CSE 341 Web Services – Sterling Porter

If you use other technologies to store, deploy or demonstrate your code, you will not receive a grade. You must use GitHub, Render and YouTube.

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L01)

Required Technologies

In this course, you will always use the following technologies:

* GitHub to store your code
* Render to deploy your code
* YouTube to demonstrate the functionality of your code

If you use other technologies to store, deploy or demonstrate your code, you will not receive a grade. You must use GitHub, Render and YouTube.

When you submit your assignments, you will submit three links like this:

<https://github.com/byui-cse/cse341-code-student>

<https://cse341-code-student.onrender.com/>

<https://youtu.be/6SIACHzJe3g>

Recommended Technologies

* [VS Code](https://code.visualstudio.com/download)
  + [Rest Client](https://marketplace.visualstudio.com/items?itemName=humao.rest-client)
  + [Prettier](https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode)
  + [ESLint](https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint)

**GitHub**

[gar21085@byui.edu](mailto:gar21085@byui.edu)

Username: garretts8

Password: 1 + 1

Owner /Repository Name

garretts8/CSE341

<https://github.com/garretts8/CSE341.git>

Course Outcomes

Students that successfully complete this course will be able to do the following:

* Be ready for an entry-level position as a Node.js backend developer
* Develop secure APIs that perform CRUD operations on a NoSQL database
* Document APIs using modern API documentation tools
* Deploy and maintain an API in a production environment
* Effectively contribute to a team while learning independently
* Contribute to a professional portfolio by deploying multiple projects

Week 1

Getting started with Node.js

This course is an API development course. An API is defined as an application programming interface, or a connection between computers or between computer programs. In web development, APIs or Web Services allow us to request data from a server (in our case, a Node.js project), which will then get data from a database and send back to the frontend whatever data they requested. This is what APIs and Web Services do.

We will build robust, secure backends that will work with websites, mobile apps, wearable apps, and many other things. We will create data endpoints that will simply deliver data to any frontend when requested.

Seeing that this entire course will use Node.js, which has officially been coined as "JavaScript on steroids" by the author of this course, if you feel you may need a JavaScript refresher before diving in, the following video is a good one. From there, feel free to search Google or YouTube for individual concepts that still aren't clear.

* [(OPTIONAL) JavaScript Tutorial for Beginners: Learn JavaScript in 1 Hour](https://youtu.be/W6NZfCO5SIk)

There are many different tech stacks and languages that can handle API development in the way that we are going to do. Some popular ones include Node, C#, Java, Ruby, and Python. More important than learning any one of these is learning how to create web backends. Once you've learned one, the rest will come with relative ease.

In this course, we will use Node.js as our backend technology of choice. To prepare you for your assignment this week, here are some helpful resources to start learning Node:

* [Node and Express Tutorial](https://codeforgeek.com/express-nodejs-tutorial/)
* [Node docs](https://nodejs.org/en/docs/guides/getting-started-guide/)
* [(OPTIONAL) YouTube: Node.js Tutorial for Beginners: Learn Node in 1 Hour](https://www.youtube.com/watch?v=TlB_eWDSMt4&t=1s)
* [(OPTIONAL) YouTube search: nodejs for beginners](https://www.youtube.com/results?search_query=nodejs+for+beginners)

Even a simple web search like "[nodejs web server example](https://www.google.com/search?q=nodejs+web+server+example&oq=nodejs+web+server+example&aqs=chrome.0.69i59j0i22i30l3j0i390l3.1545j0j7&sourceid=chrome&ie=UTF-8" \t "_blank)" can be extremely helpful.

Saving to GitHub

Publishing to Render

At this point in the lesson, you may likely be completely lost, confused, stressed, frustrated, or worried. Please remember, that this course is designed to help teach you how to find answers and information independently. With that said, you have solutions for these first four lessons, and I've added one more video for lesson one to help you learn a bit about node. Enjoy. :)

The following video may be helpful but is not required.

What is Node.js

Node.js is an open-source, cross-platform, backend JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

In short, it is JavaScript that can do a lot of incredible things. It is very commonly used for backend web development. We have the frontend which consists of HTML, CSS and a bit of JavaScript. We also have some sort of database where data is stored long-term. And we have some backend system that will connect the frontend to the database. This is what we will use Node for in this class.

1. Open the official page for [Node.js downloads](https://nodejs.org/en/download/) and download Node.js for your operating system.
2. Run the downloaded Node.js Installer, including accepting the license, selecting the destination, and authenticating for the install. *This requires Administrator privileges, and you may need to authenticate.*
3. To ensure Node.js has been installed, run node -v in your terminal—you should get something like vx.x.x
4. Update your version of npm with npm install npm --global. *This requires Administrator privileges, and you may need to authenticate*
5. Congratulations! You now have Node.js installed and are ready to start building!

YouTube

Almost all of your code assignment submissions in this class will include a YouTube video link. You will use this video as an opportunity to demonstrate the tasks mentioned within the rubric of the individual assignments. As part of the course setup for this class, please ensure that you have a YouTube account created. If you need help doing this, [this Google search for how to create a YouTube account may help](https://www.google.com/search?q=how+to+create+a+YouTube+account).

If you have never recorded a video of you sharing your screen while recording your voice, that is okay. There are many free resources that allow you to create such recordings. Here are a few popular ones:

* [Zoom](https://zoom.us/)
* [Loom](https://www.loom.com/)
* [Screencastify](https://www.screencastify.com/products/screen-recorder)
* [screencast-o-matic](https://screencast-o-matic.com/screen-recorder)
* [Best Screen Recorder for Mac](https://www.google.com/search?q=best+screen+recorder+for+mac)
* [Best Screen Recorder for Windows](https://www.google.com/search?q=best+screen+recorder+for+Windows)
* [Best Screen Recorder for Chromebook](https://www.google.com/search?q=best+screen+recorder+for+Chromebook)

Render

* What is API Deployment?

All information and videos about setting up Render for this class can be found on the deployment page.

A deployed API is the code is hosted and built on a platform away from your computer so that anyone and anything can access it via a public URL.

Deploying an API to be accessible anywhere is very similar to deploying any website. There are fewer options for deploying a backend compared to comparing a frontend, but there are many tools available to do this. Below is a list of several popular ones, but there are many more and they fluctuate frequently.

A2 Hosting

Heroku

Amazon Web Services

DigitalOcean

Glitch

Google Cloud Platform

Microsoft Azure

Platform.sh

NodeChef

Render

Fly

Railway

Vercel

* Render

We will be using Render because it...

* ...has a great free tier for API deployments with Node.js
* ...doesn't require a credit card up front
* ...supports Node.js and has a strong community
* ...supports automatic deploys from git providers like GitHub
* ...easily deploys Node.js APIs

According to [Render's website](https://render.com/), Render is a unified cloud to build and run all your apps and websites with free TLS certificates, a global CDN, DDoS protection, private networks, and auto deploys from Git.

For this class, we will use it to deploy our web projects to the internet so that they can be accessed by anyone.

* Create a Render Account

1. If you don't have a GitHub account yet, please [create one now](https://github.com/signup). All information/videos about setting up GitHub for this class can be found on the GitHub page.
2. Now that you have a GitHub account, create a Render account by going to the [Render sign up page](https://dashboard.render.com/register).
3. Click on the GitHub button as your sign up method.
4. Go through the steps to authorize GitHub and finish creating your Render account.

Here is a video going through these steps

* How to deploy Render

To deploy code to Render, you need code to deploy. :) Before attempting to deploy an API, you should first test your API thoroughly on your computer running on localhost. If you can successfully run it without errors, and test the API endpoints without errors, then you are ready to try a deploy to Render.

Render's Node.js Documentation for deployments is very simple and very good. It is recommended that you start here, but you'll use your own GitHub repository, and not their forked hello-world example.

Here are the steps for your first deploy:

Create new Web Service project

Connect to GitHub repo (following instructions in documentation link above)

If you use environment variables in your app, add them to the Environment section within Render

Wait for the build to finish, give it 10-15 minutes, then test your API with a REST client

Here is a video showing this

Dashboard.render.com

* Testing your deployed API

Testing an API is incredibly simple if you have the right tool. The tool to test a RESTful API is called a REST client, and there are many free ones to choose from. One simple, straight-forward one that can be used is a VS code extension called [REST Client](https://marketplace.visualstudio.com/items?itemName=humao.rest-client).

* Install the extension
* Read the extension documentation (you'll see this documentation on the same page that you install the extension from either in VS Code or in the browser).
* Create a **.rest** file
* Add a line to the file like this: **GET https://cse341-test.onrender.com/contacts**
* Click the **send request** link above this line.
* Verify that the API request was successful

As mentioned previously, if you've already tested your API locally, it will be the same process...just with your new url from Render.

* Building the API with Node.js

Getting started with our first APIs

Watch: <https://www.youtube.com/watch?v=K00J87SofEc>

Getting started with Node.js

This course is an API development course. An API is defined as an application programming interface, or a connection between computers or between computer programs. In web development, APIs or Web Services allow us to request data from a server (in our case, a Node.js project), which will then get data from a database and send back to the frontend whatever data they requested. This is what APIs and Web Services do.

