AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
```

```
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

CardBase.js

```
import { Card, IconButton, Text } from "react-native-paper";
import { Share, View } from "react-native";
import { Clone } from "../common/classes";
import utils from "../common/utils";
import { useNavigation } from "@react-navigation/native";
import Flexh from "./Flexh";

const CardBase=({clone,color,children})=>{
  var navigation=useNavigation()
  function onClick(){
    console.log(clone.cloneid)

navigation.navigate("admin",{cloneid:clone.cloneid,simple:false})
```

```
  }
  return (
    <View style={{width:'100%',height:"90%"}}>
      <Card
        onPress={onClick}
        style={{backgroundColor:color,width:"100%",height:"100%"}}
      >
        <Card.Content
style={{height:"100%",flexDirection:"column",justifyContent:"spac
e-between"}}>
          <Text
style={{textAlign:"center",fontSize:utils.fontSize1}}>{clone.name
}</Text>
          <Flexh>
            {children}
            <IconButton
              icon={"qrcode"}

onPress={()=>utils.navigation.navigate("share",{cloneid:clone.clo
neid})}
            ></IconButton>
          </Flexh>
        </Card.Content>
      </Card>
    </View>
  )
}
CardBase.defaultProps={
  clone:new Clone(),
  color:"white",
  children:{}
}

export default CardBase
```

CardFS.js

```
import CardBase from "./CardBase";
import ChipFS from "./ChipFS";
import { Clone } from "../common/classes";

const CardFS=({clone})=>{
  return(
    <CardBase
```

```
      clone={clone}
      color="#ffeeddff"
      >
      <ChipFS/>
    </CardBase>
  )
}

CardFS.defaultProps={
  clone:new Clone()
}
export default CardFS
```

CardKF.js

```
import { Clone } from "../common/classes";
import CardBase from "./CardBase";
import ChipKF from "./ChipKF";

const CardKF=({clone})=>{
  return(
    <CardBase
      clone={clone}
      color="#eeffee"
      >
      <ChipKF/>
    </CardBase>
  )
}

CardKF.defaultProps={
  clone:new Clone()
}
export default CardKF
```

CardShell.js

```
import CardSMS from "./CardSMS";
import CardFS from "./CardFS";
import CardKF from "./CardKF";
import { Clone } from "../common/classes";

const CardShell=({clone,type})=>{
  switch (type) {
    case "说明书":
      return <CardSMS clone={clone}/>
      break;
    case "客服":
      return <CardKF clone={clone}/>
      break;
    case "分身":
      return <CardFS clone={clone}/>
      break;
    default:
      return <CardSMS clone={clone}/>
  }
}

CardShell.defaultProps={
  clone:new Clone(),
  type:""
}
export default CardShell
```

CardSMS.js

```
import { View } from "react-native";
import CardBase from "./CardBase";
import ChipSMS from "./ChipSMS";
import { Clone } from "../common/classes";

const CardSMS=({clone})=>{
  return(
    <CardBase
      clone={clone}
      color="#ddeeffff"
      >
      <ChipSMS/>
    </CardBase>
  )
}

CardSMS.defaultProps={
  clone:new Clone()
}
export default CardSMS
```

ChipBase.js

```
import { useEffect, useState } from "react";
import { View,Dimensions } from "react-native";
import { Chip } from "react-native-paper";
import utils from "../common/utils";

const ChipBase=({type,color})=>{
  const [width,setWidth]=useState(0)
  useEffect(()=>{
    console.log("mounted")
    var _width=Dimensions.get('window').width
    setWidth(_width/5.2)
  },[])

  return (
    <View style={{ flexDirection: 'row', justifyContent: 'center',
alignItems: 'center' }}>
      <Chip
        textStyle={{fontSize:12,color:'white',textAlign:"center"}}
        style={{backgroundColor:utils.Colors[type],}}
        onPress={()=>{console.log(type)}}
      >{type}</Chip>
    </View>
  )
}
ChipBase.defaultProps={
  type:'',
  color:'',
}
export default ChipBase
```

ChipFS.js

```
import ChipBase from "./ChipBase";
import utils from "../common/utils";
const ChipFS=()=>{
  return(
    <ChipBase type={'分身'}
      color={utils.Colors.FS}
    />
  )
}
export default ChipFS
```

ChipKF.js

```
import ChipBase from "./ChipBase";
import utils from "../common/utils";

const ChipKF=()=>{
  return(
    <ChipBase type={'客服'}
      color={utils.Colors.KF}
    />
  )
}
export default ChipKF
```

ChipSMS.js

```
import ChipBase from "./ChipBase";
import utils from "../common/utils";

const ChipSMS=()=>{
  return(
    <ChipBase type={'说明书'}
      color={utils.Colors.SMS}
    />
  )
}
export default ChipSMS
```

ComEmpty.js

```
import { Icon } from "react-native-paper";

export default function ComEmpty(){
  return(
    <Icon source={"border-none-variant"} size={36}></Icon>
  )
}
```

ComQR.js

```
import React from 'react';
import { View, StyleSheet } from 'react-native';
import QRCode from 'react-native-qrcode-svg';
```

```
export default function ComQR() {
  return (
    <View style={styles.container}>
      <QRCode
        value="https://example.com"  // 这里是你想转换成二维码的信息，
例如网址、文本等。
        size={200}                   // 二维码的大小，单位是像素。
        color="black"                // 二维码的颜色。
        backgroundColor="white"      // 二维码的背景色。
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
});
```

ComSearch.js

```
import * as React from 'react';
import { Searchbar, Text } from 'react-native-paper';
import { StyleSheet } from 'react-native';
import { View } from 'react-native';
import utils from '../common/utils';

export default function ComSearch(){
  const [searchQuery, setSearchQuery] = React.useState('');

  const onChangeSearch = query => setSearchQuery(query);
  const theme=utils.theme
  function toUse(){
    var id=""
    console.log(searchQuery.length)
    if(searchQuery.length==24){
      id=searchQuery
    }else{
      id=searchQuery.slice(-24)
    }
    utils.navigation.navigate("use",{cloneid:id})
  }
  return (
    <View style={styles.centered}>
      <Searchbar
        placeholder="说明书 id"
        onChangeText={onChangeSearch}
        value={searchQuery}
        style={{ flex:8}}
        elevation={1}
        trailingIconColor="red"
        onSubmitEditing={toUse}
        theme={{ colors: { primary:
theme.colors.primary,backgroundColor:"red"} }}
      />
      <Text
        style={{flex:1,textAlign:"center"}}
        onPress={toUse}
      >跳转</Text>
    </View>
  );
}
const styles=StyleSheet.create({
  centered: {
    width:'100%',
    flexDirection:'row',
    justifyContent:'space-between',
    alignItems:"center",
    alignContent:"center",
    zIndex:200,
    position:"absolute",
    top:0
  }
})
```

ComSpacer.js

```
import { StyleSheet, View } from "react-native"

const ComSpacer=({height})=>{
  return (
    <View style={{height:height,width:'100%'}}>
    </View>
  )
}
ComSpacer.defaultProps={
  height:0
}
export default ComSpacer
```

ComState.js

```
import { View } from "react-native"
import { Card, Divider, List, Text } from "react-native-paper"
import { Clone } from "../common/classes"
import { useEffect } from "react"

const ComState=({clone,messages})=>{
  return(
    <Card style={{width:"100%"}}>
      <Card.Title title={clone.name}/>
      <Card.Content>
          {messages.map((item,index)=>
            <View key={index}>
              {index%2==1 && <Text>{item}</Text>}
              {index%2==0 && <Text
style={{color:"gray"}}>{item}</Text>}
              <Divider ></Divider>
            </View>
          )}
      </Card.Content>
    </Card>
  )
}

ComState.defaultProps={
  clone:new Clone(),
  messages:[]
}

export default ComState
```

