VIETNAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

INFORMATION TECHNOLOGY

INDUSTRIAL INTERNSHIP

Subject: Researching online social platform for VSPORTS, developing the platform for management sports leagues, tracking progress.

Student name: Phan An Duy

Student ID: 518H0616

Class:18H50301

School year: 2024-2025

HO CHI MINH CITY, 2024

VIETNAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

INFORMATION TECHNOLOGY

INDUSTRIAL INTERNSHIP

Subject: Researching online social platform for VSPORTS, developing the platform for management sports leagues, tracking progress.

Student name:	**Phan An Duy**
Student ID:	**518H0616**
Class:	**18H50301**
School year:	**2024-2025**

HO CHI MINH CITY, 2024

OPENING INTRODUCTION

For this report be done properly, massive thank you to my direct supervisor of **SUNNY VIETNAM DEVELOPMENT TECHNOLOGY CO.LTD,** Mr. Vo Tran Dong Thoai, for giving me a chance to work on such a demanding environment, to know about the work ethics, the skill needed, and with a demanding project to work on, a chance to developing such a skillful developer for such position as Mobile Developers – React Native. Such as building a platform, with specific functionality, features for the Vsports Social Platform, learning how to manage a source code with teams using Git source control management and GitLab platform as a way to store the code base and the work from my working hours.

Learning in this environment helps me with a better look at the working space. Hence everyone is working on sub-modules. It is easy to understand the fragmentation of the code should be, but with a great tech manager like Pham Le Phong, who helps me on the responsibilities of managing the code and using the CI/CD management skill to simplify code push and pull.

And with specials thanks to my instructor, Mr. Duong Huu Phuc as my direct instructor, for letting me fulfill my duties and finishing my course and be ready for graduation, This report is been through a lot of process, from researching to the development of the project, consulting with my co-worker, with Le Trung Hieu, Pham Le Phong, Nguyen Hien Vinh and Nguyen Minh Khoi as my developer co-workers and UX/UI designer respectively, and finally to finish of this report for the project. With lots of ideation, and changes before this work is completed.

TABLE OF CONTENTS

I. COMPANY HISTORY & INTRODUCTION

1. Overview of the company

Vsports – Sports Community is an organization focused on promoting physical activity and sports culture across Vietnam. Established with the mission of connecting individuals passionate about sports, Vsports operates a platform for athletes, fans, and fitness enthusiasts to interact, share experiences, and participate in a variety of sports events.

a. History and Foundation

Vsports was founded to create a comprehensive community where people can engage in competitive sports, learn from one another, and improve their athletic abilities. The organization aims to provide an inclusive environment for sports lovers of all levels, from beginners to professionals. Over the years, Vsports has gained recognition for organizing various high-profile sports events and becoming a hub for sporting talent and fans alike.

b. Field of Work

Vsports is primarily focused on facilitating sports events and competitions across multiple disciplines such as football, basketball, esports, and more. The company also works to promote sports through educational initiatives, workshops, and social media engagement. They provide platforms where athletes can showcase their skills, participate in friendly tournaments, and build a network within the sports community.

c. Key Events and Activities

One of Vsports' most notable and active events is the Vsports Esports Tournament, where top players from across the country compete in various online gaming competitions. Additionally, Vsports hosts sporting events like football tournaments, fitness challenges, and basketball leagues. These events serve to not only promote the importance of physical health but also foster a sense of community among participants.

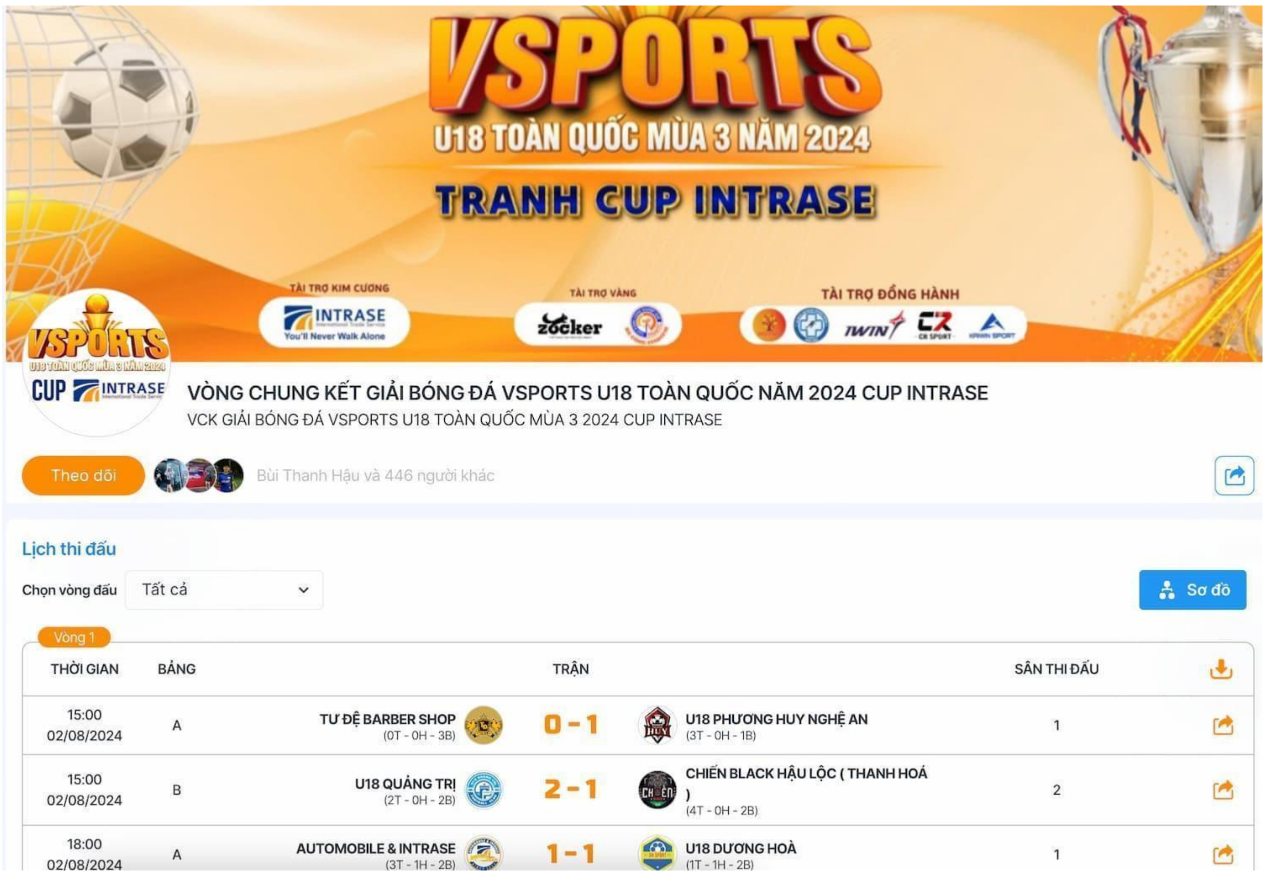
Through its strategic partnerships with local and international sports organizations, Vsports continues to grow its presence, offering an extensive range of sports activities and establishing itself as a leading name in the Vietnamese sports community.

The product of the company mainly working on is Vsports – a community ecosystem towards on creating a playfield for all the players around the country, with the objective focuses on football, basketball, Vsports covers from small sport event to major, by providing a platform for all the sports managers, to update, create and monitoring all the activities of the match. Platform works on websites as well as mobile, written in React Native, so the software is guaranteed to be compatible with all the devices. By providing a platform for monitoring the sports league and sports teams, we’ll be able to help the manager sort and create a tracking system, without having to worry about losing the information. All the information is stored digitally, converts all the hassling paperwork, reduces the paperwork clutter and easily manageable.

2. **Company History & Foundation**

Started during COVID-19, our founder sought on a solution to solve the issue of the global pandemic, when every social activity, economic situation are suspended, that's why the ideas of Vsports was born, the ideas of solution to digitize sports and all the leagues followed occurred when the pandemics were faded. In this time, the foundation of Vsports is in the middle of development, in technology, in the formation of the company and build a strong foundation.

In May of 2021, about a year in development, Vsports finally take shape. Ready to apply in small league to support, elevate the efficiency of the management, in order for the league, the coordinator and the teams to work effectively. After a period of development, the VSports platform has now reached the community with an interaction volume of 100,000 users



VÒNG CHUNG KẾT GIẢI BÓNG ĐÁ VSPORTS U18 TOÀN QUỐC NĂM 2024 CUP INTRASE

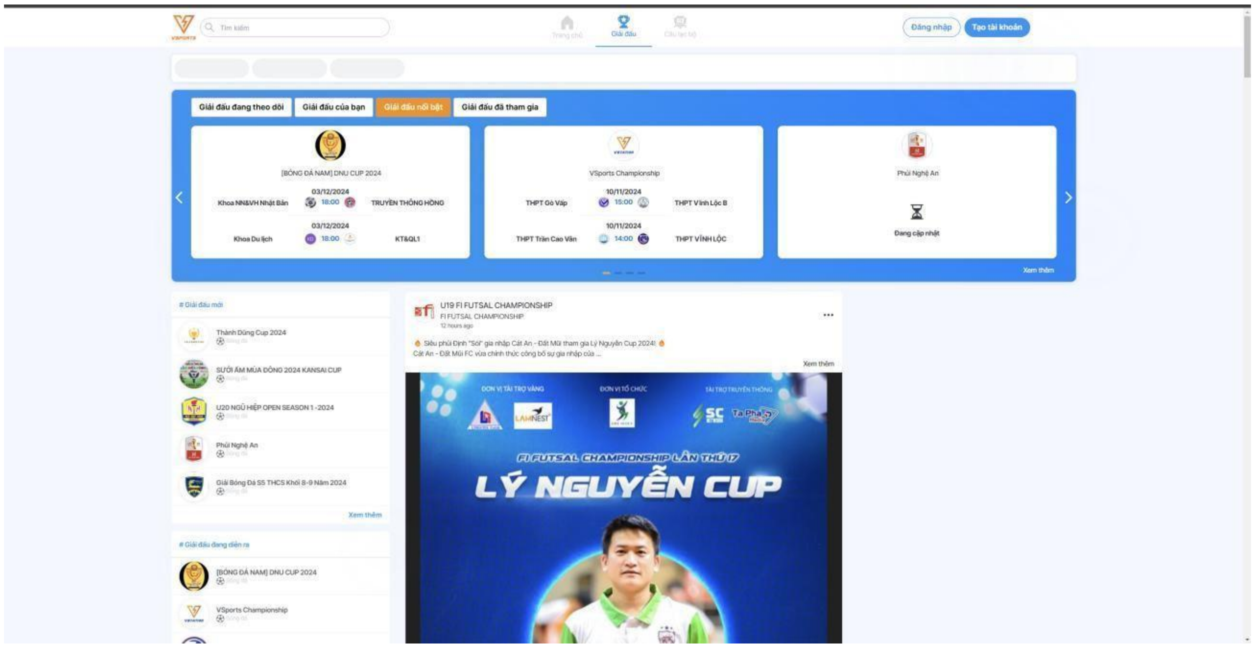
VCK GIẢI BÓNG ĐÁ VSPORTS U18 TOÀN QUỐC MÙA 3 2024 CUP INTRASE

Lịch thi đấu

Chọn vòng đấu: Tất cả

Vòng 1

THỜI GIAN	BẢNG	TRẬN			SÂN THI ĐẤU
15:00 02/08/2024	A	TỨ ĐỆ BARBER SHOP (0T - 0H - 3B)	0 - 1	U18 PHƯƠNG HUY NGHỆ AN (3T - 0H - 1B)	1
15:00 02/08/2024	B	U18 QUẢNG TRỊ (2T - 0H - 2B)	2 - 1	CHIẾN BLACK HẬU LỘC (THANH HOÁ) (4T - 0H - 2B)	2
18:00 02/08/2024	A	AUTOMOBILE & INTRASE (3T - 1H - 2B)	1 - 1	U18 DƯƠNG HOÀ (1T - 1H - 2B)	1



[BÓNG ĐÁ NAM] DNU CUP 2024

Khoa NN&VH Nhật Bản	03/12/2024 18:00	TRUYỀN THÔNG HỒNG
Khoa Du lịch	03/12/2024 18:00	KT&QL1

VSports Championship

THPT Gò Vấp	10/11/2024 15:00	THPT Vĩnh Lộc B
THPT Trần Cao Vân	10/11/2024 14:00	THPT VĨNH LỘC

Phủi Nghệ An

Đang cập nhật

Giải đấu mới

- Thành Dũng Cup 2024
- SƯỞI ẤM MÙA ĐÔNG 2024 KANSAI CUP
- U20 NGŨ HIỆP OPEN SEASON 1 -2024
- Phủi Nghệ An
- Giải Bóng Đá S5 THCS Khối 8-9 Năm 2024

Xem thêm

Giải đấu đang diễn ra

- [BÓNG ĐÁ NAM] DNU CUP 2024
- VSports Championship

U19 FI FUTSAL CHAMPIONSHIP
FI FUTSAL CHAMPIONSHIP
12 hours ago

Siêu phủi Định "Sói" gia nhập Cát An - Đất Mũi tham gia Lý Nguyễn Cup 2024!
Cát An - Đất Mũi FC vừa chính thức công bố sự gia nhập của ...

Xem thêm

Image: Vsports platform on website

Recognizing the increasing influence and demand from the community, the Vsports ecosystem was established on April 27, 2022, aiming to diversify various aspects within the Cultural and Sports domain.

Image: A sports team attending the Yen Bai Province cup.

As time has passed and Vsports has expanded its reach and diversified its fields within the ecosystem, it is now in a phase of development. It has been collaborating with numerous entities and organizations within the community to open new chapters for the sports narrative, both within and outside the country, further strengthening its presence.

Image: All teams attending the Vsports – Café De Mang Den Cup.

From this information, we can sure that the Vsports are actively working, and with such an amazing community, the Vsports will continue to thrive, and the community will only get bigger and more attractive to most young members of the community.

END OF THE INTRODUCTION

II. OVERVIEW OF THE PROJECT

1. Overview of the work

My project included a sub module of the Vsports products, such as creating an Authentication for the Sign in – Sign out, Registration, league registration, team registration. Working with the API to collect the data, read, update and create the data, upload to the server of Vsports. Working with the API is such a challenge, but with such an enthusiastic team, I can work effectively and coherently and break through all the work. So, to tell the comprehensive about the project. I'll summarize what is the work I do in the time I work at Vsports and briefly put it in the Document

I am currently contributing to the development of several key sub-modules in the Vsports platform, focusing on enhancing its functionality and improving the overall user experience. My primary responsibilities include working on the error checking, and the creation of a user interface (UI) for **Teams Management** and **League Management.** When building the components, I'm working with some specific requirement tailored to the experience of the mobile application. As of working, the project is meant to be a platform for the global and all team creators are looking for a vote for all the teams to advance in the league. This will require working on some specific UI elements that are needed . In the meantime, while building an application, I also built a voting system with the help of my co-workers. all built with **React Native**.

As for now, I am working on the Front-end side, whilst my co-workers are working on the back-end, using NodeJS, with MongoDB as the database. Because of the versatility of the project stack, the application is coherently compatible with other platforms.

Learning this will make me have to learn about Android Lifecycle as well, just because the framework works, doesn't mean that every will be safe to put the code, learning the lifecycle of the Android will do as well

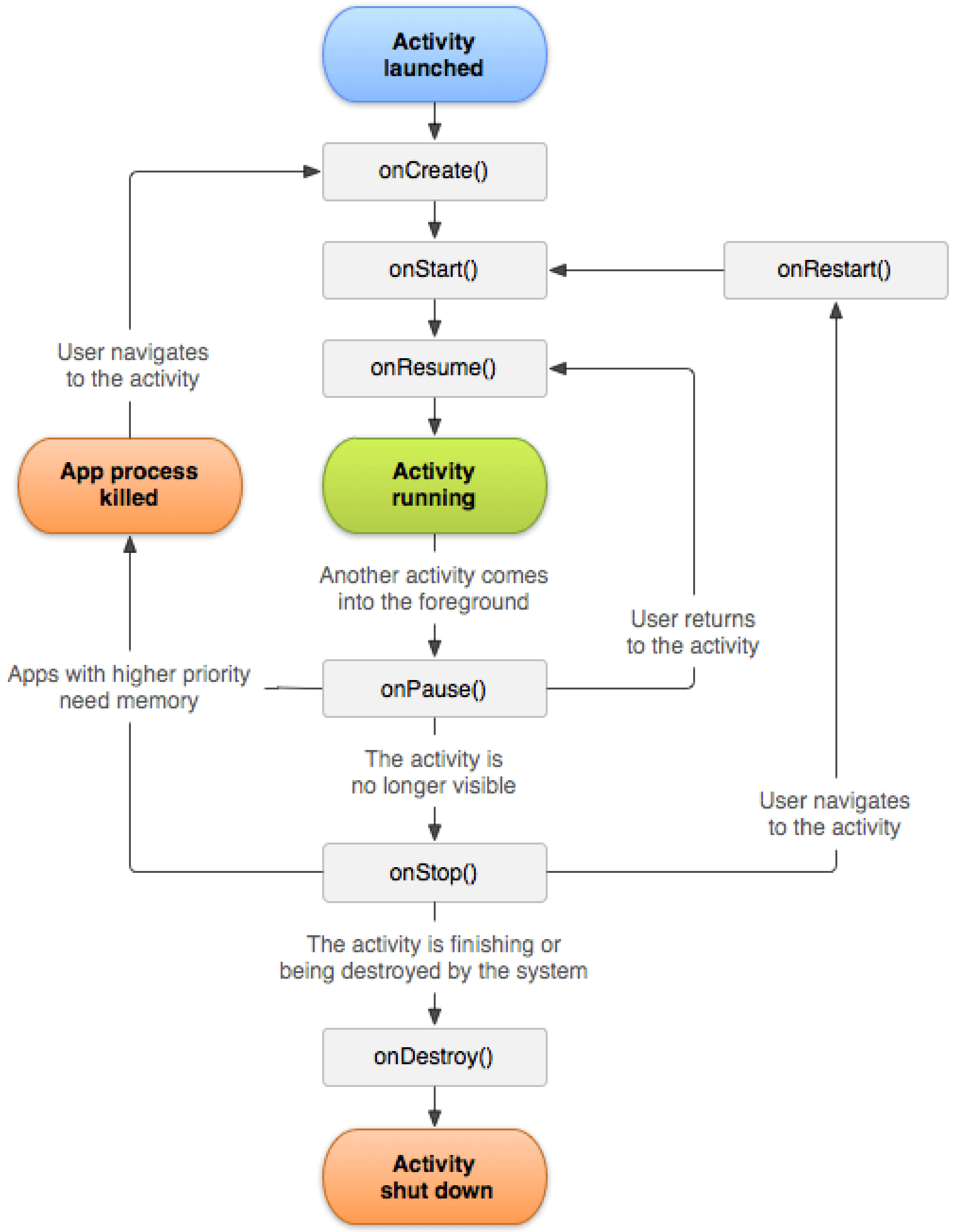

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Image: Android activity cycle

This will be the main core function about learning how to application work. On iOS we have a term Delegate, with the status of each session are `ot running, In active, Active, Background, Suspended` The iOS operating system manages the application states, but the app is responsible for handling user-experience through these state transitions.

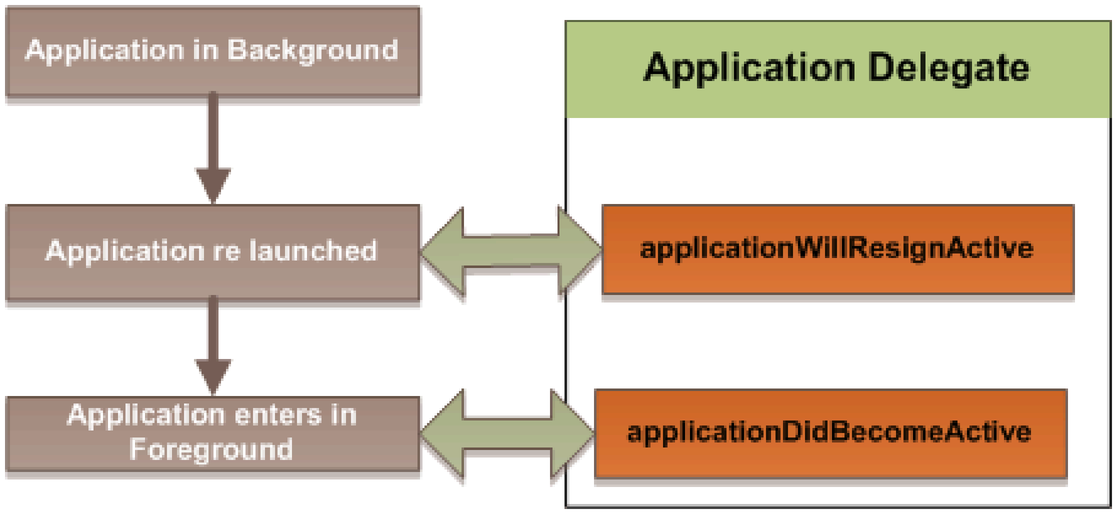
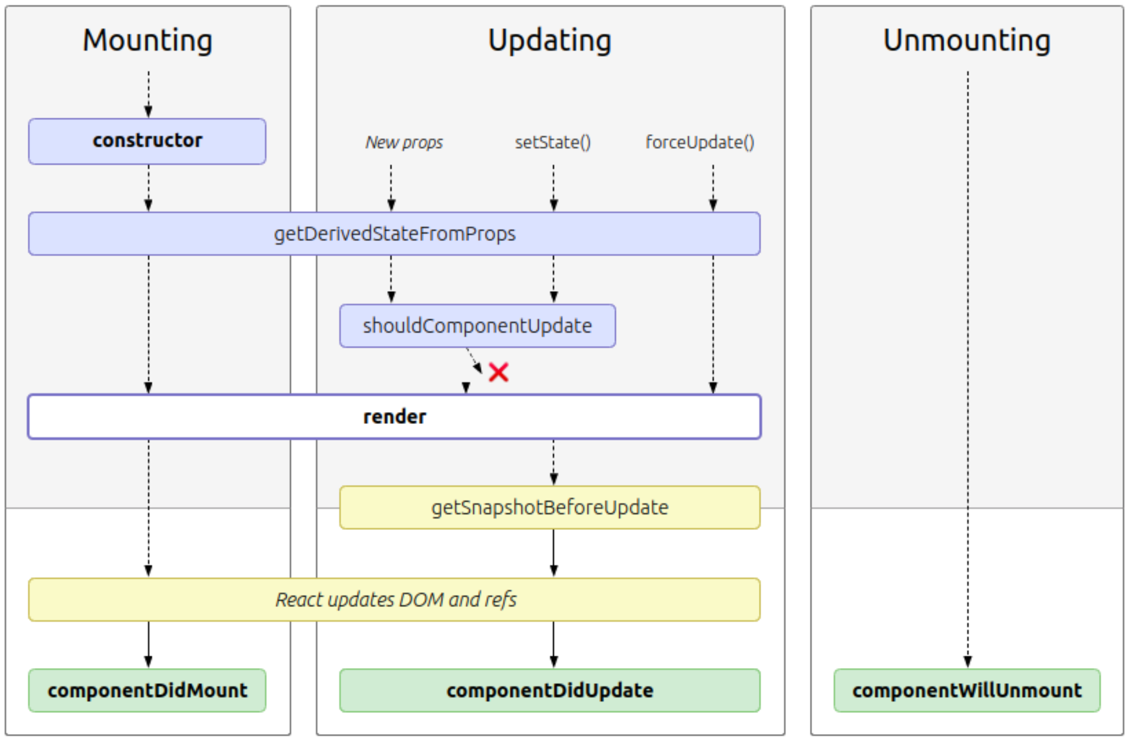

Application in Background
Application re launched
Application enters in Foreground
Application Delegate
applicationWillResignActive
applicationDidBecomeActive

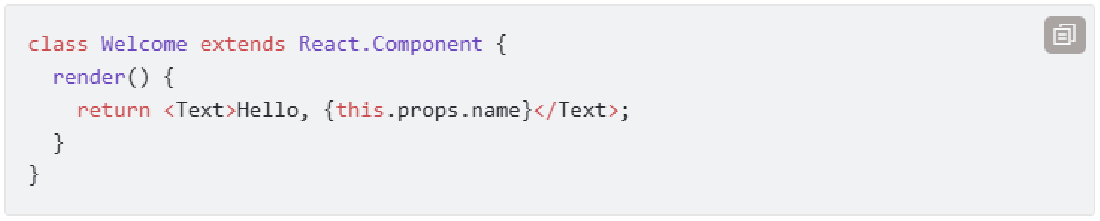
Image: How the iOS app lifecycle work

Before learning anything about the project, I need to learn how to work with teams, using React Native, using the file system control management, using platform such as Git and GitLab. Learning these will give the opportunity to learn about the life cycle of React Native, learn when the components will mount, did mount and how to unmount the components.