We will build robust, secure backends that will work with websites, mobile apps, wearable apps, and many other things. We will create data endpoints that will simply deliver data to any frontend when requested.

Seeing that this entire course will use Node.js, which has officially been coined as "JavaScript on steroids" by the author of this course, if you feel you may need a JavaScript refresher before diving in, the following video is a good one. From there, feel free to search Google or YouTube for individual concepts that still aren't clear.

* [(OPTIONAL) JavaScript Tutorial for Beginners: Learn JavaScript in 1 Hour](https://youtu.be/W6NZfCO5SIk)

There are many different tech stacks and languages that can handle API development in the way that we are going to do. Some popular ones include Node, C#, Java, Ruby, and Python. More important than learning any one of these is learning how to create web backends. Once you've learned one, the rest will come with relative ease.

In this course, we will use Node.js as our backend technology of choice. To prepare you for your assignment this week, here are some helpful resources to start learning Node:

* [Node and Express Tutorial](https://codeforgeek.com/express-nodejs-tutorial/)
* [Node docs](https://nodejs.org/en/docs/guides/getting-started-guide/)
* [(OPTIONAL) YouTube: Node.js Tutorial for Beginners: Learn Node in 1 Hour](https://www.youtube.com/watch?v=TlB_eWDSMt4&t=1s)
* [(OPTIONAL) YouTube search: nodejs for beginners](https://www.youtube.com/results?search_query=nodejs+for+beginners)

Even a simple web search like "[nodejs web server example](https://www.google.com/search?q=nodejs+web+server+example&oq=nodejs+web+server+example&aqs=chrome.0.69i59j0i22i30l3j0i390l3.1545j0j7&sourceid=chrome&ie=UTF-8" \t "_blank)" can be extremely helpful.

All information and videos about saving to GitHub for this class can be found on the GitHub page.

1. Go to [GitHub.com](http://github.com/) and sign up for a new account if you haven't already. If you already have an account, that is perfect, you can use it here.
2. After creating an account, return to GitHub.com and sign in. In the upper right corner of the screen you should see a "+" icon. Click this and select "New Repository".
3. Choose a name for your repository (likely cse341-node or something similar).
4. Leave the repo public and do not check the box for a readme. Leave .gitignore and license as "none".

Instructor Tip:

It's not technically a problem if you add these other files now, but it makes it a little more difficult when you go to push to GitHub for the first time, because you'll have conflicts that you'll have to resolve. So at this point, it's easiest to leave everything blank.

1. Click the button to create the repository.
2. On the page that follows, you should see a URL for your new repository (e.g., "https://github.com/username/repositoryname.git"). Copy this URL for use later on.

Connect Local Code to GitHub Repository

Watch on YouTube: Deploy Node JS API with Env Variables on Render Automatically with GitHub Deploys

* What is MongoDB?

MongoDB is a source-available, cross-platform, document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

For all of our projects this semester, our database for our data will be MongoDB.

Create MongoDB Account and Cluster

1. Navigate to the [MongoDB Atlas](https://www.mongodb.com/cloud/atlas) Site
2. Select **Start Free**
3. Fill out your email, name, and password to create an account.
4. Choose the free option and select **Create a Cluster**
5. Ensure that the configurations for the new cluster look similar to the following image and have the free options selected.

Lesson 1: Introducing our tech stack

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L01)

Course Setup

This week's reading will include more direction than most in an effort to get everyone set up for the course. For any of the following that you have not worked with before or need to set up for this class, please go through the installation and setup instructions for each.

Visual Studio Code

Microsoft Teams

Node.js

YouTube

Git & GitHub

MongoDB

Render

Getting started with our first APIs

The following content may be helpful but is not required. Please see the syllabus for more details.

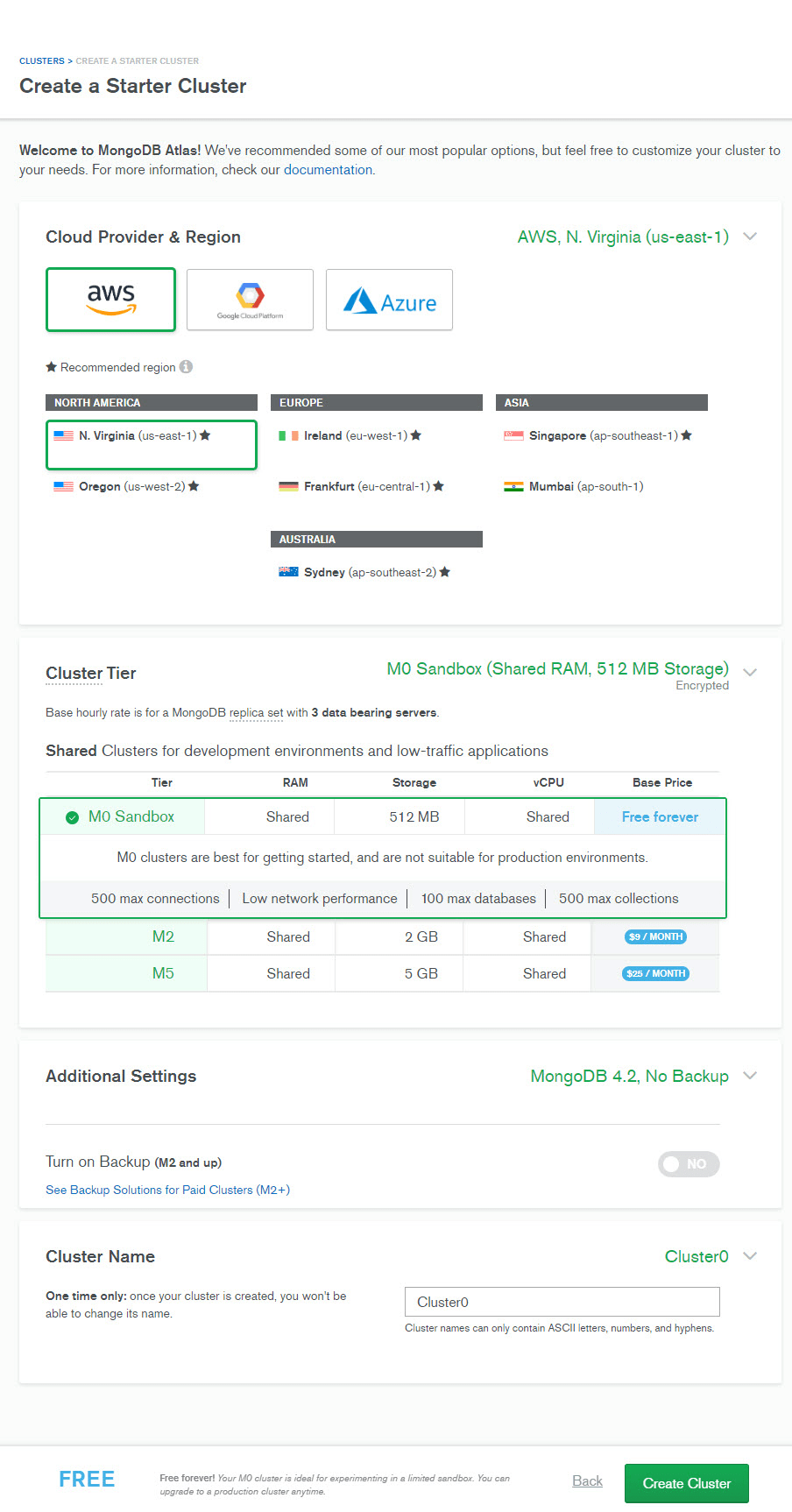
Building the API with Node.js

Saving to GitHub

Publishing to Render

At this point in the lesson, you may likely be completely lost, confused, stressed, frustrated, or worried. Please remember, that this course is designed to help teach you how to find answers and information independently. With that said, you have solutions for these first four lessons, and I've added one more video for lesson one to help you learn a bit about node. Enjoy. :)

The following video may be helpful but is not required.



1. Your account is now ready to go. You will not need to create another database account or cluster. During the semester, you will be able to add collections of data and manage this database, so make bookmarks where needed.

If you are having issues with this, please watch the following video from January, 2023:

The following video may be helpful but is not required.

CSE 341 Mongo setup

Lesson 1: Introducing our tech stack

   Personal Assignment

Overview

**Purpose**: Publish API to the web.  
**Task**: Complete the assignment.

Learning Objectives

By the end of this assignment the student will be able to do the following:

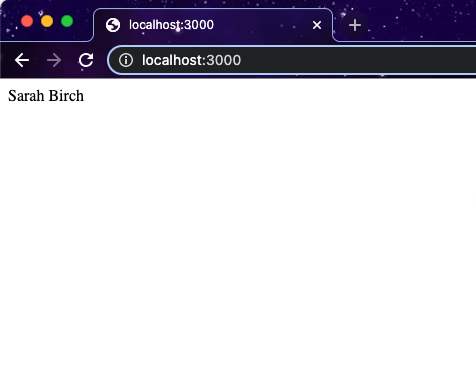
* Create a Render Link that is accessible, error free, and data is viewable on home page
* Create a node.js project that shows code for route that displays a name upon request of the home route
* Create a GitHub repository that has a node.js project containing folders called "routes" and "controllers"

Assignment Description

If you have not already, create a GitHub repository for your work. In this class, you will never submit code of any kind; you will always submit links to your work. You will always use the following tools:

* GitHub to store your code
* Render to deploy your code
* YouTube to demonstrate the functionality of your code

If you use other tools to store, deploy or demo your code, you will not receive a grade.

