Level 2: Mini-paint

Экосистема:

Требуется инициализировать базовое приложение (используя любой готовый Boilerplate вроде CRA или свои наработки), основанное на React / Vue / Angular и реализовать функционал, описанный далее. В том числе, на личном GitHub-аккаунте создать публичный репозиторий с названием этого задания (указано в заголовке документа: Innowise Lab Internship: Level 2: Mini-paint) и возможность создавать Pull request любыми лицами.

Суть приложения: галерея изображений пользователей со встроенным графическим редактором. Приложение состоит из 3 частей:

1. Авторизация / регистрация; Sign in и Register - это 2 разные страницы, соответственно разные роуты в приложении.

2. Просмотр ленты последних опубликованных изображений, реализация фильтрации по определенному пользователю.

3. Страница создания изображения на Canvas - мини-аналог встроенной программы Paint.

В качестве ориентировочного дизайна можно использовать пример мобильного приложения, приложенный внизу описания. Разумеется, это приложение надо адаптировать под веб-приложение (в произвольной форме, но я бы оставлял такую же структуру и увеличивал **max-width** страницы до **762 пикселей**).

При создании изображения пользователь должен иметь возможность выбирать кисть разной ширины, ее цвет, и рисовать в определенной области. В том числе должна быть возможность рисовать простейшие фигуры: линия, круг, прямоугольник. В качестве ориентира и примера по дизайну приложения можно взять онлайн-редактор изображений Vectr. В нем есть возможность использования без регистрации.

Все состояние приложения, сохранение и обновление данных, авторизацию надо реализовать через Firebase. Для выполнения задачи можно использовать личный Google-аккаунт.

Важно: все secret-значения (вроде данных о Firebase-приложении) не должны храниться в репозитории, их надо выносить в .env файл и добавлять в .gitignore репозитория.

Технические требования:

1. Требуется использование React-Redux / Vuex для React / Vue или NgRx для Angular для менеджмента данных.

2. Требуется использование TypeScript.

3. Для роутинга в приложении в React и Vue можно использовать сторонние библиотеки. В Angular - использовать встроенный Angular Router. Важно: если пользователь не авторизовался, его не должно пустить на страницу создания или просмотра задач.

4. Перед разработкой надо настроить ESLint для приложения. В качестве дополнения можно настроить Prettier и связать его с ESLint. *Будет дополнительным плюсом реализовать pre-commit hook, который не позволит сделать push в репозиторий, если в приложении присутствуют ошибки ESLint.*

5. Грамотная работа с базой данных. По возможности с клиента должно идти как можно меньше запросов. Банальный пример не самого оптимального использования: делать отдельный запрос на задачи для каждого дня. То есть для 30 дней месяца улетают 30 запросов. Это можно попробовать оптимизировать через Firebase Database Query.

6. Написание документации к проекту; в README-файле проекта перед сдачей задания должна быть написана краткая документация на английском или русском, состоящая из 4 пунктов: “Task” (ссылка на этот документ), “How to run the app” (инструкции по установке модулей и запуску приложения), “Database snapshot” (надо показать, как в базе данных Firebase организована работа с сущностями), и “Application stack” (описание технологического стека, использованного в приложении: какие дополнительные библиотеки и утилиты были использованы); *плюсом будет написание краткого описания структуры папок: для каждой папки описать, какого типа файлы она хранит.*

7. Обработка ошибок с сервера: если пользователь пытается авторизоваться с неправильными данными - Firebase вернет ошибку, и это надо вывести пользователю на экран в виде каких-либо Toast (для этого для скорости реализации можно использовать какую-нибудь стороннюю библиотеку).

Дополнительные баллы можно получить за:

1. Действительно удобный пользовательский интерфейс.

2. Реализованный Theme-менеджмент. Возможность с легкостью поменять цветовую схему приложения **из приложения** (надо реализовывать переключатели для пользователя).

3. Реализация рисования более сложных базовых фигур (звезда, многоугольник).