Walmart Mexico Code Challenge

Code Challenge

Introduction:

Believe it or not, Walmart Mexico is a growing startup-like division within Walmart that is looking for fullstack front-end developers like yourself to help it scale to a be a billion dollar revenue division.

This code challenge is meant to mimic the type work we do each day as front-end/fullstack developers in the Walmart Mexico division. Everyday, we ask developers such as yourself to take ownership of the implementation and management of front-end features. This requires understanding of front-end architecture, data models, how-the-feature-works-end-2-end, in addition to implementing features/fixes end-2-end.

Through this challenge, we are looking for signals that indicate that you would be able to handle the unique and interesting engineering challenges we have that take more than just writing of front-end code to fit a UI mock.

Our techstack is React server-side rendered using Next.js, Apollo Graphql interface on Node (bff layer), and a Node for microservices. We don't expect mastery - we prefer fullstack understanding, the ability to quickly learn what you don't know, use technologies properly, and write easy-to-read, reusable code.

Code Challenge

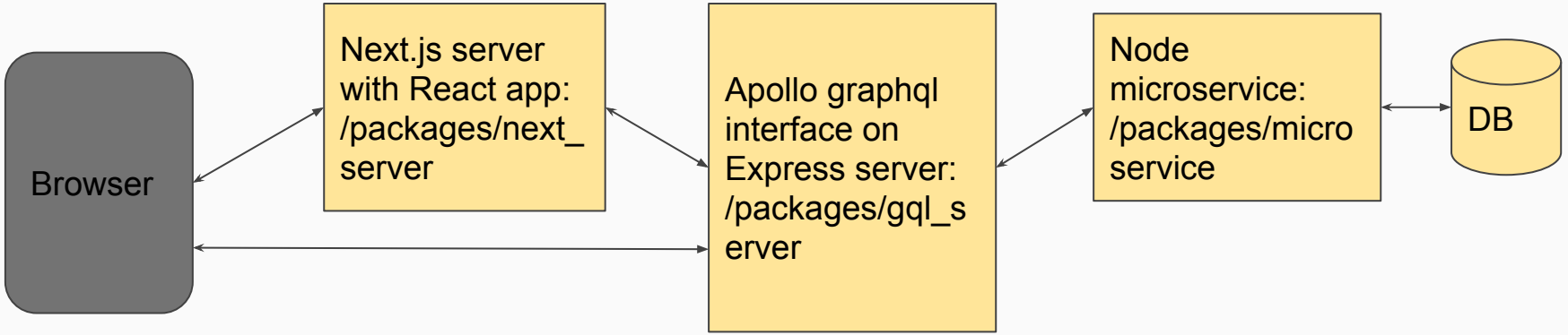
Instructions:

- Fork the code challenge repo to your personal Github account
- Once you are complete with the tasks of this challenge, push the code to your repo and send the link to the recruiter to forward to the hiring team for review.

Code Challenge

Instructions:

- Here is the system architecture so you know what package in the repo does what and how they are supposed to connect in order to start up the code (see README.md for more info):



  - How it works: When a user first lands on any page of the site, the browser will request the page from the Next.js server which will return a server-side rendered html page and the react app. For every subsequent request from the react app (now on the browser) by clicking buttons (such as product tiles), the react app will request data directly from the Apollo graphql server. The graphql server then requests that data from the microservice which is connected to our DB.

Code Challenge

Instructions:

- Here are the tasks to complete using the challenge repo:

1) Create/code a way to have script commands in the package.json of each server folder (microservice, gql, next.js) that can deploy each server in either a dev, qa, or production environment. If you are not familiar with what "environment" means - it is a dedicated set of servers that developers user to develop in (dev), test with (qa), or deploy to production for your real users (production). Then briefly write the instructions you would give to your dev-ops team on how to use it - write it in the tasks_and_solutions.md file in the repo.

2) Create/code a way to have a single command start up all 3 servers for local development (microservice, gql, next.js). Then briefly write the instructions you would give to a new developer on your team on how to use it - write it in the tasks_and_solutions.md file in the repo.

3) The dev team you are working on has asked you to make sure this repo's code always looks pretty and easy to read since other developers from all around the world will be writing to it and some will not know what formatting you like in your code (ex. they may use 1 space vs 2 spaces to indent their code). Describe how you will make sure committed code is always formatted consistently and implement the tool or process - describe it in the tasks_and_solutions.md file in the repo.

Code Challenge

Instructions:

4) Implement the UI design shown and described in the last 2 slides. To fetch data for the UI, use the 2 graphql queries written in the Apollo graphql server to fetch the *userRecommendedItems* and *item* information. Please implement *server-side* rendering for initial page load and *client-side* rendering for subsequent React app interactions as described in our system diagram and in the UI design slides.

5) Your product team wants to make a change to the UI design in the last 2 slides. For non-grocery department items (home and furniture, electronics, apparel), they want to change the specification name of “Weight” to “Shipping Weight”. And if that non-grocery item does not have a “weight” but instead has a “packaged weight”, they would like to use the “packaged weight” value to show as the “Shipping Weight”.

   Implement the code change(s) needed in any of the 3 servers to implement this feature considering that not only the React app, but also iOS and Android mobile UIs for your company will also need to make this UI change as well. Hint - Try to be DRY from the perspective of all the code that needs to be written in all UIs and servers to implement this feature. Briefly describe your solution and why it is good so your product team understands - write it in the tasks_and_solutions.md file in the repo.