* Create a GitHub repository
* Use npm init to start a new node project from scratch
* Create a web server
* Create a route in your server that will return your the name of someone you know. If requested through the browser, you should see this data on the screen as shown in the image below:  
  
* Push to GitHub.
* Publish to Render.
* Create a brief video demonstrating the functionality of your assignment. Upload it to YouTube (*public* or *unlisted*).
* Submit GitHub, Render, and YouTube links in I-Learn.
* Be sure to review the rubric below to see how you will be graded on this assignment.

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| Deployment  *(Graded via Render)* | 40% | Meets Proficient criteria and application shows data (name of someone you know) on the home page | Meets Developing criteria and there are no Render log errors | Meets Beginning criteria and Render link can be viewed in the browser | Render link provided | Render link not submitted |
| Functionality  *(Graded via YouTube)* | 30% | Meets Proficient criteria plus application displays name when home route is requested | Meets Developing criteria plus no errors are present (and no "Cannot GET /" message) when accessing Render home url | Meets Beginning criteria plus Code Repository shows code for '/' route (Render home url) | YouTube link submitted and Node.js project is presented | YouTube link not submitted |
| Architecture  *(Graded via GitHub)* | 20% | Meets Proficient criteria plus application has a folder called "controllers" that has functions for each route | Meets Developing criteria plus application has a file in the "routes" folder called "index.js" | Meets Beginning criteria plus application has a folder called "routes" that includes all routes for assignment | Node project created and present in GitHub repository | GitHub link not submitted |
| Presentation  *(Graded via YouTube)* | 10% | Meets Proficient criteria and video is less than 1 minute | Meets Developing criteria and video shows GitHub code | Meets Beginning criteria and video shows the home url data (name of someone you know) | YouTube link submitted | YouTube link not submitted |

Show Solution

Week 2

Lesson 2: REST Clients, GET Requests, Node Architecture

   Overview

Introduction

Welcome to lesson two! This lesson, you will learn how to better organize your Node.js code to create web services more efficiently. You will also learn how to use REST clients to effectively test your services when there isn't a frontend available to consume them. Lastly, you will create a number of HTTP GET requests that will push you to learn more about how HTTP requests work in general.

Topics for this lesson:

* Connect API to MongoDB Securely
* Rest Clients
* GET Requests
* Query Parameters
* Debugging Node.js Applications
* Headers
* Node Architecture and API Organization

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%202%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L02)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

Connect API to MongoDB Securely

REST Clients

GET Requests

Query Parameters

Debugging Node

Headers

Node Architecture and API Organization

   Team Assignment

Overview

**Purpose**: Develop an API for an existing frontend.  
**Task**: Complete the assignment.

To complete this assignment, you will create a backend for a frontend that is already complete. After building your API endpoints, you will be able to use the frontend to both test it and to see your data in action.

* Start by downloading the [frontend code](https://github.com/byui-cse/cse341-code-student/tree/L02-team-frontendStart).
* You will not change any code in it, you will just open the html file in the browser to test that your backend is functional. You'll see a series of ids in the HTML and the JavaScript. Look at the data that the JavaScript file uses to populate the HTML, this includes a lot of text, a few links, and a base64 image.
* Create a new node project (npm init) with a server and a REST endpoint to GET all data required by the frontend code you downloaded.
* While you're writing your code, test it with a REST Client to ensure it is working correctly.
* Once everything looks good in the REST Client, open the frontend webpage in the browser and check to see if it is working (ensure that your backend server is still running on port 8080).
* STRETCH CHALLENGE: Connect the API to MongoDB, and retrieve data from database.
* After completing the assignment or spending an hour on it, please review the solution and solution video below. Please note that there are many different ways to accomplish what is being asked and this is simply one way to do things. The following code is also a very good starting point for projects.

Show Solution

   Personal Assignment

Overview

**Purpose**: Connect API to a database.  
**Task**: Complete the assignment.

Learning Objectives

By the end of this assignment the student will be able to do the following:

* Create a database that stores sensitive information securely in an .env file and successfully reads data from MongoDB
* Create a database that successfully retrieves the data from MongoDB
* Create a GET request that returns all the data in a MongoDB collection
* Incorporate security measures to store sensitive database information in a secure way
* Connect all routes in a Node.js project with a single line of code
* Separate route rules and api logic by using controller functions

Assignment Description

For this lesson's personal assignment, you will add to your assignment from the previous lesson where you created a web server and a single GET request. In this assignment, you will create a collection in MongoDB, and then create a GET request that will return data from it. You will also practice using tools that you have been learning over the last few days. Below are the steps to complete this task:

* Run npm start in the terminal of your node project from last lesson to ensure it is working.
* Create a collection in MongoDB called contacts. Insert at least three documents either for people in this class or people that you know. Each document should have the following fields: firstName, lastName, email, favoriteColor, birthday.
* Create an .env file to store your MongoDB connection string.
* Connect your node project to MongoDB.
* Create a new route file in your node project called contacts.
* For the following steps, use the debugger in VS Code to help understand what is going on with the data. And use a REST client of your choosing to make http requests to your web server. [This REST Client works well](https://marketplace.visualstudio.com/items?itemName=humao.rest-client).
* Create a GET request in your contacts route file that will return all of the documents in your contacts collection.
* Create another GET request in your contacts route file that will return a single document from your contacts collection where an id matches the id from a query parameter.
* Once these api routes are working, push to GitHub and test on Render to ensure everything works in that environment as well. (NOTE: The connection string stored in your .env file will not ever be pushed to GitHub, which means it won't be published to Render. You will need to add config vars to Render. You can learn how to do this on thedeploy page.
* Ensure you include a .rest file for testing (similar to what you see in the example video).
* Push to GitHub.
* Publish to Render.
* Create a brief video demonstrating the functionality of your assignment. Upload it to YouTube (*public* or *unlisted*).
* Submit GitHub, Render, and YouTube links in I-Learn.
* Be sure to review the rubric below to see how you will be graded on this assignment.

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| API Endpoints  *(Graded via Render)* | 30% | Meets Proficient criteria plus application has a GET request that retrieves a single contact from their ID | Meets Developing criteria plus application has a GET request that returns all the contacts in your collection | Meets Beginning criteria plus application Render url is accessible without errors | Node project runs without errors | Render link not submitted |
| MongoDB Connection  *(Graded via YouTube)* | 30% | Meets Proficient criteria and sensitive database information is stored securely in .env file and Render Config Vars (not pushed to GitHub) | Meets Developing criteria and application successfully reads data from MongoDB | Meets Beginning criteria and application successfully connects to MongoDB | MongoDB attempt shown in video | YouTube link not submitted or no evidence of MongoDB |
| Security  *(Graded via YouTube)* | 20% | Meets Proficient criteria plus video shows that Config Vars are used in Render | Meets Developing criteria plus .env file is local project shown in video (no need to open the file) | Meets Beginning criteria plus no MongoDB credentials are stored in GitHub (.env file is NOT in GitHub) | Application connects to MongoDB | YouTube link or GitHub link not submitted or no MongoDB implementation |
| Architecture  *(Graded via GitHub)* | 20% | Meets Proficient criteria plus includes a .rest file with routes for localhost and Render | Meets Developing criteria plus each route calls a function imported from a controller | Meets Beginning criteria plus server connects to routes in project "routes" folder using only a single line of code | Server file is present in root folder (server.js, index.js or app.js) to run the Node project | Render link not submitted |

Show Solution

Week 3

Lesson 3: HTTP Requests & Dev Tools

   Overview

Introduction

Welcome to lesson three! Having mastered GET requests, you will now learn how to add POST, PUT, and DELETE requests to your API. You will also learn several tools that will make your development more efficient and productive.

Topics for this lesson:

* URI Hierarchies
* HTTP POST
* HTTP PUT
* HTTP DELETE
* MongoDB CRUD
* Linters
* Code Formatters

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%203%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L03)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

URI Hierarchies

POST

PUT

DELETE

MongoDB CRUD

Linters

Code Formatters

Project Configuration and Management

   Team Assignment

Overview

**Purpose**: Research, learn, and share industry standards, best practices, and helpful resources.  
**Task**: Complete the assignment.

