Mobile Applications

Lab 3

React Navigation: Stack, Tabs, parameters, headers, typed navigation (TS)

Mateusz Pawełkiewicz

Exercise goal

Configure React Navigation in an Expo (TypeScript) project: add Stack and Tabs, correctly type routes and params, pass data between screens, use useNavigation / useRoute, configure headers and themes.

Required software

- Project from previous labs (TS or JS), npx expo start working.
- React Navigation dependencies (install in the project directory):

```
# Core + deps
npm i @react-navigation/native
npx expo install react-native-screens react-native-safe-area-context

# Stack (native)
npm i @react-navigation/native-stack

# Tabs (bottom tabs)
npm i @react-navigation/bottom-tabs

# Icons
npx expo install @expo/vector-icons
```

If the computer and phone aren't on the same network, run the bundler in tunnel mode:

```
npx expo start --tunnel
```

Step 1: Folder structure and route types

Establish an order (example):

```
src/
  navigation/
    RootNavigator.tsx
    types.ts
  screens/
    HomeScreen.tsx
    ProfileScreen.tsx
    SettingsScreen.tsx
    LoginScreen.tsx
  components/
```

src/navigation/types.ts — central route types (TS):

```
export type RootStackParamList = {
  Login: undefined;
  AppTabs: undefined;
```

```
  Profile: { userId: string };
};


export type AppTabsParamList = {
  Home: undefined;
  Settings: { section?: string } | undefined;
};
```

Step 2: Stack + Tabs — configuration and theme

src/navigation/RootNavigator.tsx — navigators, theme and headers:

```
import React from "react";
import { NavigationContainer, DefaultTheme, DarkTheme, Theme } from "@react-navigation/native";
import { createNativeStackNavigator, NativeStackNavigationOptions } from "@react-navigation/native-stack";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import { Ionicons } from '@expo/vector-icons';
import type { RootStackParamList, AppTabsParamList } from "./types";

// Screens
import LoginScreen from "../screens/LoginScreen";
import HomeScreen from "../screens/HomeScreen";
import SettingsScreen from "../screens/SettingsScreen";
import ProfileScreen from "../screens/ProfileScreen";

const Stack = createNativeStackNavigator<RootStackParamList>();
const Tabs = createBottomTabNavigator<AppTabsParamList>();


const AppDarkTheme: Theme = {
   ...DarkTheme,
   colors: {
      ...DarkTheme.colors,
      background: "#0f1420",
      card: "#121a2a",
      text: "#e8eaed",
      border: "#1f2b40",
      primary: "#4c9aff",
   },
};

function AppTabsNavigator() {
   return (
      <Tabs.Navigator
         screenOptions={({ route }) => ({
            headerStyle: { backgroundColor: '#121a2a' },
            headerTitleStyle: { color: '#e8eaed' },
            tabBarStyle: { backgroundColor: '#121a2a', borderTopColor: '#1f2b40' },
```

```
            tabBarActiveTintColor: '#4c9aff',
            tabBarInactiveTintColor: '#a8b0bd',
            tabBarIcon: ({ focused, color, size }) => {
              const getName = () => {
                if (route.name === 'Home') {
                  return focused ? 'home' : 'home-outline';
                }
                if (route.name === 'Settings') {
                  return focused ? 'settings' : 'settings-outline';
                }
                return 'ellipse'; // fallback
              };
              return <Ionicons name={getName()} size={size} color={color} />;
            },
          })}
        >
          <Tabs.Screen name="Home" component={HomeScreen} options={{ title: 'Start' }} />
          <Tabs.Screen name="Settings" component={SettingsScreen} options={{ title: 'Ustawienia' }} />
        </Tabs.Navigator>
      );
}

export default function RootNavigator() {
      const isDark = true;
      const screenOptions: NativeStackNavigationOptions = {
        headerStyle: { backgroundColor: "#121a2a" },
        headerTitleStyle: { color: "#e8eaed" },
        headerTintColor: "#e8eaed",
      };

      return (
        <NavigationContainer theme={isDark ? AppDarkTheme : DefaultTheme}>
          <Stack.Navigator screenOptions={screenOptions} initialRouteName="Login">
            <Stack.Screen name="Login" component={LoginScreen} options={{ title: "Logowanie" }} />
            <Stack.Screen name="AppTabs" component={AppTabsNavigator} options={{ headerShown: false }} />
            <Stack.Screen
              name="Profile"
              component={ProfileScreen}
              options={{ title: "Profil użytkownika" }}
            />
          </Stack.Navigator>
        </NavigationContainer>
      );
}
```

