ArrowTrack: - Report

Project Vision

This project could be used by archers and archery clubs to track and help maintain the equipment they hold. This would also be useful to track the value of the inventory to assist in insurance calculations and track the items that may need replacing to help with budgeting.

Project Overview

This final report provides an overview of the development of 'ArrowTrack:' – a single-page application (SPA) to be used for inventory management by archers and archery clubs. This report covers how the project followed the plan laid out in the planning stage of the Software Development Lifecycle (SDLC), and subsequently evolved and problems and opportunities were faced.

As an archer and club committee member myself, I recognised the need for a tool that could be easily used to track owned items and their conditions for the purpose of maintenance and budget allocation for replacement equipment, for both individuals and clubs within the archery community, or even other sports communities.

ArrowTrack was designed with the goal of addressing these needs.

For individuals, the application provides a tool to monitor the condition of their equipment, ensuring longevity and performance, thus saving money in the long term on damages. For clubs where club-owned equipment can be spread across multiple indoor and outdoor ranges, or on loan out to club members trialling different bow setups, the ArrowTrack tool can be used to manage this in one centralised tool. ArrowTrack's total cost and by-location cost functionalities can be especially helpful to club treasurers and equipment officers, assisting in insurance estimations and budget planning.

Software Development Lifecycle (SDLC)

My approach to project management followed 'Agile' methodology. Throughout the development of the ArrowTrack project, I utilised GitHub and GitHub Projects as my platform for version control and project management; I used the project Kanban Board and Roadmap views available at [GitHub ArrowTrack Project](https://github.com/users/corey-richardson/projects/4) to manage the sprints. This includes columns for User Stories, Product Backlog, In Progress, On Hold, Done and Cancelled. By using an Agile approach to software development, I was able to iterate on features incrementally.

- Requirements Analysis
- Planning
- Research and Learning
- Implementation
- Testing

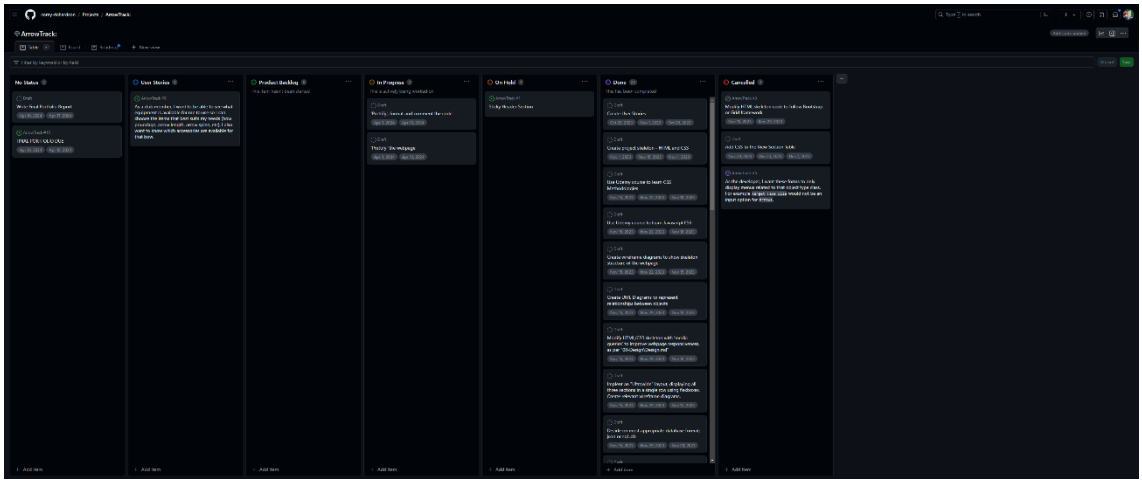

Figure 1: GitHub ArrowTrack Project - Kanban Board.

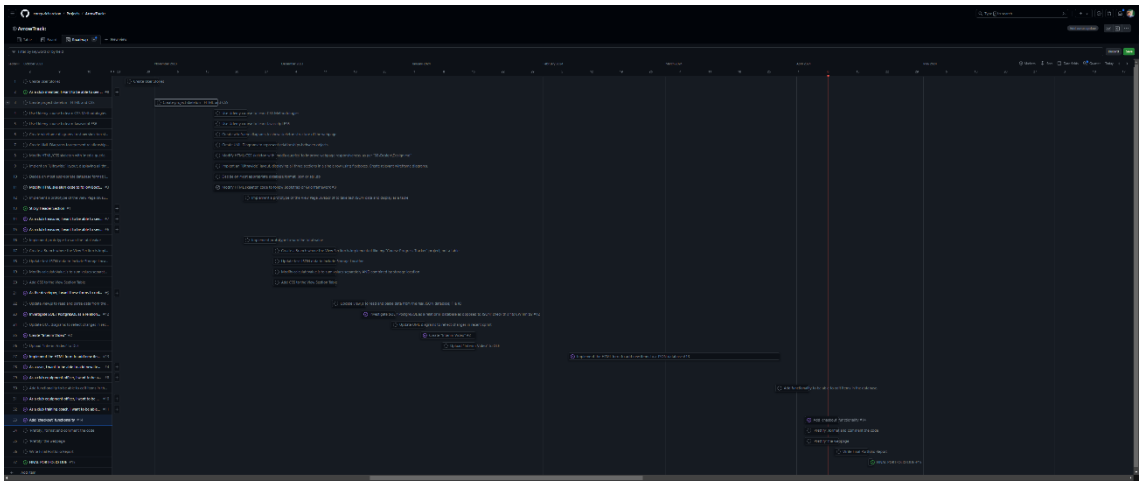

Figure 2: GitHub ArrowTrack Project – Roadmap.

