import { StyleSheet, View, Pressable, Text } from “react-native”; import FontAwesome from “@expo/vector-icons/FontAwesome”;

export default function Button({ label,theme,onPress}) { if (theme === “primary”) { return ( <View style={[styles.buttonContainer, { borderWidth: 4, borderColor: “#ffd33d”, borderRadius: 18 }]}> <Pressable style={[styles.button, { backgroundColor: “#fff” }]} onPress={() => alert(‘You pressed a button.’)} > <Text style={[styles.buttonLabel, { color: “#25292e” }]}>{label} ); } if (theme === “index”) { return ( <View style={[styles.buttonContainer, { borderWidth: 2, borderColor: “#afd33d”, borderRadius: 7,width: 110, height: 30, }]}> <Pressable style={[styles.button, { backgroundColor: “#fff” }]} onPress={() => alert(‘You pressed a button.’)} > <Text style={[styles.buttonLabel, { color: “#25292e” }]}>{label} ); } if (theme === “test”) { return ( <View style={[styles.buttonContainer, { borderWidth: 2, borderColor: “#afd33d”, borderRadius: 7,width: 110, height: 30, }]}> <Pressable style={[styles.button, { backgroundColor: “#fff” }]} onPress={onPress} > <Text style={[styles.buttonLabel, { color: “#25292e” }]}>{label} ); } return ( <Pressable style={styles.button} onPress={() => alert(“You pressed a button.”)} > {label} ); }

const styles = StyleSheet.create({ buttonContainer: { width: 160, height: 50, marginHorizontal: 30, alignItems: “center”, justifyContent: “center”, padding: 3, }, button: { borderRadius: 10, width: “100%”, height: “100%”, alignItems: “center”, justifyContent: “center”, flexDirection: “row”, }, buttonIcon: { paddingRight: 3, }, buttonLabel: { color: “#000”, fontSize: 13, }, }); import { View, Pressable, StyleSheet,Text } from ‘react-native’; import MaterialIcons from ‘@expo/vector-icons/MaterialIcons’;

export default function CircleButton({label, onPress }) { return ( <Text style={[styles.buttonLabel, { color: “#25292e” }]}>{label}

</Pressable>  
</View>

); }

const styles = StyleSheet.create({ circleButtonContainer: { width: 84, height: 84, marginHorizontal: 60, borderWidth: 4, borderColor: ‘#ffd33d’, borderRadius: 42, padding: 3, }, circleButton: { flex: 1, justifyContent: ‘center’, alignItems: ‘center’, borderRadius: 42, backgroundColor: ‘#fff’, },

buttonLabel: { color: “#000”, fontSize: 13, },

}); import React from ‘react’; import { StyleSheet } from ‘react-native’;

import { ExternalLink } from ‘./ExternalLink’; import { MonoText } from ‘./StyledText’; import { Text, View } from ‘./Themed’;

import Colors from ‘@/constants/Colors’;

export default function EditScreenInfo({ path }: { path: string }) { if (path ==“app/data.tsx”){ return ( 实时数据分析：展示用户的睡眠数据、脑电波图形以及睡眠改善效果的统计数据。

<Text  
 style={styles.getStartedText}  
 lightColor="rgba(0,0,0,0.8)"  
 darkColor="rgba(255,255,255,0.8)">  
 个性化建议：根据分析结果向用户推荐个性化的改善睡眠方法。  
 </Text>  
 </View>  
  
 <View style={styles.helpContainer}>  
 <ExternalLink  
 style={styles.helpLink}  
 href="https://www.msdmanuals.cn/home/brain-spinal-cord-and-nerve-disorders/sleep-disorders/overview-of-sleep?query=%E7%9D%A1%E8%A7%89">  
 <Text style={styles.helpLinkText} lightColor={Colors.light.tint}>  
 点击了解更多关于睡眠的知识  
 </Text>  
 </ExternalLink>  
 </View>  
 </View>  
);

}

if (path== “app/ctrl.tsx”) { return ( Open up the code for this screen:

<View  
 style={[styles.codeHighlightContainer, styles.homeScreenFilename]}  
 darkColor="rgba(255,255,255,0.05)"  
 lightColor="rgba(0,0,0,0.05)">  
 <MonoText>{path}</MonoText>  
 </View>  
  
 <Text  
 style={styles.getStartedText}  
 lightColor="rgba(0,0,0,0.8)"  
 darkColor="rgba(255,255,255,0.8)">  
 Change any of the text, save the file, and your app will automatically update.  
 </Text>  
 </View>  
  
 <View style={styles.helpContainer}>  
 <ExternalLink  
 style={styles.helpLink}  
 href="https://docs.expo.io/get-started/create-a-new-app/#opening-the-app-on-your-phonetablet">  
 <Text style={styles.helpLinkText} lightColor={Colors.light.tint}>  
 Tap here if your app doesn't automatically update after making changes  
 </Text>  
 </ExternalLink>  
 </View>  
 </View>  
);

} if (path == “app/user.tsx”) { return ( Owhat

<View  
 style={[styles.codeHighlightContainer, styles.homeScreenFilename]}  
 darkColor="rgba(255,255,255,0.05)"  
 lightColor="rgba(0,0,0,0.05)">  
 <MonoText>{path}</MonoText>  
 </View>  
  
 <Text  
 style={styles.getStartedText}  
 lightColor="rgba(0,0,0,0.8)"  
 darkColor="rgba(255,255,255,0.8)">  
 Change any of the text, save the file, and your app will automatically update.  
 </Text>  
 </View>  
  
 <View style={styles.helpContainer}>  
 <ExternalLink  
 style={styles.helpLink}  
 href="https://docs.expo.io/get-started/create-a-new-app/#opening-the-app-on-your-phonetablet">  
 <Text style={styles.helpLinkText} lightColor={Colors.light.tint}>  
 Tap here if your app doesn't automatically update after making changes  
 </Text>  
 </ExternalLink>  
 </View>  
 </View>  
);

} }

const styles = StyleSheet.create({ getStartedContainer: { alignItems: ‘center’, marginHorizontal: 50, }, homeScreenFilename: { marginVertical: 7, }, codeHighlightContainer: { borderRadius: 3, paddingHorizontal: 4, }, getStartedText: { fontSize: 14, lineHeight: 24, textAlign: ‘center’, }, helpContainer: { marginTop: 15, marginHorizontal: 20, alignItems: ‘center’, }, helpLink: { paddingVertical: 15, }, helpLinkText: { textAlign: ‘center’, }, }); import { Link } from ‘expo-router’; import \* as WebBrowser from ‘expo-web-browser’; import React from ‘react’; import { Platform } from ‘react-native’;

export function ExternalLink( props: Omit<React.ComponentProps, ‘href’> & { href: string } ) { return ( <Link target=“\_blank” {…props} // @ts-expect-error: External URLs are not typed. href={props.href} onPress={(e) => { if (Platform.OS !== ‘web’) { // Prevent the default behavior of linking to the default browser on native. e.preventDefault(); // Open the link in an in-app browser. WebBrowser.openBrowserAsync(props.href as string); } }} /> ); } import React from “react”; import { View, Text, TextInput } from “react-native”; import { styles } from “./styles”;

export default function FormField({ field, label, secureTextEntry, autoCapitalize, values, touched, errors, handleChange, handleBlur, }) { return ( {label}

<TextInput  
 style={styles.input}  
 value={values[field]}  
 onChangeText={handleChange(field)}  
 onBlur={handleBlur(field)}  
 secureTextEntry={secureTextEntry}  
 autoCapitalize={autoCapitalize || "none"}  
 />  
  
 {touched[field] && errors[field] ? (  
 <View style={styles.errorContainer}>  
 <Text style={styles.errorText}>{errors[field]}</Text>  
 </View>  
 ) : null}  
</View>

