Capstone Project Overview for Nodejs Express

In this project, you'll create a portfolio site to showcase your projects and achievements. The site will contain a landing page, an about page where you'll have a chance to share contact info and talk a little about yourself, and a series of project pages to show at least five projects your own projects or arbitrarily from other places. You shall continue growing this portfolio until the end of the term when you will need to add all your completed work in this file.

In this project: you'll create a JSON file to store all the data about the projects you've created. You'll use Pug to complete provided templates that utilize the JSON to generate the markup that is ultimately displayed in the browser.

You'll use Node.js and Express to:

- Import the required dependencies
- Link the JSON with the Pug templates
- Set up routes to handle requests
- Set up the middleware to utilize static files like CSS
- Handle errors
- Set up a server to serve the project
- After building this project, you should have a comfortable working knowledge of Node.js, Express and Pug, setting up a server, handling requests, working with server-side JavaScript, and building a back end project.

Before you start the project check and download the project files supplied:

- `mockups` - contains three .png files that demonstrate what the final project should look like.
- `public` - contains the necessary CSS and client side JS for the project. For the basic requirements of this project, you won't need to do anything with these files, except to reference the folder when you set up your middleware.
- `views` - contains the four Pug files you will need for this project. You won't need to create any new Pug files, but you will need to add info to each Pug file to make it work with your project. We've provided comments in each file to help guide you through what needs to be done in each file.

To complete this project, follow the instructions below.

Initialize your project

Open the command line, navigate to your project, and run the npm init command to set up your package.json file.

Add your dependencies

At a minimum, your project will need Express and Pug installed via the command line.

Create an images folder in your directory to store your images.

Add a profile pic of yourself that you would feel comfortable sharing with potential employers. It should present well at 550px by 350px.

Take screenshots of your projects. You will need at least two screenshots for each project.

A main shot for the landing page which should be a square image that can display well at 550px by 550px.

Between one and three additional images that can be any dimensions you want, but work well in this project as landscape images that present well at 1200px by 550px.

Add your project data to your directory

Create a data.json file at the root of your directory

The recommended structure for your JSON is to create an object literal that contains a single property called projects. The value of projects is an array containing an object for each project you wish to include in your portfolio.

Each project object should contain the following properties:

id - give each project a unique id, which can just be a single digit number starting at 0 for the first project, 1 for the second project, etc.

project_name

description

technologies - an array of strings containing a list of the technologies used in the project

image_urls - an array of strings containing file paths from the views folder to the images themselves. You'll need a main image to be shown on the landing page, and three images to be shown on the project page.

(optional) You may want to add other properties such as a URL to your live projects or link to social media or other data.

Setup your server, routes and middleware

Create an app.js file at the root of your directory.

Add variables to require the necessary dependencies. You'll need to require:

- Express
- Your data.json file
- (Optional) the path module which can be used when setting the absolute path in the express.static function.
- Set up your middleware:
- set your “view engine” to “pug”

- use a static route and the express.static method to serve the static files located in the public folder
- Set your routes. You'll need:
- An "index" route (/) to render the "Home" page with the locals set to data.projects
- An "about" route (/about) to render the "About" page
- Dynamic "project" routes (/project or /projects) based on the id of the project that render a customized version of the Pug project template to show off each project. Which means adding data, or "locals", as an object that contains data to be passed to the Pug template.
- Finally, start your server. Your app should listen on port 3000, and log a string to the console that says which port the app is listening to.

(Optional) Handle Errors

If a user navigates to a non-existent route, or if a request for a resource fails for whatever reason, your app should handle the error in a user friendly way.

Add an error handler to app.js that sets the error message to a user friendly message, and sets the status code.

Log out a user friendly message to the console when the app is pointed at a URL that doesn't exist as a route in the app, such as /error/error.

Refer to the video on Error handling Middleware, which is linked in the project resources list.

Complete your Pug files

Go through each of the four Pug templates to inject your data. The Pug files contain comments that detail each change you will need to make. You can and should delete these comments when you are finished with this step. But you should wait to do so until everything is working as it should, in case you need to refer to these notes during development.

Note: Consider adding a target attribute set to _blank on the <a> tags for the live links to your projects so that they open in a new window.

Layout, CSS and styles

The layout of the finished project should match the provided mockups.

(optional) To really make this project your own, you should customize the CSS, changing colors, fonts or anything to make your app look better.

Add good code comments

Finish and Submit. I may provide you with alternative submission instructions.