edits were made with respect to the goals of the project. There was a time when a sort feature was to be implemented on the search page, though this feature was scrapped due to it being the least important function of the artefact and time constraints.

All other goals and outcomes outlined for the artefact were completed successfully by the team.

1.2. Project Setup

1.2.1. Project Management Approach

What approach was used?

As a team, we collectively agreed that the increased flexibility, continuous improvement, and project predictability associated with an agile project management style would be best for our capstone project. Therefore, we followed an agile project management approach throughout the entire project planning, development, and implementation of our project. Within using an agile approach our team borrowed elements of the scrum framework, namely roles, stand-ups, sprints, and backlog refinement.

Why it was suitable?

This capstone project was the largest project any of our team members had ever taken on. Creating a fully functional polished website that looked both visually pleasing and had operational back-end web scrapers for multiple state's court website registries within a timeline was a daunting task. By using an agile approach with compartmentalised sprint and release plans we were able to break down the complex solution into smaller, individual tasks that could be delegated to specific team members depending on their skill set. An example of this was breaking the front-end website into pages but also page compartments, such as assigning a team member to work on a case card component while another team member worked on the page layout.

The agile approach also meant that we were able to iteratively incorporate our client's feedback into the project development of the website. Iterative client feedback this was important as Ashley, our client, had a loose idea of what he envisioned the solution to be. However, as we met with him at the end of every sprint Ashley became more aware of what he required in website and was able to communicate this wish to us. An example of where the iterative feedback came into play with the auditor feature specifying which state the auditor was approved in. Initially, this was a feature Ashley had requested but upon seeing it in use during one of our bi-weekly meetings he decided it was unnecessary to include in the final artefact because his department would be interested in any civil action regardless of the state.

How it was implemented:

Roles and responsibilities

We implemented an agile approach by identifying and assigning roles important to the successful execution of our website project. As an entity the Department of Environment and Science Contaminated Land Management Team was the main stakeholders as our court case web scrapping tool was designed for their benefit and with the team as the end users. The product owner was Ashley Seiler, he represented the Contaminated Land Management Team. Ashley worked with us in prioritising the product backlog (user stories), he also met with us at the end of every sprint so we could fill him in with progress reports and he provided feedback to us at each checkpoint. Chap was our project manager/scrum master, Alex was our business analyst, Elizabeth and Ash were developers. The roles are explained in more detail under Team Collaboration.

Project management processes

In alignment with an agile approach, we utilised an online dynamic task management tool called Trello to keep track of our sprints. During the planning stage we created cards for all the tasks outlined in our sprint plans. At the beginning of each sprint, we assigned the task cards to a team member. The team member was then responsible for updating the status of the card with either "In progress", "Finished" or "Stuck". During a few weeks when the team was were ahead of schedule tasks were marked with a "above and beyond" label. We also made use of the Trello board to keep track of general jobs along with meeting times with Ashley, Aaron (our tutor) and ourselves. *Appendix 6.2* is a screenshot exampling our team's use of the Trello board.

Git was the software development and version control tool we deployed to enable team members to create branches and work on different components of the same page. For example, sprint 2 Ashley was working on the dashboard case list, Alex was working on the case detail components that would be added to that dashboard list and Elizabeth was working on the auditor view list. This was all being developed as Chap was working on the back end setting up the first iteration of the Queensland automatic server. At the end of the sprint, we would all push our commits and then Chap merged all the branches together. *Appendix 6.3* shows a graph of our git repository mid-way through the development phase.

OneDrive was our main point of file sharing, this is where we kept our meeting minutes, research documents, product backlog spreadsheet, team agreements and any other necessary shared files. To maintain constant communication between the AACE team, we had a discord server. This tool was used daily, it was an informal way of catching up with other team members and asking questions. We had a discord voice call every two days as our stand-up scrum meeting. We would also have a longer discord call meeting later in the week to work through any issues a team member was having with a delegated task. We would usually utilise the screen sharing tool to work problem solve together.

Teams was the other communication tool, we used to keep in touch with Ashley. We had a scheduled video call with him every two weeks.

1.2.2. Client Expectations

As explained above Ashley was considered the product owner as he was representing the Contaminated Land Management Team of which he is the Team Leader. His responsibility was to give us the idea of what he wanted the product to achieve at the beginning of the planning stage, then he confirmed the prioritization of the product backlog AACE had created. A simplified version of the product backlog is outlined in the Product Agreement, which Ashley officially signed off on. From the beginning, our team was very clear in establishing which features were included in a minimal viable product as opposed to an enhanced or stretch system.

As we worked through phase 2, Ashley was responsible for giving us feedback after each sprint. Ashley was constructive at giving us achievable feedback, after our first sprint was completed, Ashley requested a list of what websites could be scrapped for cases and what the search parameters available for each website were. This meant he could have a better understanding of what our product could achieve for his team. As the client representative, Ashley also provided us with names to run tests on to which he expected to see results.

We gave Ashley our sprint and releases plans before development began so he was aware of what to expect for phase 2. During our fortnightly meetings, we notified him of our sprint progress. In sprint 4, the team was a week behind schedule because of difficulties associated with the non-Queensland scrapers. Ashley was aware we had incorporated a week of leeway into our timeline to allow for hold ups, so he wasn't worried about this delay.

Despite his best-efforts Ashley was unable to get his department's networking team involved in the project, thus we were limited to a few technical communications with the optimal deliverable format. However, we mitigated this limitation by communicating with Ashley throughout the whole progress and ensuring that our project was viable across multiple system platforms.

Upon delivering the product, Ashley was very impressed with the website. All program features were delivered, including the stretch and enhanced features. *Appendix 6.4* includes our meeting minutes with Ashley.

1.2.3. Team Collaboration

Positive Team Culture

AACE took a lot of pride in how well we worked together and supported each other throughout the project. Chap and Ashley had worked in a group with Elizabeth prior to this project and have an informal expectation of the team culture. Alex and Elizabeth knew in other prior to the project in a social context. In conclusion, we all had some form of connection and thus felt accountable for each other. This ignited the positive team culture which continued throughout the project as we agreed on team expectations in our first meeting in Semester 1. As discussed, above we met regularly and used an informal Discord chat server to ensure we felt comfortable asking questions and keeping in touch. Our discord chat and calls maintained a judgement free zone. Another method to ensure positive team culture was ensuring we were all aware of each other's strengths and abilities, ensuring our expectations of each other were level and risk of frustration and fall out was reduced.

Roles and responsibilities

Chap was our project manager/scrum master, he had the most experience in web-development, back-end programming. As the official team leader, Chap acted as a coach to the AACE team without controlling or overpowering the team. Alex was a team member that utilised his business analysis skills to lead the way on the creation of the product backlog. Thus, Alex was extremely in touch with exactly what was required and its prioritization during the development. Ashley (Philips) is a skilled developer with experience in HTML and CSS, Ashley also was a key contributor to outlining the user stories in phase one. Her attention to detail meant that she took responsibility in ensuring the team was hitting check points with the quality we expected. Elizabeth's role in the team was the legal consultant due to the other half of her dual degree. While also having the role has a developer, Elizabeth was responsible for any legal questions had while navigating the court websites and nature of civil actions.

Team agreement

Our team all agreed to the following expectations:

- Attend all tutor meetings on time
- Attend all weekly meetings
- Notify everyone of absence prior to the tutorials/meetings
- Complete assigned tasks/homework prior to attending
- Reach out to the team if there are any issues with the assignment/tasks
- Provide constructive feedback to others and help everyone
- Commitment to their assigned tasks and agreed deadlines
- Reach out for help if needed
- Tasks will be equally distributed
- Regularly update status on Trello
- Keep on track with set plans and due dates for tasks

A full version of our team agreement can be found in *Appendix 6.1*.

1.2.4. Communication plan

The AACE team agreement also outlined our communications expectations. *Table 1* below shows a summary of our communications plan.

What	Who	How	When	Why	Responsible
Project Status	Product owner, dev team	Teams Video Calls	Fortnightly	To inform product owner of project status and receive feedback	Team leader
Sprint task delegation	Dev team	Discord voice channel	Fortnightly	Delegate the tasks at the beginning of each sprint	All present
Problem Solving Workshops	Dev team	Discord voice channel using screenshare	Weekly	To combine knowledge and skills to overcome any issues faced.	All present
Team Check-Ins	Dev team	Discord Chat	At any time	To keep informal communications active between meetings	Any team member online
Team Progress	Dev team	Scrum	Every 2nd day	Check ins with each team member	All present

Table 1: Communication Plan

We agreed from the beginning of the project that all our communication would take place online as the team was geographically distributed. This worked well for our product owner as he was based in Bundaberg and would not be able to meet face to face.

The frequency at which our team touched base with each other meant that we the project was always progressing. Development issues did not linger for too long as they were worked through as a group, and no one was stuck on a problem by themselves. For example, when Elizabeth was having issues connecting to the server due to her operating system. The rest of the team combined their knowledge from other experience and helped her set up an online cluster server. We also believed by having informal means of communication such as a discord chat channel, the members would all feel comfortable asking for assistance without any feeling of judgment.

1.3. Project Plan and Risk

1.3.1. Project planning and progress

Planning and Delivery Timelines for Phase 2

The delivery timeline followed throughout Phase 2 can be seen in *Table 2* below. This timeline outlines each progress step the team completed to create the final artefact deliverable. The scheduled testing was included in each sprint. The deadlines for sprints 4 and 5 had to be extended due to risks involving time underestimation (further explained in 1.3.2. Risk Management).

Week											
1	2	3	4	5	6	7	8	9	10	11	12
Sprint 1		Sprint 2		Sprint 3		Sprint 4		Sprint 5			
Release 1 (05/08/22 - 26/08/22)				Release 2 (26/08/22 - 30/09/22)				Release 3 (30/09/22 - 21/10/22)			

Table 2: Delivery Timeline (Releases 1-3)

Release and Sprit Plans

For the development of the artefact, three releases were completed. The complete release and sprint plans can be seen in *Appendix 6.5*; we have provided only a summary of each release and their deliverables here due to the complete plans' length (the deliverables have also been further outlined in the release plan). In addition, each version of the User Stories can be seen in *Appendix 6.6*.

Release 1 was completed on the 26th of August Semester 2. It had a total number of story points of 15 and covered Sprints 1 and 2. This release focused on creating the back end of the data scrapping tool. Here, the team completed the setup for the client-side and server-side structures. The Queensland web scraper was completed and implemented into manual search and automatic search features. The deliverables of this releases included a functional dashboard page that displayed the cases found through the automatic search on auditors entered manually into the database and a manual search page were a user could enter an auditor's name, organisation, or case number to search for cases present in the Queensland court database.

Release 2 was completed on the 30th of September with total story points of 4 and covered Sprints 3 and 4. This release was focused on creating an auditor list and implementing scrapers from other states. Here, the team was able to deliver a fully functional; auditor page on the webserver where a user could view the list of auditors present in our database, as well as edit and delete them and view their previous case history. The team was also able to write scrapers for all state court websites, excluding Tasmania and Western Australia due to free form data structures and Captcha blocks respectively. These scrapers were implemented on the dashboard and manual search pages.

Release 3 was completed on the 21st of October with total story points of 15 and covered Sprint 5. This release focused on final polish of the artefact and the login functionality. Here, the team ensured all web client pages matched our proposed designed from Phase 1, which can be found in *Appendix 6.7*. The login was then implemented to use one department wide password that the networking team will oversee and, once the password has been entered, user profiles that the users can create themselves.

Following the completion of the above releases when the artefact had been finalised the team met with the industry supervisor, Ashley Seiler, on the 28th of October to conduct the product handover. In this meeting, the team ran a complete demonstration of the fully polished artefact to ensure everything met Ashley's standards. The link to the One Drive folder where the artefact Zip folder, technical documentation, demonstration videos, and all other deliverables were located was shared by email.

Regarding testing, each feature within each Sprint has a dedicated task for writing test cases and implementing these tests with an acceptance test. These tests were completed at the finalisation of each feature implementation to ensure that thorough testing was undertaken throughout the project. A minimum of 4 hours was dedicated to testing every individual feature to ensure full coverage.

Progress Tracking

The team tracked the progress of the project using Trello. Each sprint was recreated in Trello as lists, with all tasks required in the sprint allocated equally to the team. We created labels for different stages of progress (i.e., "In Progress", "Finished", and "Stuck") so we could all view each member's progress and how close/far we were from competing each sprint on time.

1.3.2. Risk Management

Risk Identification

The risks identified in our Risk Register from Phase 1 were based on clarity, scope, operation, and time, as can be seen in *Table 3* below. The risks identified through the artefact creation process were heavily based on operation and time. When completing the sprints, it was identified that some features time allocation had been both under and overestimated. These incorrect estimations occurred in sprints 4 and 5 regarding scraper creation and webserver polish respectively. It was identified that sprint 4 would need more time allocated to it for all the scrapers to be completed by the team, which would then impact the time allocation for sprint 5. Operation risks that had been identified by the team was unclear infrastructure as our industry partner was never able to get into contact with their networking team to confirm how our webserver would be hosted. Our project was not impacted by COVID-19 as our industry partner worked out of state and all communication was held remotely.

Risk Identifier	Risk Category	Risk Title	Risk Description
1	Clarity	Vague Project Scope	Not identifying clear enough goals for the scope of the project
2	Clarity	Stakeholder Miscommunication	Not communicating effectively with stakeholders, resulting in unsuitable deliverables.
3	Clarity	Unclear Deadlines	Not communicating effectively regarding timelines
4	Scope	Scope Creep	Not properly outlining the scope means certain functions may be built that are outside the scope of the project
5	Scope	Approval	Team members, who may be unaware of the scope, may add certain functions without the team realizing it, resulting in overuse of resources.
6	Operation	Unsuitable Infrastructure	Not knowing the limitations of organizations infrastructure, such as servers, can cause damage and issues to the other business operations
7	Operation	Personal Challenges	Implementing certain projects can affect the job performance and outcomes of employees already present at the organization and may result in lower work performance
8	Time	Underestimation	Underestimation of how long tasks take to complete and how long the application takes to develop
9	Time	Overestimation	Overestimation of how long tasks take to complete and how long the application takes to develop

Table 3: Risks associated with the project.

Risk Analysis

These above risks were analysed by the team internally during team meetings and externally with our industry partner to ensure the best approach was taken. The risk of underestimation held the highest impact as it heavily affected the project completion deadline. The risk of operation due to the unreachable network team held less impact as this risk has been controlled from the beginning of production and everyone was aware that it would be a possibility.

