Mtrack Documentation

Overview

Mtrack is a simple entertainment-tracking website I built to help people keep up with movies and TV shows they're interested in. The purpose of the site is to make it simple for anyone to create watchlists, mark favorites, and explore new recommendations all in one place.

The website includes three main pages: a homepage, a movies page, and a shows page. Each page features interactive elements such as forms, tables, pop-ups, and dynamic media listings. Users can browse through titles, click on specific movies or shows to view trailers or information, and manage their personal lists. You can't currently save or add to the lists right now. So the watchlist and favorites do not save after reloading, making this a planned future improvement.

The main purpose of Mtrack is to create an enjoyable, and easy experience for users who want a simple way to manage their entertainment. The design is simple and helps users find what they're looking for quickly without feeling overwhelmed.

Coding Approach & Technical Decisions

When designing and building the site, I kept things organized by splitting the project into separate HTML, CSS, JS, and JSON files. Each page (home, movies, and shows) has its own HTML file so it's easier to manage and make changes. I used one shared CSS file so everything looks consistent. The JavaScript is also split up, with each file handling its own part of the site.

To avoid repeating code, I reused CSS classes for things like cards, navigation bars, and grid layouts. I also created reusable JavaScript functions for rendering the movie and show lists, updating tables, opening pop-ups, and handling smaller interactions, keeping the code easier to manage and work with.

For layout, I used a mix of Flexbox and CSS Grid so the design could adjust to different screen sizes. Earlier versions didn't look good on mobile, so I kept working on the layout until it felt smooth and more responsive. I also removed buttons that didn't have a use yet after getting peer feedback, which actually made the design better

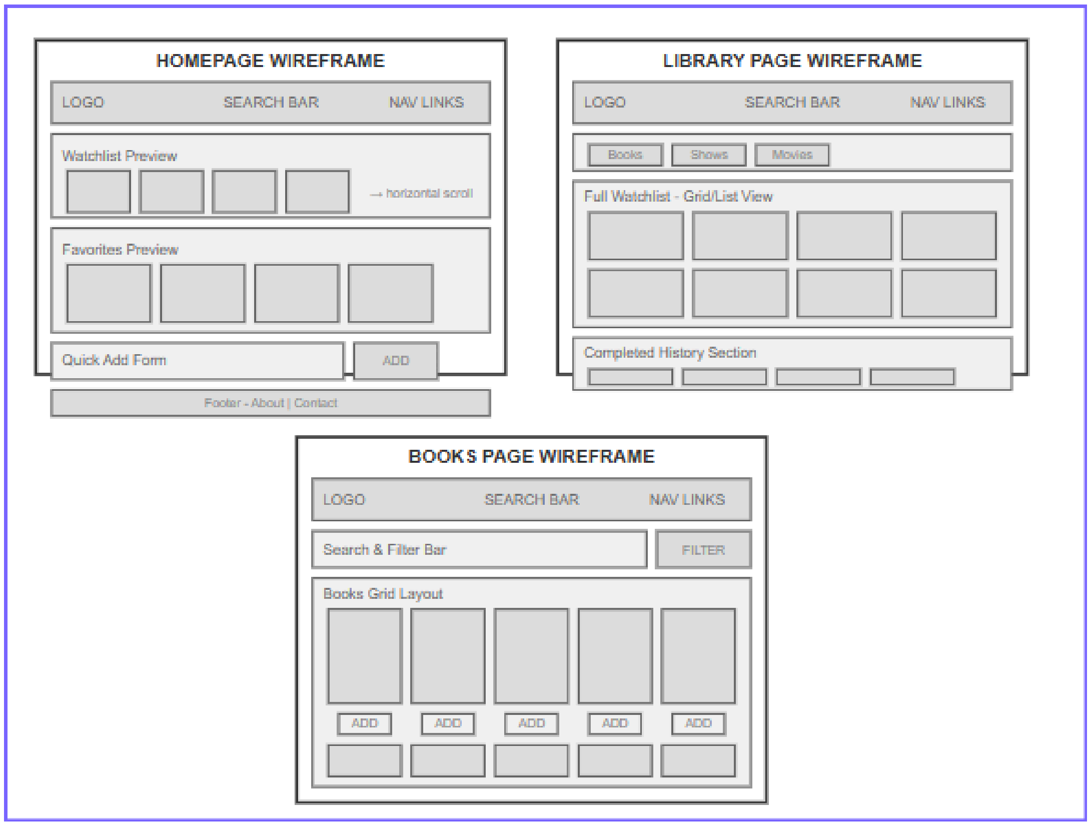
I tried to keep everything readable by using semantic HTML(header, nav, main, section, article, footer, table, for, etc) organizing JavaScript into smaller modules, and adding comments where things might be confusing. I wanted to make the code simple to understand and easier to improve later.