); } import { StyleSheet, View,Image } from ‘react-native’;

export default function ImageViewer({ placeholderImageSource, selectedImage }) { const imageSource = selectedImage ? { uri: selectedImage } : placeholderImageSource; console.log(‘Selected Image:’, selectedImage); console.log(‘Image Source:’, imageSource);

return ( ); } const styles = StyleSheet.create({ image: { width: 440, height: 240, borderRadius: 18, }, }); import React from “react”; import { SafeAreaView, View, Text, TouchableOpacity, Alert, } from “react-native”; import { StatusBar } from “expo-status-bar”; import { KeyboardAwareScrollView } from “react-native-keyboard-aware-scroll-view”; import { Formik } from “formik”; import { validationSchema } from “./validation”; import { styles } from “./styles”; import FormField from “./FormField”;

export default function RegisterForm( {setShowAppOptions} ) {

function onSubmitHandler(values) {

// Translate the content and customize the login instruction  
let loginInstruction = `请使用您的邮箱 ${values.email} \n 和密码 ${values.password} 登录，即可使用设备`;  
  
// Display the translated and customized alert message  
Alert.alert(  
 "注册成功!",  
 `${loginInstruction}`  
);  
setShowAppOptions(false);

}

function isFormValid(isValid, touched) { return isValid && Object.keys(touched).length !== 0; }

return ( <>

<StatusBar style="light" />  
  
 <SafeAreaView style={styles.container}>  
 <View style={styles.header}>  
 <Text style={styles.headerText}>Zensleep账号注册</Text>  
 </View>  
  
 {/\* https://github.com/APSL/react-native-keyboard-aware-scroll-view \*/}  
 <KeyboardAwareScrollView  
 style={styles.content}  
 showsVerticalScrollIndicator={false}  
 keyboardShouldPersistTaps="handled"  
 extraScrollHeight={150}  
 >  
 {/\* https://formik.org/docs/overview \*/}  
 <Formik  
 initialValues={{  
 name: "test",  
 email: "test@test.test",  
 password: "testtt",  
 confirmPassword: "testtt",  
 }}  
 onSubmit={onSubmitHandler}  
 validationSchema={validationSchema}  
 >  
 {({  
 handleChange,  
 handleBlur,  
 handleSubmit,  
 values,  
 errors,  
 touched,  
 isValid,  
 }) => (  
 <>  
 <FormField  
 field="name"  
 label="昵称"  
 autoCapitalize="words"  
 values={values}  
 touched={touched}  
 errors={errors}  
 handleChange={handleChange}  
 handleBlur={handleBlur}  
 />  
  
 <FormField  
 field="email"  
 label="邮箱"  
 values={values}  
 touched={touched}  
 errors={errors}  
 handleChange={handleChange}  
 handleBlur={handleBlur}  
 />  
  
 <FormField  
 field="password"  
 label="密码"  
 secureTextEntry={true}  
 values={values}  
 touched={touched}  
 errors={errors}  
 handleChange={handleChange}  
 handleBlur={handleBlur}  
 />  
  
 <FormField  
 field="confirmPassword"  
 label="确认密码"  
 secureTextEntry={true}  
 values={values}  
 touched={touched}  
 errors={errors}  
 handleChange={handleChange}  
 handleBlur={handleBlur}  
 />  
  
<TouchableOpacity onPress={ handleSubmit }>  
 <View  
 style={[  
 styles.button,  
 {  
 opacity: isFormValid(isValid, touched) ? 1 : 0.5,  
 },  
 ]}  
 >  
 <Text style={styles.buttonText}>提交</Text>  
 </View>  
 </TouchableOpacity>  
 </>  
 )}  
 </Formik>  
 </KeyboardAwareScrollView>  
 </SafeAreaView>  
</>

); } import { Text, TextProps } from ‘./Themed’;

export function MonoText(props: TextProps) { return <Text {…props} style={[props.style, { fontFamily: ‘SpaceMono’ }]} />; } import { StyleSheet, Platform } from “react-native”; import Constants from “expo-constants”;

const HEADER\_BACKGROUND = “#3498db”; const CONTENT\_BACKGROUND = “#f9f9f9”;