ComTitle.js

```
import { StyleSheet } from "react-native";
import { Text } from "react-native-paper";
```

```
const ComTitle=({name})=>{
  return (
    <Text style={styles.s0}>
      {name}
    </Text>
  )
}

ComTitle.defaultProps={
  name:""
}

const styles=StyleSheet.create({
  s0:{
    position:'absolute',
    fontSize:60,
    color:"#00000022",
    width:'100%',
    height:'100%',
    textAlign:'center'
  }
})

export default ComTitle
```

FIlePicker.js

```
import React from 'react';
import { Button, View, Text } from 'react-native';
import * as DocumentPicker from 'expo-document-picker';

const FileUploader = ({setFileUri}) => {
    const pickDocument = async () => {
        let result = await DocumentPicker.getDocumentAsync({});
        var uri=(result.assets[0].uri)
        setFileUri(uri);
        console.log(uri)
        console.log(result.assets[0])
    };
    return (
        <View>
            <Button title="Pick a document" onPress={pickDocument}
/>
```

```
        </View>
    );
};

export default FileUploader;
```

Flexh.js

```
import { StyleSheet } from "react-native";
import { View } from "react-native";

export default Flexh=({children})=>{
  return <View style={styles.centered}>{children}</View>
}
const styles=StyleSheet.create({
  centered: {
    width:'100%',
    flexDirection:'row',
    justifyContent:'space-between',
    alignItems:"center",
    alignContent:"center",
    zIndex:200
  }
})
```

Flexh2.js

```
import { StyleSheet } from "react-native";
import { View } from "react-native";

export default Flexh2=({children})=>{
  return <View style={styles.centered}>{children}</View>
}
const styles=StyleSheet.create({
  centered: {
    width:'100%',
    flexDirection:'row',
    alignItems:'center'
  }
})
```

Flexv.js

```
import { StyleSheet } from "react-native";
import { View } from "react-native";

const Flexv=({children})=>{
  return <View style={styles.centered}>{children}</View>
}
const styles=StyleSheet.create({
  centered: {
    width:'100%',
    height:'100%',
    flexDirection:'column',
    justifyContent: 'center',
    alignItems: 'center',
  }
})

export default Flexv
```

NotiBase.js

```
import { View } from "react-native"
import { Snackbar, Text } from "react-native-paper"
import { useAuth } from "../common/AuthContext"

export const
NotiBase=({visible,message,color,textColor="white",icon,iconColor
="white"})=>{
  const {state,dispatch}=useAuth()

  return(
    <View
style={{width:"100%",top:0,position:"absolute",zIndex:1000}}>
      <Snackbar
        icon="home"
        visible={visible}
        onDismiss={()=>{dispatch({type:"NOTIHIDE"})}}
        style={{backgroundColor:color,position:"absolute"}}
        duration={1000}
        action={{
          icon:icon,
          textColor:iconColor
        }}>
          <Text style={{color:textColor}}>
```

```
            {message}
          </Text>
      </Snackbar>
    </View>
  )
}
```

NotiFail.js

```
import { useAuth } from "../common/AuthContext"
import utils from "../common/utils"
const { NotiBase } = require("./NotiBase")

export const NotiFail=()=>{
  const {state,dispatch}=useAuth()

  return(
    <NotiBase
      message={state.notiMessage}
      visible={state.notiFail}
      icon={"alert"}
      color={utils.TipColor.Fail}
      iconColor="red"
    ></NotiBase>
  )
}
```

NotiInfo.js

```
import utils from "../common/utils"
import { useAuth } from "../common/AuthContext"
const { NotiBase } = require("./NotiBase")

export const NotiInfo=()=>{
  const {state,dispatch}=useAuth()

  return(
    <NotiBase
      message={state.notiMessage}
      visible={state.notiInfo}
      icon={"lightbulb-outline"}
      color={utils.TipColor.Info}
    ></NotiBase>
  )
}
```

NotiSuccess.js

```
import { useAuth } from "../common/AuthContext"
import utils from "../common/utils"
const { NotiBase } = require("./NotiBase")

export const NotiSuccess=()=>{
  const {state,dispatch}=useAuth()

  return(
    <NotiBase
      message={state.notiMessage}
      visible={state.notiSuc}
      icon={"check-circle"}
      color={utils.TipColor.Success}
      textColor={"black"}
      iconColor="black"
    ></NotiBase>
  )
}
```

PageBase0.js

```
import { View } from "react-native";
import ComTitle from "./ComTitle";
import Flexv from "./Flexv";
import { useAuth } from "../common/AuthContext";

const PageBase0=({name,children,empty})=>{
  const store=useAuth()
  if(store.state.empty){
    return
  }
  else
  return (
    <View
style={{flexDirection:'column',width:'100%',height:'100%',justify
Content:'center',alignItems:'center',}}>
      <ComTitle name={name}></ComTitle>
      <View
style={{width:'75%',height:'95%',position:'relative'}}>
```

```
        <Flexv>
          {children}
        </Flexv>
      </View>
    </View>
  )
}

PageBase0.defaultProps={
  name:"",
  children:{},
  empty:false
}

export default PageBase0
```

PageBase01.js

```
import { View } from "react-native";
import ComTitle from "./ComTitle";
import Flexv from "./Flexv";
import { useEffect } from "react";
import utils from "../common/utils";
import { useAuth } from "../common/AuthContext";

const PageBase01=({name,children})=>{
  useEffect(()=>{
    // utils.dispatch({type:"HIDE"})
  },[])
  return (
    <View
style={{flexDirection:'column',width:'100%',height:'100%',justify
Content:'center',alignItems:'center',}}>
      <ComTitle name={name}></ComTitle>
      <View
style={{width:'95%',height:'95%',position:'relative'}}>
        <Flexv>
          {children}
        </Flexv>
      </View>
    </View>
  )
}
```

```
PageBase01.defaultProps={
  name:"",
  children:{},
}

export default PageBase01
```

PageBase1.js

```
import { View } from "react-native";
import { IconButton } from "react-native-paper";
import { useNavigation } from "@react-navigation/native";
import { useEffect, useState } from "react";
import { useAuth } from "../common/AuthContext";
import ComTitle from "./ComTitle";
import Flexv from "./Flexv";
import utils from "../common/utils";

const PageBase1=({children,name})=>{
  const store=useAuth()
  const navigation=useNavigation()
  useEffect(()=>{
    utils.dispatch({type:"HIDE"})
  },[])
  return !store.state.empty2&&(
    <View style={{width:'100%',height:'100%'}}>
      <View
style={{width:'100%',height:'100%',position:'absolute'}}>
        <IconButton icon='chevron-left' size={36}
          onPress={()=>{navigation.goBack()}}
          style={{left:0,zIndex:100,position:"absolute"}}>
        </IconButton>
        <View
style={{flexDirection:'column',width:'100%',height:'100%',justify
Content:'center',alignItems:'center',}}>
          <ComTitle name={name}></ComTitle>
          <View
style={{width:'75%',height:'95%',position:'relative'}}>
            <Flexv>
              {children}
            </Flexv>
          </View>
        </View>
      </View>
    </View>
  )
}
PageBase1.defaultProps={
  children:{},
  name:"",
}

export default PageBase1
```

