Requirement	Weight
Create and use at least two pieces of custom middleware.	**5%**
Create and use error-handling middleware.	**5%**
Use at least three different data categories (e.g., users, posts, or comments).	**5%**
Utilize reasonable data structuring practices.	**5%**
Create GET routes for all data that should be exposed to the client.	**5%**
Create POST routes for data, as appropriate. At least one data category should allow for client creation via a POST request.	**5%**
Create PATCH or PUT routes for data, as appropriate. At least one data category should allow for client manipulation via a PATCH or PUT request.	**5%**
Create DELETE routes for data, as appropriate. At least one data category should allow for client deletion via a DELETE request.	**5%**
Include query parameters for data filtering, where appropriate. At least one data category should allow for additional filtering through the use of query parameters. **Note:** DO NOT use API keys; this makes it more difficult for instructors to grade finished projects efficiently.	**5%**
Utilize route parameters, where appropriate.	**5%**
Adhere to the guiding principles of REST.	**10%**
Create and render at least one view using a view template and template engine. This can be a custom template engine or a third-party engine. If you are stuck on how to approach this, think about ways you could render the current state of your API's data for easy viewing.	**8%**
Use simple CSS to style the rendered views. **Note**: This is not a test of design; it is a test of serving static files using Express. The CSS can be *very simple*.	**2%**
Include a form within a rendered view that allows for interaction with your RESTful API.	**3%**
Utilize reasonable code organization practices.	**5%**
Ensure that the program runs without errors (comment out things that do not work, and explain your blockers - you can still receive partial credit).	**10%**
Commit frequently to the git repository.	**5%**
Include a README file that contains a description of your application.	**2%**
Level of effort displayed in creativity, presentation, and user experience.	**5%**

Bonus Objectives

The objectives listed here are **not required**. Ensure that your application meets the requirements above before attempting to further expand your features.

These bonus objectives *cannot* increase your overall score above 100%. Successful completion of these objectives can, however, make up for lost points above. **Ensure your application works as outlined by the requirements above before attempting these objectives, time permitting.**

Include a practical usage of regular expressions within route paths. Note: A forced, arbitrary usage of this technique will not earn you any bonus credit. This must be **practical** and **sensible** in order to be considered for credit.	**+1%**
Research and effectively use at least one third-party Node package for practical, sensible purposes. This cannot be a package that has been used in examples and lesson materials thus far. Step outside the box and be creative!	**+1%**