CptS 489—Web Development Spring 2020

**Individual Assignment #8: OAUTH Authentication and Data Searching/Sorting**

*Last modified 31 March 2020*

|  |  |
| --- | --- |
| Worth: | 10% of overall course grade |
| Assigned: | 5 March 2020 |
| Due Dates: | * GitHub repository: 27 March (Friday) 2020 at 9:10 a.m. * OSBLE post-mortem reflection post: 27 March 2020 at 11:59 p.m. * OSBLE post-mortem response post: 28 March 2020 at 11:59 p.m. |
| Late Policy: | * 10% deduction if submitted up to 24 hours late * Assignments more than 24 hours late will *not* be accepted * Late post-mortem posts and responses will *not* be accepted |
| GitHub invite: | <https://classroom.github.com/a/K7Q46_Jh> |

# Overview

This final individual assignment covers more ground than previous IAs and is worth twice as much. Building on the single-page React.js React app you completed in IA#6 and deployed to AWS in IA#7, you will upgrade the app with the following features to make it more industrial strength:

1. Implement OAUTH authentication using Passport and at least one provider other than GitHub
2. Implement local authentication using Passport
3. Store all user accounts and user data records persistently in a [MongoDB database](https://www.mongodb.com/) (use [Cloud Atlas](https://www.mongodb.com/cloud/atlas) for your remote database)

In addition, this assignment has four **extra credit opportunities** to inspire you to take the app further as time and interest allow:

1. Implement [password encryption in your database](https://solidgeargroup.com/hashing-passwords-nodejs-mongodb-bcrypt/) for increased security
2. [Send password reset emails](https://meanstackdeveloper.in/implement-reset-password-functionality-in-node-js-express.html) to users who forget their passwords
3. Support [free-text search on data records](https://www.npmjs.com/package/react-search-box)
4. Support the [ability to sort data records](https://www.florin-pop.com/blog/2019/07/sort-table-data-with-react/) by a specific column header.
5. Use [https://](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https.html) to serve your deployed app.

Implementing each extra credit option adds up to 5% to your assignment grade—up to 25% extra!

# Learning Objectives

Through completing this individual assignment, you will learn and/or reinforce your ability to

* use Passport.js and Express-Session.js middleware to support OAUTH and local user authentication; to serialize users into a session cookie, and to deserialize users into the server session;
* implement Express.js middleware routes to respond to GET and POST client requests;
* use POSTMAN to test Express.js middleware routes (a.k.a. APIs)
* use JavaScript promises, await/async, and try/catch to support asynchronous function calls and gracefully handle run-time exceptions;
* use Mongoose.js middleware to create database schema, connect to a local or remote database, and to perform database CRUD operations in a MongoDB database;
* deploy a client/server project composed of an Express.js server and a React.js client to Amazon Elastic Beanstalk;
* set up certificate to support the https:// connections on a server;
* support sorting data records in a React.js client app both ascending and descending order by any data column header (extra credit);
* support searching data records within a React.js client app through a free text search, with search results updating on every keystroke (extra credit);
* support password encryption within a MongoDB database (extra credit);
* send emails to users to enable them to reset their passwords (extra credit);
* set up a local Git repository and a remote Git repository on GitHub.com.
* set up and use a GitHub project (a.k.a. Kanban) board to track task completion and identify issues and questions that arise as you work.
* use the GitHub to commit code to a local Git repository, and push updates to a remote Git repository on GitHub.com.
* use an IDE of your choice ([Visual Studio Code](https://code.visualstudio.com/) recommended) to write and deploy web programming projects.
* properly submit an assignment by (a) creating a new *branch* for your assignment work, (b) publishing your web app to GitHub Pages, and (c) issuing a *pull request* when your assignment is ready to be graded.

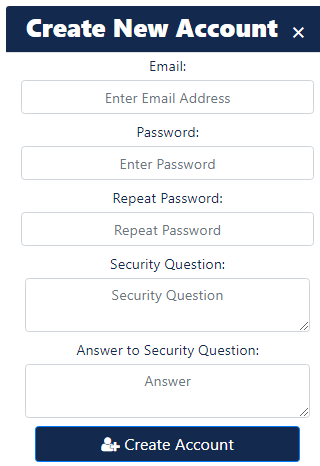
# Requirements

IA#8 builds significantly on prior projects—most notably, the client and server app project developed in IA#6 and deployed to AWS in IA#7. In addition to fulfilling the client-side app requirements for IA#6, your app must fulfill the *new* requirements for IA#8:

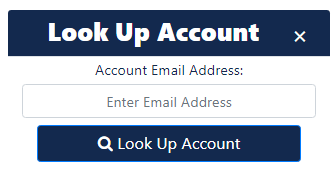
### Visual Design

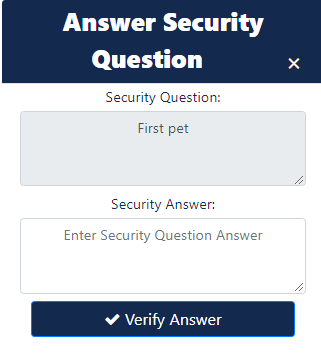
### 

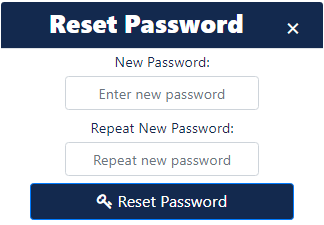
1. Your login page contains icons at the below the “Login” button to support logging in through at least one OAUTH services that is not GitHub. You must use bootstrap-social buttons and FontAwesome social icons for a clean look. (See screenshot above)
2. Account Creation and Reset Password links must be available below the “Log In” button on the Login page. (See screenshot above)
3. You must support account creation through a dialog box that looks something like this:



1. You must support password resets through a dialog box sequence that looks something like this:

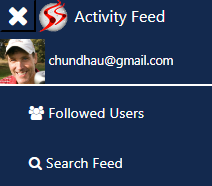






*Note: If you choose the extra credit option to implement password reset emails, then you need to show only the first dialog box in the above sequence, along with a follow-up message that says “If that account exists, a password reset email has been sent.”*

1. In the side menu, the current user’s profile image and display name appear at the top, e.g.,



1. (*Extra credit*) A search box is displayed prominently at the top of the app’s data table page.
2. (*Extra credit*) Each header column in your app’s data table page contains up and down arrow buttons immediately to the left of the header label.

### Functionality

Your app should have the same functionality as your IA#6 app plus these **additional** features:

1. **Login**. On the login page, the user may click on an OAUTH provider button below the log in button to login against the OAUTH provider. If this option is taken, the user must be redirected to the authentication page for that service. Upon successful authentication, the user must be redirected to your app’s default landing page.
2. **Create account**. From the login page, the user must be able to create a new account by clicking the “Create Account” link and then filling in the fields in the “Create New Account” dialog box.
3. **Reset password**. From the login page, the user must be able to reset their password by clicking on the “Reset your password” link and then completing the above dialog box sequence
4. **Reset password** (*extra credit option*)*.* From the login page, the user must be able to reset their password by clicking on the “Reset your password “ link, entering their email into a dialog box, and clicking on a link in a password reset email sent to them.
5. **Data Search** (*extra credit option*). On the data table page, the user must be able to enter search terms into the search box at the top of the page. On each keystroke, the data table must present an updated list of data records that match the search text. Note that records that contain the search text in any field should appear as matches, with the matching text highlighted in the search results.
6. **Data Sort** (*extra credit option*). On the data table page, the user must be able to click on the up and down arrow buttons in a given data table header to sort the data records in ascending (up arrow) or descending (down arrow) order with respect to that column’s data. The data table must instantly be updated to present the new data record order, with the arrow button just clicked highlighted to indicate that sorting has been performed.

### Code Structure, Contents and Deployment

1. You use the same project structure for an Express.js server and a React.js client described in class: a server.js and package.json in the project root directory, and a client/ subdirectory containing the React app, with the production build in the client/build subdirectory.
2. You implement the server in server.js using the following Node.js middleware packages: express.js, express-session.js, mongoose.js, passport.js, and path.js.
3. You store user account information in a users collection of a MongoDB database.
4. You store users’ data records in records collection in a MongoDB database.
5. You use React.js to implement the same app structure you implemented for IA#6. The structure includes
   1. a fixed top bar with app icon and page title
   2. a fixed bottom bar with at least three buttons that can be used to navigate to different *areas* of the site.
   3. A dynamic hamburger (side) menu whose menu selections change based on which *area* of the site is currently selected.
6. You use the create-react-app toolchain to generate your initial code.
7. The client/ directory of project has the following subdirectories, auto-created by create-react-app: public, src, src/components, src/styles, and node\_modules.
8. Your project has a working favicon.ico.
9. You deploy your project to AWS Elastic Beanstalk.
10. (*Extra credit option*) You set up a certificate (self-signed is fine) on your server [to support the https protocol](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https.html).

### Programming Process

1. In your GitHub repository, you create a project (“Kanban”) board with at least three columns: “To Do,” “Doing,” and “Done.”
2. You divide your project into bite-size tasks and create a card or issue for each task.
3. You move the cards between columns and add documentation to your cards to indicate the issues you encounter and your progress.
4. You commit code regularly throughout the assignment period. This implies that you start the project early and commit at least five times over three days. Avoid making all or most of your commits on the same day right before the deadline!
5. You use descriptive commit messages to clearly describe the code changes you are committing.
6. You post questions and issues that arise to OSBLE activity feed. You Attempt to respond to questions/issues on activity feed.