PageBase2.js

```
import { View } from "react-native";
import { IconButton } from "react-native-paper";
// import { useNavigation } from "@react-navigation/native";
import ComTitle from "./ComTitle";
import Flexv from "./Flexv";
import { useEffect,useState } from "react";
import utils from "../common/utils";
import { useAuth } from "../common/AuthContext";

export default function PageBase2({children,name}){
  const store=useAuth()
  useEffect(()=>{
    utils.dispatch({type:"HIDE"})
  },[])
  return !store.state.empty2&&(
    <View style={{width:'100%',height:'100%'}}>
      <View
style={{width:'100%',height:'100%',position:'absolute'}}>
        <IconButton icon='chevron-left' size={36}
          onPress={()=>{utils.navigation.goBack()}}

style={{left:0,zIndex:100,position:"absolute"}}></IconButton>
          <View
style={{flexDirection:'column',width:'100%',height:'100%',justify
Content:'center',alignItems:'center',}}>
            <ComTitle name={name}></ComTitle>
            <View
style={{width:'100%',height:'100%',position:'relative'}}>
              <Flexv>
                {children}
              </Flexv>
            </View>
          </View>
      </View>
    </View>
  )
}
```

ProtocolBase.js

```
import { Text } from "react-native-paper"
import utils from "../common/utils"
import { View } from "react-native"

export default ProtocolBase=({name,routeName,logup})=>{
  return(
    <View>
      <Text style={{color:"blue",padding:0}}
        onPress={()=>{
          utils.dispatch({type:"HIDEDIALOG"})
          utils.navigation.navigate(routeName,{logup:logup})
        }}
      >{name}</Text>
    </View>
  )
}
ProtocolBase.defaultProps={
  logup:false
}
```

ProtocolPrivacy.js

```
import ProtocolBase from "./ProtocolBase"

export default ProtocalPrivacy=()=>{
  return(
    <ProtocolBase name={"隐私政策"}
routeName={"privacy"}></ProtocolBase>
  )
}
```

ProtocolService.js

```
import ProtocolBase from "./ProtocolBase"
```

```
export default ProtocalService=()=>{
  return(
    <ProtocolBase name={"服务条款"}
routeName={"service"}></ProtocolBase>
  )
}
```

RouteStack.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function RouteStack({route}) {
  const main=route.params.main
  return (
    <Stack.Navigator initialRouteName={main}>
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}
export default RouteStack
```

RouteStackBottom.js

```
import { createMaterialBottomTabNavigator } from 'react-native-
paper/react-navigation';
import { View } from 'react-native';
import PageAdmin from '../pages/PageAdmin';
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import { useAuth } from '../common/AuthContext';
import TabBarIcon from './TabBarIcon';

const Tab=createMaterialBottomTabNavigator()
export const RouteStackBottom=()=>{
  const {state,dispatch}=useAuth()
  return (
    <View style={{width:"100%",height:"100%"}}>
      <Tab.Navigator barStyle={{height:68,display:"none"}}
        screenOptions={{headerShow:false}}
      >
        <Tab.Screen name="创建" component={PageCreate}
          options={{
            tabBarLabel:"创建",
            tabBarIcon:({focused,color})=>{
              return <TabBarIcon name0={"plus-box"} name1={"plus-
box-outline"} focused={focused}></TabBarIcon>
            }
          }}
        />
        <Tab.Screen name="state" component={PageState}
          options={{
            tabBarLabel:"状态",
            tabBarIcon:({focused,color})=>{
              return <TabBarIcon name0={"information"}
name1={"information-outline"} focused={focused}></TabBarIcon>
            }
          }}
        />
        {!state.logined &&
          <Tab.Screen name="login" component={PageLogin}
initialParams={{positive:true}}
            options={{
              tabBarLabel:"登录",
              tabBarIcon:({focused,color})=>{
                return <TabBarIcon name0={"login-variant"}
name1={"login"} focused={focused}></TabBarIcon>
              }
            }}
          />
        }
        {state.logined &&
          <Tab.Screen name="桌面" component={PageDesktop}
            options={{
              tabBarLabel:"桌面",
              tabBarIcon:({focused,color})=>{
                return <TabBarIcon name0={"toolbox"}
name1={"toolbox-outline"} focused={focused}></TabBarIcon>
              }
            }}
          />
        }
      </Tab.Navigator>
    </View>
  );
};
```

RouteStackCreate.js

```
import PageCreate from '../pages/PageCreate';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
import PageMark from '../pages/PageMark';
import PageShare from '../pages/PageShare';
const Stack = createStackNavigator();

function RouteStackCreate() {
  return (
    <Stack.Navigator initialRouteName={"create"} screenOptions={{
      headerShown:false,
    }}>
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="mark" component={PageMark}/>
      <Stack.Screen name="admin" component={PageAdmin}/>
      <Stack.Screen name="use" component={PageUse} />
      <Stack.Screen name="share" component={PageShare} />
    </Stack.Navigator>
  );
}
export default RouteStackCreate
```

RouteStackDesktop.js

```
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import PageSetting from '../pages/PageSetting';
import { createStackNavigator } from '@react-navigation/stack';
import PageShare from '../pages/PageShare';

const Stack = createStackNavigator();
function RouteStackDesktop() {
  return (
    <Stack.Navigator initialRouteName={"_desktop"}
screenOptions={{
      headerShown:false
    }}>
      <Stack.Screen name="_desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
      <Stack.Screen name="setting" component={PageSetting} />
      <Stack.Screen name="share" component={PageShare} />
    </Stack.Navigator>
  );
}
export default RouteStackDesktop
```

RouteStackLogin.js

```
import PageLogin from '../pages/PageLogin';
import { createStackNavigator } from '@react-navigation/stack';
import PageProtocolService from '../pages/PageProtocolService';
import PageProtocolPrivacy from '../pages/PageProtocolPrivacy';
import PageProtocolAccount from '../pages/PageProtocolAccount';
const Stack = createStackNavigator();

function RouteStackLogin() {
  return (
    <Stack.Navigator initialRouteName={"_login"} screenOptions={{
      headerShown:false
    }}>
      <Stack.Screen name="_login" component={PageLogin}
initialParams={{positive:true}}/>
      <Stack.Screen name="service"
component={PageProtocolService}/>
      <Stack.Screen name="privacy"
component={PageProtocolPrivacy}/>
      <Stack.Screen name="account"
component={PageProtocolAccount}/>
    </Stack.Navigator>
  );
}
export default RouteStackLogin
```

RouteStackOther.js

```
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

export function RouteStackOther() {
  return (
    <Stack.Navigator initialRouteName={"admin"} screenOptions={{
      headerShown:false
    }}>
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}
```

TabBarIcon.js

```
const { Icon } = require("react-native-paper")

const TabBarIcon=({name0,name1,focused})=>{
  if(focused){
    return <Icon source={name0} size={24}></Icon>
  }
  else{
    return <Icon source={name1} size={24}></Icon>
  }
}
export default TabBarIcon
```

PageAdmin.js