Course Concepts Integration

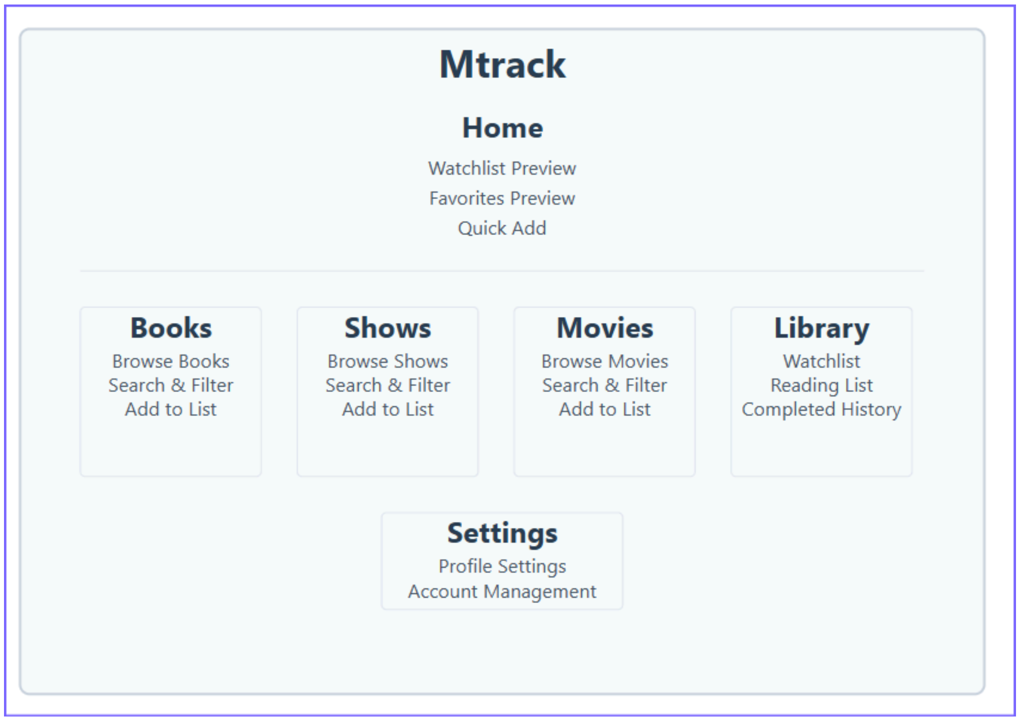
- Design:

  I chose to use the Agile method for my SDLC and started with planing a few wireframes and a sitemap, and created a timeline. I originally planned more pages like a library and settings page, but I ran out of time. Those ideas got moved to maybe future updates if I keep working on this.

Early wireframes: The wireframes below show my initial ideas for the homepage, library, and books page layouts. Each wireframe outlines the placement of key elements like the logo, navigation links, search bar, watchlist preview, favorites preview, and add forms. These visuals helped me organize content and plan user interactions before coding. The site did not fully end up like these wireframes though.



Early sitemap: The sitemap below illustrates the planned navigation flow and main sections of the site, including Home, Books, Shows, Movies, Library, and Settings. This helped me define the structure and relationships between pages, ensuring a logical and user-friendly experience. I didn’t get to some pages like books, library and settings.



- HTML:

  I used semantic HTML tags throughout the site to create a clear and accessible structure. For example, each page (homepage.html, movies.html, shows.html) uses <header> for the site title and navigation, <nav> for the main menu, <section> to group related content, and <main> to highlight the primary content area. Tables are used for listing movies and shows, and forms are included for user input, such as adding to watchlists or searching. This semantic approach improves both accessibility and SEO.

- CSS:

All styling is managed in a single, shared file (css/style.css). I established a visual hierarchy using font sizes, weights, and color contrasts to make headings and important actions stand out. The layout uses CSS Grid and Flexbox to create responsive grids for movie/show cards and to ensure navigation and content adapt smoothly to different screen sizes. Media queries are used to adjust layouts for mobile devices, ensuring the site remains readable and easy to use on any device. Consistent color schemes and spacing unify the look and feel across all pages.

- JavaScript:

  The JavaScript is modular, with each file in the js/ folder handling a specific feature: cards.js dynamically generates movie/show cards and attaches event listeners for pop-ups. Form-events.js manages form submissions, input validation, and user feedback. getMovies.js, getShows.js, and getLists.js fetch and render data from JSON files, updating the DOM in real time. recommendation.js provides dynamic recommendations based on user actions. showMediaDetails.js handles displaying detailed pop-ups for selected media items. Reusable functions are used for rendering lists, updating tables, and managing watchlist/favorite actions. Event listeners are attached to buttons and forms to provide instant feedback and interactivity, such as adding/removing items from lists or opening detail pop-ups.

- Accessibility:

  I ensured all images include descriptive alt text, making content accessible to screen readers. Color choices in style.css were selected for strong contrast, improving readability for users with visual impairments. Interactive elements like buttons and links are accessible via keyboard navigation (using tabindex and visible focus states). Forms and buttons have clear, descriptive labels, and pop-ups can be closed with both mouse and keyboard actions, supporting a wide range of users.

- Usability & UX:

  Navigation is consistent and intuitive, with a clear menu present on every page. The structure of each page is predictable, helping users find what they need quickly. Content is grouped logically, and dynamic features (like pop-ups and recommendations) are designed to enhance the experience without overwhelming the user. Feedback is provided for user actions (e.g., confirmation messages, visual highlights), and the responsive design ensures a smooth experience on both desktop and mobile devices.

Challenges & Problem-Solving

One of the main challenges was getting the layout to work on different screen sizes and coming up with a good design for the site layout. At first, the design would not resize smaller devices. I tried different approaches and eventually fixed it by restructuring the layout with Grid and Flexbox until everything fit better.

Another challenge was keeping my JavaScript organized as I added more features. It started getting messy, so I split code into smaller modules and created reusable functions, like the cards module that I can use for both movies and shows. This made everything easier to maintain. I also had to fix several accessibility issues, like missing alt text or buttons that didn’t show focus states.

Strengths & Areas For Improvement

The part of my project I am most proud of is the design layout and interactive features. Features like the movie/show pop-ups, media cards, and responsive layout help make the site feel modern and easy to use. The UI feels organized, and the browsing experience is smooth.

The area that needs the most improvement is user personalization. Without a backend, user lists can’t be saved permanently. With more time, I would add a database and user login system so users could keep different lists and settings across sessions. If I keep working on Mtrack, I’d like to improve the color scheme, upgrade the recommendation system, add better visual feedback on buttons, and do more accessibility testing. These would help make the site better and more complete.