The team activity for this lesson will consist of participating in a developer forum. Each person in this class will learn from different resources, try different things, and end up with a unique knowledge base. This forum is an opportunity for you to share industry standards and best practices regarding the technologies that we are using and that you're learning about in this lesson. To receive full credit for this assignment, you must do the following:

* Use the channel in Teams called Developer Forum for this assignment.
* Pick a topic from this lesson and write a post that teaches at least one best practice or industry standard for that topic.
* Include the lesson number at the top of your post along with your title (for example: L03 - POST requests in Node.js)
* The post should consist of at least 200 words.
* Include at least two sources to site your claims.
* The post should be informative, accurate, relevant, and not just a copy and paste of another post.
* Respond to at least two peers before the end of the lesson.
* Finally, use the I-Learn quiz to report your participation in the forum.

   Personal Assignment

Overview

**Purpose**: Perform GET, POST, PUT, and DELETE requests in an API connected to a database.  
**Task**: Complete the assignment.

Learning Objectives

By the end of this assignment the student will be able to do the following:

* Create a MongoDB collection called contacts
* Create a POST route for creating new contacts that returns the ID of the new contact and a 201 status
* Create a PUT route for updating a contact that returns a 204 status
* Create a DELETE route for deleting a contact that returns a 200 status

Assignment Description

For this assignment, you will continue to work on your project from the previous two lessons. You have already create a MongoDB cluster with a contacts collection along with a node.js project with several GET requests to retrieve different pieces of data from your database. This lesson, you will add a few more routes to your Node.js project to add POST, PUT, and DELETE requests to your API.

* Create a POST route to create a new contact. All fields are required. Return the new contact id in the response body.
* Create a PUT route to update a contact. This route should allow for a url similar to this: api-url-path/contacts/id-to-modify. (The id won't be modified, it will just be the means of finding a specific document in the database.) Return an http status code representing the successful completion of the request.
* Create a DELETE route to delete a contact. Return an http status code representing the successful completion of the request.
* You should test each of these routes thoroughly using your rest client of choice ([this REST Client works well](https://marketplace.visualstudio.com/items?itemName=humao.rest-client)) .
* Ensure you include a .rest file for testing (similar to what you see in the example video - shown in sample solution below).
* Push to GitHub.
* Publish to Render.
* Create a brief video demonstrating the functionality of your assignment. Upload it to YouTube (*public* or *unlisted*).
* Submit GitHub, Render, and YouTube links in I-Learn.
* Be sure to review the rubric below to see how you will be graded on this assignment.

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| Render Connects to MongoDB  *(Graded via YouTube)* | 30% | Meets Proficient criteria and video shows sensitive database information is stored securely in Render Config Vars (not pushed to GitHub) | Meets Developing criteria and Render application successfully reads data from MongoDB | Meets Beginning criteria and Render application successfully connects to MongoDB | Render url is submitted & MongoDB attempt is shown in video | Render link or YouTube link not submitted or no evidence of MongoDB |
| New Contact  *(Graded via YouTube)* | 15% | Meets Proficient criteria plus POST returns the id of newly created contact and status of 201 | Meets Developing criteria plus video shows that the POST route creates a new contact in MongoDB | Meets Beginning criteria plus code is present to accept a request body | Node.js project shows code for the POST route to create a contact | Render link, GitHub link, or YouTube link not submitted. Or no MongoDB reference. |
| Update Contact  *(Graded via YouTube)* | 15% | Meets Proficient criteria plus PUT returns response with status of 204 | Meets Developing criteria plus video shows that the PUT route updates a contact in MongoDB | Meets Beginning criteria plus code is present for PUT route to update a contact by their id | Node.js project shows code for the PUT route to update a contact | Render link, GitHub link, or YouTube link not submitted. Or no MongoDB reference. |
| Delete Contact  *(Graded via YouTube)* | 15% | Meets Proficient criteria plus DELETE returns response with status of 200 | Meets Developing criteria plus no errors are present and video shows that the DELETE route deletes a contact in MongoDB | Meets Beginning criteria plus code is present for DELETE route to delete a contact by their id | Node.js project shows code for the DELETE route to delete a contact | Render link, GitHub link, or YouTube link not submitted. Or no MongoDB reference. |
| Architecture  *(Graded via GitHub)* | 15% | Meets Proficient criteria plus includes a .rest file with routes for localhost and Render | Meets Developing criteria plus each route calls a function imported from a controller | Meets Beginning criteria plus server connects to routes in project "routes" folder using only a single line of code | Server file is present in root folder (server.js, index.js or app.js) to run the Node project | Render link not submitted |
| Endpoint Testing  *(Graded via GitHub)* | 10% | Meets Proficient criteria and .rest file has GET, POST, PUT and DELETE routes for Render and Localhost | Meets Developing criteria and .rest file has at least three Render routes | Meets Beginning criteria and .rest file has at least three localhost routes | Node project has .rest file | No submission, no .rest file, or no attempt made |

Week 4

Lesson 4: API Documentation

   Overview

Introduction

Welcome to lesson four! The focus of this lesson is API Documentation. We will learn about modern tools to help us design, document, test, and manage our APIs. You will also submit your first project of the semester.

Topics for this lesson:

* API Documentation
* Popular API Tools

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%204%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L04)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

API Documentation

Swagger

Other Options

   Team Assignment

Overview

**Purpose**: Add documentation to an API  
**Task**: Complete the assignment

For this team activity, your group will be given a Node.js project similar to the one you've been working on and add API documentation to it.

* Download [this project](https://github.com/byui-cse/cse341-code-student/tree/L04-team-start) as your starting point.
* Examine the code. For this assignment you will only worry about the temple routes.
* Create a new collection in MongoDB called temples.
* Import the temples.json file into your new temples collection.
* Adjust the .env file so that you can successfully run npm start and connect to the database.
* Create a new Swagger file and add documentation for each route you see there. There are many ways to do this, but [this swagger-autogen package](https://www.npmjs.com/package/swagger-autogen) may be useful.
* Generate a visual UI and living documentation of your swagger.json. There are many ways to do this, but [this swagger-ui-express package](https://www.npmjs.com/package/swagger-ui-express) may be useful.
* Now that you have documented the routes that are existing, talk with your team about what routes are missing. Add these routes to the API documentation.
* Regenerate your swagger.json, and restart your server. Test these routes in the Swagger UI by selecting the "Try it Out" then "Execute" buttons.
* Once completed, review the sample solution below and compare/contrast your findings.
* Use the I-Learn quiz to report your participation in this activity.

Show Solution

   Personal Assignment

Overview

**Purpose**: Create and publish API documentation.  
**Task**: Complete the assignment.

Learning Objectives

By the end of this assignment the student will be able to do the following:

* Create a Node Project that handles http requests
* Create a collection called "contacts" in MongoDB
* Create a Render project that shows no errors
* Construct API documentation using Swagger that includes the following routes: GET all, GET by id, PUT, Post, and DELETE

Assignment Description

In previous lessons you have written RESTful API routes in Node.js that connect the outside world to a MongoDB database. For this lesson, you will finish your Contacts project. To conclude, you will document your API using Swagger. Below are the steps to complete this task:

* Review the [project requirements and grading rubric](https://cse341.netlify.app/projects/1).
* Create Swagger documentation for this API project. One of the easiest ways to do this is to use the packages mentioned and used in the team activity.
* With your published API (not localhost), test your swagger documentation routes to ensure they work.
* Once completed, push any changes to GitHub and verify changes in Render.
* Your published project must include an "/api-docs" route that has the interactive Swagger GUI.
* Record a brief video demonstration that shows you using the Swagger documentation successfully sending requests to each route. Be sure to use your Render url for this. Also include evidence that your database is being updated. If Render isn't working, you may use localhost, but you still need to submit your Render link in this project submission. Look at the rubric to see how many points will be deducted if you use localhost in this video.
* Push to GitHub.
* Publish to Render.
* Create a brief video demonstrating the functionality of your assignment. Upload it to YouTube (*public* or *unlisted*).
* Submit GitHub, Render, and YouTube links in I-Learn.
* Be sure to review the rubric below to see how you will be graded on this assignment.
* EXTRA CREDIT OPPORTUNITY: You can get up to 10% added to your project grade if you display in your video that you successfully used the fronted project referenced below to use your backend API (details in the syllabus).

Full Stack Project Overview

In an effort to help you see how a backend project works within a fullstack project, [this React Application](https://cse341-contacts-frontend.netlify.app/) was created to work with your Contacts project. It performs all of the api requests/routes that are in your application but with a GUI instead of swagger, or another rest client. As a stretch challenge, try to test your published api with this website. The video below goes into more detail about this, and about 1 extra thing that you'll need to add to your backend api in order for this to work.

