Final Project - Internet Apps

- **Learn more**
  - https://medium.freecodecamp.org/the-definitive-node-js-handbook-69123
  - https://vegibit.com/node-package-manager-tutorial/
  - https://flaviocopes.com/package-json/
  - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
  - https://javascript.info/async-await
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await
  - https://medium.com/@habibridho/javascript-es7-async-await-tutorial-64275c81ce2e
  - **(optional)** https://reactjs.org/tutorial/tutorial.html
  - **(optional)** http://domenlightenment.com/
  - **Redis:**
    - https://hackernoon.com/using-redis-with-node-js-8d87a48c5dd7
    - https://dzone.com/articles/a-brief-introduction-to-caching-with-nodejs-and-re
    - https://try.redis.io/
    - https://redis.io/documentation

Submit a zip file <yourID>_<firstName>_<lastName>_<partnerFullName>_<partnerId>_FP.zip (e.g. '043462598_Ohad_Assulin_FP.zip').

1. In this project you will build an online store
   a. Choose name and what you are going to "sell"
   b. You should support the following screens and features
      i. Register screen - user **must login in order to see any other page** (**excluding** readme and the login screens)
      ii. Login screen + optional remember me option (otherwise cookies expire after 30 minutes)
         1. One user should already exist: "admin"/"admin"
      iii. Store screen (shows your items) support:
         1. Textual search
         2. Add to cart
      iv. Cart screen
      v. Checkout screen
      vi. All screens must support some kind of menu to enable easy navigation between the screens. It must include a logout button.

vii. Admin screen that exposes the activity of any other user
    1. Should support filtering to see only the data of one specific user
viii. Additional page 2-4 pages with additional functionality (whatever you want and fit) around concepts that connected to the items that you are selling
ix. All the data about a specific user must be kept in redis (however you want). It means that after restarting your application, the users and all of their data still exist and they can login into their accounts. Data that must be persisted:
    1. User details
    2. Cart
    3. Purchases
    4. Login activity
x. Defend as much as possible from DOS attacks

c. Add /readme.html route and describe there the following:
    i. **THE readme file must be done individually**
    ii. Store name
    iii. What are you selling?
    iv. What additional page(s) did you add? How to operate them
    v. What was hard to do?
    vi. Who is your partner? name and <id>. What did you do? What did your partner do?
    vii. Specify all the different route your app supports
    viii. How did you make your store secured?
    ix. Did you implement the store using react.js?
d. Build test.js that test intelligently all the meaningful routes that your server supports. You only need to test the server-side routes.
e. Learn how package.json works and submit your work with the relevant package.json and without the node_modules folder

2. Tech
    a. You can use any npm package you need
    b. Must work with cookies to implement user management
    c. Implement different screens in different node.js files/modules
    d. Implement connectivity to redis in redisConnector.js module
    e. Utilize async/await as much as possible
    f. Upon implementing routes, use the most suitable HTTP method (get/post/put/delete)
    g. Must handle any kind of error, exception and async errors.
    h. Use node fetch() to test your server
        i. **(Optional))Challenge** - develop tests before the application itself.
            1. https://en.wikipedia.org/wiki/Test-driven_development
3. Grading
    a. Functionality

b. Effort
c. Code (async-first, modules, efficiency)
d. User experience
e. Richness /Deepness
f. Security