

```
// ===== index.js =====
import { AppRegistry } from 'react-native';
import App from './App';
import { name as appName } from './app.json';
AppRegistry.registerComponent(appName, () => App);
// ===== App.tsx =====
import React, { useEffect } from 'react';
import { StyleSheet, StatusBar, Platform, PermissionsAndroid } from 'react-native';
import { SafeAreaView } from 'react-native-safe-area-context';
import { NavigationContainer } from '@react-navigation/native';
import { RootNavigator } from './src/navigation';
import { useAuthStore, useUserStore, useNotificationStore, usePushStore, useChatStore } from './src/store';
import { COLORS } from './src/utils/constants';
import Toast from 'react-native-toast-message';
import { easemobService, toEasemobUsername } from './src/services/easemob';
function App() {
  const { initialize: initAuth } = useAuthStore();
  const { initialize: initUser } = useUserStore();
  const { loadSettings } = useNotificationStore();
  const { initCloud } = usePushStore();
  const { loadSeqIds, syncAll } = useChatStore();
  useEffect(() => {
    async function initialize() {
      const localTasks = [initAuth(), initUser(), loadSettings(), loadSeqIds()];
      await Promise.all(localTasks);
      if (Platform.OS === 'android' && Platform.Version >= 33) {
        try {
          const granted = await PermissionsAndroid.request(
            PermissionsAndroid.PERMISSIONS.POST_NOTIFICATIONS,
            {
              title: '通知权限',
              message: '青核校园需要通知权限来接收消息提醒',
              buttonNeutral: '稍后',
              buttonNegative: '取消',
              buttonPositive: '允许',
            }
          );
          if (granted === PermissionsAndroid.RESULTS.GRANTED) {
            console.log('[Push] Android 通知权限已授予');
          }
        } catch (error) {
          console.warn('[Push] 请求通知权限失败:', error);
        }
      }
    }
    initialize();
    initCloud();
    const authed = useAuthStore.getState().isLoggedIn;
    if (authed) {
      const currentUser = useUserStore.getState().currentUser;
      if (currentUser?.id) {
        console.log('[Push] ★ 缓存登录: 准备 bindAccount, userId=', currentUser.id);
        const { bindAccount } = usePushStore.getState();
        bindAccount(currentUser.id);
      }
    }
  });
}
```

```

    const easemobUid = toEasemobUsername(currentUser.id);
    const easemobPwd = 'campus_' + currentUser.id;
    easemobService.login(easemobUid, easemobPwd).catch(() => {});
  }
  try {
    await syncAll();
  } catch (_) {}
}
}
initialize();
}, [initAuth, initUser, loadSettings, initCloud, loadSeqIds, syncAll]);
return (
  <SafeAreaView style={styles.container}>
    <StatusBar
      barStyle="dark-content"
      backgroundColor={COLORS.background} // 设置 statusBar 背景颜色
      translucent={false}
    />
    <NavigationContainer>
      <RootNavigator />
    </NavigationContainer>
    <Toast />
  </SafeAreaView>
);
}
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: COLORS.background,
  },
});
export default App;
// ===== src/navigation/RootNavigator.tsx =====
import React from 'react';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { useAuthStore } from '../store';
import TabNavigator from '../TabNavigator';
import type { RootStackParamList, AuthStackParamList } from './types';
import LoginPage from '../screens/Auth/LoginPage'; // 登录页
import VerificationPage from '../screens/Auth/VerificationPage';
import ProfileSetupWizard from '../screens/Auth/ProfileSetupWizard'; // 注册, 完成后-继续完善资料
import UserDetail from '../screens/SocialPage/UserDetail'; // 首页-用户详情
import Publish from '../screens/CampusPage/Publish';
import VideoPublish from '../screens/CampusPage/VideoPublish';
import VideoPreviewScreen from '../screens/CampusPage/VideoPreviewScreen';
import ImagePreviewScreen from '../screens/CampusPage/ImagePreviewScreen';
import AnnouncementList from '../screens/Announcement/AnnouncementListPage'; // 首页-公告 banner-公告列表
import AnnouncementDetail from '../screens/Announcement/AnnouncementDetailPage'; // 我的-我的公告-公告详情
import AnnouncementPublish from '../screens/Announcement/AnnouncementPublishPage'; // 我的-我的公告-发布公告
import ActivityGroup from '../screens/Activity/ActivityGroupPage'; // 首页-组局活动-更多
import ActivityDetail from '../screens/Activity/ActivityDetailPage'; // 首页-活动详情 活动详情

```

```
import CreateActivity from '../screens/Activity/CreateActivityPage';//首页-组局活动-更多-发起 我的-我的组局-
发起组局
import CreateGroup from '../screens/Group/CreateGroupPage';
import ChatDetail from '../screens/Chat/ChatDetailPage';
import EphemeralChat from '../screens/Chat/EphemeralChatPage';
import Notifications from '../screens/Chat/NotificationsPage';
import EditProfile from '../screens/Profile/ProfileEditPage';//我的-编辑资料
import Points from '../screens/Profile/PointPage';//我的-我的积分
import FriendsList from '../screens/Social/FriendsListPage';//我的-我的好友
import FriendRequests from '../screens/Social/FriendRequestsPage';//我的-好友申请
import MyActivities from '../screens/Activity/MyActivitiesPage';//我的-我的组局
import MyMoments from '../screens/Moment/MyMomentsPage';//我的-我的动态
import MyAnnouncements from '../screens/Announcement/MyAnnouncementListPage';//我的-我的公告
import VerifiedMap from '../screens/Map/VerifiedMapPage';//我的-认证地图
import AddFriend from '../screens/Social/AddFriendPage';//我的-我的好友-添加好友
import Blacklist from '../screens/Social/BlacklistPage';//我的-我的好友-黑名单
import FriendDetail from '../screens/Social/FriendDetailPage';//我的-我的好友-好友详情
import SoulMatch from '../screens/Social/SoulMatchPage';
import SocialGraph from '../screens/Social/SocialGraphPage';//交友-社交图谱
import MyFriends from '../screens/ProfilePage/MyFriends';
import Settings from '../screens/Profile/SettingsPage';
import Review from '../screens/Profile/ReviewPage';
import Moment from '../screens/Moment/MomentPage';
import Search from '../screens/Discover/SearchPage';
import Nearby from '../screens/Discover/NearbyPage';
import CampusRadar from '../screens/Discover/CampusRadarPage';
import Scan from '../screens/Discover/ScanPage';//首页-添加好友-扫一扫
import SchoolMap from '../screens/Map/SchoolMapPage';
const RootStack = createNativeStackNavigator<RootStackParamList>();
const AuthStack = createNativeStackNavigator<AuthStackParamList>();
function AuthNavigator() {
  return (
    <AuthStack.Navigator screenOptions={{ headerShown: false }}>
      <AuthStack.Screen name="Login" component={LoginPage} />
      <AuthStack.Screen name="Verification" component={VerificationPage} />
      <AuthStack.Screen name="ProfileSetup" component={ProfileSetupWizard} />
    </AuthStack.Navigator>
  );
}
function RootNavigator() {
  const { isLoggedIn } = useAuthStore();
  return (
    <RootStack.Navigator screenOptions={{ headerShown: false }}>
      {isLoggedIn ? (
        <>
          <RootStack.Screen name="Main" component={TabNavigator} />
          <RootStack.Screen name="Publish" component={Publish} options={{ title: '发布' }} />
          <RootStack.Screen name="VideoPublish" component={VideoPublish} options={{ title: '发布视频' }} />
          <RootStack.Screen name="VideoPreview" component={VideoPreviewScreen} options={{ headerShown:
false }} />
          <RootStack.Screen name="ImagePreview" component={ImagePreviewScreen} options={{ headerShown:
false, animation: 'fade' }} />
          <RootStack.Screen name="UserDetail" component={UserDetail} options={{ title: '用户详情' }} />
        </>
      ) : (
        <RootStack.Screen name="Main" component={TabNavigator} />
      )
    </RootStack.Navigator>
  );
}
```

```

    <RootStack.Screen name="AnnouncementList" component={AnnouncementList} options={{ title: '公告列表' }} />
    <RootStack.Screen name="AnnouncementDetail" component={AnnouncementDetail} options={{ title: '公告详情' }} />
    <RootStack.Screen name="AnnouncementPublish" component={AnnouncementPublish} options={{ title: '发布公告' }} />
    <RootStack.Screen name="MyAnnouncements" component={MyAnnouncements} options={{ title: '我的公告' }} />
    <RootStack.Screen name="ActivityGroup" component={ActivityGroup} options={{ title: '活动组局' }} />
    <RootStack.Screen name="ActivityDetail" component={ActivityDetail} options={{ title: '活动详情' }} />
    <RootStack.Screen name="CreateActivity" component={CreateActivity} options={{ title: '发起活动' }} />
    <RootStack.Screen name="MyActivities" component={MyActivities} options={{ title: '我的活动' }} />
    <RootStack.Screen name="CreateGroup" component={CreateGroup} options={{ title: '创建群聊' }} />
    <RootStack.Screen name="ChatDetail" component={ChatDetail} options={{ title: '聊天详情' }} />
    <RootStack.Screen name="EphemeralChat" component={EphemeralChat} options={{ title: '临时聊天' }} />
    <RootStack.Screen name="Notifications" component={Notifications} options={{ title: '通知' }} />
    <RootStack.Screen name="FriendsList" component={FriendsList} options={{ title: '好友列表' }} />
    <RootStack.Screen name="FriendDetail" component={FriendDetail} options={{ title: '好友详情' }} />
    <RootStack.Screen name="FriendRequests" component={FriendRequests} options={{ title: '好友申请' }} />
    <RootStack.Screen name="AddFriend" component={AddFriend} options={{ title: '添加好友' }} />
    <RootStack.Screen name="Blacklist" component={Blacklist} options={{ title: '黑名单' }} />
    <RootStack.Screen name="SoulMatch" component={SoulMatch} options={{ title: '灵魂匹配' }} />
    <RootStack.Screen name="SocialGraph" component={SocialGraph} options={{ title: '社交图谱' }} />
    <RootStack.Screen name="EditProfile" component={EditProfile} options={{ title: '编辑资料' }} />
    <RootStack.Screen name="MyFriends" component={MyFriends} options={{ title: '我的好友' }} />
    <RootStack.Screen name="Settings" component={Settings} options={{ title: '设置' }} />
    <RootStack.Screen name="Points" component={Points} options={{ title: '积分' }} />
    <RootStack.Screen name="Review" component={Review} options={{ title: '审核' }} />
    <RootStack.Screen name="MyMoments" component={MyMoments} options={{ title: '我的动态' }} />
    <RootStack.Screen name="Moment" component={Moment} options={{ title: '动态' }} />
    <RootStack.Screen name="Search" component={Search} options={{ title: '搜索' }} />
    <RootStack.Screen name="Nearby" component={Nearby} options={{ title: '附近' }} />
    <RootStack.Screen name="CampusRadar" component={CampusRadar} options={{ title: '校园雷达' }} />
    <RootStack.Screen name="Scan" component={Scan} options={{ title: '扫一扫' }} />
    <RootStack.Screen name="SchoolMap" component={SchoolMap} options={{ title: '校园地图' }} />
    <RootStack.Screen name="VerifiedMap" component={VerifiedMap} options={{ title: '认证地图' }} />
    <RootStack.Screen name="Verification" component={VerificationPage} options={{ title: '身份认证' }} />
  </>
) : (
  <RootStack.Screen name="Auth" component={AuthNavigator} />
)
</RootStack.Navigator>
);
}
export default RootNavigator;
// ===== src/navigation/TabNavigator.tsx =====
import React from 'react';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import type { MainTabParamList } from './types';
import HomePage from '../screens/HomePage/HomePage';
import SocialPage from '../screens/SocialPage/SocialPage';
import CampusPage from '../screens/CampusPage/CampusPage';
import ChatListPage from '../screens/Chat/ChatListPage';
import ProfilePage from '../screens/ProfilePage/ProfilePage';
import { COLORS } from '../utils/constants';

```

```

import {
  Home,
  Heart,
  Newspaper,
  MessageCircle,
  User,
} from 'lucide-react-native';
import { useNotificationStore } from '../store';
import { useNotificationSSE } from '../hooks/useNotificationSSE';
const Tab = createBottomTabNavigator<MainTabParamList>();
function TabNavigator() {
  const { totalUnreadCount } = useNotificationStore();
  useNotificationSSE();
  const renderIcon = (
    IconComponent: React.FC<{ size?: number; color?: string }>,
  ) => ({ focused, color, size }: { focused: boolean; color: string; size: number }) => (
    <IconComponent
      size={size}
      color={focused ? COLORS.primary : COLORS.textMuted}
    />
  );
  return (
    <Tab.Navigator
      screenOptions={{
        headerShown: false,
        tabBarActiveTintColor: COLORS.primary,
        tabBarInactiveTintColor: COLORS.textMuted,
        tabBarStyle: {
          borderTopWidth: 0.5,
          borderTopColor: COLORS.border,
          paddingBottom: 8,
          paddingTop: 8,
          height: 60,
        },
      }}
    >
      <Tab.Screen name="Home" component={HomePage} options={{ title: '首页', tabBarIcon:
renderIcon(Home), }} />
      <Tab.Screen name="Social" component={SocialPage} options={{ title: '交友', tabBarIcon:
renderIcon(Heart), }} />
      <Tab.Screen name="Campus" component={CampusPage} options={{ title: '动态', tabBarIcon:
renderIcon(Newspaper), }} />
      <Tab.Screen name="Chat" component={ChatListPage} options={{
        title: '消息',
        tabBarIcon: renderIcon(MessageCircle),
        tabBarBadge: totalUnreadCount > 0 ? (totalUnreadCount > 99 ? '99+' : String(totalUnreadCount)) :
undefined,
      }} />
      <Tab.Screen name="Profile" component={ProfilePage} options={{ title: '我的', tabBarIcon:
renderIcon(User), }} />
    </Tab.Navigator>
  );
}
export default TabNavigator;
// ===== src/navigation/index.ts =====
export { default as RootNavigator } from './RootNavigator';