This is the main idea of how the components should work in every React Native cycle. When the user did the component mounting to the project, the component will render it when it first mounting. After every time a value change, if we passed a value dependencies, it will change the components each time the value change.

To declare class components, we can work as such as this:



```
class Welcome extends React.Component {
  render() {
    return <Text>Hello, {this.props.name}</Text>;
  }
}
```

React Native work best when passing a props from one to another, by create this instance, we can determine how the application works

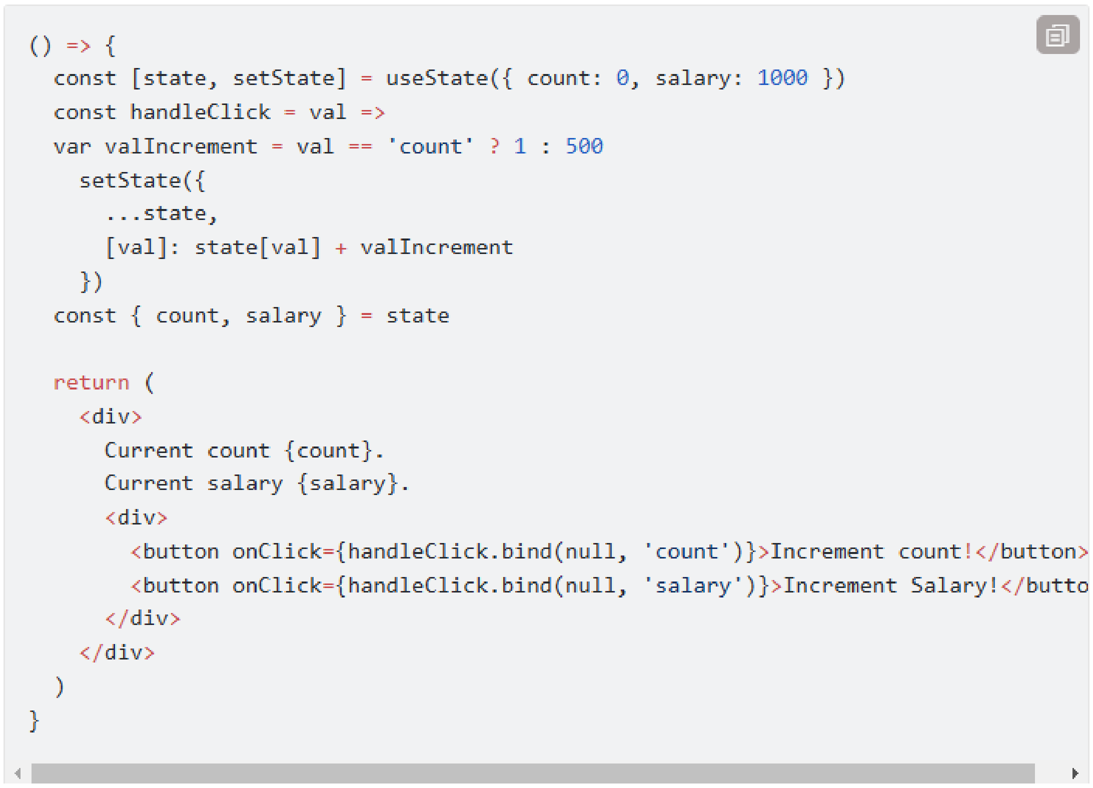
```
constructor(props) {
  super(props);
  // Don't call this.setState() here!
  this.state = { counter: 0 };
  this.handleClick = this.handleClick.bind(this);
}
```

This constructor will be the main core function to let the function handle and communicate data to each other.

Also, to worth mentioning about the React Native Hook, a concept to handle function with easy solution, handling data and error catching solution. Hook basically a bridge to communicate from the SDK, API, without ever to rewrite every single time.

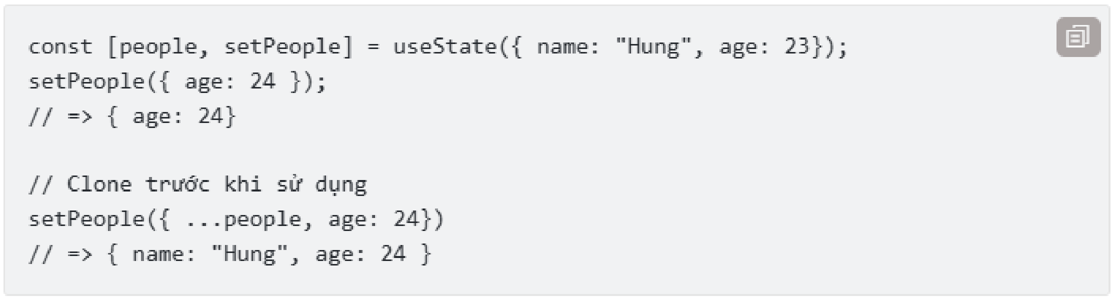
```
import React, { useState, useEffect } from 'react'
import { View, Text, Button } from 'react-native'

export const MyFunctionalComponent = () => {
 const [counter, setCounter] = useState(0)
useEffect(() => {
 console.log('Counter has changed!')
 return () => setCounter(0)
}, [counter])
const handleIncreaseCounterPress = () => {
 setCounter(counter + 1)
}
return (
 <View>
  <Text>Counter is {counter}.</Text>
  <Button onPress={handleIncreaseCounterPress} title='Increase       Counter'/>
 </View>
 )
}
```

This is the example of Hook in the shell of useState(). The useState is a very interesting subject, due to the how to the data handle through out the applications.



```
() => {
  const [state, setState] = useState({ count: 0, salary: 1000 })
  const handleClick = val =>
  var valIncrement = val == 'count' ? 1 : 500
    setState({
      ...state,
      [val]: state[val] + valIncrement
    })
  const { count, salary } = state

  return (
    <div>
      Current count {count}.
      Current salary {salary}.
      <div>
        <button onClick={handleClick.bind(null, 'count')}>Increment count!</button>
        <button onClick={handleClick.bind(null, 'salary')}>Increment Salary!</butto
      </div>
    </div>
  )
}
```

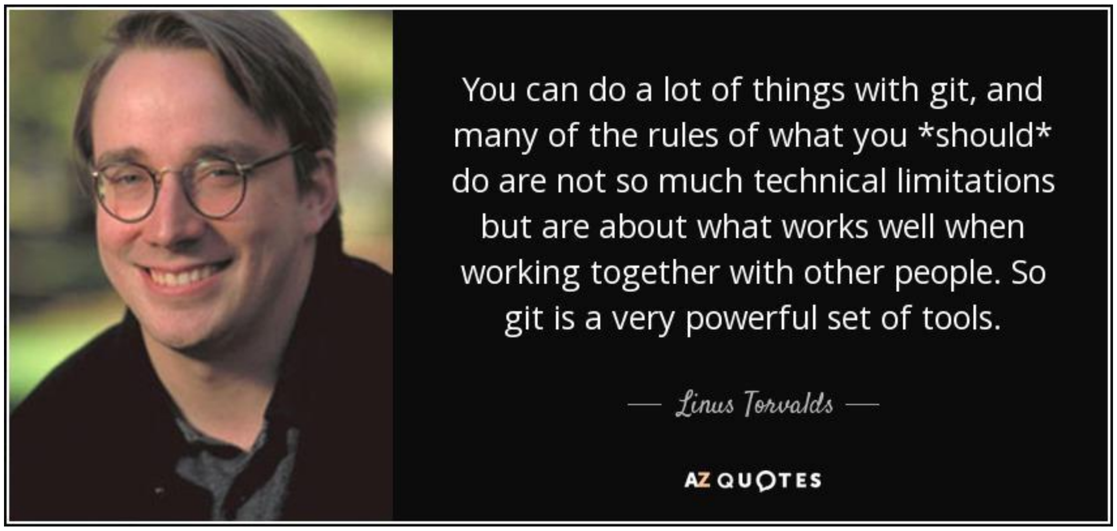
This is an example of how useState work on ReactJS



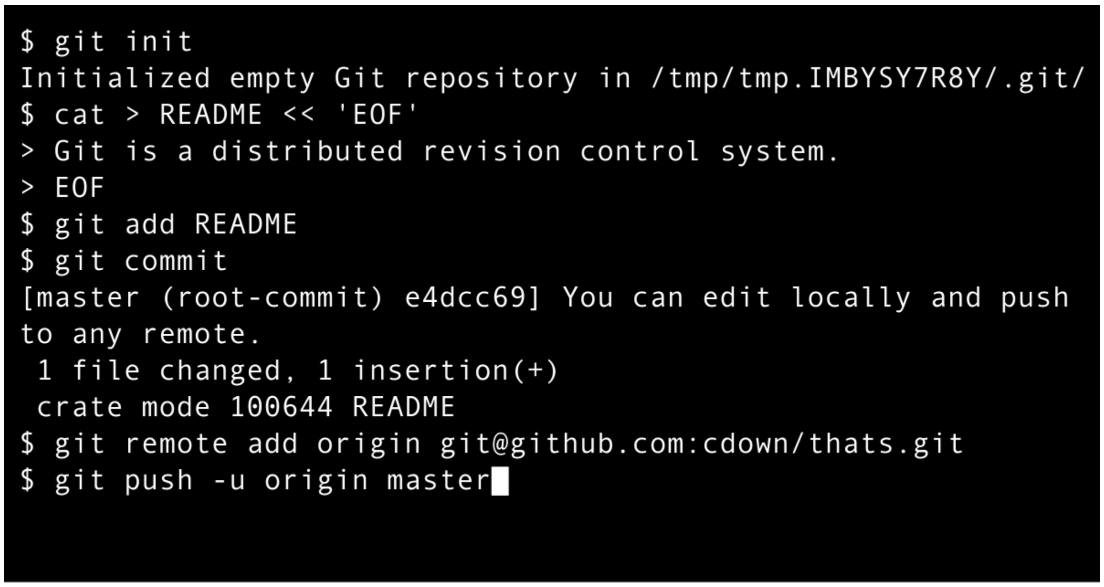
```
const [people, setPeople] = useState({ name: "Hung", age: 23});
setPeople({ age: 24 });
// => { age: 24}

// Clone trước khi sử dụng
setPeople({ ...people, age: 24})
// => { name: "Hung", age: 24 }
```

This is how the useState work in React Native

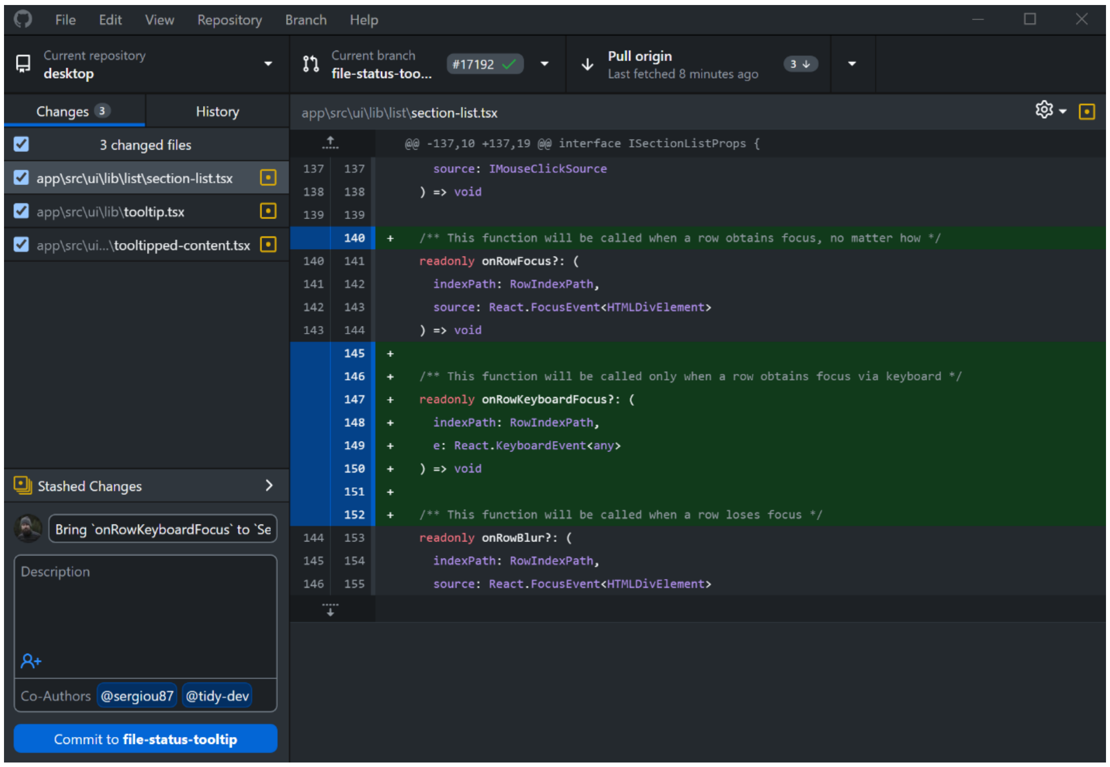
After working on how to do in React Native, the next thing I have to tackle about the work ethics is to create a workflow with Git and GitLab. Git basically a tools to change, add the code changes to the repository, get new update, update our updates to the code. A tool to manage the code so that every one in the team can see the changes. Git original author are Linus Torvalds, the father of Linux kernel.


You can do a lot of things with git, and many of the rules of what you *should* do are not so much technical limitations but are about what works well when working together with other people. So git is a very powerful set of tools.
— Linus Torvalds —
AZ QUOTES

Git is his own personal project, not meant to be a full set of powers tools to work for the business industry. A personal hobby is to simply managing the code, but the other ways seem to output as a major project.



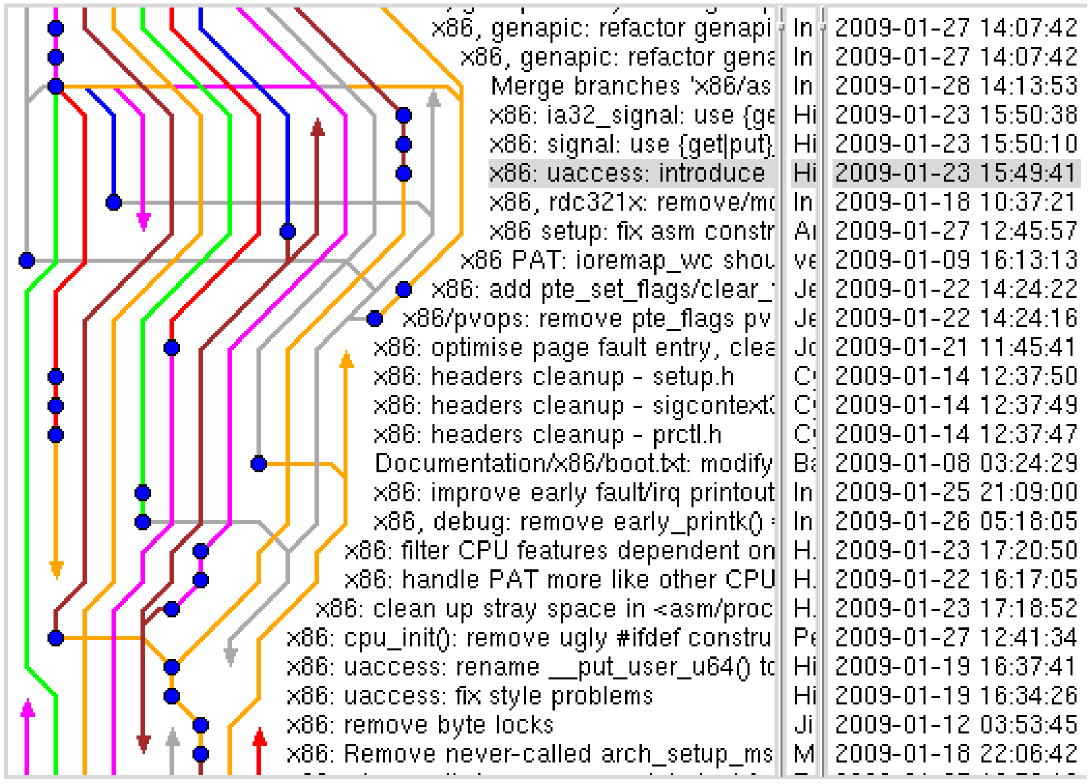
```
$ git init
Initialized empty Git repository in /tmp/tmp.IMBYSY7R8Y/.git/
$ cat > README << 'EOF'
> Git is a distributed revision control system.
> EOF
$ git add README
$ git commit
[master (root-commit) e4dcc69] You can edit locally and push
to any remote.
 1 file changed, 1 insertion(+)
 crate mode 100644 README
$ git remote add origin git@github.com:cdown/thats.git
$ git push -u origin master
```


File
Edit
View
Repository
Branch
Help
Current repository
desktop
Current branch
file-status-too...
#17192
Pull origin
Last fetched 8 minutes ago
3
Changes 3
History
3 changed files
app\src\ui\lib\list\section-list.tsx
app\src\ui\lib\tooltip.tsx
app\src\ui...\tooltipped-content.tsx
Stashed Changes
Bring `onRowKeyboardFocus` to `Se
Description
Co-Authors @sergiou87 @tidy-dev
Commit to file-status-tooltip

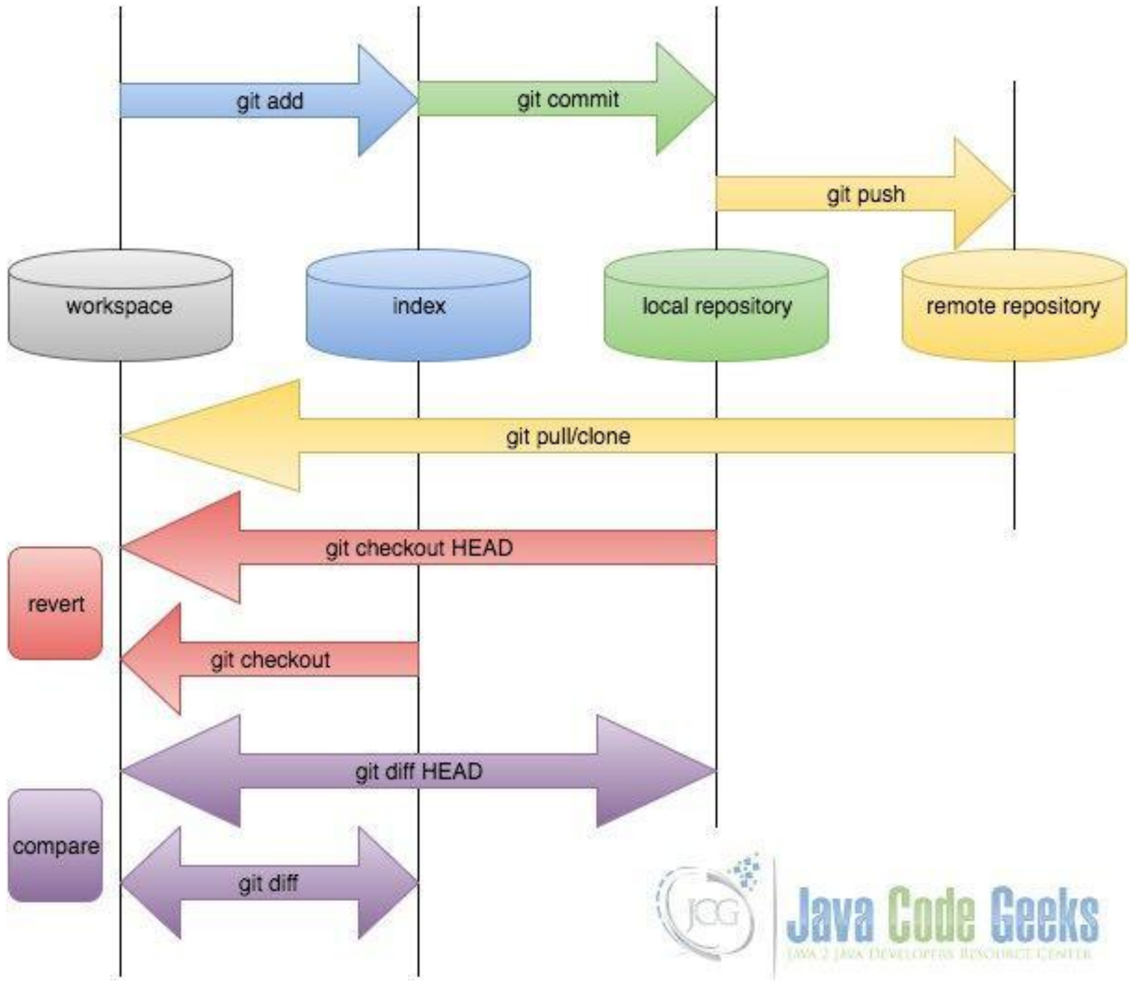
app\src\ui\lib\list\section-list.tsx

```
@@ -137,10 +137,19 @@ interface ISectionListProps {
      source: IMouseClickSource
    ) => void

+   /** This function will be called when a row obtains focus, no matter how */
    readonly onRowFocus?: (
      indexPath: RowIndexPath,
      source: React.FocusEvent<HTMLDivElement>
    ) => void
+
+   /** This function will be called only when a row obtains focus via keyboard */
+   readonly onRowKeyboardFocus?: (
+     indexPath: RowIndexPath,
+     e: React.KeyboardEvent<any>
+   ) => void
+
+   /** This function will be called when a row loses focus */
    readonly onRowBlur?: (
      indexPath: RowIndexPath,
      source: React.FocusEvent<HTMLDivElement>
```

Git mostly be done in the terminal or in the desktop application. But the work can be confused because of the lack of visualization.

