



NodeJS

PROJECT: INVENTORY APPLICATION

Introduction

Alright! Let's flex our skills a little! That tutorial was humongous, and you learned a *lot* of things. The only way to make it stick is to practice! For this project you are going to create an Inventory management app for an imaginary store. It's up to you what kind of business this is – you could be selling groceries, car parts, baby-toys, musical-instruments, ponies or anything!

Your Inventory app should have categories and items, so when the user goes to the home-page they can choose a category to view, and then get a list of every item in that category. You should include all of the CRUD methods for both items and categories, so anybody that's visiting the site can Create, Read, Update or Delete any Item or Category.

Assignment

1. Before you begin, take a moment to write down all of the models you'll need and the fields that should go in them. It might help to grab a pencil and some paper and literally draw a diagram like you saw in the [MDN tutorial on databases](#).

- Items should at least have: a name, description, category, price, number-in-stock and URL, though you should feel free to add more fields if it seems relevant to the type of business you've chosen.
- Categories should at least have a name, a description and a URL.

1. We're going to follow the basic path that was demonstrated by the MDN tutorial to set up and flesh out your app, so first choose a templating language and generate the boilerplate skeleton with express-generator.

2. Create a new Mongo Collection using the web-interface as demonstrated in the tutorial and then set up your database schemas and models.

3. In the Library tutorial you populated your database with some sample data that was provided in a `populatedb.js` file. Actually understanding how that worked was over your head at the time, but now that you've finished that tutorial you'll be able to understand how it works. [Download the file here](#) and edit it, or re-write it using the specifics of your models and then run it to populate your database!



- 4. Set up the routes and controllers you’re going to need.
- 5. Create all of the ‘READ’ views (i.e. view category, and view item)
- 6. Create all the forms and build out the controllers you need for the rest of the CRUD actions.
- 7. EXTRA CREDIT: For bonus points, try to figure out how to add and upload images for each item. Use [this middleware](#) which was created by the Express team. The documentation in the README there should be enough to get you going.
- 8. EXTRA CREDIT: We will learn about creating users with secure passwords in a later lesson, but for now we don’t want just *anyone* to be able to delete and edit items in our inventory! Figure out how to protect destructive actions (like deleting and updating) by making users enter a secret admin password to confirm the action.
- 9. Deploy it and show off what you’ve done!

Solutions:

Add Solution

NodeJS: (Inventory Application)

6	♥	sher_s7	View Code	Live Preview	
3	♥	pklepa	View Code	Live Preview	
2	♥	AtifCppProgXXI	View Code	Live Preview	
1	♥	Sebastian S	View Code	Live Preview	
1	♥	Alex Pérez	View Code	Live Preview	
1	♥	Daxas	View Code	Live Preview	
1	♥	lprzybyslawski	View Code	Live Preview	
1	♥	DoviMaj	View Code	Live Preview	
1	♥	TryMe	View Code	Live Preview	





Showing 10 most liked submissions - [View full list of solutions](#)

View Course

Login to track progress

Next Lesson

 [Improve this lesson on GitHub](#)

Have a question?

Chat with our friendly Odin community in our Discord chatrooms!

Open Discord

Are you interested in accelerating your web development learning experience?

Get started



[Thinkful](#)



5-6 months



Job Guarantee



1-on-1 Mentorship





About
FAQ
Blog

Success Stories
Contribute
Terms of Use