```

```
export { default as TabNavigator } from './TabNavigator';
export type {
  RootStackParamList,
  AuthStackParamList,
  MainTabParamList,
  HomeStackParamList,
  SocialStackParamList,
  CampusStackParamList,
  ProfileStackParamList,
  RootParamList,
} from './types';
// ===== src/navigation/types.ts =====
export interface IProfileSetupData {
  nickname?: string;
  gender?: 'male' | 'female';
  grade?: string;
  universityId?: string;
  universityName?: string;
  college?: string;
  major?: string;
  interests: string[];
}
export type RootStackParamList = {
  Auth: undefined;
  Main: undefined;
  Publish: undefined;
  VideoPublish: undefined;
  VideoPreview: { url: string };
  ImagePreview: { images: string[]; initialIndex?: number };
  UserDetail: { userId: string };
  AnnouncementList: undefined;
  AnnouncementDetail: { announcementId: string };
  AnnouncementPublish: { id?: string };
  MyAnnouncements: undefined;
  ActivityGroup: { status?: string; reviewStatus?: string } | undefined;
  ActivityDetail: { activityId: string };
  CreateActivity: undefined;
  MyActivities: undefined;
  CreateGroup: undefined;
  ChatDetail: { userId?: string; nickname?: string; avatar?: string; groupId?: string; groupName?: string;
groupEasemobId?: string };
  ChatPage: { userId: string; nickname?: string; avatar?: string };
  EphemeralChat: undefined;
  Notifications: undefined;
  FriendsList: undefined;
  FriendDetail: { userId: string };
  FriendRequests: undefined;
  AddFriend: undefined;
  Blacklist: undefined;
  SoulMatch: undefined;
  SocialGraph: undefined;
  EditProfile: undefined;
```

```
MyFriends: undefined;
Settings: undefined;
Points: undefined;
Review: undefined;
MyMoments: undefined;
Moment: undefined;
Search: undefined;
Nearby: undefined;
CampusRadar: undefined;
Scan: undefined;
SchoolMap: undefined;
VerifiedMap: undefined;
Verification: undefined;
};
export type AuthStackParamList = {
  Login: undefined;
  Register: undefined;
  Verification: undefined;
  ProfileSetup: {
    phone?: string;
    universityId?: string;
    universityName?: string;
  };
};
export type MainTabParamList = {
  Home: undefined;
  Social: undefined;
  Campus: undefined;
  Chat: undefined;
  Profile: undefined;
};
export type HomeStackParamList = {
  HomeMain: undefined;
  UserDetail: { userId: string };
};
export type SocialStackParamList = {
  SocialMain: undefined;
  UserDetail: { userId: string };
  MatchList: undefined;
  SoulMatch: undefined;
  SocialGraph: undefined;
  FriendsList: undefined;
  FriendDetail: { userId: string };
  FriendRequests: undefined;
  Blacklist: undefined;
  AddFriend: undefined;
};
export type CampusStackParamList = {
  CampusMain: undefined;
  Publish: undefined;
};
```

```
VideoPublish: undefined;
};
export type ProfileStackParamList = {
  ProfileMain: undefined;
  EditProfile: undefined;
  MyFriends: undefined;
  MyPosts: undefined;
  Points: undefined;
  FriendRequests: undefined;
  MyActivities: undefined;
  MyMoments: undefined;
  MyAnnouncements: undefined;
  VerifiedMap: undefined;
  Settings: undefined;
  Notifications: undefined;
  Review: undefined;
  UserDetail: { userId: string };
  AnnouncementList: undefined;
  AnnouncementDetail: { announcementId: string };
  AnnouncementPublish: { id?: string };
  SchoolMap: undefined;
};
export type DiscoverStackParamList = {
  DiscoverMain: undefined;
  Search: undefined;
  Nearby: undefined;
  CampusRadar: undefined;
  Scan: undefined;
  UserDetail: { userId: string };
  SchoolMap: undefined;
  VerifiedMap: undefined;
};
export type RootParamList = RootStackParamList;
// ===== src/screens/Auth/LoginPage.tsx =====
import React, { useState, useEffect, useCallback } from 'react';
import {
  View, Text, TextInput, TouchableOpacity, StyleSheet, Alert, KeyboardAvoidingView, Platform, ScrollView, Modal,
} from 'react-native';
import type { NativeStackScreenProps } from '@react-navigation/native-stack';
import type { AuthStackParamList } from '../../navigation/types';
import { COLORS, SPACING, BORDER_RADIUS } from '../../utils/constants';
import Button from '../../components/ui/Button';
import { Eye, EyeOff, ChevronRight, Check, X, CheckCircle2 } from 'lucide-react-native';
import Toast from 'react-native-toast-message';
import ToastUtil from '../../utils/toastUtil';
import { useAuth } from '../../hooks/useAuth';
import { register, sendCaptcha, } from '../../api/auth';
import { getSchools } from '../../api/school';
import { ISchool } from '../../types';
import { useAuthStore, useUserStore } from '../../store';
type Props = NativeStackScreenProps<AuthStackParamList, 'Login'>;
const USER_AGREEMENT = `《青核用户协议》\n\n欢迎您使用青核！\n\n一、服务条款的确认\n1. 本协议是您与青核之间关于使用本平台服务的协议。
2. 您使用本平台服务即表示您已阅读并同意本协议的全部内容。
\n\n二、账号注册与安全\n1. 您应当提供真实、准确的个人信息进行注册。
2. 您有责任妥善保管账号密码，对账号下的所有行为负责。
\n\n三、用户行为规范\n1. 遵守法律法规，不得发布违法违规内容。
2. 尊重他人权益，不得侵犯他人隐私和知识产权。
\n\n四、隐私保护\n我们重视您的隐私保护，具体请参见《隐私政策》。`;
const PRIVACY_POLICY = `《青核隐私政策》\n\n一、信息收集\n我们可能收集以下信息：\n1. 注册信息：手机号等基本信息。
2. 使用信息：您使用我们服务时产生的日志信息。
\n\n二、信息使用\n我们使用您的信息用于：\n1. 提供和改进我们的服务。
2. 个性化推荐和内容展示。
\n\n三、信息安全\n我们采取多种安全措施保护您的信息安全。`;
```

```

function LoginPage({ navigation }: Props) {
  const [isLoading, setIsLoading] = useState(false);
  const [isRegisterMode, setIsRegisterMode] = useState(false);
  const { login } = useAuth();
  const [loginType, setLoginType] = useState<'password' | 'code'>('password');//登录类型
  const [phone, setPhone] = useState('');//手机号
  const [password, setPassword] = useState('');//密码
  const [showPassword, setShowPassword] = useState(false);
  const [verifyCode, setVerifyCode] = useState('');//验证码
  const [countdown, setCountdown] = useState(0);//倒计时
  const [showUniModal, setShowUniModal] = useState(false);//学校选择框显示控制
  const [uniSearch, setUniSearch] = useState('');//学校搜索框
  const [schools, setSchools] = useState<ISchool[]>([]);//学校列表
  const [selectedUniversity, setSelectedUniversity] = useState<string>('');//选中的学校
  const [agreed, setAgreed] = useState(false);//是否同意协议
  const [showAgreement, setShowAgreement] = useState(false);//用户协议显示
  const [showPolicy, setShowPolicy] = useState(false);//隐私政策显示
  const [showRegisterSuccess, setShowRegisterSuccess] = useState(false);
  const { setToken, setLogin } = useAuthStore();
  const { setCurrentUser } = useUserStore();
  useEffect(() => {
    fetchSchools();
  }, []);
  useEffect(() => {
    fetchSchools(uniSearch);
  }, [uniSearch]);
  useEffect(() => {
    if (countdown > 0) {
      const timer = setTimeout(() => setCountdown(countdown - 1), 1000);
      return () => clearTimeout(timer);
    }
  }, [countdown]);
  const fetchSchools = async (keyword?: string) => {
    try {
      const res = await getSchools(keyword);
      if (res.code !== 200) {
        ToastUtil.error(res.message);
        return;
      }
      setSchools(res.data.items || []);
    } catch (error) {
      ToastUtil.error('获取学校列表失败');
    }
  };
  const validatePhone = useCallback((p: string): boolean => {
    if (!p) {
      ToastUtil.error('请输入手机号');
      return false;
    }
    if (!/^1[3-9]\d{9}$/.test(p)) {

```

```
    ToastUtil.error('请输入正确的手机号');
    return false;
  }
  return true;
}, []);
const sendCode = useCallback(async () => {
  if (!validatePhone(phone)) return;
  try {
    const res = await sendCaptcha({ phone });
    if (res.code === 200) {
      setCountdown(60);
      ToastUtil.success('验证码已发送');
    } else {
      ToastUtil.error(res.message || '发送失败');
    }
  } catch {
    ToastUtil.error('发送失败，请稍后重试');
  }
}, [phone, validatePhone]);
const handleLogin = useCallback(async () => {
  if (!validatePhone(phone)) return;
  if (loginType === 'password' && !password) {
    ToastUtil.error('请输入密码');
    return;
  }
  if (loginType === 'code' && !verifyCode) {
    ToastUtil.error('请输入验证码');
    return;
  }
  setIsLoading(true);
  try {
    const pass = loginType === 'password' ? password : '';
    const code = loginType === 'code' ? verifyCode : '';
    const res = await login({ phone: phone, password: pass, captcha: code });
    if (res.code !== 200) {
      ToastUtil.error(res.message);
      return;
    }
    ToastUtil.success('登录成功');
  } catch (e: any) {
    ToastUtil.error(e?.message || (loginType === 'password' ? '手机号或密码错误' : '登录失败'));
  } finally {
    setIsLoading(false);
  }
}, [phone, password, verifyCode, loginType, login, validatePhone]);
const handleRegister = useCallback(async () => {
  if (!validatePhone(phone)) return;
  if (!verifyCode) {
    Toast.show({type: 'error', text1: '请输入验证码',});
    return;
  }
}
```

```
    }
    if (!verifyCode) {
      Toast.show({type: 'error',text1: '验证码错误',});
      return;
    }
    if (!password || password.length < 6) {
      Toast.show({type: 'error',text1: '密码至少 6 位',});
      return;
    }
    if (!agreed) {
      Toast.show({type: 'error',text1: '请同意用户协议和隐私政策',});
      return;
    }
    if (!selectedUniversity) {
      Toast.show({type: 'error',text1: '请选择你的学校',});
      return;
    }
    setIsLoading(true);
    try {
      const res = await register({
        phone:phone,
        password:password,
        captcha:verifyCode,
        universityName:selectedUniversity,
      });
      if(res.code !== 200){
        ToastUtil.error(res.message);
        return;
      }
      await setToken(res.data.token);
      await setCurrentUser(res.data.user);
      setShowRegisterSuccess(true);
    } catch (error: any) {
      ToastUtil.error(error?.message || '注册失败');
      return;
    } finally {
      setIsLoading(false);
    }
  }, [phone, verifyCode, password, agreed, selectedUniversity, navigation, validatePhone]);
const toSupplements = () => { //完善个人资料
  navigation.navigate('ProfileSetup', {
    phone,
    universityId: schools.find((u) => u.name === selectedUniversity)?.id,
    universityName: selectedUniversity,
  });
};
const enterAppDirectly = async (phone: string,password: string) => { //注册后已经缓存了 token/user,点击进入, 直接跳转
  await setLogin(true);
  ToastUtil.success('登录成功');
};
```

```

const renderLogin = () => {
  return (
    <>
      <View style={styles.tabRow}>
        <TouchableOpacity
          style={[styles.tab, loginType === 'password' && styles.tabActive]}
          onPress={() => setLoginType('password')}
        >
          <Text style={[styles.tabText, loginType === 'password' && styles.tabTextActive]}>密码登录
        </Text>
      </TouchableOpacity>
      <TouchableOpacity
        style={[styles.tab, loginType === 'code' && styles.tabActive]}
        onPress={() => setLoginType('code')}
      >
        <Text style={[styles.tabText, loginType === 'code' && styles.tabTextActive]}> 验证码登录</Text>
      </TouchableOpacity>
    </View>
    <View style={styles.form}>
      <View style={styles.inputWrapper}>
        <Text style={styles.inputIcon}>+86</Text>
        <TextInput
          style={styles.input}
          placeholder="请输入手机号"
          keyboardType="phone-pad"
          value={phone}
          onChangeText={setPhone}
          maxLength={11}
          placeholderTextColor={COLORS.textMuted}
        />
      </View>
      {loginType === 'password' ? (
        <View style={styles.inputWrapper}>
          <TextInput
            style={[styles.input, { flex: 1 }]}
            placeholder="请输入密码"
            secureTextEntry={!showPassword}
            value={password}
            onChangeText={setPassword}
            placeholderTextColor={COLORS.textMuted}
          />
          <TouchableOpacity onPress={() => setShowPassword(!showPassword)} style={{ marginLeft:
SPACING.sm }}>
            {showPassword ? <EyeOff size={20} color={COLORS.textMuted} /> : <Eye size={20}
color={COLORS.textMuted} />}
          </TouchableOpacity>
        </View>
      ) : (
        <View style={styles.codeRow}>
          <View style={[styles.inputWrapper, { flex: 1 }]}>
            <TextInput
              style={[styles.input, { flex: 1 }]}
              placeholder="请输入验证码"
            />
          </View>
        </View>
      )
    </View>
  )
}

