LAB 8

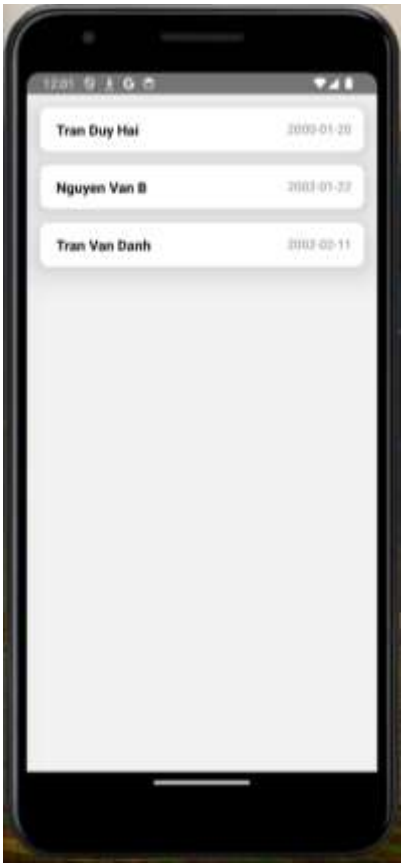
MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Thiết lập axios cho ứng dụng
- ✓ Hiểu rõ các phương thức tương tác với dữ liệu
- ✓ Đưa dữ liệu lên giao diện ứng dụng

NỘI DUNG

BÀI 1: HIỆN DANH SÁCH LẤY TỪ API



Yêu cầu:

- Tự động gọi api để lấy dữ liệu từ server khi vừa mở màn hình lên

Hướng dẫn:

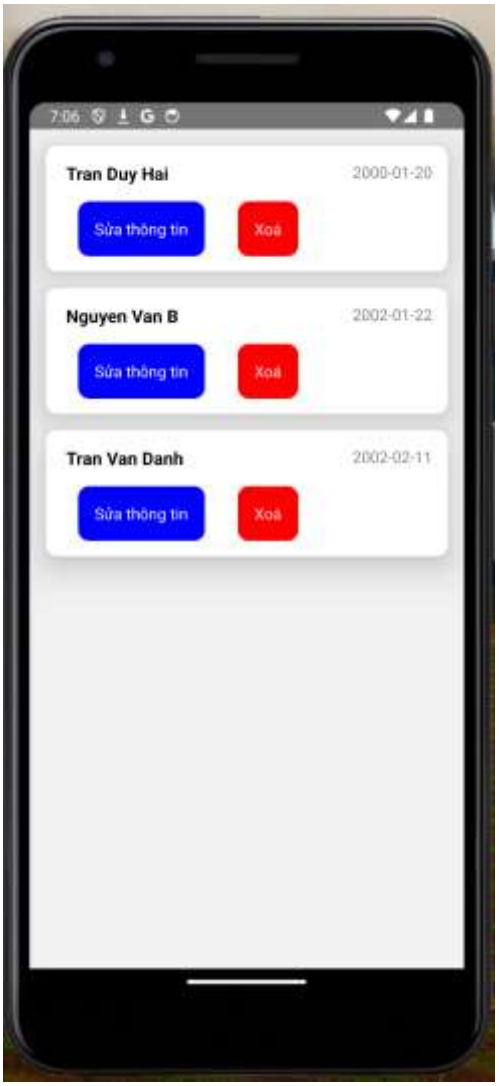
- Cách thiết lập axio cho ứng dụng đã được hướng dẫn ở Slide 8, nếu quên các bạn có thể mở lại để xem.
- Viết hàm gọi api để lấy danh sách user. **setUsers** là một biến state dùng để lưu lại danh sách.

```
const onGetUserList = async () => {
  showLoading();
  const response = await usersAPI.getUserList().finally(() =>
  hideLoading());

  if (Array.isArray(response)) {
    setUsers(response);
  }
};
```

- Để bắt được sự kiện người dùng đang ở màn hình danh sách, sử dụng hook **useFocusEffect**, để gọi api **onGetUserList.**

```
useFocusEffect(
  useCallback(() => {
    onGetUserList();
  }, []),
);
```

BÀI 2: XOÁ VÀ CHỈNH SỬA DỮ LIỆU VỚI AXIOS



Yêu cầu:

- Thêm 2 nút “Sửa thông tin” và “Xoá”.
- Khi người dùng nhấn vào nút “Sửa thông tin” di chuyển đến màn hình chỉnh sửa tài khoản.
- Khi người dùng nhất nút “Xoá” thực hiện xoá tài khoản ngay lập tức.
- Ở màn hình “Sửa thông tin” khi sửa thành công, tự động back lại màn hình danh sách và thông báo cho người dùng đã cập nhật thành công.