The initial requirements analysis, planning, design, research and learning stages took place over the course of a month, serving as a foundation to build the ArrowTrack project on top off. During this time, user stories were created to outline the desired functionality of the web application. A significant portion of this stage was used to research and learn about technologies that would be used throughout the project, including HTML/CSS styling methodologies such as grids and flexboxes, JavaScript ES6 for web programming and JSON databasing.

Based on the design wireframes created during this stage and using knowledge acquired from research and learning, a project skeleton was created. This skeleton was used as a foundation which the subsequent sprints would build on, adding the actual functionality of the web application.

Features that were originally planned in this stage were later cancelled after discovering they were unsuitable for the project or no longer matched how a feature had been implemented. This iterative process derived from sprints determined how ArrowTrack evolved in line with its original outline.

User Stories

User Story	Status	Notes
As a club member, I want to be able to see what equipment is available for me to use so I can choose the items that best suits my needs (bow poundage, arrow length, arrow spine, etc). I also want to know which accessories are available for that bow.	Completed in part	The limitation of linking accessories to equipment was due to a limitation in the database structure; a flat-file JSON file rather than a relational database. This hindered integration leading to the incomplete representation of accessories as individual items as opposed to child-items. See: https://github.com/corey-richardson/ArrowTrack/issues/9
As a club treasurer, I want to be able to see the values for equipment stored at each venue/range as well as the combined value to understand what the club has in inventory for insurance calculations.	Completed	Client-side JavaScript file 'calculateValue.js' handles totalling calculations of all equipment by combined value and by-location value. See: https://github.com/corey-richardson/ArrowTrack/issues/7
As a club treasurer, I want to be able to see the condition of equipment in storage so I can prioritise spending towards items that are in direst need of replacing to help with budgeting.	Completed	All items have an attribute 'Condition' with options Great, Good, Okay, Poor and Unsafe. This allows individuals and clubs to prioritise their budget allocation to items in direst need of repair or replacement. Items marked as unsafe are dynamically highlighted in red using jQuery's CSS styling method. See: https://github.com/corey-richardson/ArrowTrack/issues/7
As a user, I want to be able to add new items via a HTML form.	Completed	Items can be added via a HTML form. See: https://github.com/corey-richardson/ArrowTrack/issues/4 and https://github.com/corey-richardson/ArrowTrack/issues/13
As a club equipment officer, I want to be able to add and remove items from the database so that I can keep track of the equipment available for members to use.	Completed	Items can be edited and removed via a HTML form. See: https://github.com/corey-richardson/ArrowTrack/issues/10

As a club equipment officer, I want to be able to edit/alter the condition of equipment as it gets used.	Completed	The “Condition” attribute of an item can be edited using the “Edit an Existing Item” HTML form. See: https://github.com/corey-richardson/ArrowTrack/issues/10
As a club training coach, I want to be able to 'checkout' club-owned bows to members on training courses to ensure they are using the same bow throughout (and potentially beyond) their beginner’s course.	Completed	The “Checked Out” attribute of an item can be edited using the “Edit an Existing Item” HTML form. When an item is checked out by a user, the background colour of the item’s container gets changed to a darker shade of grey using jQuery’s CSS styling method to indicate that the item is not available for general use. See: https://github.com/corey-richardson/ArrowTrack/issues/11 and https://github.com/corey-richardson/ArrowTrack/issues/14
As the developer, I want these forms to only display menus related to that object type class. For example, ‘Target Face Size’ would not be an input option for ‘Arrows’.	Cancelled	All items now share the same set of attributes, eliminating the need for distinct classes for different item types; this issue no longer applies to the use case of the web application. This change was done due to the non-relational structure of the database. By standardising all items to use the same attributes, data management is simplified. See: https://github.com/corey-richardson/ArrowTrack/issues/5

Problems Faced During the Development Lifecycle

During the planning stage, I had hoped to implement a relational database, where items could be linked together where required. This would allow accessories to be connected to the equipment they were attached to.

This would also allow different ‘classes’ of item, representing bows, arrows, targets, etc - all of which would inherit from a base class. It was determined that this overcomplicated the relationships for a flat-file database such as JSON, and would have required a backend handler, such as a SQL server, to manage. This would have fallen out of the scope of the coursework specifications.

As such, it was decided to instead implement a single class with attributes applicable to all items.