```

```

        keyboardType="number-pad"
        value={verifyCode}
        onChangeText={setVerifyCode}
        maxLength={6}
        placeholderTextColor={COLORS.textMuted}
      />
    </View>
    <TouchableOpacity
      style={[styles.codeBtn, countdown > 0 && styles.codeBtnDisabled]}
      onPress={sendCode}
      disabled={countdown > 0}
    >
      <Text style={styles.codeBtnText}>
        {countdown > 0 ? `${countdown}s` : '获取验证码'}
      </Text>
    </TouchableOpacity>
  </View>
  )}
  <Button title="登录" onPress={handleLogin} loading={isLoading} style={styles.submitBtn} />
  <TouchableOpacity
    style={styles.switchModeBtn}
    onPress={() => {
      setIsRegisterMode(!isRegisterMode);
      setPassword('');
      setVerifyCode('');
    }}
  >
    <Text style={styles.switchModeText}>>没有账号? 去注册</Text>
  </TouchableOpacity>
</View>
</>
)
}
const renderRegister = () => {
  return (
    <>
      <View style={styles.form}>
        <View style={styles.inputWrapper}>
          <Text style={styles.inputIcon}>>+86</Text>
          <TextInput
            style={styles.input}
            placeholder="请输入手机号"
            keyboardType="phone-pad"
            value={phone}
            onChangeText={setPhone}
            maxLength={11}
            placeholderTextColor={COLORS.textMuted}
          />
        </View>
        <View style={styles.inputWrapper}>

```

```

      <TextInput
        style={[styles.input, { flex: 1 }]}
        placeholder="请输入密码"
        secureTextEntry={!showPassword}
        value={password}
        onChangeText={setPassword}
        placeholderTextColor={COLORS.textMuted}
      />
      <TouchableOpacity onPress={() => setShowPassword(!showPassword)} style={{ marginLeft:
SPACING.sm }}>
        {showPassword ? <EyeOff size={20} color={COLORS.textMuted} /> : <Eye size={20}
color={COLORS.textMuted} />}
      </TouchableOpacity>
    </View>
    <View style={styles.codeRow}>
      <View style={[styles.inputWrapper, { flex: 1 }]}>
        <TextInput
          style={[styles.input, { flex: 1 }]}
          placeholder="请输入验证码"
          keyboardType="number-pad"
          value={verifyCode}
          onChangeText={setVerifyCode}
          maxLength={6}
          placeholderTextColor={COLORS.textMuted}
        />
      </View>
      <TouchableOpacity
        style={[styles.codeBtn, countdown > 0 && styles.codeBtnDisabled]}
        onPress={sendCode}
        disabled={countdown > 0}
      >
        <Text style={styles.codeBtnText}>
          {countdown > 0 ? `${countdown}s` : '获取验证码'}
        </Text>
      </TouchableOpacity>
    </View>
    <TouchableOpacity
      style={styles.selectRow}
      onPress={() => setShowUniModal(true)}
    >
      <Text style={styles.selectText}>
        {selectedUniversity || '请选择学校'}
      </Text>
      <ChevronRight size={20} color={COLORS.textMuted} />
    </TouchableOpacity>
    <TouchableOpacity style={styles.agreeRow} onPress={() => setAgreed(!agreed)} >
      <View style={[styles.checkbox, agreed && styles.checkboxChecked]}>
        {agreed && <Check size={12} color="#fff" />}
      </View>
      <Text style={styles.agreeText}>
        我已阅读并同意
      <Text

```

```

        style={styles.linkText}
        onPress={() => setShowAgreement(true)}
      > 《用户协议》 </Text>
      和
      <Text
        style={styles.linkText}
        onPress={() => setShowPolicy(true)}
      > 《隐私政策》 </Text>
    </Text>
  </TouchableOpacity>
  <Button title="注册" onPress={handleRegister} loading={isLoading} style={styles.submitBtn} />
  <TouchableOpacity
    style={styles.switchModeBtn}
    onPress={() => {
      setIsRegisterMode(!isRegisterMode);
      setPassword('');
      setVerifyCode('');
    }}
  >
  <Text style={styles.switchModeText}>已有账号? 去登录</Text>
</TouchableOpacity>
</View>
</>
)
}
const renderRegisterSuccess = () => (
  <View style={styles.registerSuccess}>
    <View style={styles.successModalContent}>
      <View style={styles.successIconWrapper}>
        <CheckCircle2 size={40} color="#22c55e" />
      </View>
      <Text style={styles.successTitle}>注册成功</Text>
      <Text style={styles.successSubtitle}>欢迎加入青核, 开始你的校园之旅</Text>
      <Button
        title="完善资料"
        onPress={toSupplements}
        style={styles.successButton}
      />
      <TouchableOpacity
        onPress={() => enterAppDirectly(phone,password)}
        style={styles.successSkipBtn}
      >
        <Text style={styles.successSkipText}>跳过, 直接进入</Text>
      </TouchableOpacity>
    </View>
  </View>
);
return (
  <KeyboardAvoidingView
    style={styles.container}

```

```

behavior={Platform.OS === 'ios' ? 'padding' : undefined}
>
<ScrollView contentContainerStyle={styles.scrollContent} keyboardShouldPersistTaps="handled">
  <View style={styles.header}>
    <View style={styles.logoCircle}>
      <Text style={styles.logoText}>青</Text>
    </View>
    <Text style={styles.title}>青核</Text>
    <Text style={styles.subtitle}>
      {isRegisterMode ? '创建你的校园账号' : '欢迎回来'}
    </Text>
  </View>
  {!isRegisterMode ? renderLogin() :
    showRegisterSuccess ? renderRegisterSuccess()
    : renderRegister()
  }
</ScrollView>
<Modal
  visible={showUniModal}
  transparent
  animationType="slide"
  onRequestClose={() => setShowUniModal(false)}
>
  <View style={styles.modalOverlay}>
    <View style={styles.modalContent}>
      <View style={styles.modalHeader}>
        <Text style={styles.modalTitle}>选择学校</Text>
        <TouchableOpacity onPress={() => setShowUniModal(false)}>
          <X size={22} color={COLORS.textMuted} />
        </TouchableOpacity>
      </View>
      <TextInput
        style={styles.searchInput}
        placeholder="搜索学校"
        value={uniSearch}
        onChangeText={setUniSearch}
        placeholderTextColor={COLORS.textMuted}
      />
    <ScrollView style={styles.uniList}>
      {schools.map((uni) => (
        <TouchableOpacity
          key={uni.id}
          style={styles.uniItem}
          onPress={() => {
            setSelectedUniversity(uni.name);
            setShowUniModal(false);
            setUniSearch('');
          }}
        >
          <Text style={styles.uniName}>{uni.name}</Text>

```

```

        <Text style={styles.uniCity}>{uni.city}</Text>
      </TouchableOpacity>
    )}}
  </ScrollView>
</View>
</View>
</Modal>
<Modal visible={showAgreement} transparent animationType="slide">
  <View style={styles.modalOverlay}>
    <View style={styles.modalContent}>
      <View style={styles.modalHeader}>
        <Text style={styles.modalTitle}>用户协议</Text>
        <TouchableOpacity onPress={() => setShowAgreement(false)}>
          <X size={22} color={COLORS.textMuted} />
        </TouchableOpacity>
      </View>
      <ScrollView style={styles.textScroll}>
        <Text style={styles.textBody}>{USER_AGREEMENT}</Text>
      </ScrollView>
    </View>
  </View>
</Modal>
<Modal visible={showPolicy} transparent animationType="slide">
  <View style={styles.modalOverlay}>
    <View style={styles.modalContent}>
      <View style={styles.modalHeader}>
        <Text style={styles.modalTitle}>隐私政策</Text>
        <TouchableOpacity onPress={() => setShowPolicy(false)}>
          <X size={22} color={COLORS.textMuted} />
        </TouchableOpacity>
      </View>
      <ScrollView style={styles.textScroll}>
        <Text style={styles.textBody}>{PRIVACY_POLICY}</Text>
      </ScrollView>
    </View>
  </View>
</Modal>
</KeyboardAvoidingView>
);
}
const styles = StyleSheet.create({
  container: { flex: 1, backgroundColor: COLORS.background },
  scrollContent: { flexGrow: 1, paddingHorizontal: SPACING.lg, paddingTop: 60 },
  header: { alignItems: 'center', marginBottom: SPACING.xl },
  logoCircle: {
    width: 72,
    height: 72,
    borderRadius: 36,
    backgroundColor: COLORS.primary,
    justifyContent: 'center',
  }
});