Risk Response

In response to the underestimation risk present in sprint 4, the deadline was extended by one week to allow for more time to complete the required features. The deadline for sprint 5 was also extended due to this. These changes did not impact the final deadline for project handover as when crafting the project handover agreement excess time was allocated between when the project was expected to be completed and when it had to be handed over to ensure there were no issues in the final product quality. The risk of operation was accepted as the product was able to be handed off as a Zip folder with deployment instructions so the DES could use the web application without need for the networking team’s sign off.

1.4. Artefact Description

1.4.1. Functionality

The functionality of the program went through several iterations, however, after lengthy discussions with the client, we settled on a basis of user stores that outline our must-haves, should-haves, and could-haves. These user stories were often mentioned and viewed each sprint as we outlined our exact goals and requirements for each week.

As shown in *Appendix 6.8*, our must-haves, labelled in red for urgency, revolved around our minimum viable product. For example, the software was required to perform automatic searches through our court databases and thus we have multiple user stories that revolve around this including enabling the program to make searches, receiving search results and more. It also mentions the ability remove and add search parameters as well as having our program be accessible.

Our should-haves expand the functionality a bit and allow for more search manipulation. We aimed to add some specificity to the searches and therefore allowed the user to; manually search, change auditor details, disable search parameters, and search specific states when manually searching. We also outline the automatic scraping of search parameters as well as login potential for security and data recording. Perhaps the most significant of the should-haves was the ability to review cases which would remove them from the dashboard.

The could-have areas of our user stories were items that were not necessary for base function but did provide a level of convenience for the client that wouldn't usually be available. These user stories revolve more around efficient data storage and viewing, allowing the user to view specific cases related to each auditor and which users reviewed which cases.

1.4.2. Architecture

The architecture of the program is like several webapps and websites in that the user interacts with our data via a frontend user interface. This is where the user can login and make requests to the local database. It also allows users to view this data in a formatted way and change settings of the application. There is no admin interface as any changes that need to be made will be made directly into the source code although this is unlikely.

The original goal was to have the program hosted on a local server within our clients existing system architecture. However, as their network team was unavailable for discussion, this never occurred. In that case, we assume that the program will be hosted locally on the user's computer. The server also makes requests to email programs and sends notifications to users occasionally.

System Architecture Diagram

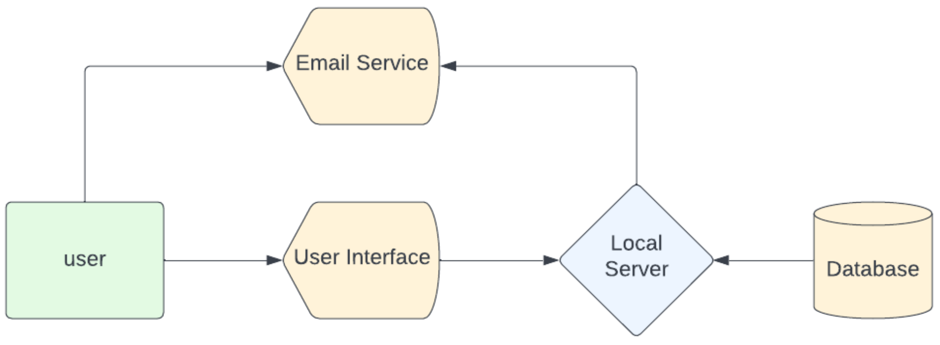


Figure 1: System architecture diagram

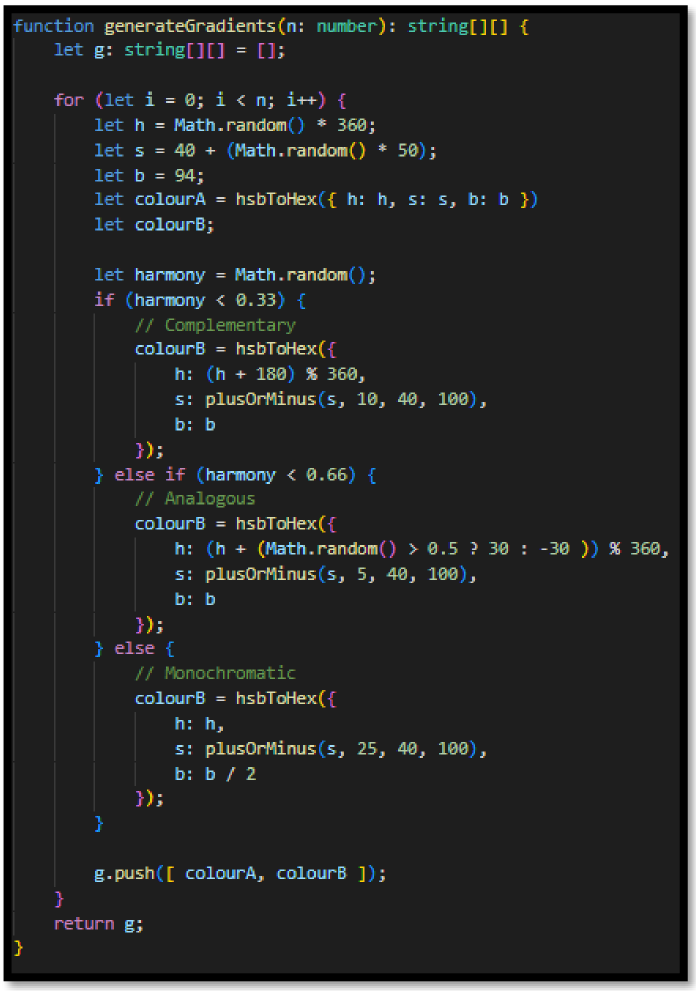
1.4.3. Technical Description

To best describe the technical aspect of the program, we will move through it as a user would starting with the login and moving towards the important areas and problems that needed to be solved.

Login

The login page and authentication are relatively simple. As shown in our presentation video, the user's login with a set department password and can control the different department profiles. Users can create profiles, and this then determines who reviewed which cases for maximum transparency. Although not an important aspect of the system, the code to provide different gradients and colours is unique and adds some individuality to the profiles as shown below.

Gradient Generator Code

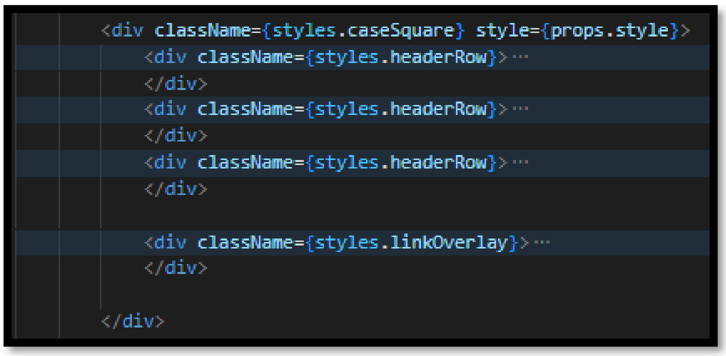


```
function generateGradients(n: number): string[][] {
    let g: string[][] = [];

    for (let i = 0; i < n; i++) {
        let h = Math.random() * 360;
        let s = 40 + (Math.random() * 50);
        let b = 94;
        let colourA = hsbToHex({ h: h, s: s, b: b })
        let colourB;

        let harmony = Math.random();
        if (harmony < 0.33) {
            // Complementary
            colourB = hsbToHex({
                h: (h + 180) % 360,
                s: plusOrMinus(s, 10, 40, 100),
                b: b
            });
        } else if (harmony < 0.66) {
            // Analogous
            colourB = hsbToHex({
                h: (h + (Math.random() > 0.5 ? 30 : -30 )) % 360,
                s: plusOrMinus(s, 5, 40, 100),
                b: b
            });
        } else {
            // Monochromatic
            colourB = hsbToHex({
                h: h,
                s: plusOrMinus(s, 25, 40, 100),
                b: b / 2
            });
        }

        g.push([ colourA, colourB ]);
    }
    return g;
}
```

Dashboard

Once logged in, the user is greeted with a dashboard page. Perhaps the most crucial area of the program as this where cases are first shown and the path to review cases begins. This was done via a dashboard page, court case display component and a courtCase object that was then filled via our scrapers. The case component was then called on the dashboard page and provided with an infinite scroll feature to keep the dashboard constantly updating without having to call all the items from the database at once.

Case Component (Minimised)

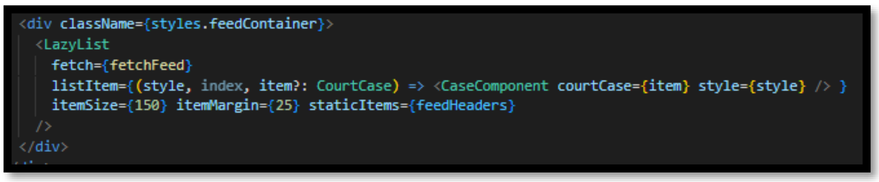


```
<div className={styles.caseSquare} style={props.style}>
    <div className={styles.headerRow}>···
    </div>
    <div className={styles.headerRow}>···
    </div>
    <div className={styles.headerRow}>···
    </div>

    <div className={styles.linkOverlay}>···
    </div>

</div>
```

Please note this section of the code has been shrunk to not take up too much space.

This case component code consisted of relevant case details such as the case title, location and parties involved. This was filled via the information scraped from our court scrapers.

Dashboard Case Components Code

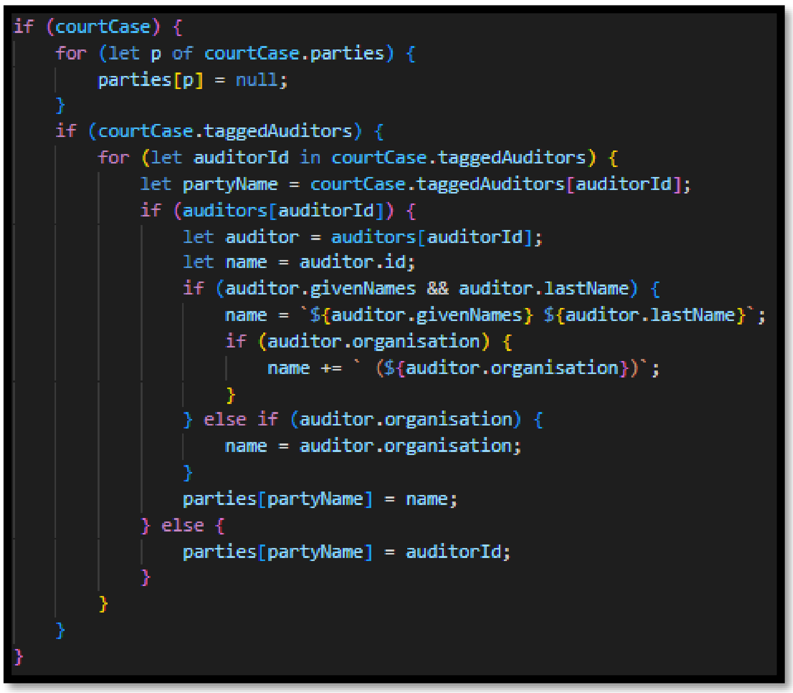


```
<div className={styles.feedContainer}>
  <LazyList
    fetch={fetchFeed}
    listItem={(style, index, item?: CourtCase) => <CaseComponent courtCase={item} style={style} /> }
    itemSize={150} itemMargin={25} staticItems={feedHeaders}
  />
</div>
```

Case Details

The case details page displays different details not shown on the case component page. The biggest issue with this page was displaying the list of parties and matching those parties to our known list of auditors. This allowed the user to know which parties were relevant to them. As shown below, this was done using a for loop and matching the party details to auditor names via the courtCase object.

Parties For Loop



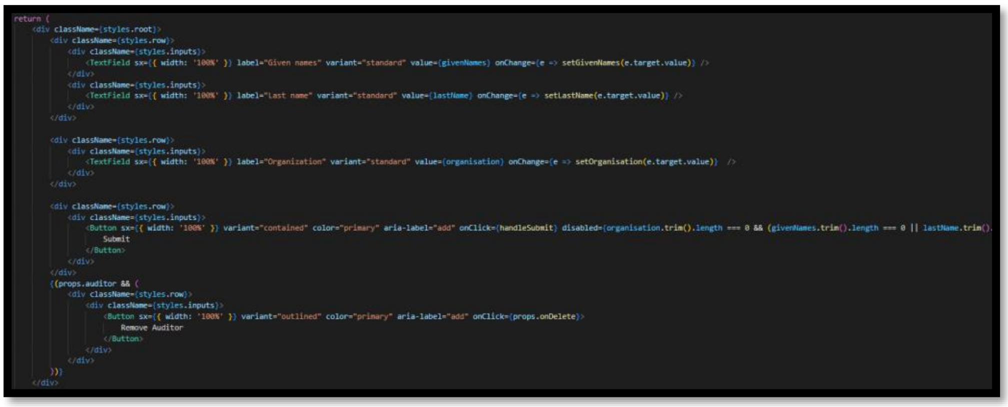
```
if (courtCase) {
    for (let p of courtCase.parties) {
        parties[p] = null;
    }
    if (courtCase.taggedAuditors) {
        for (let auditorId in courtCase.taggedAuditors) {
            let partyName = courtCase.taggedAuditors[auditorId];
            if (auditors[auditorId]) {
                let auditor = auditors[auditorId];
                let name = auditor.id;
                if (auditor.givenNames && auditor.lastName) {
                    name = `${auditor.givenNames} ${auditor.lastName}`;
                    if (auditor.organisation) {
                        name += ` (${auditor.organisation})`;
                    }
                } else if (auditor.organisation) {
                    name = auditor.organisation;
                }
                parties[partyName] = name;
            } else {
                parties[partyName] = auditorId;
            }
        }
    }
}
```

Auditors List

Shown on the auditor's page, it was crucial that we allowed users to view and edit the list of auditors who acted as our search parameters for each daily search. It was also nice to view each auditor's court cases that had previously been found and see who these cases were reviewed by.

