|  |  |
| --- | --- |
| Sprint 15, Assignment 7.6 Please also update the doc name with correct numbers. | |
| Assignment type: JS Interactive | |
| Assignment name  Can remain the same as the assignment topic name, or…  can be phrased in the active/imperative voice(depending on the task), e.g. “Create your own X” | "Create a User Registration Model" |
| Description  The essential tasks a student must perform in order to complete the assignment | It's time to use your new database powers in the \*most\* useful way possible!! ... Create a database, using express and mongo, of your favorite fictional characters - say, those most-excellent Marvel superheros, the X-Men! Or, you know, whomever \*you\* want to make lists about ... They're probably almost as cool. |
| Link to full code file (Answer)  Push a full code file to our [GitHub repo](https://github.com/bitdegree-foundation/academy-assignments-code-files) for this sprint’s assignment and paste a link to that file here. Please write the file in a concise and clear format, according to the “[example-js-code-file](https://github.com/bitdegree-foundation/academy-assignments-code-files/blob/master/example-js-code-file.js)” found in our repo. |  |

|  |  |  |
| --- | --- | --- |
| # | Step  Write each small step of the task. These assignments should be doable by students on their own (without any BitDegree instructor help). It can be any number of steps, but keep them pretty clear and separate (don’t combine 2 steps into 1 step). | 1x hint  Write brief text or partial code that would help the student figure out how to move forward or accomplish the given step. |
| 1 | Set-up: Create a folder named "7.6 express-project", navigate inside of that folder (using terminal), and run "npm init -y". Update your package.json to include "type": "module". Run "npm install mongoose". Create an "index.js" file. |  |
| 2 | (In index.js) import mongoose form "mongoose" and log a test message to console; run your index.js file with node to make sure everything is working properly. |  |
| 3 | Create a database (say, named "marvel") and a collection (say, named "characters") using MongoDB's Compass. Once created, navigate (in compass) to the "characters" document - there should currently be no entries in the database ... | Hint: if everything went well in assignment 6.6, you should see your current local databases on the left-side of interface. Click the "+" icon at the bottom-left to create the new database |
| 4 | (Back in your index.js) Connect your index.js to your database, and set up a basic error message by entering the following four lines of code (in index.js): |  |
| 5 | Create a schema of at least four properties for your database entries (say - "code\_name", "real\_name", "team", and "marvel\_comic" - all as strings) and store it in a constant named "xmenCharactersSchema" | Hint: your constant should store a new mongoose Schema |
| 6 | Create a model and store it in a constant called "Member" | Hint: use mongoose's .model method  Hint: the .model method will take two parameters: your database's collection name (in capitalized, singular-form ie "Tank" instead of "tanks"), and the schema you've just proscribed |
| 7 | Using your model, declare four constants, and store in them instances of the members of the all-mighty X-Men (including their code names, real names, and team (some are on 'blue' team, and some on 'gold)). Make sure to have at least one member from gold, and one from blue in your. Make your index.js save each entry to your database after being declared. Run your index.js with node, stop it after a few moments, and then check your database to see whether your entries have saved succesfully. | Hint: your model will act like a constructor, so, each constant should be a 'new Member' |
| 8 | Store your new Member declarations inside of an asynchronus function named "createMembers". Declare a new asynchronous function named findMember. Have it 1) [a]wait until a member of 'gold team' is found 2) store the result in a constant named "target", and 3) log 'target'. Call your findMember function at the end of index.js |  |
| 9 | Declare a new, asynchronous function and name it "deleteAll". Have it 1) find all the characters on the "blue" team and log the results, delete all of those entries, and then 3) exit the process. Call your function (and turn off your other calls - findMember(), createMember() etc). You should have at least one entry remaining in your database ... |  |
| 10 | Turn off your deleteAll call, turn on your createMembers call, and re-run index.js. Check your database - you should have redundant entries. |  |
| 11 | Declare a new, asynchronous function and name it "updateOne". Have it 1) find the first instance of a redundant entry in your database, and store that in a constant named "target", 2) log that old entry/"target", 3) (find and) update the target with new, unique information (change the code name to a new x-men character etc) and store the result in a constant named "result", and 4) log the newly-updated "result". Have your function exit | Hint: you will need to use await  Hint: you can use $set to update more than one field of a database entry at a time |
| 12 | Run your index.js file until you have only unique members of the x-men in your database (no duplicate entries of a character) |  |
| ... |  |  |