Step 3: Safe Area integration and App.tsx

Wrap everything in SafeAreaProvider in App.tsx, and render the navigator:

```
// App.tsx
import React from "react";
import { StatusBar } from "expo-status-bar";
import { SafeAreaProvider } from "react-native-safe-area-context";
import RootNavigator from "./src/navigation/RootNavigator";

export default function App() {
  return (
    <SafeAreaProvider>
      <StatusBar style="light" />
      <RootNavigator />
    </SafeAreaProvider>
  );
}
```

NavigationContainer (in RootNavigator) cooperates with react-native-safe-area-context, so layouts and headers will respect safe areas.

Step 4: Screens and navigation — typed props, useNavigation, useRoute

src/screens/LoginScreen.tsx — a simple login screen that navigates to Tabs:

```
import React, { useState } from "react";
import { View, Text, StyleSheet, TextInput, Pressable, Switch } from "react-native";
import { NativeStackScreenProps } from "@react-navigation/native-stack";
import { SafeAreaView } from "react-native-safe-area-context";
import { Ionicons } from "@expo/vector-icons";
import type { RootStackParamList } from "../navigation/types";

type Props = NativeStackScreenProps<RootStackParamList, "Login">;

const COLORS = {
  bg: "#0f1420",
  card: "#121a2a",
  border: "#1f2b40",
  text: "#e8eaed",
  subtext: "#a8b0bd",
  primary: "#4c9aff",
};

export default function LoginScreen({ navigation }: Props) {
  const [email, setEmail] = useState("");
  const [pwd, setPwd] = useState("");
  const [showPwd, setShowPwd] = useState(false);
  const [remember, setRemember] = useState(true);

  const canLogin = email.trim().length > 3 && pwd.length >= 3;

  const onLogin = () => {
    if (!canLogin) return;
    navigation.replace("AppTabs");
  };
```

```
  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: COLORS.bg }}>
      <View style={styles.container}>
        <View style={styles.header}>
          <View style={styles.logoWrap}>
            <Ionicons name="sparkles-outline" size={28} color={COLORS.primary} />
          </View>
          <Text style={styles.title}>Zaloguj się</Text>
          <Text style={styles.subtitle}>Witaj ponownie</Text>
        </View>

        <View style={styles.card}>
          <Text style={styles.label}>E-mail</Text>
          <View style={styles.inputWrap}>
            <Ionicons name="mail-outline" size={18} color={COLORS.subtext} />
            <TextInput
              style={styles.input}
              value={email}
              onChangeText={setEmail}
              placeholder="you@example.com"
              placeholderTextColor="#7c8799"
              keyboardType="email-address"
              autoCapitalize="none"
              autoCorrect={false}
              returnKeyType="next"
            />
          </View>

          <Text style={[styles.label, { marginTop: 14 }]}>Hasło</Text>
          <View style={styles.inputWrap}>
            <Ionicons name="lock-closed-outline" size={18} color={COLORS.subtext} />
            <TextInput
              style={styles.input}
              value={pwd}
              onChangeText={setPwd}
              placeholder="••••••••"
              placeholderTextColor="#7c8799"
              secureTextEntry={!showPwd}
              autoCapitalize="none"
            />
            <Pressable onPress={() => setShowPwd((v) => !v)} style={styles.iconBtn}>
              <Ionicons name={showPwd ? "eye-off-outline" : "eye-outline"} size={18}
color={COLORS.subtext} />
            </Pressable>
          </View>

          <View style={styles.row}>
            <View style={styles.rowLeft}>
              <Switch
                value={remember}
                onValueChange={setRemember}
                trackColor={{ false: "#3a455a", true: "#2e6db3" }}
                thumbColor={remember ? COLORS.primary : "#dfe3ea"}
              />
              <Text style={styles.remember}>Zapamiętaj mnie</Text>
            </View>
            <Pressable>
```

```
