EECS 448 - Project 4

Fantasy Football (Team 8)

Garrett Mills, Alec Horlick-Mills, Evan Powell, Lucas Brakenridge, and Javier Barea-Lara

Repository available here:
https://github.com/EECS-448-Battleship/fantasy-fooball-backend

Deployment Plan

Our fantasy football application is a web-application built on Node.js using a framework called Flitter. It uses a MongoDB-Flitter-Vanilla-Node stack, so the deployment is reasonably straightforward. There are a few requirements, as we'll detail.

For a small-to-medium deployment, a single server that runs both the database and the main application should be sufficient. First, we will obtain a virtual server from a cloud hosting provider (we used SkySilk for this project) with at least 512MB of RAM and 10GB of available disk space. For our small deployment, this cost approximately $5/month, or $60/year. (For larger deployments, a distributed system, or even physical server infrastructure might make the application more robust and scalable.)

The server should run some form of a Linux OS. We used Ubuntu Server for our deployment. Once you have the server, it's best to set up the domain name to point to that server's IP address (eecs448.garrettmills.dev). We used a subdomain for free, but obtaining a reasonable domain shouldn't cost more than $30/year.

Install a modern version of MongoDB on the server (4.x or newer). This is required by the application for persistent storage. Then, install an LTS version of Node.js (14.x or newer). Clone the application from the project repository using GIT (which you'll need to install on the server) and copy the example environment config file to ".env" and update the values to match the server's environment. Specifically, modify the database host, database name, and database credentials as well as the app URL to match the domain name.

Next, you'll need to use CertBot to obtain a free SSL certificate for the domain you chose (our domain is HSTS). This can be done using the "certbot" command-line utility. Copy this certificate to a safe location on the deployment server and modify the environment config to point to the certificate and certificate key. This will cause the framework to serve the application using TLS. We're counting this as a 'required' step since any modern web application should use TLS.

Now, for security purposes, you'll need to enable the firewall on the server and allow incoming connections ONLY on port 443 (the application's HTTPS port). Once the application is ready, run the "setup.js" file to pre-seed the database with player information and weekly player statistics which are used by the game to calculate fantasy team scores.

Thus, the base cost to deploy the application to a minimal environment is less than $100 for the first year.

The market for our application is open to a broad range of sporting and statistics enthusiasts, since we provide a few additional statistics about the players in the game. The game progresses in a timeline from draft-season, through several weeks of gameplay. The team scores are accumulated and teams are ranked in a league. The competitive nature of the gameplay will likely appeal to those who like to gamble as well.

Because our application is a web-app, this means that there is a very low bar to entry for the end users. No app to install, no software to download. This should help increase user engagement with the game. However, this does mean it might be less accessible to those in developing areas, or those without ready access to broadband internet.

Integration Plan

For the continuation and final step for our project Fantasy Football, it was time to put everything together using an integration strategy. We had our prototype finished, which we used JavaScript both for the front-end, with HTML and CSS and the Vue.js framework, and for the server-side part using the framework Node.js, a popular free and open-source framework that allows to quickly and efficiently deploy run-time back-end environments for your web app servers, along with the flitter web framework, sportsdata.io, and SkySilk. First, for Project3, our task was to design a prototype that was logically functional that can later be used as the base for the functioning Fantasy Football app. First, Garret designed and built the skeleton for the prototype of the project, choosing what tools such as libraries, APIs and frameworks, and languages we would use for our project.

We implemented a top-down integration strategy for our project. We started by creating the different components that the project would need in order to make a functioning game. Then, we would join those components between each other using different tools and frameworks built specifically for those tasks. So, in a way a lot of our logic was functioning behind these frameworks that helped our processes. Each team member had the task to implement a different component of the game, and for the front-end, we created simple web UIs with basic functionality for the components of the game such as the Login, Draft, addPlayer, myTeam, LeagueStandings and other pages. Vue.js was used to wire up the front-end and to organize the different components of the game as we did on our Project1 with the Battleship game. For the back-end part, we used Node.js as our primary framework for the server-side part, along with Flitter web framework, to interface with our virtual database and feed data to it through the endpoints.

Person Hours Estimate

Looking back to Project 3, It took a total combined 35.5 hours. Because project 3 was a demo and was only the front end of the project. We had a total of 1324 lines of code in project 3. We will divide that by 35.5 the total hours it took to complete. That leaves us with 37.3. Then divide that by all 5 group members. Leaving us with 7.5. We said that it would take roughly the same amount for project 4, because we finished our demo of the front end and thought the back end would take roughly the same amount of time to complete. By using the LOC method we said that it would take roughly 35 hours to complete project 4.

Maintenance Plan

Our objective was to create a real-time functional fantasy football game web application with real data pulled from NFL sites. Even though it is not up to par to the top fantasy football web apps, our game does have the necessary and core components needed to run a fantasy football game. You can create your own account, draft your own team and players, get real and updated scores of the games and stats of the players as you play. Most of the bulk of the code was done using free and open-source frameworks. Because we only prototyped and built a functioning fantasy football app, we didn't use any paid software, libraries and frameworks, as there was no need for it. With our back-end being not so heavy yet, we didn't find a good reason yet to buy servers and a proper database and store it in a server in order to make our game functional. For that, we used the open source frameworks to our disposal to make it work as intended. Right now, the game is not yet stored on a domain of our own running on a hosted server, but rather on Garret's own virtual server, given that we still don't need to do those things yet.