```
import React, { useEffect, useState,useRef } from 'react';
import { View, Text,Dimensions } from 'react-native';
import { Button, TextInput } from 'react-native-paper';
import utils from '../common/utils';
import { Clone } from '../common/classes';
import Flexh from '../component/Flexh';
import PageBase1 from '../component/PageBase1';
import ComSpacer from '../component/ComSpacer';
import AsyncStorage from '@react-native-async-storage/async-
storage';
import { useNavigation } from '@react-navigation/native';
import Axios from '../common/Axios';
import { useAuth } from '../common/AuthContext';
import axios from 'axios';

function PageAdmin({route}) {
  var navigation=useNavigation()
  var {cloneid,simple}=route.params
  const [clone,setClone]=useState(new Clone())
  const [edit,setEdit]=useState(false)
  const [disable,setDisable]=useState(true)
  const [disablebtupdate,setDisablebtupdate]=useState(false)
  const {state,dispatch}=useAuth()
  const { width, height } = Dimensions.get('window');
  const maxLength=300

  useEffect(()=>{
    AsyncStorage.getItem(cloneid)
    .then(res=>{
      var clone=JSON.parse(res)
      setClone(new
Clone(clone.name,clone.type,clone.prompt,clone.adminid))
      console.log(clone)
    })
    .catch(res=>{
      axios.get(
        utils.url2+"adminclone/"+cloneid,
        {
          headers: {
            'Authorization': `Bearer ${utils.token}`
          }
        }
      )
      .then(res=>{
        var data=res.data
        console.log("adminid ")
        console.log(data)
        setClone(data)
      })
      .catch(res=>{
        console.log(res)
      })
    })
  },[])

  function Test(){
    navigation.navigate("use",{cloneid:cloneid})
  }

  function Update(){
    setDisablebtupdate(true)
    Axios.post(
      url=utils.url+"updateprompt",
      data={
        cloneid:cloneid,
        name:clone.name,
        prompt:clone.prompt,
      }
    )
    .then(res=>{
      var data=res.data
      console.log(data)
      if(data.success){
        AsyncStorage.setItem(cloneid,JSON.stringify(clone))
        dispatch({type:"SUCCESS",message:data.message})
      }else{
        dispatch({type:"FAIL",message:data.message})
      }
    })
    .catch(res=>{
      console.log(res)
    })
    .finally(res=>{
      setDisablebtupdate(false)
    })
  }
  return (
    <PageBase1 name="管理页">
      {(function(){
        if(simple)
          return
        if(edit){
          return <TextInput value={clone.name}
          mode='outlined'
          label="名称"
          placeholder={utils.AdminTextinputlabelClonename_sms}

style={{fontSize:utils.fontSize1,width:'100%',textAlign:'center'}}
          onChangeText={(value)=>{
            setClone(new
Clone(value,clone.type,clone.prompt,clone.adminid))
            setDisable(false)
          }}></TextInput>
        }
        else{
          return <Text onPress={()=>{setEdit(true)}}
style={{fontSize:utils.fontSize1}}>{clone.name}</Text>
        }
      })()}
      <ComSpacer height={20}></ComSpacer>
      <TextInput mode='outlined' value={clone.prompt}
        contentStyle={{maxHeight:height*0.4}}
        maxLength={maxLength}
        onChangeText={(value)=>{
          if(value.length>=maxLength){
            utils.dispatch({type:"FAIL",message:"字数超过限制,多余部
分自动删除"})
            value=value.slice(0,maxLength)
          }
          setClone(new
Clone(clone.name,clone.type,value,clone.adminid))
          setDisable(false)
        }}
        onFocus={()=>{setEdit(true)}}
        multiline
        label={"指令"}
        placeholder={utils.AdminTextinputlabelPrompt_sms}
        style={{width:'100%'}}
      ></TextInput>
      <ComSpacer height={20}></ComSpacer>
      <Flexh>
        <Button
          mode='contained-tonal'
          onPress={Update}
          disabled={disable||disablebtupdate}
          icon={"update"}
        >更新</Button>
        <Button
          mode='contained'
          onPress={Test}
          icon={"play-circle"}
        >测试</Button>
      </Flexh>
    </PageBase1>
  );
}

export default PageAdmin;
```

PageCreate.js

```
import { Text, Icon, IconButton } from 'react-native-paper';
import * as React from 'react';
import  PageBase01 from '../component/PageBase01';
import { View } from 'react-native';
import { useState } from 'react';
import { useAuth } from '../common/AuthContext';
import { useFocusEffect } from '@react-navigation/native';
import ComSearch from '../component/ComSearch';
import utils from '../common/utils';

function PageCreate({navigation}) {
  const store=useAuth()
  if(store.state.empty){
    return
  }
  console.log("pagecreate")
  const {state,dispatch}=useAuth()
  const [disablecreate,setDisablecreate]=useState(false)

  useFocusEffect(
    React.useCallback(()=>{
      dispatch({type:"SHOW"})
    },[])
  )

  function createClone(){
    setDisablecreate(true)
    dispatch({type:"SUCCESS",message:"创建成功"})
    navigation.navigate("mark",{cloneid:""})
    setDisablecreate(false)
  }
  return (
    <PageBase01 name={""}>
      <ComSearch></ComSearch>
      <View style={{flexDirection:'column',alignItems:"center"}}>
        <IconButton
          icon="plus-circle-outline"
          size={utils.fontSize5}
          onPress={createClone}
          disabled={disablecreate}
          iconColor={utils.theme.colors.primary}
          />
        <Text style={{textAlign:'center',fontSize:24}}>
          创建说明书
        </Text>
      </View>
    </PageBase01>
  )
};
export default PageCreate;
```

PageDesktop.js

```
import { useEffect, useState,useCallback } from "react"
import {  View } from "react-native"
import utils from "../common/utils"
import PageBase0 from "../component/PageBase0";
import CardShell from "../component/CardShell";
import AsyncStorage from "@react-native-async-storage/async-
storage";
import Flexh from "../component/Flexh";
import { Button, IconButton } from "react-native-paper";
import { useAuth } from "../common/AuthContext";
import { useFocusEffect } from "@react-navigation/native";
import Axios from "../common/Axios";
import { UserInfo } from "../common/classes";

const PageDesktop=({navigation})=>{
  const [clones,setClones]=useState([])
  const {state,dispatch}=useAuth()
  useFocusEffect(
    useCallback(()=>{
      dispatch({type:"SHOW"})
      AsyncStorage.getItem("token")
      .then(token=>{
        Axios.get(
          "/getuserinfo",
        )
        .then(res=>{
          var data=res.data
          console.log(data)
          setClones(data.clones)
          utils.userinfo=new
UserInfo(data.cloneids,data.clones,data.maxfscount)
          data.clones.forEach((item,index)=>{

AsyncStorage.setItem(item.cloneid,JSON.stringify(item))
          })

AsyncStorage.setItem("clones",JSON.stringify(data.clones))
        })
        .catch(res=>{
          console.log(res)
        })
      })
    },[])
  )
  return(
    <View style={{width:'100%',height:'100%'}}>
      <View
style={{width:'100%',height:'100%',position:'absolute'}}>
        <Flexh>
          <IconButton icon={"cog"} size={utils.fontSize2}
            style={{position:"absolute",top:0,left:0,zIndex:100}}
            onPress={()=>{utils.navigation.navigate("setting")}}
          />
          <IconButton icon={"logout"} size={utils.fontSize2}
            style={{position:"absolute",top:0,right:0,zIndex:100}}
            onPress={()=>{
              dispatch({type:"SUCCESS",message:"已退出登录"})
              dispatch({type:"LOGOUT"})
            }}
          />
        </Flexh>
        <PageBase0 name={"桌面页"}>
          {clones!=null && clones.map((item,index)=>(
            <View key={index} style={{width:"100%",height:"25%"}}>
              <CardShell clone={item} type={item.type}/>
            </View>
          ))}
          {(clones==null || clones.length==0) &&
            <Button mode="elevated" icon={"plus"}

labelStyle={{fontSize:utils.fontSize1,textAlign:"center"}}
              onPress={()=>navigation.navigate("create")}
            >创建</Button>
          }
        </PageBase0>
      </View>
    </View>
  )
}

