

Library Project - Phase 3 of 4 (ReST Interface Integration)

Getting Started

Recall that you already have both a local and a remote repository for your Capstone Library project. In Phase 1 of the Capstone Project, you built the engine that will drive your application using JavaScript, in Phase 2 you explored the pre-built User Interface for your digital library application and you connected your engine to your UI with the help of jQuery, `init()`, `bindEvents()`, and `eventHandlers()`. **Before you begin Phase 3, please merge your final copy of Phase 3 into your master branch if you have not already done so.**

Your task is to set up a database for your capstone project and then create HTTP/AJAX requests to take the place of your HTML5 local storage so that the information in your database can be displayed in your DataTableUI. We have spent time getting your mLabs, Robo 3T, and Postman accounts/apps set and ready for you. If any of these aren't working, please let Kyle or Shambre know so we can help. To do this project, you will first need to be able to navigate to the library-service directory through the terminal and open up your node server.js. This will get your Library API (application programming interface) up and running. From here, although unnecessary and entirely optional, it can be nice to have Robo 3T open so that you can verify when a new data entry is created through your POST method or retrieved through your GET method.

Set Up

I'm working on a formal write-up of the procedures we went through to get our databases set up. These will be shared with you separately from this document at a later time as a reference.

Your file structure at this point should not need to be modified, however, you may want to open up the library-service folder inside of your Atom/VSCode window so that you can reference both your library project and your service layer as you work. You can do this by opening your library project in Atom/VSCode and then navigating to File → Add Project Folder → library-service directory. It is also fine to have the two directories open in separate windows.

Requirements

- Create jQuery Ajax or HTTP requests (it's up to you which to use) to perform the following 5 jobs by interacting with your Library UI
 - a. GET
 - Remove your local storage functionality by commenting it out, or deleting it
 - Render your book data in your table through a GET/Library request
 - This will happen on page load. Reach out to your GET/Library API at your local host address using either (<http://127.0.0.1:3000/Library> OR localhost:3000/Library)
 - Store the results of the GET somewhere, such as in your bookShelf array
 - b. GET:ID
 - Render an individual random book
 - Remember to reference the MongoDB through the id
 - c. POST
 - Create a new book that displays on your DataTableUI view
 - d. DELETE:ID
 - Remove a book from your library database array and from your DataTableUI
 - Add your id on as a data attribute so that you can hook into it to delete that entry
 - e. PUT:ID
 - Update a book object's properties in your DataTableUI view and your database array
 - Remember to reference the MongoDB through the id

What to Do

1. Review the requirements
2. Plan out how you think you can best accomplish these targets and determine various approaches you can try
3. Research and read about asynchronous callbacks and how they work, AJAX requests, and HTTP requests in the documentation from Helpful Materials below
4. Start writing your code

5. Discuss issues with your peers and ask for help early and often!
6. Slack Shambre if you have a pressing issue that you cannot solve
7. Commit and push every time a new module is completed and functional

Helpful Materials

<https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82>
<https://www.upwork.com/hiring/development/intro-to-apis-what-is-an-api/>
<https://medium.freecodecamp.org/understanding-asynchronous-javascript-callbacks-through-household-chores-e3de9a1dbd04>
<https://medium.freecodecamp.org/javascript-callbacks-explained-using-minions-da272f4d9bcd>
<http://callbackhell.com/>
<https://restfulapi.net/>
<https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/>
https://www.w3schools.com/jquery/jquery_ajax_get_post.asp
<https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them>
<http://mongoosejs.com/docs/api.html>
<https://docs.mongodb.com/manual/>
<https://www.codementor.io/wapjude/creating-a-simple-rest-api-with-expressjs-in-5min-bbtmk51mq>
<https://expressjs.com/en/guide/routing.html>
<https://stackoverflow.com/questions/38306569/what-does-body-parser-do-with-express>