```

```
      alignItems: 'center',
      marginBottom: SPACING.md,
    },
    logoText: { fontSize: 32, fontWeight: '700', color: '#fff' },
    title: { fontSize: 28, fontWeight: '700', color: COLORS.text, marginBottom: SPACING.xs },
    subtitle: { fontSize: 15, color: COLORS.textMuted },
    tabRow: { flexDirection: 'row', marginBottom: SPACING.lg },
    tab: { flex: 1, paddingVertical: SPACING.sm, alignItems: 'center' },
    tabActive: { borderBottomWidth: 2, borderBottomColor: COLORS.primary },
    tabText: { fontSize: 15, color: COLORS.textMuted, fontWeight: '500' },
    tabTextActive: { color: COLORS.primary, fontWeight: '600' },
    form: { gap: SPACING.md },
    inputWrapper: {
      flexDirection: 'row',
      alignItems: 'center',
      backgroundColor: COLORS.card,
      borderRadius: BORDER_RADIUS.card,
      paddingHorizontal: SPACING.md,
      height: 50,
      borderWidth: 1,
      borderColor: COLORS.border,
    },
    inputIcon: { fontSize: 15, color: COLORS.text, marginRight: SPACING.sm, fontWeight: '500' },
    input: { flex: 1, fontSize: 15, color: COLORS.text, padding: 0 },
    eyeIcon: { fontSize: 20, marginLeft: SPACING.sm },
    codeRow: { flexDirection: 'row', gap: SPACING.sm },
    codeBtn: {
      width: 110,
      height: 50,
      borderRadius: BORDER_RADIUS.card,
      backgroundColor: COLORS.primary,
      justifyContent: 'center',
      alignItems: 'center',
    },
    codeBtnDisabled: { backgroundColor: COLORS.border },
    codeBtnText: { color: '#fff', fontSize: 13, fontWeight: '600' },
    selectRow: {
      flexDirection: 'row',
      alignItems: 'center',
      justifyContent: 'space-between',
      backgroundColor: COLORS.card,
      borderRadius: BORDER_RADIUS.card,
      paddingHorizontal: SPACING.md,
      height: 50,
      borderWidth: 1,
      borderColor: COLORS.border,
    },
    selectText: { fontSize: 15, color: COLORS.text },
    arrow: { fontSize: 20, color: COLORS.textMuted },
    agreeRow: { flexDirection: 'row', alignItems: 'flex-start', gap: SPACING.sm, marginTop: SPACING.xs },
```

```
checkbox: {
  width: 20,
  height: 20,
  borderRadius: 4,
  borderWidth: 1,
  borderColor: COLORS.border,
  justifyContent: 'center',
  alignItems: 'center',
  marginTop: 2,
},
checkboxChecked: { backgroundColor: COLORS.primary, borderColor: COLORS.primary },
checkMark: { color: '#fff', fontSize: 12, fontWeight: '700' },
agreeText: { flex: 1, fontSize: 13, color: COLORS.textMuted, lineHeight: 20 },
linkText: { color: COLORS.primary, fontWeight: '500' },
submitBtn: { marginTop: SPACING.md, height: 50 },
switchModeBtn: { alignItems: 'center', marginTop: SPACING.sm, marginBottom: SPACING.xl },
switchModeText: { fontSize: 14, color: COLORS.primary, fontWeight: '500' },
modalOverlay: { flex: 1, backgroundColor: 'rgba(0,0,0,0.4)', justifyContent: 'flex-end' },
modalContent: {
  backgroundColor: COLORS.card,
  borderTopLeftRadius: BORDER_RADIUS.card,
  borderTopRightRadius: BORDER_RADIUS.card,
  maxHeight: '80%',
},
modalHeader: {
  flexDirection: 'row',
  justifyContent: 'space-between',
  alignItems: 'center',
  padding: SPACING.md,
  borderBottomWidth: 1,
  borderBottomColor: COLORS.border,
},
modalTitle: { fontSize: 17, fontWeight: '600', color: COLORS.text },
modalClose: { fontSize: 22, color: COLORS.textMuted, padding: SPACING.xs },
searchInput: {
  margin: SPACING.md,
  backgroundColor: COLORS.background,
  borderRadius: 8,
  paddingHorizontal: SPACING.md,
  height: 44,
  fontSize: 15,
  color: COLORS.text,
},
unilist: { maxHeight: 300 },
uniItem: {
  flexDirection: 'row',
  justifyContent: 'space-between',
  padding: SPACING.md,
  borderBottomWidth: 1,
  borderBottomColor: COLORS.border,
}
```

```
    },
    uniName: { fontSize: 15, color: COLORS.text },
    uniCity: { fontSize: 13, color: COLORS.textMuted },
    textScroll: { padding: SPACING.md },
    textBody: { fontSize: 14, color: COLORS.text, lineHeight: 22 },
    registerSuccess: {
    },
    successModalContent: {
      backgroundColor: COLORS.card,
      borderRadius: BORDER_RADIUS.card,
      padding: SPACING.xl,
      marginHorizontal: SPACING.xs,
      alignItems: 'center',
    },
    successIconWrapper: {
      width: 56,
      height: 56,
      backgroundColor: '#dcfce7',
      borderRadius: 28,
      justifyContent: 'center',
      alignItems: 'center',
      marginBottom: SPACING.lg,
    },
    successTitle: {
      fontSize: 20,
      fontWeight: '600',
      color: COLORS.text,
      marginBottom: SPACING.xs,
    },
    successSubtitle: {
      fontSize: 14,
      color: COLORS.textMuted,
      marginBottom: SPACING.xl,
      textAlign: 'center',
    },
    },
    successButton: {
      width: '100%',
      height: 44,
      marginBottom: SPACING.md,
    },
    },
    successSkipBtn: {
      paddingVertical: SPACING.sm,
    },
    },
    successSkipText: {
      fontSize: 13,
      color: COLORS.textMuted,
    },
    },
  });
export default LoginPage;
// ===== src/screens/Auth/ProfileSetupWizard.tsx =====
import React, { useState, useEffect } from 'react';
```

```

import {
  View,
  Text,
  StyleSheet,
  ScrollView,
  TouchableOpacity,
  TextInput,
  Modal,
  FlatList,
  Alert,
} from 'react-native';
import type { NativeStackScreenProps } from '@react-navigation/native-stack';
import type { AuthStackParamList } from '../..//navigation/types';
import { COLORS, SPACING, BORDER_RADIUS } from '../..//utils/constants';
import Button from '../..//components/ui/Button';
import Card from '../..//components/ui/Card';
import {
  ChevronRight,
  ChevronLeft,
  Check,
  MapPin,
  Heart,
  User,
  BookOpen,
  Briefcase,
} from 'lucide-react-native';
import { updateProfile } from '../..//api/auth'
import { getAreaTree } from '../..//api/area'
import { getSchools, getCollegesByUniversity, getMajorsByCollege } from '../..//api/school'
import { getInterests, type IDictItem } from '../..//api/dict'
import type { IUserInfor } from '../..//types'
import type { IArea } from '../..//types/area'
import type { ISchool, ICollege, IMajor } from '../..//types/school'
import ToastUtil from '../..//utils/toastUtil'
import { useAuthStore } from '../..//store/authStore'
import { useUserStore } from '../..//store'
type Props = NativeStackScreenProps<AuthStackParamList, 'ProfileSetup'>;
interface AreaOption {
  value: string;
  label: string;
}
}
const GRADES = [
  { value: '大一', label: '大一' },
  { value: '大二', label: '大二' },
  { value: '大三', label: '大三' },
  { value: '大四', label: '大四' },
  { value: '研一', label: '研一' },
  { value: '研二', label: '研二' },
  { value: '研三', label: '研三' },
  { value: '博士', label: '博士' },

```

```
];
const STEPS = [
  { title: '基本信息', icon: User, desc: '昵称与性别' },
  { title: '教育信息', icon: BookOpen, desc: '教育形式与学校专业' },
  { title: '个人信息', icon: Heart, desc: '家乡与兴趣' },
];
function ProfileSetupWizard({ navigation, route }: Props) {
  const [currentStep, setCurrentStep] = useState(0);
  const [formData, setFormData] = useState<IUserInfo>({
    interests: [],
    universityId: route.params?.universityId,
    universityName: route.params?.universityName,
  });
  const [interestTags, setInterestTags] = useState<IDictItem[]>([]);
  const [modalOpen, setModalOpen] = useState(false);
  const [modalConfig, setModalConfig] = useState<{
    title: string;
    options: { value: string; label: string; city?: string }[]; // 选项列表
    value: string; // 当前选中的值
    onSelect: (value: string, label: string) => void; // 选择回调
    placeholder: string; // 提示符
  } | null>(null);
  const [searchValue, setSearchValue] = useState('');
  const { setLogin } = useAuthStore();
  const [provinces, setProvinces] = useState<AreaOption[]>([]);
  const [areaTree, setAreaTree] = useState<IArea[]>([]); // 省市区县
  const [universities, setUniversities] = useState<ISchool[]>([]);
  const [colleges, setColleges] = useState<ICollege[]>([]);
  const [majors, setMajors] = useState<IMajor[]>([]);
  useEffect(() => {
    fetchAreaTree();
    fetchUniversities();
    fetchInterests();
  }, []);
  useEffect(() => {
    if (formData.universityId) {
      fetchColleges(formData.universityId);
    }
  }, [formData.universityId]);
  useEffect(() => {
    if (formData.collegeId) {
      fetchMajors(formData.collegeId);
    }
  }, [formData.collegeId]);
  const fetchInterests = async () => {
    try {
      const tags = await getInterests();
      setInterestTags(tags);
    } catch (error) {
      console.error('获取兴趣标签失败:', error);
    }
  }
}
```

```
    }
  };
  const fetchAreaTree = async () => {
    try {
      const res = await getAreaTree();
      if (res.code === 200 && res.data && res.data.length > 0) {
        const chinaData = res.data[0];
        setAreaTree(chinaData.children || []);
        setProvinces((chinaData.children || []).map(p => ({
          value: String(p.areaId),
          label: p.name,
        })));
      }
    } catch (error) {
      console.error('获取地区数据失败:', error);
    }
  };
  const fetchUniversities = async (keyword?: string) => {
    try {
      const res = await getSchools(keyword);
      if (res.code === 200 && res.data && res.data.items.length > 0) {
        setUniversities(res.data.items);
      }
    } catch (error) {
      console.error('获取学校数据失败:', error);
    }
  };
  const getCities = (): AreaOption[] => {
    if (formData.hometownProvince) {
      const province = areaTree.find(p => String(p.name) === formData.hometownProvince);
      return province?.children?.map(c => ({
        value: String(c.areaId),
        label: c.name,
      })) || [];
    }
    return [];
  };
  const getDistricts = (): AreaOption[] => {
    if (formData.hometownCity) {
      for (const province of areaTree) {
        const city = province.children?.find(c => String(c.name) === formData.hometownCity);
        if (city) {
          return city.children?.map(d => ({
            value: String(d.areaId),
            label: d.name,
          })) || [];
        }
      }
    }
    return [];
  };
}
```

```
};
const fetchColleges = async (universityId: string) => {
  if (!universityId) {
    setColleges([]);
    return;
  }
  try {
    const res = await getCollegesByUniversity({ id: universityId });
    if (res.code === 200 && res.data) {
      setColleges(res.data);
    } else {
      setColleges([]);
    }
  } catch (error) {
    console.error('获取学院数据失败:', error);
    setColleges([]);
  }
};
const fetchMajors = async (collegeId: string) => {
  if (!collegeId) {
    setMajors([]);
    return;
  }
  try {
    const res = await getMajorsByCollege({ collegeId });
    if (res.code === 200 && res.data) {
      setMajors(res.data);
    } else {
      setMajors([]);
    }
  } catch (error) {
    console.error('获取专业数据失败:', error);
    setMajors([]);
  }
};
const toggleInterest = (id: string) => {
  setFormData(prev => ({
    ...prev,
    interests: prev.interests?.includes(id)
      ? prev.interests.filter(i => i !== id)
      : [...(prev.interests || []), id].slice(0, 8),
  })));
};
const handleNext = () => {
  if (currentStep === 1 && !formData.universityId) {
    ToastUtil.error('请选择您的大学');
    return;
  }
  if (currentStep < STEPS.length - 1) {
    setCurrentStep(currentStep + 1);
  }
}
```

```
    } else {
      const profileData: IUserInfor = {
        nickname: formData.nickname || '',
        gender: formData.gender || '',
        educationType: formData.educationType || '',
        universityId: formData.universityId || '',
        universityName: formData.universityName || '',
        collegeId: formData.collegeId || '',
        collegeName: formData.collegeName || '',
        majorId: formData.majorId || '',
        majorName: formData.majorName || '',
        secondMajorName: formData.secondMajorName,
        grade: formData.grade || '',
        highSchool: formData.highSchool || '',
        hometownProvince: formData.hometownProvince || '',
        hometownCity: formData.hometownCity || '',
        hometownDistrict: formData.hometownDistrict || '',
        interests: formData.interests,
      };
      updateProfile(profileData)
        .then((res) => {
          if (res.code !== 200) {
            ToastUtil.error(res.message || '资料更新失败, 请稍后重试');
            return;
          }
          ToastUtil.success('资料设置成功');
          useUserStore.getState().updateProfile({
            nickname: formData.nickname,
            gender: formData.gender,
            universityName: formData.universityName,
            majorName: formData.majorName,
            grade: formData.grade,
            interests: formData.interests,
            hometown: formData.hometownCity || formData.hometownProvince,
          });
          setLogin(true);
        })
        .catch((error) => {
          console.error('更新资料失败:', error);
          ToastUtil.error('资料更新失败, 请稍后重试');
        });
    }
  };
  const handlePrev = () => {
    if (currentStep > 0) {
      setCurrentStep(currentStep - 1);
    } else {
      navigation.goBack();
    }
  };
};
```

```

const openSelectModal = (
  title: string,
  options: { value: string; label: string }[],
  value: string,
  onSelect: (value: string, label: string) => void,
  placeholder: string
) => {
  setModalConfig({ title, options, value, onSelect, placeholder });
  setSearchValue('');
  setModalOpen(true);
  if (title === '大学') {
    fetchUniversities();
  }
};

const filteredOptions = modalConfig
  ? modalConfig.title === '大学'
    ? universities.map(u => ({ value: u.id, label: u.name, city: u.city }))
    : modalConfig.options.filter(opt =>
      opt.label.toLowerCase().includes(searchValue.toLowerCase())
    )
  : [];

const renderStepIndicator = () => (
  <View style={styles.stepIndicatorContainer}>
    <View style={styles.stepRow}>
      {STEPS.map((step, index) => (
        <React.Fragment key={index}>
          <View style={styles.stepItem}>
            <View
              style={[
                styles.stepCircle,
                index < currentStep && styles.stepCircleCompleted,
                index === currentStep && styles.stepCircleActive,
              ]}
            >
              {index < currentStep ? (
                <Check size={16} color="#fff" />
              ) : (
                <Text style={styles.stepNumber}>{index + 1}</Text>
              )}
            </View>
          </View>
          {index < STEPS.length - 1 && (
            <View
              style={[
                styles.stepLine,
                index < currentStep && styles.stepLineActive,
              ]}
            />
          )}
        </React.Fragment>
      )}
    </View>
  )

```

```

    )))
  </View>
  <View style={styles.stepInfo}>
    <Text style={styles.stepTitle}>{STEPS[currentStep].title}</Text>
    <Text style={styles.stepDesc}>{STEPS[currentStep].desc}</Text>
  </View>
</View>
);
const renderSelectButton = (
  label: string, // 标签名称
  displayValue: string, // 显示值
  placeholder: string, // 占位符
  options: { value: string; label: string }[], // 选项列表
  value: string, // 当前值
  onSelect: (value: string, label: string) => void // 选择回调
) => (
  <View style={styles.selectContainer}>
    <Text style={styles.label}>{label}</Text>
    <TouchableOpacity
      style={styles.selectButton}
      onPress={() => openSelectModal(label, options, value, onSelect, placeholder)}
    >
      <Text style={[[styles.selectText, displayValue ? styles.selectTextFilled :
styles.selectTextEmpty]]}>
        {displayValue || placeholder}
      </Text>
      <ChevronRight size={14} color={COLORS.textMuted} />
    </TouchableOpacity>
  </View>
);
const renderStep0 = () => (
  <View style={styles.stepContent}>
    <View style={styles.fieldContainer}>
      <Text style={styles.label}>昵称</Text>
      <TextInput
        style={styles.input}
        placeholder="给自己取个好听的昵称"
        value={formData.nickname || ''}
        onChangeText={(text) =>
          setFormData(prev => ({ ...prev, nickname: text }))
        }
        maxLength={20}
      />
    </View>
    <View style={styles.fieldContainer}>
      <Text style={styles.label}>性别</Text>
      <View style={styles.genderRow}>
        <TouchableOpacity
          style={[
            styles.genderBtn,
            formData.gender === 'male' && styles.genderBtnActive,

