SER421: Web Applications and Mobile Systems Fall 2020 Lab 6 40 points

The goal of Lab 6 is to convert your Lab 5 HTTP-based API to a RESTful API.

In the code walkthrough videos for "Coding REST" I went through a non-REST and RESTful API development of 2 of your Activity 1 requirements from Lab 5 below. This lab asks you to complete the RESTful API for the remaining requirements. Lab 2 Activity 1 requirements are in **black**, Lab 5 extensions are in **red**, and Lab 6 (this lab) requirements are in **blue**. But again, in a nutshell, you are simply converting your existing API to a REST design. NOTE: There is NO Activity 2 (you do not have to update the client-side app).

Activity 1 (40 points): NewsService REST API
For this activity you will implement a news service in NodeJS Javascript code. *The set of requirements R1-R6 below is exactly the same as Lab 2 Activity 1. However, the difference is this time you will need to use an endpoint (URL) and accept input data via HTTP as specified below in blue. You will need to decide whether to use GET, POST, PUT, PATCH, or DELETE, and what the format of the input data should be. Good principles of design will be part of the grading criteria for R1-R5 and the new requirement R7.*

NewsService Requirements
The NewsService is an object representing a content management system for a hypothetical news organization. The NewsService should provide these features:

R1. The ability to write a new news story to the persistent store. *You will need the author and the story attributes (C2). This should be at endpoint ~~/create~~. This one has been done for you in the starter code.*

R2. The ability to update the headline of an existing news story. *You will need the author and new headline (title) and whatever you consider identifying information to know what headline to change. This should be at endpoint ~~/editTitle~~. Now you have to determine how to design and implement a RESTful update to an existing title. This includes the endpoint, verb, response codes, navigation, errors, and payload format.*

R3. The ability to change the content of an existing news story. *You will need the author and new content (title) and whatever you consider identifying information to know what story content to change. This should be at endpoint ~~/editContent~~. Now you have to determine how to design and implement a RESTful update to existing story content. This includes the endpoint, verb, response codes, navigation, errors, and payload format.*

R4. The ability to delete an existing news story. *You will need the identifying information of a story to know what to delete. This should be at endpoint ~~/delete~~. You have to determine how to design and implement a RESTful delete of an existing story. This includes the endpoint, verb, response codes, navigation, errors, and payload format.*

R5. The ability to return a collection of NewsStory objects based on a filter, where the filter checks for one or more of:

   a. Substring of the title
   b. Date Range
   c. Author

   You will need to determine how to accept input for each of these query parameters. This should be at endpoint ~~/search~~. This one has been partially done for you. It does a basic search based on title and author, but needs to be extended to include date range, which will require some massaging of the parameters. Also, better error cases need to be handled.

R6. Construct 10 total test cases in a file named NewsService_test.txt or .json that exercises each one of R1-R5, R7. *Guidance:*

   a. For R2-R4, perform an update headline, change content, or delete on an existing news item. Then do one where you are attempting to perform any one of the operations on a non-existing new item. (4 total test cases)
   b. For R5, provide one test case involving all 3 filter attributes that succeeds, and one that fails (error in query syntax) (2 total test cases).
   c. Create one new test case for R7 below that succeeds (returns a story), and one that returns nothing (2 total test cases).

For Lab 6, do not worry about R1 (since it is already done), but update your test cases to use your new REST API. NOTE: the token is gone from this lab, so you do not have to worry about it in your tests. As with Lab 5, you should be using cURL or Postman collections to construct and submit your test cases.

Additional Requirement (R7) in Lab 6: One common endpoint provided in RESTful APIs is the ability to get a specific resource by an identifier. The API we gave you returns such as identifier in the POST. Add one more REST API for retrieving a specific story by id. Note that this is distinct from R5!

Constraints:

C1. Your server should listen on port 3000

C2. While there is starter code, you do not have to use it; you may instead choose to use Express or Restify (in fact I recommend you do, even though you would have to re-implement R1 and R5 from scratch, in the long run you will have cleaner code, work faster, and learn something resume-friendly!). If you use a framework, provide a package-lock.json so we can npm install (do not submit your entire node_modules directory!!!)

C3. You may not use other RESTful frameworks or libraries. However, any other npm modules or libraries you think you might need you can use. Make sure your package.json indicates only what you are using!

Grading Guidance:

1. Each of R2, R3, R4, and R7 is worth 8 points. Completing R5 is worth 4 points. R6: 8 tests are worth 4 points
2. For each of R2,R3,R4, and R7, *4 of the 8 points will be on REST design*. Are you using the proper Verb? Are you using the proper endpoint and parameters/payloads? Are you using the proper response codes? Are you returning a RESTful payload? Are you handling errors?

EXTRA CREDIT – these are new to Lab 6 (20 points):

EC1. (8 pts) Provide an https-compatible version of your API (Hint: see the https example in your NodeHTTP folder in the class repository). Make sure you turn in your keys so we can run the solution.

EC2. (12 pts) Presently our solutions are subject to a type of injection attack called JSON injection. How? Well, we accept story content in 2 of our API methods, we store that content in a file, and then we potentially (as in lab 5) deliver that content to browser-based applications. To combat this, input data needs to be sanitized. Implement a sanitization algorithm to prevent JSON injection attacks.

SUBMISSION INSTRUCTIONS (READ CAREFULLY and ASK QUESTIONS!):

1. Create a zipfile named <asurite>_421Lab6.zip where <asurite> is your ASURITE id. No RAR or 7zip files!
2. The zipfile should have a root folder with the NodeJS code and related files. Keep the given code we gave you under ./newsModel. You may organize the directories how you wish, and may or may not use modules and package management (npm). Your README should explain exactly what is required to run your code, and you should test this process!
3. If you do choose to use npm (ask us to do an npm install), please do NOT submit your node_modules directory you generate. This makes your submission very large, and we want to rely on your package(-lock).json.
4. I strongly suggest that you get a stable solution to a part, save it, and then move on. We do give partial credit. We allow as many submissions as you want to do and only grade the latest! For this lab this means you should do each requirement of Activity 1 with its corresponding test cases one at a time. They are each independent! Turn in what is **stable!**
5. If you decide to do either EC, please document in the Readme, and I advise submitting in a new directory (EC) as these tasks might get tricky and you do not want to inadvertently jeopardize your main solution.