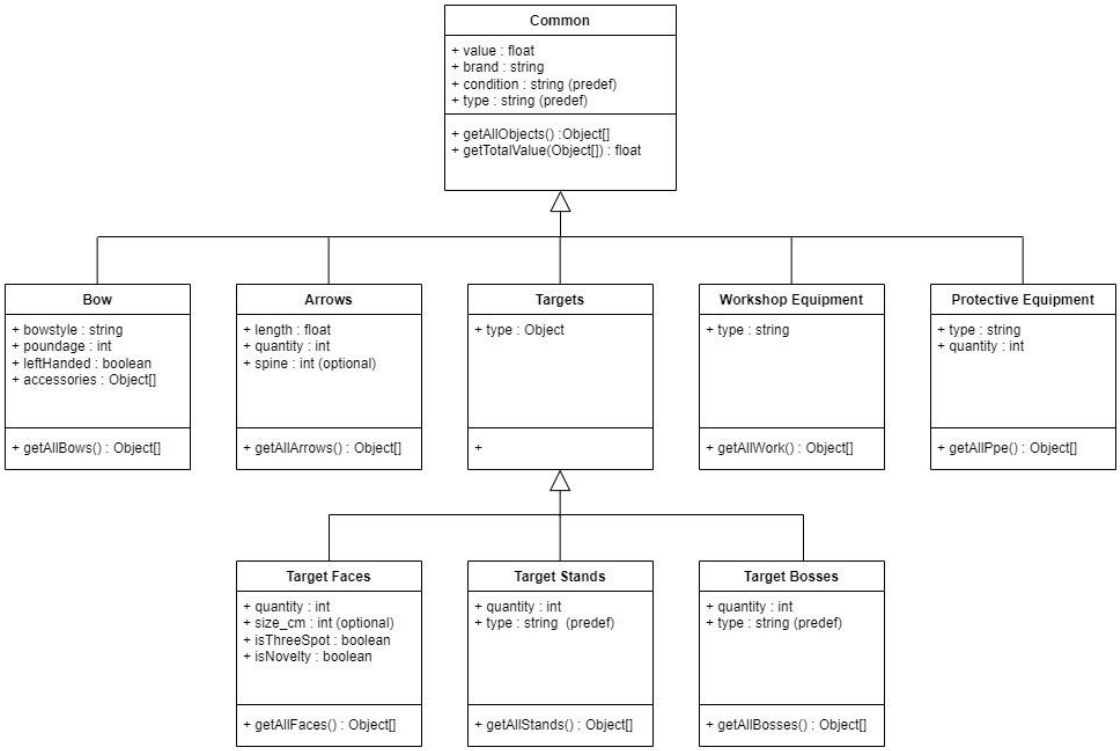

Figure 3: Original planned UML diagram for relational items.

Most sprints and feature implementations were assigned a timespan of one or two weeks, depending on the expected 'cost' of that feature. However, the longest and most costly feature took *eight* weeks to implement as described. This feature was the HTML form which would be used to add new items to the database. This issue arose as at the time the application was using a local JSON file to store item entries, which, using the Fetch API [0] could not be written to by the browser for security reasons – only read.

This led to me needing to find a new solution to how the database was persisted between sessions. To find this solution I turned to Dr Angela Yu's 'The Complete 2024 Web Development Bootcamp' course [1] with the expectation of using the module on 'Node.js' [2] and the 'filesystem' [3] JavaScript module to manage the reading and writing of the database. Upon reaching this module, I realised that it would not be suitable for this single-page application and would fall outside of the set coursework specifications, as it would require a server-side backend.

After this setback, I then found the solution implemented in the current iteration of the project: Browser Local Storage [4]. Data stored in local storage persists, even if the browser window is closed, unless explicitly cleared by the user or the application itself.

I still do not believe this is the optimal solution, however it does reach a minimum viable product solution. There is a local storage limitation of 5-10MB which for a JSON file and this use case, is unlikely to be surpassed. The 'IndexedDB API' [5] could be used here for better performance when storing large amounts of structured data, by using asynchronous transactions. IndexedDB also supports querying and indexing, which would be useful to search, sort and filter entries according to the attached attributes. For example, a club equipment officer could use this to sort entries by the 'condition' attribute to highlight items most in need of replacement or repair.

Design Document

Responsive Web Design and the Mobile-First Approach

Responsive Web Design (RWD) was a key design goal for this project. By implementing RWD principles, I aimed to ensure that the web application was accessible on a range of devices and that the functionality of the application would seamlessly adapt to various display sizes. Not only does this approach enhance the users experience by providing optimised layouts across devices, but it also assists with web accessibility. [6]

Web accessibility is crucial for catering to users who rely on assistive technologies such as screen readers and magnifiers. RWD provides accessibility by ensuring the displayed content remains usable across devices, whether a user is accessing ArrowTrack from a desktop computer with an ultrawide monitor, or a mobile phone with a small screen; all can interact with the application effectively.

Flexboxes [7] and Media Queries [8] have been used to implement RWD. By using flex containers, the arrangement of elements dynamically changes based on the available screen size. Media Queries are used to apply specific styling rulesets based on the current screen compared with the minimum targeted width thresholds. Following a mobile-first design approach, media queries were used to affect the styling and layout of the page with progressively increasing viewport sizes. Prioritising smaller, mobile screens – which the default styling does – improves the performance of the web application when it runs on these devices, which often have limited bandwidth when compared to larger devices.