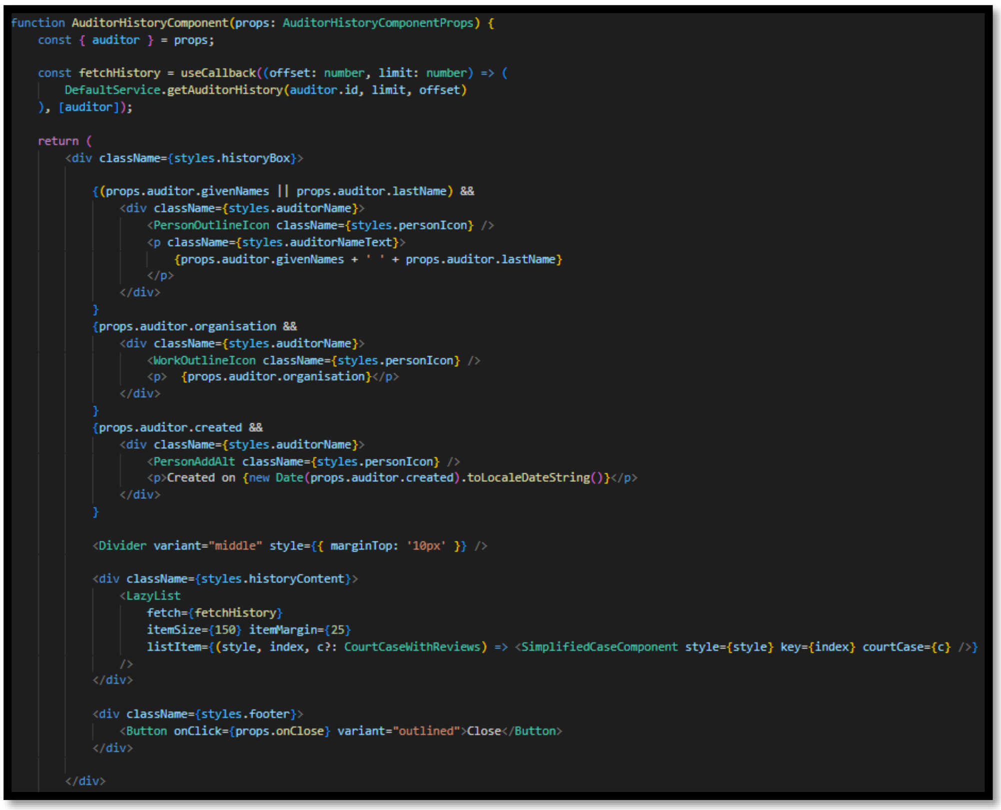
Edit/Delete Auditor Code

The auditor edit/delete code is too large to show, although it simply uses basic form principles to allow the user to update current auditors, add new orders and delete them. A small snippet from the code which outlines the form components.



```
return (
    <div className={styles.root}>
        <div className={styles.row}>
            <div className={styles.inputs}>
                <TextField sx={{ width: '100%' }} label="Given names" variant="standard" value={givenNames} onChange={e => setGivenNames(e.target.value)} />
            </div>
            <div className={styles.inputs}>
                <TextField sx={{ width: '100%' }} label="Last name" variant="standard" value={lastName} onChange={e => setLastName(e.target.value)} />
            </div>
        </div>

        <div className={styles.row}>
            <div className={styles.inputs}>
                <TextField sx={{ width: '100%' }} label="Organization" variant="standard" value={organisation} onChange={e => setOrganisation(e.target.value)}  />
            </div>
        </div>

        <div className={styles.row}>
            <div className={styles.inputs}>
                <Button sx={{ width: '100%' }} variant="contained" color="primary" aria-label="add" onClick={handleSubmit} disabled={organisation.trim().length === 0 && (givenNames.trim().length === 0 || lastName.trim().
                    Submit
                </Button>
            </div>
        </div>
        {(props.auditor && (
            <div className={styles.row}>
                <div className={styles.inputs}>
                    <Button sx={{ width: '100%' }} variant="outlined" color="primary" aria-label="add" onClick={props.onDelete}>
                        Remove Auditor
                    </Button>
                </div>
            </div>
        ))}
    </div>
```

Auditor History Code



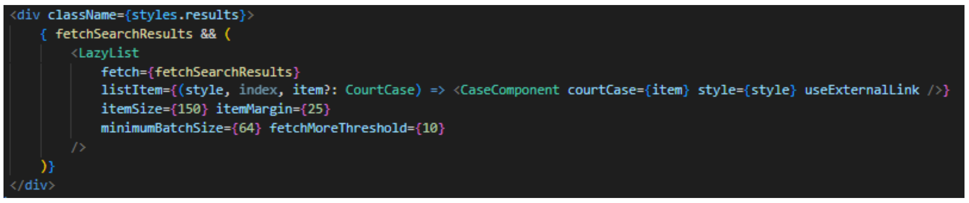
```
function AuditorHistoryComponent(props: AuditorHistoryComponentProps) {
    const { auditor } = props;

    const fetchHistory = useCallback((offset: number, limit: number) => (
        DefaultService.getAuditorHistory(auditor.id, limit, offset)
    ), [auditor]);

    return (
        <div className={styles.historyBox}>

            {(props.auditor.givenNames || props.auditor.lastName) &&
                <div className={styles.auditorName}>
                    <PersonOutlineIcon className={styles.personIcon} />
                    <p className={styles.auditorNameText}>
                        {props.auditor.givenNames + ' ' + props.auditor.lastName}
                    </p>
                </div>
            }
            {props.auditor.organisation &&
                <div className={styles.auditorName}>
                    <WorkOutlineIcon className={styles.personIcon} />
                    <p>  {props.auditor.organisation}</p>
                </div>
            }
            {props.auditor.created &&
                <div className={styles.auditorName}>
                    <PersonAddAlt className={styles.personIcon} />
                    <p>Created on {new Date(props.auditor.created).toLocaleDateString()}</p>
                </div>
            }

            <Divider variant="middle" style={{ marginTop: '10px' }} />

            <div className={styles.historyContent}>
                <LazyList
                    fetch={fetchHistory}
                    itemSize={150} itemMargin={25}
                    listItem={(style, index, c?: CourtCaseWithReviews) => <SimplifiedCaseComponent style={style} key={index} courtCase={c} />}
                />
            </div>

            <div className={styles.footer}>
                <Button onClick={props.onClose} variant="outlined">Close</Button>
            </div>

        </div>
```

This code snippet is the function used to grab the auditor history and display it. The Lazy List used the fetchHistory function and displays all court case components with the relevant auditor ID.

Manual Search Page

The manual search page is used for when the user wants to make a search outside the scope of the auditors list. This manual search call each of our scrapers and runs the search query and returns the results as a list of case components, like the dashboard page.

Search Results Code



```
<div className={styles.results}>
    { fetchSearchResults && (
        <LazyList
            fetch={fetchSearchResults}
            listItem={(style, index, item?: CourtCase) => <CaseComponent courtCase={item} style={style} useExternalLink />}
            itemSize={150} itemMargin={25}
            minimumBatchSize={64} fetchMoreThreshold={10}
        />
    )}
</div>
```

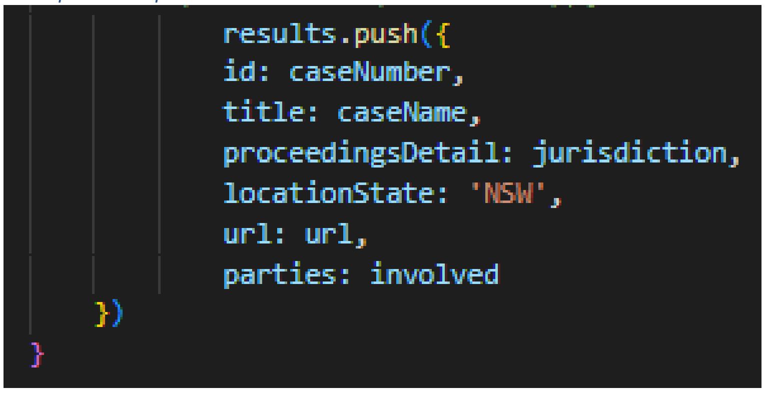
The above code snippet uses the fetchSearchResults function and the Lazy List to structure the case components.

Scrapers

Easily the backbone of the program and where all our data is gathered are our scraper programs. We have 7 out of 8 court websites that can be scraped as their functions are called whenever a search is made whether automatically or manual.

The scraper output must have certain required items according to our courtCase object. Below, I have provided an example code of the NSW scraper to show this data is formatted.

Scraper Output Code



```
                results.push({
                id: caseNumber,
                title: caseName,
                proceedingsDetail: jurisdiction,
                locationState: 'NSW',
                url: url,
                parties: involved
        })
    }
```

All scrapers follow a similar output to the screenshot above as if our results did not match the courtCase object, then an error would occur. This allowed for consistency across the program.

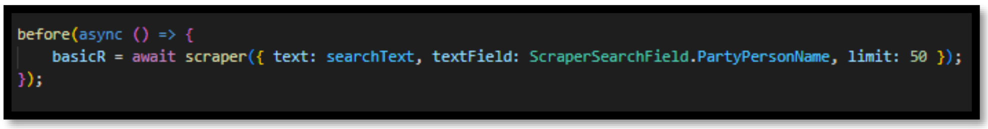
1.4.4. Quality and Metrics

The main unit testing came from having to test our scrapers to ensure they delivered the results in the correct format. These tests involved several different criteria having to be met and where setup against each of our 7 scrapers.

Scraper Background

For this testing to make sense, you need to know that when calling a scraper function a few items had to be provided. Firstly, the search text which was the actual search query. The text field, which is what is being searched such as the person or organization name. Thirdly, the limit which is how many results could be shown. Lastly, the offset which was how many results the search would skip over before it started displaying.

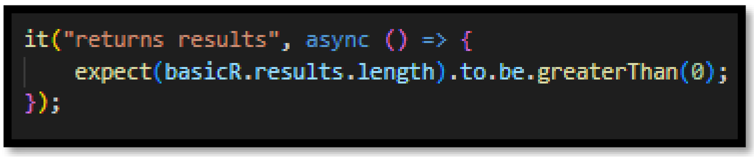
Function Call



```
before(async () => {
    basicR = await scraper({ text: searchText, textField: ScraperSearchField.PartyPersonName, limit: 50 });
});
```

This is the way the scraper test function is called. It grabs the search Text which was used as ‘a’ and provided a limit of 50 and the default offset value.

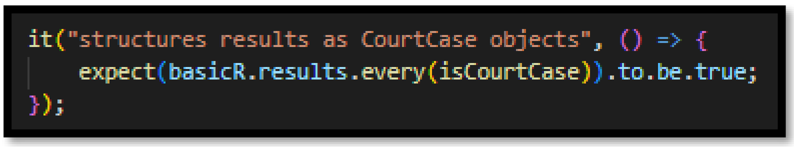
Results Total



```
it("returns results", async () => {
    expect(basicR.results.length).to.be.greaterThan(0);
});
```

This test made sure that the total results was greater than zero.

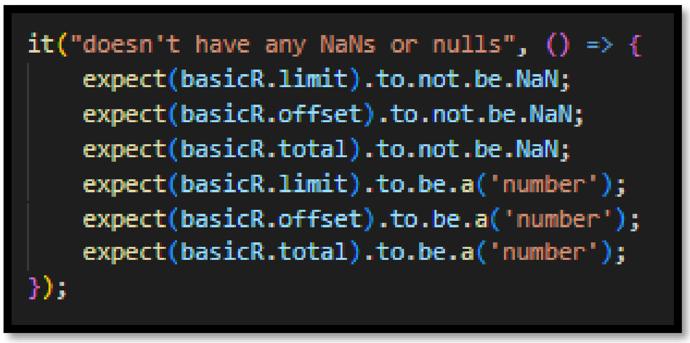
Court Case Structure



```
it("structures results as CourtCase objects", () => {
    expect(basicR.results.every(isCourtCase)).to.be.true;
});
```

This test ensured each item of the result was structured as a Court Case object,

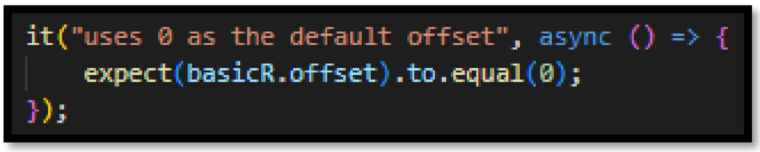
Null Test



```
it("doesn't have any NaNs or nulls", () => {
    expect(basicR.limit).to.not.be.NaN;
    expect(basicR.offset).to.not.be.NaN;
    expect(basicR.total).to.not.be.NaN;
    expect(basicR.limit).to.be.a('number');
    expect(basicR.offset).to.be.a('number');
    expect(basicR.total).to.be.a('number');
});
```

This test ensured that each item returned something of value and wasn’t empty. An empty item would mean a case component with no information.

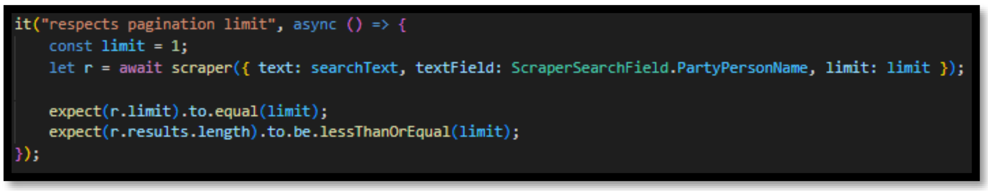
Offset Check



```
it("uses 0 as the default offset", async () => {
    expect(basicR.offset).to.equal(0);
});
```

This test ensures that the default offset is 0 as if the user doesn’t provide an offset, no results should be passed over.

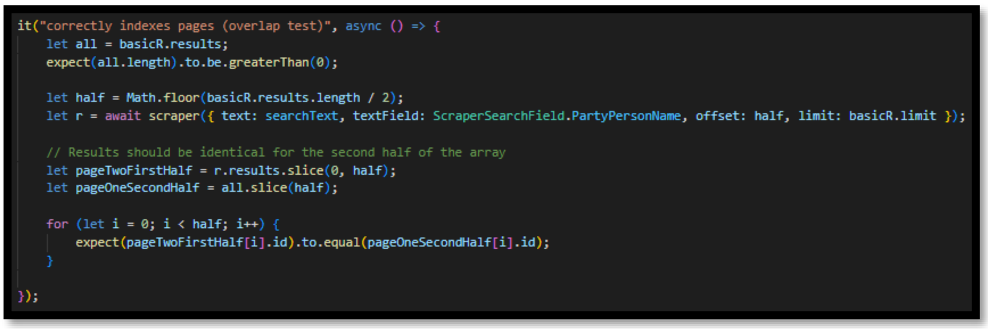
Pagination Limit



```
it("respects pagination limit", async () => {
    const limit = 1;
    let r = await scraper({ text: searchText, textField: ScraperSearchField.PartyPersonName, limit: limit });

    expect(r.limit).to.equal(limit);
    expect(r.results.length).to.be.lessThanOrEqual(limit);
});
```

This test ensures that the pagination limit is met.