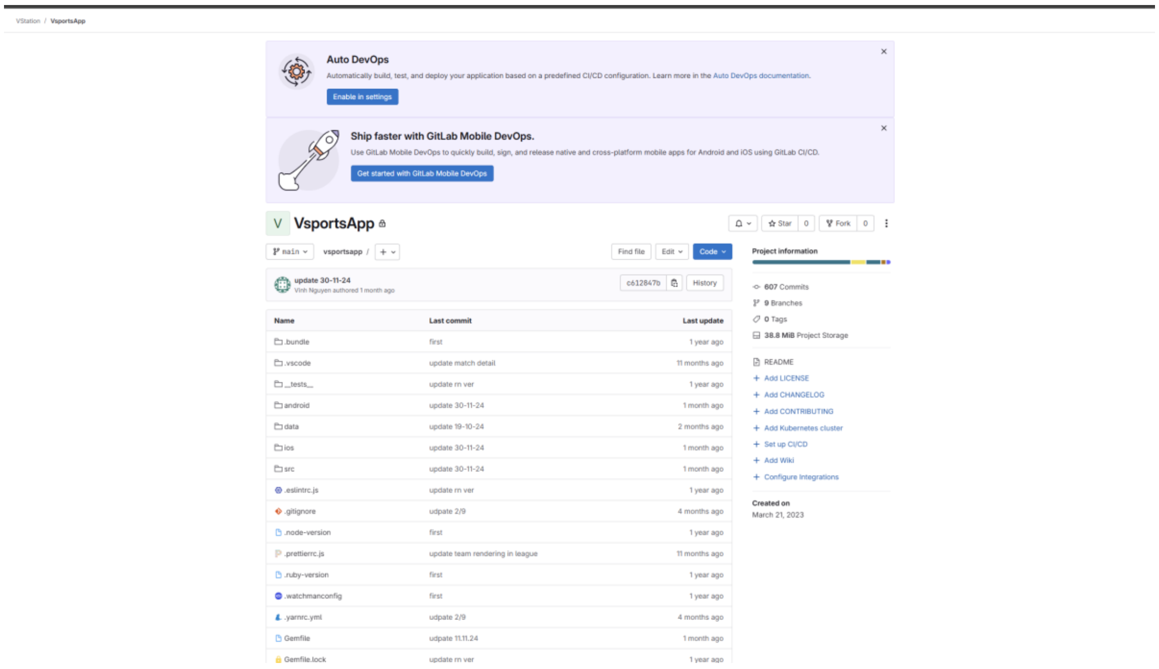


```
x86, genapic: refactor genapi   In   2009-01-27 14:07:42
x86, genapic: refactor gena     In   2009-01-27 14:07:42
Merge branches 'x86/as          In   2009-01-28 14:13:53
x86: ia32_signal: use {ge       Hi   2009-01-23 15:50:38
x86: signal: use {get|put}      Hi   2009-01-23 15:50:10
x86: uaccess: introduce         Hi   2009-01-23 15:49:41
x86, rdc321x: remove/mc         In   2009-01-18 10:37:21
x86 setup: fix asm constr       Ar   2009-01-27 12:45:57
x86 PAT: ioremap_wc shou        ve   2009-01-09 16:13:13
x86: add pte_set_flags/clear_   Je   2009-01-22 14:24:22
x86/pvops: remove pte_flags pv  Je   2009-01-22 14:24:16
x86: optimise page fault entry, clea  Jo  2009-01-21 11:45:41
x86: headers cleanup - setup.h  C    2009-01-14 12:37:50
x86: headers cleanup - sigcontext3  C  2009-01-14 12:37:49
x86: headers cleanup - prctl.h  C    2009-01-14 12:37:47
Documentation/x86/boot.txt: modify  Ba  2009-01-08 03:24:29
x86: improve early fault/irq printout  In  2009-01-25 21:09:00
x86, debug: remove early_printk()  In  2009-01-26 05:18:05
x86: filter CPU features dependent on  H.  2009-01-23 17:20:50
x86: handle PAT more like other CPU  H.  2009-01-22 16:17:05
x86: clean up stray space in <asm/proc  H.  2009-01-23 17:18:52
x86: cpu_init(): remove ugly #ifdef constru  Pe  2009-01-27 12:41:34
x86: uaccess: rename __put_user_u64() to  Hi  2009-01-19 16:37:41
x86: uaccess: fix style problems  Hi   2009-01-19 16:34:26
x86: remove byte locks          Ji   2009-01-12 03:53:45
x86: Remove never-called arch_setup_ms  M  2009-01-18 22:06:42
```



These are the visualization of Git Repository, and how to enhance the workflow of the project. By enhancing the work of managing source code to the repository upload to the database, there's no denying that Git has change the way I work with Vsports

GitLab is a place to store the code, a platform to enhance the power of Git to the platform. In the case where I need to store the code, I want to share the code to one another, or simply by changing the computer to another, the code will stay there no matter what, this will simply the code merging and conversion to another device


VStation / VsportsApp

Auto DevOps

Automatically build, test, and deploy your application based on a predefined CI/CD configuration. Learn more in the Auto DevOps documentation.

Ship faster with GitLab Mobile DevOps.

Use GitLab Mobile DevOps to quickly build, sign, and release native and cross-platform mobile apps for Android and iOS using GitLab CI/CD.

update 30-11-24
Vinh Nguyen authored 1 month ago
c612847b

Name	Last commit	Last update
.bundle	first	1 year ago
.vscode	update match detail	11 months ago
__tests__	update rn ver	1 year ago
android	update 30-11-24	1 month ago
data	update 19-10-24	2 months ago
ios	update 30-11-24	1 month ago
src	update 30-11-24	1 month ago
.eslintrc.js	update rn ver	1 year ago
.gitignore	udpate 2/9	4 months ago
.node-version	first	1 year ago
.prettierrc.js	update team rendering in league	11 months ago
.ruby-version	first	1 year ago
.watchmanconfig	first	1 year ago
.yarnrc.yml	udpate 2/9	4 months ago
Gemfile	udpate 11.11.24	1 month ago
Gemfile.lock	update rn ver	1 year ago

Project information

607 Commits
9 Branches
0 Tags
38.8 MiB Project Storage

README
Add LICENSE
Add CHANGELOG
Add CONTRIBUTING
Add Kubernetes cluster
Set up CI/CD
Add Wiki
Configure Integrations

Created on
March 21, 2023

This is GitLab repository for my Vsports Applcations, the very UI is similar to the project on GitHub.

These are the mandatory of what I work during my time in Vsports. Next I want to discuss about the details on the projects

Details on what I do on this project

a. *Building a team management module for Sports*

Building the team management is one of the main aspects of Vsports is the fact of using the power of mobile to implement the management system. Is a key feature designed to enhance the functionality and user experience of Vsports, a digital platform dedicated to managing and organizing sports activities. This module is tailored to streamline the organization, collaboration, and performance tracking of sports teams. Each team will have an ID, that ID will be the main aspect to registration the team for league competition, award winning system, voting, CRUD the teams.

Core Objectives

1. ***Team Organization****: Allow team owners, coaches, and captains to*

create, customize, and manage team profiles, including details like team names, logos, player rosters, and roles.

2. ***Scheduling and Coordination**: Enable the creation of match schedules, practice sessions, and other team events, with notifications and reminders for team members.*
3. ***Performance Tracking**: Provide tools for tracking team and individual player statistics, such as scores, attendance, and performance metrics over time.*
4. ***Communication**: Facilitate seamless communication within teams through integrated chat, announcements, and notifications.*
5. ***Integration with Vsports Ecosystem**: Ensure compatibility with other modules in Vsports, such as event registration, league management, and score tracking.*

b. *Building a league management module for Sports*

The League Management Module is designed to make running and participating in sports leagues as seamless, flexible, and engaging as possible. Whether you're an experienced league organizer or someone starting a community tournament for the first time, this module gives you the tools to create, manage, and adapt leagues effortlessly.

Why It Matters

Running a league is more than just schedules and scores—it's about building connections, fostering competition, and creating unforgettable moments for teams and fans. The League Management Module puts people at the center of the experience by making league organization intuitive and enjoyable for everyone involved. Whether it’s a youth soccer league, a recreational basketball tournament, or a corporate cricket competition, the League Management Module is built with the flexibility to handle diverse needs.

- ***For Organizers****: Take the stress out of juggling schedules, stats, and communication. Focus on making the league fun and memorable.*
- ***For Teams and Players****: Enjoy a transparent and fair system that prioritizes engagement and growth.*
- ***For Fans and Communities****: Stay connected to the action with live updates and easy access to standings.*

c. Building a voting system:

A voting system is a place to gather all the teams, raising a vote to see which team is capable of going to the next round or capable of going to the league. If the voting system is close, then it will decide and notify the teams in advance. The team will notice it, with the help of the social network as well, publishing on Facebook, Instagram, or even on the website with link. This system is well integrated, with the API as well also, the development team put so much effort on doing it, making a seamlessly with the website. To make sure the application can work on any device.

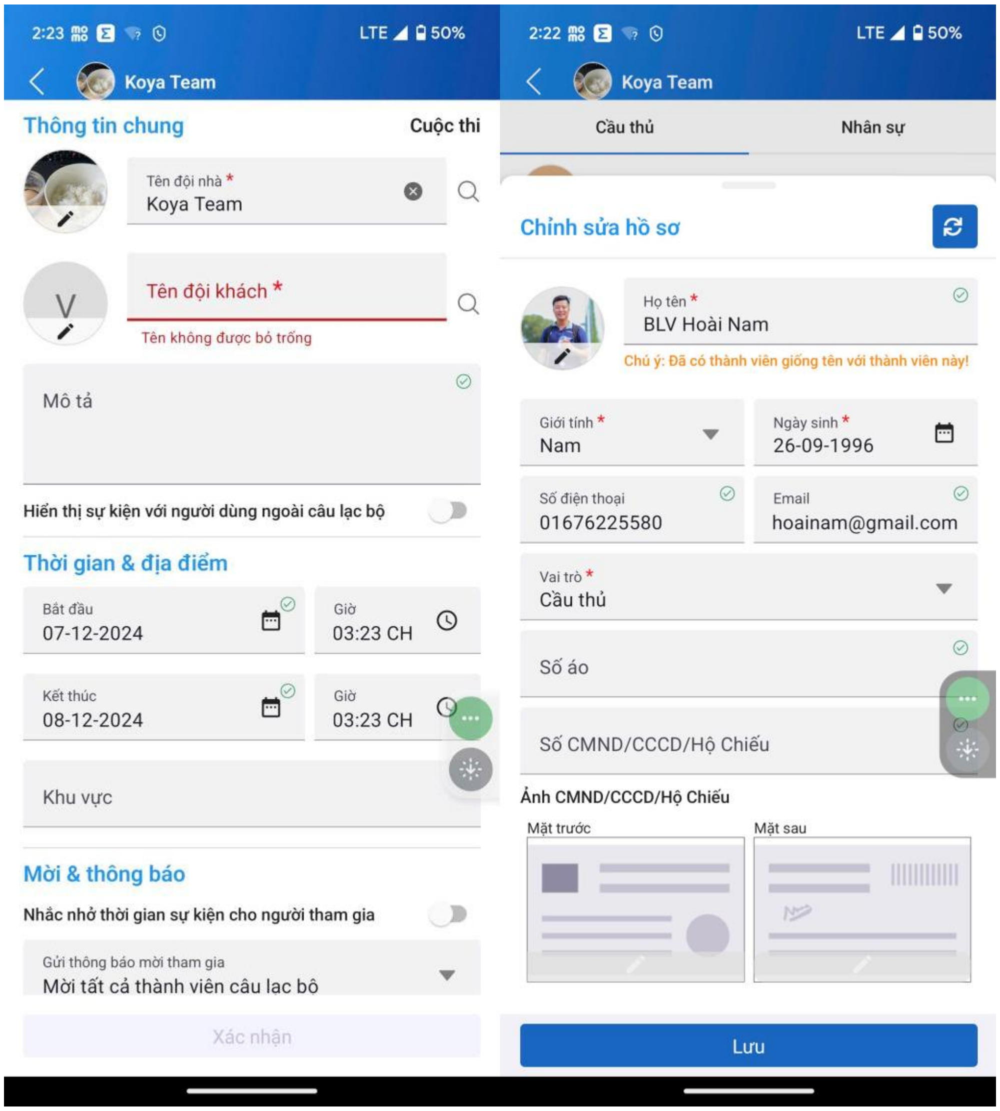
Why the voting system?

To test it out the possibilities what can do in the applications, also to give it a versatile of the products, to make it more feasible to the user, by providing its core components for users to test it out.

d. Build custom components, custom list, custom menu

Another key aspect of my work involves building **custom components, list and menu** tailored to the specific needs of the Vsports platform. These components include reusable elements such as buttons, form controls, modal windows, and tables, which are customized to fit the design guidelines of the platform and improve user experience. In example of the custom components are, when user prompt a function to add teams to the league, or simply adding members to teams, instead of opening a new page, a Bottom Sheet will appear and carry the same functionality, while reloading at the same time, while in addition to that, we can customize the functionality of the bottom sheet, to work as a full page and CRUD all the information.

By the means of custom list and menu, since the applications are using subscription to check if the user using the applications are admin, user or the owner of the teams, league, or the admin of the applications. Often times the UI will be rendering to the most suitable need


2:23 LTE 50%

Koya Team

Thông tin chung — Cuộc thi

Tên đội nhà * Koya Team

Tên đội khách *

Tên không được bỏ trống

Mô tả

Hiển thị sự kiện với người dùng ngoài câu lạc bộ

Thời gian & địa điểm

Bắt đầu 07-12-2024 — Giờ 03:23 CH

Kết thúc 08-12-2024 — Giờ 03:23 CH

Khu vực

Mời & thông báo

Nhắc nhở thời gian sự kiện cho người tham gia

Gửi thông báo mời tham gia: Mời tất cả thành viên câu lạc bộ

Xác nhận

2:22 LTE 50%

Koya Team

Cầu thủ — Nhân sự

Chỉnh sửa hồ sơ

Họ tên * BLV Hoài Nam

Chú ý: Đã có thành viên giống tên với thành viên này!

Giới tính * Nam — Ngày sinh * 26-09-1996

Số điện thoại 01676225580 — Email hoainam@gmail.com

Vai trò * Cầu thủ

Số áo

Số CMND/CCCD/Hộ Chiếu

Ảnh CMND/CCCD/Hộ Chiếu

Mặt trước — Mặt sau

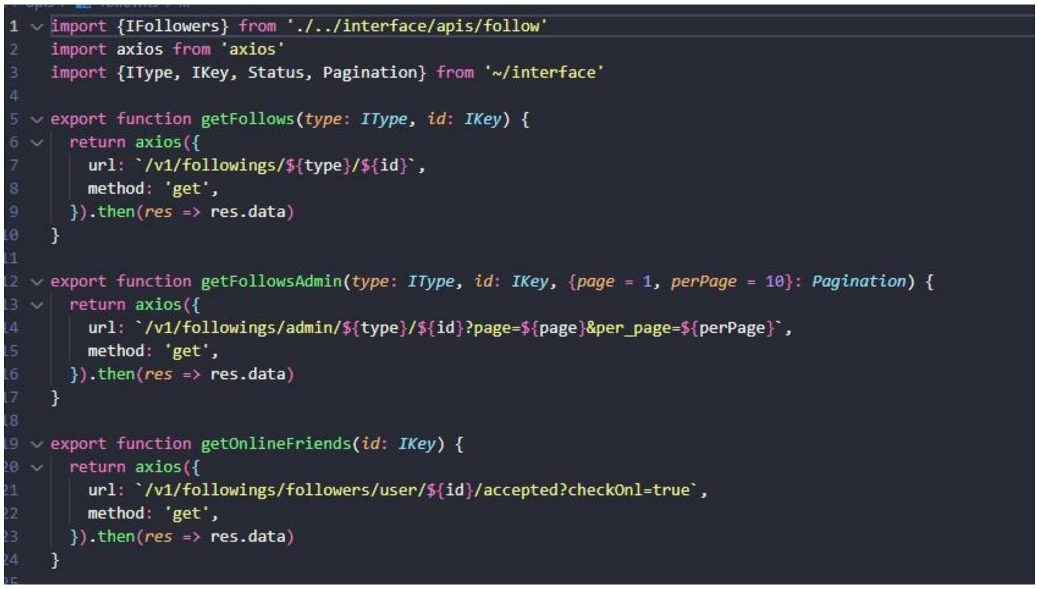
Lưu

Image 2.2.1. Adding teams using full page / custom components

As a key part of building using a custom component, because we have such a good level of customization, with versatility of handling loading functionality, refreshing a page, or simply reducing the renderability of the page. And for those reasons, it is a developer's dream to work with React Native, because of the versatile effects of rendering.

c. *Custom UI with specific requirements*

As I working with the project, one of the key aspects of the Vsports is the level of customizability, with each page have a unique function, or with unique custom requirement, so in order to work with the specific requirements, I need to collaborate with Back-end team, to clear out most of the flow of the requirements and ready to work with on between front-end and back-end



```
import {IFollowers} from './../interface/apis/follow'
import axios from 'axios'
import {IType, IKey, Status, Pagination} from '~/interface'

export function getFollows(type: IType, id: IKey) {
  return axios({
    url: `/v1/followings/${type}/${id}`,
    method: 'get',
  }).then(res => res.data)
}

export function getFollowsAdmin(type: IType, id: IKey, {page = 1, perPage = 10}: Pagination) {
  return axios({
    url: `/v1/followings/admin/${type}/${id}?page=${page}&per_page=${perPage}`,
    method: 'get',
  }).then(res => res.data)
}

export function getOnlineFriends(id: IKey) {
  return axios({
    url: `/v1/followings/followers/user/${id}/accepted?checkOnl=true`,
    method: 'get',
  }).then(res => res.data)
}
```

Image 2.4: Example of adding an API to Front-end

Adding API to the front-end gives it a flexibility of CRUD data for users. Because this application heavily uses data (Data-driven Application). It is crucial to handle the API and give user information to use, read and work with it

d. *Adding and Update user data*

This application is data driven, which means every part of the application works by feeding the data from the user, managing it and updating based on real time scenarios. Vsports is a community app for sports enthusiasts and all the sport managers, the requirement for working with real time data is a must for every developer to develop. And this project require me to create a system to match such a vigorous requirement

III. DETAILS ABOUT THE PROJECT

1. ***User Authentication***

The application is working on and needs user authentication, so the software collects user databases. This user database is crucial to give the application a way to communicate with other API. Sign in and out mostly work with

The front-end of Authentication are like this:

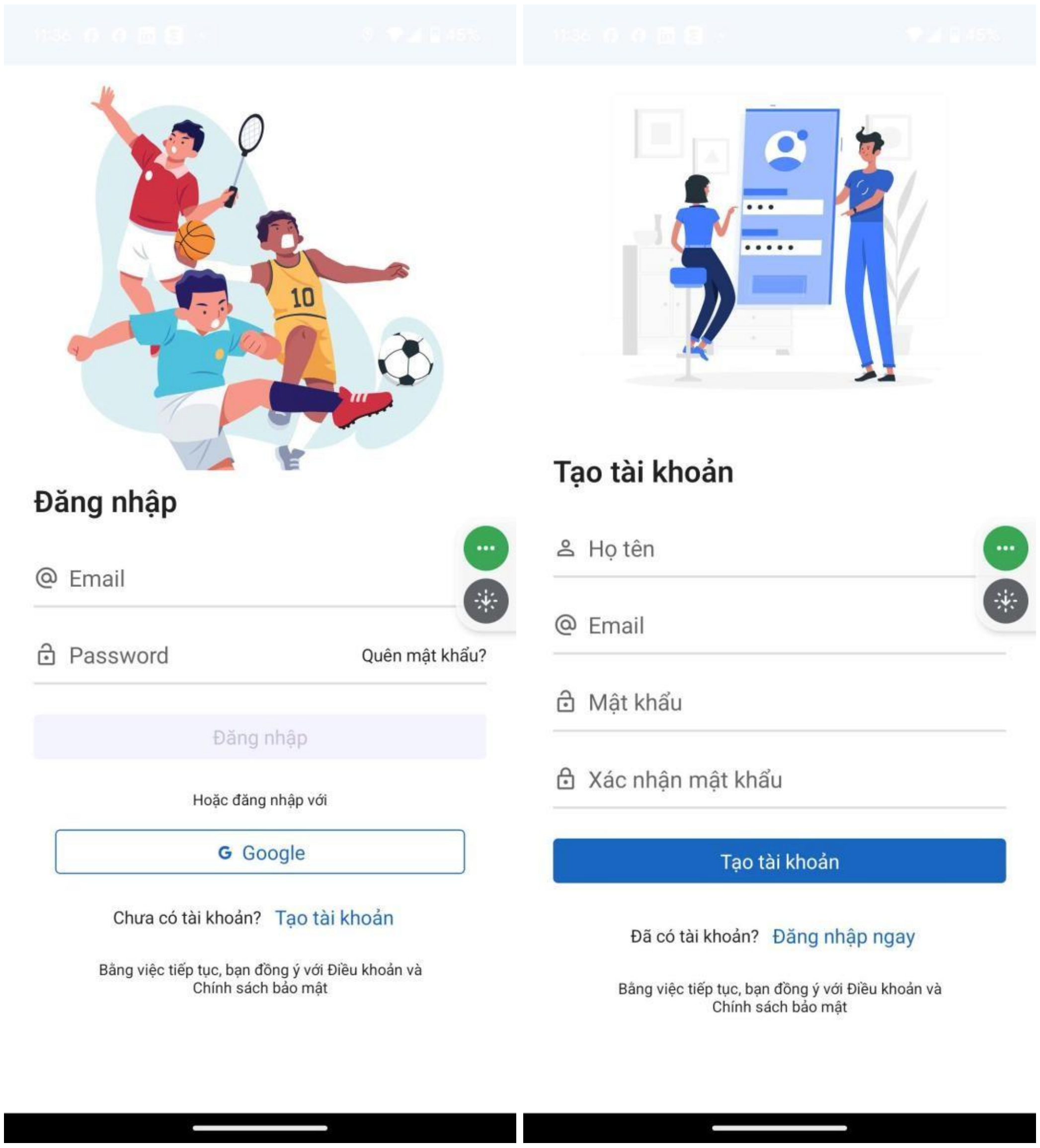
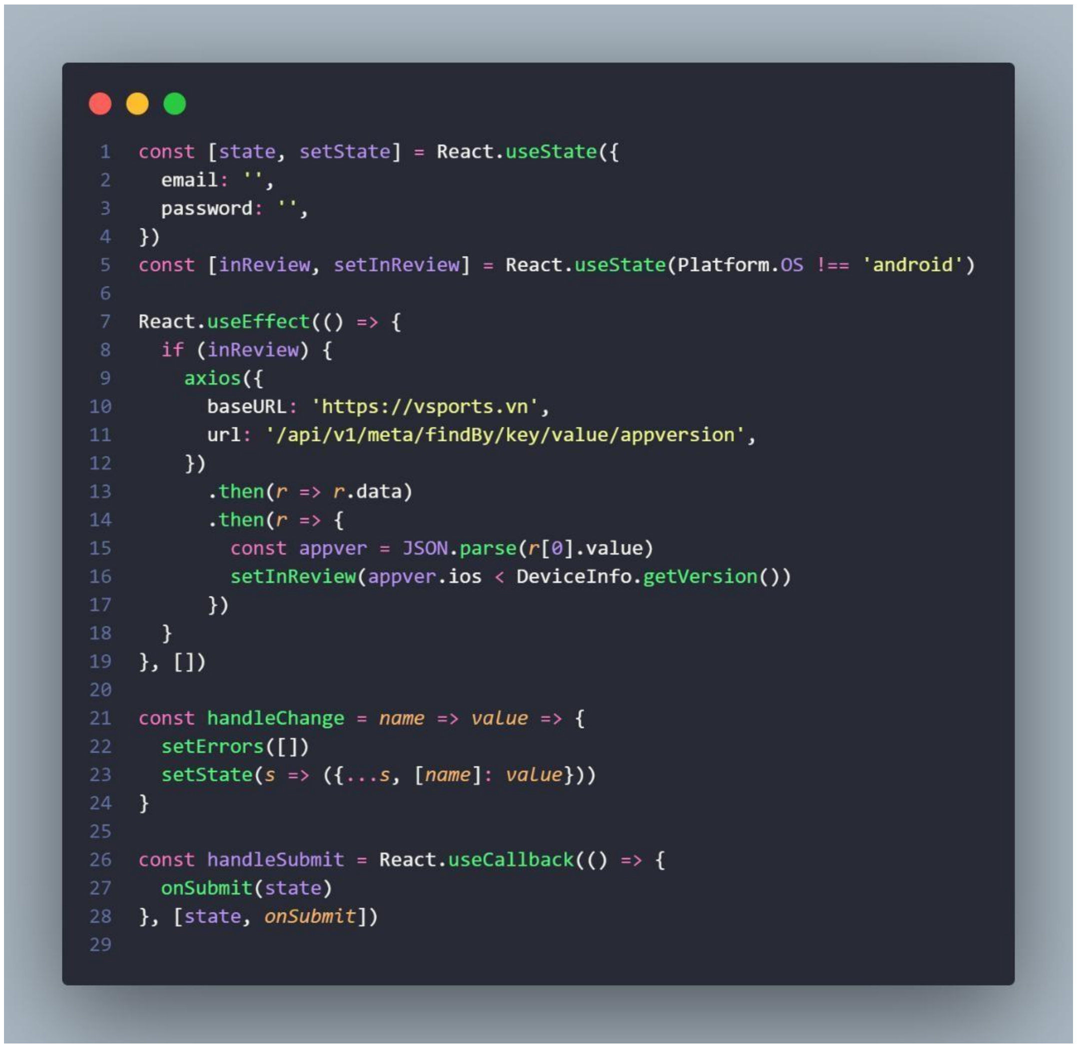