On small screens with a width of less than 992px across the three content sections - View, Add and Edit - are styled in 3 rows across 1 column. In addition to this, the navigation header and footer are also adjusted to follow a single column design. When this threshold is surpassed, the header and footer expand, and the contents sections are split into 2 columns. In this layout, the View section spans 2 columns. For larger displays, such as ultrawide desktop monitors, all three sections are displayed in their own column. This is done to enhance visibility, usability and accessibility of the content.

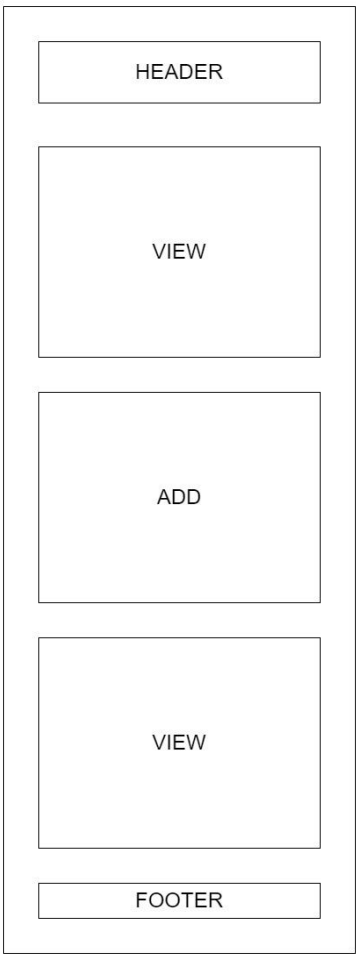

Figure 4: Wireframe diagram for screens less than 992px wide.

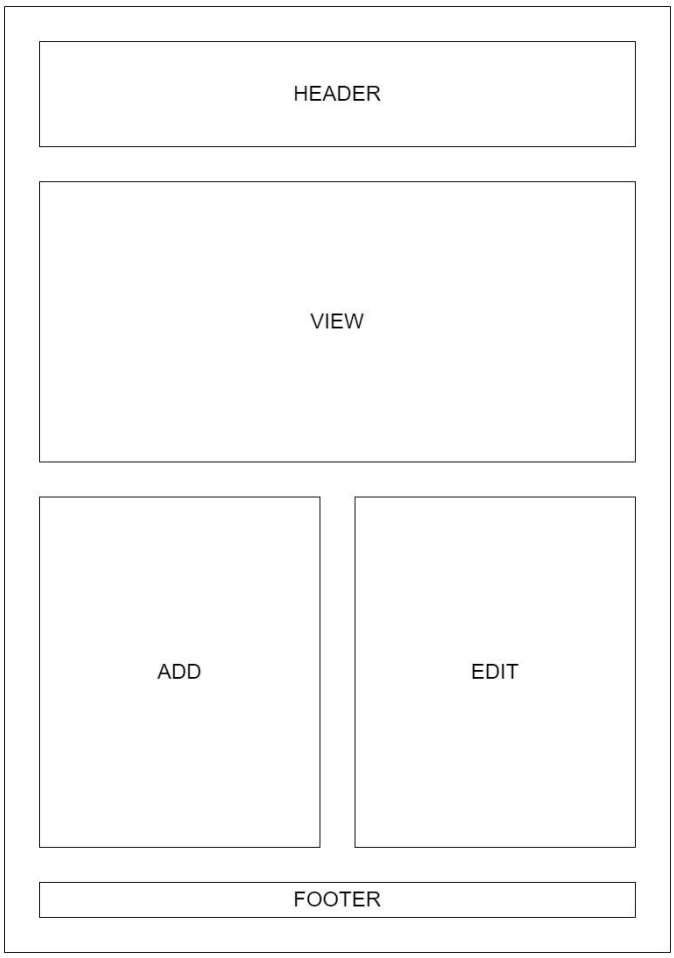

Figure 5: Wireframe diagram for screens between 992px and 2560px.

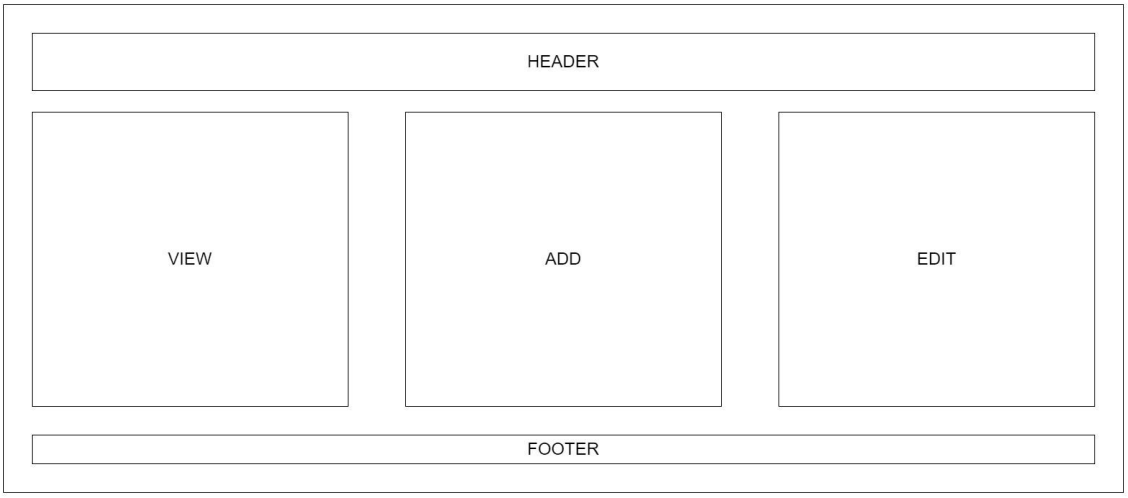

Figure 6: Wireframe diagram for screens more than 2560px wide.