```

```

    }}
    onPress={() =>
      setFormData(prev => ({ ...prev, gender: 'male' }))
    }
  >
    <Text style={styles.genderEmoji}>👦</Text>
    <Text style={styles.genderLabel}>男生</Text>
  </TouchableOpacity>
  <TouchableOpacity
    style={[
      styles.genderBtn,
      formData.gender === 'female' && styles.genderBtnActive,
    ]>
    onPress={() =>
      setFormData(prev => ({ ...prev, gender: 'female' }))
    }
  </TouchableOpacity>
  </View>
</View>
</View>
);
const renderStep1 = () => (
  <View style={styles.stepContent}>
    <View style={styles.fieldContainer}>
      <Text style={styles.label}>教育形式</Text>
      <View style={styles.educationRow}>
        <TouchableOpacity
          style={[
            styles.educationBtn,
            formData.educationType === 'fulltime' && styles.educationBtnActive,
          ]>
          onPress={() =>
            setFormData(prev => ({ ...prev, educationType: 'fulltime' }))
          }
        >
          <Text style={[styles.educationLabel, formData.educationType === 'fulltime' &&
styles.educationBtnActiveText]}>全日制</Text>
        </TouchableOpacity>
        <TouchableOpacity
          style={[
            styles.educationBtn,
            formData.educationType === 'parttime' && styles.educationBtnActive,
          ]>
          onPress={() =>
            setFormData(prev => ({ ...prev, educationType: 'parttime' }))
          }
        >
          <Text style={[styles.educationLabel, formData.educationType === 'parttime' &&
styles.educationBtnActiveText]}>非全日制</Text>
        </TouchableOpacity>
      </View>
    </View>
  </View>
);

```

```

    </TouchableOpacity>
  </View>
  {formData.educationType === 'parttime' && (
    <View style={styles.parttimeSection}>
      <View style={styles.parttimeHeader}>
        <Briefcase size={16} color={COLORS.primary} />
        <Text style={styles.parttimeTitle}>请填写您的工作信息</Text>
      </View>
      <View style={styles.fieldContainer}>
        <Text style={styles.label}>工作单位</Text>
        <TextInput
          style={styles.smallInput}
          placeholder="请输入您的工作单位名称"
          value={formData.companyName || ''}
          onChangeText={(text) =>
            setFormData(prev => ({ ...prev, companyName: text }))
          }
        />
      </View>
      <View style={styles.fieldContainer}>
        <Text style={styles.label}>所属行业</Text>
        <View style={styles.industryRow}>
          {[ '互联网/IT', '金融', '教育', '医疗', '制造', '房地产', '贸易', '其他' ].map((industry) => (
            <TouchableOpacity
              key={industry}
              style={[
                styles.industryBtn,
                formData.companyIndustry === industry && styles.industryBtnActive,
              ]}
              onPress={() =>
                setFormData(prev => ({ ...prev, companyIndustry: industry }))
              }
            >
              <Text style={ [styles.industryLabel, formData.companyIndustry === industry &&
styles.industryBtnActiveText]}>{industry}</Text>
            </TouchableOpacity>
          ))}
        </View>
      </View>
      <View style={styles.fieldContainer}>
        <Text style={styles.label}>岗位名称</Text>
        <TextInput
          style={styles.smallInput}
          placeholder="请输入您的岗位名称"
          value={formData.jobPosition || ''}
          onChangeText={(text) =>
            setFormData(prev => ({ ...prev, jobPosition: text }))
          }
        />
      </View>
    </View>
  )
}

```

```

    })
  </View>
  {renderSelectButton(
    '大学',
    formData.universityName || '',
    '选择或搜索大学',
    universities.map(u => ({ value: u.id, label: u.name, city: u.city })),
    formData.universityId || '',
    (value, label) => {
      setFormData(prev => ({
        ...prev,
        universityId: value,
        universityName: label,
        collegeId: undefined,
        collegeName: undefined,
        majorId: undefined,
        majorName: undefined,
      }));
      fetchColleges(value);
    })
  }
})
{renderSelectButton(
  '学院',
  formData.collegeName || '',
  formData.collegeId ? '选择学院' : '请先选择大学',
  colleges.map(c => ({ value: c.id, label: c.name })),
  formData.collegeId || '',
  (value, label) => {
    setFormData(prev => ({
      ...prev,
      collegeId: value,
      collegeName: label,
      majorId: undefined,
      majorName: undefined,
    }));
    fetchMajors(value);
  })
}
{renderSelectButton(
  '专业',
  formData.majorName || '',
  formData.majorId ? '选择专业' : '请先选择学院',
  majors.map(m => ({ value: m.id, label: m.name })),
  formData.majorId || '',
  (value, label) => setFormData(prev => ({ ...prev, majorId: value, majorName: label}))
})
<View style={styles.fieldContainer}>
  <Text style={styles.label}>年级</Text>
  <View style={styles.gradeGrid}>
    {GRADES.map((grade) => (

```

```

    {QUICK_FILTER_TAGS.map((tag) => {
      const isActive = quickFilters.includes(tag.key);
      if (isActive) {
        return (
          <React.Fragment key={tag.key}>
            <Gradient
              colors={[...tag.gradient]}
              borderRadius={BORDER_RADIUS.full}
            >
              <TouchableOpacity
                style={st.quickTagActive}
                activeOpacity={0.85}
                onPress={() => toggleQuickFilter(tag.key)}
              >
                <Text style={st.quickTagTextActive}>{tag.label}</Text>
              </TouchableOpacity>
            </Gradient>
          </React.Fragment>
        );
      }
    })
  }
  return (
    <TouchableOpacity
      key={tag.key}
      style={st.quickTag}
      activeOpacity={0.85}
      onPress={() => toggleQuickFilter(tag.key)}
    >
      <Text style={st.quickTagText}>{tag.label}</Text>
    </TouchableOpacity>
  );
}
}
</ScrollView>
<View style={st.searchRow}>
  <View style={st.searchInputWrap}>
    <Search size={14} color={COLORS.textMuted} />
    <TextInput
      style={st.searchInput}
      placeholder="搜索昵称、学校、专业..."
      placeholderTextColor={COLORS.textMuted}
      value={searchQuery}
      onChangeText={setSearchQuery}
    />
  </View>
  <TouchableOpacity
    style={[st.filterBtn, activeFilterCount > 0 && st.filterBtnActive]}
    activeOpacity={0.85}
    onPress={() => setShowFilterModal(true)}
  >
    <SlidersHorizontal size={16} color={activeFilterCount > 0 ? '#fff' : COLORS.textMuted} />
    {activeFilterCount > 0 && (

```

```

        <View style={st.filterBadge}>
          <Text style={st.filterBadgeText}>{activeFilterCount}</Text>
        </View>
      )}
    </TouchableOpacity>
  </View>
</View>
<View style={st.listSection}>
  <View style={st.listHeader}>
    <View style={st.listTitleRow}>
      <Users size={14} color={COLORS.text} />
      <Text style={st.listTitle}>
        推荐同学
      <Text style={st.listCount}> ({filteredUsers.length})</Text>
    </Text>
  </View>
</View>
</View>
{isLoading ? (
  <View style={st.skeletonContainer}>
    {[1, 2, 3, 4, 5, 6].map((i) => (
      <View key={i} style={st.skeletonCardWrapper}>
        {renderSkeletonCard()}
      </View>
    ))}
  </View>
) : filteredUsers.length === 0 ? (
  <View style={st.emptyState}>
    <Text style={st.emptyText}>没有找到符合条件的用户</Text>
  </View>
) : (
  <FlatList
    data={filteredUsers}
    renderItem={renderItem}
    keyExtractor={keyExtractor}
    contentContainerStyle={st.listContent}
    showsVerticalScrollIndicator={false}
    removeClippedSubviews={true}
    maxToRenderPerBatch={10}
    windowSize={5}
    initialNumToRender={6}
    onEndReached={loadMore}
    onEndReachedThreshold={0.3}
    ListFooterComponent={
      isLoadingMore ? (
        <View style={st.footerLoader}>
          <Text style={st.footerLoaderText}>加载中...</Text>
        </View>
      ) : hasMore ? null : filteredUsers.length > 0 ? (
        <View style={st.footerLoader}>
          <Text style={st.footerLoaderText}>已经到底了</Text>
        </View>
      ) : null
    }
  >

```

```

        </View>
      ) : null
    }
  />
)}
</View>
<FilterDrawer
  visible={showFilterModal}
  filters={filters}
  filteredCount={filteredUsers.length}
  hasActiveFilters={hasActiveFilters}
  onClose={() => setShowFilterModal(false)}
  onFilterChange={handleFilterChange}
  onSchoolTextChange={handleSchoolText}
  onMajorTextChange={handleMajorText}
  onClear={clearFilters}
  schoolOptions={schoolOptions}
/>
</View>
);
};
const st = StyleSheet.create({
  container: { flex: 1, backgroundColor: COLORS.background },
  topBar: {
    backgroundColor: COLORS.card,
    paddingHorizontal: SPACING.md,
    paddingTop: SPACING.sm,
    paddingBottom: SPACING.sm,
    borderBottomWidth: 1,
    borderBottomColor: COLORS.border,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.05,
    shadowRadius: 4,
    elevation: 2,
  },
  quickEntryRow: { flexDirection: 'row', gap: SPACING.sm, marginBottom: SPACING.sm },
  quickEntryBtn: { flex: 1, borderRadius: BORDER_RADIUS.button, overflow: 'hidden' },
  quickEntryBtnInner: {
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'center',
    gap: 6,
    paddingVertical: SPACING.sm,
  },
  quickEntryBtnText: { color: '#fff', fontSize: 12, fontWeight: '600' },
  quickFilterRow: { gap: SPACING.sm, marginBottom: SPACING.sm },
  quickTag: {
    paddingHorizontal: 12,
    paddingVertical: 6,
  },
});