# Assignment Submission

1. When you start the assignment, do the following:
   1. Clone your final code from the IA#8 assignment repo.
   2. Fork a new branch called IA8. Work on the assignment in this branch.
   3. Create a new project (Kanban) board for this project.
2. As you work on the assignment, do the following:
   1. Add to-do items to your project board and move them from the “To Do” to the “Doing” to the “Done” column to track your progress.
   2. As appropriate convert items on your project board to issues and comment on those issues to track your progress in resolving them.
   3. Commit your code frequently to the remote repository using clear and descriptive commit messages. Remember that you need to perform **at least five commits over a three-day period** to receive full credit.
3. When you are done with the assignment, do the following:
   1. issue a *pull request* in the IA8 branch to indicate that it is ready for review. You must issue the pull request by the assignment deadline.
   2. Deploy your app to AWS Elastic Beanstalk using the procedure described in the W07C2 class notes (and also used for IA#7).
   3. confirm that your web app is accessible at the location to which you deployed in on an AWS server.
   4. **Submit the URLs of your GitHub repo and your app through the IA#8 assignment in OSBLE.**

# Post-Mortem Reflection and Response Posts

After you complete a programming task, a best practice is to take a step back and reflect on how you did and what you learned. You are required to write a “post mortem” reflection in the OSBLE reflection assignment associated with this assignment. Y**our reflection post must be at least 200 words and should make an earnest attempt to reflect on how you did, what you learned, and what you could do better next time**. The following prompts are intended to help get you started by thinking about what you did more deeply:

* Describe a struggle that you overcame when working on this assignment.
* Describe an issue with your assignment that you were unable to resolve.
* Assess your process: What did you do well and what could you improve on to be more effective?
* Provide advice to a future student on how he or she might succeed on this assignment.
* Describe the most fun aspect of the assignment.
* Describe the most challenging aspect of the assignment.
* Describe the most difficult aspect of the assignment to understand.
* Provide any suggestions for improving the assignment in the future.

Once you have posted your reflection, you are required to post a reply of at least **50 words** to the reflection of one other peer. *When selecting a post to reply to, you must choose a post no one else has replied to*. You may not reply to your own post!

At a minimum, your reply post should compare and contrast what you did with what the author of the post did. Here are some prompts to guide your reply to another student’s post:

* What did the student do or learn that was similar to or different from what you did or learned?
* What did the student struggle with that was similar to or different from what you struggled with?
* What can you learn from the student’s reflection?
* What advice do you have for the student?

# Assessment

We will grade your submission according to the rubric on the following page. The grading rubric for your post-mortem reflection posts and responses appears on the page after that. These rubrics also appear in the corresponding assignments in OSBLE, where we will complete the rubrics and provide feedback.

**Grading Rubric for Individual Assignments**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | Weight | D-F Level: Emerging (0-6 pts) | B-C Level: Developing (7-8 pts) | A Level: Mastering (9-10 pts) |
| 1. Visual Design | 25 | Your app fails to address two or more “Visual Design” requirements adequately. | Your app fails to address one “Visual Design” requirement adequately. | Your app fully addresses all “Visual Design” requirements defined for this assignment. |
| 1. Functionality | 35 | Your app does not display when launched and/or fails to address two or more “Functionality” requirements adequately. | Your app displays when launched but fails to address one “Functionality” requirement adequately. | Your app displays when launched and correctly and fully implements all “Functionality” requirements defined for this assignment. |
| 1. Code | 25 | Your app fails to address one “Code Structure and Content” requirement adequately. | Your app fails to address one “Code Structure and Content” requirement adequately. | Your code base fully addresses all “Code Structure and Content” requirements defined for this assignment. |
| 1. Process | 15 | Your development process fails to address two or more “Process” requirements adequately. | Your development process fails to address one “Process” requirement adequately. | Your development process fully addresses all “Process” requirements defined for this assignment. |

**Grading Rubric for Post-Mortem Reflection Posts and Responses**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | Weight | D-F Level: Emerging (0-6 pts) | B-C Level: Developing (7-8 pts) | A Level: Mastering (9-10 pts) |
| 1. Reflection Post | 80 | Post is on time, is at least 200 words and appears to be hastily written; in many places, it is off-topic, difficult to understand, is superficial and/or it has grammar or spelling errors that are distracting. (*Late posts or posts that are not at least 200 words receive a 0*.) | Post is on time, is at least 200 words and makes attempt to reflect on programming process; however, it may be difficult to understand or superficial in a few places and/or have clearly noticeable grammar or spelling errors. | * Post is on time and at least 200 words * Post makes an earnest attempt to reflect on programming process: what you did, what you learned, and what you can do differently next time * Post is easy to read and follow; grammar and spelling are mostly correct |
| 1. Reply to Reflection Post | 20 | Reply is on time, at least 50 words and appears to be hastily written; in many places, it is off-topic, difficult to understand, is superficial and/or has grammar or spelling errors that are distracting. (*Late posts or posts that are not at least 50 words receive a 0*.) | Reply is on time, at least 50 words and makes attempt to compare/contrast own experiences with those of person to whom replying; however, it may reply to a post with other replies; be difficult to understand or superficial in a few places; and/or have clearly-noticeable grammar or spelling errors. | * Reply is on time and at least 50 words. * Reply makes an earnest attempt to compare/contrast own experiences with those of person to whom replying. * Post is easy to read and follow; grammar and spelling are mostly correct. |