export default PageDesktop
```

PageLogin.js

```
import { Button, Checkbox, Dialog, Portal, Text, TextInput } from
"react-native-paper";
import PageBase0 from "../component/PageBase0";
import Flexh from "../component/Flexh";
import Flexh2 from "../component/Flexh2";
import ComSpacer from "../component/ComSpacer";
import { useEffect, useState,useCallback } from "react";
import utils from "../common/utils";
import AsyncStorage from "@react-native-async-storage/async-
storage";
import { useAuth } from "../common/AuthContext";
import { useFocusEffect } from "@react-navigation/native";
import Axios from "../common/Axios";
import ProtocolService from "../component/ProtocolService";
import ProtocolPrivacy from "../component/ProtocolPrivacy";
import ProtocolBase from "../component/ProtocolBase";

const PageLogin=({navigation,route})=>{
  const [username,setUsername]=useState("")
  const [password,setPassword]=useState("")
  const {positive=true}=route.params
  const {state,dispatch}=useAuth()
  const [dislogin,setDislogin]=useState(false)
  const [dislogup,setDislogup]=useState(false)
  useFocusEffect(
    useCallback(()=>{
      dispatch({type:"SHOW"})
      utils.get401=false
    },[])
  )
  function toDesktop(){
    dispatch({type:"LOGIN"})
    navigation.navigate("desktop")
  }
  function logup(){
    setDislogup(true)
    var data={
      username:username,
      password:password
    }
    Axios.post(
      "/logup",
      data
    )
    .then(res=>{
      var data=res.data
      if(data.success){
        dispatch({type:"SUCCESS",message:data.message})
        if(positive)
          toDesktop()
        else
          navigation.goBack()
      }else{
        dispatch({type:"FAIL",message:data.message})
      }
    })
    .catch(res=>{
      console.log(res)
    })
    .finally(res=>{
      setDislogup(false)
    })
  }
  function login(){
    setDislogin(true)
    var data={
      username:username,
      password:password
    }
    Axios.post(
      "/login",
      data
    )
    .then(res=>{
      console.log(res)
      var data=res.data
      utils.token=data.token
      AsyncStorage.setItem("token",utils.token)
      AsyncStorage.setItem("maxfscount",data.maxfscount)
      console.log(data)
      if(data.success){
        dispatch({type:"SUCCESS",message:data.message})
        if(positive)
          toDesktop()
        else
          navigation.goBack()
      }else{
        dispatch({type:"FAIL",message:data.message})
      }
    })
    .catch(res=>{
      console.log(res.request)
    })
    .finally(res=>{
      setDislogin(false)
    })
  }
  return (
    <PageBase0 name={'登录页'}>
      <TextInput label="手机号/用户名"
onChangeText={(value)=>{setUsername(value)}} value={username}
mode="outlined" style={{width:'100%'}}></TextInput>
      <ComSpacer height={15}></ComSpacer>
      <TextInput label="密码" secureTextEntry
onChangeText={(value)=>{setPassword(value)}} value={password}
mode="outlined" style={{width:'100%'}}></TextInput>
      <ComSpacer height={15}></ComSpacer>
      <Flexh2>
        <Text>点击登录视为您已同意</Text>
        <ProtocolService></ProtocolService>
        <Text>和</Text>
        <ProtocolPrivacy></ProtocolPrivacy>
      </Flexh2>
      <ComSpacer height={15}></ComSpacer>
      <Flexh>
        <Button mode='elevated' icon='account-plus-outline'
          disabled={dislogup}
          onPress={()=>{
            console.log(`username$username`)
            if(username==""||password==""){
              utils.dispatch({type:"FAIL",message:"请填写用户名、密
码"})
              return
            }
            dispatch({type:"SHOWDIALOG"})
          }}
        >注册</Button>
        <Button mode='contained' icon='login'
        disabled={dislogin}
          onPress={login}
        >登录</Button>
      </Flexh>
      <Portal>
        <Dialog visible={state.showProtocolDialog}
onDismiss={()=>{dispatch({type:"HIDEDIALOG"})}}>
          <Dialog.Content>
            <Flexh2>
              <Text>你是否同意</Text>
              <ProtocolBase name={"服务协议"} routeName={"service"}
logup={true}/>
              <Text>、</Text>
              <ProtocolBase name={"隐私条款"} routeName={"privacy"}
logup={true}/>
              <Text>、</Text>
              <ProtocolBase name={"账号协议"} routeName={"account"}
logup={true}/>
              <Text>?</Text>
            </Flexh2>
          </Dialog.Content>
          <Dialog.Actions>
            <Button onPress={()=>{dispatch({type:"HIDEDIALOG"})}}>
不同意</Button>
            <Button onPress={()=>{
              dispatch({type:"HIDEDIALOG"})
              logup()
            }}>同意</Button>
          </Dialog.Actions>
        </Dialog>
      </Portal>
    </PageBase0>
  )
}
export default PageLogin
```

PageMain.js

```
import * as React from 'react';
import { View } from 'react-native';
import PageState from './PageState';
import { useNavigation } from '@react-navigation/native';
import { createMaterialBottomTabNavigator } from 'react-native-
paper/react-navigation';
import TabBarIcon from '../component/TabBarIcon';
import RouteStackCreate from '../component/RouteStackCreate';
import RouteStackDesktop from '../component/RouteStackDesktop';
import utils from '../common/utils';
import { useAuth } from '../common/AuthContext';
import { setupInterceptors } from '../common/Axios';
import AsyncStorage from '@react-native-async-storage/async-
storage';
import { NotiSuccess } from '../component/NotiSuccess';
import { NotiFail } from '../component/NotiFail';
import { NotiInfo } from '../component/NotiInfo';
import RouteStackLogin from '../component/RouteStackLogin';

const Tab=createMaterialBottomTabNavigator()
const PageMain = () => {
  utils.navigation=useNavigation()
  const {state,dispatch}=useAuth()
  utils.dispatch=dispatch
  const store=useAuth()
  setupInterceptors(store)
  React.useEffect(()=>{
    AsyncStorage.getItem("token")
    .then(token=>{
      console.log("token: "+token)
      utils.token=token
      if(token!=""&& token!=null){
        dispatch({type:'LOGIN'})
      }
      else{
        dispatch({type:"LOGOUT"})
      }
    })
  },[dispatch])

  return (
    // <View
style={{width:"100vw",height:"100vh",flexDirection:"column",justi
fyContent:"center",alignItems:"center"}}>
    // <View style={{width:"55vh",height:"90vh"}}>
    <View style={{width:"100%",height:"100%"}}>
    <View style={{width:"100%",height:"100%"}}>
      <NotiSuccess></NotiSuccess>
      <NotiFail></NotiFail>
      <NotiInfo></NotiInfo>
      <Tab.Navigator
barStyle={{height:70,display:store.state.display}}>
        <Tab.Screen name="创建" component={RouteStackCreate}
          options={{
            tabBarLabel:"创建",
            tabBarIcon:({focused,color})=>{
              return <TabBarIcon name0={"plus-circle"}
name1={"plus-circle-outline"} focused={focused}></TabBarIcon>
            }
          }}
        />
        <Tab.Screen name="state" component={PageState}
          options={{
            tabBarLabel:"状态",
            tabBarIcon:({focused,color})=>{
              return <TabBarIcon name0={"information"}
name1={"information-outline"} focused={focused}></TabBarIcon>
            }
          }}
        />
        {!state.logined &&
          // <Tab.Screen name="login" component={PageLogin}
initialParams={{positive:true}}
          <Tab.Screen name="login" component={RouteStackLogin}
initialParams={{positive:true}}
            options={{
              tabBarLabel:"登录",
              tabBarIcon:({focused,color})=>{
                return <TabBarIcon name0={"login-variant"}
name1={"login"} focused={focused}></TabBarIcon>
              }
            }}
          />
        }
        {state.logined &&
          <Tab.Screen name="desktop" component={RouteStackDesktop}
            options={{
              tabBarLabel:"桌面",
              tabBarIcon:({focused,color})=>{
                return <TabBarIcon name0={"toolbox"}
name1={"toolbox-outline"} focused={focused}></TabBarIcon>
              }
            }}
          />
        }
      </Tab.Navigator>
    </View>
    </View>
  );
};