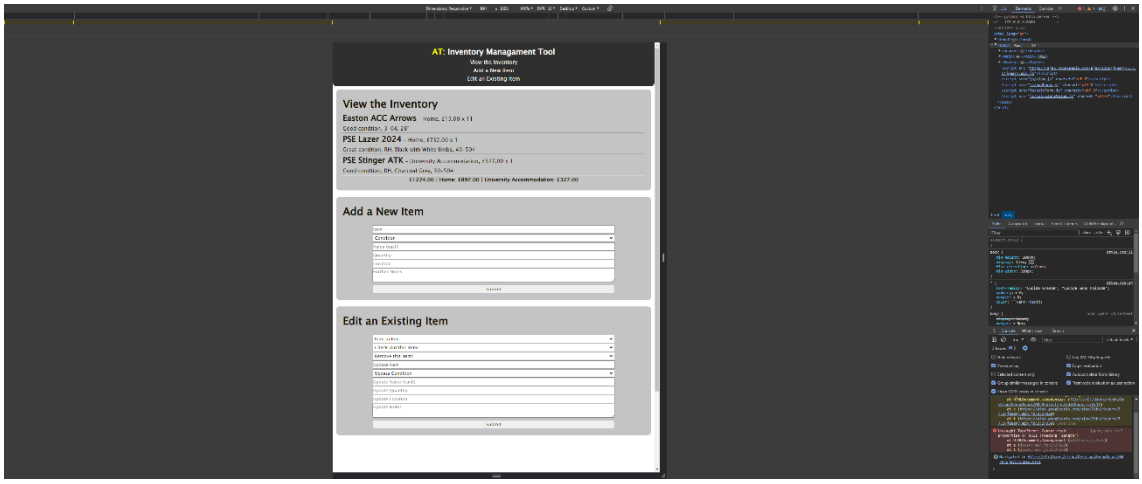

Figure 7: Web Application on screens less than 992px wide.

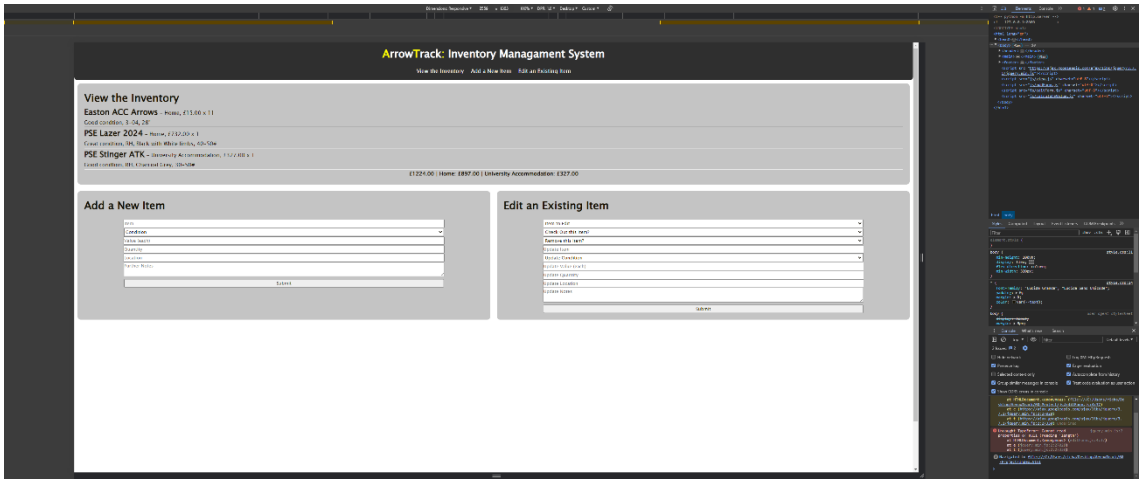

Figure 8: Web Application on screens between 992px and 2560px wide.