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| HTTP Requests  *(Graded via YouTube)* | 25% | Meets Proficient criteria and performs PUT request. MongoDB is updated | Meets Developing criteria and performs POST and DELETE requests. MongoDB is updated | Meets Beginning criteria and performs GET requests from MongoDB | Node project has evidence of http requests | GitHub link, or YouTube link not submitted. Or no evidence of HTTP requests |
| API Documentation  *(Graded via YouTube)* | 25% | Meets Proficient criteria and the swagger.json contains a contact schema that is used in all of the routes | Meets Developing criteria and the route: "/api-docs" is accessible and is able to test the endpoints | Meets Beginning criteria and the following routes are present in the swagger.json: GET all, GET by id, PUT, POST, and DELETE | Node project contains a swagger.json file with some content in it relevant to the "contacts" project | No API documentation present (no swagger.json or "api-docs" route in project for swagger) |
| Deployment  *(Graded via YouTube)* | 25% | Meets Proficient criteria and uses Render Config Vars (no database credentials stored in GitHub - must show GitHub repo in video) | Meets Developing criteria and video shows project functionality using Render (not localhost) | Meets Beginning criteria, project is deployed to Render and Render shows no errors | Render project has been created | Render link, GitHub link, or YouTube link not submitted |
| Database  *(Graded via YouTube)* | 15% | Meets Proficient criteria and the contacts in the database have the following fields: firstName, lastName, email, favoriteColor, and birthday | Meets Developing criteria and the contacts collection has at least five contacts | Meets Beginning criteria and MongoDB has a collection called "contacts" | MongoDB account exists and was demonstrated in the video | No evidence of database present or no YouTube link |
| Architecture  *(Graded via GitHub)* | 10% | Meets Proficient criteria plus includes a .rest file with routes for localhost and Render | Meets Developing criteria plus each route calls a function imported from a controller | Meets Beginning criteria plus server connects to routes in project "routes" folder using only a single line of code | Server file is present in root folder (server.js, index.js or app.js) to run the Node project | Render link not submitted |
| Extra Credit  Frontend Integration  *(Graded via YouTube)* | 10% | Meets Developing criteria and the backend PUT works on the frontend and the data change shows in MongoDB | Meets Developing criteria and the backend POST works on the frontend and the data change shows in MongoDB | Meets Beginning criteria and the backend DELETE works on the frontend and the data change shows in MongoDB | The backend GET works on the frontend | No attempt made or nothing works |

Show Solution

Week 5

Lesson 5: REST & JSON Alternatives

   Overview

Introduction

Welcome to lesson five! This lesson teaches several alternatives to REST and JSON. It is intended to provide you with an understanding of other technologies out there, but we will continue to focus on RESTful APIs that return JSON in this course. You finished your Contacts project in the previous lesson, and in this lesson you'll start a new project of your choosing.

Topics for this lesson:

* Services and Standards
* Types of Data

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%205%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L05)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

JSON vs XML

REST Overview

RPC

XML+RPC

SOAP

GraphQL

   Team Assignment

Overview

**Purpose**: Research, learn, and share industry standards, best practices, and helpful resources.  
**Task**: Complete the assignment.

The team activity for this lesson will consist of participating in a developer forum. Each person in this class will learn from different resources, try different things, and end up with a unique knowledge base. This forum is an opportunity for you to share industry standards and best practices regarding the technologies that we are using and that you're learning about in this lesson. To receive full credit for this assignment, you must do the following:

* Use the channel in Teams called Developer Forum for this assignment.
* Pick a topic from this lesson and write a post that teaches at least one best practice or industry standard for that topic.
* Include the lesson number at the top of your post along with your title (for example: L03 - POST requests in Node.js)
* The post should consist of at least 200 words.
* Include at least two sources to site your claims.
* The post should be informative, accurate, relevant, and not just a copy and paste of another post.
* Respond to at least two peers before the end of the lesson.
* Finally, use the I-Learn quiz to report your participation in the forum.

   Personal Assignment

Overview

**Purpose**: Start developing a new API.  
**Task**: Complete the assignment.

Learning Objectives

By the end of this assignment the student will be able to do the following:

* Initiate a new Node.js backend API
* Create a new database in MongoDB and connect to it from the Node.js application
* Create GET and POST requests
* Create API documentation to test API endpoints

Assignment Description

For your assignment this lesson, you will create a new project of your choosing. You will have approximately four lessons in this course to complete it, similar to the last project. This project will perform CRUD operations on a MongoDB database, be published to the web, incorporate security measures, and include API documentation.

* Decide on the project you're going to create.
* As you decide on a project, review the [overall project requirements](https://cse341.netlify.app/projects/2) to ensure it is a good fit.
* Create a new Node.js project
* You'll use the same MongoDB cluster/account, but you should create a new database for this project. (When you connect to MongoDB Compass, you connect to your account, and your list of databases shows up in the bottom left corner.)
* Create your first REST API routes (similar to what we did with the routes and controllers folders in our first project). You will need at least one GET and one POST that are functional for this assignment.
* The Api-docs need to show routes for GET, GET by ID, POST, PUT, and DELETE. This is can be done by adding code to the routes folder and having swagger-autogen generate the swagger.json for you. Or you can modify the swagger.json directly. This is for a design-first mindset for your project.
* Create a new Render project
* Be sure to create an env file for your local MongoDB credentials, and add config vars to your Render project. Your MongoDB credentials should never get pushed to git, so be sure to include your env file in your gitignore.
* Push to GitHub.
* Publish to Render.
* Create a brief video demonstrating the functionality of your assignment. Upload it to YouTube (*public* or *unlisted*).
* Submit GitHub, Render, and YouTube links in I-Learn.
* Be sure to review the rubric below to see how you will be graded on this assignment.

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| API Endpoints  *(Graded via YouTube)* | 30% | Meets Proficient criteria and the POST request returns the new id in the response body and a status code of 201 | Meets Developing criteria and performs POST. Video shows MongoDB being updated | Meets Beginning criteria and performs GET requests from MongoDB | Node project has evidence of HTTP requests | GitHub link or YouTube link not submitted, or no evidence of HTTP requests |
| Deployment  *(Graded via YouTube)* | 30% | Meets Proficient criteria and Render CONFIG VARS are being used | Meets Developing criteria and routes for GET and POST don't contain errors | Meets Beginning criteria and routes for GET and POST are present | Render (not localhost) is being used in the video and Render url loads without errors | Render link or YouTube link not submitted, or Render errors are present |
| API Documentation  *(Graded via YouTube)* | 30% | Meets Proficient criteria and the "/api-docs" page (or Apollo server link) can test the endpoints | Meets Developing criteria and the route: "/api-docs" (or Apollo server link) is accessible | Meets Beginning criteria and the following routes are present in the documentation: GET all, GET by id, PUT, POST, and DELETE | Exidence of API documentation is present in project (either a swagger.json file or Apollo server) | YouTube link not submitted or no API documentation present (no swagger.json for swagger or Apollo server link for GraphQL) |
| Architecture  *(Graded via GitHub)* | 10% | Meets Proficient criteria plus includes a .rest file with routes for localhost and Render | Meets Developing criteria plus each route calls a function imported from a controller | Meets Beginning criteria plus server connects to routes in project "routes" folder using only a single line of code | Server file is present in root folder (server.js, index.js or app.js) to run the Node project | Render link not submitted |

Show Solution

Week 6

Lesson 6: Validation & Error Handling

   Overview

Introduction

Welcome to lesson six! During this lesson you will continue work on the project you started in the previous lesson. You will learn how to add validation to your routes, and how to handle errors in your routes.

Topics for this lesson:

* Validation
* Error Handling

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%206%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L06)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

Validation

Error Handling

   Team Assignment

Overview

**Purpose**: Gain exposure to Mongoose, which is widely used in industry in association with MongoDB.  
**Task**: Complete the assignment.

Your team assigment this lesson will give you experience working with a complete RESTful API that uses MongoDB and Mongoose.

* You may use the code from your completed lesson 04 project, or from [this repository.](https://github.com/byui-cse/cse341-code-student/tree/L06-team-start)
* Have a member of your team set up the MongoDB connection.
* Discuss as a group what validation and what types of error handling could be added to the API.
* Add error handling and validation to each route (either divide and conquer, or do pair programming).
* Use a REST client to test your routes as you work.
* Once completed, review the sample solution below and compare or contrast your findings.
* Use the I-Learn quiz to report your participation in this activity.

Show Solution

   Personal Assignment

Overview

**Purpose**: Gain experience performing CRUD operations  
**Task**: Complete the assignment

Learning Objectives

By the end of this assignment the student will be able to do the following:

* Create PUT and DELETE API endpoints
* Incorporate data validation into Node.js application
* Incorporate error handling into Node.js application
* Document API routes throughout project development

Assignment Description

For this assignment, you will build on the project you started in the previous lesson. Your tasks are as follows:

* Add PUT and DELETE routes to your API. Verify in MongoDB that these work as intended.
* Add validation to your routes
* Add error handling to your routes
* Be sure to update your API documentation to include these news routes
* Push to GitHub.
* Publish to Render.
* Create a brief video demonstrating the functionality of your assignment. Upload it to YouTube (*public* or *unlisted*).
* Submit GitHub, Render, and YouTube links in I-Learn.
* Be sure to review the rubric below to see how you will be graded on this assignment.