Page Index



```
it("correctly indexes pages (overlap test)", async () => {
    let all = basicR.results;
    expect(all.length).to.be.greaterThan(0);

    let half = Math.floor(basicR.results.length / 2);
    let r = await scraper({ text: searchText, textField: ScraperSearchField.PartyPersonName, offset: half, limit: basicR.limit });

    // Results should be identical for the second half of the array
    let pageTwoFirstHalf = r.results.slice(0, half);
    let pageOneSecondHalf = all.slice(half);

    for (let i = 0; i < half; i++) {
        expect(pageTwoFirstHalf[i].id).to.equal(pageOneSecondHalf[i].id);
    }

});
```

This test ensures that the scraper indexes and does not overlap.

2.0. Individual chapter: Chap Callanan

2.1. Project Setup

2.1.1. Project Management Approach

As the project manager, I played a large role both the in the initial set up and continual maintenance of our code repository and project management tooling. To ensure all team members could work concurrently without stepping on each other's work, it was critical that our workflow was set up correctly from the beginning.

As mentioned, we utilised Git as our version control system. While this is essentially a given for any software project in the modern workplace, in its flexibility Git still leaves many workflow-related decisions up to individual teams. I elected to treat the 'master' branch as a semi-read-only branch. Although we didn't provide the client multiple releases as we developed, I instructed the team to treat master as though it were a series of releases. That is, any code pushed to master had to successfully compile, pass all our automated tests, and include only features that were fully completed.

To ensure these standards were upheld, the repository was configured to required team members to develop new features in separate branches, and then submit pull requests when they were finished. I could then checkout each member's code, review it, and allow it to be merged into the master branch.

Our repository was hosted on GitHub, so we were able to link it with our Trello board using the GitHub integration. This allowed team members to attach their pull requests to a task card in Trello, which made the job of tracking progress and reviewing code easy.

2.1.2. Client Expectations

I was the primary point of contact for our industry partner and was responsible for relaying information back and forth between them and our team. I also led our fortnightly client and tutor meetings, where I was responsible for demonstrating our progress and answering stakeholder questions.

A large part of running effective and efficient meetings was the team preparation we undertook beforehand. Prior to each stakeholder meeting (be it tutor or industry), we held a "pre-meeting meeting" in which I outlined the key topics I thought would be discussed in the meeting and worked with the team members to ensure they had any resources they required and were confident in presenting a portion of the meeting. This fostered team-wide engagement during meetings which was commented on by both our industry partner and tutor.

Following our virtual meetings with our industry partner, we immediately convened to review the minutes taken by Ash. In these short discussions, we transferred any actionable requests from the industry partner into tasks on our team Trello board.

2.1.3. Team Collaboration

Our positive team culture was established via consistent team meetings in which any problems could be addressed quickly and without stress. Our schedule generally involved an early-week meeting in which tasks were allocated, a mid-week meeting to check progress and a final meeting at the end of the week (prior to our industry/tutor meeting) to review. I attended each meeting and, if I was to be unavailable, ensured the team was provided with adequate resources to conduct the meeting in my absence.

2.1.4. Communication
I sent emails to our industry partner regularly that included meeting schedules and agendas. Other team members were CC'ed into every conversation.

2.2. Project Plan and Risk
2.2.1. Project planning and progress
Along with working with the rest of the team in generally building out our prioritised requirements list, as project lead, I was also responsible for presenting this document to our industry partner. I explained each item to the industry partner and liaised with them to ensure they were happy with the features we could provide and the priorities we assigned them.

2.2.2. Risk Management
One of the key risks I planned for and mitigated as the project lead was the possibility of the team having to continue sprints in the event of my absence due to illness or other unexpected complications. Whenever we discussed key objectives or dates for an upcoming sprint or meeting, I ensured that the information was documented in written form in our shared OneNote. This planning proved useful when I had to miss a couple of industry partner meetings, as the team was easily able to run the meeting themselves from our notes.

2.3. Project Experience
An interesting challenge faced in this project was the need to develop a series of web scrapers for seven different websites. Creating web scrapers is rather niche activity and was one that only I had completed previously. We knew from the outset of the project that this was the case. As such, I scheduled with the team a series of informal tutorial sessions in which I would explain the process and provide extra assistance with their scraper-related tasks. I also planned to complete my scraper first, such that it could serve as a basis for others to work from and I could iron out any related issues in the larger project first.

Unfortunately, our scraper sprint ran approximately 1 week over time. This did not affect our final deliverable time, as we had planned a "buffer week" into the delivery date from the beginning. However, I would have preferred to stay on track and leave the buffer week for any unforeseen emergencies.

In retrospect, I believe the steps taken to educate the team on scrapers were all worthwhile, but perhaps not enough on their own. I underestimated the amount of time it would take for the team to acquire the skills required and complete the actual tasks. It would have been better to allocate more time for this task. Additionally, our sprint plan included all our scrapers in one sprint, as we were aiming to logically group related tasks together. Given the difficulty we knew we would encounter in this period, I believe it would have been more appropriate to stagger the scrapers across sprints, allowing me to spend more time on each team member.

Another issue related to planning our delivery process was the lack of communication with the DES networking team. Our industry partner tried to arrange a meeting with them, but unfortunately was unable to do so. As such, we were unable to confirm the exact hardware/software stack the team was using to host their existing web applications. We would have liked to integrate with the team's existing environment, but instead had to deliver the application in a generic package. To assist the networking team with deployment, we ensured the deliverable included detailed descriptions of the project and full instructions for a typical deployment.

3.0. Individual chapter: Ashley Phillips

3.1. Project Setup

3.1.1. Project Management Approach

For the setup of the project, I participated in all team meetings held. I contributed to the creation of the Trello tasks where we listed each sprint and their corresponding features. When we were in the process of sprint completion, I would ensure I labelled my assigned tasks as “In Progress” and “Completed” on the Trello when required. I also participated in the team meeting, run by Chap, where we all set up our GitHub repository on our personal PCs.

3.1.2. Client Expectations

To prepare for our fortnightly client meetings with our project supervisor, I would collate all the work I had completed over the course of the previous sprint, as well as organise the Trello be marking any of my assigned cards with their appropriate tags (E.g., completed, working on, etc.). In these meetings, I participated by taking notes throughout the meetings. We would outline our progress to our client, similar to how we presented in our tutor meetings. I would explain to the client what I had completed over the course of the sprint and explain how each section worked in relation to user capabilities. When direct questions were asked, I answered honestly and descriptively to help our client understand.

3.1.3. Team Collaboration

I participated in all team meetings whenever possible, notifying the team when unable to make it and working with everyone to find suitable times when required to reschedule. We communicated regularly through our Discord server; whenever I needed help from anyone in the team, I would ask on the Discord via text chat and we would enter a voice call together if needed. I ensured to always have my notifications on for our server so if my help was required on anything I could respond right away and help when able.

3.1.4. Communication

I ensured I attended and participated in all meetings scheduled with the team, client, and tutor. In the team meetings, I presented my current progress and asked for guidance when I required help. I worked with the team to schedule the next meeting for a suitable time slot for everybody. In the client and tutor meetings, I would present my progress and the finalised features for the current sprint, ensuring everything I showcased was understood and sufficient. To ensure I was able to communicate professionally and effectively with both parties I would script out what I needed to cover in the meeting; what I completed for the previous sprint, what I was working on at the time of the meeting, and any other questions or information I had for them. I did not communicate via email with either the client or our tutor. The team’s group chat was informal and was mostly used to schedule meetings, get help on features we were struggling with, and allocation of tasks.

3.2. Project Plan and Risk

3.2.1. Project planning and progress

As a programmer for the group, I was allocated coding tasks that matched my skill level. When allocating tasks, I volunteered for tasks I knew I would be able to complete within the time frame. When the tasks were more challenging, I would still volunteer to work on them and would communication to the team any issues or challenges I faced during the completion process. For the overall project planning and progress, I participated with the rest of the team in our weekly team meetings and would indicate my progress on the Trello.

3.2.2. Risk Management

In my allocated tasks, I was able to identify if we had underestimated the amount of time a task would take and voiced my concern with the team. From there we deliberated on how we would handle the risk together. When I found that I was unable to complete a task I would communicate this to the group immediately as to not waste time. When others would bring up their concerns around risks, I worked with the team to remedy them, helping take on the task when required. With the team, I actively monitored and mitigated risks as they developed.

3.3. Project Experience

3.3.1. Learning and Using a New Programming Environment

Situation and Task

For creation of the artefact, we used React with typescript and NodeJS, all of which I had never worked with before. My previous experience with programming languages required for this project was only HTML and CSS. I had to learn how to use React and code typescript to program the artefact. As this was the case for all team members, excluding Chap, he directed us to an introductory tutorial where we could learn more about what the artefact would require.

Action

During the break between phases 1 and 2 I completed the tutorial sent by Chap. When it came to applying this knowledge to writing code for our product it was rather challenging. I noticed gaps in my knowledge early on where I would then research to try and gain more understanding. When required, Chap would lead tutorial meetings for us to help us with our issues. I spent a lot of time writing and debugging my code to further enhance my knowledge, often doing more work than required to then cut back to a simpler method so I could fully understand the process.

Result and Learning

Over the course of the project my skills in React and typescript became much greater, and I was able to compete tasks easily without much guidance. I was able to code using typescript as I would have been able to code in a language, I was familiar in. If I were to do a project like this again, I would ensure I gave myself enough time to learn the new environment and would take more help from those offering it, when possible, as I wasted a lot of time researching simple issues when Chap was willing to help. I will be able to apply the skills in React, typescript, and NodeJS I gained over this project in my future career and development projects.

3.3.2. Creating a Web/CSV Scraper

Situation and Task

As above when learning a new language and environment, I had never created a web scraper before. I was assigned the Northern Territory court database to scrape. Unlike the others, my database utilised a csv file that I could use to retrieve the required information, which was significantly easier than what they had to do. There was the option to scrape the Northern Territory web version of this csv but with own time constraints we decided parsing the csv file into our application and scraping that would be more effective.

Action

Scraping the csv file was less challenging than it would have been to scrape the web page as I had more experience working with csv files. To retrieve the information required I had to parse the headers of the important data columns into a new array variable that would hold all the data within it. From there, I was able to transform the data format into the Court Case object we required.

Result and Learning
By undergoing this process, I was able to simply scrape the Northern Territory database for the required information. While I would have enjoyed scraping the web page so I could learn how to do it along with the others, I think the choice to use the csv file was better as I encountered less issues with it that the others. In the future, I do wish to analyse the code of my team members to understand howe the web page scraping worked so I can do it in future projects either in my career or personal endeavours.

3.3.3. Issues with Infinitely Scrolling Data

Situation and Task
Once again, this was something I had never done before. This was encountered on both the dashboard and manual search pages as we wanted to be able to view all the results without having to load them all at once, slowing the program down. Here, I worked with Chap to implement the infinite scrolling feature.

Action
Before attempting to code this scroller, I went through a tutorial provided by Chap to understand how it worked. I then attempted to implement this code into our program on the manual search page. I encountered multiple issues when doing this and needed constant guidance and help in debugging and rewriting code.

Result and Learning
In the end I was unable to complete the infinite scroll feature as I could not gain the understanding in the allocated time to complete this feature, so it was instead handed off the Chap. Once he had completed it, I analysed his code in my own time to try to grasp where I had gone wrong. This experience allowed me to accept that sometimes I need to step back from a task that is outside of my ability and allow someone more skilled have a go. I feel that while I would have wanted to complete the code myself, this experience will be beneficial in my future as a programmer as sometimes we need to accept when we cannot do something.

3.3.4. Working on a Large-Scale Web Application

Situation and Task
Like the others, I had never had the opportunity to develop a project this size before. There was quite a learning curve to ensure that, when working on the same sections of the artefact as others, that there was no overlap in code written so when everything was pushed to GitHub nothing overwrote anything. I had to ensure there was proper communication between myself and the team, so this never occurred.

Action
When working on a feature I would mark it on the Trello board as "In Progress" and assign myself to the task so the team knew that I was working on the task. If there were components that I was working on that another team member required for their section, I made sure to complete it efficiently to not waste their time. When working on the same pages as others, for example when I was polishing the dashboard page whilst Chap was debugging the infinite scroll there, I ensured that I had all the latest commits pulled into my branch as to not cause conflicts. I also had to learn to create pull requests when I had finished my sections so Chap could then push my changes into the master branch and ensure everything was working correctly.

Result and Learning
From this project I gained a greater understanding of GitHub and how it works, which will benefit me for other large-scale projects in the future. I also gained stronger teamwork capabilities as I know

that I can ask for help when needed, I can efficiently communicate with the team as to my progress even if struggling, and I can work with others on the same file to work on different issues without code conflicts.

3.3.5. Managing my Time Effectively

Situation and Task

As this capstone project took so much time and effort to complete, it forced me to plan out and manage all other responsibilities efficiently so I could adhere to the required deadlines. As someone who struggles with ADHD this has always been an issue of mine. In the past I have only been able to focus on one assessment item at a time, often to my own detriment, but this project forced me to work on multiple tasks from different units at once.

Action

As we were required to complete sprints every fortnight and finalise features in time for tutor and industry partner meetings, I had to ensure I could complete my tasks on time. I treated each milestone as a required deadline to give me the motivation required to complete my work on time. I struggled to balance my time between these deadlines and those set by other units' assessment.

Result and Learning

Through these challenges I was able to find a way to manage my time and prioritise tasks effectively so I could compete all assessment on time and feel happy with their finished quality, something that I had always had issue with. I believe the techniques I learnt to manage my time will benefit me greatly in the workplace and all other personal endeavours as I feel, through this project, I am able to work on multiple tasks at once without significantly sacrificing quality of work. This will help me in the workplace as I will be more reliable when allocated tasks.

4.0. Individual chapter: Elizabeth Robinson

4.1. Project Setup

4.1.1. Project Management Approach

An agile project management approach pushes all team members to actively participate in the project management setup. As the common link between all team members, I took it upon myself to break the ice and encourage the group to get to know each other. One year is a long time to work with a group, so I believed it was essential that we laid the foundations of trust and comradeship. These features were essential for project management because it meant that when tasks were broken up and delegate team members would be accountable, in turn reducing the stress for the team as a whole.