export const styles = StyleSheet.create({ topSafeArea: { backgroundColor: HEADER\_BACKGROUND, }, container: { flex: 1, paddingTop: Constants.statusBarHeight, backgroundColor: Platform.OS === “ios” ? CONTENT\_BACKGROUND : HEADER\_BACKGROUND, }, header: { height: 60, justifyContent: “center”, alignItems: “center”, backgroundColor: HEADER\_BACKGROUND, }, headerText: { color: “#fff”, fontSize: 18, }, content: { padding: 20, backgroundColor: CONTENT\_BACKGROUND, }, formGroup: { marginBottom: 10, }, label: { color: “#7d7e79”, fontSize: 16, lineHeight: 30, }, input: { height: 50, paddingHorizontal: 20, borderRadius: 5, borderWidth: 2, borderColor: “#e3e3e3”, backgroundColor: “#fff”, }, errorContainer: { marginVertical: 5, }, errorText: { color: “#ff7675”, }, button: { marginTop: 20, backgroundColor: “#2980b9”, padding: 15, borderRadius: 15, }, buttonText: { color: “#fff”, fontWeight: “bold”, fontSize: 18, textAlign: “center”, }, }); /\*\* \* Learn more about Light and Dark modes: \* https://docs.expo.io/guides/color-schemes/ \*/

import { Text as DefaultText, View as DefaultView } from ‘react-native’;

import Colors from ‘@/constants/Colors’; import { useColorScheme } from ‘./useColorScheme’;

type ThemeProps = { lightColor?: string; darkColor?: string; };

export type TextProps = ThemeProps & DefaultText[‘props’]; export type ViewProps = ThemeProps & DefaultView[‘props’];

export function useThemeColor( props: { light?: string; dark?: string }, colorName: keyof typeof Colors.light & keyof typeof Colors.dark ) { const theme = useColorScheme() ?? ‘light’; const colorFromProps = props[theme];

if (colorFromProps) { return colorFromProps; } else { return Colors[theme][colorName]; } }

export function Text(props: TextProps) { const { style, lightColor, darkColor, …otherProps } = props; const color = useThemeColor({ light: lightColor, dark: darkColor }, ‘text’);

return <DefaultText style={[{ color }, style]} {…otherProps} />; }

export function View(props: ViewProps) { const { style, lightColor, darkColor, …otherProps } = props; const backgroundColor = useThemeColor({ light: lightColor, dark: darkColor }, ‘background’);

return <DefaultView style={[{ backgroundColor }, style]} {…otherProps} />; } // This function is web-only as native doesn’t currently support server (or build-time) rendering. export function useClientOnlyValue<S, C>(server: S, client: C): S | C { return client; } import React from ‘react’;

// useEffect is not invoked during server rendering, meaning // we can use this to determine if we’re on the server or not. export function useClientOnlyValue<S, C>(server: S, client: C): S | C { const [value, setValue] = React.useState<S | C>(server); React.useEffect(() => { setValue(client); }, [client]);

return value; } export { useColorScheme } from ‘react-native’; // NOTE: The default React Native styling doesn’t support server rendering. // Server rendered styles should not change between the first render of the HTML // and the first render on the client. Typically, web developers will use CSS media queries // to render different styles on the client and server, these aren’t directly supported in React Native // but can be achieved using a styling library like Nativewind. export function useColorScheme() { return ‘light’; } import \* as Yup from “yup”;

// https://github.com/jquense/yup export const validationSchema = Yup.object().shape({ name: Yup.string().required(“必填项：请输入昵称”), email: Yup.string() .email(“请输入有效的电子邮件地址”) .required(“必填项：请输入已注册的电子邮件”), password: Yup.string() .required(“必填项：请输入密码”) .min(6, “密码长度至少为6个字符”), confirmPassword: Yup.string() .required(“请确认密码”) .oneOf([Yup.ref(“password”)], “密码和确认密码不匹配”), }); default: run test monitor clean build deploy lint analysis: echo ‘Analysising!’ run: echo ‘Running!’ test: echo ‘Testing!’ monitor: echo ‘Monitoring!’ clean: echo ‘Cleaning!’ build: echo ‘Building!’ deploy: echo ‘Deploying!’ lint: echo ‘Linting!’

alias R :=run alias A :=analysis alias T :=test alias M :=monitor alias C :=clean alias B :=build alias D :=deploy alias L :=lint