export default PageMain;
```

PageMain2.js

```
import * as React from 'react';
import { View } from 'react-native';
import { useNavigation } from '@react-navigation/native';
import utils from '../common/utils';
import { useAuth } from '../common/AuthContext';
import { setupInterceptors } from '../common/Axios';
import AsyncStorage from '@react-native-async-storage/async-
storage';
import { createNativeStackNavigator } from '@react-
navigation/native-stack';
import { RouteStackBottom } from '../component/RouteStackBottom';
import { RouteStackOther } from '../component/RouteStackOther';

const RootStack=createNativeStackNavigator()
const PageMain = () => {
  utils.navigation=useNavigation()
  const {state,dispatch}=useAuth()
  const store=useAuth()
  setupInterceptors(store)
  React.useEffect(()=>{
    AsyncStorage.getItem("token")
    .then(token=>{
      console.log("token: "+token)
      utils.token=token
      if(token!=""&& token!=null){
        store.dispatch({type:'LOGIN'})
      }
      else{
        store.dispatch({type:"LOGOUT"})
      }
    })
  },[dispatch])

  return (
    <View style={{width:"100%",height:"100%"}}>
      <RootStack.Navigator>
        <RootStack.Screen name='bottom'
component={RouteStackBottom} options={{
          headerShown:false,
        }}/>
        <RootStack.Screen name="other"
component={RouteStackOther}/>
      </RootStack.Navigator>
    </View>
  );
};

export default PageMain;
```

PageMark.js

```
import React, { useEffect, useState,useRef } from 'react';
import { Button, TextInput } from 'react-native-paper';
import PageBase1 from '../component/PageBase1';
import Axios from '../common/Axios';
import { useAuth } from '../common/AuthContext';
import utils from '../common/utils';

export default function PageMark({navigation,route}) {
  var {cloneid}=route.params
  const {state,dispatch}=useAuth()
  const [dismark,setDismark]=useState(false)

  useEffect(()=>{
    console.log("intoMark")
  },[])

  function mark(){
    setDismark(true)
    var userinfo=utils.userinfo

if(utils.token!=""&&userinfo.maxfscount<=userinfo.cloneids.length
){
      dispatch({type:"FAIL",message:"已拥有,跳转至桌面"})
      utils.navigation.reset({
        index: 0,
        routes: [{
          name: 'desktop',
          params: { positive:false},
        }],
      });
      return
    }
    Axios.post("/markclone")
    .then(res=>{
      var data=res.data
      console.log(data)
      if(data.success){

navigation.replace("admin",{cloneid:data.cloneid,simple:true})
        dispatch({type:"SUCCESS",message:data.message})
      }else{
        dispatch({type:"FAIL",message:data.message})
      }
    })
    .catch(err=>{
      console.log(err.response)
    })
    .finally(res=>{
      setDismark(false)
    })
  }
  return (
    <PageBase1 name="标记页">
      <Button
```

```
        mode='outlined'
        icon={"star-plus-outline"}
        disabled={dismark}
        onPress={()=>{
          mark()
        }}
      >标记</Button>
    </PageBase1>
  )
}
```

PageProtocolAccount.js

```
import PageProtocolBase from "./PageProtocolBase"

const url="https://bxjs.store:8443/account"
export default function PageProtocolAccount({route}){
  return(
    <PageProtocolBase url={url}
logup={route.params.logup}></PageProtocolBase>
  )
}
```

PageProtocolBase.js

```
import { IconButton, Text } from "react-native-paper"
import WebView from "react-native-webview"
import { View } from "react-native"
import utils from "../common/utils"
import { useEffect } from "react"

export default PageProtocolBase=({url,logup})=>{
  useEffect(()=>{
    return ()=>{
      if(logup){
        utils.dispatch({type:"SHOWDIALOG"})
      }
    }
  },[])
  return(
    <View style={{height:"100%"}}>
      <IconButton icon='chevron-left' size={36}
        onPress={()=>{
          utils.dispatch({type:"SHOWDIALOG"})
          utils.navigation.goBack()
        }}

style={{left:0,zIndex:100,position:"absolute"}}></IconButton>
      <WebView
        source={{uri:url}}

style={{width:"100%",height:"100%",zIndex:100,position:"absolute"
}}
      />
    </View>
  )
}
PageProtocolBase.defaultProps={
  logup:false
}
```

PageProtocolPrivacy.js

```
import { useEffect } from "react"
import PageProtocolBase from "./PageProtocolBase"

const url="https://bxjs.store:8443/privacy"
export default PageProtocolPrivacy=({route})=>{
  useEffect(()=>{
    console.log(route.params.logup)
  },[])
  return(
    <PageProtocolBase url={url}
logup={route.params.logup}></PageProtocolBase>
  )
}
```

PageProtocolService.js

```
import PageProtocolBase from "./PageProtocolBase"

const url="https://bxjs.store:8443/service"
export default function PageProtocolService({route}){
  return(
    <PageProtocolBase url={url}
logup={route.params.logup}></PageProtocolBase>
  )
}
```

PageSetting.js

```
import { Card, Text } from "react-native-paper";
import PageBase1 from "../component/PageBase1";

export default function PageSetting(){
  return(
    <PageBase1 name={"设置页"}>
      <Card style={{width:"100%"}}>
        <Card.Content>
          <Text>客服电话:13324052902</Text>
        </Card.Content>
      </Card>
    </PageBase1>
  )
}
```

PageShare.js

```
import { Button, IconButton, Text } from "react-native-paper";
import PageBase2 from "../component/PageBase2";
import QRCode from "react-qr-code";
import * as Clipboard from 'expo-clipboard';
import config from "../common/config";
import utils from "../common/utils";
import { View } from "react-native";
import Flexh from "../component/Flexh";
import ComSpacer from "../component/ComSpacer";
import { Clone } from "../common/classes";
import { useState,useEffect } from "react";
import AsyncStorage from "@react-native-async-storage/async-
storage";

export default function PageShare({route}){
  const {cloneid}=route.params
  const url=config.web+"use?cloneid="+cloneid
  const [clone,setClone]=useState(new Clone())
  const [textColor,setTextColor]=useState("")

  useEffect(()=>{
    AsyncStorage.getItem(cloneid)
    .then(res=>{
      var clone=JSON.parse(res)
      setClone(new
Clone(clone.name,clone.type,clone.prompt,clone.adminid))
```

```
      setTextColor(utils.Colors[clone.type])
    })
  })
  return(
    <PageBase2>
      {/* <QRCode
        value="cribug"
        size={200}
        color="black"
        backgroundColor="white"      // 二维码的背景色。
      /> */}
      <View style={{width:
"65%",flexDirection:"column",justifyContent:"center",alignItems:"
center"}}>
        <QRCode
          size={240}
          style={{flex:1}}
          value={url}
          viewBox={`0 0 200 200`}
        />
        <ComSpacer height={20}></ComSpacer>
        <Flexh>
          <Text numberOfLines={1} ellipsizeMode="tail"
style={{fontSize:utils.fontSize1,width:"60%",color:textColor}}>{c
lone.name}</Text>
          <Button
            icon={"content-copy"}
            onPress={()=>{
              Clipboard.setStringAsync(url)
              console.log(url)
              utils.dispatch({type:"SUCCESS",message:"已复制,粘贴给
好友吧"})}}
              >复制链接
          </Button>
        </Flexh>
      </View>
    </PageBase2>
  )
}
```

PageState.js