            <Text style={styles.forgot}>Nie pamiętasz hasła?</Text>
          </Pressable>
        </View>

        <Pressable
          onPress={onLogin}
          android_ripple={{ color: "rgba(0,0,0,0.12)" }}
          style={[styles.btn, !canLogin && { opacity: 0.5 }]}
        >
          <Text style={styles.btnText}>Wejdź do aplikacji</Text>
        </Pressable>

        <Pressable
          onPress={() => navigation.replace("AppTabs")}
          android_ripple={{ color: "rgba(255,255,255,0.08)" }}
          style={styles.btnGhost}
        >
          <Ionicons name="person-outline" size={16} color={COLORS.text} />
          <Text style={styles.btnGhostText}>Kontynuuj jako gość</Text>
        </Pressable>

        <View style={styles.orRow}>
          <View style={styles.line} />
          <Text style={styles.orText}>albo</Text>
          <View style={styles.line} />
        </View>

        <View style={styles.socialRow}>
          <Pressable android_ripple={{ color: "rgba(255,255,255,0.08)" }} style={styles.socialBtn}>
            <Ionicons name="logo-google" size={18} color={COLORS.text} />
            <Text style={styles.socialText}>Google</Text>
          </Pressable>
          <Pressable android_ripple={{ color: "rgba(255,255,255,0.08)" }} style={styles.socialBtn}>
            <Ionicons name="logo-github" size={18} color={COLORS.text} />
            <Text style={styles.socialText}>GitHub</Text>
          </Pressable>
        </View>
      </View>
    </View>
  </SafeAreaView>
 );
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: 24, gap: 16 },
  header: { alignItems: "center", gap: 6, marginTop: 12, marginBottom: 6 },
  logoWrap: {
    width: 56,
    height: 56,
    borderRadius: 16,
    backgroundColor: "rgba(76,154,255,0.12)",
    borderWidth: 1,
    borderColor: "rgba(76,154,255,0.25)",
    alignItems: "center",
    justifyContent: "center",
  },
  title: { color: COLORS.text, fontSize: 22, fontWeight: "800" },
```

```
  subtitle: { color: COLORS.subtext, fontSize: 14 },
  card: {
    backgroundColor: COLORS.card,
    borderRadius: 16,
    borderWidth: 1,
    borderColor: COLORS.border,
    padding: 16,
    gap: 10,
  },
  label: { color: COLORS.subtext, fontSize: 12, textTransform: "uppercase", letterSpacing: 0.4 },
  inputWrap: {
    flexDirection: "row",
    alignItems: "center",
    gap: 10,
    backgroundColor: "rgba(255,255,255,0.03)",
    borderWidth: 1,
    borderColor: "rgba(255,255,255,0.08)",
    borderRadius: 12,
    paddingHorizontal: 12,
    height: 48,
  },
  input: { flex: 1, color: COLORS.text, fontSize: 16 },
  iconBtn: { padding: 4, borderRadius: 8 },
  row: { flexDirection: "row", alignItems: "center", justifyContent: "space-between", marginTop: 2 },
  rowLeft: { flexDirection: "row", alignItems: "center", gap: 8 },
  remember: { color: COLORS.subtext, fontSize: 13 },
  forgot: { color: COLORS.primary, fontSize: 13, fontWeight: "700" },
  btn: { backgroundColor: COLORS.primary, borderRadius: 12, alignItems: "center", justifyContent: "center",
height: 48, marginTop: 8 },
  btnText: { color: "#0b0e13", fontWeight: "800", fontSize: 15 },
  btnGhost: {
    marginTop: 10,
    height: 46,
    borderRadius: 12,
    borderWidth: 1,
    borderColor: "rgba(255,255,255,0.18)",
    alignItems: "center",
    justifyContent: "center",
    flexDirection: "row",
    gap: 8,
  },
  btnGhostText: { color: COLORS.text, fontWeight: "700" },
  orRow: { flexDirection: "row", alignItems: "center", gap: 8, marginTop: 12 },
  line: { flex: 1, height: 1, backgroundColor: "rgba(255,255,255,0.08)" },
  orText: { color: COLORS.subtext, fontSize: 12 },
  socialRow: { flexDirection: "row", gap: 10 },
  socialBtn: {
    flex: 1,
    height: 46,
    borderRadius: 12,
    borderWidth: 1,
    borderColor: "rgba(255,255,255,0.18)",
    alignItems: "center",
    justifyContent: "center",
    flexDirection: "row",
    gap: 8,
  },
```

```
  socialText: { color: COLORS.text, fontWeight: "700" },
});
```

