SER421: Web Applications and Mobile Systems Fall 2020 Lab 2

The goal of Lab 2 is to get you working in NodeJS, understanding the event loop, and the low-level HTTP module.

Important: You may not use any Javascript in the browser in this assignment. You may use CSS if you wish but it cannot be used to implement any of the functionality, it may only be used for aesthetics (we do not grade you on aesthetics however, so to be safe we recommend simply not using any CSS or JS in the browser at all).

Activity 1 (25 points): NewsService

For this activity you will implement a news service in NodeJS Javascript code.

NewsService Requirements (each requirement is worth 5 points)

The NewsService is an object representing a content management system for a hypothetical news organization. The NewsService should provide these features: *R1-R5 are worth 4 points*

- R1. The ability to write a new news story to the persistent store
- R2. The ability to update the headline of an existing news story
- R3. The ability to change the content of an existing news story
- R4. The ability to delete an existing news story
- 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
- R6. Construct 10 total test cases in a file named NewsService_test.js that exercises each one of R1-R5. *Guidance:*
  - a. For R1, simply write one new story (1 test case)
  - b. For R2-R4, perform an update headline, change content, or delete on an existing news item. Then do a 4th one where you are attempting to perform any one of these operations on a non-existing new item. (4 test cases)
  - c. For R5, 3 test cases, one per each of a-c. 1 test case for *a* and *c*. 1 test case for *a*, *b*, and *c*.

Test cases should be written as a sequence of calls seeding then using the persistence store. We will modify to make the results change, testing boundary conditions against empty sets, multiple results, etc. ½ point per valid test case for 5 points.

Constraints/Non-Functional Requirements (NFRs) on the NewsService:

- C1. IMPORTANT: Your "persistence store" here is an internal (in-memory) data structure that you design. You determine the data/object structure to use. There is no File I/O required (but see the Extra Credit 1).
- C2. A NewsStory is a type of object that represents a news article and is described by the following attributes: author (a username), title, public flag, story content, and date.

Overall Constraints:

- C3. Put your code in 1 file, NewsService.js. There is no need for modules, though you may use a module.exports if you wish (and document in the README). Include your test cases as NewsService_test.js. If you have additional test cases merely include them in the zipfile and describe in your README. *(5 pts)*
- C4. IMPORTANT: You do not write a user interface, you specify the API for your object types in your README.txt, and provide example starting files and a sample test case of each service. That is, you must provide examples of how to instantiate your service object, and how to invoke it so we can test it manually. *(5 pts)*
- C5. Your code must be clear and well-written. *(5 pts, additional if we cannot understand your code well enough to grade)*

Activity 2 (55 points): Implement a simple News delivery application using Node HTTP

For this activity you will implement a web application leveraging your solution from Activity 1.

Background:

There are 3 roles in the system: "Author", "Guest", and "Subscriber". Here is what each can do:

- a. Guests can read any public news story
- b. Subscribers can read any news story, public or private
- c. Authors can read any public story, and private stories that s/he has authored

Login/Logout (use HTTP POST): **(12 points)**

- R7. On the landing (home) page, provide an HTML form asking for a username, a password, and a radio button that selects the Role (default to Guest). Your login logic: *4 points*
  - a. We will fudge authentication by simple testing that the username and passwords match (obviously not how we really want to do it!). If the login fails, display a page indicating failure with a link back to the landing page.
  - b. If login is successful take the user to the "View News" page (R10 with no filter by default)
  - c. If you are not logged in this is the only page you can see!
- R8. All screens for a logged in user should indicate the username and role, and provide a "Logout" link that logs the user out and returns the user to the landing page. *4 points*
- R9. When the user returns to the landing page of the application in their browser, the application should remember what username that was last used *on that browser*, and their role. The landing page should say "Welcome [Guest/Author/Subscriber] <username>, please enter your password", and have the username pre-filled in the textbox for entering a username. *4 points, and this should be done via a specific mechanism (you need to figure this out) discussed in the class. If done any other way we will deduct 2 points.*

Features (use HTTP GET or POST as you feel is best): **(43 points)**

R10. A View News ("sub" landing) page will display the titles of all news stories for all users. This page should do the following:
   a. If the user in her/his role can View the story, then the title should be a hyperlink to that story (invokes R12).
   b. If the user in her/his role is not allowed to view the story, then only display the title of the story (not hyperlinked).

