# ArrowTrack: - Report

Contents

[ArrowTrack: - Report 1](#_Toc163497554)

[Project Overview 1](#_Toc163497555)

[Software Development Lifecycle (SDLC) 1](#_Toc163497556)

[Problems Faced During the Development Lifecycle 3](#_Toc163497557)

[Design Document 4](#_Toc163497558)

[UML Diagrams 7](#_Toc163497559)

[References, Links and Further Information 7](#_Toc163497560)

## 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)

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.

|  |
| --- |
| Figure 1: GitHub ArrowTrack Project - Kanban Board. |

|  |
| --- |
| Figure 2: GitHub ArrowTrack Project – Roadmap. |

The initial planning, design and research and learning stage took place over the course of a month, serving as a foundation to build the ArrowTrack project. 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

## 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 don’t 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 (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. |

## UML Diagrams

### Use Case Diagrams

|  |
| --- |
| Figure 10: Use Case – View |
| Figure 11: Use Case - Add |

|  |
| --- |
| Figure 12: Use Case – Edit |

### Class Diagrams

|  |
| --- |
|  |

## Noted Issues and Constraints

As mentioned in the “Problems Faced During the Development Lifecycle” section of this document, I dedicated a large 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 topic this 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.

|  |
| --- |
|  |

Burn out.

|  |
| --- |
|  |

## 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/](https://www.udemy.com/course/the-complete-web-development-bootcamp/?couponCode=KEEPLEARNING) (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)