Our next steps would be to buy a specific domain for our site and buy a hosting service to run our site. These will allow us to run the program straight from our domain and make it 100% public to everyone. Because this is a web app, it doesn't need any sort of downloads or fees to host it to a store or something like that, as the game is completely free for everyone to use. Right now, the costs to run this game are pretty straightforward given the nature of the tools that we used to build it, we ran this game on a small hosting server as our app is still very light and handles not too much requests given the amount of users it still has:

Domain: $16/year (personal specific domain)

Hosting: $60/year

If we want to scale this game further for public availability and intended for mass use, we're going to have to make some additions in terms of development, costs and tools used. In order to further test this game and add more functionality like tier-1 fantasy football apps, we're going to have to hire more developers to further improve the game and make it more accessible and user-friendly to everyone. We believe as far as salaries go, we won't need to spend much because of the nature of how fantasy football games work, there's not really much complex functionality. And plus, because this is an entry-level game done by us, we don't need the best of the best.

A salary of $1000/month per developer (3 or 4) should be enough, half of them working on the front-end to make it user-friendly and improve the user experience, and the other half working on the backend of the game, improving scalability of users and how the data is managed, to avoid bottlenecks and keep efficiency of the site.

There's going to be a time in which we may decide to port our game to more higher-scale tools to use our databases with that may have fees and costs to use them, however, given the state of web applications, there are hundreds of tools available that make it pretty much free to handle a game such as fantasy football without spending thousands of $$ on the backend part. Therefore, that's something that needs to be decided next year or in the coming future, but right now, salaries would be the bulk of the maintenance costs. Keeping in track that the site runs correctly, data is correctly fed, and users are able to have a good experience playing the game.

Accounting of Actual Person-Hours

Date	Person	# Hours	Task
2020-11-01	Garrett	2	Set up repo w/ framework and start some basic API endpoints
2020-11-04	Garrett	1	Modify front-end prototype from project 3 to work with backend
2020-11-04	Garrett	1	Add logic to populate teams & players in front-end from the API
2020-11-04	Garrett	2	Write patch to populate data from NFL APIs
2020-11-05	Garrett	1.5	Finish save logic for my team page, write API backend for draft page
2020-11-07	Garrett	1	Write patch to schedule weekly matchups
2020-11-07	Garrett	0.5	Write patch for scoring weekly matchups
2020-11-07	Garrett	0.5	Update stat model & patch to include some stats about the player
2020-11-07	Evan	1	JSDoc commenting, Front and back end
2020-11-07	Garrett	0.5	Write league standings logic & hook up to front-end
2020-11-08	Garrett	1	Start writing tests for Models in backend
2020-11-08	Garrett	1	Finish JSDoc comments, prepare for auto-generated documentation
2020-11-08	Garrett	0.5	Move documentation folder to correct dir and set up doc generation
2020-11-08	Garrett	0.5	Deploy project to demo server
2020-11-08	Garrett	1	Finish writing tests for front-end
2020-11-08	Garrett	0.5	Finish writing tests for back-end
2020-11-08	Evan	0.5	Video
2020-11-08	Alec	1	Testing for bugs
2020-11-08	Alec	1	time estimate
2020-11-08	Lucas	1.5	Integration Strategy
2020-11-08	Javier	1.5	Fix Integration Strategy
2020-11-08	Evan	1.5	Testing for the backend
2020-11-08	Lucas	2	Maintenance Plan
2020-11-08	Lucas	0.5	Video
2020-11-08	Javier	2	Finish Maintenance Plan Essay
2020-11-08	Javier	0.5	VIdeo Clip - AddPlayers
2020-11-08	Alec	0.5	Video Clip -League page

Defect Tracking Tool

For our defect tracking tool, we used a standard Google Sheets document with columns for the various dates found, finder, and notes for bug & fix.

The document can be viewed here: https://garrettmills.dev/go/448-bugs

But, for a quick reference, here's what it looked like at the time of submitting this document:

Date Found	Found By	Date Fixed	Fixed By	Bug Notes	Fix Notes
2020-11-03	Garrett	2020-11-07	Garrett	Need to hide/show draft page and add players page depending on whether it's draft season or not	Added a /status endpoint - if the endpoint shows that it's draft season, hide the correct menu items
2020-11-05	Garrett			Add Players & Draft Board pages take forever to load because they render all 2k+ players at once	
2020-11-07	Garrett			Remove unauthenticated endpoints for listing all teams & creating a new team	
2020-11-08	Garrett			No way to log out - add a Log-Out button to the navigation bar	
2020-11-8	Alec			Shouldn't be able to place players in possitions they dont play for the my team page	
2020-11-8	Alec			Don't have a player position on my team for kickers.	
2020-11-08	Alec			When we update the weeks not all players get the points that they scored each week	
2020-11-08	Alec			When a team wins it doesn't update to the league page	

Test Suite

Our test suite is in-repo and is broken down into front-end and back-end tests.

You can run the test suite by cloning the repo, installing the dependencies, and executing the command:

```
yarn run test
```