I had a proactive and agile attitude towards the project. Before this capstone, I had never heard of the JavaScript library called React. This lack of knowledge did not deter me from staying positive and committed to completing tutorials before commencing the programming phase of the capstone. My open mindset also came into play when learning how to work with an API database plugin, I was nervous at first because I did not want to let the team down but after several independent attempts and seeking help from the more experienced team members, I was able to complete the NSW scraper.

4.1.2. Client Expectations

The team concurred that Chap, our Team Leader, would lead the client meetings so the consultations were streamlined and coherent. However, when sections of the artefact I had developed were demonstrated I would confidently discuss these features with Ashley (the product owner). This would also involve answering questions about future feature touch ups. Prior to client meetings, we discussed with the team leader what each of us were ready to present. Chap then gave an overview before we individually presented our progress.

4.1.3. Team Collaboration

I actively participated in our dev team meetings by always keeping the dev team updated with what part of a task I currently in the process of deploying. I also encouraged the others to share how they were tracking in addition to sharing if they were having any troubles. To promote a positive team culture, I advocated asking how team members were going outside of the project. I believed that this helped connect the team members to a greater depth and thus in turn strengthened our team's trust and confidence in each other.

4.1.4. Communication

As explained in my client expectation section, I presented the sections I was reasonable for to Ashley. I also did this for the tutor meetings with Aaron. Each group member described what they have completed the week before the meeting and what they were working the week after the meeting. For example, I presented the Auditor table view feature and the view history side component. In addition, I answered any ad hoc legal questions to the best of my ability during meetings. The AACE was very reliable when it organised meetings times, however I still sent reminders into the chat about meetings and times as a precautionary measure.

4.2. Project Plan and Risk

4.2.1. Project planning and progress

While all team members were involved in almost every stage and process of the project, we also assigned people to take responsibility over different sections. In phase 1, once the product backlog was created and refined, I had the responsibility of creating the sprint and release plan outlines. I drafted up the plans by diving the user stories into tasks according to their prioritization. I created a timeline based on the expected difficulty of the tasks and my teammates programming level of ability. To make these predictions, I had to learn more about the team members by identifying and understanding their strengths and weaknesses. Once I had drafted these plans and timelines the team made adjustments before the development began.

In phase 2, the beginning of each sprint started with a meeting of assigning tasks. I would put my hand up for jobs that best suited my skill set but I also was happy and open to tasks that would have stretched my programming abilities. Alongside my degree in Computer Science, I also study a law degree. This education background meant I was assigned the role of being the "legal consultant" of the team. During the project set up, I was responsible for researching what information we could gain from the court websites and how relevant it was for the artefact. I was also the point of contact for other team members if they had questions about the civil procedures. As part of the planning stage, I conducted initial research tests on the 28 Queensland approved contaminated land auditors. I searched all the auditor names in the e-court registry, exactly as the contaminated land management team would have had to do prior to our artefact. I compiled the results into a document to test against our scraper results. This document can be found in the data folder.

4.2.2. Risk Management

I contributed to the mitigation of vague project scope by identifying project goals. I played a part in the creation of user stories and then the adjustments of product backlog during the post sprint meetings with Ashley. To mitigate the risk of my personal challenges getting in the way of the project, I allocate several hours a week to work on my capstone tasks. If I had other commitments, I would immediately let the team know. I believed being transparent and honest was the best option. To alleviate the risk of unclear deadlines, I made a timeline of expected completion dates. The team also used Trello to keep on top of tasks. The Trello encapsulated the sprint specific deadlines, and I updated the progress status of all my allocated tasks to keep on top of deadlines.

4.3. Project Experience

4.3.1 React

Situation

I started the project with a very low-level understanding of web development. A base introduction to HTML and CSS from a first-year unit was the extent of my knowledge. Chap, who knew a lot about web development, suggested we use React, a JavaScript library, to build our user interface.

Task

My task was to utilise the React library to build a user interface that was visually pleasing and matched the mock-ups designed in Phase 1.

Action

To make the most of what React had to offer, I completed a several online React tutorials to familiarise myself with its components and features. I also read several blog articles about React and how to apply it to a project. I was also sure to ask Chap any questions I had about applying React. Chap also suggested we send him any articles or tutorials we were using to solve an issue we had before embarking on applying it to the solution so he could check if it was the most relevant and best option.

Result

In the end, I was able to contribute my share of the client-side build. While I still have a long way before I consider myself an expect in React and web development, I was able to overcome the struggle first associated with using a completely new library.

Learning

I learnt there is a diverse range of resources available such as tutorials and articles for open-source libraries. I found that by doing the initial tutorials it made me understand the basics of React, so I was able to use more complex features in the project. I feel a lot more comfortable with React now and can see myself applying it to future projects at a more difficult level.

4.3.2 NSW Scraper

Situation

A stretch feature outlined in our team agreement was including searches for other state's court databases. Once the Queensland scraper was built, we focused on completing other Australian state searches.

Task

I was tasked with building the NSW court scraper.

Action

Firstly, I studied Chap's Queensland scraper. Then, I downloaded the Insomnia application to play around with the API requests parameters available to the NSW court list search APIs. After establishing the query parameters and return types and formats, I was able to start programming the scraper. I had to encapsulate resulting data into the data types we had predefined as a team so it would correctly interact with the other components of the artefact.

Result

I was able to build an algorithm that accepted query parameters expected from our intended users to search through the NSW case database for relevant cases.

Learning

This task took me longer than I had expected because I was confused about the JSON data type returns as well as how to utilise async functions in JavaScript. However, I learnt that Alex was having similar issues and we were able to discuss and share out findings together which made it easier to fix. I learnt the importance of vocalising drawbacks as soon as possible because others may be having the same problems.

5.0. Individual chapter: Alex Warton

5.1. Project Setup

5.1.1. Project Management Approach

The project management approach that we used as a group was the Agile method. Being only a group of four, the Agile method allowed us to be more flexible with our sprints as capstone was not our only subject. By making iterative development a priority, we provided the minimum viable product and expanded on that which allowed for more flexible and efficient deployment of our program. I, and all my group members, assisted in setting up this project management style by always attending group meetings, having clear communication, and prioritising our deadlines accordingly. I know that I had an active attitude towards this by constantly trying to update my skills I did not think were up to standard including prioritising JavaScript tutorials as these were extremely necessary for the project.

5.1.2. Client Expectations

Liaison with our client was quite a simple task due to our organisation of client meetings and our clear communication of timelines and deadlines. I would assist the other group members in this task by ensuring what we had to present to our client was coherent and easily understood. In situations where our usual project manager Chap was unable to be present at these meetings, I was more than happy to take a more leading role when talking with our client. During these meetings however, we each displayed our individual items and explained them. Individually I made sure I understood my components well enough to make sure the client could easily understand the work that had taken place.

5.1.3. Team Collaboration

Team collaboration was easily maintained by listening and liaising with our project manager Chap. He would often distribute tasks for us to complete. To facilitate this, I would constantly keep the team updated as to any issues that I was facing. This would allow them to come back to me with a solution or at least let them know about my current situation. We organised meetings weekly where I, and my other team members, would present our work to each other to keep us accountable.

5.1.4. Communication

As discussed above, communication with clients and group was mostly done by distributing each section of work that we had completed during that speech. All meetings with our client were done over video calls and I would individually present my part of the work for that sprint and explain it. For group meetings, we would also set times through the week where I would allow my work to be presented and commented on for any changes.

5.2. Project Plan and Risk

5.2.1. Project planning and progress

In the first semester of capstone, we as group would distribute our tasks evenly. Perhaps my most crucial job for the first section was developing our first iteration of user stories to present to our client. These became the basis for our minimum viable product and were later expanded upon to become our requirements for the program.

The second semester involved a lot more work for me. The first sprints revolved around creating basic functionality and wireframes for us to get an idea of what the program would look like. I was assigned with creating the first case component mock-up. In following sprints, I also created the basis for the case details page and the auditor edit/delete form.

5.2.2. Risk Management

The greatest risk to any capstone project is time management. As we are working with non-existent programs, time management needs to take priority as we are also university students that manage work and other classes. I would keep myself reliable my allocating certain days to only do capstone work as well as making myself available via messages any time in case there were any errors or quick fixes that I needed to change. This did occur towards the end of our project and luckily, I was available via message to quickly try and find a solution. Had I not allocated this time, chances are we would have had to deploy without that specific feature.

5.3. Project Experience

5.3.1. Situation

At the very beginning of semester 2, I was unfamiliar with the framework of react. This caused a bit of anxiety for myself and made me question my abilities.

5.3.2. Task

As this was the framework we were using, it was tasked to me that I would become familiar with it and its principles, so we work as efficiently as possible.

5.3.3. Action

Dedicating a large amount of time to watching tutorials and understanding this framework is what helped me overcome my fears. I would consider this as ‘studying’ and would treat it the same as I would a university subject by taking notes as well as getting some practical experience by making my own applications from scratch.

5.3.4. Result

Having a now better understanding of React has helped me a large amount in the long run not only with our specific project but with my goals of web development. Having not heard of React, it was incredibly interesting to see just how many apps are based on this framework. This has encouraged me to look further into react.

5.3.5. Learning

Constantly staying up to date in the world of programming is almost impossible with so many changes and so many ways of thinking. This has made me realise that sticking to one framework is perhaps the best option and learning it to the best of my ability is extremely beneficial.

6.0. Appendices

6.1. Team Agreement

IFB398 & IFB399 - Capstone Project
SEMESTER 1 & 2, 2022
Team AACE
TEAM AGREEMENT

Team Member	Role	Student ID
Chap Callanan	Project Manager	n10470026
Ash Phillips	Developer (Front and Back end)	n10477659
Elizabeth “Libby” Robinson	Developer (Front and Back end)	n10397167
Alexander Warton	UX Developer/Business Analyst	n10529861

Project: Web Scraping of Court Data

Individual expectations:

Team Member	Expectations
Chap Callanan	- Attend all tutor meetings on time - Attend weekly meetings - Complete assigned tasks prior to the meeting/tutorial - Actively participate in the work - Maintain constant communication - Do the best they can as this affects others
Ash Phillips	- Attend all tutor meetings on time - Attend weekly meetings - Complete assigned tasks prior to the meeting/tutorial - Actively participate in the work - Maintain constant communication - Do the best they can as this affects others
Elizabeth “Libby” Robinson	- Attend all tutor meetings on time - Attend weekly meetings - Complete assigned tasks prior to the meeting/tutorial - Actively participate in the work - Maintain constant communication - Do the best they can as this affects others
Alexander Warton	- Attend all tutor meetings on time - Attend weekly meetings - Complete assigned tasks prior to the meeting/tutorial - Actively participate in the work

	- Maintain constant communication - Do the best they can as this affects others

Team's expectations of commitment and communication:

- Attend all tutor meetings on time
- Attend all weekly meetings
- Notify everyone of absence prior to the tutorials/meetings
- Complete assigned tasks/homework prior to attending
- Reach out to the team if there are any issues with the assignment/tasks
- Provide constructive feedback to others and help everyone
- Commitment to their assigned tasks and agreed deadlines
- Reach out for help if needed
- Tasks will be equally distributed
- Regularly update status on Trello
- Keep on track with set plans and due dates for tasks

The team will adopt the following team behaviors from Agile's way of working:

- We as a team will take ownership of the assigned tasks.
- We will stick to our agreed working patterns.
- We will plan for one offline meeting session between tutor meetings.
- We will let the team know when one of us is late or going early.
- We will ask for help when needed.
- We will try to ensure equal contribution for each weekly task.
- We will try and educate the non-contributing members to enhance their contributions.
- We will ensure tutors are made aware of non-contributing members or consistent lack of participation for anyone of the team members.
- We will use Trello (or similar) as the main method for our team communications.
- We will use Trello as the main tool to record our intermediary and final outcomes.

Meeting times:

Out of class

- Monday: 8:30pm-9:30pm (Online)
- Thursday: 8:30pm-9:30pm (Online)

Tutor Meetings

- Friday: 9:45am-10:00am (Over Zoom)

Resolving member issues:

If there is an issue with a member during the project, the following steps will be followed:

1. A team meeting will be held to discuss and get in contact with the team member regarding the contract breaches that were made
2. If the step above fails, the tutor will be contacted and asked for help
3. If the step above fails, the unit coordinator will be notified and asked for help

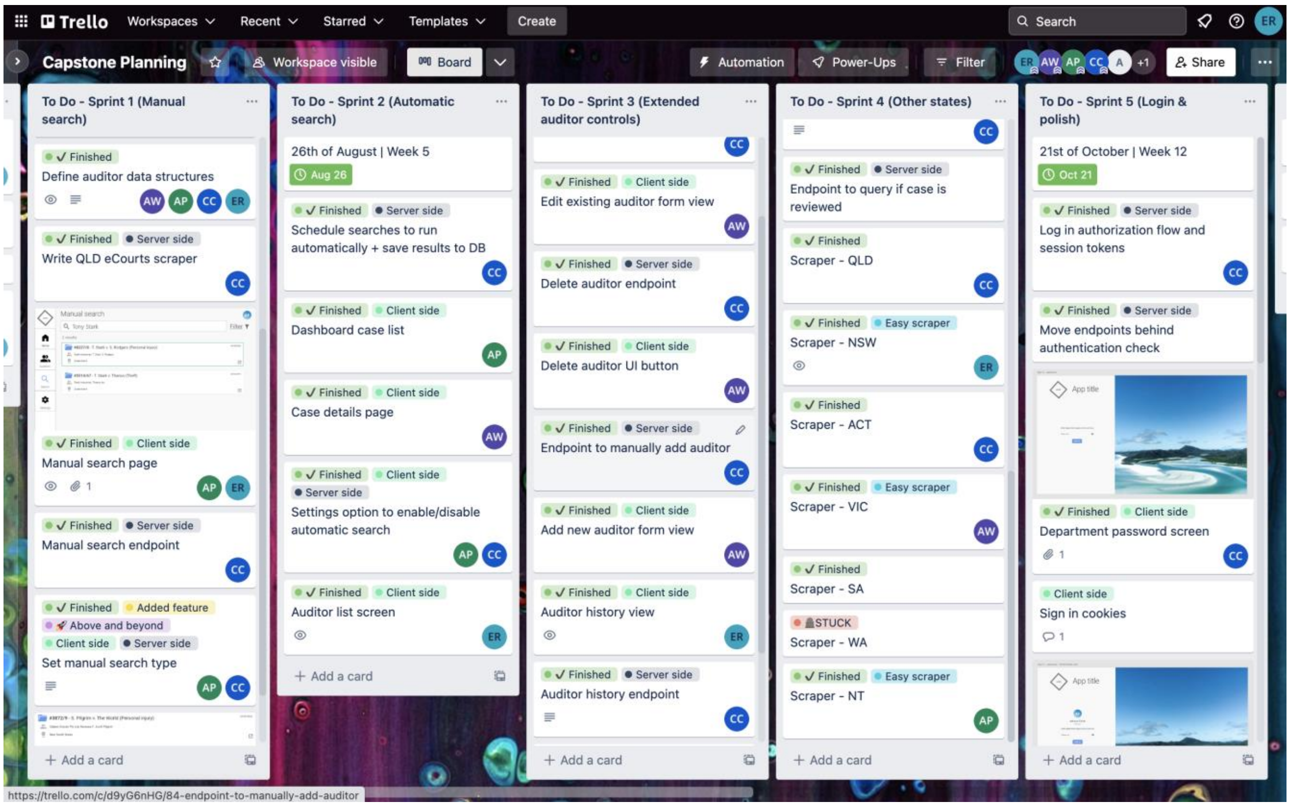
By signing this document, I agree to comply with the expectations of the Team as stated above