Đăng nhập
Email
Password
Quên mật khẩu?
Đăng nhập
Hoặc đăng nhập với
Google
Chưa có tài khoản? Tạo tài khoản
Bằng việc tiếp tục, bạn đồng ý với Điều khoản và Chính sách bảo mật
10
Tạo tài khoản
Họ tên
Email
Mật khẩu
Xác nhận mật khẩu
Tạo tài khoản
Đã có tài khoản? Đăng nhập ngay
Bằng việc tiếp tục, bạn đồng ý với Điều khoản và Chính sách bảo mật

Image 3.1: User authentication

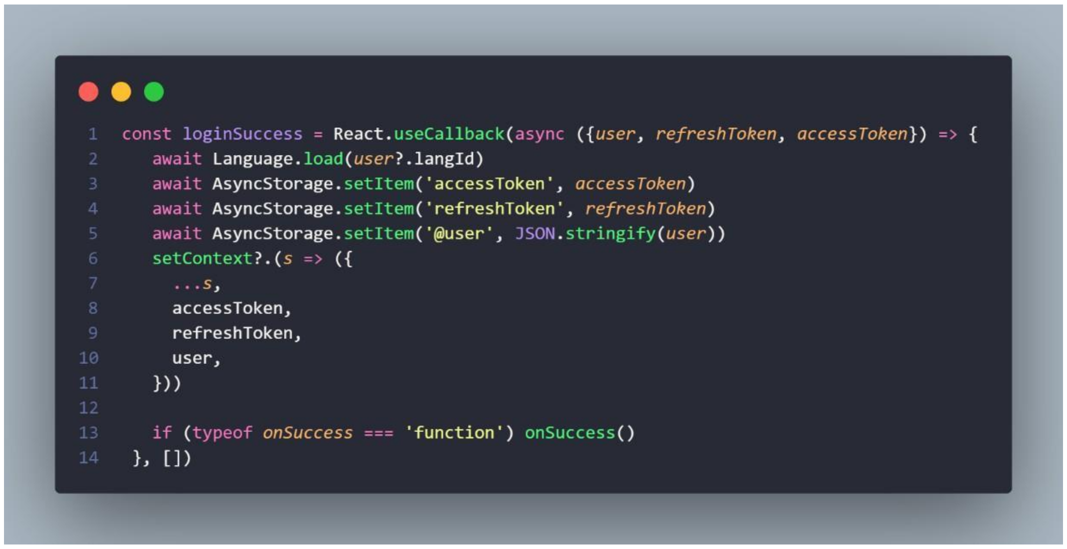
This page shows a normal authentication of the application, using JWT and Access Token with AsyncStorage. This core library is meant to store session tokens of the users, tokens are encrypted to protect the user database, since in my role as a front-end, the encryption process cannot be revealed.



```
const [state, setState] = React.useState({
  email: '',
  password: '',
})
const [inReview, setInReview] = React.useState(Platform.OS !== 'android')

React.useEffect(() => {
  if (inReview) {
    axios({
      baseURL: 'https://vsports.vn',
      url: '/api/v1/meta/findBy/key/value/appversion',
    })
      .then(r => r.data)
      .then(r => {
        const appver = JSON.parse(r[0].value)
        setInReview(appver.ios < DeviceInfo.getVersion())
      })
  }
}, [])

const handleChange = name => value => {
  setErrors([])
  setState(s => ({...s, [name]: value}))
}

const handleSubmit = React.useCallback(() => {
  onSubmit(state)
}, [state, onSubmit])
```

Image 3.2: Process of handling Authentication

As the code stated, the process of handling authentication works directly to the React **useEffect().** Instead of creating a separate function to handle authentication, putting in the **useEffect()** with no dependencies is an efficient way to not have to manually render the process. Also, while checking the form to sign in / up, it will automatically detect which version of the version of Vsports it is using, if the user is using an older version of Vsports, it will automatically prompt the user to update the application to the newest version of Vsports.



```
const loginSuccess = React.useCallback(async ({user, refreshToken, accessToken}) => {
   await Language.load(user?.langId)
   await AsyncStorage.setItem('accessToken', accessToken)
   await AsyncStorage.setItem('refreshToken', refreshToken)
   await AsyncStorage.setItem('@user', JSON.stringify(user))
   setContext?.(s => ({
     ...s,
     accessToken,
     refreshToken,
     user,
   }))

   if (typeof onSuccess === 'function') onSuccess()
 }, [])
```

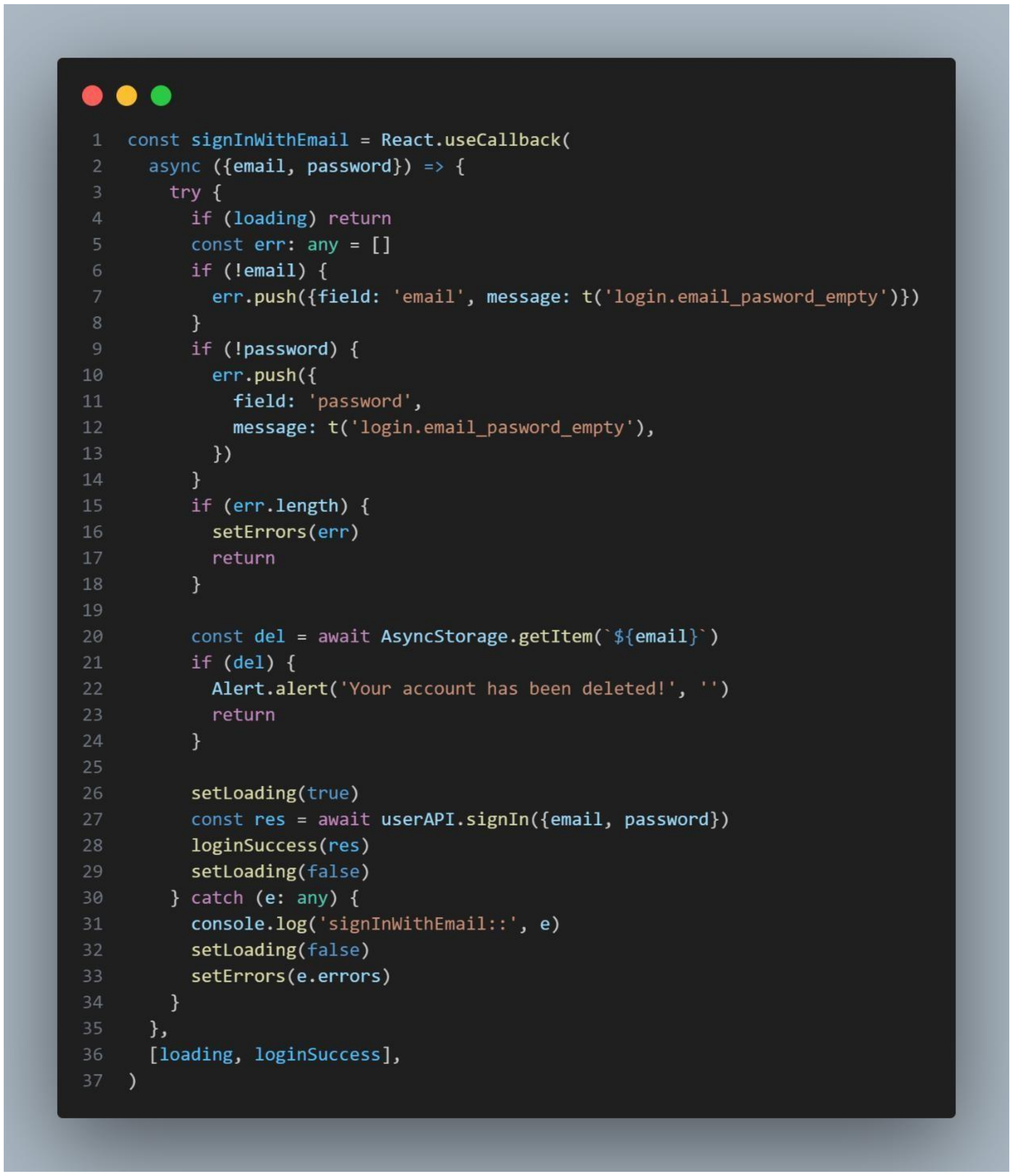
Image 3.3: Process of handling Authentication using token

As stated, in order to access the login securely, we'll optimize how to use token as a part of the authentication. With access to **refreshToken** and **accessToken** to authenticate the user information. If we are using **detokenizer** to decode the token, will see the user information inside the body containing:

users: {

> **name: string | undefined password:**
>
> **string | undefined id: number |**
>
> **undefined**

}

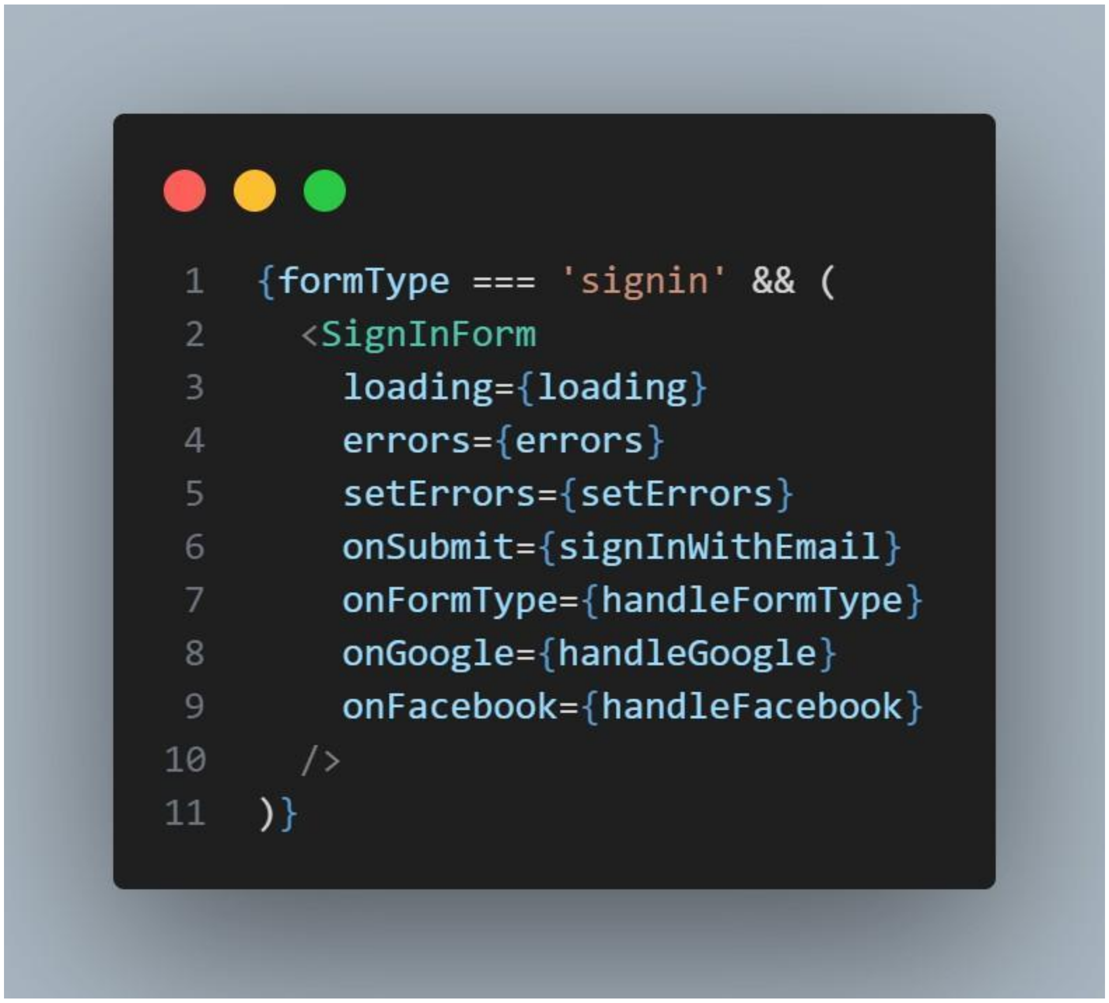


```
const signInWithEmail = React.useCallback(
  async ({email, password}) => {
    try {
      if (loading) return
      const err: any = []
      if (!email) {
        err.push({field: 'email', message: t('login.email_pasword_empty')})
      }
      if (!password) {
        err.push({
          field: 'password',
          message: t('login.email_pasword_empty'),
        })
      }
      if (err.length) {
        setErrors(err)
        return
      }

      const del = await AsyncStorage.getItem(`${email}`)
      if (del) {
        Alert.alert('Your account has been deleted!', '')
        return
      }

      setLoading(true)
      const res = await userAPI.signIn({email, password})
      loginSuccess(res)
      setLoading(false)
    } catch (e: any) {
      console.log('signInWithEmail::', e)
      setLoading(false)
      setErrors(e.errors)
    }
  },
  [loading, loginSuccess],
)
```

Image 3.1.3: Example of Sign In with email

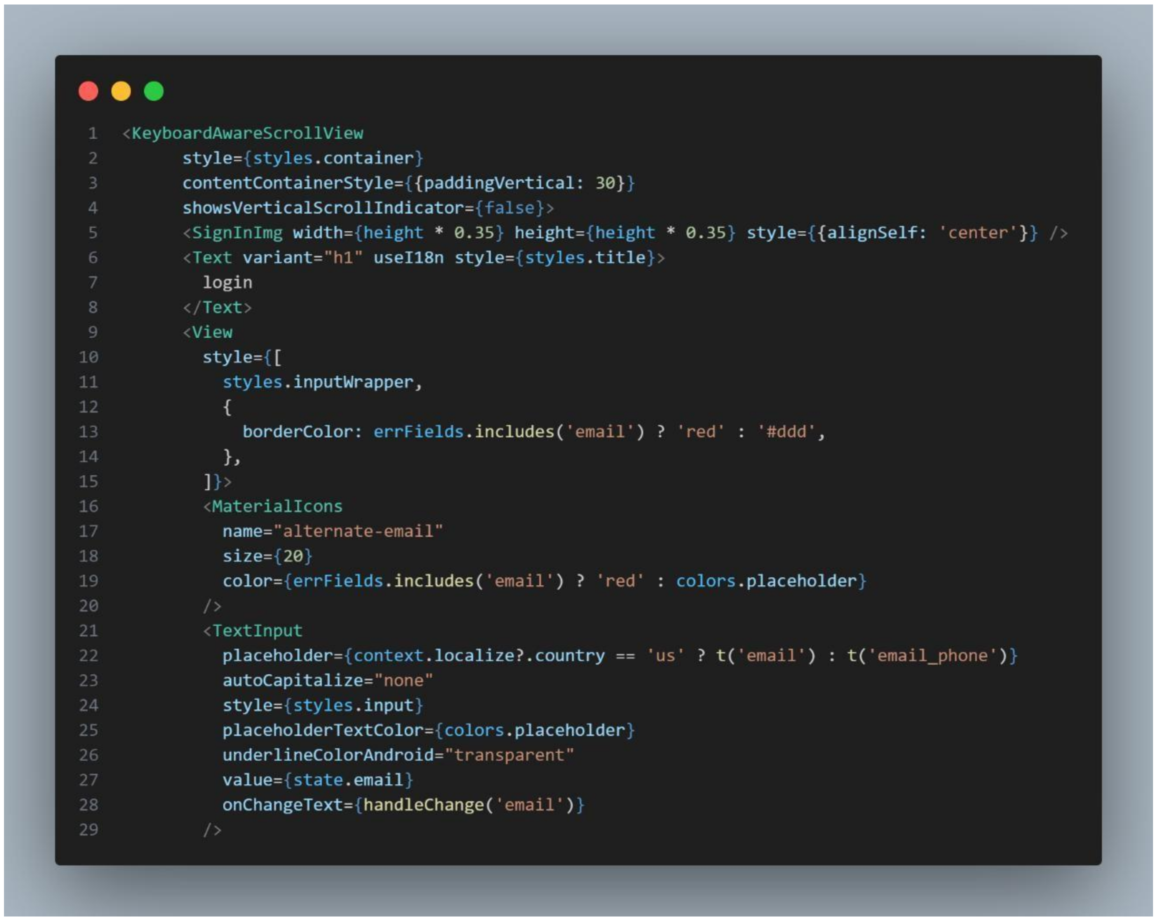
This code will handle the processing authentication, catching error when sign in with incorrect credentials, deleted account.

And from this state, we will update with UI elements stated in image 3.1

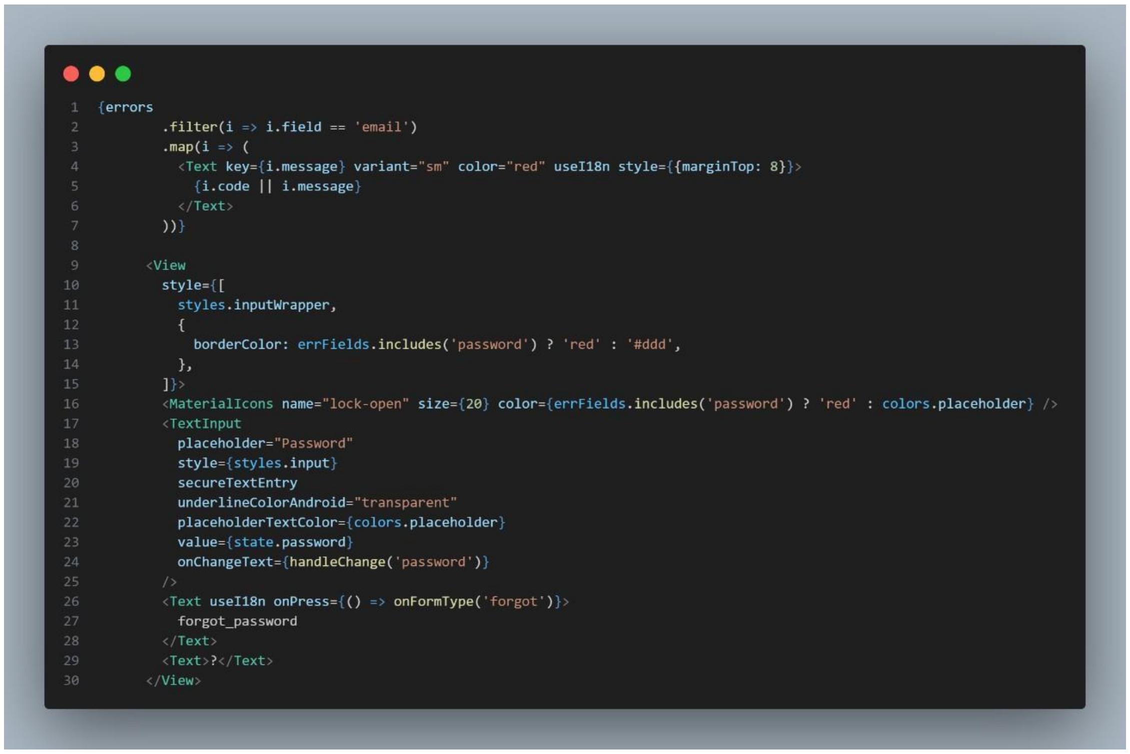


```
{formType === 'signin' && (
  <SignInForm
    loading={loading}
    errors={errors}
    setErrors={setErrors}
    onSubmit={signInWithEmail}
    onFormType={handleFormType}
    onGoogle={handleGoogle}
    onFacebook={handleFacebook}
  />
)}
```

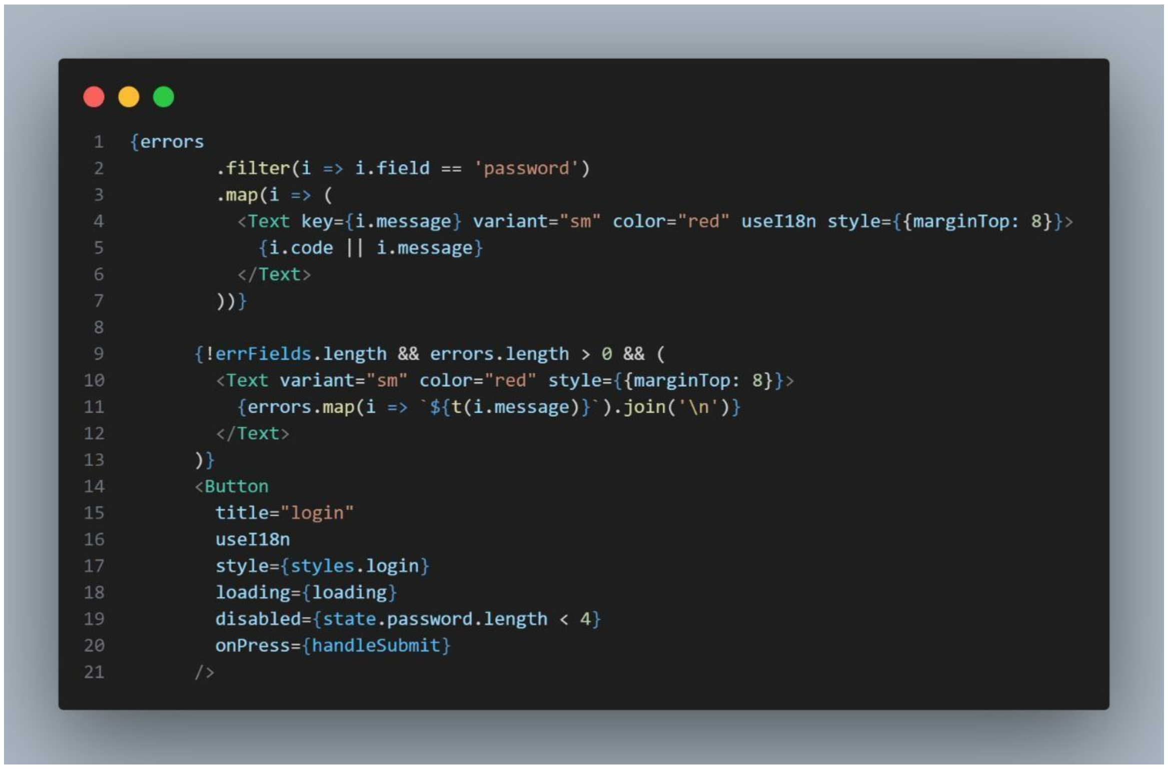
Image 3.1.4: An Sign In Form using as a components style



