**Develop**

* **Config**
  + **Middleware**
    - isAuthenticated.js
      * This file is used to restrict routes and not allow users to see pages unless they are logged into the site
      * If the user is logged in they will be directed to an internal pages
      * If a user is not logged in they will be automatically directed to the login page of the site
  + **Config.json**
    - Keeps all information for the database configuration
    - You will edit the development section to add in your database credentials such as: password, database name, host and port
      * This will be used for testing locally
  + **Passport.js**
    - Requirements:
      * Requires passport
      * Requires the models folder, use: var db = require("../models"); to require all files within this folder
    - Communicates with passport and tells it to use a Local Strategy, which means to login with an email and password
      * An email is required for login
        + If the email if not found a message will alert the user
        + If the password entered does not match the saved email a message will alert the user
        + If the email and password are correct then the user will be allowed to login
      * Authentication
        + Use sequelize to serialize and deserialize users login credentials

These credentials will not be saved directly into the database but rather saved as encrypted information

* + - Export!
* **Models**
  + **Index.js**
    - Requirements:
      * FS, Path, Sequilize
      * Requires entire config folder using: require(\_\_dirname + '/../config/config.json')[env];
    - Used to run sequilize for all user information that is entered
    - Created a model for each new user and puts the information into an empty object
    - Export!
  + **User.js**
    - This file is used to create a function that will create encrypted user information from the login credentials added in the index.js file
    - Requirements:
      * Need bcrypt to hash passwords for encryption
    - Creates User Models
      * The first object is to check if the email is valid
        + Not null, unique and is an email
      * The second object is to check if the password is valid
        + Not null
    - Next we will run a function to create a custom method and will check and see if the password entered by the user can be matched the hashed password
    - A Hook is then created to run before a user is officially created and will automatically hash the password that is entered
* **Public**
  + JS
    - Login.js
      * Pull references from form inputs
      * Login Form Function
        + Validates the email and password that is entered into the form
        + If an email and password are entered

The form is cleared

A post call is sent to the api/login route and if successful the user is redirected to the members page

Otherwise an error is logged

* + - Members.js
      * A Get request that figures out which user is logged in based on their member-name pulled during login
    - Signup.js
      * Pull references from form inputs
      * Signup Form Function
        + On click, validation will run for the email and password that were entered by the user to be sure no fields are blank

If email and password are present, signup function will run

* + - * + A post call is sent to api/signup route and if successful the user is redirected to the members page
        + Otherwise an error is logged
  + Stylesheets
    - CSS Styles
  + HTML Files
    - Login.html
    - Members.html
    - signup.html
* **Routes**
  + Api-routes.js
    - Requirements:
      * /models & passport
    - Log In Route
      * Utilizes the password.authentication middleware with local strategy
      * Redirects valid users to members page
    - Sign up Route
      * Automatically hashes user login info and stored securely based on the sequelize user model
    - User Logout Route
    - Client-side User Data
      * If the user is logged in their email and id will be pulled into api/user\_data
      * If not logged in an empty object will return
  + Html-routes.js
    - Requirements:
      * Path & isAuthenticated middleware
    - Set up module esports for all routes
      * App.get (“/”) directs to the default page
        + If user has an account they will redirect to their members page
        + If not, they will be sent to the signup page
      * App.get(“/login”)
        + If user is logged in they will be directed to the members page
        + If not, they will be sent to the login page
      * App.get(“/members”) uses the isAuthenticated middleware
        + If a user is not logged in and tried to access the members pages, they will be redirected to a signup page

* Package.json
  + Holds all information about the application including:
    - Scripts for the node server for testing
    - All dependencies needed to run the application
      * Look here to either manually run each of the dependencies in your terminal or use:
        + Npm install
        + All dependencies listed here will install automatically
* Server.js
  + Requirements:
    - Express, express-sessions, passport
  + Used to set up ports, require models configure middleware, keep track of user status during sessions, requiring routes and syncing to our database