```
import ComState from "../component/ComState"
import utils from "../common/utils"
```

```
const { useEffect, useState,useCallback } = require("react")
const { default: PageBase0 } = require("../component/PageBase0")
const { default: AsyncStorage } = require("@react-native-async-
storage/async-storage")
import axios from "axios"
import ComSpacer from "../component/ComSpacer"
import { View } from "react-native"
import { Text } from "react-native-paper"
import { useAuth } from "../common/AuthContext"
import { useFocusEffect } from "@react-navigation/native"

const PageState=({navigation})=>{
  // const navigation=useNavigation()
  const [states,setStates]=useState({})
  const [clones,setClones]=useState([])
  const {state,dispatch}=useAuth()
  useFocusEffect(
    useCallback(()=>{
      dispatch({type:"SHOW"})
      AsyncStorage.getItem("clones")
      .then(res=>{
        setClones(JSON.parse(res))
        payAttention()
      })
      .catch(res=>{
        console.log(res)
        setClones([])
      })
    },[])
  )

  function payAttention(){
    // axios.post(
    //   utils.url+"/payattention",

    // )
    // .then(res=>{
    //   console.log(res.data)
    // })
    // .catch(res=>{
    //   console.log(res)
    // })
    console.log(utils.token)
    AsyncStorage.getItem("token")
    .then(token=>{
      axios.get(
        utils.url+"getclonestate",
        {
          headers:{
            'Authorization': `Bearer ${utils.token}`
          }
        }
      )
      .then(res=>{
        var data=res.data
        console.log(data.states)
        var _states=data.states
        var states={}
        _states.forEach((state,index)=>{
          console.log(state)
          var cloneid=state.id
          var answers=state.answers
          if(answers==null){
            answers=[]
          }
          states[cloneid]=answers
        })
        setStates(states)
      })
      .catch(err=>{
        console.log("getclonestate "+err)
      })
    })
  }

  if(clones!=null){
    return(
      <PageBase0 name={"状态页"}>
        {clones.map((clone,index)=>
          <View key={index} style={{width:"100%"}}>
            <ComState clone={clone}
messages={states[clone.cloneid]}></ComState>
            <ComSpacer height={20}/>
          </View>
        )}
      </PageBase0>
    )
  }else{
    return <PageBase0 name={"状态页"}>
      <Text style={{fontSize:utils.fontSize2,color:"gray"}}>空无一
物</Text>
    </PageBase0>
  }

}
export default PageState
```

PageUpdate.js

```
import { View, Button } from 'react-native';
import * as Updates from 'expo-updates';

function update() {
  async function onFetchUpdateAsync() {
    try {
      const update = await Updates.checkForUpdateAsync();

      if (update.isAvailable) {
        await Updates.fetchUpdateAsync();
        await Updates.reloadAsync();
      }
    } catch (error) {
      // You can also add an alert() to see the error message in
case of an error when fetching updates.
      alert(`Error fetching latest Expo update: ${error}`);
    }
  }

  return (
    <View>
      <Button title="Fetch update" onPress={onFetchUpdateAsync} />
    </View>
  );
}
export default update
```

PageUse.js

```
import { View,Text, Button } from 'react-native'
import {Clone} from '../common/classes'
import { useEffect, useState } from 'react'
import PageBase2 from '../component/PageBase2'
import { IconButton, TextInput } from 'react-native-paper'
import axios from 'axios'
import utils from '../common/utils'
import { StyleSheet } from 'react-native'
import { ActivityIndicator } from 'react-native-paper';
import { Audio } from 'expo-av'
import FormData from 'form-data'
import config from '../common/config'

export default function PageUse({route}){
  const cloneid=route.params.cloneid
  const [clone,setClone]=useState(new Clone("","","","",""))
  const [icon,setIcon]=useState('microphone-outline')
  const [ans,setAns]=useState("")
  const [inputByAudio,setInputByAudio]=useState(true)
  const [messageToSend,setMessageToSend]=useState("")
  const [disabledBtSend,setDisableBtSend]=useState(true)
  const [disabledBtMic,setDisabledBtMic]=useState(false)
  const [messages,setMessages]=useState([])
  const [sending,setSending]=useState(false)
  const [havePerm,setHavePerm]=useState(false)
  const [ansAlign,setAnsAlign]=useState("left")
  const [cloneidOK,setCloneidOK]=useState(false)

  useEffect(()=>{
    var cloneIdLen=cloneid.length
    if(cloneIdLen!=24){
      utils.dispatch({type:"FAIL",message:"不存在,请检查 id 对错"})
      setCloneidOK(false)
      return
    }
    else{
      setCloneidOK(true)
    }
    clone.visitclone(cloneid,setClone)
    check()
  },[])
  async function check(){
    const { status } = await Audio.getPermissionsAsync();
    if (status !== 'granted') {
      setHavePerm(false)
    }else{
      setHavePerm(true)
    }
  }
  function SetAns(ans){
    setAns(ans)
    if(ans.length>10){
      setAnsAlign("left")
    }else{
      setAnsAlign("center")
    }
  }
  function rec(fileUri){
    setDisabledBtMic(true)
    const file = {
      uri: fileUri,
      type: 'audio/aac',  // 这里以 MP3 文件为例
      name: 'example.aac'  // 你希望上传后保存的文件名
    };
    const formData = new FormData();
    formData.append('audio', file);
    axios.postForm(utils.url+"rec",formData,{
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded'
      },
    })
    .then(response => {
      var recMsg=response.data.value
      toGpt(clone.type,recMsg)
    })
    .catch(error => {
        console.error('上传文件时出错', error);
    })
    .finally(res=>{
      setDisabledBtMic(false)
    });
  }
  function toGpt(type,msg){
    if(type=="说明书"){
      console.log("getgpt")
      getGpt(msg)
    }else{
      postGpt(msg)
    }
  }
  function getGpt(currentMsg){
    setMessageToSend("")
    setSending(true)
    axios.get(
      config.url+"gpt/"+cloneid+"/"+currentMsg,
    )
    .then(res=>{
      console.log(res.data)
      var data=res.data
      var answer=data.msg
      messages.push(currentMsg)
      messages.push(answer)
      SetAns(answer)
      if(messages.length>4){
          messages.splice(0,2)
        }
      setMessages(messages)
    })
    .catch(res=>{
      setMessageToSend(currentMsg)
      console.log(res)
    })
    .finally(res=>{
      setSending(false)
    })
  }
  function postGpt(currentMsg){
    setMessageToSend("")
    setSending(true)
    const data = {
      messages: [...messages,currentMsg],
      cloneid:cloneid
    };
    axios.post(
      utils.url+"gpt",
      data,
    )
    .then(res=>{
      var data=res.data
      var answer=data.msg
      messages.push(currentMsg)
      messages.push(answer)
      SetAns(answer)
      if(messages.length>4){
          messages.splice(0,2)
        }
      setMessages(messages)
    })
    .catch(res=>{
      setMessageToSend(currentMsg)
    })
    .finally(res=>{
      setSending(false)
    })
  }
  async function checkPermission(){
    const { status } = await Audio.getPermissionsAsync();
    if (status !== 'granted') {
      const { status: newStatus } = await
Audio.requestPermissionsAsync();
      if (newStatus === 'granted') {
        setHavePerm(true)
        console.log("得到权限")
        return
      } else {
        setHavePerm(false)
        console.log("未得到权限")
        return
      }
    } else {
      utils.startRecording()
      setIcon("microphone");
    }
  }
  if(!cloneidOK){
    return(
      <PageBase2>
        <Text style={{fontSize:utils.fontSize2,color:"gray"}}>
          空无一物
        </Text>
      </PageBase2>
    )
  }
  else
  return (
    <PageBase2>
      <View
style={{width:'100%',height:'96%',flexDirection:"column",justifyC
ontent:'space-between',alignItems:'center'}}>
```