Hướng dẫn:

- Khi người dùng nhấn nút delete gọi hàm **onDeleteUser.**

```
const onDeleteUser = async (id: number) => {
  await usersAPI.deleteUser(id).then(onGetUserList);
};
```

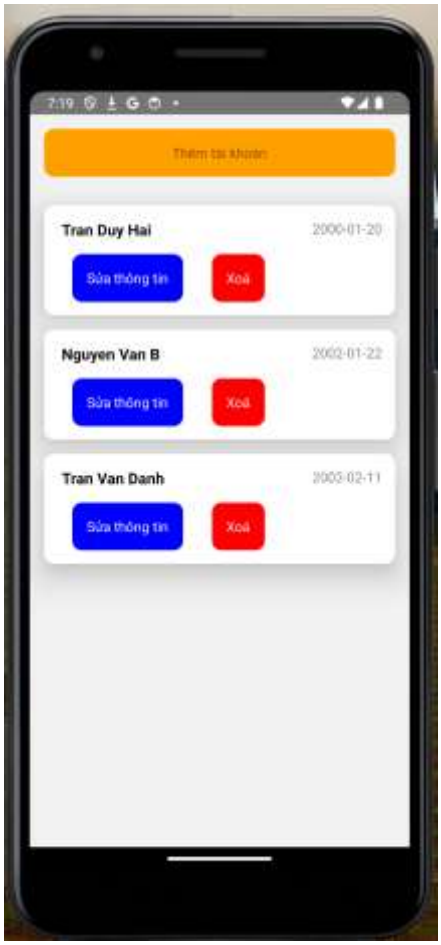
- Ở màn hình **EditUser** lấy các route nhận từ màn hình danh sách user, sau đó đổ dữ liệu lên các input.

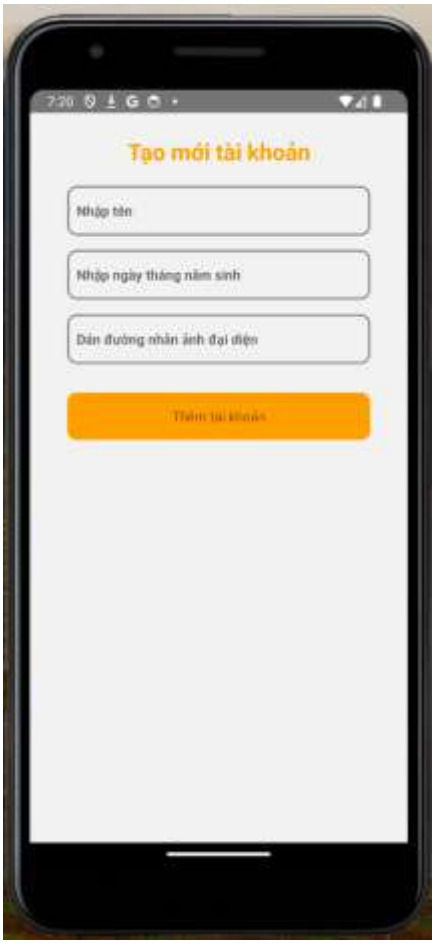
```
export const EditUser: FC<
  NativeStackScreenProps<RootStackParamList, 'EditUser'>
> = ({route}) => {
  const {id, name = '', birthday = '', avatar = ''} = route?.
  params || {};

  const [user, setUser] = useState<UserType>({name, birthday,
  avatar});
```

- Gọi hàm **updateUser** để cập nhật dữ liệu vào DB.

```
const onSubmit = async () => {
  if (id) {
    await usersAPI.updateUser(id, user).then(onSuccess);
  }
};
```

BÀI 3: THÊM TÀI KHOẢN MỚI, MÀN HÌNH CHI TIẾT TÀI KHOẢN.





Thông tin chi tiết tài khoản

Yêu cầu:

- Thêm nút "Tạo tài khoản"
- Nhấn nút "Tạo tài khoản" chuyển tới màn hình "Thêm tài khoản".

- Ở màn hình "Tạo tài khoản" khi tạothành công, tự động back lại màn hình danh sách và thông báo cho người dùng đã tạo tài khoản thành công. Nếu lỗi, thông báo đến người dùng.

Hướng dẫn:

- Bạn có thể dùng chung giao diện của màn hình chỉnh sửa tài khoản từ bài 2 để làm chức năng tạo mới tài khoản.

- Gọi hàm **onGetUserDetail** để lấy thông tin chi tiết của tài khoản từ id. Nếu lấy thành công set dữ liệu vào, nếu lỗi thông báo cho người dùng biết

```
const onSuccess = (item: UserType) => {
  setUser(item);
};

const onGetUserDetail = async () => {
  const response = await usersAPI.getUserDetail(id);

  if ('isError' in response) {
    onError();
  } else {
    onSuccess(response);
  }
};
```

BÀI 4: GV CHO THÊM

*** YÊU CẦU NỘP BÀI:

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.

--- Hết