Assignment tasks

1.    Home page

1.1.    Change the colour of the `About` button from blue to white.

2.    Header

2.1.  Refactor `Hamburger` menu

2.1.1.When the `User` clicks on the `hamburger` menu the side drawer should open on the right side of the screen instead of the left over the scroll bar.

2.1.2.The drawer should appear under the header and not over.

2.1.3.Add icons for each menu item by your judgment.

2.1.4.Rename the `Login` menu item based on the current User state to `Login` if the User hasn’t logged in (the user is guest) and `User` if he is logged.

2.1.5.The `Tickets` menu item shouldn’t be visible for guests.

2.1.6.The header should go full width.

3.    Database

We have already created a base setup for connecting to a Firestore Firabase. You will have to create your own database in Firebase and copy paste the required data in ` Task\src\firebaseSetup.ts`.

4.    User

The functionality for signing in and out is already implemented and should be working if the database is set correctly.

4.1.  UI

4.1.1.Add `user` icon that the user can click and redirect the sign in/up page next to the `hamburger` menu on the right.

4.2.  UX

4.2.1.If the `User` is logged in the icon should be a human silhouette.

4.2.2.If the `User` isn’t logged (he is guest) in the icon should be a door representing that the user can sign in/up.

4.2.3.When a logged in `User` clicks on the `user` icon he should be redirected to a page that gives him the option to either log out or change his password.

4.2.4.When the `User` enters wrong username or/and password while signing in/up he should receive a message telling him that there’s an error.

4.2.5.In the sign in/up window add button/text that will redirect the `Guest` to a page to reset his password (functionality is partially implemented).

4.3.   Technical design

You already have boilerplate created for the actions, store and a reducer **rename** and use them as you feel suitable.

4.3.1.Create a slice in the Redux store to keep the state of the `User`.

4.3.2.Extract the requests in separate actions and reducer files

4.3.3.Create reducer that will intercept those actions and add them to the store

5.    Create `Tickets` page

5.1.  UI

5.1.1.This page shouldn’t be reachable for guests.

5.1.2.The tickets should be created from the `Contact Us` form,

5.1.3.You have the absolute freedom to implement your own design for listing, editing, and removing tickets

5.2.   UX

5.2.1.The user should be able to see only the tickets that he has created

5.2.1.1.              (Optional)

5.2.1.1.1.                    Infinite scroll

5.2.1.1.2.                    Pagination

5.3.  Operations

5.3.1.Remove an existing ticket

5.3.2.Change the ticket state. The state can be changed only as fallows

5.3.2.1.              Open -> Closed

5.3.2.2.              Open -> Resolved

5.4.  Technical design

You already have boilerplate created for the actions, store and a reducer rename and use them as you feel suitable.

5.4.1.Use `Message` model and service as well as rename them accordingly to `Tickets`

5.4.2.Create a slice in the Redux store to keep the state of the Tickets

5.4.3.Extract the requests in separate action files

5.4.4.Create reducer that will intercept those actions and add them to the store

6.    Tests (Optional)

Test as many functionalities that you feel suitable with Jest.