~~.~~

1. ~~Create a new file called config.js within the server/config directory using the config.example.js~~
   * ~~Add in your MongoDB Atlas URI (or localhost if you prefer this)~~
   * ~~Sign-up for a free public key~~ [~~https://opencagedata.com/~~](https://opencagedata.com/) ~~and add it to your config file - This key will allow us to access open cage data's geocoding api to retreive coordinates for new buildings you add to the application.~~

**~~YOU MUST USE THE CONFIG.JS, DO NOT HARDCODE YOUR URI OR API KEY ANYWHERE. YOU MUST PLACE EACH VALUE SOLELY IN THE CONFIG.JS -- NO EXCEPTIONS~~**

1. Complete the app configuration in server.js.

* ~~It should serve the static files found in the client folder when a user makes a request to the path /.~~ [~~Refer to this documentation~~](http://expressjs.com/en/starter/static-files.html) ~~for help.~~

Already Done

* ~~It should use the listings router for requests going to the /api/listings path.~~

Followed According to Cheat Sheet

* ~~Last it should direct users to the client side index.html file for requests to any other path~~

Followed According to Cheat Sheet and guiding text

* ~~now run npm start (note that this is now using nodemon instead of node to run your server. This package allows the server to refresh whenever changes to the file are made) to see how our server is working and navigate to http://localhost:5000 in the browser. Try some of the routes and see what happens. Right now, we only serve up index.html and the server hangs for all the other routes in our server.js file~~

Tested and works

1. ~~In MongoDB Atlas, delete your database from Bootcamp2 so you can start clean. Setup an empty database to begin our development.~~

Didn’t delete db but made new one works just fine

1. Implement the ListingModel.js file. This is our mongoose model for our database, it should have the same schema as the previous bootcamp.
   * **~~before you start coding~~** ~~navigate to server directory, review and~~ **~~run~~** ~~the model test jest listings.server.model.test.js~~*~~note:~~* ~~The functionality your code needs to pass, which tests pass, and the content of your database~~

Same tests as Camp2

* + **~~now open~~** ~~the ListingModel.js file, you'll notice that this file looks a lot like the ListingModel.js file from Bootcamp Assignment #2 because it is exactly the same so it should be easy to implement the functionality.~~

Copied and pasted Listingmodel from Camp 2 as instructed

* + **test your implementation** again by running the tests jest listings.server.model.test.js

Passed all tests

1. ~~Reinitialize your database by running jest listings.restart.test.js while cd'd in the server directory. Check your database and make sure you have 147 documents before proceeding.~~

It populated mongodb as instructed

1. Implement the request handlers **update, delete, list** in listingsController.js
   * **implement functionality** update, delete, list in listingsController.js see notes and tutorials identified earlier in this README file for help. **You may want to use** [**postman**](https://www.getpostman.com/downloads/) **to develop and manually test your update, delete, list routes** (*Note:* You will have to start your server to use Postman run nodemon server/server. ) (i) The following should GET a single listing by ID, copy and paste a single entry "id" from your database to replace the tag http://localhost:5000/api/listings/<ID> (ii) The following should GET all the listing - http://localhost:5000/api/listings (iii) Try also to test out your post and update requests.

**Update is from cheat sheet but different if requirement and error handling**

**Delete is from cheat sheet but different error handling**

**List from cheat sheet but different error handling**

* + **run automated tests on your implementation** by simply running npm test. *YOUR SERVER MUST ALREADY BE RUNNING*: nodemon server/server. This executes a chain of tests to help you quickly reset your database and verify the integrity of each component of your project.

1. Make sure your server is functioning correctly by starting it up by running the command nodemon server/server. Manually **browser test your routes** try adding a test to create or update a listing and check the database to see if it is there. You may manually have to deleted the listing once you add it, unless you add a test to delete it.
2. Complete the coordinatesController.js you will need to connect to opencagedata.com to lookup the address provided in the listing and add the latitude and longitude coordinates for the new listing you've added. This will require you to
   * Consider using **postman** to help you test out the right connection string format, see [open cage postman tutorial](https://opencagedata.com/tutorials/geocode-in-postman)
   * **parse the response** for the results you need: latitude & longitude. Note: listingsController.js is using the req.results to update the listing. *Assumption:* For the purpose of this assignment, assume we find a match and the first result returned is our address. The results for building on campus aren't super robust in the open cage data. Resolving this issues is beyond this scope of this assignment.
   * **be sure to add error handling** if the address provided fails due to formatting. You don't want your application to crash.
   * **test** your implementation again to ensure that your application is actually changing the database and retrieving from the database.
   * Consider writing or adapting a test to ensure the latitude and longitude are properly saved.