Front End Documentation

The following document contains information about some of technologies used for the Front End development of the LIFE League Web game.

Head Developer: Johnny Goulbourne
Contact: johnny.goulb@gmail.com
Linkedin: https://www.linkedin.com/in/johnnygoulb/

Notable Technologies: Nuxt, Pinia, Axios, PhaserIO

Refer to comments within the source code to see how functions work.

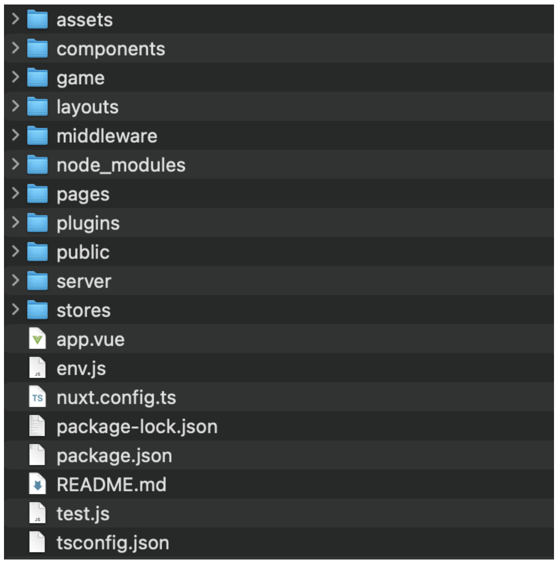
For local development, a ".env" file is required to connect to the backend server. It will be included in a separate file when submitting the final deliverables. Refer to the "README.md" file on how to install the frontend for local development.

Nuxt

Nuxt is an open source framework that makes web development intuitive and powerful. Create performant and production-grade full-stack web apps and websites with confidence.
Link: https://nuxt.com/

Nuxt is a superset (extension) of a popular JavaScript framework known as Vue.JS. It uses modern technologies that enable developers to deliver responsive and performative web applications. Nuxt/Vue is easy to learn, especially for those with backgrounds in other popular frameworks like React or Angular.

Within the "frontend" directory in the source code, you can find the entirety of the website. This includes all assets, game files, and all frontside functionalities.



```
assets
components
game
layouts
middleware
node_modules
pages
plugins
public
server
stores
app.vue
env.js
nuxt.config.ts
package-lock.json
package.json
README.md
test.js
tsconfig.json
```

The following link describes how each of these directories are used within the Nuxt environment.
https://nuxt.com/docs/guide/directory-structure/app

In the "pages" directory, you'll find ".vue" files that make up every page on the website. ".vue" files are documents that contain all HTML, CSS, and JavaScript code for a webpage. ".vue" files can be built as entire pages or be used as pieces of reusable code. The reusable code would be considered a component and placed in the "components" directory.

The "middleware" directory is for code that runs between every page change. There is only one TypeScript file in there "auth.global.ts" which is used to check if a player is authorized to view a page. Authorized meaning if they are logged in or not.

The "plugins" directory contains third party technologies used to extend the functionalities of the site. The two plugins currently used are Axios and Pinia. Pinia is documented later in the document.

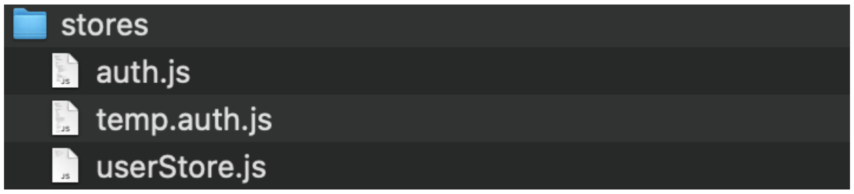
Axios is a promise-based HTTP client that I used to communicate with the backend server. It's in the plugins directory so Nuxt can use the Axios object globally.
Link: https://axios-http.com/

Pinia

The intuitive store for Vue.js. Type Safe, Extensible, and Modular by design. Forget you are even using a store.
Link: https://pinia.vuejs.org/

Pinia is used to store information in a persistent state so when the user refreshes a web page, the information that is tracked will remain. This program is essentially acting as a session and uses cookies within its code.

Within the "stores" directory, there will be stores for authorization. Despite the file name, temp.auth.js is the most important of all of these. The store directly communicates with the backend server to retrieve account information and save necessary information from the server to a local cookie so the frontend can access this cookie's information and display it throughout the site as needed. The cookie will also act as an authorization token.



```
stores
    auth.js
    temp.auth.js
    userStore.js
```

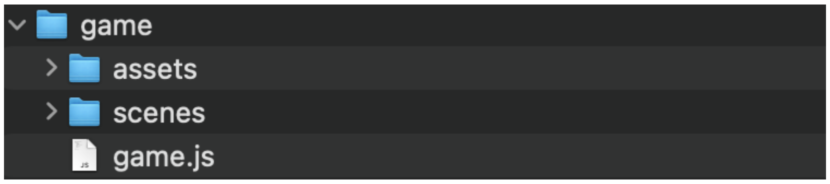
If time allowed, these stores were going to be used to track player progression.

At the moment, the stores currently handle basic authorization cookies for login/registration.

PhaserIO

A fast, free and fun open source framework for Canvas and WebGL powered browser games.
Link: https://phaser.io/

PhaserIO was used as the game engine of choice due to time constraints, ease of use, and learnability. The following directory "game" can be found within the frontend file structure.



```
game
  assets
  scenes
  game.js
```

The gameplay was intended to be a story-based, choose your own adventure type of game. An educational aspect would also be included into the game in order to teach players how to handle athletic life between high school and college.

All game assets (images, sounds, etc) are located within the "assets" directory.
The "scenes" directory contains JavaScript files that are essentially the scenes that the player will see during the game. As of 11/13, a small portion of Act One has been made.

The current implementation of the game was made in a way so if the game were to be completely redone, the process of doing so would be fairly simple. If the game were to have a future, I would highly recommend looking into technologies like Unity. Link: https://unity.com/.