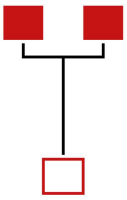
Web Technologies Report: Tournify



Summary

For our Web Technologies final project we decided to put together a useful tool that would simplify some of the operations of local sports teams. In particular, we plan to make use of this in collaboration with the University of Bristol Badminton Society.

The site we have made facilitates the storage and communication of information regarding sports teams and tournaments. It also offers a brackets tool that can be used to input team names and stake these teams against each other in knockout tournaments.

Mark Estimation

- A* for HTML
- A* for CSS
- A* for JS
- A for PNG
- A* for SVG
- A* for Server
- A* for Dynamic Pages
- A* for Depth

HTML

When we developed the first iteration of our website we produced an MVP consisting of static web pages that were built primarily using HTML5. Although this facilitated fast loading times and was an economical approach at developing a modest website, we decided to adapt this into a dynamic structure. We did this because it enabled us to experiment with other languages, program structures and methods of development. Moreover, it enabled us to keep our code DRY which is an essential part of the University of Bristol's House Style Guides.

Our static pages relied on the open-source CSS framework 'Bootstrap'. This contains CSS and Javascript-based design templates that allowed us to develop slick, economical and aesthetically pleasing design. It was particularly useful for typesetting, forms, buttons, interface components such as navigations bars and forms.

So as not to waste our efforts we decide to import the bootstrap developed HTML code into React and used both these frameworks for a 'Reactstrap' stack. This involved taking all of the static html and breaking it up into sections which could then be fit into an object oriented programming style. This was a challenge because we had to learn various new methods and grammars as the HTML used by React is slightly different to HTML5. Things like comments and container syntax are different. Moreover, we also had to come to terms with simultaneously interspersing HTML5 with functions so that we could dynamically serve our pages.

CSS

In addition to the powerful functionality offered by Bootstrap, React also reconceptualises how CSS styling can be implemented within a document. Much like in HTML5 and Bootstrap we made use of some plain CSS stylesheets and Inline Styling, these methods allowed for project wide changes and specific modifications respectively. Rather than rely solely on the props offered by React and Bootstrap we also developed our own to suit some of the less conventional aspects of our website such as our tournament brackets. Another particularly powerful tool offered by React is the

CSS-in-JS capability it has, this enabled us to use styling with specific components rather than on a document level.

JS

The ReactJS framework means working with a lot of JavaScript for front-end functions. Even the HTML code used in the project is processed as 'JSX' by React internally. The Login, Tournify and Register pages especially have a significant amount of JavaScript to create the dynamic aspects of the pages/api calls.

SVG

The SVG is mainly used for the logo of the site as well as the favicon that is displayed next to the tab, we designed and exported these using Inkscape. Although we did also try out Vectr which is an online alternative to Inkscape, we eventually decided to opt for Inkscape because of its open-source nature and the vast array of features it offered. For our logos and favicon grouping was used to colour in the shapes modularly.

Another SVG related challenge we faced was handled these vectors to make them automatically fit the dimensions of the containers that they were occupying. Typically they will either resize their containers or spill over the edge of them. To make the size responsively fit the toolbars of screens of various devices we were required to implement custom styling in CSS that controls the vectors handling.

PNG

We made use of the PNG file format for various images within our site such as backgrounds and photos. The background was heavily edited using gimpshop, an open source image manipulation program. We added additional layers and shading so that they did not dominate the pages and so that they could fit our colour scheme effectively.

Likewise, we made use of lightroom for some basic touchup of our photos in order to make the subjects stand out better and to make the lighting appear more natural. These tools are all important methods for making images on a website appear professional.

Server

The server primarily functions as an api for the database and for authentication of secure pages/ making tournaments. As mentioned later in the resources section it is built with a node+express backend. It is an API server which listens for requests and responds with JSONs with requested info/success and failure parameters. It's main functionality currently is providing a registration/login api and controlling access to secure information or preventing. SSL Certification and HTTPS have been used to secure transmission of critical information.

Dynamic Pages

The dynamic aspect in this website comes from the ReactJS front-end of the website. The header and footer are persistent and the middle content is rendered depending on the url accessed. The React App functions as a single page which is loaded in full and then the different elements are displayed depending on user actions. This makes the initial load slightly slower, however, as there is no requirement for multiple server transactions for every page that needs to be loaded. This makes the website much more responsive, as much of the content is essentially cached on the client-side.

Depth

The project aims to create a tournament making website to simplify organizing and managing tournaments at home as well as allowing users to publish them so others can view it.

There was a certain amount of flux in the design decisions of the project, and as a consequence we learned a variety of different approaches to web design. Our initial design was very static, simply a number of static web-pages made in bootstrap. The next iteration of the project was working on the design of a server-rendered dynamic website. A fair amount of progress was made towards learning and deploying the project in such a fashion however, we decided to not go for this, instead using ReactJS in the end, to gain a better understanding of single-page-applications. The final website design uses a react front end app which interfaces with an independent api server for dynamic functions, as well as login and logout (authentication) functionalities.

We think the overall aesthetic design of the website is strong, and the colour design works well with the competitive theme of the brand. We think additionally that the logo has turned out well. It is an SVG graphic made of a tournament bracket in the shape of a T to represent the first letter of the name ‘Tournify’. The API server implements a number of different database interfaces, and has good implementations of a JSON Web Token based authentication system. The frameworks used don’t have any known security flaws, and there is an HTTPS connection used to transmit critical data.

Resources Used

- React.js front-end framework. To make the website design more responsive and to further our learning we decided to use ReactJS to improve our understanding of Single-Page-Application style websites.
- Bootstrap is used in addition to react

- NodeJS + Express + SQLite for the API server. Express was used to minimise boilerplate code required and simplify routing. In addition it has strong support for middleware integration.
- PassportJS - PassportJS was used for login / credential authentication using a JSON Web Token (JWT) strategy. It provides support for seamless authentication of web tokens as middleware for express functions.
- Credential is a node module which provides functions for secure hashing and authentication of passwords so that they can be stored in the database.
- OpenSSL is used to create ssl certificates for https testing.
- The code for the core tournament logic is adapted from an online repo. The code was modified significantly to fit into the existing framework. The api connection system was set up entirely by us, however.