# keys for calling commands in editor to execute other tmux panel

a: tmux send-keys -t 2 “just analysis” C-m; r: tmux send-keys -t 2 “r”; t: tmux send-keys -t 2 “just test” C-m; n: tmux send-keys -t 2 “just monitor” C-m; e: tmux send-keys -t 2 “just clean” C-m; i: tmux send-keys -t 2 “just build” C-m; o: tmux send-keys -t 2 “just deploy” C-m; l: tmux send-keys -t 2 “just lint” C-m; s: tmux send-keys -t 2 “just default” C-m; c: tmux send-keys -t 2 C-C; { “extends”: “expo/tsconfig.base”, “compilerOptions”: { “strict”: true, “paths”: { “@/\*“: [ ”./\*” ] } },”include”: [ “\*\*/\*.ts”, “\*\*/\*.tsx”, “.expo/types/\*\*/\*.ts”, “expo-env.d.ts” ] } { “expo”: { “name”: “ZenSleep”, “slug”: “ZenSleep”, “version”: “1.0.0”, “orientation”: “portrait”, “icon”: “./assets/images/icon.png”, “scheme”: “myapp”, “userInterfaceStyle”: “automatic”, “splash”: { “image”: “./assets/images/splash.png”, “resizeMode”: “contain”, “backgroundColor”: “#ffffff” }, “assetBundlePatterns”: [ “\*\*/\*” ], “ios”: { “supportsTablet”: true }, “android”: { “adaptiveIcon”: { “foregroundImage”: “./assets/images/adaptive-icon.png”, “backgroundColor”: “#ffffff” } }, “web”: { “bundler”: “metro”, “output”: “static”, “favicon”: “./assets/images/favicon.png” }, “plugins”: [ “expo-router” ], “experiments”: { “typedRoutes”: true } } } ///

// NOTE: This file should not be edited and should be in your git ignoreimport { ScrollViewStyleReset } from ‘expo-router/html’;

// This file is web-only and used to configure the root HTML for every // web page during static rendering. // The contents of this function only run in Node.js environments and // do not have access to the DOM or browser APIs. export default function Root({ children }: { children: React.ReactNode }) { return (

{/\* Disable body scrolling on web. This makes ScrollView components work closer to how they do on native. However, body scrolling is often nice to have for mobile web. If you want to enable it, remove this line. \*/}

{/\* Using raw CSS styles as an escape-hatch to ensure the background color never flickers in dark-mode. \*/} <style dangerouslySetInnerHTML={{ \_\_html: responsiveBackground }} /> {/\* Add any additional

elements that you want globally available on web… \*/}

{children}

); }

const responsiveBackground = body { background-color: #fff; } @media (prefers-color-scheme: dark) { body { background-color: #000; } }; import { Link, Stack } from ‘expo-router’; import { StyleSheet } from ‘react-native’;

import { Text, View } from ‘@/components/Themed’;

export default function NotFoundScreen() { return ( <> <Stack.Screen options={{ title: ‘Oops!’ }} /> This screen doesn’t exist.

Go to home screen! </> ); }

const styles = StyleSheet.create({ container: { flex: 1, alignItems: ‘center’, justifyContent: ‘center’, padding: 20, }, title: { fontSize: 20, fontWeight: ‘bold’, }, link: { marginTop: 15, paddingVertical: 15, }, linkText: { fontSize: 14, color: ‘#2e78b7’, }, }); import FontAwesome from ‘@expo/vector-icons/FontAwesome’; import { DarkTheme, DefaultTheme, ThemeProvider } from ‘@react-navigation/native’; import { useFonts } from ‘expo-font’; import { Stack } from ‘expo-router’; import \* as SplashScreen from ‘expo-splash-screen’; import { useEffect } from ‘react’;

import { useColorScheme } from ‘@/components/useColorScheme’;

export { // Catch any errors thrown by the Layout component. ErrorBoundary, } from ‘expo-router’;

export const unstable\_settings = { // Ensure that reloading on /modal keeps a back button present. initialRouteName: ‘(tabs)’, };

// Prevent the splash screen from auto-hiding before asset loading is complete. SplashScreen.preventAutoHideAsync();

export default function RootLayout() { const [loaded, error] = useFonts({ SpaceMono: require(‘../assets/fonts/SpaceMono-Regular.ttf’), …FontAwesome.font, });

