**MMO Manager** - IT2830 Final

By Gerard Lucas

[Video Walkthrough](https://www.youtube.com/watch?v=F9ZlS_k0NhE)

MMO Manager allows authorized users to add and update characters and npcs for an MMORPG. It is split into two separate projects, a backend server using nodejs and express, and a front-end react single page application that acts as the user interface.

Back-End

This project lives in the "backend" folder. To install, cd to the backend folder and run:

npm install

To start the server cd to backend folder and run:

npm start

The backend project is an express server running on nodejs. It stores player information in a Mongo database. The connection string is configured in the .env file, along with the port and session secret key. Sessions & authorization are managed by Passport and express session and stored with an encrypted password in the database via Passport Local Mongoose. Mongoose is used throughout the project to communicate with the database. I used the CORS module to allow my front-end to communicate with my backend across different domains, and nodemon is used to automatically push any changes during development. API endpoints are configured in the controllers folder and added to express routing. Models & schemas for mongoose are in the models folder. The server.js file is the main entry point of the application. The application requires authentication for all requests aside from counts of objects, and returns 401 responses for any unauthorized users.

The project uses the following packages:

* Express
* Express-Session
* Express Async Handler
* Mongoose
* CORS
* Passport
* Passport Local Mongoose
* dotenv (dev)
* nodemon (dev)

Front-End

This project lives in the "frontend" folder. To install, cd to the frontend folder and run:

npm install

To start the server cd to frontend folder and run:

npm start

The front-end is a react single page application created with the "Create React App" tool. a more detailed README can be found in the frontend folder that is specific to create react app built projects. The main entry point is index.js in the src folder, but much of the app lives in the App.js file, including routing. The application checks for 401 unauthorized responses and redirects to the login page. Pages exist for registration, login, view all characters, manage a character, view all npcs, and manage an npc. There is also a landing page the shows a welcome message and a count of each type of object in the database. Each page is used as a shell around embedded components, and props are passed down from the App.js file.

The project uses the following packages:

* React
* React Bootstrap
* React Router
* React Router Bootstrap
* React Icons
* Axios