```

```
borderRadius: BORDER_RADIUS.full,
backgroundColor: COLORS.muted,
},
quickTagActive: {
paddingHorizontal: 12,
paddingVertical: 6,
borderRadius: BORDER_RADIUS.full,
},
quickTagText: { fontSize: 12, fontWeight: '500', color: COLORS.mutedForeground },
quickTagTextActive: { fontSize: 12, fontWeight: '500', color: '#fff' },
searchRow: { flexDirection: 'row', alignItems: 'center', gap: SPACING.sm },
searchInputWrap: {
flex: 1,
flexDirection: 'row',
alignItems: 'center',
backgroundColor: `${COLORS.muted}60`,
borderRadius: BORDER_RADIUS.button,
paddingHorizontal: SPACING.sm,
height: 36,
},
searchInput: { flex: 1, fontSize: 13, color: COLORS.text, paddingVertical: 0, marginLeft: 4 },
filterBtn: {
width: 36,
height: 36,
borderRadius: BORDER_RADIUS.button,
backgroundColor: COLORS.muted,
justifyContent: 'center',
alignItems: 'center',
position: 'relative',
},
filterBtnActive: { backgroundColor: COLORS.primary },
filterBadge: {
position: 'absolute',
top: -4,
right: -4,
width: 16,
height: 16,
borderRadius: 8,
backgroundColor: COLORS.red,
justifyContent: 'center',
alignItems: 'center',
},
filterBadgeText: { color: '#fff', fontSize: 10, fontWeight: '700' },
listSection: { flex: 1, paddingHorizontal: SPACING.md },
listHeader: { paddingVertical: SPACING.sm },
listTitleRow: { flexDirection: 'row', alignItems: 'center', gap: 4 },
listTitle: { fontSize: 13, fontWeight: '600', color: COLORS.text },
listCount: { fontSize: 11, color: COLORS.textMuted, fontWeight: '400' },
listContent: { paddingBottom: SPACING.xl },
userCard: {
```

```
    backgroundColor: COLORS.card,
    borderRadius: BORDER_RADIUS.card,
    marginBottom: SPACING.sm,
    padding: 10,
  },
  userCardInner: { flexDirection: 'row', alignItems: 'center' },
  userAvatarWrap: {
    marginRight: 10,
    position: 'relative',
  },
  identityBadge: {
    position: 'absolute',
    bottom: -2,
    right: -2,
    width: 16,
    height: 16,
  },
  userInfo: { flex: 1 },
  userNameRow: { flexDirection: 'row', alignItems: 'center', gap: 6, marginBottom: 2 },
  userName: { fontSize: 14, fontWeight: '600', color: COLORS.text, flexShrink: 1 },
  matchBadge: {
    flexDirection: 'row',
    alignItems: 'center',
    gap: 2,
    paddingHorizontal: 4,
    paddingVertical: 1.5,
    backgroundColor: `${COLORS.primary}10`,
    borderRadius: 4,
  },
  matchBadgeText: { fontSize: 9, fontWeight: '500', color: COLORS.primary },
  userSchoolRow: { flexDirection: 'row', alignItems: 'center', gap: 2, marginBottom: 3 },
  userSchool: { fontSize: 11, color: COLORS.textMuted, flexShrink: 1 },
  userTags: { flexDirection: 'row', flexWrap: 'wrap', gap: 3 },
  userTag: {
    fontSize: 9,
    paddingHorizontal: 6,
    paddingVertical: 1.5,
    backgroundColor: COLORS.muted,
    borderRadius: 4,
    color: COLORS.mutedForeground,
  },
  greetBtn: {
    width: 32,
    height: 32,
    borderRadius: 16,
    backgroundColor: `${COLORS.primary}15`,
    justifyContent: 'center',
    alignItems: 'center',
  },
  greetBtnDone: { backgroundColor: `${COLORS.success}15` },
```

```
emptyState: {
  backgroundColor: COLORS.card,
  borderRadius: BORDER_RADIUS.card,
  paddingVertical: SPACING.xl * 2,
  alignItems: 'center',
  marginTop: SPACING.md,
},
emptyText: { fontSize: 13, color: COLORS.textMuted },
skeletonContainer: {
  marginTop: SPACING.sm,
  gap: SPACING.sm,
},
skeletonCardWrapper: {
  backgroundColor: COLORS.card,
  borderRadius: BORDER_RADIUS.card,
  padding: 10,
  shadowColor: '#000',
  shadowOffset: { width: 0, height: 1 },
  shadowOpacity: 0.05,
  shadowRadius: 3,
  elevation: 2,
},
skeletonCard: {
  flexDirection: 'row',
  alignItems: 'center',
  gap: 10,
},
skeletonAvatar: {
  width: 44,
  height: 44,
  borderRadius: 22,
  backgroundColor: COLORS.muted,
},
skeletonInfo: {
  flex: 1,
  gap: 4,
},
skeletonName: {
  height: 16,
  width: '40%',
  backgroundColor: COLORS.muted,
  borderRadius: 4,
},
skeletonSchool: {
  height: 12,
  width: '60%',
  backgroundColor: COLORS.muted,
  borderRadius: 4,
},
skeletonTags: {
```

```

        flexDirection: 'row',
        gap: 4,
      },
      skeletonTag: {
        height: 12,
        width: 24,
        backgroundColor: COLORS.muted,
        borderRadius: 4,
      },
      skeletonButton: {
        width: 32,
        height: 32,
        borderRadius: 16,
        backgroundColor: COLORS.muted,
      },
      footerLoader: {
        paddingVertical: SPACING.md,
        alignItems: 'center',
      },
      footerLoaderText: {
        fontSize: 12,
        color: COLORS.textMuted,
      },
    });
export default SocialPage;
// ===== src/screens/SocialPage/UserDetail.tsx =====
import React, { useState, useEffect } from 'react';
import {
  View,
  Text,
  StyleSheet,
  ScrollView,
  TouchableOpacity,
  Image,
  Alert,
  Dimensions,
  ActivityIndicator,
} from 'react-native';
import LinearGradient from 'react-native-linear-gradient';
import { useNavigation, useRoute, RouteProp } from '@react-navigation/native';
import { NativeStackNavigationProp } from '@react-navigation/native-stack';
import { RootStackParamList } from '../../navigation/types';
import Avatar from '../../components/ui/Avatar';
import Badge from '../../components/ui/Badge';
import { COLORS, BORDER_RADIUS, SPACING } from '../../utils/constants';
import { getFriendProfile, acceptFriendRequestByUserId, addFriend } from '../../api/friendship_full';
import { IFriendDetail } from '../../types/friendship';
import type { IApiResponse } from '../../types/common';
import { ChevronLeft, Check, MessageCircle, GraduationCap, MapPin, BookOpen, Users, Shield } from 'lucide-react-native';
import ToastUtil from '../../utils/toastUtil';
type NavigationProp = NativeStackNavigationProp<RootStackParamList>;

```

```
type RouteProps = RouteProp<RootStackParamList, 'UserDetail'>;
const { width: SCREEN_WIDTH } = Dimensions.get('window');
interface IFriend {
  id: string;
  nickname: string;
  avatar: string;
  major: string;
}
const UserDetails: React.FC = () => {
  const navigation = useNavigation<NavigationProp>();
  const route = useRoute<RouteProps>();
  const { userId } = route.params;
  const [user, setUser] = useState<IFriendDetail | null>(null);
  const [isLoading, setIsLoading] = useState(false);
  const [isFriendRequested, setIsFriendRequested] = useState(false);
  const [isFriend, setIsFriend] = useState(false);
  const [hasReceivedRequest, setHasReceivedRequest] = useState(false);
  useEffect(() => {
    const fetchUserProfile = async () => {
      try {
        const response = await getFriendProfile({ userId });
        if (response.data) {
          setUser(response.data);
          setIsFriend(response.data.isMutualFollow || response.data.relationStatus === 'ACCEPTED');
          setIsFriendRequested(response.data.relationStatus === 'PENDING');
          setHasReceivedRequest(response.data.relationStatus === 'RECEIVED');
        }
      } catch (error: any) {
        ToastUtil.error(error.message || '获取用户信息失败');
      } finally {
        setIsLoading(false);
      }
    };
    fetchUserProfile();
  }, [userId]);
  const handleAddFriend = async () => {
    setIsLoading(true);
    try {
      const res = await addFriend({ friendId: userId });
      if (res.code === 200) {
        setIsFriendRequested(true);
        ToastUtil.success('好友申请已发送，等待对方确认');
      } else {
        Alert.alert('提示', res.message || '发送好友申请失败');
        if (res.message?.includes('已存在') || res.message?.includes('待处理')) {
          setIsFriendRequested(true);
        }
      }
    } catch (error) {
      const errMsg = error instanceof Error ? error.message : '';
    }
  };
};
```

```
    if (errMsg.includes('已存在') || errMsg.includes('待处理')) {
      setIsFriendRequested(true);
      Alert.alert('提示', '已存在好友关系或待处理申请');
    } else {
      Alert.alert('错误', '网络异常, 请稍后再试');
    }
  } finally {
    setIsLoading(false);
  }
};

const handleAcceptFriend = async () => {
  setIsLoading(true);
  try {
    const res = await acceptFriendRequestByUserId(userId);
    if (res.code === 200) {
      setHasReceivedRequest(false);
      setIsFriend(true);
      Alert.alert('提示', '已成功添加好友');
    } else {
      Alert.alert('提示', res.message || '同意好友申请失败');
    }
  } catch (error) {
    ToastUtil.error((error as Error).message || '添加好友接口异常');
  } finally {
    setIsLoading(false);
  }
};

const handleMessage = () => {
  if (isFriend) {
    Alert.alert('聊天', `打开与 ${user?.nickname} 的聊天`);
  }
};

if (isLoading) {
  return (
    <View style={styles.loadingContainer}>
      <ActivityIndicator size="large" color={COLORS.primary} />
    </View>
  );
}

return (
  <View style={styles.container}>
    <ScrollView showsVerticalScrollIndicator={false}>
      <View style={styles.coverArea}>
        <Image
          source={{ uri: user?.avatar || '' }}
          style={styles.coverImage}
          resizeMode="cover"
        />
        <View style={styles.coverOverlay} />
        <TouchableOpacity
```

```

        style={styles.backBtn}
        activeOpacity={0.7}
        onPress={() => navigation.goBack()}
      >
        <ChevronLeft size={24} color="#fff" />
      </TouchableOpacity>
    </View>
    <View style={styles.profileArea}>
      <View style={styles.profileHeader}>
        <View style={styles.avatarContainer}>
          <Avatar name={user?.nickname} size={96} />
          {isFriend && (
            <View style={styles.friendBadge}>
              <Check size={12} color="#fff" />
            </View>
          )}
        </View>
      </View>
      <View style={styles.profileInfo}>
        <View style={{ flexDirection: 'row', alignItems: 'center', gap: 6 }}>
          <Text style={styles.nickname}>{user?.nickname}</Text>
          {user?.xuexinCertified && (
            <Badge
              label="已认证"
              variant="success"
              textStyle={{ fontSize: 10, paddingVertical: 2, paddingHorizontal: 6 }}
            />
          )}
        </View>
        <Text style={styles.schoolGrade}>
          {user?.universityName} · {user?.grade}
        </Text>
      </View>
      <View style={styles.actionButtons}>
        {!isFriend && !isFriendRequested && !hasReceivedRequest && (
          <TouchableOpacity
            style={[
              styles.addFriendBtn,
              isLoading && styles.addFriendBtnDisabled,
            ]}
            activeOpacity={0.7}
            onPress={handleAddFriend}
            disabled={isLoading}
          >
            <LinearGradient
              colors={['#17a3bf', '#06b6d4']}
              start={{ x: 0, y: 0 }}
              end={{ x: 1, y: 0 }}
              style={StyleSheet.absoluteFill}
            />
            <View style={styles.iconRow}>

```

```

        <Text style={styles.addFriendBtnText}>+ 加好友</Text>
      </View>
    </TouchableOpacity>
  )}
  {isFriendRequested && !isFriend && (
    <TouchableOpacity
      style={[styles.addFriendBtn, styles.addFriendBtnDisabled]}
      activeOpacity={0.7}
      disabled
    >
      <LinearGradient
        colors={['#9ca3af', '#6b7280']}
        start={{ x: 0, y: 0 }}
        end={{ x: 1, y: 0 }}
        style={StyleSheet.absoluteFill}
      />
      <View style={styles.iconRow}>
        <Check size={13} color="#fff" />
        <Text style={styles.addFriendBtnText}> 等待同意</Text>
      </View>
    </TouchableOpacity>
  )}
  {hasReceivedRequest && !isFriend && (
    <TouchableOpacity
      style={styles.addFriendBtn}
      activeOpacity={0.7}
      onPress={handleAcceptFriend}
    >
      <LinearGradient
        colors={['#22c55e', '#16a34a']}
        start={{ x: 0, y: 0 }}
        end={{ x: 1, y: 0 }}
        style={StyleSheet.absoluteFill}
      />
      <View style={styles.iconRow}>
        <Check size={13} color="#fff" />
        <Text style={styles.addFriendBtnText}> 同意添加</Text>
      </View>
    </TouchableOpacity>
  )}
  {isFriend && (
    <TouchableOpacity
      style={styles.chatBtn}
      activeOpacity={0.7}
      onPress={handleMessage}
    >
      <View style={styles.iconRow}>
        <MessageCircle size={13} color={COLORS.primary} />
        <Text style={styles.chatBtnText}> 聊天</Text>
      </View>
    </TouchableOpacity>
  )}

```

```

        </TouchableOpacity>
      })
    </View>
  </View>
</View>
{isFriendRequested && !isFriend && (
  <View style={styles.statusBanner}>
    <View style={styles.statusBannerInner}>
      <Check size={18} color="#2563eb" />
      <Text style={styles.statusText}>好友申请已发送</Text>
      <Text style={styles.statusSubtext}>等待对方确认</Text>
    </View>
  </View>
)}
<View style={styles.section}>
  <View style={styles.card}>
    <View style={styles.infoGrid}>
      <View style={styles.infoItem}>
        <View style={styles.infoIconBox}>
          <LinearGradient
            colors={['rgba(23,163,191,0.2)', 'rgba(6,182,212,0.2)']}
            start={{ x: 0, y: 0 }}
            end={{ x: 1, y: 1 }}
            style={{ ...StyleSheet.absoluteFill, borderRadius: 12 }}
          />
          <GraduationCap size={20} color={COLORS.primary} />
        </View>
        <View style={styles.infoContent}>
          <Text style={styles.infoLabel}>专业</Text>
          <Text style={styles.infoValue} numberOfLines={1}>
            {user?.majorName}
          </Text>
        </View>
      </View>
      <View style={styles.infoItem}>
        <View style={styles.infoIconBox}>
          <LinearGradient
            colors={['rgba(249,115,22,0.2)', 'rgba(251,146,60,0.2)']}
            start={{ x: 0, y: 0 }}
            end={{ x: 1, y: 1 }}
            style={{ ...StyleSheet.absoluteFill, borderRadius: 12 }}
          />
          <MapPin size={20} color="#f39c12" />
        </View>
        <View style={styles.infoContent}>
          <Text style={styles.infoLabel}>所在地</Text>
          <Text style={styles.infoValue}>
            {user?.location || user?.hometownCity || user?.hometownProvince || '未知'}
          </Text>
        </View>
      </View>
    </View>
  </View>