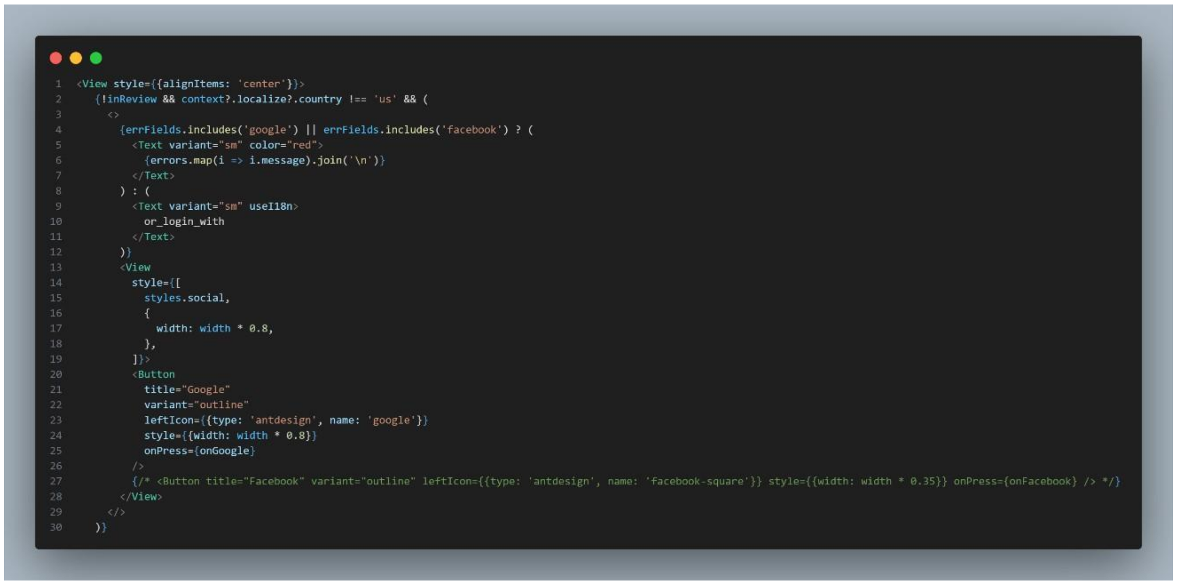
```
<KeyboardAwareScrollView
      style={styles.container}
      contentContainerStyle={{paddingVertical: 30}}
      showsVerticalScrollIndicator={false}>
      <SignInImg width={height * 0.35} height={height * 0.35} style={{alignSelf: 'center'}} />
      <Text variant="h1" useI18n style={styles.title}>
        login
      </Text>
      <View
        style={[
          styles.inputWrapper,
          {
            borderColor: errFields.includes('email') ? 'red' : '#ddd',
          },
        ]}>
        <MaterialIcons
          name="alternate-email"
          size={20}
          color={errFields.includes('email') ? 'red' : colors.placeholder}
        />
        <TextInput
          placeholder={context.localize?.country == 'us' ? t('email') : t('email_phone')}
          autoCapitalize="none"
          style={styles.input}
          placeholderTextColor={colors.placeholder}
          underlineColorAndroid="transparent"
          value={state.email}
          onChangeText={handleChange('email')}
        />
```



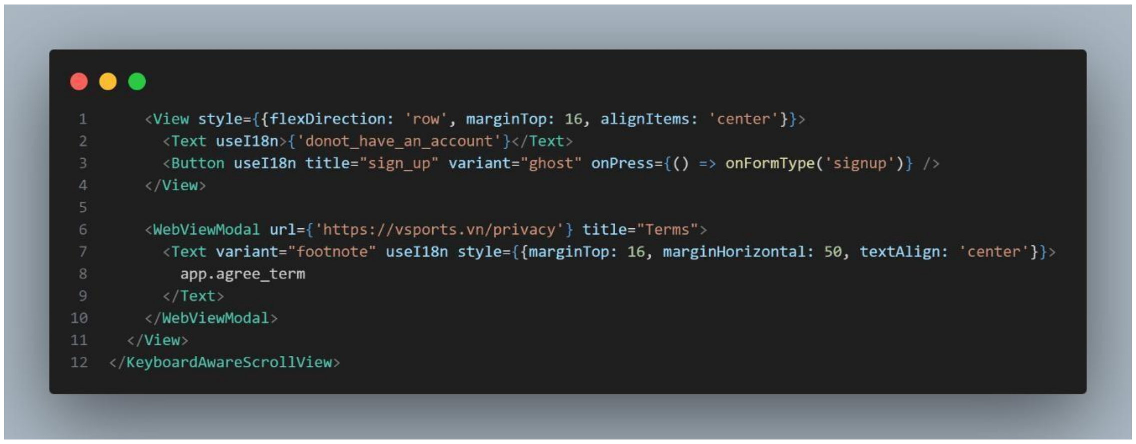
```
{errors
        .filter(i => i.field == 'email')
        .map(i => (
          <Text key={i.message} variant="sm" color="red" useI18n style={{marginTop: 8}}>
            {i.code || i.message}
          </Text>
        ))}

      <View
        style={[
          styles.inputWrapper,
          {
            borderColor: errFields.includes('password') ? 'red' : '#ddd',
          },
        ]}>
        <MaterialIcons name="lock-open" size={20} color={errFields.includes('password') ? 'red' : colors.placeholder} />
        <TextInput
          placeholder="Password"
          style={styles.input}
          secureTextEntry
          underlineColorAndroid="transparent"
          placeholderTextColor={colors.placeholder}
          value={state.password}
          onChangeText={handleChange('password')}
        />
        <Text useI18n onPress={() => onFormType('forgot')}>
          forgot_password
        </Text>
        <Text>?</Text>
      </View>
```



```jsx
{errors
        .filter(i => i.field == 'password')
        .map(i => (
          <Text key={i.message} variant="sm" color="red" useI18n style={{marginTop: 8}}>
            {i.code || i.message}
          </Text>
        ))}

      {!errFields.length && errors.length > 0 && (
        <Text variant="sm" color="red" style={{marginTop: 8}}>
          {errors.map(i => `${t(i.message)}`).join('\n')}
        </Text>
      )}
      <Button
        title="login"
        useI18n
        style={styles.login}
        loading={loading}
        disabled={state.password.length < 4}
        onPress={handleSubmit}
      />
```



```jsx
<View style={{alignItems: 'center'}}>
  {!inReview && context?.localize?.country !== 'us' && (
    <>
      {errFields.includes('google') || errFields.includes('facebook') ? (
        <Text variant="sm" color="red">
          {errors.map(i => i.message).join('\n')}
        </Text>
      ) : (
        <Text variant="sm" useI18n>
          or_login_with
        </Text>
      )}
      <View
        style={[
          styles.social,
          {
            width: width * 0.8,
          },
        ]}>
        <Button
          title="Google"
          variant="outline"
          leftIcon={{type: 'antdesign', name: 'google'}}
          style={{width: width * 0.8}}
          onPress={onGoogle}
        />
        {/* <Button title="Facebook" variant="outline" leftIcon={{type: 'antdesign', name: 'facebook-square'}} style={{width: width * 0.35}} onPress={onFacebook} /> */}
      </View>
    </>
  )}
```



```
    <View style={{flexDirection: 'row', marginTop: 16, alignItems: 'center'}}>
      <Text useI18n>{'donot_have_an_account'}</Text>
      <Button useI18n title="sign_up" variant="ghost" onPress={() => onFormType('signup')} />
    </View>

    <WebViewModal url={'https://vsports.vn/privacy'} title="Terms">
      <Text variant="footnote" useI18n style={{marginTop: 16, marginHorizontal: 50, textAlign: 'center'}}>
        app.agree_term
      </Text>
    </WebViewModal>
  </View>
</KeyboardAwareScrollView>
```

Image 3.1.5: Sample code of creating an UI Elements for Sign In

2. ***Evaluation and Error Handling***

Working evaluation and **error handling** are crucial aspects of software development, ensuring that systems function reliably and efficiently under various conditions. These practices involve systematically testing and monitoring code execution, identifying potential issues, and implementing mechanisms to handle errors gracefully.

1. Working Evaluation

Working evaluation refers to the process of assessing the functionality and performance of a system or application. It is to make sure:

- **Testing application Validation**: Ensuring that code work as expected through unit, integration, and system testing, and consulting with
- **Behavior Verification**: Verifying that the system meets business requirements and provides a seamless user experience. Like when user click a button, an action with a delay will occurs and will open a Bottom Sheet to show UI Elements

Effective working evaluation relies on continuous integration tools and automated testing frameworks, which help developers catch issues early in the development lifecycle.

2. Error Handling

Error handling focuses on anticipating, detecting, and managing runtime errors to maintain system stability. Key aspects include:

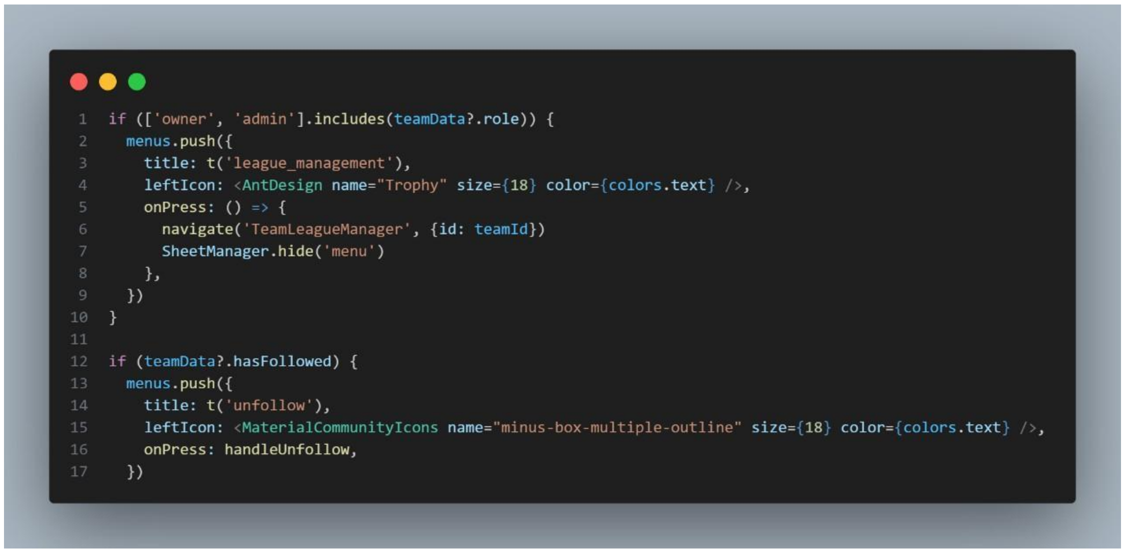
- **Error Detection**: Using try-catch blocks, error codes, or logging to identify issues during code execution.
- **Error Recovery**: Implementing strategies to recover from errors, such as retrying operations or reverting to a stable state.
- **User-Friendly Feedback**: Communicating issues to users in a clear and actionable manner without exposing sensitive system details.

  Proactive error handling minimizes disruptions, prevents data loss, and ensures consistent application behavior even in edge cases.

Importance in Development

- **Reliability**: Ensures applications remain functional despite unexpected inputs or failures.
- **Maintainability**: Simplifies debugging by providing detailed error logs and well-defined exception handling patterns.
- **User Experience**: Protects end-users from encountering crashes or confusing error messages.

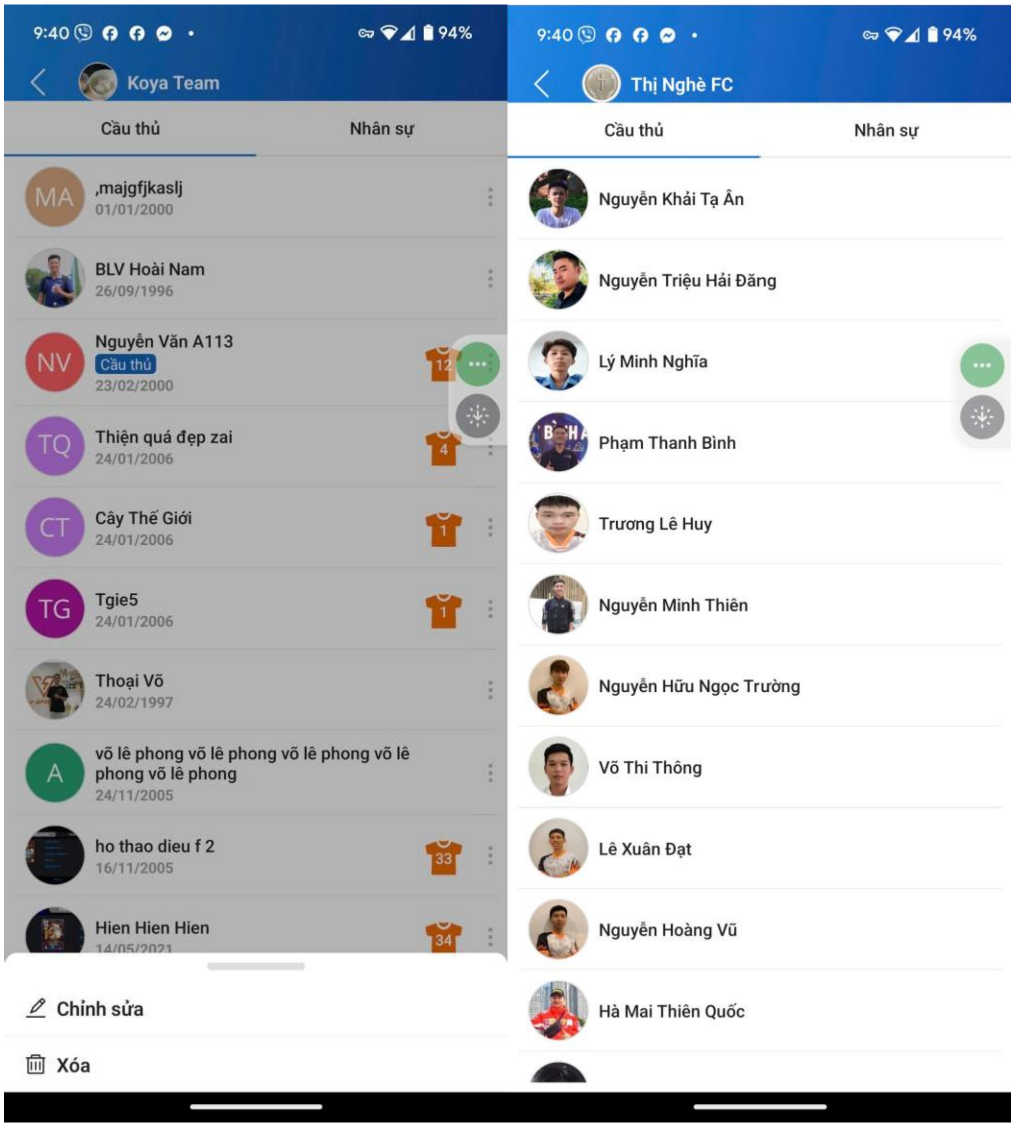
By integrating robust evaluation and error-handling practices, developers can build resilient applications that deliver high-quality user experiences.



```
if (['owner', 'admin'].includes(teamData?.role)) {
  menus.push({
    title: t('league_management'),
    leftIcon: <AntDesign name="Trophy" size={18} color={colors.text} />,
    onPress: () => {
      navigate('TeamLeagueManager', {id: teamId})
      SheetManager.hide('menu')
    },
  })
}

if (teamData?.hasFollowed) {
  menus.push({
    title: t('unfollow'),
    leftIcon: <MaterialCommunityIcons name="minus-box-multiple-outline" size={18} color={colors.text} />,
    onPress: handleUnfollow,
  })
```

Image 3.2.1: This is the example of error checking and handling

In the example, we can see the option whether the Administration or the Owner of the team, the UI Bottom Sheet will handle the rendering some of the options for user to check out. Here’s are the example of the following rules to be check out



MA ,majgfjkaslj 01/01/2000

BLV Hoài Nam 26/09/1996

NV Nguyễn Văn A113 Cầu thủ 23/02/2000 12

TQ Thiện quá đẹp zai 24/01/2006 4

CT Cây Thế Giới 24/01/2006 1

TG Tgie5 24/01/2006 1

Thoại Võ 24/02/1997

A võ lê phong võ lê phong võ lê phong võ lê phong võ lê phong 24/11/2005

ho thao dieu f 2 16/11/2005 33

Hien Hien Hien 14/05/2021 34

Chỉnh sửa

Xóa

Nguyễn Khải Tạ Ân

Nguyễn Triệu Hải Đăng

Lý Minh Nghĩa

Phạm Thanh Bình

Trương Lê Huy

Nguyễn Minh Thiên

Nguyễn Hữu Ngọc Trường

Võ Thi Thông

Lê Xuân Đạt

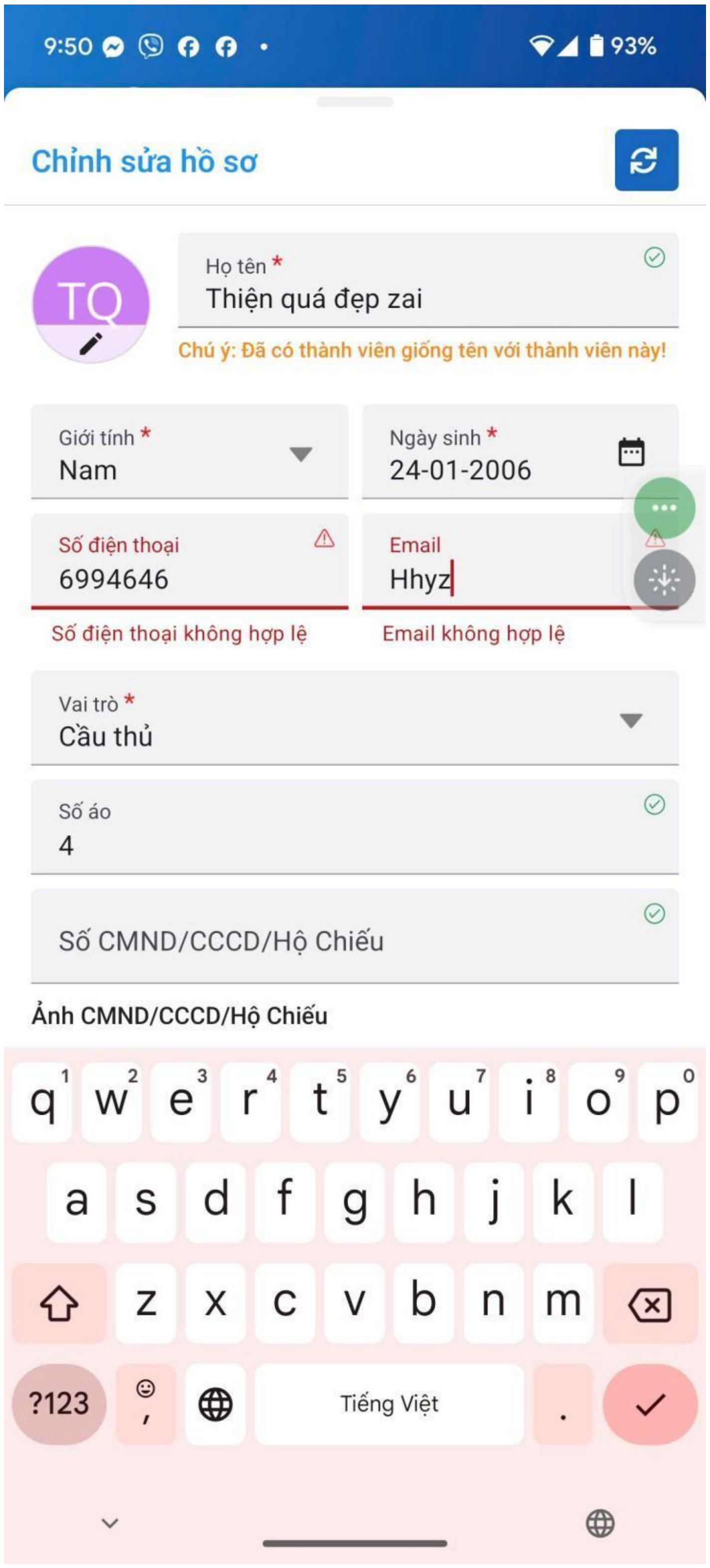
Nguyễn Hoàng Vũ

Hà Mai Thiên Quốc

Image: Sample of rendering the options for team data

This is the example on evaluation on how the rendering work with permission of which is to check whether myself is the owner or the admin or just the guest

Beside creating an evaluation of the project, error handling is a crucial part for this project, because of the necessity to input some of the data to the database, and all the database is linked together. A proper error handling is crucial for letting the correct data to the database, without having to try-catch too much to the code.



9:50 93%

Chỉnh sửa hồ sơ

TQ

Họ tên *
Thiện quá đẹp zai
Chú ý: Đã có thành viên giống tên với thành viên này!

