Lab 9

Due Nov 23, 2016 by 5pm **Points** 100 **Submitting** a file upload

CS-546 Lab 9

Dynamic Pages and Preventing Security Holes

You will create a very simple, unauthenticated application that stores notes in files and retrieves them to display on the page. You will accomplish this by creating 3 pages that are enhanced with jQuery and AJAX.

Data you will store for each note

How you store all this data is up to you, however it must be stored in a file and retrieved from a file.

You will have to store:

- Note title
- Note due date
- Note summary, which can be formatted with HTML.
- Note body, which can be formatted with HTML.

Pages

Page 1: /

The first page of your application will list out the list of stored notes by title, displaying their summary, with a link to each note's individual note page. The notes will be sorted by due date

Page 2: /new

The second page will provide a form with all needed content for a note. When submitted, the event will be captured and sent via AJAX to the server to create a new note.

If a new note is created, you will automatically redirect to the note's display page. (page 3)

If there is an error, the error will be displayed to the user without the user leaving the page.

This page must have a link to the note-list (page 1)

Page 3: /:note

This page must have a link to the note-list (page 1)

This page will show all the information about the note.

You will provide a link to the *next note due after this note*, and when clicking that link you will, using AJAX and jQuery, update the content of the page to be the content of the next note due.

Vulnerabilities

XSS Attacks

I will be testing for XSS attacks, so you must never allow yourself to be vulnerable to an XSS attack!

File Vulnerability Attacks

Remember, you are storing data in files! You will have to make sure that I can't gain access to *any* information that I am not supposed to gain access to.

References and Packages

Basic CSS info can easily be referenced in the **MDN CSS tutorial** **(https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started)** . If you need a quick CSS referfence,

You will use the **express-handlebars** package as your templating engine.

You can reference the **express-handlebars repository** **(https://github.com/ericf/express-handlebars/)** for details on adding the module; you may also want to check out the **handlebars website** **(http://handlebarsjs.com/)** for details.

You will use the **express** package as your server.

You can read up on **express** **(http://expressjs.com/)** on its home page. Specifically, you may find the **API Guide section on requests** **(http://expressjs.com/en/4x/api.html#req)** useful.

You may use the **lecture 4 code** **(https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_4)** as a guide.

You may use the **lecture 5 code** **(https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_5)** as a guide.

You may use the **lecture 6 code** **(https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_6)** as a guide.

You may use the **lecture 8 code** **(https://github.com/Stevens-CS546/CS-546-WS-Summer-1/tree/master/Lecture%20Code/lecture_8)** as a guide.

Requirements

1. You **must not submit** your node_modules folder
2. You **must remember** to save your dependencies to your package.json folder
3. You must do basic error checking in each function
    1. Check for arguments existing and of proper type.
    2. Throw if anything is out of bounds (ie, trying to perform an incalculable math operation or accessing data that does not exist)
    3. If a function should return a promise, instead of throwing you should return a rejected promise.
4. You **must remember** to update your package.json file to set `app.js` as your starting script!

5. **Your HTML must be valid (https://validator.w3.org/#validate_by_input)** or you will lose points on the assignment.
6. Your HTML must make semantical sense; usage of tags for the purpose of simply changing the style of elements (such as `i`, `b`, `font`, `center`, etc) will result in points being deducted; think in terms of content first, then style with your CSS.
7. **You can be as creative as you'd like to fulfill front-end requirements**; if an implementation is not explicitly stated, however you go about it is fine (provided the HTML is valid and semantical). Design is not a factor in this course.
8. **Your client side JavaScript must be in its own file and referenced from the HTML accordingly**.
9. You must pass accessibility tests.
10. You must plug in all common security issues!
11. You must error-check all forms for common errors.