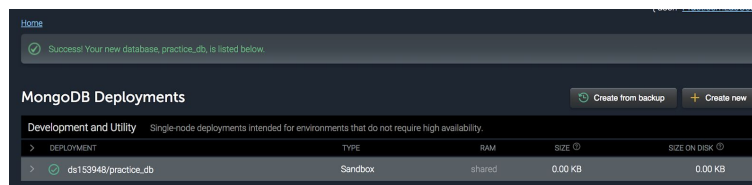
Submitting

- Project due by the end of day Monday, **November 12th, 2018**
 - If you do not think you can finish by this time, Kyle, Brett, and I must receive a Slack message from you saying so by **9:00 am** on November 12th, 2018 (it's okay to tell us you won't finish and then discover that you are able to, but we want to know if this pace is too strenuous and adjust if necessary)

Connecting to a Database in mLab

- mLab

- Open a browser and go to mLab (<https://mlab.com/signup/>). Use the Sign Up/ Now button to access the registration form.
- To create your account, you will need to remember your account name and your username, so make them easy for yourself and write them down, use your techtonic email address, check the agreement box, and click the Create Account button. Be aware that you will be creating more than one set of user credentials through this process and you will have to keep them straight. This can trip you up later, so making them the same can be helpful. Do not use any special characters in your Username or Password, as it will cause issues later. If you must, please be aware of how to convert them properly using ASCII codes and a URL encoder.
- Verify that you are not a robot if prompted to do so.
- Verify your email by going to your techtonic email address, opening the email, and clicking the link provided to verify your email.
- Now, you will click the yellow plus sign to Create New MongoDB Deployments.
- Click on the yellow plus sign to create a new database.
 - There are a few options but choose Amazon Web Services
 - Then click on Sandbox to access the free version
 - Click the blue Continue button
 - Select the U East (Virginia) (us-east-1) region
 - Give your database a name that you will remember and that makes sense for your project, such as “library,” with lowercase letters and no spaces (use dashes or underscores instead)
 - Click Submit order
 - You have a database!!
 - Click on the database name next to the green checkmark



- Click the Add a Collection button
- Name your collection “libraries”

Collection name*

libraries

Preallocate size in bytes†

† If capped, the preallocate size also becomes the max size. The default is ~8 KB. Does not include index storage.

☐ Capped?

CANCEL CREATE

NAME	DOCUMENTS	CAPPED?	SIZE
libraries	0	false	7.98 KB

- Now, you have to create a database user to connect to this database, so click on the Users tab

Database: practice_db

To connect using the mongo shell:

```
% mongo ds153948.mlab.com:53948/practice_db -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI ([what's this?](#)):