// Expo Router uses Error Boundaries to catch errors in the navigation tree. useEffect(() => { if (error) throw error; }, [error]);

useEffect(() => { if (loaded) { SplashScreen.hideAsync(); } }, [loaded]);

if (!loaded) { return null; }

return ; }

function RootLayoutNav() { const colorScheme = useColorScheme();

return ( <ThemeProvider value={colorScheme === ‘dark’ ? DarkTheme : DefaultTheme}> <Stack.Screen name=“(tabs)” options={{ headerShown: false }} /> ); } import { StatusBar } from ‘expo-status-bar’; import { Platform, StyleSheet } from ‘react-native’;

import EditScreenInfo from ‘@/components/EditScreenInfo’; import { Text, View } from ‘@/components/Themed’;

export default function ModalScreen() { return ( 光音控制指南

{/\* Use a light status bar on iOS to account for the black space above the modal \*/} <StatusBar style={Platform.OS === ‘ios’ ? ‘light’ : ‘auto’} /> ); }

const styles = StyleSheet.create({ container: { flex: 1, alignItems: ‘center’, justifyContent: ‘center’, }, title: { fontSize: 20, fontWeight: ‘bold’, }, separator: { marginVertical: 30, height: 1, width: ‘80%’, }, }); import { StatusBar } from ‘expo-status-bar’; import { Platform, StyleSheet } from ‘react-native’;

import EditScreenInfo from ‘@/components/EditScreenInfo’; import { Text, View } from ‘@/components/Themed’;

export default function ModalScreen() { return ( 图表与建议指南

{/\* Use a light status bar on iOS to account for the black space above the modal \*/} <StatusBar style={Platform.OS === ‘ios’ ? ‘light’ : ‘auto’} /> ); }

const styles = StyleSheet.create({ container: { flex: 1, alignItems: ‘center’, justifyContent: ‘center’, }, title: { fontSize: 20, fontWeight: ‘bold’, }, separator: { marginVertical: 30, height: 1, width: ‘80%’, }, }); import { StatusBar } from ‘expo-status-bar’; import { Platform, StyleSheet } from ‘react-native’;

import EditScreenInfo from ‘@/components/EditScreenInfo’; import { Text, View } from ‘@/components/Themed’;

export default function ModalScreen() { return ( 用户反馈使用指南

{/\* Use a light status bar on iOS to account for the black space above the modal \*/} <StatusBar style={Platform.OS === ‘ios’ ? ‘light’ : ‘auto’} /> ); }

const styles = StyleSheet.create({ container: { flex: 1, alignItems: ‘center’, justifyContent: ‘center’, }, title: { fontSize: 20, fontWeight: ‘bold’, }, separator: { marginVertical: 30, height: 1, width: ‘80%’, }, }); import React from ‘react’; import FontAwesome from ‘@expo/vector-icons/FontAwesome’; import { Link, Tabs } from ‘expo-router’; import { Pressable } from ‘react-native’; import Colors from ‘@/constants/Colors’; import { useColorScheme } from ‘@/components/useColorScheme’; import { useClientOnlyValue } from ‘@/components/useClientOnlyValue’;

// You can explore the built-in icon families and icons on the web at https://icons.expo.fyi/ function TabBarIcon(props: { name: React.ComponentProps[‘name’]; color: string; }) { return <FontAwesome size={28} style={{ marginBottom: -3 }} {…props} />; }

export default function TabLayout() { const colorScheme = useColorScheme();

return ( <Tabs screenOptions={{ tabBarActiveTintColor: Colors[colorScheme ?? ‘light’].tint, // Disable the static render of the header on web // to prevent a hydration error in React Navigation v6. headerShown: useClientOnlyValue(false, true), }}> <Tabs.Screen name=“index” options={{ title: ‘数据中心’, tabBarIcon: ({ color }) => , headerRight: () => ( {({ pressed }) => ( <FontAwesome name=“info-circle” size={25} color={Colors[colorScheme ?? ‘light’].text} style={{ marginRight: 15, opacity: pressed ? 0.5 : 1 }} /> )} ), }} /> <Tabs.Screen name=“two” options={{ title: ‘光音控制’, tabBarIcon: ({ color }) => , headerRight: () => ( {({ pressed }) => ( <FontAwesome name=“info-circle” size={25} color={Colors[colorScheme ?? ‘light’].text} style={{ marginRight: 15, opacity: pressed ? 0.5 : 1 }} /> )} ),