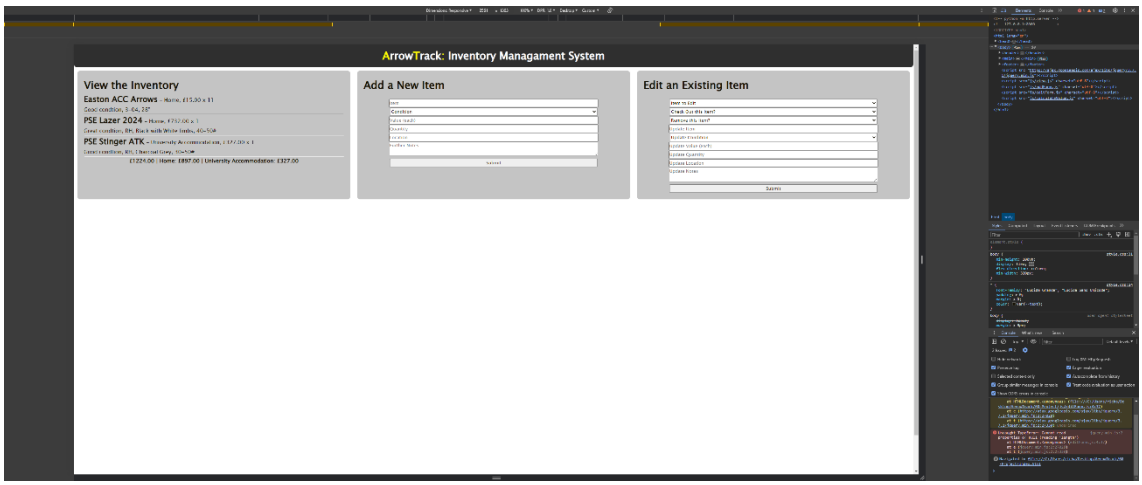

Figure 9: Web Application on screens more than 2560px wide.

Full-sized versions of these images can be found in **ArrowTrack/08-report/res/**

Colour Theme

In the planning and design stage of ArrowTrack, I used an online tool called 'Realtime Colors' [9] to experiment with a variety of colour choices and visualise them on a template webpage. As the web application is archery related, I wanted to select colours that would be appropriate for the theme, whilst still maintaining accessibility, readability, and being visually appealing to the user.

As such, I chose yellow as the accent colour, drawing inspiration from the centre of an archery target that all archers aim to hit. To ensure a visual balance between elements, I selected more neutral colours for the primary and secondary background colours: various shades of grey. These colours allow the accent colour to stand out, whilst also ensuring that text remains readable, enhancing user experience and accessibility.

These design choices were iterative, with my original colour selections reflecting a more "natural" colour theme to evoke the imagery of outdoor environments often associated with the archery theme. However, as the design progressed, the colour theme was refined to its current iteration to strike a balance between not only the thematic considerations, but also the accessibility of the content.

Web accessibility ensures that content can be accessed and used by a wide audience, regardless of impairments they may have. It is also a legal obligation in the UK under the Equality Act 2010 [10]. The Web Content Accessibility Guidelines (WCAG) [11] provide a framework to follow for ensuring accessibility.

	Text	Background	Primary	Secondary	Accent
Initial Theme					
Final Theme					

Other Considered Colours		
Yelverton Bowmen Blue	**Archery GB Red**	**Merlin Archery Orange**