```

```

</View>
<View style={styles.infoItem}>
  <View style={styles.infoIconBox}>
    <LinearGradient
      colors={[ 'rgba(139,92,246,0.2)', 'rgba(168,85,247,0.2)' ]}
      start={{ x: 0, y: 0 }}
      end={{ x: 1, y: 1 }}
      style={{ ...StyleSheet.absoluteFill, borderRadius: 12 }}
    />
    <BookOpen size={20} color="#8b5cf6" />
  </View>
  <View style={styles.infoContent}>
    <Text style={styles.infoLabel}>年级</Text>
    <Text style={styles.infoValue}>{user?.grade}</Text>
  </View>
</View>
<View style={styles.infoItem}>
  <View style={styles.infoIconBox}>
    <LinearGradient
      colors={[ 'rgba(16,185,129,0.2)', 'rgba(52,211,153,0.2)' ]}
      start={{ x: 0, y: 0 }}
      end={{ x: 1, y: 1 }}
      style={{ ...StyleSheet.absoluteFill, borderRadius: 12 }}
    />
    <Users size={20} color="#27ae60" />
  </View>
  <View style={styles.infoContent}>
    <Text style={styles.infoLabel}>共同好友</Text>
    <Text style={styles.infoValue}>{user?.mutualFriendCount} 位</Text>
  </View>
</View>
</View>
</View>
</View>
<View style={styles.section}>
  <View style={styles.card}>
    <Text style={styles.cardTitle}>个人简介</Text>
    <Text style={styles.bioText}>{user?.bio || '暂无简介'}</Text>
  </View>
</View>
<View style={styles.section}>
  <View style={styles.card}>
    <Text style={styles.cardTitle}>兴趣爱好</Text>
    <View style={styles.interestTags}>
      {(user?.interests || []).map((interest, idx) => (
        <View key={`_${idx}-${interest}`} style={{ marginRight: 0 }}>
          <Badge
            label={interest}
            variant="default"
            style={styles.interestTag}
          />
        </View>
      ))}
    </View>
  </View>
</View>

```

```

    />
  </View>
  )}}
</View>
</View>
</View>
<View style=[[styles.section, styles.sectionLast]]>
  <View style={styles.card}>
    <View style={styles.mutualHeader}>
      <View>
        <Text style={styles.cardTitle}>共同好友</Text>
        <Text style={styles.mutualSubtext}>
          你们有 {user?.mutualFriendCount} 位共同好友
        </Text>
      </View>
      <TouchableOpacity activeOpacity={0.6}>
        <Text style={styles.viewAllText}>查看全部</Text>
      </TouchableOpacity>
    </View>
    <ScrollView
      horizontal
      showsHorizontalScrollIndicator={false}
      contentContainerStyle={styles.mutualScroll}
    >
      {(user?.mutualFriends || []).map((friend) => (
        <TouchableOpacity
          key={friend.id}
          style={styles.mutualFriendItem}
          activeOpacity={0.7}
          onPress={() =>
            navigation.navigate('UserDetail', {
              userId: friend.id
            } as never)
          }
        >
          <Avatar name={friend.nickname} size={48} />
          <Text style={styles.mutualFriendName} numberOfLines={1}>
            {friend.nickname}
          </Text>
        </TouchableOpacity>
      )}}
    </ScrollView>
  </View>
</View>
</ScrollView>
</View>
);
});
const styles = StyleSheet.create({
  container: {

```

```
    flex: 1,
    backgroundColor: COLORS.background,
  },
  loadingContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: COLORS.background,
  },
  headerArea: {
    position: 'relative',
  },
  coverArea: {
    height: 160,
    position: 'relative',
    overflow: 'hidden',
  },
  coverImage: {
    width: '100%',
    height: '100%',
    opacity: 0.8,
  },
  coverGradient: {
    ...StyleSheet.absoluteFillObject,
    background: 'linear-gradient(to bottom, rgba(23,163,191,0.3), rgba(6,182,212,0.2),
    rgba(139,92,246,0.3))',
  },
  coverOverlay: {
    ...StyleSheet.absoluteFillObject,
    background: 'linear-gradient(to top, rgba(248,249,250,1), transparent, transparent)',
  },
  backBtn: {
    position: 'absolute',
    top: SPACING.sm,
    left: SPACING.sm,
    width: 32,
    height: 32,
    borderRadius: BORDER_RADIUS.button,
    backgroundColor: 'rgba(0,0,0,0.3)',
    justifyContent: 'center',
    alignItems: 'center',
    zIndex: 10,
  },
  profileArea: {
    paddingHorizontal: SPACING.md,
    marginTop: -32,
    position: 'relative',
    zIndex: 5,
  },
  profileHeader: {
    flexDirection: 'row',
```

```
      alignItems: 'flex-end',
      gap: SPACING.md,
    },
    avatarContainer: {
      position: 'relative',
      flexShrink: 0,
    },
    avatar: {
      width: 96,
      height: 96,
      borderRadius: BORDER_RADIUS.card,
      borderWidth: 4,
      borderColor: COLORS.card,
      backgroundColor: COLORS.border,
    },
    friendBadge: {
      position: 'absolute',
      bottom: -3,
      right: -3,
      width: 28,
      height: 28,
      borderRadius: 14,
      backgroundColor: '#22c55e',
      justifyContent: 'center',
      alignItems: 'center',
      borderWidth: 2,
      borderColor: COLORS.card,
      shadowColor: '#000',
      shadowOffset: { width: 0, height: 1 },
      shadowOpacity: 0.1,
      shadowRadius: 2,
    },
    profileInfo: {
      flex: 1,
      paddingBottom: 4,
    },
    nickname: {
      fontSize: 20,
      fontWeight: '700',
      color: COLORS.text,
      marginBottom: 2,
    },
    schoolGrade: {
      fontSize: 13,
      color: COLORS.textMuted,
    },
    actionButtons: {
      flexDirection: 'column',
      gap: SPACING.xs,
      paddingBottom: 4,
    },
  },
},
```

```
    },
    addFriendBtn: {
      paddingHorizontal: SPACING.md,
      paddingVertical: SPACING.sm,
      borderRadius: BORDER_RADIUS.full,
      height: 36,
      justifyContent: 'center',
      alignItems: 'center',
      overflow: 'hidden',
    },
    addFriendBtnDisabled: {
      opacity: 0.6,
    },
    addFriendBtnText: {
      color: '#fff',
      fontSize: 12,
      fontWeight: '600',
      marginLeft: 4,
    },
    chatBtn: {
      backgroundColor: COLORS.card,
      paddingHorizontal: SPACING.md,
      paddingVertical: SPACING.sm,
      borderRadius: BORDER_RADIUS.full,
      height: 36,
      justifyContent: 'center',
      alignItems: 'center',
      borderWidth: 1,
      borderColor: COLORS.border,
    },
    chatBtnText: {
      color: COLORS.primary,
      fontSize: 12,
      fontWeight: '600',
      marginLeft: 4,
    },
    statusBanner: {
      paddingHorizontal: SPACING.md,
      paddingTop: SPACING.sm,
    },
    statusBannerInner: {
      flexDirection: 'row',
      alignItems: 'center',
      gap: SPACING.sm,
      backgroundColor: '#eff6ff',
      borderWidth: 1,
      borderColor: '#bfdbfe',
      borderRadius: 12,
      padding: SPACING.sm,
    },
  },
```

```
statusText: {
  fontSize: 13,
  fontWeight: '600',
  color: '#2563eb',
},
statusSubtext: {
  fontSize: 11,
  color: '#60a5fa',
},
section: {
  paddingHorizontal: SPACING.md,
  paddingTop: SPACING.sm,
},
sectionLast: {
  paddingBottom: SPACING.xl,
},
card: {
  backgroundColor: COLORS.card,
  borderRadius: BORDER_RADIUS.card,
  padding: SPACING.md,
  shadowColor: '#000',
  shadowOffset: { width: 0, height: 1 },
  shadowOpacity: 0.04,
  shadowRadius: 4,
  borderWidth: 0,
},
cardTitle: {
  fontSize: 14,
  fontWeight: '600',
  color: COLORS.text,
  marginBottom: SPACING.sm,
},
infoGrid: {
  flexDirection: 'row',
  flexWrap: 'wrap',
  gap: SPACING.md,
},
infoItem: {
  flex: 1,
  minWidth: '45%',
  flexDirection: 'row',
  alignItems: 'center',
  gap: SPACING.sm,
},
infoIconBox: {
  width: 40,
  height: 40,
  borderRadius: 12,
  justifyContent: 'center',
  alignItems: 'center',
```

```
      overflow: 'hidden',
    },
    infoContent: {
      flex: 1,
    },
    infoLabel: {
      fontSize: 11,
      color: COLORS.textMuted,
      marginBottom: 2,
    },
    infoValue: {
      fontSize: 13,
      fontWeight: '600',
      color: COLORS.text,
    },
    iconRow: {
      flexDirection: 'row',
      alignItems: 'center',
    },
    bioText: {
      fontSize: 13,
      color: COLORS.textMuted,
      lineHeight: 22,
    },
    interestTags: {
      flexDirection: 'row',
      flexWrap: 'wrap',
      gap: SPACING.sm,
    },
    interestTag: {
      background: 'linear-gradient(to right, rgba(23,163,191,0.1), rgba(6,182,212,0.1))',
      paddingHorizontal: 12,
      paddingVertical: 6,
      borderRadius: BORDER_RADIUS.full,
    },
    interestTagText: {
      fontSize: 11,
      fontWeight: '600',
      color: COLORS.primary,
    },
    mutualHeader: {
      flexDirection: 'row',
      justifyContent: 'space-between',
      alignItems: 'flex-start',
      marginBottom: SPACING.sm,
    },
    mutualSubtext: {
      fontSize: 11,
      color: COLORS.textMuted,
      marginTop: 2,
```

```
    },
    viewAllText: {
      fontSize: 12,
      color: COLORS.primary,
      fontWeight: '500',
    },
    mutualScroll: {
      gap: SPACING.md,
    },
    mutualFriendItem: {
      alignItems: 'center',
      minWidth: 60,
    },
    mutualFriendAvatar: {
      width: 48,
      height: 48,
      borderRadius: 24,
      borderWidth: 2,
      borderColor: COLORS.card,
      backgroundColor: COLORS.border,
    },
    mutualFriendName: {
      fontSize: 10,
      color: COLORS.textMuted,
      marginTop: SPACING.xs,
      textAlign: 'center',
      maxWidth: 60,
    },
  },
});
export default UserDetails;
// ===== src/data/mockData.ts =====
import { IUser, IPost, IComment } from '../types';
export const mockCurrentUser: IUser = {
  id: 'user_001',
  nickname: '张小海豚',
  avatar: '',
  school: '海豚大学',
  major: '计算机科学与技术',
  grade: '大三',
  interests: ['编程', '摄影', '篮球', '旅行'],
  bio: '热爱技术, 热爱生活, 期待遇见有趣的灵魂!',
  isReviewer: true,
};
export const mockUsers: IUser[] = [
  {
    id: 'user_002',
    nickname: '李小明',
    avatar: '',
    school: '海豚大学',
    major: '软件工程',
    grade: '大二',
  },
];
```

```
      interests: ['游戏', '音乐', '编程'],
      bio: '全栈开发爱好者, 正在寻找志同道合的朋友',
    },
    {
      id: 'user_003',
      nickname: '王小红',
      avatar: '',
      school: '海豚大学',
      major: '设计艺术',
      grade: '大三',
      interests: ['绘画', '摄影', '旅行'],
      bio: '用镜头记录生活, 用画笔描绘梦想',
    },
    {
      id: 'user_004',
      nickname: '赵小强',
      avatar: '',
      school: '海洋学院',
      major: '数据科学',
      grade: '大四',
      interests: ['AI', '机器学习', '跑步'],
      bio: '即将毕业, 寻找一起奋斗的小伙伴',
    },
    {
      id: 'user_005',
      nickname: '陈小美',
      avatar: '',
      school: '海豚大学',
      major: '英语',
      grade: '大一',
      interests: ['阅读', '电影', '美食'],
      bio: '新生报到, 期待认识学长学姐!',
    },
    {
      id: 'user_006',
      nickname: '刘小杰',
      avatar: '',
      school: '海洋学院',
      major: '金融',
      grade: '研究生',
      interests: ['投资', '健身', '旅行'],
      bio: '研究生在读, 喜欢探索新事物',
    },
  ];
export const mockPosts: IPost[] = [
  {
    id: 'post_001',
    author: mockUsers[0],
    category: '失物招领',
    title: '图书馆捡到校园卡一张',
  },
];
```

```
content: '今天下午在图书馆三楼捡到一张校园卡，卡主是计算机学院的同学，请失主联系我认领！',
images: [],
contact: '微信: lxming123',
likes: 12,
comments: 5,
favorites: 3,
createdAt: '2026-05-19 14:30',
isLiked: false,
isFavorited: false,
},
{
  id: 'post_002',
  author: mockUsers[1],
  category: '二手',
  title: '出闲置 iPad Air 4',
  content: '去年购入的 iPad Air 4, 64G WiFi 版, 深空灰色, 成色 95 新, 带原装充电器和保护壳。因为换了 Pro 所以出掉, 价格可小刀。',
  images: [],
  contact: 'QQ: *****',
  likes: 28,
  comments: 15,
  favorites: 20,
  createdAt: '2026-05-19 10:15',
  isLiked: true,
  isFavorited: false,
},
{
  id: 'post_003',
  author: mockUsers[2],
  category: '活动',
  title: '周末摄影社外拍活动报名',
  content: '本周六下午 2 点, 摄影社组织去海边外拍, 欢迎新老社员参加! 请自带相机, 不会用手机的也可以来学习。集合地点: 南门广场。',
  images: [],
  likes: 45,
  comments: 22,
  favorites: 18,
  createdAt: '2026-05-18 18:00',
  isLiked: false,
  isFavorited: true,
},
{
  id: 'post_004',
  author: mockUsers[3],
  category: '兼职',
  title: '家教兼职, 初中数学',
  content: '找一位数学好的同学做家教, 初二学生, 每周两次, 每次 2 小时, 地点在学校附近, 薪资面议。',
  images: [],
  contact: '电话: 138****0000',
  likes: 8,
  comments: 3,
  favorites: 5,
```

```
    createdAt: '2026-05-18 09:00',
  },
  {
    id: 'post_005',
    author: mockUsers[4],
    category: '求助',
    title: '求推荐好用的笔记软件',
    content: '最近想找一个适合做课堂笔记的软件，希望支持 Markdown、同步功能好、最好免费。用过 Notion 和 Obsidian，想听听大家的推荐！',
    likes: 35,
    comments: 28,
    favorites: 12,
    createdAt: '2026-05-17 20:30',
    isLiked: false,
    isFavorited: false,
  },
  {
    id: 'post_006',
    author: mockCurrentUser,
    category: '二手',
    title: '出考研资料',
    content: '出计算机考研全套资料，包括 408 四门课的教材和辅导书，九成新，打包出售。',
    images: [],
    contact: '微信: zht2024',
    likes: 15,
    comments: 8,
    favorites: 10,
    createdAt: '2026-05-17 15:00',
    reviewStatus: 'approved',
  },
];
export const mockComments: IComment[] = [
  {
    id: 'cmt_001',
    author: mockUsers[1],
    content: '是我的！太感谢了，我马上来取！',
    createdAt: '2026-05-19 14:45',
    likes: 2,
  },
  {
    id: 'cmt_002',
    author: mockUsers[3],
    content: '我也在图书馆丢过东西，还好被好心人捡到了',
    createdAt: '2026-05-19 15:00',
    likes: 1,
  },
  {
    id: 'cmt_003',
    author: mockUsers[4],
    content: '请问还在吗？我想看看实物',
    createdAt: '2026-05-19 11:00',
  },
];
```

```

        likes: 0,
    },
    {
        id: 'cmt_004',
        author: mockUsers[0],
        content: 'iPad 成色怎么样? 电池健康度多少? ',
        createdAt: '2026-05-19 11:30',
        likes: 1,
    },
];
export const mockMatchedUsers: IUser[] = [mockUsers[0], mockUsers[1]];
export const mockHotTopics = [
    '#期末复习互助',
    '#校园美食推荐',
    '#实习经验分享',
    '#考研资料交换',
    '#社团招新',
];
// ===== android/app/src/main/java/com/collegecli/MainActivity.kt =====
package com.collegecli
import com.facebook.react.ReactActivity
import com.facebook.react.ReactActivityDelegate
import com.facebook.react.defaults.DefaultNewArchitectureEntryPoint.fabricEnabled
import com.facebook.react.defaults.DefaultReactActivityDelegate
class MainActivity : ReactActivity() {
    override fun getMainComponentName(): String = "collegeCli"
    override fun createReactActivityDelegate(): ReactActivityDelegate =
        DefaultReactActivityDelegate(this, mainComponentName, fabricEnabled)
}
// ===== android/app/src/main/java/com/collegecli/MainApplication.kt =====
package com.collegecli
import android.app.Application
import com.facebook.react.PackageList
import com.facebook.react.ReactApplication
import com.facebook.react.ReactHost
import com.facebook.react.ReactNativeApplicationEntryPoint.loadReactNative
import com.facebook.react.defaults.DefaultReactHost.getDefaultReactHost
class MainApplication : Application(), ReactApplication {
    override val reactHost: ReactHost by lazy {
        getDefaultReactHost(
            context = applicationContext,
            packageList =
                PackageList(this).packages,
        )
    }
    override fun onCreate() {
        super.onCreate()
        try {
            System.loadLibrary("NitroSound")
        } catch (_: Throwable) {
        }
        loadReactNative(this)
    }
}