Code Challenge

Instructions:

6) The dev-ops team has noticed that calls from the React app to fetch the userRecommendedItems are taking a very long time and it is causing the microservice server to almost crash during peak site usage. Please fix this issue by making changes to either the React app or the graphql server or both to try and reduce this load on the microservice. Briefly describe your solution so the devops team understands - write it in the tasks_and_solutions.md file in the repo.

7) The QA team has also found that if they make a request for an item page for an item id that has a “$” in it (ex. '000$'), they can cause the microservice server to crash because the DB cannot handle an invalid item id with $ value. Design and code a solution in any or in all 3 servers (Next.js, gql server, or microservice) to solve this problem. Hint - Try to be DRY from the perspective of all the code that needs to be written in all UIs and servers to implement this fix. Briefly describe your solution so the QA team understands the fix - write it in the tasks_and_solutions.md file in the repo.

Code Challenge

We know that you may have questions and typically, at work, those questions are resolved when receiving mocks or during sprint-planning. But since this is an interview, make your best guess to unblock yourself and keep on moving. Use comments in code if needed to convey your thoughts. In our division, this type of problem comes up every day and you must be able to make good judgements and be able to proceed even with the lack of clarity.

Also… fix any bugs you find along the way. Not sure if we left some in there just like in real life :P

Good luck! We look forward to reading your implementation.

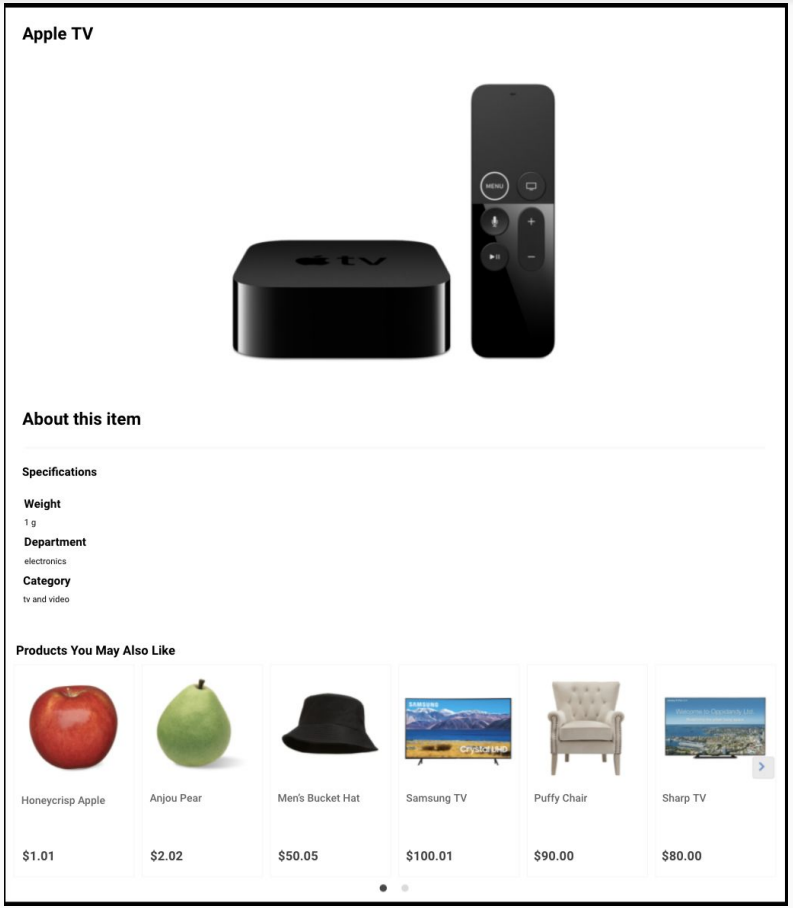
UI Design

- There is no / path page.
- This page here is .../ip/apple_tv/0007?username=james
- The nomenclature for available pages is .../ip/ [the item's name with spaces replaced with _] / [the item's id]

  And if you pass in query string param ?username=james, you are requesting the page to include the recommended items for the specific user which are then used to populate the "Products You May Also Like" carousel.

  And if you pass in ...?username=someoneWhoDoesNotExist, you should not see the carousel appear since there is no data for unknown users

- Technologies to use:
  - use the Next.js server and routing
  - use React Query for fetching data from the Apollo graphql server
  - use React hooks



UI Design

- For the product carousel, if there are too many items, show the little arrow and the dots below indicating there are more than 6 items in the carousel. The maximum number of items in the carousel can only be 12 so if you click on the right arrow, you should reach the end of the carousel and the left arrow appears while the right arrow disappears.
- You don't need to follow it, but if it makes it easier, the font used was Roboto. Try and guesstimate the font sizing.
- Also, notice how the Weight field shows a space between the unit of measure (in this case "g") and the numerical weight.
- Also, when you click on any of the product tiles in the carousel, the React app should make a client-side request to the Apollo graphql server to fetch the item data for the page.
- When you first land on any page of the site (ex. typing the url into the browser and pressing enter) or refresh a page, the page should be server-side rendered.
- FYI - large images for all products and arrows are in the packages/next_server/public/images folder. Please create thumbnail versions for the product tiles if needed.