Back End Documentation

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

Github Repo (Source Code): https://github.com/calmhand/LIFELeague-x-NMI-Capstone

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

Notable Technologies: Node.JS, Express.JS, Prisma

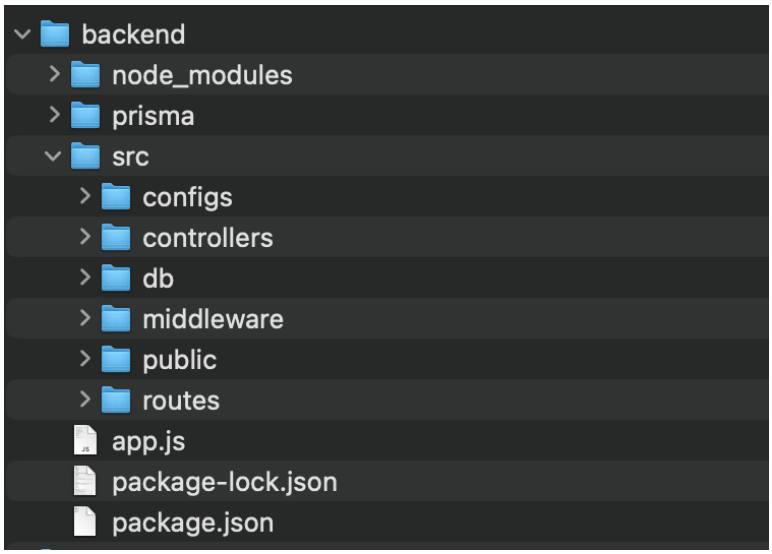
Refer to comments within the source code to see how functions work.

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.

Node.js + Express.js

Node.js is an open-source, cross-platform JavaScript runtime environment.
Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
Link: https://nodejs.org/en/, https://expressjs.com/

Within the "backend" directory in the app source code, you can find the entirety of the back end functionality. Express.js is used to create an API (Application Program Interface) which is essentially a program that allows the frontend to contact the server to either run code or connect to the database. The farthest development has reached in the backend was a simple login/registration system. The next step in development was to implement routes for admins to control different parts of the frontend.



Currently, the backend handles authorization using JWT (JSON Web Tokens). *JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.*
Link: https://jwt.io/

This API also uses Prisma to leverage connections between the backend and the database (mySQL). For information on Prisma can be found later in the documentation.

Prisma + mySQL

Next-generation Node.js and TypeScript ORM (Object Relational Mapper).
Link: https://www.prisma.io/

From their website, “The Prisma schema is intuitive and lets you declare your database tables in a human-readable way — making your data modeling experience a delight. You define your models by hand or introspect them from an existing database.”

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

model AccessKeys {
  keyId       Int     @id @default(autoincrement()) @db.UnsignedInt
  accessKey   String  @db.Char(6)
  description String? @db.Text
}

model AccountStatus {
  statusId Int    @id @default(autoincrement()) @db.UnsignedInt
  status   String @db.Char(255)
}

model AccountType {
  accountTypeId Int    @id @default(autoincrement()) @db.UnsignedInt
  accountType   String @db.Char(255)
}
```

Prisma was used to reduce time in having to constantly type out lengthy queries to the local mySQL server. It also allows us to quickly write and update tables from just a single file rather than having to write them out. Prisma also provides a way to create backups in case of any errors that are made along the way.

As mentioned above, for local development, Prisma will need to read from a “.env” file to connect to the database. This ".env” file will be provided in the submission of our final deliverables.