Chap Callanan	CC
Ash Phillips	AP

Elizabeth “Libby” Robinson	ER
Alexander Warton	AW

Date: 14/04/2022

6.2. Trello Board



To Do - Sprint 1 (Manual search)

- ✓ Finished — Define auditor data structures (AW AP CC ER)
- ✓ Finished, Server side — Write QLD eCourts scraper (CC)
- ✓ Finished, Client side — Manual search page (AP ER)
- ✓ Finished, Server side — Manual search endpoint (CC)
- ✓ Finished, Added feature, 🚀 Above and beyond, Client side, Server side — Set manual search type (AP CC)
- + Add a card

To Do - Sprint 2 (Automatic search)

- 26th of August | Week 5 — Aug 26
- ✓ Finished, Server side — Schedule searches to run automatically + save results to DB (CC)
- ✓ Finished, Client side — Dashboard case list (AP)
- ✓ Finished, Client side — Case details page (AW)
- ✓ Finished, Client side, Server side — Settings option to enable/disable automatic search (AP CC)
- ✓ Finished, Client side — Auditor list screen (ER)
- + Add a card

To Do - Sprint 3 (Extended auditor controls)

- ✓ Finished, Client side — Edit existing auditor form view (AW)
- ✓ Finished, Server side — Delete auditor endpoint (CC)
- ✓ Finished, Client side — Delete auditor UI button (AW)
- ✓ Finished, Server side — Endpoint to manually add auditor (CC)
- ✓ Finished, Client side — Add new auditor form view (AW)
- ✓ Finished, Client side — Auditor history view (ER)
- ✓ Finished, Server side — Auditor history endpoint (CC)
- + Add a card

To Do - Sprint 4 (Other states)

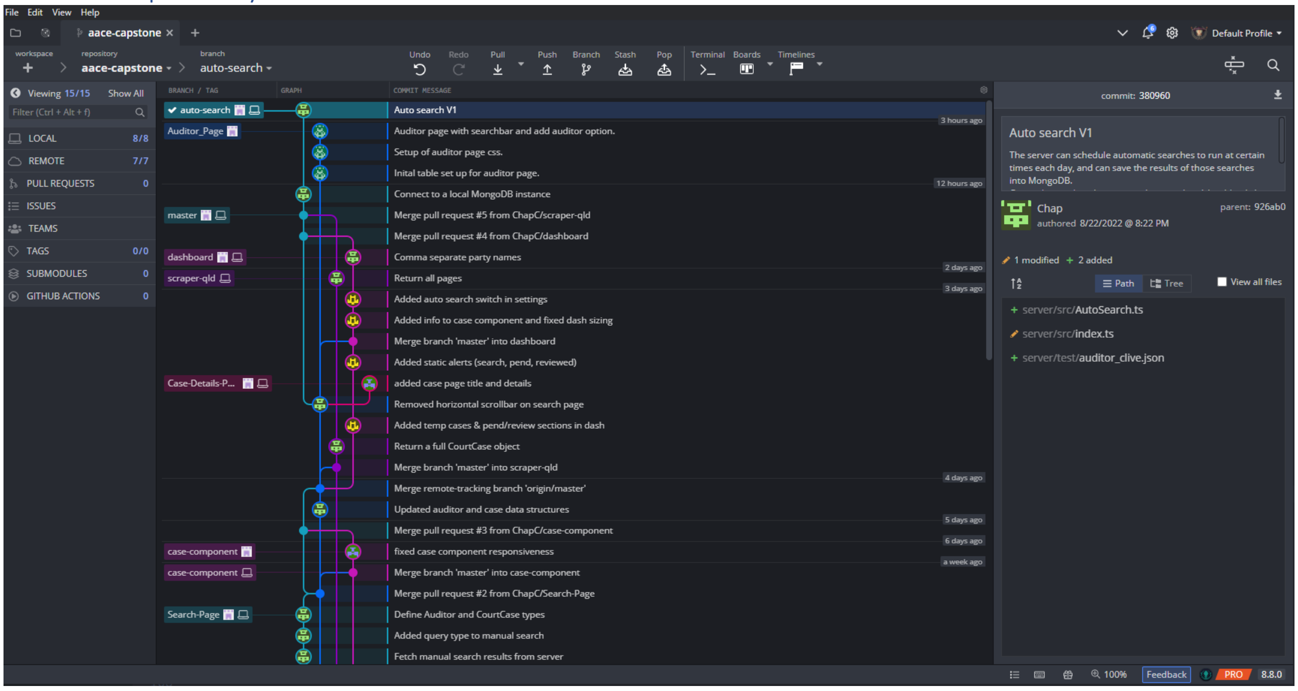
- ✓ Finished, Server side — Endpoint to query if case is reviewed
- ✓ Finished — Scraper - QLD (CC)
- ✓ Finished, Easy scraper — Scraper - NSW (ER)
- ✓ Finished — Scraper - ACT (CC)
- ✓ Finished, Easy scraper — Scraper - VIC (AW)
- ✓ Finished — Scraper - SA
- STUCK — Scraper - WA
- ✓ Finished, Easy scraper — Scraper - NT (AP)
- + Add a card

To Do - Sprint 5 (Login & polish)

- 21st of October | Week 12 — Oct 21
- ✓ Finished, Server side — Log in authorization flow and session tokens (CC)
- ✓ Finished, Server side — Move endpoints behind authentication check
- ✓ Finished, Client side — Department password screen (CC)
- Client side — Sign in cookies
- + Add a card

https://trello.com/c/d9yG6nHG/84-endpoint-to-manually-add-auditor

6.3. Git Repository



repository: aace-capstone — branch: auto-search

BRANCH / TAG	COMMIT MESSAGE	
auto-search	Auto search V1	3 hours ago
Auditor_Page	Auditor page with searchbar and add auditor option.	
	Setup of auditor page css.	
	Inital table set up for auditor page.	
	Connect to a local MongoDB instance	12 hours ago
master	Merge pull request #5 from ChapC/scraper-qld	
	Merge pull request #4 from ChapC/dashboard	
dashboard	Comma separate party names	2 days ago
scraper-qld	Return all pages	3 days ago
	Added auto search switch in settings	
	Added info to case component and fixed dash sizing	
	Merge branch 'master' into dashboard	
	Added static alerts (search, pend, reviewed)	
Case-Details-P...	added case page title and details	
	Removed horizontal scrollbar on search page	
	Added temp cases & pend/review sections in dash	
	Return a full CourtCase object	
	Merge branch 'master' into scraper-qld	
	Merge remote-tracking branch 'origin/master'	4 days ago
	Updated auditor and case data structures	
	Merge pull request #3 from ChapC/case-component	5 days ago
case-component	fixed case component responsiveness	6 days ago
case-component	Merge branch 'master' into case-component	a week ago
	Merge pull request #2 from ChapC/Search-Page	
Search-Page	Define Auditor and CourtCase types	
	Added query type to manual search	
	Fetch manual search results from server	

commit: 380960

Auto search V1

The server can schedule automatic searches to run at certain times each day, and can save the results of those searches into MongoDB.

Chap — authored 8/22/2022 @ 8:22 PM — parent: 926ab0

1 modified + 2 added

- + server/src/AutoSearch.ts
- server/src/index.ts
- + server/test/auditor_clive.json

6.4. Client meeting minutes

Client Meeting 1: 29/07

Goals:

- Review prototype
- Review release/sprint plan
- Walkthrough plan for the semester
- Ask for feedback and answer any questions presented

Meeting Notes:

- Case details: info is good
- What states were we able to scrape? - everything but Tasmania and South Australia
- Provide list of the links for the court website we'll be searching;
  - show them what is excluded rather than included,
  - mutual recognition for certain states – interested in knowing if they have applicants from certain locations if there ISN'T info in our scraper for them so they can contact the relevant people to find any issues
- Hosting, if they can't provide us access we will need to rent server space; monthly fee, not much as it is not a high traffic application; firewall issues
- In their google search their systems don't allow access to the ad hits when clicked on
- Checking with their application manager to provide more info on where our application can be hosted
- Do we really need the login the user account as it adds another level of management to it; only use is for the review cases

Client Meeting 2: 12/08

Goals:

- Show progress on web app
- Talk through the form to complete
- Discuss next steps
- Ask for feedback and answer any questions presented

Meeting Notes:

- GHD large engineering corp
- Shouldn't be a problem hosting the web app:
  - steer clear of access into their own systems, private info not publicly available,
  - ongoing maintenance and format on where its hosted,
  - sever size and storage space, application that is not data hungry not a large database didn't seem to be a problem but still need to talk to the data service guys
- Send through our trello board
- How we're working together, how breaking up our roles and tasks
- Meeting with web services next week to get more info for us
- User guides and technical details on our application

- Take screenshots of code of our pages as we work on them to send to help with the meetings. Short slide show to show what we're working on for his team, GIF

To Send:
- Our Trello board either screenshot or otherwise
- Info on possible user guides and technical details of our application
- Screenshots/GIFs our progress on the code and the application itself in a slideshow
- Info on what we can grab from the court databases

Client Meeting 3:02/09
TO DO: Ask Ashley/networking team about an email address for the app.
Meeting Notes:
- Still aiming for the intranet web
- Mentioned email, he will look into it – cost associated with email addresses, they have a general inbox they use for everything

Client Meeting 4: 9/09
- Search for any auditor in all states
- Each search will search all of the court states and auditors so no risk messing a hit
- Would one or a number of us be interested with coming along to one of his meetings with him with his boss – further towards the end we'll set something up

Client Meeting 5: 20/09
Goals:
- Showcase new state scrapers
    - Show current progress on Trello
    - Show either code files or screenshots of code from everyone
- The plan from Friday onwards is to integrate the completed scrapers into the application and test them
- Ask if there is any info from the network team

Send the meeting review

6.5. Release and Sprint Plans

Release Plan
Release 0
Delivery date: **10th June 2022**
Release 0 is focused on creating a comprehensive delivery plan and prototype using the information we found through our client meetings and independent research.
This plan will be implemented in the second half of this year according to below release plans.

Requirements from Client
Requirement collecting began before the first meeting with the client. The project overview was used as a foundation for the project requirements. The two initial meetings with the client helped further establishes the necessary requirements. It must be noted that to remain agile we must

accept that these requirements may slightly vary as the project develops and the client has a clearer understanding of what they want.

User Stories Completed
Using the requirements collected from the client a list of user stories that encompasses the client's must-haves, should-haves and could-have will be created. During our meetings the client will be able to agree to or adjust these stories, depending on how they envision the final product to behaviour.

Clickable Prototype
Once a clear outline of what the product must be able to achieve, a prototype can be created on Adobe XD. The prototype will be a clickable version that can be run through to emulate how the final product should behaviour once completed.

Release 1

Delivery date: **26th August 2022** Total Story Points: **15**

Release one will be focused on creating the back end of the data scrapping tool. The UI will not have been implanted yet, rather the foundations of the program will have been established. Mass searches with the currently approved Queensland contaminated land auditors can be done upon request as well as manually entering a single auditor's name to check if they are involved in any civil action.

Queensland Auditor Name Search

Client can search the list of current Queensland approved auditors

Story ID	Story Title	Story Points
14	Qld Gov approved auditors list	4
11	Manual auditor search	2
1	View list of searchable auditors	2
	Story Point Sub-Total: 6	

Automated Name Search

The program will automatically and periodically search the provided list of auditors

Story ID	Story Title	Story Points
8	Program automated search	4
9	Disable program automated search	1
6	Periodic results	4
	Story Point Sub-Total: 9	

Release 2

Delivery date: **30th September 2022** Total Story Points: 4

This release is focused on implementing our foundational scrapper program into a website that the client can access through a URL.

Accessible Website

Client is provided with login details and can access website

Story ID	Story Title	Story Points
13	Website deliverable	4
	Story Point Sub-Total: 4	

Release 3

Delivery date: **21st October 2022** Total Story Points: 15

Release 3 administers features targeted at making the scrapper tool more useful for the client. It enables the scrapper to be a more dynamic tool. The client will be able to add auditor names to the approved auditor list

Adjustable Approved Auditor List

Client can add, remove, or adjust auditors' details.

Story ID	Story Title	Story Points
2	Change auditor details	1
3	Remove auditor	1
4	Add person to auditor list	1
	Story Point Sub-Total: 3	

Viewing Results

The client can choose the type of notification they receive when a civil action is identified. Client can also select filters on results and review what cases have had an outcome.

Story ID	Story Title	Story Points
5	Alter shown results	4
7	Notification type	2
10	Review actioned cases	2
	Story Point Sub-Total: 8	

Outer-state Auditors

This feature will include finding civil actions in other state jurisdictions for Queensland auditors.