UML Diagrams

Use Case Diagrams

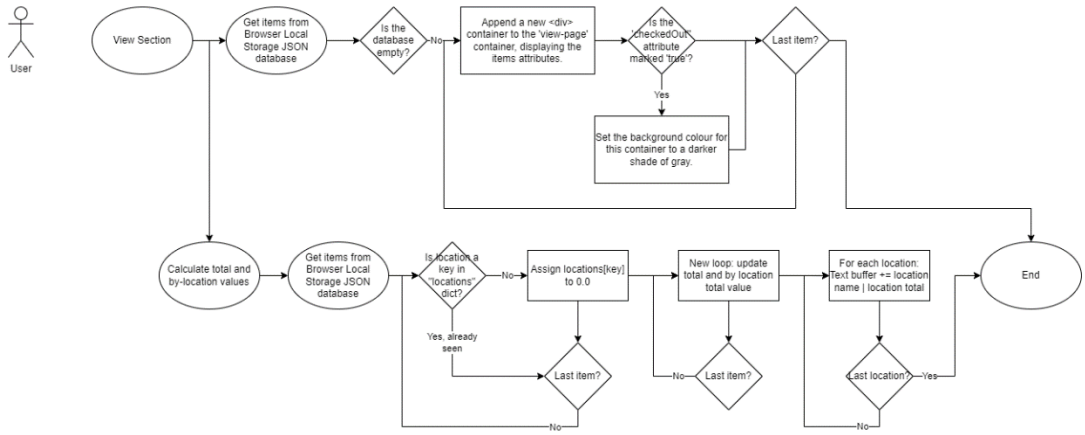

Figure 10: Use Case – View

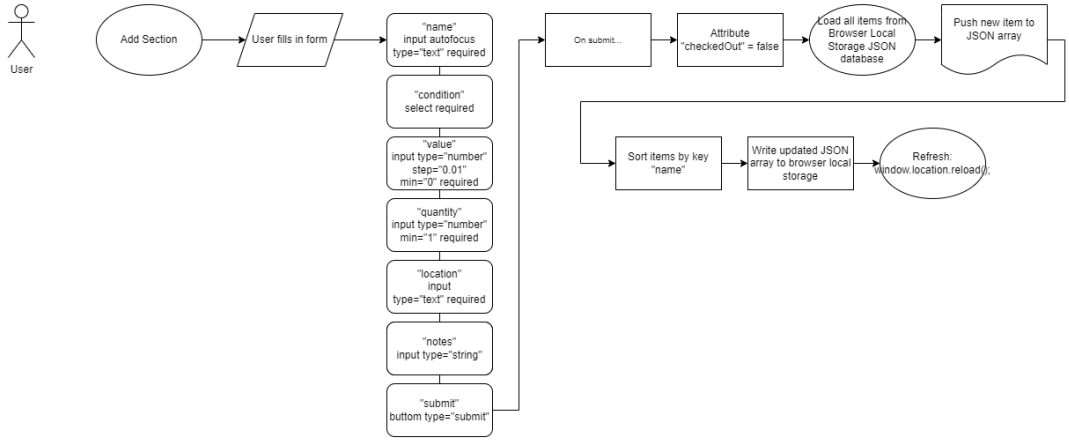

Figure 11: Use Case - Add

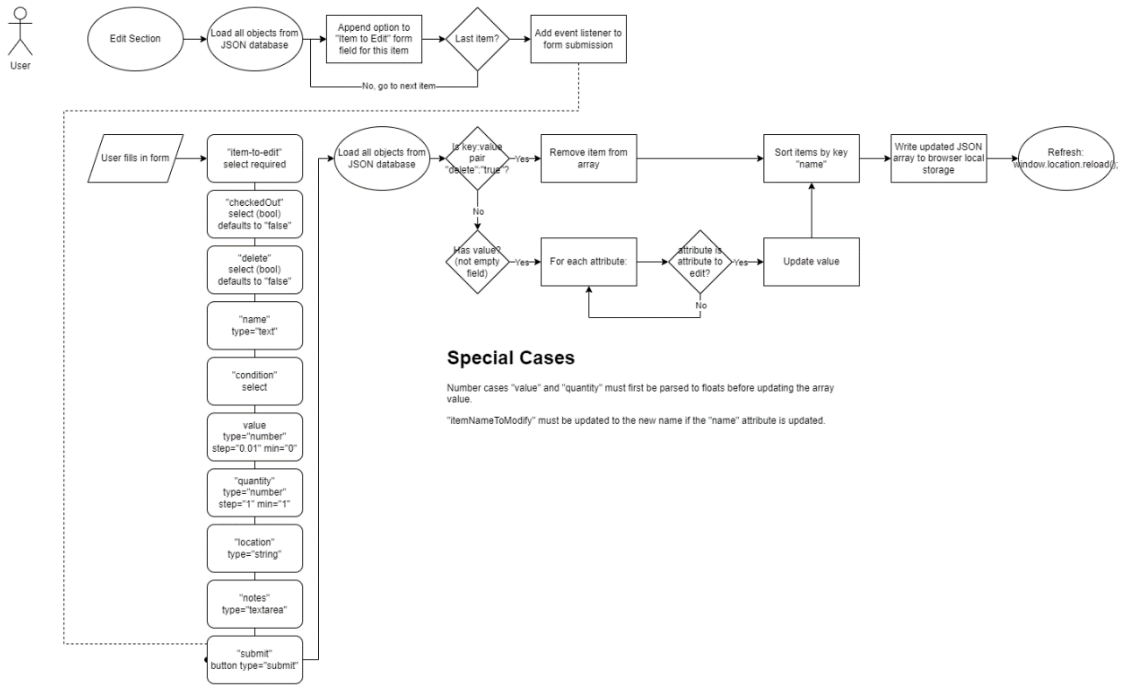

Figure 12: Use Case – Edit

Full-sized versions of these images can be found in **ArrowTrack/08-report/res/**

Class Diagrams

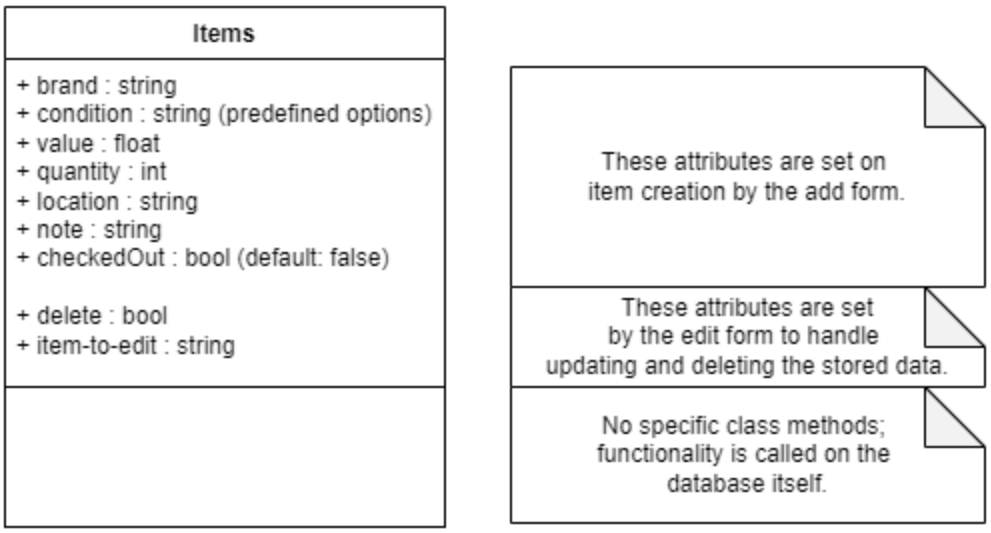

Figure 13: UML Class Diagram

A full-sized version of this image can be found in **ArrowTrack/08-report/res/**

Noted Issues and Constraints

As mentioned in the "Problems Faced During the Development Lifecycle" section of this document, I dedicated a substantial portion of my allotted time on this project for learning around the subject, including taking relevant online courses such as Dr Angela Yu's "The Complete 2024 Web Development Bootcamp" [1]. This course provided me with the knowledge and web development experience which has been crucial in the development of this project. Subjects covered include the different HTML/CSS styling methodologies, an introduction to client-side JavaScript ES6, an understanding of the Document Object Model (DOM) and how to manipulate elements dynamically with jQuery.