src/screens/HomeScreen.tsx — navigate to a details screen and pass a sample param:

```
import React from "react";
import { View, Text, StyleSheet, Pressable, FlatList, Platform } from "react-native";
import { useNavigation } from "@react-navigation/native";
import type { NativeStackNavigationProp } from "@react-navigation/native-stack";
import { Ionicons } from "@expo/vector-icons";
import type { RootStackParamList } from "../navigation/types";

type Nav = NativeStackNavigationProp<RootStackParamList>;

type Tile = {
  key: string;
  title: string;
  icon: keyof typeof Ionicons.glyphMap;
  color: string;
  onPress: () => void;
};

export default function HomeScreen() {
  const navigation = useNavigation<Nav>();

  const tiles: Tile[] = [
    { key: "profile",  title: "Profil",     icon: "person",      color: "#4C9AFF", onPress: () =>
navigation.navigate("Profile", { userId: "user_123" }) },
    { key: "settings", title: "Ustawienia",  icon: "settings",    color: "#7C4DFF", onPress: () =>
navigation.navigate("AppTabs", { screen: "Settings" } as never) },
    { key: "news",     title: "Aktualności", icon: "newspaper",   color: "#00C853", onPress: () => {} },
    { key: "help",     title: "Pomoc",      icon: "help-circle",  color: "#FF7043", onPress: () => {} },
  ];

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Home</Text>

      <FlatList
        data={tiles}
        keyExtractor={(item) => item.key}
        numColumns={2}
        columnWrapperStyle={styles.row}
        ItemSeparatorComponent={() => <View style={{ height: GAP }} />}
        renderItem={({ item }) => (
          <Pressable
            onPress={item.onPress}
            android_ripple={{ color: "rgba(0,0,0,0.12)" }}
            style={({ pressed }) => [
              styles.tile,
              { backgroundColor: item.color },
              pressed && Platform.OS === "ios" && { opacity: 0.8 },
            ]}
          >
            <Ionicons name={item.icon} size={28} color="#FFFFFF" />
            <Text style={styles.tileText}>{item.title}</Text>
```

```
            </Pressable>
          )}
          contentContainerStyle={{ paddingBottom: GAP }}
        />
      </View>
    );
}

const GAP = 16;

const styles = StyleSheet.create({
  container: { flex: 1, backgroundColor: "#0f1420", padding: 24 },
  title: { color: "#e8eaed", fontSize: 22, fontWeight: "700", marginBottom: 16 },
  row: { justifyContent: "space-between", marginBottom: GAP },
  tile: {
    borderRadius: 16,
    padding: 16,
    height: 110,
    flex: 1,
    marginRight: GAP / 2,
    marginLeft: GAP / 2,
    justifyContent: "center",
    alignItems: "center",
    borderWidth: 1,
    borderColor: "rgba(255,255,255,0.12)",
  },
  tileText: { marginTop: 8, color: "#FFFFFF", fontWeight: "700" },
});
```