```
mongodb://<dbuser>:<dbpassword>@ds153948.mlab.com:53948/practice_db
```

mongodb version: 3.6.6 (MAPv1)

Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on [how to upgrade](#).

A database user is required to connect to this database. To create one now, visit the 'Users' tab and click the 'Add database user' button.

Collections Users Stats Backups Tools

Database Users

[None at this time]

Add database user

- Click on the Add Database User button. Create a new username for your database user (you can make it the same as the mLab account you set up previously if you want to).

NAME	READ ONLY?
Practicedbusername	false

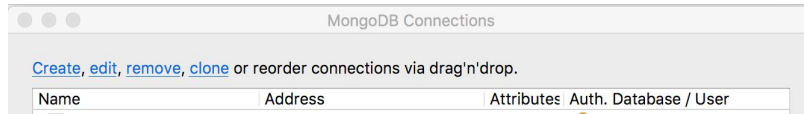
- Now, look at the top of the screen to locate the MongoDB URI, also know as your “connection string.”

To connect using a driver via the standard MongoDB URI ([what's this?](#)):

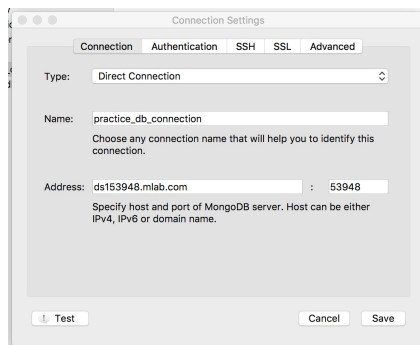
```
mongodb://<dbuser>:<dbpassword>@ds153948.mlab.com:53948/practice_db
```

- Copy this entire line.

- Robo3T
 - Open up Finder (or use Command + spacebar) to search for Robo3T in your applications folder. Open Robo3T.
 - A MongoDB Connections window will pop up. Click Create in the upper left corner.



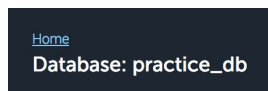
- The type should remain Direct Connection.
- The Name should be any connection name that will help you identify this specific connection.
- Paste the connection string you copied before into the Address input field. It will look like this:
`mongodb://<dbuser>:<dbpassword>@ds153948.mlab.com:53948/practice_db`
- Then, delete everything from the beginning of the string through the @ symbol and everything from the semicolon through the end of the string. It will then look like this: `ds153948.mlab.com`. This goes in the input field on the left. In the smaller input box to the right, copy the last 5 numbers after the ds string (or the whole string of numbers after the colon). The numbers in this example are 53948.



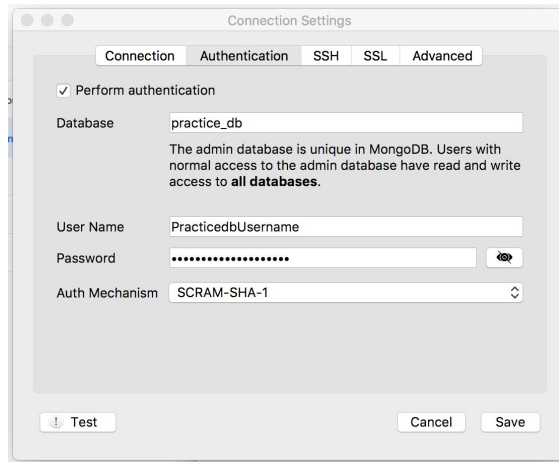
- Click Test to see if the first line is successful. Don't worry about the second line for now, because that is what we will do next.

✓ Connected to **ds153948.mlab.com:53948**

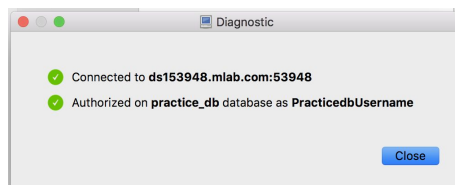
- Click Save and Click on the Authentication Tab.
- Click on the checkbox to Perform Authentication.
- Take the name of your database, which can be seen in the upper left corner of mLab, and input it into the Database input field.



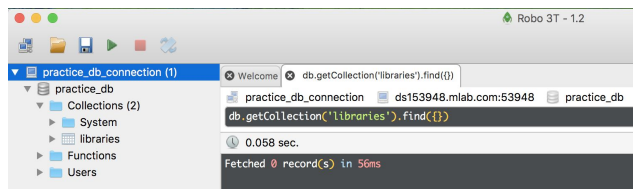
- Use the credentials you created for your database user as the credentials here and leave the Auth Mechanism as SCRAM-SHA-1.



-
- Click Test to make sure both lines are successful.



-
- Click Save and then Connect.
- You should see something like this window. From here, you will be able to see all of your database information after you connect with your database through node.js.



- Postman
 - Open up Finder (or use Command + spacebar) to search for Postman in your applications folder. Open Postman.
 - Terminal
 - Now, go to , accept the assignment, and clone the library-node repository
 - Open up Finder (or use Command + spacebar) to search for the Command Line Terminal in your applications folder. Open your terminal.
 - Once in your terminal, you will need to navigate to the folder you just cloned the repo to using the following commands (yours may be different than mine. I saved mine in my Projects folder as library-node.)