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| API Endpoints  *(Graded via YouTube)* | 30% | Meets Proficient criteria. The PUT returns 204 status and the DELETE returns a 200 status | Meets Developing criteria and performs at least one PUT request. MongoDB is modified | Meets Beginning criteria and performs at least one DELETE request. MongoDB is modified | Video shows PUT and DELETE API endpoints being tested on Render | Render link or YouTube link not submitted; or no evidence of API endpoints or Render connection |
| Data Validation  *(Graded via YouTube)* | 25% | Meets Proficient criteria and each route has data validation, and returns some type of 400 or 500 error if data requirements aren't met | Meets Developing criteria and data validation is being used in each route in project (from previous lesson: GET and POST) | Meets Beginning criteria and data validation is being used in each route for this assignment (PUT and DELETE) | Node project has evidence of data validation (for example: checks for at least one variable's existence, or for a specific data type/rule before continuing) | GitHub link or YouTube link not submitted; or no evidence of data validation |
| Error Handling  *(Graded via YouTube)* | 25% | Meets Proficient criteria and each route has error handling, and it returns some type of 400 or 500 status if error is caught | Meets Developing criteria and error handling is being used in each route in project (from previous lesson: GET and POST) | Meets Beginning criteria and error handling is being used in each route for this assignment (PUT and DELETE) | Node project has evidence of error handling (for example: at least one try/catch) | GitHub link or YouTube link not submitted; or no evidence of error handling |
| Update API Documentation  *(Graded via YouTube)* | 20% | Meets Proficient criteria and contains documentation for each route in the application | Meets Developing criteria and the API documentation can test the endpoints on Render | Meets Beginning criteria and at least one of the following HTTP request types are present in the documentation: GET all, GET by id, PUT, POST, and DELETE | The swagger.json file is present (or Apollo server link for GraphQL) | Render, or YouTube links not submitted; or no evidence of API documentation |

Week 7

Lesson 7: OAuth

   Overview

Introduction

Welcome to lesson seven! This lesson, we will start learning about authentication in our applications.

Topics for this lesson:

* Authentication Overview
* OAuth Introduction
* OAuth2 vs OAuth1
* Getting started with OAuth
* Node.js API + OAuth + MongoDB

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%207%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L07)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

Authentication Overview

OAuth Introduction

OAuth2 vs OAuth1

Getting started with OAuth

Node.js API + OAuth + MongoDB

   Team Assignment

Overview

**Purpose**: Identify peers with similar interests and create teams for final project  
**Task**: Complete the assignment

For this lesson, you will form teams for your final project. You will be given the opportunity to form your own team based on project ideas that you like, or other students that you enjoy working with. We are forming teams in this lesson so that you have enough time to get to know your team, decide on project specifics, and figure out how you'll work together before you submit your project proposal in lesson 9. Then, in lesson 10, you'll start development work on the project.

* Each person will post an idea to the Final Project Ideas channel in Teams.
* You need to post at least one solid idea. Briefly include what the app is, what it'll do, what all would go into, and how the database would look.
* Comment on each others's posts as you see ones that you are interested in. Try to find a couple of students that you would enjoy doing a particular project with. Communicate with them about forming a team together.
* Once you have a group of up to three students, go into I-Learn, select the "People" tab in the side menu, select the tab W07–W13, and pick a group that hasn't already been taken for each of you to join. Each of you will need to drag your own name into your group.
* Submit the quiz in I-Learn. This will include the student names in your group so that your instructor can create private channels for each group in Teams.
* If you are unable to find a group, or do not submit this assignment, you will be assigned a group to work with and you will likely have very little say in what project you will work on for the remainder of the semester.
* Your instructor will add you to a private group channel in Teams.

   Personal Assignment

Overview

**Purpose**: Gain experience developing a modern API  
**Task**: Complete the assignment

Learning Objectives

By the end of this assignment the student will be able to do the following:

* Continue learning through continued development efforts

Assignment Description

For your personal assignment you will work on your project that you've been working on since lesson 05.

* Your assignment this lesson is to simply work on your project. Please be sure to review the [project requirements and rubric](https://cse341.netlify.app/projects/2) so you know what all to include by the end of lesson 08.
* Be aware that your project will require a login system. You are encouraged to start implementing this with OAuth during this lesson.
* As you work on your project, make frequent commits. The instructor should be able to see your project progressing. This will also help you to finish in a timely manner.
* In lesson 08, you will need to have a login system implemented in your application. So the more you can finish this lesson, the better.
* Push to GitHub.
* Publish to Render.
* Create a brief video demonstrating the functionality of your assignment. Upload it to YouTube (*public* or *unlisted*).
* Submit GitHub, Render, and YouTube links in I-Learn.
* Be sure to review the rubric below to see how you will be graded on this assignment.

Week 8

Lesson 8: OAuth

   Overview

Introduction

Welcome to lesson eight! This lesson focuses on authorization, and how to use OAuth to effectively authenticate users.

Topics for this lesson:

* OAuth and Swagger
* JSON Web Tokens (JWT)
* JWT meets OAuth

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%208%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L08)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

OAuth and Swagger

JSON Web Tokens (JWT)

JWT meets OAuth

   Team Assignment

Overview

**Purpose**: Research, learn, and share industry standards, best practices, and helpful resources.  
**Task**: Complete the assignment.

The team activity for this lesson will consist of participating in a developer forum. Each person in this class will learn from different resources, try different things, and end up with a unique knowledge base. This forum is an opportunity for you to share industry standards and best practices regarding the technologies that we are using and that you're learning about in this lesson. To receive full credit for this assignment, you must do the following:

* Use the channel in Teams called Developer Forum for this assignment.
* Pick a topic from this lesson and write a post that teaches at least one best practice or industry standard for that topic.
* Include the lesson number at the top of your post along with your title (for example: L03 - POST requests in Node.js)
* The post should consist of at least 200 words.
* Include at least two sources to site your claims.
* The post should be informative, accurate, relevant, and not just a copy and paste of another post.
* Respond to at least two peers before the end of the lesson.
* Finally, use the I-Learn quiz to report your participation in the forum.

   Personal Assignment

Overview

**Purpose**: Implement an authentication system.  
**Task**: Complete the assignment.

Learning Objectives

By the end of this assignment the student will be able to do the following:

* Deploy Node.js API to the web
* Incorporate authentication system with OAuth into Node.js application
* Host GET, POST, PUT and DELETE endpoints in application
* Produce API documentation that allows API testing
* Validate all data before processing API requests
* Handle errors in application

Assignment Description

For your personal assignment you will complete your project that you started in lesson 5. You should already have all of your routes set up with documentation, validation, error handling, and so on. Now, you will also add security either through JWTs or some from of OAuth.

* Please be sure to review the [project requirements and rubric](https://cse341.netlify.app/projects/2).
* For this assignment (if you have not done so already), you will add security measures to your personal project using OAuth. The user should be able to create an account, log in and log out successfully, and view things that aren't available if they are not logged in.
* If you store user credentials in MongoDB, be sure to use bcrypt (or a package of your choosing) to hash passwords so no plain-text passwords are ever stored in the database.
* Be sure your Swagger documentation is accurate, including all of your security measures (and routes that have been effected by these measures).
* Once completed, push any changes to GitHub and verify changes in Render.
* Record a brief video demonstration that shows you using the Swagger documentation successfully sending requests to each route. Also include evidence that your MongoDB cluster is being updated.
* Post this video to YouTube (public or unlisted are both fine, whatever you prefer).
* Submit the following links in I-Learn (Your assignment will receive a zero if these three links are not included)
  + GitHub repo
  + Render site API Contracts (it should be formatted like this: https://cse341-code-student.onrender.com//api-docs)
  + YouTube video