src/screens/ProfileScreen.tsx — receive the parameter with typed useRoute:

```
import React, { useMemo, useState } from "react";
import { View, Text, StyleSheet, ScrollView, Pressable, Switch, Platform } from "react-native";
import { RouteProp, useRoute } from "@react-navigation/native";
import { SafeAreaView } from "react-native-safe-area-context";
import { Ionicons } from "@expo/vector-icons";
import type { RootStackParamList } from "../navigation/types";

type Route = RouteProp<RootStackParamList, "Profile">;

const COLORS = {
  bg: "#0f1420",
  card: "#121a2a",
  border: "#1f2b40",
  text: "#e8eaed",
  subtext: "#a8b0bd",
  primary: "#4c9aff",
  success: "#00C853",
  warning: "#FFB300",
};
```

```
export default function ProfileScreen() {
  const route = useRoute<Route>();
  const { userId } = route.params;

  const displayName = useMemo(() => {
    const id = `${userId}`.replace(/[^a-zA-Z0-9]+/g, " ").trim();
    if (!id) return "Użytkownik";
    const parts = id.split(" ");
    if (parts.length >= 2) return `${capitalize(parts[0])} ${capitalize(parts[1])}`;
    return capitalize(parts[0]);
  }, [userId]);

  const initials = useMemo(() => {
    const letters = displayName.split(" ").map(s => s[0]?.toUpperCase()).filter(Boolean);
    return (letters[0] || "U") + (letters[1] || "");
  }, [displayName]);

  const [isPrivate, setIsPrivate] = useState(false);
  const [twoFA, setTwoFA] = useState(Platform.OS === "ios");

  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: COLORS.bg }}>
      <ScrollView contentContainerStyle={styles.content} showsVerticalScrollIndicator={false}>
        <View style={styles.headerCard}>
          <View style={styles.avatar}>
            <Text style={styles.avatarText}>{initials}</Text>
          </View>
          <Text style={styles.name}>{displayName}</Text>
          <Text style={styles.handle}>@{userId}</Text>
          <View style={styles.actions}>
            <Pressable style={[styles.button, styles.buttonPrimary]} android_ripple={{ color:
"rgba(0,0,0,0.12)" }}>
              <Ionicons name="create-outline" size={16} color="#0b0e13" />
              <Text style={[styles.buttonText, { color: "#0b0e13" }]}>Edytuj profil</Text>
            </Pressable>
            <Pressable style={[styles.button, styles.buttonGhost]} android_ripple={{ color:
"rgba(255,255,255,0.08)" }}>
              <Ionicons name="chatbubbles-outline" size={16} color={COLORS.text} />
              <Text style={styles.buttonText}>Wiadomość</Text>
            </Pressable>
          </View>
          <View style={styles.stats}>
            <Stat value="128" label="Obserwujący" />
            <Stat value="87" label="Obserwowani" />
            <Stat value="24" label="Posty" />
          </View>
        </View>

        <View style={styles.card}>
          <Text style={styles.cardTitle}>Informacje</Text>
          <Item icon="mail-outline" label="E-mail" value="user@example.com" />
          <Divider />
          <Item icon="call-outline" label="Telefon" value="+48 600 000 000" />
          <Divider />
          <Item icon="globe-outline" label="Strona" value="example.com" />
          <Divider />
          <Item icon="calendar-outline" label="Dołączył" value="12.03.2024" />
```

```
          </View>

          <View style={styles.card}>
            <Text style={styles.cardTitle}>Bezpieczeństwo</Text>
            <ToggleItem icon="shield-outline" label="Uwierzytelnianie 2FA" value={twoFA}
onValueChange={setTwoFA} />
            <Divider />
            <ToggleItem icon="lock-closed-outline" label="Konto prywatne" value={isPrivate}
onValueChange={setIsPrivate} />
          </View>

          <View style={styles.card}>
            <Text style={styles.cardTitle}>Akcje</Text>
            <Pressable style={styles.rowPress} android_ripple={{ color: "rgba(255,255,255,0.06)" }}>
              <View style={styles.rowLeft}>
                <IconBadge name="share-social-outline" />
                <Text style={styles.rowLabel}>Udostępnij profil</Text>
              </View>
              <Ionicons name="chevron-forward" size={18} color={COLORS.subtext} />
            </Pressable>
            <Divider />
            <Pressable style={styles.rowPress} android_ripple={{ color: "rgba(255,255,255,0.06)" }}>
              <View style={styles.rowLeft}>
                <IconBadge name="download-outline" />
                <Text style={styles.rowLabel}>Pobierz dane</Text>