Giới tính *
Nam

Ngày sinh *
24-01-2006

Số điện thoại
6994646
Số điện thoại không hợp lệ

Email
Hhyz
Email không hợp lệ

Vai trò *
Cầu thủ

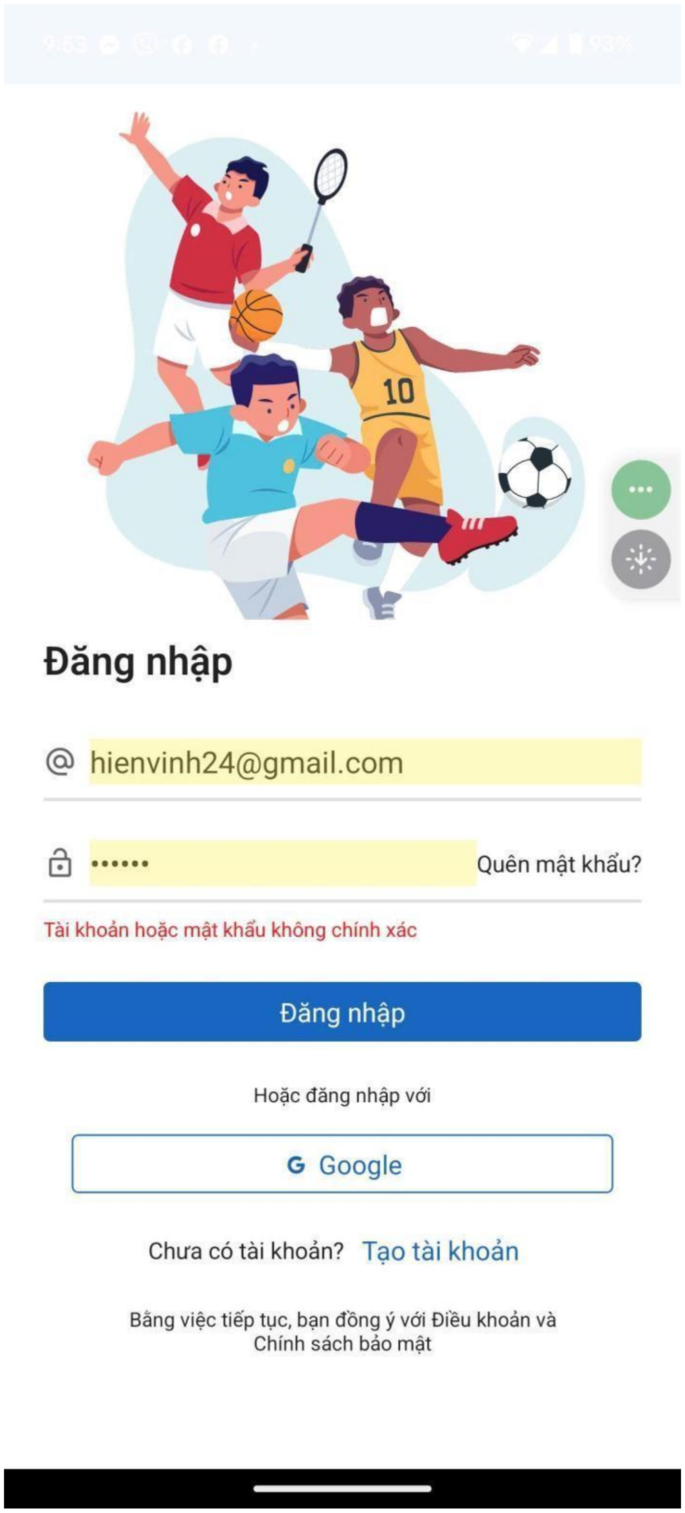
Số áo
4

Số CMND/CCCD/Hộ Chiếu

Ảnh CMND/CCCD/Hộ Chiếu

Image: Sample of data checking, error validation

From the above, we can see the error is handling when user is input the incorrect data to the database. The input need to be input correctly in order to perform the next step.



Đăng nhập

hienvinh24@gmail.com

•••••• Quên mật khẩu?

Tài khoản hoặc mật khẩu không chính xác

Đăng nhập

Hoặc đăng nhập với

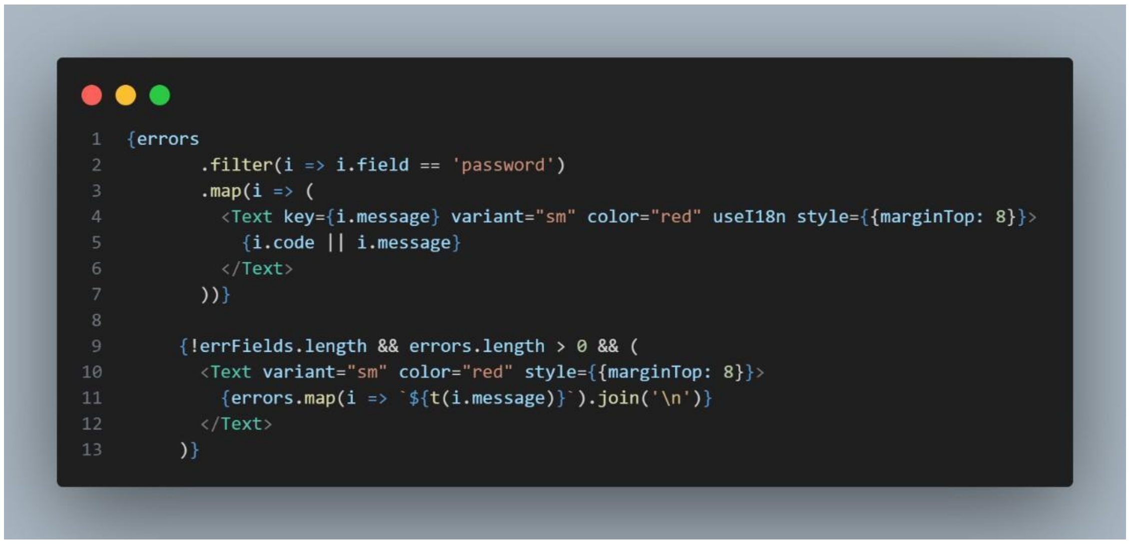
Google

Chưa có tài khoản? Tạo tài khoản

Bằng việc tiếp tục, bạn đồng ý với Điều khoản và Chính sách bảo mật

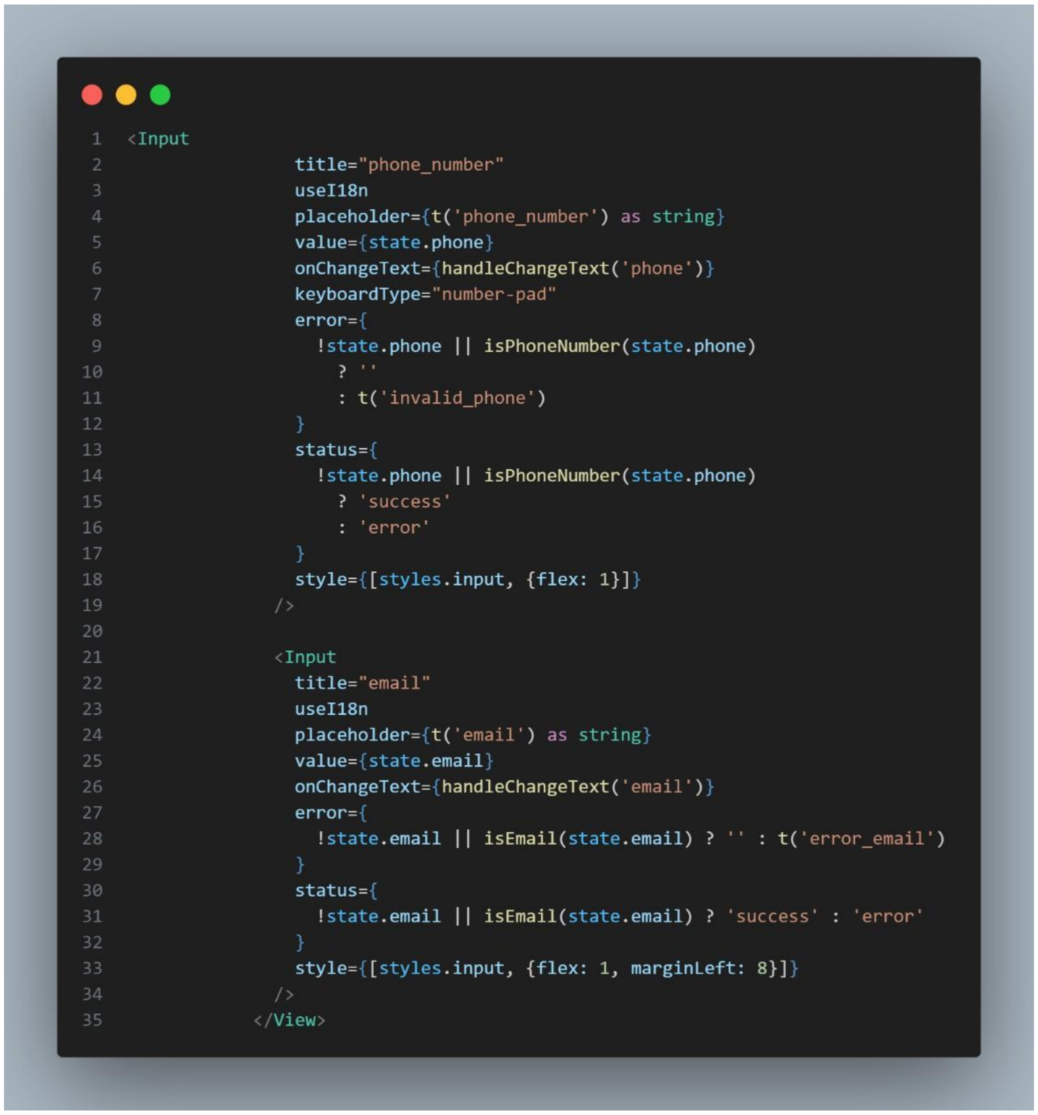
Image: Sample of the sign in with incorrect credentials

Same as the example of the error validation, this error will rejects if user input the incorrect credentials, so the user cannot sign in to the application and need to be sign in properly, in order to use the app with all the functionality



```
{errors
       .filter(i => i.field == 'password')
       .map(i => (
         <Text key={i.message} variant="sm" color="red" useI18n style={{marginTop: 8}}>
           {i.code || i.message}
         </Text>
       ))}

     {!errFields.length && errors.length > 0 && (
       <Text variant="sm" color="red" style={{marginTop: 8}}>
         {errors.map(i => `${t(i.message)}`).join('\n')}
       </Text>
     )}
```

Image: Error validation of the signing modal



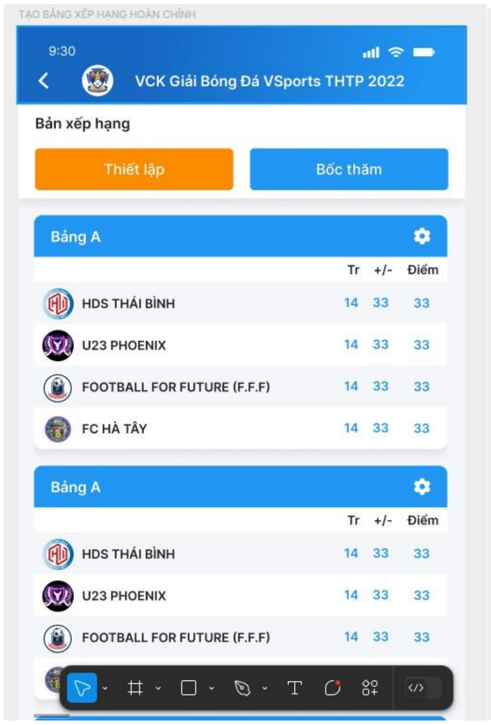
```
<Input
                title="phone_number"
                useI18n
                placeholder={t('phone_number') as string}
                value={state.phone}
                onChangeText={handleChangeText('phone')}
                keyboardType="number-pad"
                error={
                  !state.phone || isPhoneNumber(state.phone)
                    ? ''
                    : t('invalid_phone')
                }
                status={
                  !state.phone || isPhoneNumber(state.phone)
                    ? 'success'
                    : 'error'
                }
                style={[styles.input, {flex: 1}]}
              />

              <Input
                title="email"
                useI18n
                placeholder={t('email') as string}
                value={state.email}
                onChangeText={handleChangeText('email')}
                error={
                  !state.email || isEmail(state.email) ? '' : t('error_email')
                }
                status={
                  !state.email || isEmail(state.email) ? 'success' : 'error'
                }
                style={[styles.input, {flex: 1, marginLeft: 8}]}
              />
            </View>
```

Image: This is the example of checking errors at the team edit code.

As I continue to work, because of the complexity of the project, a more specific code base is required to check.

3. ***Building custom components***

With strict requirements on how the application work, some information need to be render it properly, Custom components play a pivotal role in developing the Vsports, enabling the creation of unique, reusable, and visually cohesive user interfaces tailored to the application's functionality and branding. Sports details, team details, modal, or even some achievement related to the sports, are made it from the UX/UI designer.


TẠO BẢNG XẾP HẠNG HOÀN CHỈNH

9:30

VCK Giải Bóng Đá VSports THTP 2022

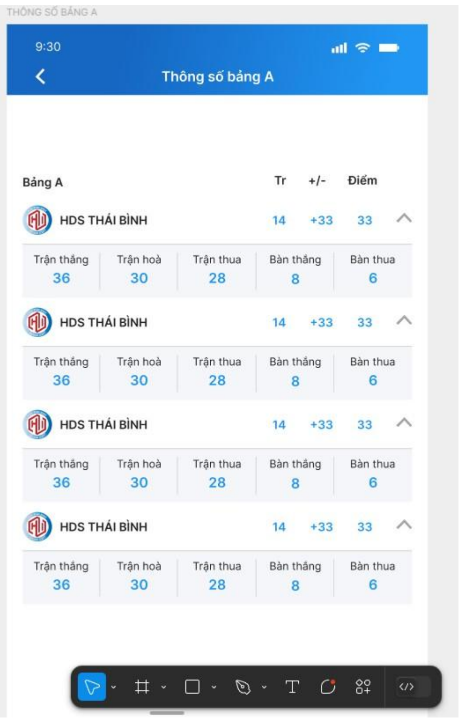
Bản xếp hạng

Bảng A

	Tr	+/-	Điểm
HDS THÁI BÌNH	14	33	33
U23 PHOENIX	14	33	33
FOOTBALL FOR FUTURE (F.F.F)	14	33	33
FC HÀ TÂY	14	33	33

Bảng A

	Tr	+/-	Điểm
HDS THÁI BÌNH	14	33	33
U23 PHOENIX	14	33	33
FOOTBALL FOR FUTURE (F.F.F)	14	33	33


THÔNG SỐ BẢNG A

9:30

Thông số bảng A

Bảng A	Tr	+/-	Điểm
HDS THÁI BÌNH	14	+33	33

Trận thắng	Trận hoà	Trận thua	Bàn thắng	Bàn thua
36	30	28	8	6

	Tr	+/-	Điểm
HDS THÁI BÌNH	14	+33	33

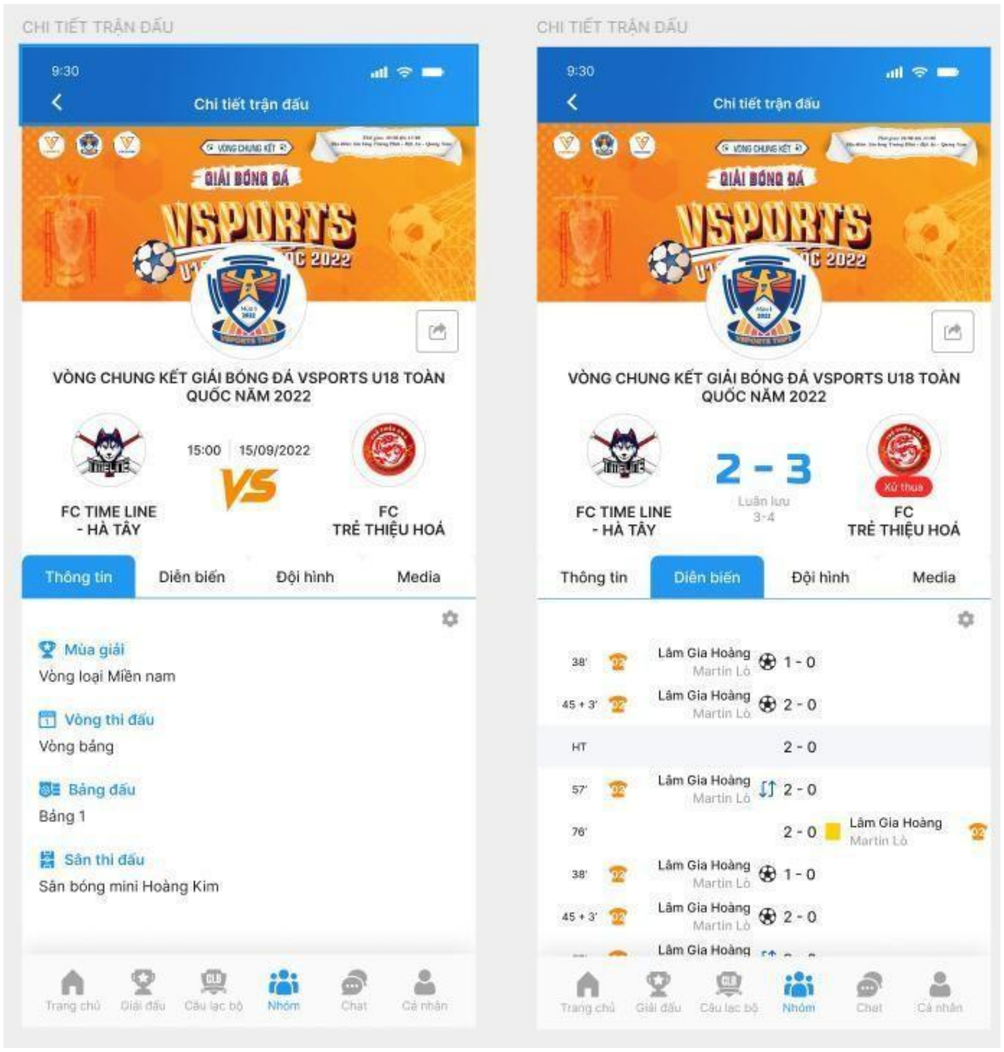
Trận thắng	Trận hoà	Trận thua	Bàn thắng	Bàn thua
36	30	28	8	6

	Tr	+/-	Điểm
HDS THÁI BÌNH	14	+33	33

Trận thắng	Trận hoà	Trận thua	Bàn thắng	Bàn thua
36	30	28	8	6

	Tr	+/-	Điểm
HDS THÁI BÌNH	14	+33	33

Trận thắng	Trận hoà	Trận thua	Bàn thắng	Bàn thua
36	30	28	8	6



VÒNG CHUNG KẾT GIẢI BÓNG ĐÁ VSPORTS U18 TOÀN QUỐC NĂM 2022

FC TIME LINE - HÀ TÂY 15:00 15/09/2022 VS FC TRẺ THIỆU HOÁ

Mùa giải
Vòng loại Miền nam

Vòng thi đấu
Vòng bảng

Bảng đấu
Bảng 1

Sân thi đấu
Sân bóng mini Hoàng Kim

VÒNG CHUNG KẾT GIẢI BÓNG ĐÁ VSPORTS U18 TOÀN QUỐC NĂM 2022

FC TIME LINE - HÀ TÂY 2 - 3 Luân lưu 3-4 Xử thua FC TRẺ THIỆU HOÁ

38'	Lâm Gia Hoàng Martin Lò	1 - 0	
45 + 3'	Lâm Gia Hoàng Martin Lò	2 - 0	
HT		2 - 0	
57'	Lâm Gia Hoàng Martin Lò	2 - 0	
76'		2 - 0	Lâm Gia Hoàng Martin Lò
38'	Lâm Gia Hoàng Martin Lò	1 - 0	
45 + 3'	Lâm Gia Hoàng Martin Lò	2 - 0	
	Lâm Gia Hoàng		

Image: These are these examples of custom components.

From the example above, as it represents its core of the custom components, such as building a functional Tab, custom table for reviewing table score, table comparison of the score and teams, with custom rendering method for more information.

These are the work I must do to keep up with the demand of the team, or the expansion of the league that needs more information included in the application. All the source code can be tricky, but a little bit of work can make some of the differences.

Why are we building custom components instead?

These are some of the examples of creating a custom component:

- ***Utilize Props for Customization***

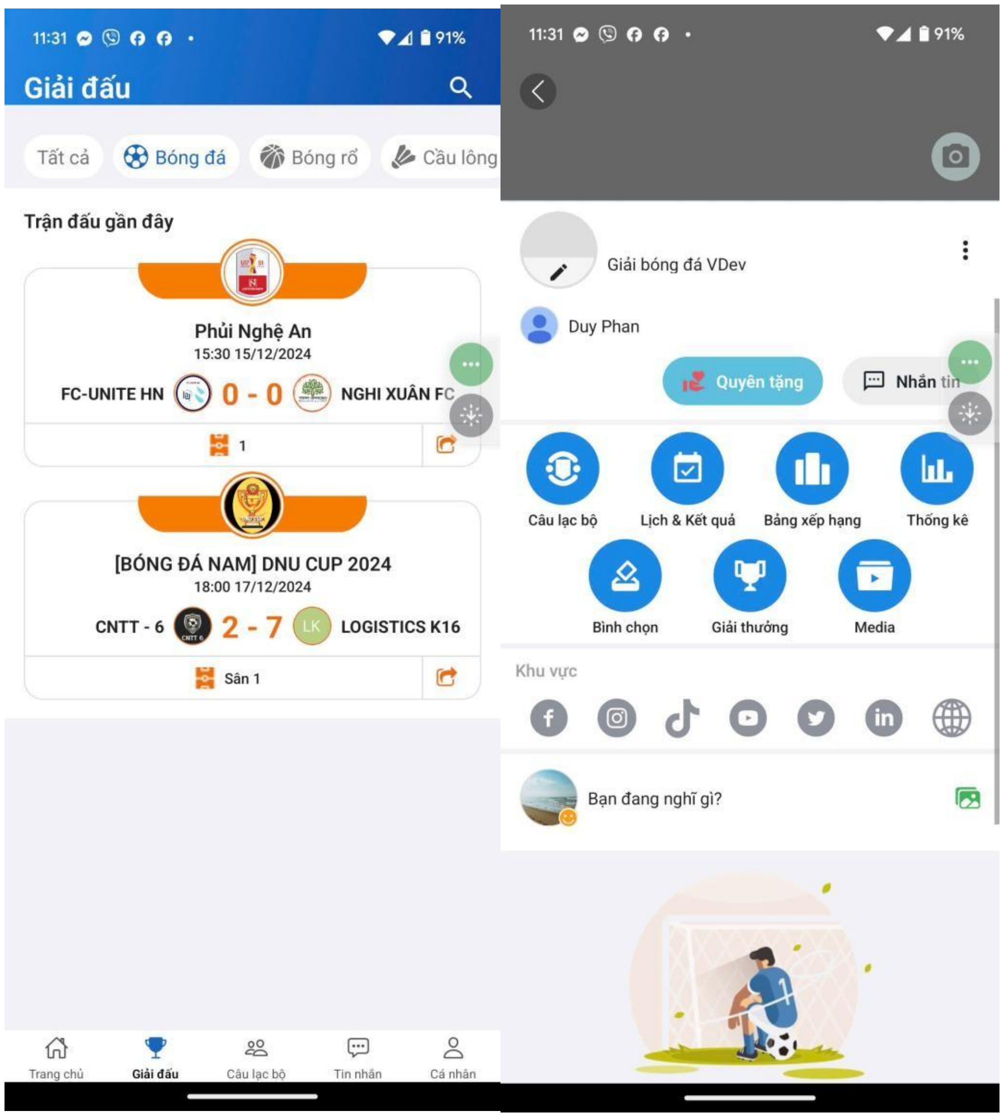
  By creating the custom components, your props for the components are unlimited, we have the control for all the aspect of the rendering method

- ***Handle State for Interactivity***

  When building custom components, we can freely custom using Hooks, **useEffect()** inside within the components. For example, when a list is rendered, it can automatically refresh with **useEffect()** and add the loading when the user pulls down.

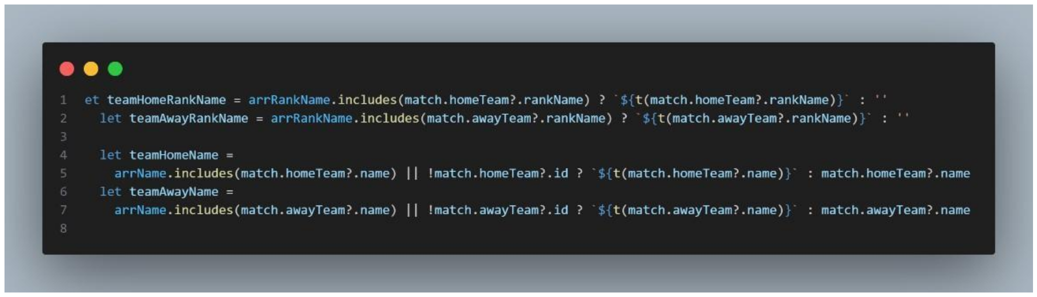
- ***Reuse Across Screens****:*

  Components like "Tabs" or “MatchCard” can be reused across screens such as "Profiles," "Match Details," and "Home" ensuring modularity and reduced code duplication.