Story ID	Story Title	Story Points
12	Different state search	4
	Story Point Sub-Total: 4	

Sprint Plan

Sprint 1

Total Story Points: 8 Total Hours: 32

Current Velocity: 4 hrs/story

S1 and S14: Qld Gov approved auditors list

Task ID	Task Description	Estimate	Taken
T01	Create data structure for creating auditor objects	3	
T02	Create database for active auditors	3	
T03	Write test cases	2	
T04	Create scrapper tool utilising Department Website	5	
T05	Create user interface of display of list	3	
T06	Verify story is complete (acceptance test)	1	
	Story Points: 6	17	

S11: Manual auditor search

Task ID	Task Description	Estimate	Taken
T07	Create database for litigation involvement	3	
T08	Create scrapper algorithm to search court's databases	6	
T09	Write test cases	2	
T10	Create link between customer input and database	3	
T11	Verify story is complete (acceptance test)	1	
	Story Points: 2	15	

Sprint 2

Total Story Points: 9 Total Hours: 13
Current Velocity: 1.44 hrs/story

S08 and S06: Program automated search

Task ID	Task Description	Estimate	Taken
T12	Automate scrapper T07 and T09 on a daily basis	1	
T13	Write test cases	1	
T14	Create user interface to display when court case has been identified	3	
T15	Verify story is complete (acceptance test)	1	
	Story Points: 8	6	

S9: Disable program automated search

Task ID	Task Description	Estimate	Taken
T16	Write test cases	1	
T17	Create user option to stop S08	1	
T18	Create executable function execute T7	2	
T19	Create UI option that links function	2	
T20	Verify story is complete (acceptance test)	1	
	Story Points: 1	7	

Sprint 3

Total Story Points: 6 Total Hours: 19
Current Velocity: 3.17

S 13: Website deliverable

Task ID	Task Description	Estimate	Taken
T21	Write test cases	2	
T22	Establish server	3	
T23	Design website layout	6	
T24	Add all features	5	
T25	Test cases	2	
T26	Verify story is complete (acceptance test)	1	
	Story Points: 6	19	

Sprint 4

Total Story Points: 15   Total Hours: 42
Current Velocity:2.8 hrs/story

S 2: Change auditor details

Task ID	Task Description	Estimate	Taken
T27	Create auditor editable field	1	
T28	Write test cases	1	
T29	Extend website to include UI option to change auditor's details	2	
T30	Verify story is complete (acceptance test)	1	
	Story Points: 1	5	

S 3: Remove auditor

Task ID	Task Description	Estimate	Taken
T31	Write test cases	1	
T32	Extend website to include UI option to remove an auditor from displayed list	2	
T33	Verify story is complete (acceptance test)	1	
	Story Points: 1	4	

S 4: Add person to auditor list

Task ID	Task Description	Estimate	Taken
T34	Write test cases	1	
T35	Extend website to include UI option to add an auditor to database	1	
T36	Verify story is complete (acceptance test)	1	
	Story Points: 1	3	

S 5: Alter shown results

Task ID	Task Description	Estimate	Taken
T37	Extend UI to add view preferences	1	
T38	Write test cases	1	
T39	Verify story is complete (acceptance test)	1	
	Story Points: 4	3	

S 7: Notification type

Task ID	Task Description	Estimate	Taken
T40	Collate list of different notification types	1	
T41	Write test cases	1	
T42	Create notification flag	1	
T43	Extend UI to allow user to choose notification type	2	
T44	Verify story is complete (acceptance test)	1	
	Story Points: 2	6	

S 10: Review actioned cases

Task ID	Task Description	Estimate	Taken
T45	Create flag for case object to identify if reviewed or not	1	
T46	Write test cases	1	
T47	Extend UI to enable user to set case flag to be reviewed	2	
T48	Verify story is complete (acceptance test)	1	
	Story Points: 2	5	

S 12: Different state search

Task ID	Task Description	Estimate	Taken
T49	Complete S1 and S14 for each state's website.	10	
T50	Write test cases	2	
T51	Create new database	3	
T52	Verify story is complete (acceptance test)	1	
	Story Points: 4	16	

Planning Timeline (Release 0)

Requirements from Client	**User Stories Completed**	**Clickable Prototype**	**Project Presentation**	**Project Report**
1st May	16th May 2022	25th May 2022	1st June 2022	10th June 2022

Delivery Timeline

Week											
1	2	3	4	5	6	7	8	9	10	11	12
Sprint 1		Sprint 2		Sprint 3				Sprint 4		Sprint 5	
Release 1 (05/08/22 - 26/08/22)				Release 2 (26/08/22 - 30/09/22)				Release 3 (30/09/22 - 21/10/22)			

6.6. Product Backlog/User Stories

6.6.1. Product Backlog

PRODUCT BACKLOG

ID	As a ...	I want ...	So that ...	Acceptance criteria			Priority	Story points
				Given	When	Then		
1	User	to see the list of auditors currently being searched	I can review any of their ongoing cases	that an option to "View List"	I click "View List"	The system shows the list of current auditors in the system	Should	2

2	User	to change the details of an auditor	if needed, I can edit their information	that an option to "Edit Auditor" is displayed	I click on "Edit Auditor"	The system should display the auditor details to be able to edit	Should	1
3	User	to remove an auditor from the list of staff	auditors that are no longer employed cannot be searched for	that an option to "Delete Auditor" is displayed	I click on "Delete Auditor"	The system should display a popup asking for confirmation before deleting the auditor	Must	1
4	User	to add a person to the list of auditors manually	newly employed auditor can be searched for	that an option to "Add Auditor" is displayed	I click on "Add Auditor"	The system should display via a website form to input auditor details including their name, organization, etc.	Must	1
5	User	to alter what results are shown from the search	I can filter the results	there is a "Filter" option	I click on "Filter"	the system should give a drop down to select filters for the search	Could	4
6	User	to receive these search results periodically	I can be notified of any changes	there is a setting to "Automate System"	I click "Enable"	The system should send notifcations to my preferred channel when any ongoing cases are found	Must	4
7	User	to change the way that I receive search results	If needed, I can receive results on different platforms	there is an option to "Edit Notification Type"	I click "Edit Notification Type"	The system should display a list of notification types (e.g. email, text, etc.) where the information can be input and enabled	Could	2
8	User	to enable the program to make searches	the process can be automatic, saving me time	there is a setting to "Automate System"	I click "Enable"	the system should automate the searching and notification process	Must	4
9	User	to disable the program to make searches	the process can be made manual	there is a setting to "Automate System"	I click "Disable"	the system should only run through manual searches	Should	1
10	User	to be able to review which cases have been actioned	I can see which auditors have broken the rules	on the home page there is an "Actioned" section	I click on an actioned case	the system sets the status to "Reviewed"	Should	2
11	User	manually search for an auditor	I can review if there are any ongoing cases for certain people	There is a "Search" bar	I enter information on an auditor (name) and click "Search"	the system returns the results of my search	Must	2
12	User	change the state of search for cases in	I can check other state databases for ongoing cases	there is an option of "State"	I click "State"	the system should display a drop-down menu of searchable state to change to	Should	4
13	User	a easily accessible website	I can view the information in a straightforward manner	There is a URL	I enter the URL	I am led to the website I can log in to and view my desired information	Must	6

14	User	the auditors on the QLD government website to be automatically added to the system	I can save time editing the auditor list in the system	There is a list of auditors in the system	the auditor list on the QLD government website is updated	the system updates the website list accordingly	Should	4
15	User	to manually search for an auditor organization	I can get the results I want	There is a "Search" bar	I enter an auditor's organization and click "Search"	the system returns the results of my search	Must	2
16	User	to enable the search for any court type	I can change what court results I would like to see	there is a drop-down menu for "Court Type"	I select a court from the drop-down menu	the system returns results from that court type	Could	4
17	User	to be able to view an auditor's case history	I can review their past cases	there is a "History" section in an auditor's information page	I look under that section	all the auditors past cases are displayed and viewable	Could	4
18	User	to login with my department login details	I can be admitted to the site and make changes where required	there is a "Login" page	I enter my department login details	I am given access to the site	Should	2
19	User	the name of the user who reviews a case to be logged	I can keep track of which users were responsible for which decisions	there is a "Editor ID" in the review form	I enter my name/ID	My details can be seen in the reviewed case	Could	2

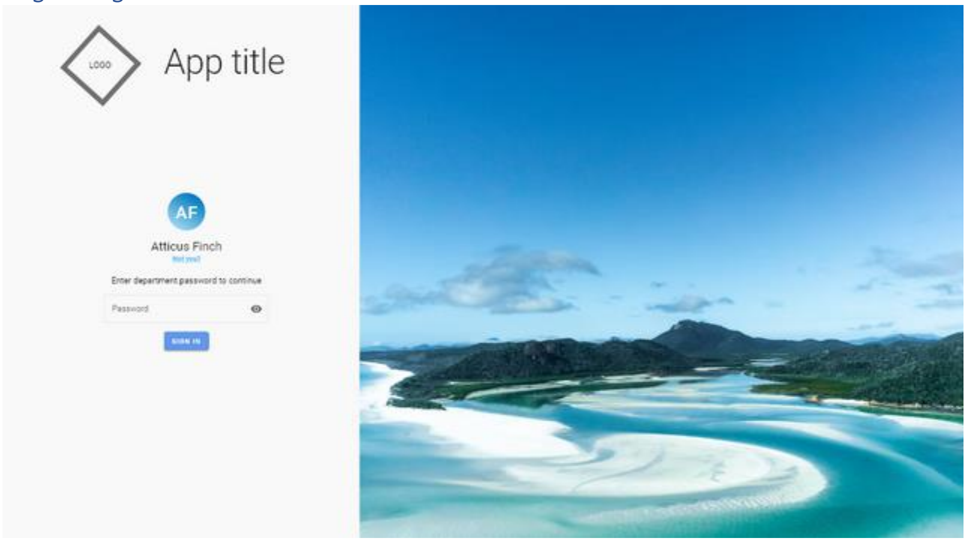
6.6.2. Prioritised Requirements List (PRL)

Prioritised Requirements List				
Reqs ID	**Requirement description**	**Story Pts**	**MosCow**	**Description (Justification)**
R1	See List of Auditors	2	Should	Users need to be able to see the list of auditors that are currently being searched. Prioritisation increased from a Could to Should due to the realisation of the importance of this list as without it the user could not see which auditors are present in the system.
R2	Change Auditor Details	1	Should	Users should be able to change and update an auditor's details when required. This is a Should as it is not required for the application to function but would be nice to have for the user.
R3	Remove an Auditor	1	Must	Users need to be able to remove an auditor from the system if they no longer work for the DES. This is a must as if auditors cannot be removed, searches will occur on unrequired auditors.
R4	Add an Auditor	1	Must	Users need to be able to add auditors to the system for the searches to function. This is a must as without auditors, there is nothing to search for.
R5	Alter Search Results	4	Could	Users can alter/filter search results. This is a could as it is not required but may be useful for the user to save time analysing the results.
R6	Receive Search Results Periodically	4	Must	Users need to be able to automatically receive search results. This is a must as without automatic updates the application has no use.

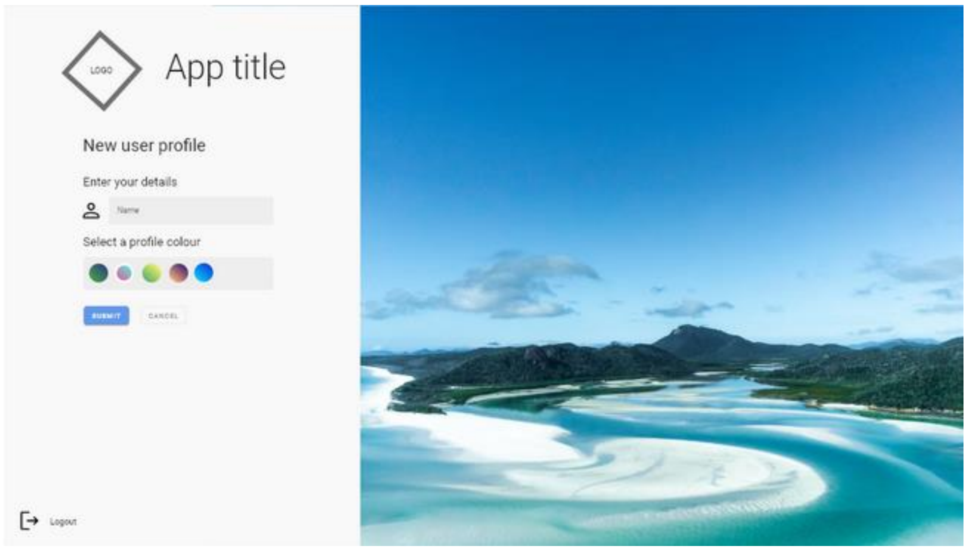
R7	Change Search Result Notification Platform	2	Could	Users may change the platform where they receive notifications of search results. This is a could as it is not required but gives the user the option to be notified differently.
R8	Enable Automatic Searches	4	Must	Users need to be able to enable automatic searches as the DES requires them. This is a must as without the option to enable it, there is no use for the application.
R9	Disable Automatic Searches	1	Should	Users need to disable automatic searches when required. This is a should as it is not required for the application to function, but the user's satisfaction may decrease.
R10	Review Actioned Cases	2	Should	Users should be able to review actioned cases. This is a should as it is not required but would help the users to understand auditors' cases.
R11	Manually Search for Auditor	2	Must	Users should be able to manually search for auditors. Prioritisation increased from Should to Must upon the realisation that without it, users can only rely on automatic searches, disallowing them from searching for specific auditors.
R12	Change State	4	Should	Users should be able to change the state for which they search for auditors. This is a should as it is not required for the system but allows users to refine their searches when desired.
R13	Easily Accessible	6	Must	Users need to be able to easily access the website when required. This is a must as if it were not the application could not be edited.
R14	Auditors Automatically Added	4	Should	Auditors should be automatically added to the system based on the list present on the DES website. This is a should as it is not required but would save users time adding auditors that are registered on the website.
R15	Manually Search for Organisation	2	Must	Users need to be able to input manual searches via the organisation. This is a must as without it, users can only rely on automatic searches, disallowing them from searching for specific auditors.
R16	Enable Search for Any Court Type	4	Could	Users can search in any court type. This is a could as it is not required but will allow users to search beyond civil actions.
R17	View Case History	4	Could	Users can view the case history of auditors. This is a could as it is not required by the system but allows users to review auditors past cases to view patterns, etc.
R18	Login to Application	2	Should	Users should be able to login to the application. This is a should as it is not required but will protect the privacy and security of the application and the auditors mentioned.
R19	Log Name of User	2	Could	Users can log their name when reviewing cases. This is a could as it is not required but it could be nice for the users to see who has actioned previous cases.