              </View>
              <Ionicons name="chevron-forward" size={18} color={COLORS.subtext} />
            </Pressable>
          </View>
        </ScrollView>
      </SafeAreaView>
    );
}

function Stat({ value, label }: { value: string; label: string }) {
    return (
      <View style={styles.stat}>
        <Text style={styles.statValue}>{value}</Text>
        <Text style={styles.statLabel}>{label}</Text>
      </View>
    );
}

function IconBadge({ name }: { name: keyof typeof Ionicons.glyphMap }) {
    return (
      <View style={styles.iconBadge}>
        <Ionicons name={name} size={18} color={COLORS.primary} />
      </View>
    );
}

function Item({ icon, label, value }: { icon: keyof typeof Ionicons.glyphMap; label: string; value: string }) {
    return (
      <View style={styles.row}>
        <View style={styles.rowLeft}>
          <IconBadge name={icon} />
          <Text style={styles.rowLabel}>{label}</Text>
```

```
        </View>
        <Text style={styles.rowValue}>{value}</Text>
      </View>
    );
}

function ToggleItem({
                    icon,
                    label,
                    value,
                    onValueChange,
                  }: {
    icon: keyof typeof Ionicons.glyphMap;
    label: string;
    value: boolean;
    onValueChange: (v: boolean) => void;
}) {
    return (
      <View style={styles.row}>
        <View style={styles.rowLeft}>
          <IconBadge name={icon} />
          <Text style={styles.rowLabel}>{label}</Text>
        </View>
        <Switch trackColor={{ false: "#3a455a", true: "#2e6db3" }} thumbColor={value ? COLORS.primary :
"#dfe3ea"} value={value} onValueChange={onValueChange} />
      </View>
    );
}

function Divider() {
    return <View style={styles.divider} />;
}

function capitalize(s: string) {
    return s.charAt(0).toUpperCase() + s.slice(1).toLowerCase();
}

const styles = StyleSheet.create({
    content: { padding: 16, gap: 16 },
    headerCard: {
      backgroundColor: COLORS.card,
      borderRadius: 16,
      borderWidth: 1,
      borderColor: COLORS.border,
      padding: 16,
      alignItems: "center",
      gap: 12,
    },
    avatar: {
      width: 84,
      height: 84,
      borderRadius: 42,
      backgroundColor: "rgba(76,154,255,0.15)",
      borderWidth: 2,
      borderColor: "rgba(76,154,255,0.35)",
      alignItems: "center",
      justifyContent: "center",
```

```
  },
  avatarText: { color: COLORS.primary, fontSize: 28, fontWeight: "800" },
  name: { color: COLORS.text, fontSize: 20, fontWeight: "800" },
  handle: { color: COLORS.subtext, fontSize: 14 },
  actions: { flexDirection: "row", gap: 10, marginTop: 4 },
  button: { minHeight: 40, paddingHorizontal: 14, borderRadius: 12, flexDirection: "row", alignItems: "center",
gap: 8, borderWidth: 1 },
  buttonPrimary: { backgroundColor: COLORS.primary, borderColor: "transparent" },
  buttonGhost: { backgroundColor: "transparent", borderColor: "rgba(255,255,255,0.18)" },
  buttonText: { color: COLORS.text, fontWeight: "700" },
  stats: { flexDirection: "row", gap: 18, marginTop: 6 },
  stat: { alignItems: "center", paddingHorizontal: 10 },
  statValue: { color: COLORS.text, fontSize: 16, fontWeight: "800" },
  statLabel: { color: COLORS.subtext, fontSize: 12 },
  card: {
    backgroundColor: COLORS.card,
    borderRadius: 16,
    borderWidth: 1,
    borderColor: COLORS.border,
    paddingVertical: 8,
    overflow: "hidden",
  },
  cardTitle: { color: COLORS.subtext, fontSize: 12, textTransform: "uppercase", letterSpacing: 0.4,
paddingHorizontal: 12, marginBottom: 6, marginTop: 6 },
  row: { minHeight: 54, paddingHorizontal: 12, flexDirection: "row", alignItems: "center", justifyContent:
"space-between" },
  rowPress: { minHeight: 54, paddingHorizontal: 12, flexDirection: "row", alignItems: "center", justifyContent:
"space-between" },
  rowLeft: { flexDirection: "row", alignItems: "center", gap: 10 },
  iconBadge: {
    width: 28,
    height: 28,
    borderRadius: 8,
    backgroundColor: "rgba(76,154,255,0.12)",
    alignItems: "center",
    justifyContent: "center",
    borderWidth: 1,
    borderColor: "rgba(76,154,255,0.25)",
  },
  rowLabel: { color: COLORS.text, fontSize: 16, fontWeight: "600" },
  rowValue: { color: COLORS.subtext, fontSize: 14 },
  divider: { height: 1, backgroundColor: "rgba(255,255,255,0.06)", marginHorizontal: 12 },
});
```