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| Deployed to the Web  *(Graded via YouTube)* | 20% | Meets Proficient criteria and video shows use of Render CONFIG VARS to connect to MongoDB | Meets Developing criteria and app connects to MongoDB | Meets Beginning criteria and Render url can be opened without any errors | Node.js app deployed to Render | Render link or YouTube link not submitted |
| OAuth  *(Graded via YouTube)* | 20% | Meets Proficient criteria and the video shows that each protected route (needs authentication) requires authentication before access | Meets Developing criteria and user can log out using OAuth | Meets Beginning criteria and user can log in using OAuth | Evidence of OAuth is in the Node.js project | GitHub link or YouTube link not submitted, or No evidence of OAuth in project |
| Database  *(Graded via YouTube)* | 15% | Meets Proficient criteria and at least one collection features documents with at least 7 fields | Meets Developing criteria and database has at least 2 collections | Meets Beginning criteria and database has a single collection | Database exists and is shown in the video | GitHub link or YouTube link not submitted, or No evidence of Database in project |
| HTTP Requests  *(Graded via YouTube)* | 15% | Meets Proficient criteria plus at least 2 collections have a PUT or DELETE request (MongoDB change is shown in video) | Meets Developing criteria plus at least 2 collections have a POST request (MongoDB change is shown in video) | Meets Beginning criteria plus at least one HTTP Request is present for each collection in the database | Several HTTP Requests are in the project and work (shown in video) | GitHub link, Render link or YouTube link not submitted |
| API Documentation  *(Graded via Render)* | 10% | Meets Proficient criteria and the documentation can test each endpoint (GET, POST, PUT, DELETE are all modified when testing from the documentation) | Meets Developing criteria and the documentation is published (either to Render at "/api-docs" route, or to apollo server) | Meets Beginning criteria and documentation is organized by collection | The swagger.json file is present (or Apollo server is shown in video for graphql extra credit) | Render link, GitHub link, or YouTube link not submitted. Or no evidence of API documentation |
| Data Validation  *(Graded via GitHub)* | 10% | Meets Proficient criteria and each route has data validation, and returns some type of 400 or 500 error if data requirements aren't met | Meets Developing criteria and data validation is being used for each PUT and DELETE route | Meets Beginning criteria and data validation is being used for each GET and POST route | Node project has evidence of data validation (for example: checks for at least one variable's existence before continuing) | GitHub link or Render link not submitted |
| Error Handling  *(Graded via GitHub)* | 10% | Meets Proficient criteria and each route has error handling, and returns some type of 400 or 500 status when errors get thrown | Meets Developing criteria and error handling is being used for each PUT and DELETE route | Meets Beginning criteria and error handling is being used for each GET and POST route | Node project has evidence of error handling (for example: at least one try/catch) | GitHub link or Render link not submitted |
| Extra Credit  GraphQL instead of REST  *(Graded via YouTube)* | 20% | Meets Proficient criteria and REST is not used at all, only GraphQL (should be shown in YouTube video) | Meets Developing criteria and project uses GraphQL to access and modify MongoDB (should be shown in YouTube video) | Meets Beginning criteria and project uses GraphQL for some aspects (should be shown in YouTube video) | GitHub link (perhaps in previous commits) and YouTube video show that GraphQL was attempted | GitHub link, Render link, or YouTube link not submitted |
| Extra Credit  TypeScript  *(Graded via YouTube)* | 20% | Meets Proficient criteria and there is not a single file in project with ".js" extension. All JavaScript files have been replaced with error-free TypeScript (.ts) files | Meets Developing criteria and project uses TypeScript for all files in "controllers" folder | Meets Beginning criteria and project uses TypeScript for some aspects (should be shown in YouTube video and evident in GitHub repository) | GitHub link (perhaps in previous commits) and YouTube video show that TypeScript was attempted | GitHub link, Render link, or YouTube link not submitted |

Week 9

Lesson 9: API Gateways and Managers

   Overview

Introduction

Congratulations! You've made it to lesson 9! You will no longer have any individual assignments in this course that you need to submit, you will just work on your final project. Outside of submitting a project proposal this week, you will also learn about API Gateways and Managers.

Topics for this lesson:

* API Gateways
* Final Project Proposal

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%209%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L09)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

What Is An API Gateway?

Popular API Gateways and Managers

   Team Assignment

Overview

**Purpose**: Create a detailed plan of your final project.  
**Task**: Complete the assignment.

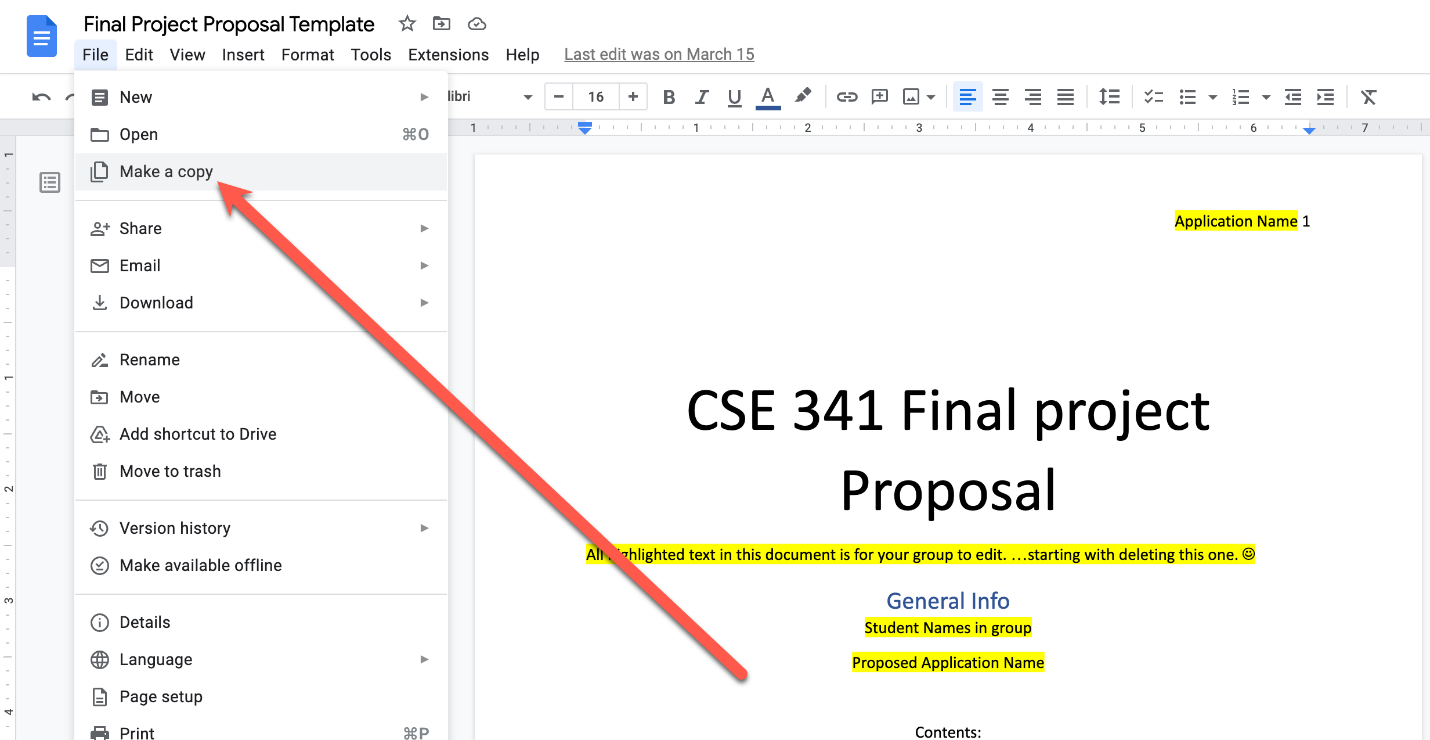
Learning Objectives

By the end of this assignment the student will be able to do the following:

* Work with a team to plan a backend API
* Produce project documentation
* Rank yourself against your peers

Assignment Description

Final Project Proposal

* You have had some time to get to know your team and discuss different projects. For this assignment, you will submit your official project proposal.
* As you decide on a project, review the [overall project requirements](https://cse341.netlify.app/projects/3) to ensure it is a good fit.
* Use this [starting template](https://docs.google.com/document/d/1utfhj7zHVOzpxvuFYoETTOaTgzHHX0av8dvn104vR9A/edit?usp=sharing). You won't be able to edit it, so please make a copy:
* You will notice that this template is relatively comprehensive. It may take a while to properly complete, but this will assist you greatly as you begin work on your API contracts in lesson 10.
* This template will prompt you to answer important questions and plan out how project development will look. You will submit the link to your team Google Doc in I-Learn. Be sure you share the Google Doc with your instructor and TA (or make public).

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| Professionalism  *(Graded via Google Doc)* | 20% | Meets Proficient criteria and document has no grammar or spelling errors | Meets Developing criteria and no highlighting is left. All previously highlighted sections are complete | Meets Beginning criteria and document has all of the original template sections | Google doc link submitted | No Google doc link submitted (or grader doesn't have permission to view it) |
| Database  *(Graded via Google Doc)* | 20% | Meets Proficient criteria and plans are in place for at least 4 different collections | Meets Developing criteria and a list of types of data/collections is present | Meets Beginning criteria and description of what database will be used is present | Google doc link submitted | No Google doc link submitted (or grader doesn't have permission to view it) |
| Schedule  *(Graded via Google Doc)* | 20% | Meets Proficient criteria and each task has a student assigned to it | Meets Developing criteria and schedule shows what tasks will be completed in each lesson | Meets Beginning criteria and Project Scheduling section has content | Google doc link submitted | No Google doc link submitted (or grader doesn't have permission to view it) |
| API Endpoints  *(Graded via Google Doc)* | 20% | Meets Proficient criteria and each collection has API endpoints for GET, POST, PUT and DELETE | Meets Developing criteria and API Endpoints are shown for each database collection planned | Meets Beginning criteria and API Endpoint section has content | Google doc link submitted | No Google doc link submitted (or grader doesn't have permission to view it) |
| Team Evaluation  *(Graded via I-learn)* | 20% | Meets Proficient criteria and each team member name is in order of who did the most work | Meets Developing criteria and each team member name is listed | Meets Beginning criteria and spread factor (high, medium, or low) is included | Quiz submitted in I-learn | No submission in I-learn |

Week 10

Lesson 10: API Gateways and Managers

   Overview

Introduction

Welcome to lesson ten! This week you will continue to work on your final project by writing API contracts.

