Introduction

For this challenge, you are requested to create a simple product renting and buying/selling products application called Teebay (You can provide an alternate name if you like for creativity purposes). This solution will have a Front End (FE), Back End (BE) and a DataBase (DB).

PART 1: Preliminary features

For this part, please create the following preliminary features

- Login
- User registration

Assumptions for this part:

- You **do not** have to make the Login feature fully secured. It can be simple string matching for the purposes of this challenge.

PART 2: Implementation documentation

For this part, please create the following features -

- As a user, you can -
  - Add your product
  - Edit your product
  - Delete your product

More about data modelling -

- Teebay will have the concept of categories. A product can be **under one or more categories**. The categories are -
  - ELECTRONICS
  - FURNITURE
  - HOME APPLIANCES
  - SPORTING GOODS
  - OUTDOOR
  - TOYS

PART 3: Rent and buy/sell

For the final part, please create the following features -

- List all products created by all users
- Ability to buy a product. You can assume once you accept buying a product, then the product has been bought.
- Ability to rent a product
- Display all the products bought/sold/borrowed/lent by the user

PART 4: Implementation documentation [MANDATORY]

Please attach a **Part_4_documentation.md** with your submission and **in brief** provide an explanation of how you solved each part of the problem. Think of this part as a “technical documentation” you are providing to the engineering
team who would want to know about this application. Feel free to discuss some of the corner cases that are worth documenting and how you went ahead in solving it.

WIREFRAME DESIGN

A rough wireframe has been provided to you. But you can get creative and create your own designs (with CSS) as long as the functionalities in each of the 3 parts exist. **But the priority is to complete the challenge within the specified time as opposed to better design**.

Password for wireframe access: sazim.10

TIPS FOR THE CANDIDATE

- Please use the following technologies to complete the application -
FE -> React (Use Apollo as the GraphQL client)
- BE -> Backend of your choice (I suggest NodeJS) and **GraphQL** -
FE fetching Data from BE - **GraphQL** only
- The information must be stored in the **apollo cache** (i.e. inMemoryCache) at the Front-end

- When removing any information from DB it must be removed from the **apollo cache** (i.e. inMemoryCache) as well.
- Unnecessary information stored in the apollo cache is restricted.

- DB -> Postgres
- Forms -> React Hooks Form/Formik
- UI -> There are multiple npm packages out there for UI. Feel free to use any or you can choose to write the CSS from scratch. Popular libraries are -> MaterialUI, Semantic UI React (from personal experience, this one is easier to work with), Bootstrap.

- Think of this challenge as not just a coding exercise but software you will be deploying to production. So there are various nuances to think about that **are intentionally left open ended**. For example -
  - Testing
  - User experience (which includes input validation and proper user feedback)
  - FE component architecture and reusability
  - FE routing
  - Database modelling
  - Readability and software best practices
  - Handling practical application corner cases. (For example, what happens when there is a rent time overlap?)
  - Handling error cases

SUBMISSION GUIDELINE

- Finish **as much as possible** in the provided time slot and if you are not able to complete the entire challenge, we would highly encourage you to still submit and we can discuss how you would solve the pending items.
- Please share the challenge via a github link. The challenge needs to run on localhost only. The challenge must have a README that lists out all the steps required to run the project. The challenge can be reviewed in

any type of machine. NOTE: You can assume the reviewer will have Docker installed

- Avoid Code duplication like copy-pasting the same thing (multiple same input fields, cards, etc). Focus on code reusability.

ASSESSMENT EVALUATION POINTS

- **Correctness**: Is your solution complete?
- **Code organization and readability**: Is your code easy to read and maintainable and easily testable?
- **Code design**: Choice of component architecture, data structures and efficiency
- **Framework knowledge:** How well the tools and libraries were used ● **Communication:** How professionally was the challenge written in terms of code comments or any other documentation.

Finally, if you have any questions/concerns or if you think there is any error in any of the parts, please email at: ehsanur.rahman@sazim.io