## **Используя ReactJS, TypeScript создать SPA приложение калькулятора сумм валюты из одной в другую. Необходимо реализовать пользовательские истории**

## **Как неавторизованный пользователь я хочу получить от системы сумму в выбранной валюте, указав исходную сумму и валюту для того, чтобы получить рассчитанное значение моей суммы, но в другой валюте.**

### **Критерии приемки**

* При указании исходной валюты и желаемой валюты и исходной суммы в поле ответа отображается сумма в выбранной валюте (исходная сумма \* курс конвертации)
* Список валют ограничен EUR, USD, JPY, RUB, GBP
* Если валюта источника совпадает с желаемой валютой - сбрасывать поле ответа (должно быть пусто)

## **Как авторизованный пользователь я хочу получать от системы уведомление при изменении суммы на заданное значение для того, чтобы (например) вовремя продать или купить валюту**

### **Критерии приемки**

* Если пользователь авторизован, то ему становится доступны поля "Уведомить меня, если сумма изменится на ХХХ ({выбранная валюта }" и option "В большую сторону" и "Меньшую сторону"
* При "срабатывании" события - осуществляется вывод значения дата + время

## **Дополнительный сценарий: Как авторизованный пользователь я хочу иметь возможность добавлять на страницу несколько записей конвертации, в том числе и уведомления для того, чтобы следить за своим портфелем.**

* При нажатии кнопки "Добавить" значения из формы добавляются в список контролируемых курсов
* При авторизации пользователя в список добавленных добавляется и настройки уведомления

## **Подробности и необходимые вводные для реализации**

1. Для получения данных курсов конвертации нужно использовать API [https://rapidapi.com/kasimurugasen1995/api/currency-value](https://rapidapi.com/kasimurugasen1995/api/currency-value%C2%A0%D0%BD%D0%B0) на бесплатном сайте [https://rapidapi.com/ (для](https://rapidapi.com/%C2%A0(%D0%B4%D0%BB%D1%8F) этого нужно зарегистрироваться-это бесплатно)
2. Авторизацию можно не реализовывать, если вход через любой OAuth2 провайдер будет реализован - будет плюсом
3. Пользователь считается авторизованным, если в Local Storage присутствует значение isAuth = true. Отсутствие флага или установка его в любое значение, кроме True считается неавторизованным
4. Внешний вид, верстка не является критерием прохождения тестового задания.
5. В исходном коде должны присутствовать блоки обработки исключительных ситуаций с информированием пользователя (алерт или сообщение на SPA)
6. Рекомендуется использовать axios для получения данных от API

## **Ожидаемый результат**

* Исходный код выложен в личном git репо (github, gitlab, bitbucket). Репо должно быть доступно для клонирования
* Код должен собраться стандартным способом (npm | yarn install)
* В коде должны быть тесты, запускаемые командой в клонированом репо (указать команду запуска)
* Код должен запуститься локально в браузере
* Реализация дополнительного сценария не ожидается, но приветствуется (и будет существенным плюсом)