PROG 3017 Assignment 2 - PHASE 2

**FULL STACK DEVELOPMENT**

# Prerequisites

### Required Labs: Lab 2 (NodeJS) and Lab 3 (Express JS)

**Required Assignments: Assignment 1, Assignment 2-Phase 1**

# Summary

In assignment 2A, you began to create the API that will interact with your MongoDB data documents. Because this API will have the ability to add, modify and delete your data in addition to retrieving data, it will also require the ability to Add (register) and Authenticate (login) users to be able to use the API

Before beginning the outlined requirements below, make sure that you have completed the following pre-requirements:

1. Be sure to have completed the requirements for Assignment 2 – Phase 1 which include the ability to connect to your MongoDB data, define models via Mongoose, and set environment variables via the *dotenv* npm package.

# Application Requirements (35 points)

1. **Add Basic Validation for your Custom Model (5 pts)**

Given the unique nature of each of your models for your own API, implement validation that you deem to be appropriate for your model. To keep things simple, we will be relying solely on Mongoose’s validation functionality and we will not be using Joi. Refer to the Mongoose validation documentation for more information (<https://mongoosejs.com/docs/validation.html>)  
   
*(Note: you can manage two separate layers of validation by implementing both Joi and Mongoose, but this can be cumbersome to manage. There is also a package called* ***joigoose*** *(*[*https://www.npmjs.com/package/joigoose*](https://www.npmjs.com/package/joigoose)*) that seeks to combine Joi and Mongoose together so that you only have to manage one Joi-based schema. Feel free to explore this option but you are not required to implement it for the assignment)*

1. **Define the User model for users of your API. (10 pts)**

Define a new user model that will store user information in MongoDB. The user model should define at the very least the following fields and should have validation implemented which corresponds as follows:

* firstName – required, maximum length 100 characters.
* lastName – required, maximum length 100 characters.
* email – required, must be unique, must be a valid email address.
* password – required, maximum length 255 characters.

Validation should be implemented using Mongoose validation. (<https://mongoosejs.com/docs/validation.html>)

Ensure that the model is exported as a module so that it can be used in any required route endpoints.

1. **Create an endpoint to Register new users (10 pts)**

Define and implement a new endpoint with the URL **/api/users/register.** When users post new user data to this endpoint a new user should be created with the needed information defined in the User model.

If the submitted registration data is deemed appropriate and validates, the saved password portion of the new user entry should be hashed using the *bcrypt* npm package. The endpoint should return the email and newly generated \_id of the created user.

All possible HTTP response scenarios should be accounted for with the correct corresponding HTTP response code provided.

1. **Create an endpoint to Login existing users (10 pts.)**

Define and implement a new endpoint with the URL **/api/users/login** which will receive an email and password as post data. The endpoint will validate and authenticate the submitted data against the data stored in the database.

Be sure to appropriately validate the email and password entries using Mongoosevalidation using the same rules enforced during registration.

If the supplied email and password authenticate successfully, the response should include a newly generated JSON Web Token (JWT - <http://jwt.io>) using the *jsonwebtoken* npm package.

Assign the JWT to a custom response header called *x-auth-token*. Ensure that your JWT secret is stored correctly as an environment variable in your .env file.

All possible response scenarios should be accounted for with the correct corresponding HTTP response code provided.

# Instructions

1. Don’t forget that a code review is a **necessary** part of this assignment. You will need to show your code to the instructor in class on the due date while going through an evaluation of the site’s functionality. You will need to explain how the code works and complete the code review part of the rubric. You will need to do this to at least a developing level (see the Note in the rubric below).
2. ***Late submissions will be subject to the late penalties laid out in the course outline.***