```
pwd
cd Projects
cd library-phase-3
```

- NOW - BEFORE WE MOVE FORWARD, WE MUST OPEN THE DIRECTORY IN ATOM

```
atom .
```

- WITHIN LIBRARY-NODE IN ATOM, OPEN UP THE DB.JS FILE
 - Paste the connection string from mLab into the `mongoose.connect()` method parentheses, like so:
`mongoose.connect('mongodb://<dbuser>:<dbpassword>@ds153948.mlab.com:53948/practice_db');`
 - Replace the `<dbuser>` with your username and `<dbpassword>` with your password.

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://PracticedbUsername:PracticedbPassword!6@ds153948.mlab.com:53948/practice_db');
```

- Save these changes before the next step.
- Go back to the terminal and from the same place you were before, run the following commands:

```
npm install
```

- You will need to install the npm package the first time you want to run the server. However, if you don't remember if you've already installed it, you can check to see if the file contains a `node_modules` directory or a `package.json` file. If so, the necessary dependencies have already been installed and you can just navigate to the folder you want to run your app from. Then execute one of the following commands.

```
npm start
```

OR

```
node server.js
```

OR

```
node server
```


- Any of the three commands above will start your node server. If your node server is operating correctly, you will see “Express server listening on port 3000” in the output of your terminal screen.
- Open up postman again and type localhost:3000/Library. Check to see that you have the request option set to GET and press send.



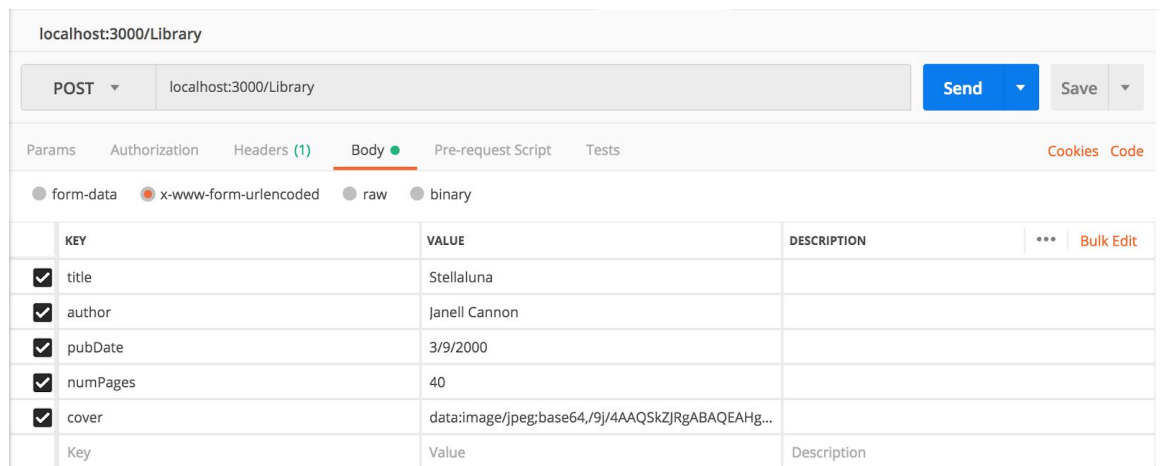
A screenshot of the Postman interface showing a GET request to the URL 'localhost:3000/Library'. The 'Send' button is highlighted in blue.

- Check to see what your status was. Hopefully, it will be a 200. Since you don't have any data in your library database yet, that is about all you should see.



A screenshot of the Postman response bar showing a status of '200 OK', a time of '303 ms', and a size of '262.47 KB'.

- Next, leave the URL unchanged and change the request type to POST. Click on Body, then select x-www-form-urlencoded. Populate the key fields for every key you created in your book constructor or in the Library.js mongoose schema. You may have to adjust your library constructor so that it matches the key names in the schema. Do not change the schema itself. Fill in the value fields with actual data you want to put into your library. Note that the cover image must be passed in as a base64 encoded string. You can use the <https://www.base64-image.de/> to help you switch over your images to the correct format.