}} /> <Tabs.Screen name=“three” options={{ title: ‘用户中心’, tabBarIcon: ({ color }) => , headerRight: () => ( {({ pressed }) => ( <FontAwesome name=“info-circle” size={25} color={Colors[colorScheme ?? ‘light’].text} style={{ marginRight: 15, opacity: pressed ? 0.5 : 1 }} /> )} ),

}} /> ); } import {ActivityIndicator, Image, TextInput,ScrollView, StyleSheet } from ‘react-native’; import Button from ‘@/components/Button’; import React,{useState}from “react”; import Svg, { Circle, // SvgUri, // SvgCssUri, Ellipse, G, TSpan, TextPath, Path, Polygon, Polyline, Line, Rect, Use, Symbol, Defs, LinearGradient, RadialGradient, Stop, ClipPath, Pattern, Mask, } from ‘react-native-svg’; import EditScreenInfo from ‘@/components/EditScreenInfo’; import { Text, View } from ‘@/components/Themed’; import {SvgCssUri} from ‘react-native-svg/css’;

import ImageViewer from “../../components/ImageViewer”; import IconButton from “@/components/CircleButton”;

const Heathimprove= require(“../../assets/images/healthimprove.png”); const Brainwave= require(“../../assets/images/brainwave.png”); const Sleepdata= require(“../../assets/images/sleepdata.png”);

export default function TabOneScreen(props) {

const [selectedImage, setSelectedImage] = useState(null);

const onReset2 = (img) => { setSelectedImage(Brainwave); }; const onReset3 = (img) => { setSelectedImage(Heathimprove); }; const onReset = (img) => { setSelectedImage(Sleepdata); };

return (

<View style={{ padding: 10 }}> <ImageViewer placeholderImageSource={selectedImage ? selectedImage : Heathimprove} />

<TextInput style={{ height: 100,width:300, borderColor: ‘gray’, borderWidth: 1, padding: 5 }} placeholder=“询问关于你的个性化睡眠建议” editable={true} /> <TextInput style={{ height: 100,width:300, borderColor: ‘gray’, borderWidth: 1, padding: 5 }} placeholder=“AI响应将出现在这里…” editable={false} />

);

}

const styles = StyleSheet.create({ row: { padding: 20 , flexDirection: ‘row’, alignItems: ‘center’, justifyContent: ‘space-around’, marginVertical: 10, },

cloumn: { padding: 30, flexDirection: ‘column’, alignItems: ‘center’, justifyContent: ‘space-around’, marginVertical: 10, },

container: { flex: 1, alignItems: ‘center’, justifyContent: ‘center’, }, title: { fontSize: 20, fontWeight: ‘bold’, }, separator: { height: 400, width: ‘100%’, }, imageContainer: { flex: 1, paddingTop: 58, },

image: { width: 320, height: 440, borderRadius: 18, },

scrollContainer: { flexGrow: 1, justifyContent: ‘center’, alignItems: ‘center’, }, }); import { StyleSheet } from ‘react-native’;

import EditScreenInfo from ‘@/components/EditScreenInfo’; import { Text, View } from ‘@/components/Themed’; import {React,useState}from “react”; import { SafeAreaView, Image, TouchableOpacity, TextInput, } from “react-native”; import { StatusBar } from “expo-status-bar”; import LogoBannerImage from “../../assets/login-page/logo.png”; import RegisterForm from “../../components/RegisterForm”;

export default function TabThreeScreen() { const [showAppOptions, setShowAppOptions] = useState(false);

return ( <>

{/\* https://reactnative.dev/docs/image \*/} {showAppOptions ?( ):( <TextInput style={[styles.input, styles.inputUsername]} placeholder=“电话号码或电子邮件” placeholderTextColor=“#cdcdcf” />