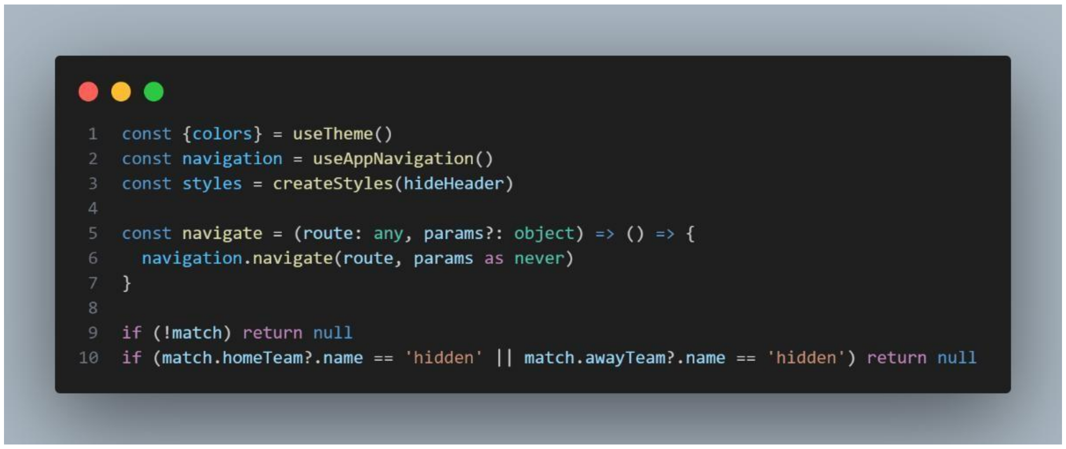
Examples of custom UI

From the example, we can see the the custom components such as match card, and custom button with rendering



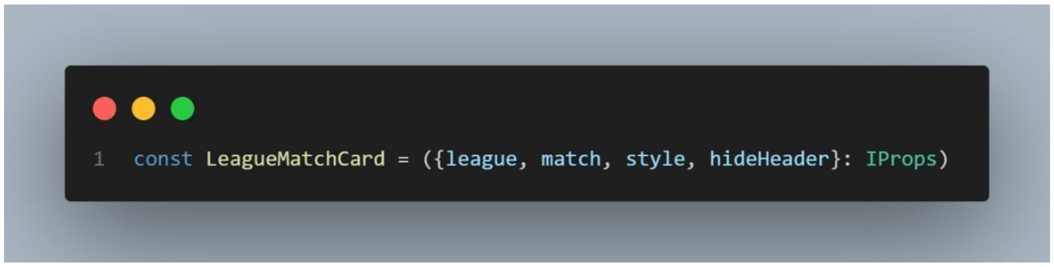
```
et teamHomeRankName = arrRankName.includes(match.homeTeam?.rankName) ? `${t(match.homeTeam?.rankName)}` : ''
  let teamAwayRankName = arrRankName.includes(match.awayTeam?.rankName) ? `${t(match.awayTeam?.rankName)}` : ''

  let teamHomeName =
    arrName.includes(match.homeTeam?.name) || !match.homeTeam?.id ? `${t(match.homeTeam?.name)}` : match.homeTeam?.name
  let teamAwayName =
    arrName.includes(match.awayTeam?.name) || !match.awayTeam?.id ? `${t(match.awayTeam?.name)}` : match.awayTeam?.name

```

Sample of render teams name of the match card



```
const {colors} = useTheme()
const navigation = useAppNavigation()
const styles = createStyles(hideHeader)

const navigate = (route: any, params?: object) => () => {
  navigation.navigate(route, params as never)
}

if (!match) return null
if (match.homeTeam?.name == 'hidden' || match.awayTeam?.name == 'hidden') return null
```

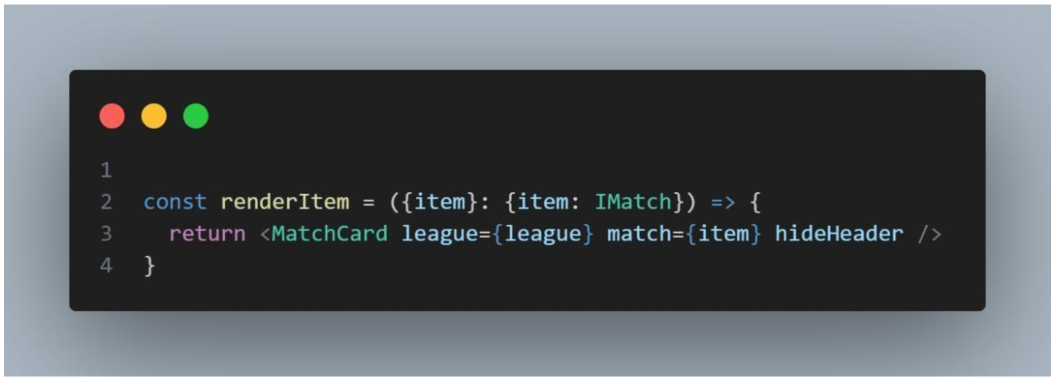
Using hook and navigation when user press



```
const LeagueMatchCard = ({league, match, style, hideHeader}: IProps)
```

Adding a global props to the components

These are the sample of what it can do with the custom components, and from there, I can called whether I want it need to be rendered

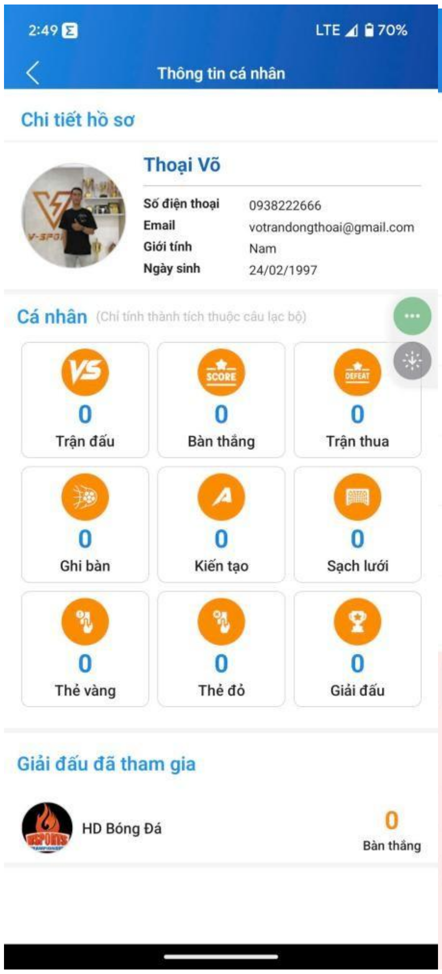


```

const renderItem = ({item}: {item: IMatch}) => {
  return <MatchCard league={league} match={item} hideHeader />
}
```

As for the method of **renderItem()** it will return a **MatchCard** that is represented for the components we’ve just used earlier. We’ll call the components and add the props needed for the page. The use for custom components really boils down to the user experience, while just sticking with the native library is enough, the more digging through the applications, the more I can find about the usability need to be elevated, so the need for custom components is a must for every screen needed, and by passing more properties to the applications components, reusing it with every other page take really less time to configure more

4. ***Adding and Update user data***

This application is working by driving the data, and most of the functions are created using the data. These are some examples on using the data as the main core function of the application. In this example is to update the user data, where field need to update is name, phone number, email address, birthday and so much more. By updating these field, it will directly to update down further to the as working with the voting system, or simply creating the league for future game.


2:49 LTE 70%

Thông tin cá nhân

Chi tiết hồ sơ

Thoại Võ

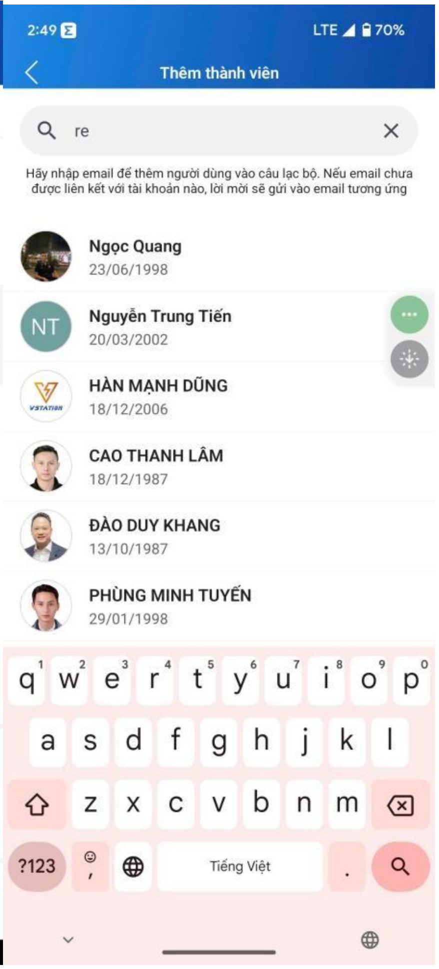
Số điện thoại	0938222666
Email	votrandongthoai@gmail.com
Giới tính	Nam
Ngày sinh	24/02/1997

Cá nhân (Chỉ tính thành tích thuộc câu lạc bộ)

0 Trận đấu	0 Bàn thắng	0 Trận thua
0 Ghi bàn	0 Kiến tạo	0 Sạch lưới
0 Thẻ vàng	0 Thẻ đỏ	0 Giải đấu

Giải đấu đã tham gia

HD Bóng Đá — 0 Bàn thắng

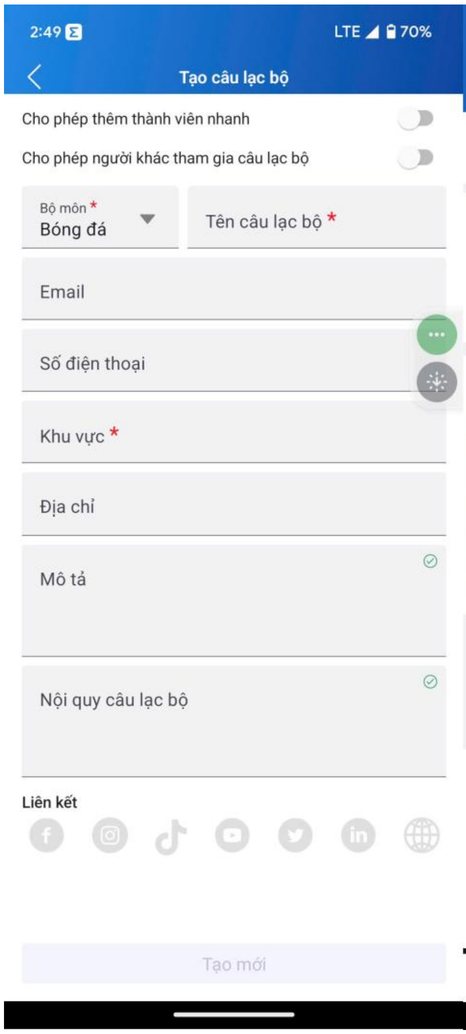

2:49 LTE 70%

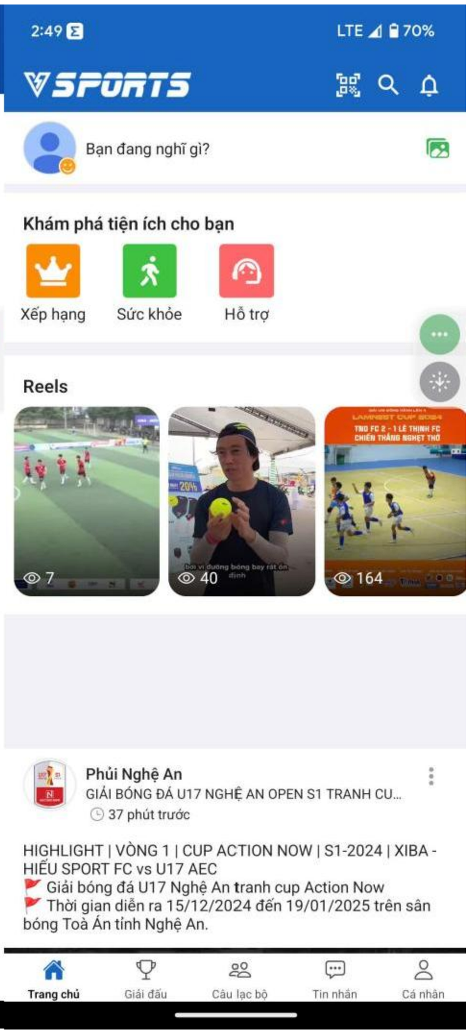
Thêm thành viên

re

Hãy nhập email để thêm người dùng vào câu lạc bộ. Nếu email chưa được liên kết với tài khoản nào, lời mời sẽ gửi vào email tương ứng

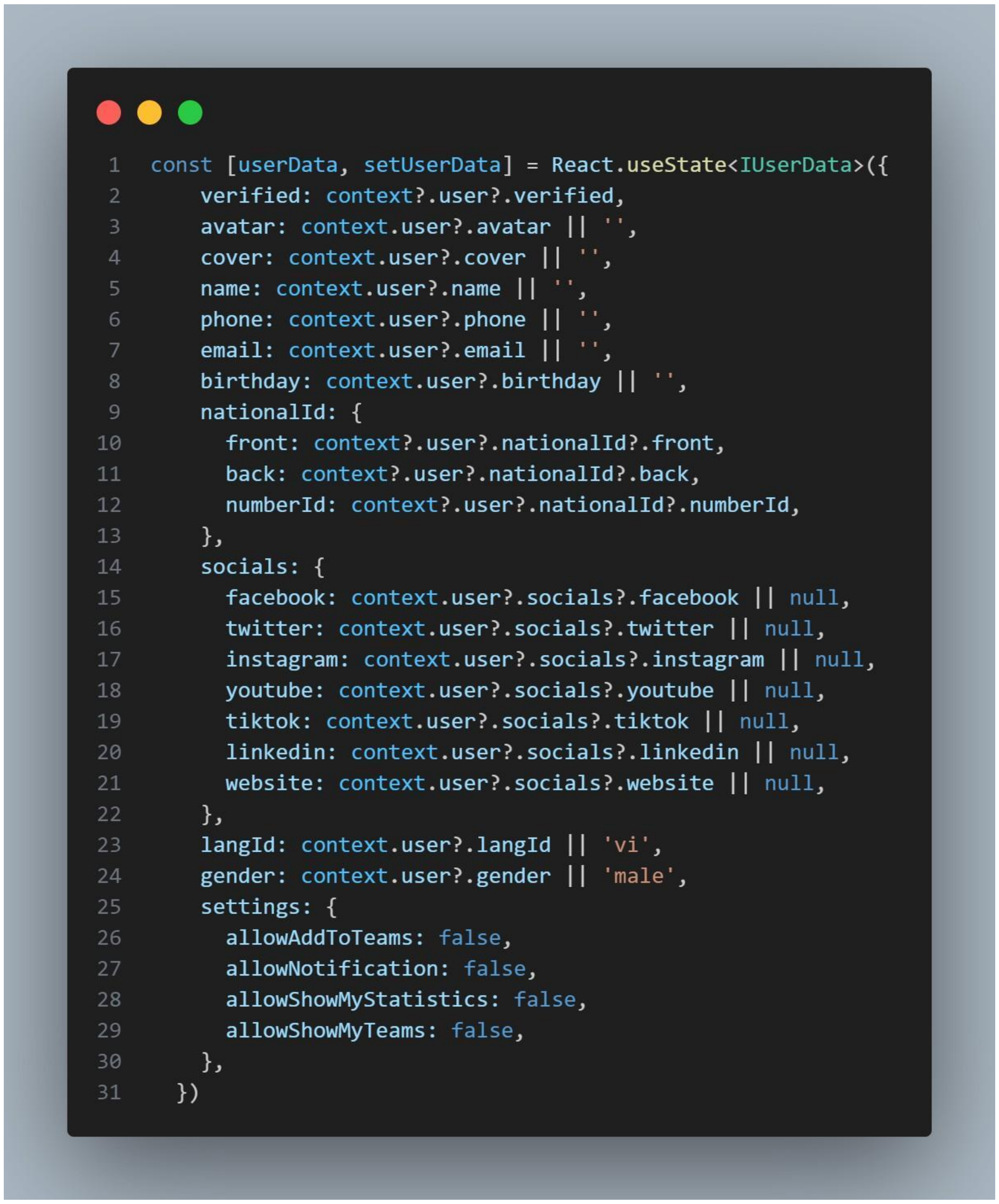
Ngọc Quang	23/06/1998
Nguyễn Trung Tiến	20/03/2002
HÀN MẠNH DŨNG	18/12/2006
CAO THANH LÂM	18/12/1987
ĐÀO DUY KHANG	13/10/1987
PHÙNG MINH TUYẾN	29/01/1998





HIGHLIGHT | VÒNG 1 | CUP ACTION NOW | S1-2024 | XIBA - HIẾU SPORT FC vs U17 AEC
🚩 Giải bóng đá U17 Nghệ An tranh cup Action Now
🚩 Thời gian diễn ra 15/12/2024 đến 19/01/2025 trên sân bóng Toà Án tỉnh Nghệ An.

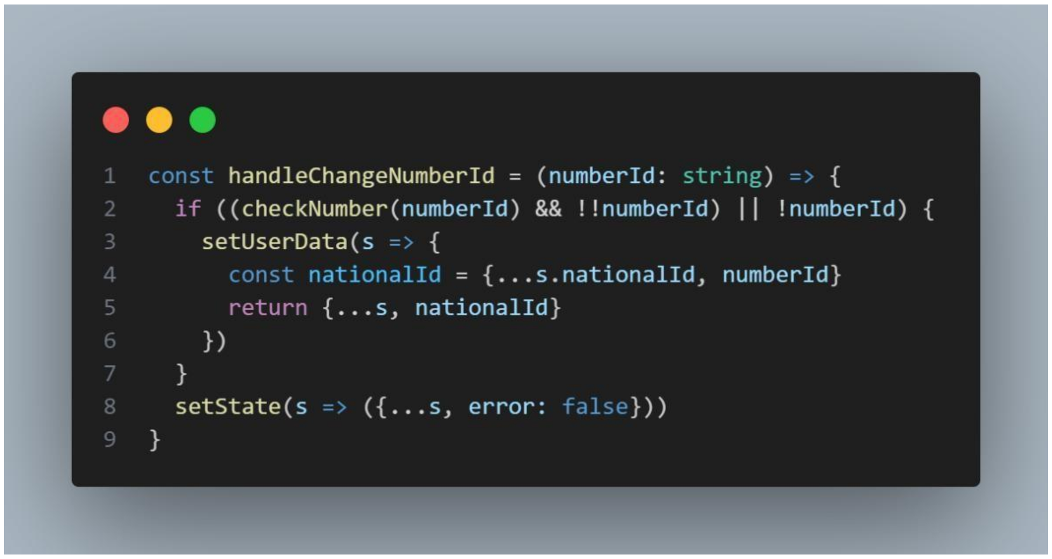
These are the examples of creating an application with data feeding to the application, there are some that need to be input correctly in order to complete the form of updating, because it is integrated deeply with the API, the form of data needs to be carefully input in. This can be handle with a simple error checking throughout the input field, making it a checking for every update down to very last input, even further, to check if the input is correct, uploading to the API will need for further checking, for example if the user data is correctly updated, but the data is already in the database, the API will spit an error, and Front-end must catch those event to update and show the error. For example of updating user information



```tsx
const [userData, setUserData] = React.useState<IUserData>({
    verified: context?.user?.verified,
    avatar: context.user?.avatar || '',
    cover: context.user?.cover || '',
    name: context.user?.name || '',
    phone: context.user?.phone || '',
    email: context.user?.email || '',
    birthday: context.user?.birthday || '',
    nationalId: {
      front: context?.user?.nationalId?.front,
      back: context?.user?.nationalId?.back,
      numberId: context?.user?.nationalId?.numberId,
    },
    socials: {
      facebook: context.user?.socials?.facebook || null,
      twitter: context.user?.socials?.twitter || null,
      instagram: context.user?.socials?.instagram || null,
      youtube: context.user?.socials?.youtube || null,
      tiktok: context.user?.socials?.tiktok || null,
      linkedin: context.user?.socials?.linkedin || null,
      website: context.user?.socials?.website || null,
    },
    langId: context.user?.langId || 'vi',
    gender: context.user?.gender || 'male',
    settings: {
      allowAddToTeams: false,
      allowNotification: false,
      allowShowMyStatistics: false,
      allowShowMyTeams: false,
    },
  })
```

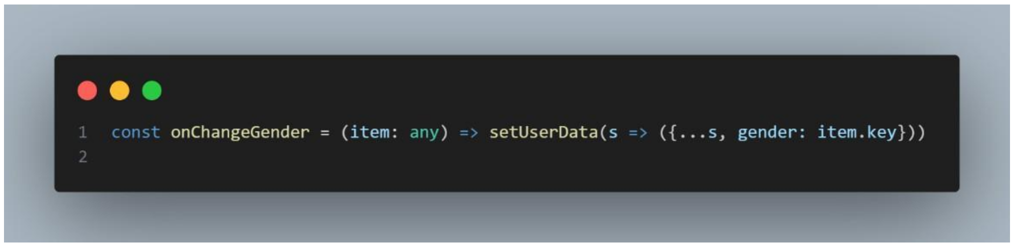
Image: Example of body of the input user

By setting a state to hold all of the data input in, we can verify each user information, on how to update, which type are input in and which is the correct one



```
const handleChangeNumberId = (numberId: string) => {
  if ((checkNumber(numberId) && !!numberId) || !numberId) {
    setUserData(s => {
      const nationalId = {...s.nationalId, numberId}
      return {...s, nationalId}
    })
  }
  setState(s => ({...s, error: false}))
}
```

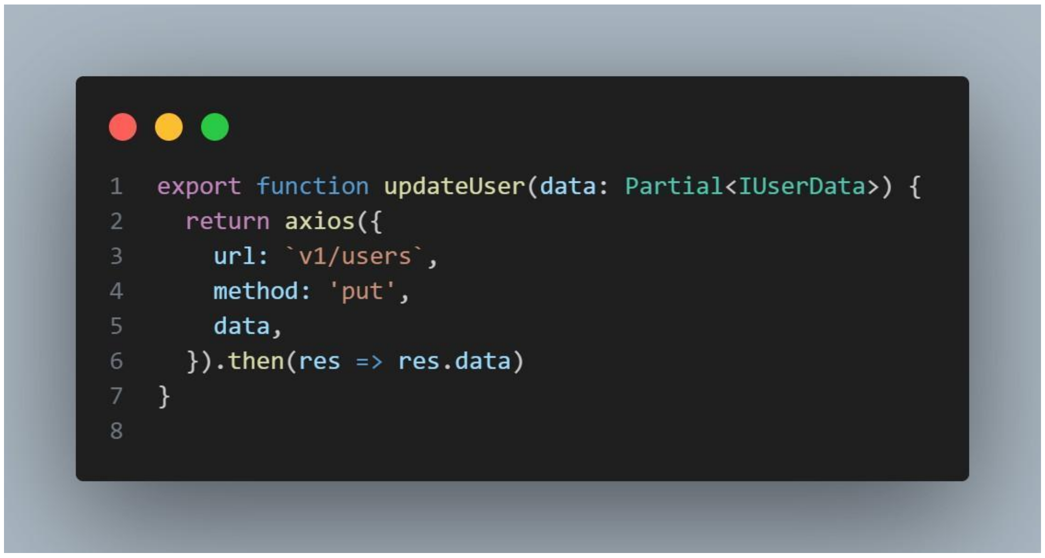
Image: Example of handling change numberId



```
const onChangeGender = (item: any) => setUserData(s => ({...s, gender: item.key}))