Based on the role-based requirements above. You have to test this requirement, both parts, for all roles:

- *For a Guest, only Public stories should be hyperlinked and viewable (3 pts), and non-public only title (3 pts)*
- *For an Author all Public plus stories that Author has authored should be hyperlinked and viewable (3 pts), and non-public/non-authored only title (3 pts)*
- *For a Subscriber, all stories should be hyperlinked and viewable (3 pts)*

R11. If the Role is Author then the View News page should have a link at the top for Creating a new news story. If clicked, an HTML form should be displayed allowing the Author to create a new news story with all of the fields, with "Save" and "Cancel" options to the form. Upon "Save", persist the story via the NewsService. *(6 points)*
   a. If the save fails, re-display the Create Story page and provide an informative error message
   b. If the save succeeds, then return the user to an updated View News page (R10)

R12. Render news stories that are hyperlinked from the View News page in an HTML-friendly format that you design. We do not grade on aesthetics, and no Javascript is allowed. It just has to be complete (all fields of a news story) and readable. News stories are provided by the NewsService. Each story structure is defined in Activity 1 Constraint C2. *(6 pts).*

R13. If the user is the Author of the story being viewed, then provide a link on the page rendering the story that allows the Author to Delete the story *(6 pts)*
   a. If Delete is selected, invoke the delete functionality from Activity 1 R4 and return to the View News page
   b. If the Delete fails for some reason, re-display the story with an error message at the top.

R14. A user should not be able to go to any page directly without logging in first. That is, you should not allow "bookmarkable" URLs; if the user attempts to go directly to a page via the browser bar then redirect to the landing page. *(4 points)*

R15. Error-checking:
   a. You must decide the most appropriate HTTP method to use for any link or form (GET or POST). Your code should use only the proper method for a given action. If the wrong method is used, you should return the proper HTTP response code (you should look this up!). *4 pts*
   b. Any unknown URL presented to the web server should return the proper response code. *2 pts*

Constraints (points shown are deductions; you do not "earn" points for adhering to constraints: (*points are deductions*)

C1. No Javascript in the browser whatsoever. CSS may only be used for embedded styling (no external links) and may not be used to satisfy any functional requirements (i.e. no "hiding" content via CSS to accomplish role differentiation).

- *If Javascript or CSS is used on the front-end to satisfy any requirement then give 0 points for that requirement).*

C2. No 3rd party packages or libraries unless we have discussed them in class. If you have a question on it then ask before you just go use something! *Functionality supported by unapproved 3rd party libraries will receive 0 points. This constraint is specifically mean to deter you from using a framework like Express. However parsing libraries are fine.*

C3. URLs: the landing page should be at the root URL (/). You must run on port 3000. *3 points*

C4. As stated above your web application must use the NewsService from Activity 1 without modification. *10 points off for modifications*

C5. End users should never see stacktraces or other "developer" errors. Trap all errors you can think of and present user-friendly error messages and directions for recovering to the user. *3 points*

C6. Name your main solution file activity2.js. Our test scripts will rely on this so please follow this instruction! You may use multiple-file organization and modules if you choose, again a well-written README is your (and our) friend! *(5 pts)*

EXTRA CREDIT:

EC1. (10 pts) You may implement Activity 1's NewsService using File I/O with the 'fs' package. You may choose between synchronous and asynchronous I/O as per your preference (IMHO synchronous is way easier). You may use features we have not discussed yet, namely promises and/or async (if you don't know what they are then I wouldn't try this route), or you may simply deal with callbacks, your choice. Activity 2 should work without modification.

EC2. (10 pts) In Activity 2, introduce an event emitter that emits an event every time a user logs in, logs out, fails a login, or there is an HTTP error. The event should capture the client IP address and information relevant to the event, like the username for login/logout, or contextual information if it is an HTTP error. Add a listener that logs these events to a file activity2.log. How often should a logger write to a logfile? You may not use a 3rd party logger for this, I am asking you to write your own!

SUBMISSION INSTRUCTIONS (READ CAREFULLY and ASK QUESTIONS!):

1. Create a zipfile named <asurite>_421Lab2.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. You may populate whatever test cases you like in these directories with your code.
3. In the root folder include a README.txt with any information you want us to know, and requested by the lab specification..
4. If you are familiar with node modules and packaging best practices, you may use them to build and deploy your solution. Use the Readme to describe how you do it. You do not have to do this, we are fine running a straight command-line program.
5. I strongly suggest, especially on programming problems, 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!
6. If you decide to do either EC, please document in the Readme, and indicate if it is in the regular submission or if you chose to implement it in a new directory. I recommend a new directory so you do not jeopardize your regular solution.