A screenshot of the Postman interface showing a POST request to 'localhost:3000/Library' with the body type set to 'x-www-form-urlencoded'. The body is populated with the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> title	Stellaluna	
<input checked="" type="checkbox"/> author	Janell Cannon	
<input checked="" type="checkbox"/> pubDate	3/9/2000	
<input checked="" type="checkbox"/> numPages	40	
<input checked="" type="checkbox"/> cover	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAHg...	
Key	Value	Description

- Click Send. You should see something like this in the bottom half of the page and a 200 status. The id is auto-generated by the database, as well

BodyCookiesHeaders (7)Test Results

Status: 200 OKTime: 198 msSize: 38.38 KBDownload

PrettyRawPreviewJSON

1

{

2

"id": "5be21bd5c56e061292f6e823",

3

"title": "Stellaluna",

4

"author": "Janel Cannon",

5

"pubDate": "2018-11-06T22:55:17.666Z",

6

"numPages": 40,

7

"cover": "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAAgAeAAD/2wBDAAUDBAQEAwUEBAQFBQUGBwwIBwchBw8LCwkMEQ8SEhEPERETFhwXExQaFRE

8

"__v": 0




9

}

- ```
GET localhost:3000/Library Send Save







Pretty Raw Preview JSON

{
 "cover": "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEASABIAAD/4SvRZhpZgAATU0AKgAAAQABwESAAMAAAABAAEAAAEaAAUAAAABAAE",
 "__v": 0
},
{
 "_id": "5aff52fe326d247ba89d4eaa",
 "title": "Animal Farm",
 "author": "George Orwell",
 "pubDate": "2018-12-31T00:00:00.000Z",
 "numPages": 150,
 "cover": "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAQ/ZwCEAAkGBxQTEHUEXMFhUXGB0aGBgYGRofGxseHR8iGhobHRcaHyg",
 "__v": 0
},
{
 "_id": "5b041e2289f3344640d0fdb7",
 "title": "1984",
 "author": "George Orwell",
 "pubDate": "2018-12-31T00:00:00.000Z",
 "numPages": 250,
 "cover": "data:image/jpeg;base64,/9j/4QAYRZhpZgAASUkAAgAAAAAAAAAAAAAAAP/sABFEWnreQABAAQAAABFAAD/7gAQQWRvYmUAZMAAAAA",
 "__v": 0
},
{
 "_id": "5be21bd5c56e061292f6e823",
 "title": "Stellaluna",
 "author": "Janel Cannon",
 "pubDate": "2018-11-06T22:55:17.666Z",
 "numPages": 40,
 "cover": "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAHAgAeAAD/ZwBDAAUDBAQEAWUEBAQFBQUGBwwIBwcHBw8LCwkMEQ8SEhEPERETFhw",
 "__v": 0
}
]
```

- | Collections |           |         |                                                                                          |  Delete all collections |  Add collection |
|-------------|-----------|---------|------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| NAME        | DOCUMENTS | CAPPED? | SIZE  |                                                                                                              |                                                                                                      |
| libraries   | 13        | false   | 493.91 KB                                                                                |                                                                                                              |                                                                                                      |

- If you click on the collection itself, you can also see each entry you entered with Postman (the last entry, in my case).

records / page 10 < > << first < prev [11 - 13 of 13]

|                                                                                                                                                                                           |                                                                                                                                                                         |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>{   "_id": {     "\$oid": "5aff52fe326d247ba89d4eaa"   },   "title": "Animal Farm",   "author": "George Orwell",   "pubDate": {     "\$date": "2018-11-06T22:55:17.666Z"   } }</pre> |   |
| <pre>{   "_id": {     "\$oid": "5b041e2289f3344640d0fdb7"   },   "title": "1984",   "author": "George Orwell",   "pubDate": {     "\$date": "2018-11-06T22:55:17.666Z"   } }</pre>        |   |
| <pre>{   "_id": {     "\$oid": "5be21bd5c56e061292f6e823"   },   "title": "Stellaluna",   "author": "Janell Cannon",   "pubDate": {     "\$date": "2018-11-06T22:55:17.666Z"   } }</pre>  |   |

records / page 10 < > << first < prev [11 - 13 of 13]

•