```

Image: Example of handling change gender

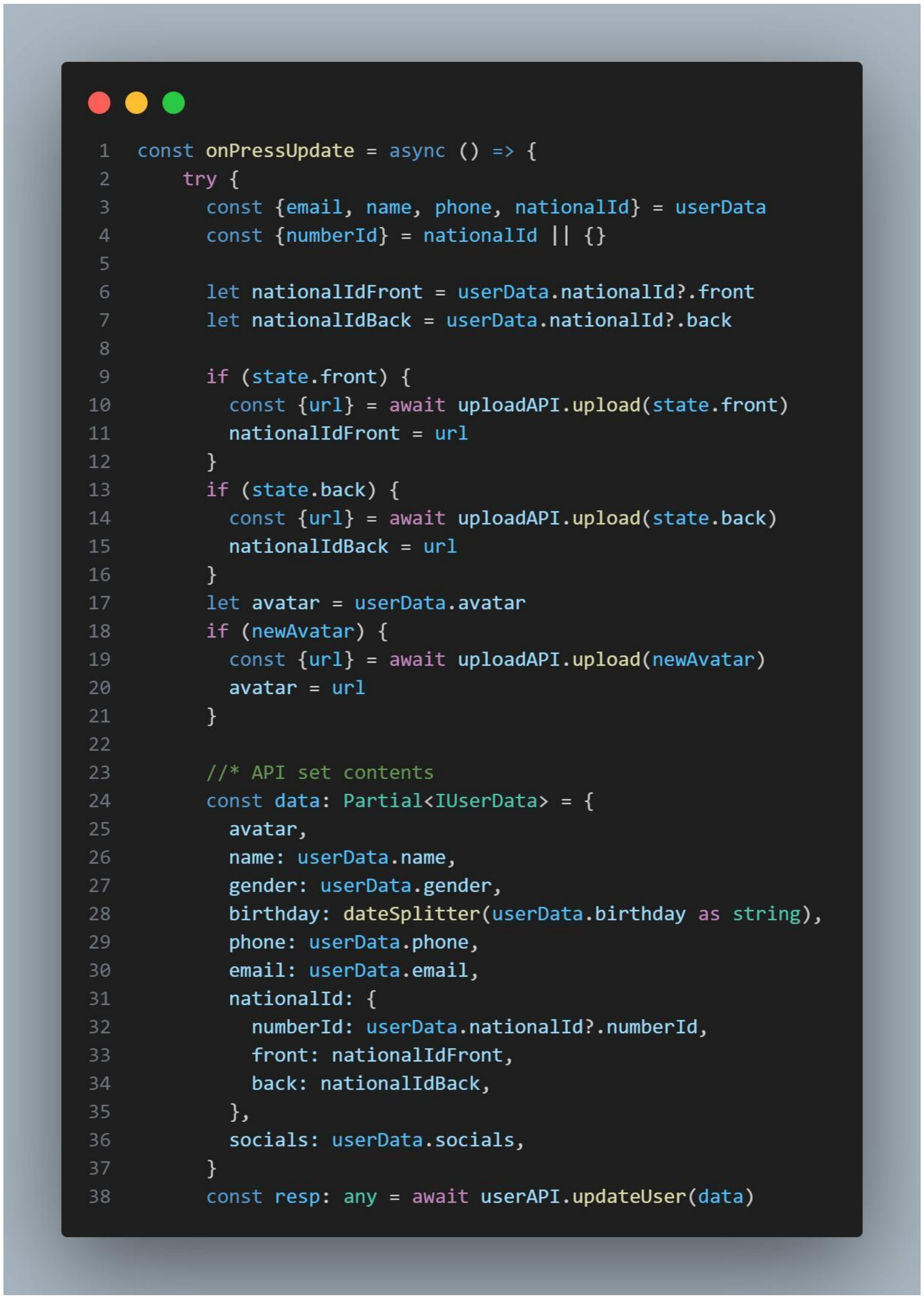
These are some of the validations, and because these are update to the API as well



```
export function updateUser(data: Partial<IUserData>) {
  return axios({
    url: `v1/users`,
    method: 'put',
    data,
  }).then(res => res.data)
}

```

This is the sample of an API with a PUT method.

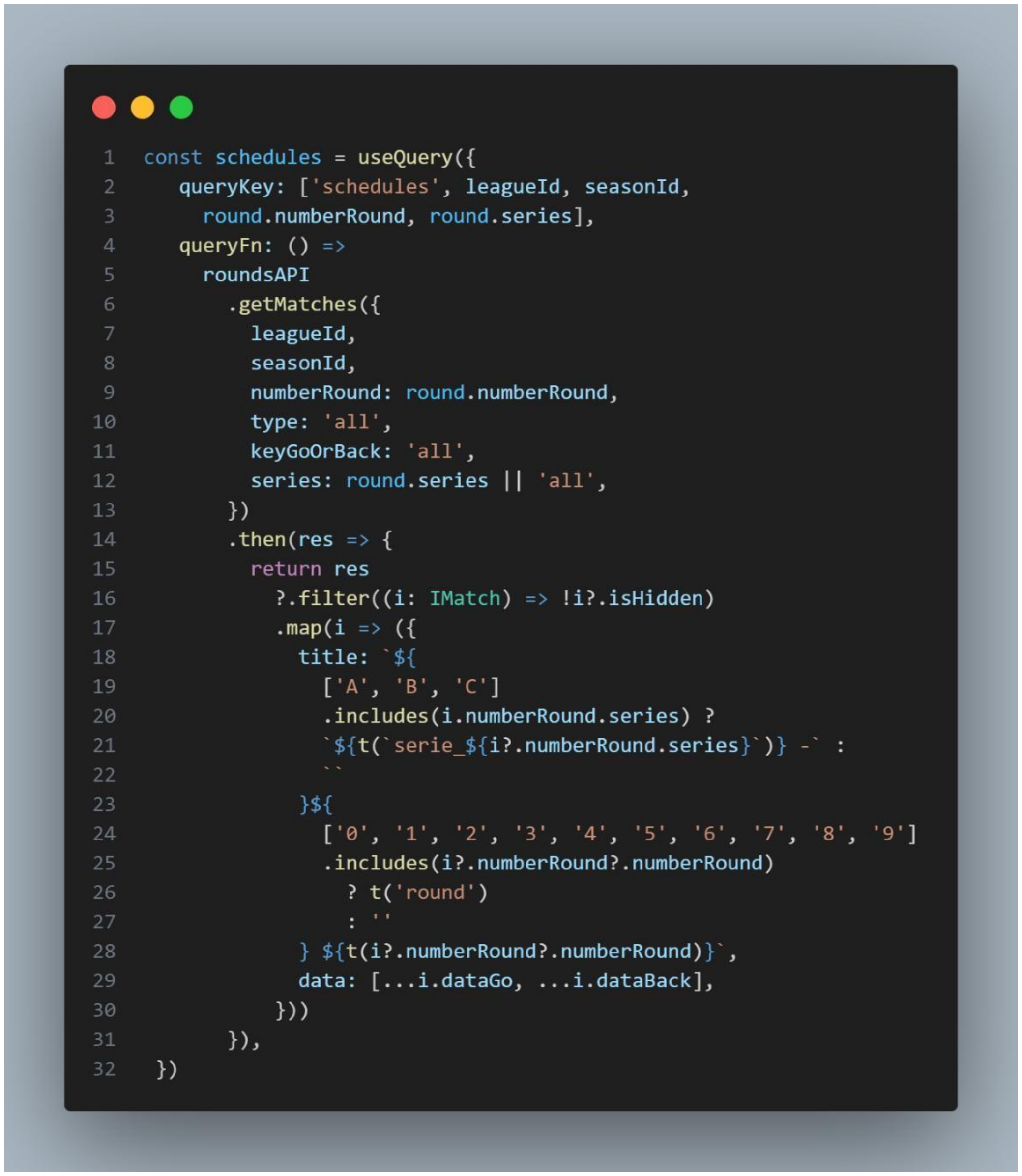
As with the API update, on the front-end side, I can work my way with handling by calling the API to handle the uploading



```
const onPressUpdate = async () => {
    try {
      const {email, name, phone, nationalId} = userData
      const {numberId} = nationalId || {}

      let nationalIdFront = userData.nationalId?.front
      let nationalIdBack = userData.nationalId?.back

      if (state.front) {
        const {url} = await uploadAPI.upload(state.front)
        nationalIdFront = url
      }
      if (state.back) {
        const {url} = await uploadAPI.upload(state.back)
        nationalIdBack = url
      }
      let avatar = userData.avatar
      if (newAvatar) {
        const {url} = await uploadAPI.upload(newAvatar)
        avatar = url
      }

      //* API set contents
      const data: Partial<IUserData> = {
        avatar,
        name: userData.name,
        gender: userData.gender,
        birthday: dateSplitter(userData.birthday as string),
        phone: userData.phone,
        email: userData.email,
        nationalId: {
          numberId: userData.nationalId?.numberId,
          front: nationalIdFront,
          back: nationalIdBack,
        },
        socials: userData.socials,
      }
      const resp: any = await userAPI.updateUser(data)
```

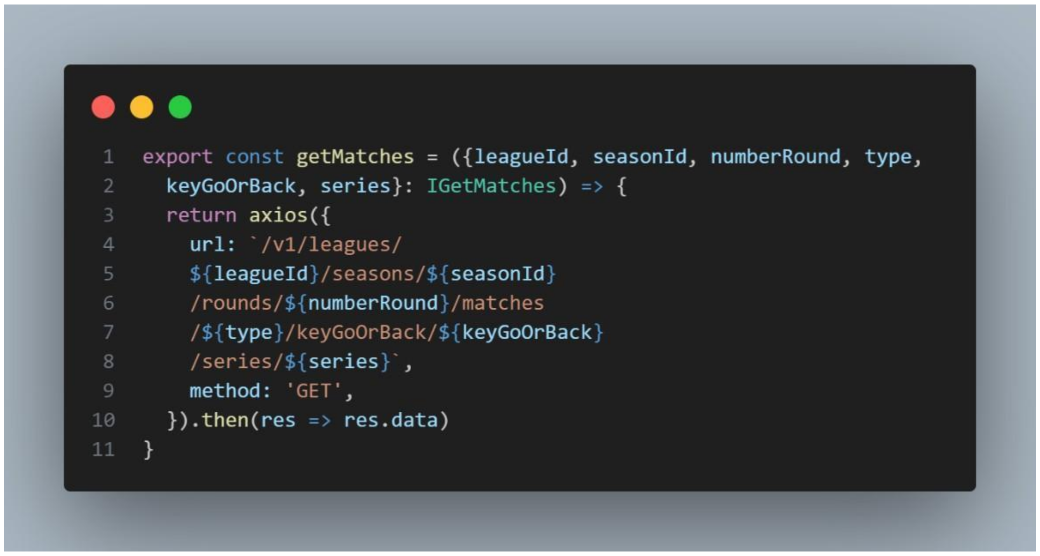
This line of code responsible for update the user, while checking the input at the same time. This to ensure all the data are validated until the data updated to the server

Another example is to create a list of all the schedules such as this



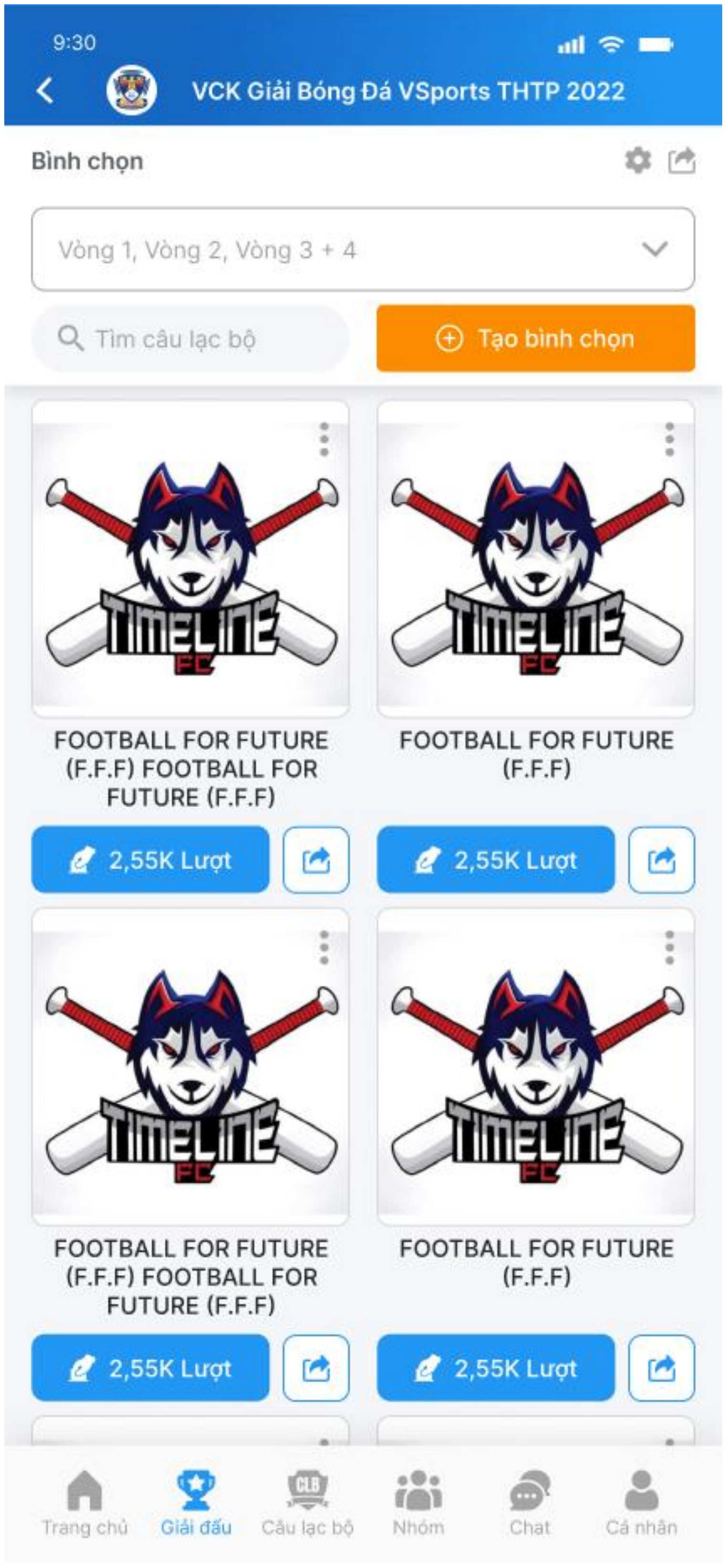
```
const schedules = useQuery({
   queryKey: ['schedules', leagueId, seasonId,
     round.numberRound, round.series],
   queryFn: () =>
     roundsAPI
       .getMatches({
         leagueId,
         seasonId,
         numberRound: round.numberRound,
         type: 'all',
         keyGoOrBack: 'all',
         series: round.series || 'all',
       })
       .then(res => {
         return res
           ?.filter((i: IMatch) => !i?.isHidden)
           .map(i => ({
             title: `${
               ['A', 'B', 'C']
               .includes(i.numberRound.series) ?
               `${t(`serie_${i?.numberRound.series}`)} -` :
               ``
             }${
               ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
               .includes(i?.numberRound?.numberRound)
                 ? t('round')
                 : ''
             } ${t(i?.numberRound?.numberRound)}`,
             data: [...i.dataGo, ...i.dataBack],
           }))
       }),
 })
```

This code handle the query of the match, by called to API, and set it to update, fetching the data from the API, and map to state of React Native.



```
export const getMatches = ({leagueId, seasonId, numberRound, type,
  keyGoOrBack, series}: IGetMatches) => {
  return axios({
    url: `/v1/leagues/
    ${leagueId}/seasons/${seasonId}
    /rounds/${numberRound}/matches
    /${type}/keyGoOrBack/${keyGoOrBack}
    /series/${series}`,
    method: 'GET',
  }).then(res => res.data)
}
```

This API is responsible for getting information from the API, with some parameters to get it specific to our needs. As the UI needed to list it out the matches as an Array and from the array, we will handle by map those array to get the matches to list of seeable one



9:30

VCK Giải Bóng Đá VSports THTP 2022

Bình chọn

Vòng 1, Vòng 2, Vòng 3 + 4

Tìm câu lạc bộ

Tạo bình chọn

FOOTBALL FOR FUTURE (F.F.F) FOOTBALL FOR FUTURE (F.F.F)

2,55K Lượt

FOOTBALL FOR FUTURE (F.F.F)

2,55K Lượt

FOOTBALL FOR FUTURE (F.F.F) FOOTBALL FOR FUTURE (F.F.F)

2,55K Lượt

FOOTBALL FOR FUTURE (F.F.F)

2,55K Lượt

Trang chủ

Giải đấu

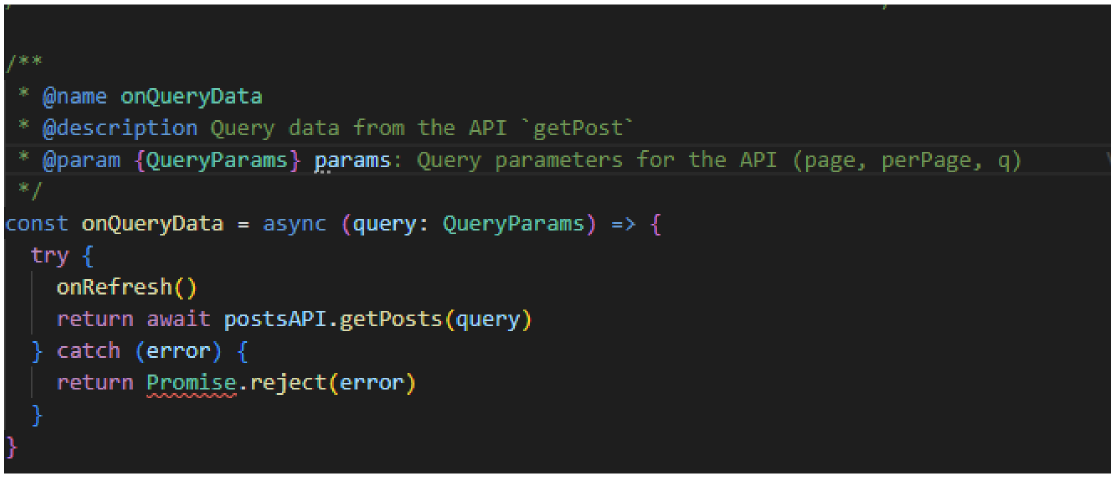
Câu lạc bộ

Nhóm

Chat

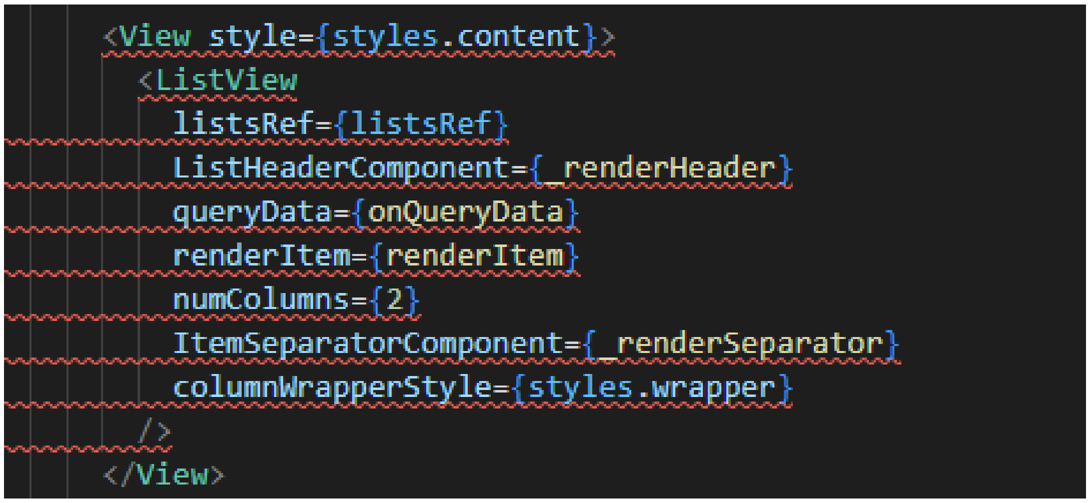
Cá nhân

The voting system is to be a place to pick all the teams that will advance to the specific league. The goal is to build a system simple for all the managers to do, manage all the voting numbers, rankings, and edit that information for the user. If the voting is complete, it will be added to the notification about who is the winner of the specific league, by clients using the applications, and voting will be fair because it will only work once, a user cannot vote multiple times at the same time.



```
/**
 * @name onQueryData
 * @description Query data from the API `getPost`
 * @param {QueryParams} params: Query parameters for the API (page, perPage, q)
 */
const onQueryData = async (query: QueryParams) => {
  try {
    onRefresh()
    return await postsAPI.getPosts(query)
  } catch (error) {
    return Promise.reject(error)
  }
}
```

This is the sample code of creating a Vsports Voting system, getting the data from the API, and return as an array, will update from the List



```
<View style={styles.content}>
  <ListView
    listsRef={listsRef}
    ListHeaderComponent={_renderHeader}
    queryData={onQueryData}
    renderItem={renderItem}
    numColumns={2}
    ItemSeparatorComponent={_renderSeparator}
    columnWrapperStyle={styles.wrapper}
  />
</View>
```

This will result in the the UI we mentioned above

IV. SUMMARY

Overall, after spending several months working on this project. I think it is practical for me to learn most of the project, learning about how the application works, learning how the community works and with the power framework, I can create such a unique application and unique ways to solve the problem.

With massive thanks on my instructor, Mr. Duong Huu Phuc as my direct instructor and my direct supervisor, Mr. Vo Tran Dong Thoai, for giving me the opportunity at the company to finish this internship report, this report is possible with the help of every person in team of Vsports as well, from my manager, Mr. Pham Le Phong, Mr. Le Trung Hieu, Mr. Nguyen Vinh Hien, and all the other members of Vsports, guiding, and aid in every situation.

Working on this environment help me get a summarize on several things:

1. Learning with teams, managing the code base is much harder than working in the school field

2. Coding with React Native, in such a way may be an easy choice, but learning throughout the years, helps me realize on how hard to managing the state, the controls of how it's rendering on mobile than rendering on the website

3. Working ethics are such a fun experience, because of how small connection with co-workers can massively improve workflow.

Another aspect of working with Vsports is the work ethics onsite, while working with technical is a must for all the

Learning with React Native, helps me on how to create a good application and building a product, with little to no time, but with learning on building a team to work throughout the working days are such a valuable experience no school can comprehend.

REFERENCES

These are the references of where I find the resource to complete this project.

https://viblo.asia/p/usestate-trong-react-hook-gDVK2romKLj

https://viblo.asia/p/su-dung-react-native-hooks-Az45bDoVZxY

https://viblo.asia/p/vong-doi-cua-1-ung-dung-ios-Qbq5QMNJ5D8