Take-Home Exercise

Introduction

Welcome ! We're excited to present you with this take-home exercise to assess your technical skills and your ability to integrate within our development team. In this exercise, you are tasked with developing a very simple frontend application that interacts with our mock backend services to manage user data efectively. The application will facilitate user authentication, registration, card listings and profle management, providing a hands-on way to showcase your abilities with React Native or React.js, your profciency in using Redux Saga for state management, and your approach to building scalable and efcient user interfaces.

Technical Preferences

- **Primary Technologies:** React Native (highly preferred), React.js
- **Language:** Typescript is highly encouraged
- **State Management:** Redux Saga highly preferred

- **For React Native, we prefer using Expo**
- **Ensure you utilize robust state management techniques.**

API Details

- Base URL: `http://143.198.168.244:3000/api/users`
- Typescript Interface for User:

```
interface User {
     _id: string
     firstName: string
     lastName: string
     email: string
     userName: string
     address: string
     profilePic(optional): string
     isBuyer: boolean
}
```

Screens and API Integration

1. Signup Screen

- **Endpoint:** `POST /register/v2`

- **Functionality:** Allows new users to register by providing necessary details. The input includes **frstName** and **lastName** (you can coalesce them into a fullName on the form as long as you send them to the backend separately), along with **email**, **userName**, **password**, **confrmPassword**, **address**, **proflePicture (optional simple uri string)**, and **isBuyer** status.

- **Description:** Ready, set, sign up! Before users can explore all the features your app has to ofer, they'll need to create an account. You're on deck to craft this essential frst step. During the registration process, users will provide the following information:

    1. **frstName**
    2. **lastName**
    3. **email**
    4. **userName**
    5. **password**
    6. **confrmPassword**
    7. **address**
    8. **isBuyer**
    9. **proflePic (optional simple uri string)**

    Here are a few special notes to keep in mind:

    a. You have the fexibility to design this as a single-page layout or distribute the content across multiple interconnected pages to avoid a congested layout. The choice is yours.
    b. You can can coalesce **frstName** and **lastName** into a fullName on the input form as long as you send them to the backend separately
    c. The **isBuyer** attribute is a boolean, by default set to false.
    d. The **address** should be formatted as 'City, Country' (e.g., 'Addis Ababa, Ethiopia'). For pinpoint accuracy, consider using tools like Google Places Autocomplete.
    e. **proflePicture:** Do not worry too much about this; just put in a random uri from google.

    BONUS POINT

Although not mandatory, adding a current location fetcher will not only boost user experience but will also enhance your submission.

2. Login Screen

- **Endpoint:** `POST /login`
- **Functionality:** Users can log in by providing their email and password.

- **Description:** Users must be able to log in using their email and password. Ensure that the email format is validated correctly and that the password input is handled via secure text entry to maintain confdentiality. Additionally, if authentication fails, it is crucial to relay the backend error message directly to the user to facilitate troubleshooting and enhance user experience.

3. Home Screen

- **Endpoint:** `GET /fetch/dummy/user-v2`
- **Functionality:** Retrieves and displays a list of 100 mock users. You are responsible for ensuring the list renders smoothly and efciently.

- **Description:** This API accepts two optional queries: **`page`** and **`limit`**. By default, **`page`** is set to 1, and **`limit`** is set to 10 values per page. Thus, when you call the endpoint without any query it displays the frst 10 users by default. We aim for the screen to have the capability to display every user as needed, either by scrolling or by clicking through pages in the UI. Consequently, you will need to update these queries when calling the API. Your task is to efectively call this API to render the users on the screen in a very smooth and optimized manner. We strongly encourage using a state management tool (such as Redux Saga) to track and update the fetched users.

  On the screen, each user will be displayed in a card featuring their information, such as frst name, last name, and userName. **Since the profle pictures of these random users are invalid, you may use a single random URI from Google to display an image in their respective cards. It is perfectly acceptable for all cards to use the same image URI.** If the user is a buyer (isBuyer is true), this should be indicated on the card. You could add a 'Buyer' label or use icons to denote that the user is a buyer; the choice of representation is yours. Although not a critical requirement, we strongly encourage you to develop an elegant design and layout for the screen.

4. Edit Profle Screen

- **Endpoint:** `PUT /profle?id=${userId}`
- **Functionality:** Enables logged-in users to view and edit their personal information such as frstName or email. When calling the endpoint, userId is required as a parameter and any property you want to change such as frstName, lastName or email is expected in the body.

- **Description:** Initially, this page will display the logged-in user's information, such as frstName, lastName, and email, which should be rendered directly from the state where it was stored at login; thus, no initial API call is required. Just like the cards in the HomeScreen, if the logged in user does not have a valid uri in their proflePic, simply put in a random uri from google as a placeholder. In this screen, users have the option to edit their information. To update user details, you will need to invoke the specifed API endpoint, sending the logged-in user's ID as the userId parameter and any changed properties (such as frstName, lastName, and email) in the request body. To ensure a smooth transition during state updates, we recommend using Redux Saga for efective state management. Also, while not mandatory, we encourage you to utilize good design and layout in your implementation.

Submission Guidelines

- **Deadline:** Thuesday, April 23, 2024, 11:59 PM East African Time.
- **Repository:** Submit your project to a GitHub repository named "Frontend Exercise" with a descriptive README. We also encourage you to write comments on your code to ease readability.

Contact Information

For any questions or queries, feel free to contact .

We look forward to seeing your solution and how you tackle the challenges of building an interactive and user-friendly application. Good luck!