```
        <Text
style={{fontSize:utils.fontSize2}}>{clone.name}</Text>
        <Text
style={{fontSize:utils.fontSize2,width:"80%",textAlign:ansAlign}}>
          {ans}
        </Text>
        <View
style={{flexDirection:"column",justifyContent:"center",alignItems
:"center",width:"100%",}}>
          {inputByAudio && <IconButton icon={icon} size={72}
          disabled={disabledBtMic}
          style={{position:"absolute",zIndex:100}}
            onPressIn={()=>{
              checkPermission()
            }}
            onPressOut={()=>{
              if(havePerm){
                utils.stopRecording(rec)
                setIcon("microphone-outline")
              }
            }}
            >
          </IconButton>}
          <View style={styles.centered}>
            <View style={{flex:1.5,left:0,padding:0}}>
              {inputByAudio && <IconButton icon={"keyboard"}
animated={true} iconColor='red' size={utils.fontSize3}
                onPress={()=>{
                  setInputByAudio(!inputByAudio)
                }}
                ></IconButton>}
              {!inputByAudio && <IconButton icon={"microphone-
settings"} animated={true} iconColor='red' size={utils.fontSize3}
                onPress={()=>{
                  setInputByAudio(!inputByAudio)
                }}
              ></IconButton>}
            </View>
            {!inputByAudio && <TextInput mode='outlined' label={"
发送文字"} value={messageToSend} multiline
              style={{flex:7,padding:0}} onChangeText={(value)=>{
                setMessageToSend(value)

value==""?setDisableBtSend(true):setDisableBtSend(false)
```

```
              }}
            ></TextInput>}
            <View style={{flex:1.5,padding:0}} >
              {!inputByAudio && !sending && <IconButton
icon={"send-circle"} animated={true}
iconColor={utils.theme.colors.primary} size={utils.fontSize3}
                disabled={disabledBtSend}
                onPress={()=>{
                  toGpt(clone.type,messageToSend)
                }}
                ></IconButton>}
              {!inputByAudio && sending &&
                <ActivityIndicator animating={true}
color="#6200EE" size={utils.fontSize2} />
              }
            </View>
          </View>
        </View>
      </View>
    </PageBase2>
  );
}


const styles=StyleSheet.create({
  centered: {
    width:'100%',
    flexDirection:'row',
    alignItems:"center",
    justifyContent:"space-between"
  }
})
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();
```

```
function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
```

```
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage: 
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage: 
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});
```

```
// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
```

```
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
```

```
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
```

```
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
```

```
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
```

```
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}
```

```
export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}
```

```
export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
```

```
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
```

```
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
```

```
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();
```

```
function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
```

```
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});
```

```
// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
```

```
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
```

```
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage: 
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
```

```
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
```

```
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
```

```
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage: 
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage: 
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage: 
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}
```

```
export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}
```

```
export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
```

```
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
```

```
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
```

```
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();
```

```
function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
```

```
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage: 
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage: 
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});
```

```
// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```

AppNavigator.js

```
import PageCreate from '../pages/PageCreate';
import PageState from '../pages/PageState';
import PageLogin from '../pages/PageLogin';
import PageDesktop from '../pages/PageDesktop';
import PageAdmin from '../pages/PageAdmin';
import PageUse from '../pages/PageUse';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();

function AppNavigator() {
  return (
    <Stack.Navigator initialRouteName="desktop">
      <Stack.Screen name="create" component={PageCreate} />
      <Stack.Screen name="state" component={PageState} />
      <Stack.Screen name="login" component={PageLogin} />
      <Stack.Screen name="desktop" component={PageDesktop} />
      <Stack.Screen name="admin" component={PageAdmin} />
      <Stack.Screen name="use" component={PageUse} />
    </Stack.Navigator>
  );
}

export default AppNavigator
```

AuthContext.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import React, { createContext, useContext, useReducer } from
'react';
import utils from './utils';

const AuthContext = createContext();
const initialState = {
  empty:false,
  empty2:false,
  logined: false,
  display:'all',
  notiSuc:false,
  notiInfo:false,
  notiFail:false,
  notiMessage:"",
  showProtocolDialog:false,
};
const HideAllNoti=(state)=>{
  return { ...state, notiSuc: false, notiFail: false, }
}
function authReducer(state, action) {
  switch (action.type) {
    case 'LOGOUT':
      AsyncStorage.setItem("token","")
      AsyncStorage.clear()
      utils.token=""
      return { ...state, logined: false };
    case 'LOGIN':
      utils.get401=false
      return { ...state, logined: true };
    case 'HIDE':
      return { ...state,empty:true ,empty2:false, display:
```

```
'none'};
    case 'SHOW':
      return { ...state,empty:false,empty2:true, display: 'all' };
    case 'SUCCESS':
      state=HideAllNoti(state)
      return { ...state, notiSuc: true, notiMessage:
action.message};
    case 'FAIL':
      state=HideAllNoti(state)
      console.log("FAIL")
      return { ...state, notiFail: true, notiMessage:
action.message };
    case 'INFO':
      state=HideAllNoti(state)
      return { ...state, notiInfo: true, notiMessage:
action.message };
    case 'NOTIHIDE':
      return HideAllNoti(state)
    case 'SHOWDIALOG':
      console.log("showdia")
      return { ...state, showProtocolDialog:true };
    case 'HIDEDIALOG':
      return { ...state, showProtocolDialog:false };
    default:
      return state;
  }
}
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);
  return (
    <AuthContext.Provider value={{ state, dispatch }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  return useContext(AuthContext);
}
```

Axios.js

```
import AsyncStorage from '@react-native-async-storage/async-
storage';
import axios from 'axios';
import utils from './utils';
import config from './config';

// 创建 axios 实例
const Axios = axios.create({
    baseURL: config.url, // 替换成你的 Axios 基础链接
    timeout: 5000 // 请求超时时间
});

// 请求拦截器
Axios.interceptors.request.use(
    config => {
      if(utils.token==""){
        utils.navigation.navigate("login",{positive:false})
      }
      const token =utils.token; // 从 localStorage 获取 token
      if (token) {
          config.headers['Authorization'] = 'Bearer ' + token; //
将 token 加入到请求头
      }
      return config;
    },
    error => {
        return Promise.reject(error);
    }
);

// 响应拦截器
export function setupInterceptors(store) {
  Axios.interceptors.response.use(
      response => response,
      error => {
        if (!utils.get401 &&error.response &&
error.response.status === 401) {
              utils.get401=true
              AsyncStorage.setItem("token","")
              store.dispatch({ type: 'LOGOUT' });
              utils.navigation.navigate("login",{positive:false})
              console.log("Token expired. Please login again.");
        }
        if(error.response && error.response.status === 502){
          utils.dispatch({type:"FAIL",message:"服务器宕机,全力修复中
"})
        }
        return Promise.reject(error);
      }
  );
}

export default Axios;
```