Form mutation coding task

Background

At Livitay we exclusively work with React functional components and not class based components. If this is new to you this blog post from the React team is worth a quick read - https://reactjs.org/docs/hooks-intro.html. We also place a large focus on best practices, working well as a team and adopting new technologies. As such we are interested in your ability to learn new things and produce easy to understand code for others.

The task

Your task will be to make a simple app that allows you to update an existing activity. As a more representative task for the work we do at Livitay this will be under the following constraints:

1. Working in Typescript
2. React functional components only
3. Usage of apollo generated hooks for queries
4. Usage of Formik to handle the form using Yup for validation
5. Usage of Material-ui for elements

A partially complete project has been setup for use this should form the basis of your work it is available here - https://github.com/livitay/LivitayCodeTest2-Incomplete.

Your goals should be:

1. Update Apollo client networking to automatically attach the logged in users authentication token to headers when possible. The authtoken is given by the following function:
   ```
   firebase.auth().currentUser.getIdToken()
   ```
2. Update the given GraphQL queries to return the required data (use “npm run gen:schema” after each update to update the “generated/graphql.tsx” file)
3. Build a brief activities page using the generated Apollo custom hooks to get the activities for the given user (check the generated file for info here) - To do this you must add a “where” constraint via the “sequelizeString”:
   ```
   sequelizeString: `{"where" :{"partnerId": "${authUser.uid}"} }`
   ```
4. Use formik to build a form with the following fields: “name”, “category”, “collection” and “about”. Clicking on an activity in the activity page should bring this form up as a Dialog (check material-ui). Check constants.tsx for the options for category and collection
5. Make a Yup schema that formik uses to validate the fields. All fields should be required, the name should fall within 5 and 40 characters, the category and collection fields should have either one or two options selected and the about should fall within 10 and 1000 characters.

6. Finally use the update activity mutation to update the given activity

Signin will be handled via firebase and an auth context and hook is supplied. A user account has been set up for you to login with - details below.

Email	jamie@livitay.com
Password	test123

VSCode setup

We would like you to install the following three extensions into VSCode before attempting this task.

Prettier: Prettier auto formats your code and is a very valuable extension. A configuration file is included in the repository. You should enable the “format on save” option for VSCode.
https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode

ESLint: ESLint is a javascript and typescript linting tool and checks your code for you. It is configured for the repository
https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint

Sort-imports: Sort imports orders your imports to keep things tidy. Again there is a setup file. You should enable the “format on save” option for VSCode.
https://marketplace.visualstudio.com/items?itemName=amatiasq.sort-imports

Data

At Livitay we use the apollo client to manage our GraphQL queries and GraphQL Code Generator to generate the Typescript types and Apollo query hooks to interact with our API. A project has been setup with the basic queries in the “queries” folder. By running “npm run schema:gen” GraphQL Code Generator will scan this folder and update the “generated/graphql.tsx” file. The queries in the “queries” folder need updating to ensure you get the data you need. After doing this you will need to run “npm run gen:schema”. If you would like to understand the data returned a bit better you can use our graphical API interface available here - https://livitay-test.appspot.com/graphql

Library info

This task specifically is aimed at giving you some of the libraries we use at Livitay in order to see how you adjust to new items and can cope with the technologies we use. Ensure that you use them as specified. More info for each can be found at the relevant links:

Material UI - https://material-ui.com/
Formik - https://jaredpalmer.com/formik/docs/overview

Apollo Client React - https://www.apollographql.com/docs/react/v3.0-beta/get-started/
GraphQL Code Generator - https://graphql-code-generator.com/

Notes & hints

- All the required libraries are already installed - I would check the package.json for what libraries should be used as a helping hand
- The login form does not use formik, this is on purpose so that you must learn about formik yourself
- The generation also generates types you *should* use
- You may have a submission issue if you don't remove "__typename" and "id" keys. An efficient way to remove them is:
  ```
  const { __typename, id, ...updatedActivityData } = values
  ```
- The activity item can be as simple as a horizontal component that specifies a few bits of data like activity name, price etc.
- Do not worry about things looking amazing design wise - we are not testing your artistic ability - this is entirely up to you
- While working in a team we want neat, concise and understandable code produced so please consider this

Example

A short video file is included in the repository indicating the finished project. If you have any issues watching this file please let us know.