src/screens/SettingsScreen.tsx — optional parameters (e.g., opening a specific section):

```
import React, { useMemo, useState } from "react";
import {
  View,
  Text,
  StyleSheet,
  ScrollView,
  Pressable,
```

```
  Switch,
  Modal,
  TouchableOpacity,
  Platform,
} from "react-native";
import { RouteProp, useNavigation, useRoute } from "@react-navigation/native";
import type { AppTabsParamList, RootStackParamList } from "../navigation/types";
import type { NativeStackNavigationProp } from "@react-navigation/native-stack";
import { SafeAreaView } from "react-native-safe-area-context";
import { Ionicons } from "@expo/vector-icons";

type Route = RouteProp<AppTabsParamList, "Settings">;
type RootNav = NativeStackNavigationProp<RootStackParamList>;

const COLORS = {
  bg: "#0f1420",
  card: "#121a2a",
  border: "#1f2b40",
  text: "#e8eaed",
  subtext: "#a8b0bd",
  primary: "#4c9aff",
  accent: "#4c9aff",
  success: "#00C853",
  danger: "#ff6b6b",
  overlay: "rgba(0,0,0,0.6)",
};

export default function SettingsScreen() {
  const route = useRoute<Route>();
  const navigation = useNavigation<RootNav>();
  const section = route.params?.section ?? "general";

  const [pushEnabled, setPushEnabled] = useState(true);
  const [emailEnabled, setEmailEnabled] = useState(false);
  const [analyticsEnabled, setAnalyticsEnabled] = useState(true);
  const [faceIdEnabled, setFaceIdEnabled] = useState(Platform.OS === "ios");

  const [language, setLanguage] = useState<"pl" | "en">("pl");
  const [theme, setTheme] = useState<"system" | "light" | "dark">("dark");

  const [showLangModal, setShowLangModal] = useState(false);
  const [showThemeModal, setShowThemeModal] = useState(false);

  const sectionTitle = useMemo(() => {
    switch (section) {
      case "general":
        return "Ustawienia ogólne";
      case "notifications":
        return "Powiadomienia";
      case "privacy":
```

```
        return "Prywatność";
      default:
        return "Ustawienia";
    }
  }, [section]);

  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: COLORS.bg }}>
      <ScrollView
        contentContainerStyle={styles.scrollContent}
        showsVerticalScrollIndicator={false}
      >
        <Text style={styles.screenTitle}>Ustawienia</Text>
        <Text style={styles.sectionHint}>{sectionTitle}</Text>

        <GroupCard title="Konto">
          <Row
            icon="person-circle"
            label="Profil użytkownika"
            value="user_123"
            onPress={() => navigation.navigate("Profile", { userId: "user_123" })}
          />
          <Divider />
          <Row icon="key-outline" label="Zmień hasło" onPress={() => {}} />
          <Divider />
          <Row
            icon="log-out-outline"
            label="Wyloguj"
            valueStyle={{ color: COLORS.danger }}
            rightIconColor={COLORS.danger}
            rightIcon="chevron-forward"
            onPress={() => {}}
          />
        </GroupCard>

        <GroupCard title="Ogólne">
          <Row
            icon="language-outline"
            label="Język aplikacji"
            value={language === "pl" ? "Polski" : "English"}
            onPress={() => setShowLangModal(true)}
          />
          <Divider />
          <Row
            icon="color-palette-outline"
            label="Motyw"
            value={
              theme === "system" ? "Systemowy" : theme === "light" ? "Jasny" : "Ciemny"
            }
            onPress={() => setShowThemeModal(true)}
```

```
        />
      </GroupCard>

      <GroupCard title="Powiadomienia">
        <ToggleRow
          icon="notifications-outline"
          label="Powiadomienia push"
          value={pushEnabled}
          onValueChange={setPushEnabled}
        />
        <Divider />
        <ToggleRow
          icon="mail-unread-outline"
          label="Powiadomienia e-mail"
          value={emailEnabled}
          onValueChange={setEmailEnabled}
        />
      </GroupCard>

      <GroupCard title="Prywatność i bezpieczeństwo">
        <ToggleRow
          icon="finger-print-outline"
          label={Platform.OS === "ios" ? "Face ID" : "Biometria"}
          value={faceIdEnabled}
          onValueChange={setFaceIdEnabled}
        />
        <Divider />
        <ToggleRow
          icon="analytics-outline"
          label="Analityka użytkowania"
          caption="Anonimowe dane pomagają ulepszać aplikację"
          value={analyticsEnabled}
          onValueChange={setAnalyticsEnabled}
        />
        <Divider />
        <Row icon="document-lock-outline" label="Polityka prywatności" onPress={() => {}} />
      </GroupCard>

      <GroupCard title="O aplikacji">
        <Row icon="information-circle-outline" label="Wersja" value="1.0.0" />
        <Divider />
        <Row icon="chatbubbles-outline" label="Opinie i wsparcie" onPress={() => {}} />