6.7. Initial Artefact Designs (Phase 1)

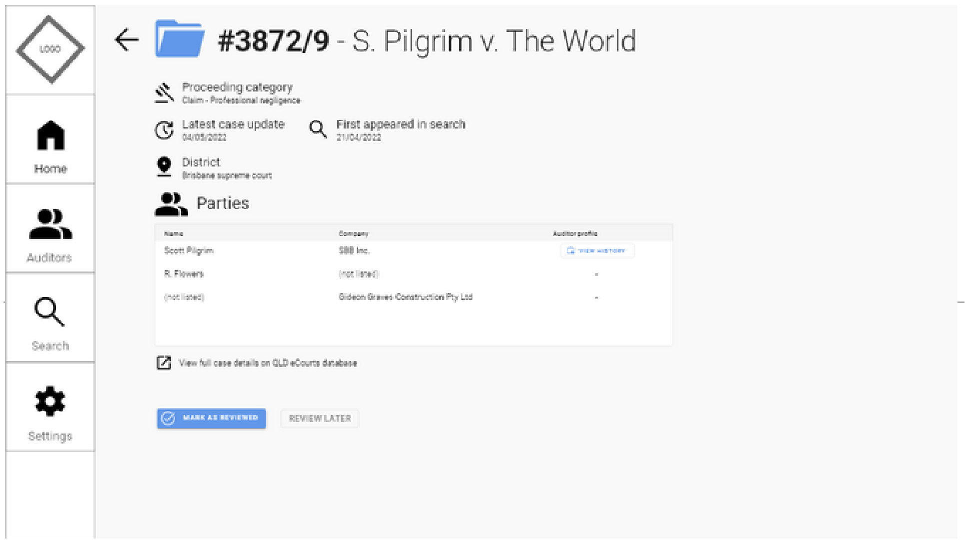
Login Page



New User



Case Details



#3872/9 - S. Pilgrim v. The World

Proceeding category
Claim - Professional negligence

Latest case update
04/05/2022

First appeared in search
21/04/2022

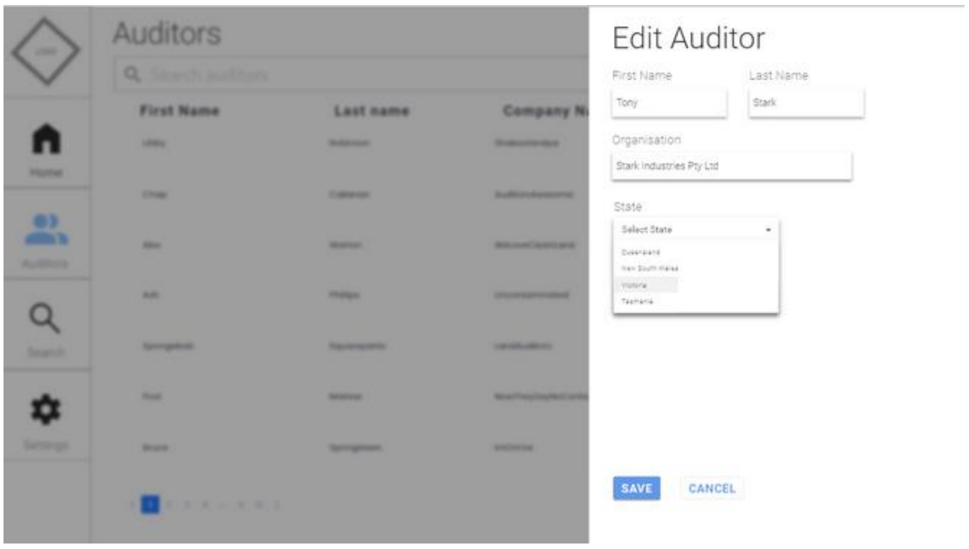
District
Brisbane supreme court

Parties

Name	Company	Auditor profile
Scott Pilgrim	SBB Inc.	VIEW HISTORY
R. Flowers	(not listed)	-
(not listed)	Gideon Graves Construction Pty Ltd	-

View full case details on QLD eCourts database

Edit Auditors



Edit Auditor

First Name: Tony

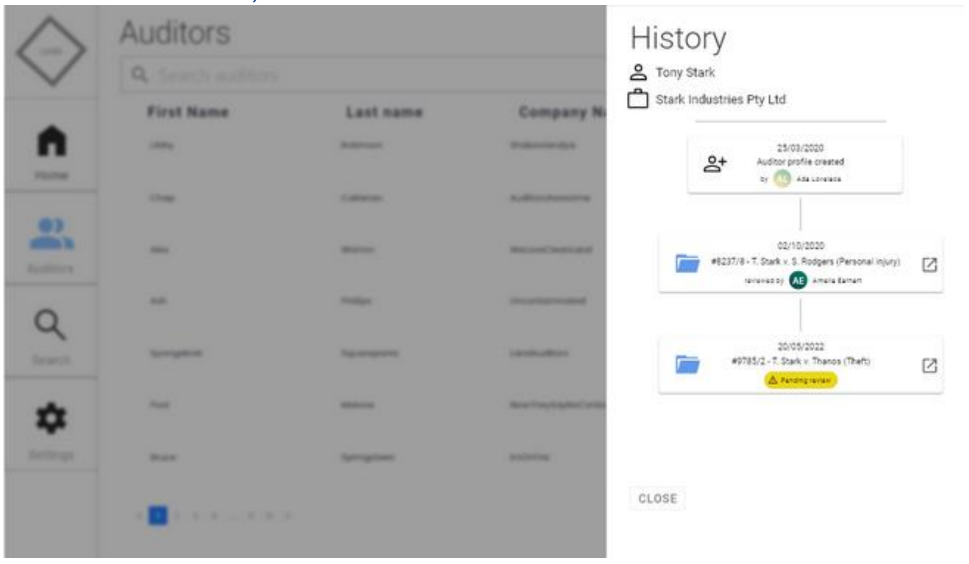
Last Name: Stark

Organisation: Stark Industries Pty Ltd

State: Select State

- Queensland
- New South Wales
- Victoria
- Tasmania

View Auditor History



History

Tony Stark

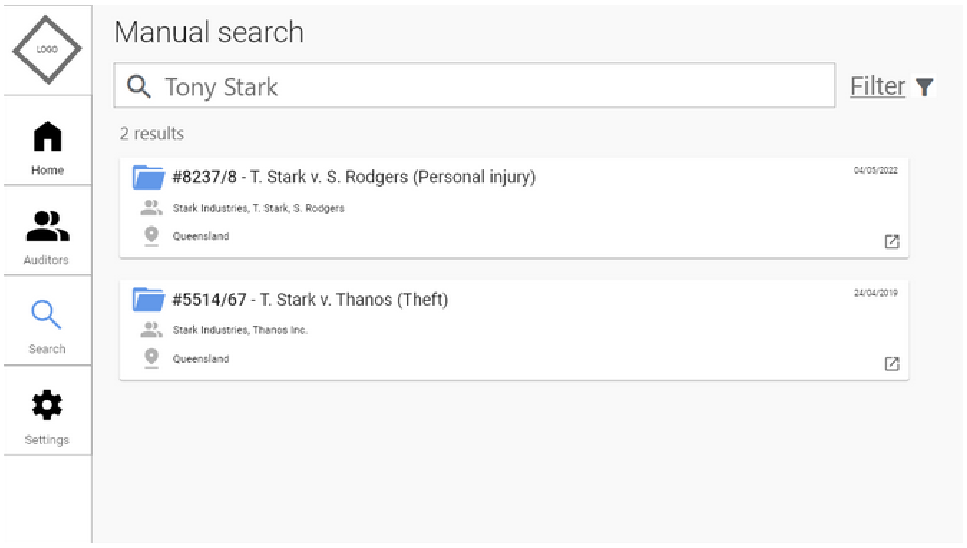
Stark Industries Pty Ltd

25/03/2020
Auditor profile created
by AL Ada Lovelace

02/10/2020
#8237/8 - T. Stark v. S. Rodgers (Personal injury)
reviewed by AE Amelia Earhart

20/05/2022
#9785/2 - T. Stark v. Thanos (Theft)
Pending review

Manual Search



Manual search

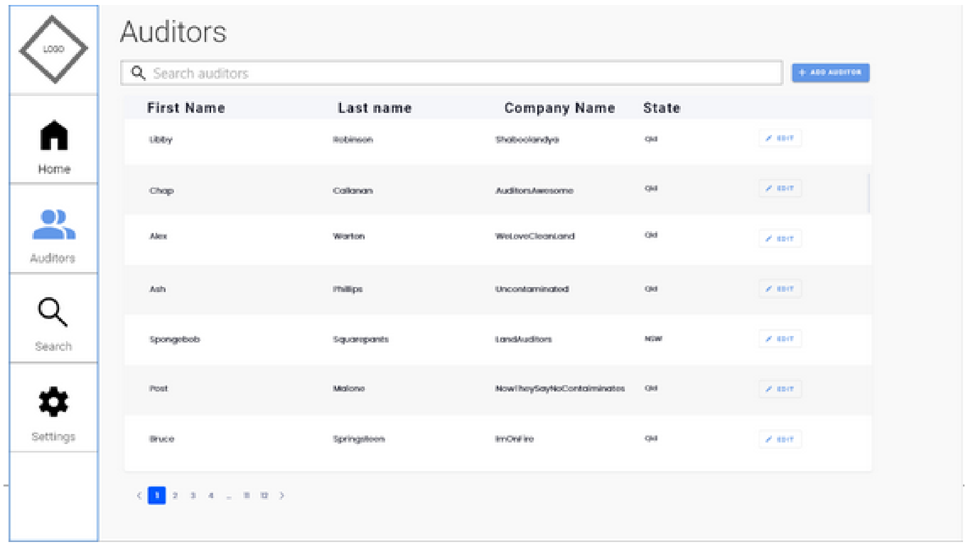
Tony Stark

2 results

#8237/8 - T. Stark v. S. Rodgers (Personal injury) — 04/05/2022
Stark Industries, T. Stark, S. Rodgers
Queensland

#5514/67 - T. Stark v. Thanos (Theft) — 24/04/2019
Stark Industries, Thanos Inc.
Queensland

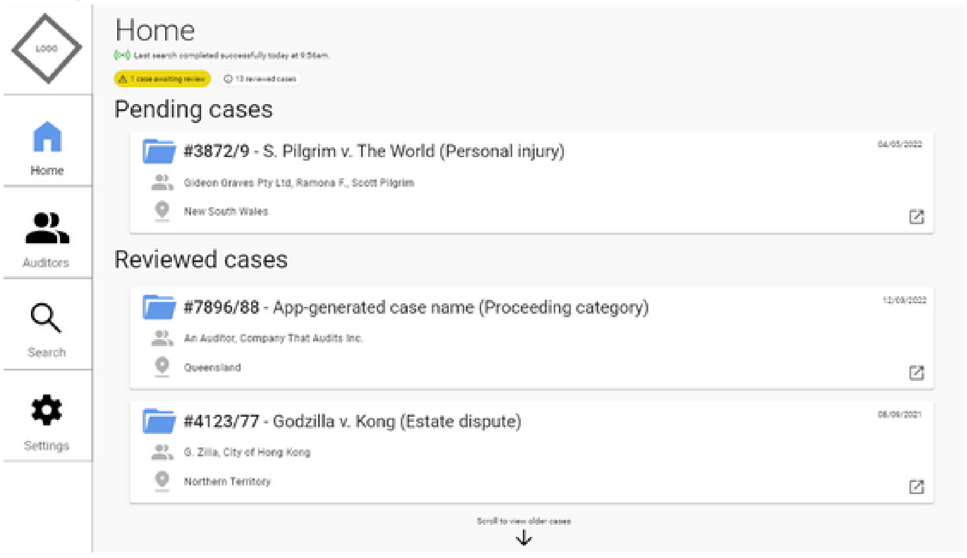
Auditors Page



Auditors

First Name	Last name	Company Name	State	
Libby	Robinson	Shabooiandya	Qld	EDIT
Chap	Callanan	AuditorsAwesome	Qld	EDIT
Alex	Warton	WeLoveCleanLand	Qld	EDIT
Ash	Phillips	Uncontaminated	Qld	EDIT
Spongebob	Squarepants	LandAuditors	NSW	EDIT
Post	Malone	NowTheySayNoContaminates	Qld	EDIT
Bruce	Springsteen	ImOnFire	Qld	EDIT

Home/Dashboard



Home

Last search completed successfully today at 9:56am.

Pending cases

#3872/9 - S. Pilgrim v. The World (Personal injury) — 04/05/2022
Gideon Graves Pty Ltd, Ramona F., Scott Pilgrim
New South Wales

Reviewed cases

#7896/88 - App-generated case name (Proceeding category) — 12/09/2022
An Auditor, Company That Audits Inc.
Queensland

#4123/77 - Godzilla v. Kong (Estate dispute) — 08/09/2021
G. Zilla, City of Hong Kong
Northern Territory

6.8. Artefact Agreement

QUT Capstone Project Deliverable Agreement

AACE

This document is an agreement between AACE and The Department of Environment and Science for the final deliverables and expectations for the website scraper.
This document will discuss the required features for a minimum viable product. Below are all features considered as 'Must-haves'.

Program Features

The minimum viable product (mvp) will allow the program/user to:

- Scrape data, based on a list of auditors, from the available Queensland court website
- Enable these scrapes to be made automatically
- Add auditors to the list of personnel
- Remove auditors from the list of personnel
- Manually scrape data using a manual search

The enhanced system will allow the user to:

- Add scrapers from other states available court websites
- Update the details of an auditor
- Disable scrapes being made automatically
- Review individual cases found in scrapes
- Login with department details

The stretch system will allow the user to:

- View the list of auditors currently being searched
- View an auditor's case history
- Provide the name of the user who reviewed a case
- Change the state search when manual searching

Deliverable Details

The deliverables provided will be the following:

- Five 'tutorial' videos to assist in common application uses which are;
    - Viewing the dashboard and marking cases as reviewed
    - Adding, updating and deleting auditors

    - Running a manual search
    - Creating a new user
    - Exploring the settings page
- Swagger API documentation, detailing the REST API provided by the server, such that future developers could easily integrate with it
- Written section for miscellaneous deployment and maintenance of the program, target at network administrators
- A zip file containing all program documents

Handover

The handover will occur on the 28th of October 2022. A link will be provided by email to the industry partner to a OneDrive folder containing all the outlined deliverables.