Topics for this lesson:

* API Gateway Practice
* Final Project Contracts

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%2010%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L10)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

Setting up an Azure Gateway

   Team Assignment

Overview

**Purpose**: Design an API and create accessible API documentation  
**Task**: Complete the assignment

Learning Objectives

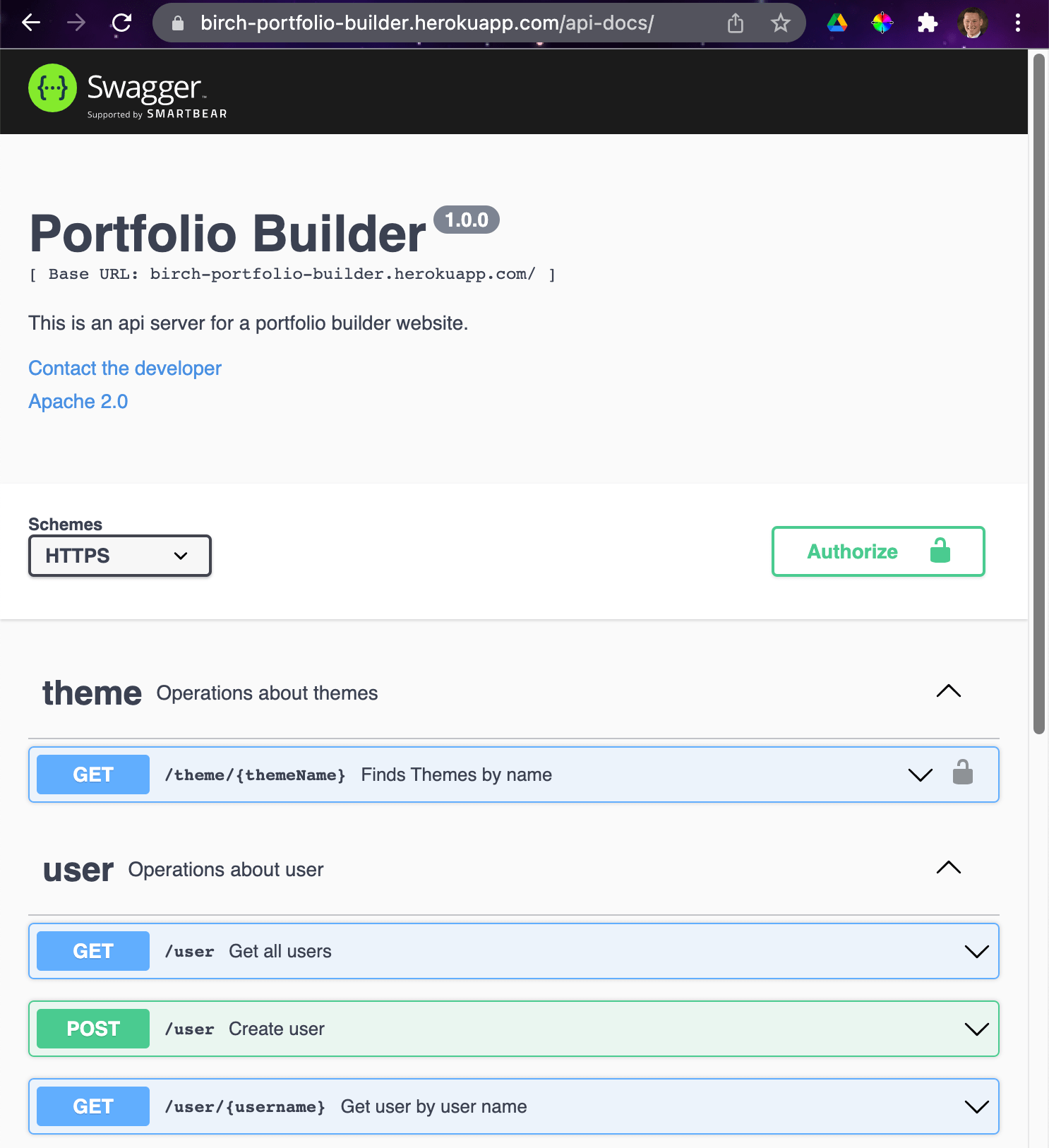
By the end of this assignment the student will be able to do the following:

* Work with a team to design an API with documentation
* Deploy API documentation
* Rank yourself against your peers

Assignment Description

Final Project Contracts

* Your assignment for this lesson will be 100% dedicated to your team project for this portion of the semester.
* Create your Swagger API documentation that will be used by your backend team and the frontend team (for on campus students). It might be helpful to review the resources from week 4. *Having these contracts in place is probably one of the most difficult parts of a project.* Once these are in place, you can just write code until your Swagger doc can successfully execute these requests. It sets up a plan, an outline and a way to test your entire application.
* Your Swagger documentation should be published to Render and available at the following route: /api-docs.
* Create a brief video that goes through the endpoints planned out in the documentation. Upload it to YouTube.
* Submit your YouTube link and your API contracts in I-Learn. (This will be your Render link, for example: https://cse341-code-student.onrender.com//api-docs as shown in the following example:



Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| Project Creation  *(Graded via YouTube)* | 45% | Meets Proficient criteria and has documentation in place ("/api-docs" or Apollo Server for GraphQL) | Meets Developing criteria and is deployed to Render | Meets Beginning criteria and project runs without errors | Node project has been created | GitHub, Render or YouTube links not submitted |
| Deployed  *(Graded via YouTube)* | 45% | Meets Proficient criteria and the documentation can test at least one endpoint successfully (Should show in video) | Meets Developing criteria and documentation has endpoints for each API Endpoint planned in the Lesson 9 Proposal (API Endpoint Planning section) | Meets Beginning criteria and documentation is organized by collection type | Deployed link submitted - Render link submitted with "/api-docs" in the url (or Apollo link for GraphQL). Video uses this deployed link for all demonstrations | Render link or YouTube link not submitted |
| Team Ranking  *(Graded via I-learn)* | 10% | Meets Proficient criteria and includes additional comments about how the team worked together | Meets Developing criteria and includes list of team members in order of work load this lesson | Meets Beginning criteria and includes spread (high, medium, low) | Submitted ranking | No ranking submitted |

Week 11

Lesson 11: Testing

   Overview

Introduction

Welcome to lesson eleven! This lesson you are going to learn how to write unit tests using the Jest framework. You will also continue to work on your final project.

Topics for this lesson:

* Why Testing?
* Testing Frameworks for JavaScript
* Writing Unit Tests with Jest

[Have Feedback?](mailto:birchn@byui.edu?subject=CSE%20341%20Course%20Feedback%20%20%20%20%20%20%20%20&body=Hi%20there,%0D%0A%0D%0A%20I%20have%20feedback%20regarding%20lesson%2011%20of%20CSE%20341.%20%20%20%20%20%20%20%20%20%0D%0A%0D%0A.....%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20anything%20in%20this%20lesson%20that%20isn%27t%20clear,%20or%20could%20be%20explained%20better?%20%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Do%20you%20see%20any%20typos%20or%20inconsistencies%20that%20should%20be%20fixed?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Is%20there%20an%20aspect%20of%20this%20lesson%20that%20is%20extremely%20difficult?%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Please%20be%20detailed%20in%20your%20message%20(including%20screenshots%20if%20applicable).%20%20%20%20%20%20%20%20%0D%0A%20%20%20%20%20%20Thank%20you%20for%20helping%20make%20this%20course%20more%20awesome!%20%20%20%20%20%20%20%20%0D%0A....." \t "_blank)

   Learning Material

Resources for this lesson are accessible to you in [this repository](https://github.com/byui-cse/cse341-code-student/branches/all?query=L11)

The following learning material may be helpful but is not required. Please reference the syllabus for more details.

Why Testing?

JavaScript Testing Frameworks

Writing Unit Tests with Jest

   Team Assignment

Overview

**Purpose**: Gain experience working in a team.  
**Task**: Complete the assignment.

Learning Objectives

By the end of this assignment the student will be able to do the following:

* Start development work on a Node.js API as part of a team.
* Rank yourself against your peers.

Assignment Description

Continue work on your team projects (keeping in mind the [Final Project Rubric](https://cse341.netlify.app/projects/3)). You will submit a status update in I-Learn.

*Remember that you will need to write tests for each of your api endpoints. Lesson 12 will also focus on testing.*

Rubric

| **Criteria** | **Weight** | **Mastery** | **Proficient** | **Developing** | **Beginning** | **Missing/Incomplete** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **100%** | **90%** | **78%** | **65%** | **0%** |
| Team Ranking  *(Graded via I-learn)* | 100% | Meets Proficient criteria and includes additional comments about how the team worked together | Meets Developing criteria and includes list of team members in order of workload this lesson | Meets Beginning criteria and includes spread (high, medium, low) | Submitted ranking in I-Learn | No ranking submitted in I-Learn |