# Developer at Droppe - React refactoring task

First of all, thanks for hopping on this, I think you'll enjoy this short task! I've prepared this assignment for you in order for you to really shine and showcase your experience level in high-quality web development and coding practices. My team and I have spent some time making it exciting, so we hope you enjoy it. Ok, let's hop into the story now!

### Story

We won't spend time building anything from scratch today but rather take a look at something that has already been implemented. This app should still be reviewed and refactored before it can be merged back into our production code.

You have volunteered to help out a fictitious "inexperienced developer" who has built the app to its current state. We took a quick look at the code and suspect that there might be a few architectural, as well as quality and performance-related problems that need to be taken care of. Hence your task is to find these problems and improvement areas in the app and fix them.

If you make any changes or fix some parts you need to be able to explain why these changes were required. The current implementation works as intended, but the code is quite bad, **in fact, we’ve identified bad practices in all areas of the codebase**. If you decided not to refactor some parts that you find obvious shortcomings we expect you to explain why you chose not to refactor those parts. In a later interview, we will ask some questions about your changes and request you to motivate your choices.

### Instructions

1. In the drive folder that was shared, you will find a zip file that is the entire app itself, download the zip
2. Initiate your own repo with the existing codebase, create the first commit and push
3. Start working on refactoring and fixing the app, pay attention to [good git commit practices](https://www.codewithjason.com/atomic-commits-testing/#:~:text=%E2%80%9CAtomic%20commit%E2%80%9D%20is%20basically%20a,or%20two%20lines%20of%20code.) and [even better git messages](https://www.freecodecamp.org/news/how-to-write-better-git-commit-messages/) as a way of documenting your changes
4. Identify the most critical part in the app and write a unit test for it, we want to get a sense of your testing strategy but don't look for the whole app to be covered by tests
5. Submit the completed task by sending me the link to the GitHub repo with its complete commit history and a clear list of the most important changes you made
6. The deadline for this task is one week from receiving this link from me in the email, please let me know in advance if the timeline is unfeasible for you

**Estimated workload**: maximum of a couple of hours, don’t spend too much time and see “what we value”-section

### What we value

Two main things: **recognize what's actually important** and **quality over completeness**. Try to complete areas you've started on rather than touching multiple areas and leaving them uncompleted. Note: if something is left undone you should have a clear todo-list of what still needs to be done and where. Try to avoid leaving parts uncompleted, however.

#### Things we pay great attention to:

* Clean code, code quality, modularity & scalability
* Strong understanding of React, javascript, and typescript best practices
* Application architecture and code structure
* Code clarity, and quality of documentation and communication
* Holistic view over the application and questioning all improvement areas
* Confidence and the ability to clearly justify the reasoning behind your choices
* Ability to showcase your experience and skills during the task

### Further points and reading

It’s important to take a very broad and holistic look over the whole application with all of its parts and not only the lines of code (performance, assets, quality, naming, modularity, dependencies, modern best practices, repo structure, …). The guiding attitude should be: what do I need to change so that we can ship this thing of beauty into production with the highest possible quality.

[Fake Production API](https://fakestoreapi.com/), handle this API and its documentation as it would be our production-grade API. That is, interacting with it and its data correctly also from the code.

Most probably all of these points were old news for you with terms such as DRY, best practices, modularity, and so forth. I have my full trust in you and look forward to seeing the end result. Show me your best game! Good luck and hear from you soon!

**Final tips:** Submit the assignment by sending (as a reply to the email you received) us a link to the Github repository and make sure we can access it. Double-check everything before returning the code.