```

```

    }
}
// ===== android/app/src/main/java/com/collegecli/MyMessageReceiver.kt =====
package com.collegecli
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.util.Log
import com.alibaba.sdk.android.push.MessageReceiver
import com.alibaba.sdk.android.push.notification.CPushMessage
class MyMessageReceiver : MessageReceiver() {
    override fun onNotification(
        context: Context?,
        title: String?,
        summary: String?,
        extraMap: MutableMap<String, String>?
    ) {
        Log.d(TAG, "onNotification: title=$title, summary=$summary, extra=$extraMap")
    }
    override fun onMessage(context: Context?, message: CPushMessage?) {
        Log.d(TAG, "onMessage: ${message?.title} - ${message?.content}")
    }
    override fun onNotificationOpened(
        context: Context?,
        title: String?,
        summary: String?,
        extraMap: String?
    ) {
        Log.d(TAG, "onNotificationOpened: title=$title, summary=$summary, extra=$extraMap")
        if (context == null) return
        try {
            val extras = parseExtras(extraMap)
            val pushType = extras["type"] ?: "" // "chat" | "notification"
            val peerId = extras["peerId"] ?: ""
            val notificationId = extras["notificationId"] ?: ""
            val relatedId = extras["relatedId"] ?: ""
            val launchIntent = context.packageManager
                .getLaunchIntentForPackage(context.packageName)
                ?.apply {
                    addFlags(Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TOP)
                    putExtra("pushType", pushType)
                    putExtra("pushPeerId", peerId)
                    putExtra("pushNotificationId", notificationId)
                    putExtra("pushRelatedId", relatedId)
                    putExtra("pushTitle", title ?: "")
                    putExtra("pushSummary", summary ?: "")
                }
            if (launchIntent != null) {
                context.startActivity(launchIntent)
            }
        } catch (e: Exception) {
            Log.d(TAG, "通知点击跳转: type=$pushType, peerId=$peerId, notificationId=$notificationId")
        }
    }
}

```

```

    } catch (e: Exception) {
        Log.e(TAG, "处理通知点击异常: ${e.message}", e)
    }
}
override fun onNotificationClickedWithNoAction(
    context: Context?,
    title: String?,
    summary: String?,
    extraMap: String?
) {
    Log.d(TAG, "onNotificationClickedWithNoAction: title=$title, summary=$summary")
    onNotificationOpened(context, title, summary, extraMap)
}
override fun onNotificationRemoved(context: Context?, messageId: String?) {
    Log.d(TAG, "onNotificationRemoved: messageId=$messageId")
}
override fun onNotificationReceivedInApp(
    context: Context?,
    title: String?,
    summary: String?,
    extraMap: MutableMap<String, String>?,
    openType: Int,
    openActivity: String?,
    openUrl: String?
) {
    Log.d(TAG, "onNotificationReceivedInApp: title=$title")
}
private fun parseExtras(extraMap: String?): Map<String, String> {
    if (extraMap.isNullOrEmpty()) return emptyMap()
    return try {
        if (extraMap.startsWith("{") && extraMap.endsWith("}")) {
            parseJsonExtras(extraMap)
        } else {
            parseQueryStringExtras(extraMap)
        }
    } catch (e: Exception) {
        Log.w(TAG, "解析 extraMap 失败: $extraMap, error=${e.message}")
        emptyMap()
    }
}
private fun parseJsonExtras(json: String): Map<String, String> {
    val result = mutableMapOf<String, String>()
    val content = json.trim().removeSurrounding("{", "}")
    val pairs = splitJsonPairs(content)
    for (pair in pairs) {
        val colonIdx = pair.indexOf(":")
        if (colonIdx == -1) continue
        val key = pair.substring(0, colonIdx).trim().removeSurrounding("\"")
        val value = pair.substring(colonIdx + 1).trim().removeSurrounding("\"")
        result[key] = value
    }
}

```

```

    }
    return result
}
private fun splitJsonPairs(content: String): List<String> {
    val pairs = mutableListOf<String>()
    var depth = 0 // 跟踪大括号深度
    var inString = false
    var start = 0
    for (i in content.indices) {
        val c = content[i]
        if (c == '"' && (i == 0 || content[i - 1] != '\\')) inString = !inString
        if (!inString) {
            if (c == '{' || c == '[') depth++
            if (c == '}' || c == ']') depth--
        }
        if (!inString && depth == 0 && c == ',') {
            pairs.add(content.substring(start, i).trim())
            start = i + 1
        }
    }
    if (start < content.length) {
        pairs.add(content.substring(start).trim())
    }
    return pairs
}
private fun parseQueryStringExtras(query: String): Map<String, String> {
    val result = mutableMapOf<String, String>()
    query.split("&").forEach { pair ->
        val parts = pair.split("=", limit = 2)
        if (parts.size == 2) {
            result[parts[0]] = java.net.URLDecoder.decode(parts[1], "UTF-8")
        }
    }
    return result
}
}
companion object {
    private const val TAG = "CollegeCliPush"
    const val EXTRA_PUSH_TYPE = "pushType"
    const val EXTRA_PEER_ID = "pushPeerId"
    const val EXTRA_NOTIFICATION_ID = "pushNotificationId"
    const val EXTRA_RELATED_ID = "pushRelatedId"
    const val EXTRA_TITLE = "pushTitle"
    const val EXTRA_SUMMARY = "pushSummary"
}
}
// ===== ios/collegeCli/AppDelegate.swift =====
import UIKit
import React
import React_RCTAppDelegate
import ReactAppDependencyProvider
import CloudPushSDK

```

```

import UserNotifications // [NEW]
@main
class AppDelegate: UIResponder, UIApplicationDelegate {
    var window: UIWindow?
    var reactNativeDelegate: ReactNativeDelegate?
    var reactNativeFactory: RCTReactNativeFactory?
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]? = nil
    ) -> Bool {
        let delegate = ReactNativeDelegate()
        let factory = RCTReactNativeFactory(delegate: delegate)
        delegate.dependencyProvider = RCTAppDependencyProvider()
        reactNativeDelegate = delegate
        reactNativeFactory = factory
        window = UIWindow(frame: UIScreen.main.bounds)
        factory.startReactNative(
            withModuleName: "collegeCli",
            in: window,
            launchOptions: launchOptions
        )
        CloudPushSDK.sendNotificationAck(launchOptions)
        registerAPNs(application)
        if #available(iOS 10.0, *) {
            UNUserNotificationCenter.current().delegate = self
        }
        return true
    }
    func registerAPNs(_ application: UIApplication) {
        if #available(iOS 10.0, *) {
            let center = UNUserNotificationCenter.current()
            center.requestAuthorization(options: [.alert, .badge, .sound]) { granted, error in
                if granted {
                    DispatchQueue.main.async {
                        application.registerForRemoteNotifications()
                    }
                }
            }
        } else {
            let settings = UIUserNotificationSettings(types: [.alert, .badge, .sound], categories: nil)
            application.registerUserNotificationSettings(settings)
            application.registerForRemoteNotifications()
        }
    }
    func application(
        _ application: UIApplication,
        didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data
    ) {
        CloudPushSDK.registerDevice(deviceToken) { result in
            if result.success {

```

```

        NSLog("Register deviceToken success.")
    } else {
        NSLog("Register deviceToken failed, error: %@", result.error?.localizedDescription ?? "unknown")
    }
}
}
func application(
    _ application: UIApplication,
    didFinishRegisterForRemoteNotificationsWithError error: Error
) {
    NSLog("Register APNs failed, error: %@", error.localizedDescription)
}
func application(
    _ application: UIApplication,
    didReceiveRemoteNotification userInfo: [AnyHashable: Any],
    fetchCompletionHandler completionHandler: @escaping (UIBackgroundFetchResult) -> Void
) {
    CloudPushSDK.sendNotificationAck(userInfo)
    if let aps = userInfo["aps"] as? [String: Any] {
        if let badge = aps["badge"] as? Int {
            DispatchQueue.main.async {
                UIApplication.shared.applicationIconBadgeNumber = badge
            }
        }
    }
    completionHandler(.newData)
}
@available(iOS 10.0, *)
func userNotificationCenter(
    _ center: UNUserNotificationCenter,
    willPresent notification: UNNotification,
    withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void
) {
    if #available(iOS 14.0, *) {
        completionHandler([.banner, .badge, .sound])
    } else {
        completionHandler([.alert, .badge, .sound])
    }
}
@available(iOS 10.0, *)
func userNotificationCenter(
    _ center: UNUserNotificationCenter,
    didReceive response: UNNotificationResponse,
    withCompletionHandler completionHandler: @escaping () -> Void
) {
    let userInfo = response.notification.request.content.userInfo
    CloudPushSDK.sendNotificationAck(userInfo)
    if let aps = userInfo["aps"] as? [String: Any],
        let badge = aps["badge"] as? Int {
        DispatchQueue.main.async {

```

```
        UIApplication.shared.applicationIconBadgeNumber = badge
    }
}
completionHandler()
}
}
extension AppDelegate: UNUserNotificationCenterDelegate {}
class ReactNativeDelegate: RCTDefaultReactNativeFactoryDelegate {
    override func sourceURL(for bridge: RCTBridge) -> URL? {
        self.bundleURL()
    }
    override func bundleURL() -> URL? {
#if DEBUG
        RCTBundleURLProvider.sharedSettings().jsBundleURL(forBundleRoot: "index")
#else
        Bundle.main.url(forResource: "main", withExtension: "jsbundle")
#endif
    }
}
```