<TextInput style={[styles.input, styles.inputPassword]} secureTextEntry={true} placeholder=“密码” placeholderTextColor=“#cdcdcf” />

登录

忘记密码？

或者

<TouchableOpacity style={[styles.button, styles.buttonRegister]} onPress={()=>setShowAppOptions(true)}>  
 <Text style={[styles.buttonText, styles.buttonRegisterText]}>  
 创建新账号  
 </Text>  
 </TouchableOpacity>  
   
  
 </View>  
 </SafeAreaView>  
 )}  
</>

);

}

const styles = StyleSheet.create({ title: { fontSize: 20, fontWeight: ‘bold’, }, separator: { marginVertical: 30, height: 1, width: ‘80%’, },

banner: { resizeMode: “contain”, width: “100%”, height: null, aspectRatio: 750 / 460, // Image ratio }, container: { flex: 1, justifyContent: “space-between”, }, content: { padding: 22, }, input: { borderWidth: 1, borderColor: “#cdcdcf”, color: “#333333”, fontSize: 16, height: 44, paddingHorizontal: 15, }, inputUsername: { borderBottomWidth: 0, borderTopLeftRadius: 3, borderTopRightRadius: 3, }, inputPassword: { borderBottomLeftRadius: 3, borderBottomRightRadius: 3, }, button: { height: 42, borderRadius: 6, backgroundColor: “#1977f3”, justifyContent: “center”, marginVertical: 15, }, buttonText: { color: “#b4cafb”, textAlign: “center”, fontSize: 16, }, link: { paddingVertical: 8, }, linkText: { color: “#1c6ede”, textAlign: “center”, fontSize: 16, fontWeight: “500”, }, footer: { alignItems: “center”, padding: 22, paddingBottom: 0, }, divider: { flexDirection: “row”, alignItems: “center”, width: “70%”, marginBottom: 10, }, dividerLine: { flex: 1, borderBottomWidth: 1, borderColor: “#cbccd0”, }, dividerText: { width: 50, textAlign: “center”, }, buttonRegister: { width: “100%”, backgroundColor: “#e7f3ff”, }, buttonRegisterText: { color: “#1077f7”, },

}); import React from ‘react’; import { StyleSheet, ScrollView, View, Text } from ‘react-native’; import Button from ‘@/components/Button’; import Slider from ‘@react-native-community/slider’;

export default function TabTwoScreen() { const [lightIntensity, setLightIntensity] = React.useState(50); const [soundVolume, setSoundVolume] = React.useState(50); const [pulseFrequency, setPulseFrequency] = React.useState(1); const [pulseDuration, setPulseDuration] = React.useState(30); const [isLightEnabled, setIsLightEnabled] = React.useState(false); const [isSoundEnabled, setIsSoundEnabled] = React.useState(false); const [isPulseEnabled, setIsPulseEnabled] = React.useState(false);

return ( 通用

光调节

<Slider style={{ width: 200 }} minimumValue={0} maximumValue={100} value={lightIntensity} onValueChange={(value) => setLightIntensity(value)} /> <Slider style={{ width: 200 }} minimumValue={0} maximumValue={100} value={lightIntensity} onValueChange={(value) => setLightIntensity(value)} />

音调节

<Slider style={{ width: 200 }} minimumValue={0} maximumValue={100} value={soundVolume} onValueChange={(value) => setSoundVolume(value)} />

脉冲调节

<Slider style={{ width: 200 }} minimumValue={1} maximumValue={10} value={pulseFrequency} onValueChange={(value) => setPulseFrequency(value)} />

); }

const styles = StyleSheet.create({ row: { flexDirection: ‘row’, alignItems: ‘center’, justifyContent: ‘space-around’, marginVertical: 10, }, container: { flex: 1, alignItems: ‘center’, justifyContent: ‘center’, }, title: { fontSize: 20, fontWeight: ‘bold’, }, separator: { marginVertical: 30, height: 1, width: ‘80%’, },

scrollContainer: { flexGrow: 1, justifyContent: ‘center’, alignItems: ‘center’, }, });