I had hoped this course would also be useful to find a solution to the writing to a database issue I was having by utilising Node.js file system libraries, however after reaching this section of the course I realised that I would not be able to use this topic for my coursework project, ArrowTrack. This came as a large setback as I had invested a large amount of time and effort into this potential solution before realising it was not feasible, then having to find an alternative solution to the database integration issue.

Figure 14: WakaTime Statistics for Course Based Learning

In addition to the technical challenges faced during the development lifecycle of ArrowTrack, another significant factor that delayed a large portion of its development was a stage of burnout. As the project progressed and technical challenges arose, the amount of learning required to resolve these issues expanded. With deadlines approaching for other coursework projects and whilst undertaking online courses to assist with all the different taught content between a variety of modules, and preparation for summer placement, I had to take a step back from the project.

Whilst the stage of burnout introduced a setback in the projects timeline and development I had envisioned, it provided me with valuable experience to understand the importance of time management and maintaining a healthy work/learning-life balance.

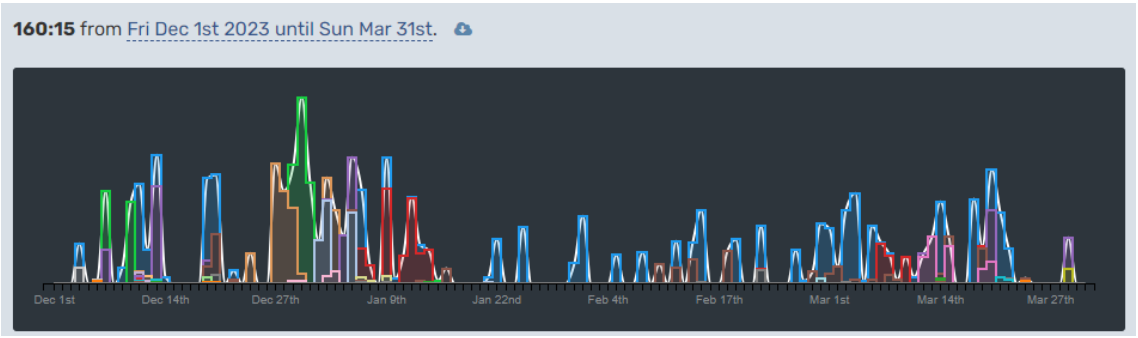

Figure 15: WakaTime Statistics, Visualising a Difference in Productivity Before and During the Stage of Burnout

GitHub Repository

The GitHub repository for the project can be found here: https://github.com/corey-richardson/ArrowTrack

Additionally, the WakaTime time tracker project page can be found here: https://wakatime.com/projects/ArrowTrack?start=2023-10-01&end=2024-05-31

References, Links and Further Information

'GitHub ArrowTrack Project' Available at: https://github.com/users/corey-richardson/projects/4 (Accessed 8th April 2024)

'GitHub Issue: HTML form to add new item` Available at: https://github.com/corey-richardson/ArrowTrack/issues/13 (Accessed 8th April 2024)

[0] 'Fetch API' Available at: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (Accessed 8th April 2024)

[1] Yu, Dr Angela 'The Complete 2024 Web Development Bootcamp'. Available at: https://www.udemy.com/course/the-complete-web-development-bootcamp/ (Accessed 8th April 2024)

[2] 'About Node.js' Available at: https://nodejs.org/en/about (Accessed 8th April 2024)

[3] 'filesystem API' Available at: https://nodejs.org/api/fs.html (Accessed 8th April 2024)

[4] 'localStorage Property' Available at: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (Accessed 8th April 2024)

[5] 'IndexedDB API' Available at: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API (Accessed 8th April 2024)

[6] web.dev 'Accessible Responsive Design' Available at: https://web.dev/articles/accessible-responsive-design (Accessed 8th April 2024)

[7] mdn web docs 'Flexbox' Available at: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox (Accessed 8th April 2024)

[8] mdn web docs 'Using Media Queries' Available at: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_media_queries/Using_media_queries (Accessed 8th April 2024)

[9] 'Realtime Colors' Available at: https://www.realtimecolors.com/?colors=130e01-ffffff-2f2f2f-c4c4c4-ffff00&fonts=Poppins-Poppins (Accessed 10th April 2024)

[10] UK Government 'Equality Act 2010' Available at: https://www.legislation.gov.uk/ukpga/2010/15/contents (Accessed 10th April 2024)

[11] W3C 'Web Content Accessibility Guidelines (WCAG) 2.1' Available at: https://www.w3.org/TR/2023/REC-WCAG21-20230921/ (Accessed 10th April 2024)