      </GroupCard>
    </ScrollView>

    <PickerModal
      visible={showLangModal}
      title="Wybierz język"
      options={[
        { key: "pl", label: "Polski" },
```

```
            { key: "en", label: "English" },
          ]}
          selectedKey={language}
          onSelect={(k) => setLanguage(k as "pl" | "en")}
          onClose={() => setShowLangModal(false)}
        />

        <PickerModal
          visible={showThemeModal}
          title="Wybierz motyw"
          options={[
            { key: "system", label: "Systemowy" },
            { key: "light", label: "Jasny" },
            { key: "dark", label: "Ciemny" },
          ]}
          selectedKey={theme}
          onSelect={(k) => setTheme(k as "system" | "light" | "dark")}
          onClose={() => setShowThemeModal(false)}
        />
      </SafeAreaView>
    );
}



function GroupCard({ title, children }: { title: string; children: React.ReactNode }) {
  return (
    <View style={styles.card}>
      <Text style={styles.cardTitle}>{title}</Text>
      <View style={styles.cardBody}>{children}</View>
    </View>
  );
}

function Divider() {
  return <View style={styles.divider} />;
}

function Row({
            icon,
            label,
            value,
            onPress,
            rightIcon = "chevron-forward",
            rightIconColor = COLORS.subtext,
            valueStyle,
          }: {
  icon: keyof typeof Ionicons.glyphMap;
  label: string;
  value?: string;
```

```
  onPress?: () => void;
  rightIcon?: keyof typeof Ionicons.glyphMap;
  rightIconColor?: string;
  valueStyle?: any;
}) {
  return (
    <Pressable
      onPress={onPress}
      android_ripple={{ color: "rgba(255,255,255,0.08)" }}
      style={({ pressed }) => [styles.row, pressed && Platform.OS === "ios" && { opacity: 0.7 }]}
    >
      <View style={styles.rowLeft}>
        <View style={styles.rowIconWrap}>
          <Ionicons name={icon} size={20} color={COLORS.accent} />
        </View>
        <Text style={styles.rowLabel}>{label}</Text>
      </View>

      <View style={styles.rowRight}>
        {!!value && <Text style={[styles.rowValue, valueStyle]}>{value}</Text>}
        {onPress && <Ionicons name={rightIcon} size={18} color={rightIconColor} />}
      </View>
    </Pressable>
  );
}

function ToggleRow({
                icon,
                label,
                caption,
                value,
                onValueChange,
              }: {
  icon: keyof typeof Ionicons.glyphMap;
  label: string;
  caption?: string;
  value: boolean;
  onValueChange: (v: boolean) => void;
}) {
  return (
    <View style={styles.row}>
      <View style={styles.rowLeft}>
        <View style={styles.rowIconWrap}>
          <Ionicons name={icon} size={20} color={COLORS.accent} />
        </View>
        <View>
          <Text style={styles.rowLabel}>{label}</Text>
          {!!caption && <Text style={styles.rowCaption}>{caption}</Text>}
        </View>
      </View>
```

```
        <Switch
          trackColor={{ false: "#3a455a", true: "#2e6db3" }}
          thumbColor={value ? COLORS.primary : "#dfe3ea"}
          value={value}
          onValueChange={onValueChange}
        />
      </View>
    );
}

function PickerModal({
                    visible,
                    title,
                    options,
                    selectedKey,
                    onSelect,
                    onClose,
                  }: {
    visible: boolean;
    title: string;
    options: { key: string; label: string }[];
    selectedKey: string;
    onSelect: (key: string) => void;
    onClose: () => void;
}) {
    return (
      <Modal transparent visible={visible} animationType="fade">
        <View style={styles.modalOverlay}>
          <View style={styles.modalCard}>
            <Text style={styles.modalTitle}>{title}</Text>

            {options.map((opt) => {
              const selected = opt.key === selectedKey;
              return (
                <TouchableOpacity
                  key={opt.key}
                  onPress={() => {
                    onSelect(opt.key);
                    onClose();
                  }}
                  style={styles.modalOption}
                  activeOpacity={0.7}
                >
                  <Text style={[styles.modalOptionText, selected && { color: COLORS.primary }]}>
                    {opt.label}
                  </Text>
                  {selected && <Ionicons name="checkmark" size={18} color={COLORS.primary} />}
                </TouchableOpacity>
              );
            })}
```

```
            <TouchableOpacity onPress={onClose} style={styles.modalClose} activeOpacity={0.7}>
              <Text style={styles.modalCloseText}>Zamknij</Text>
            </TouchableOpacity>
          </View>
        </View>
      </Modal>
    );
}


const styles = StyleSheet.create({
  scrollContent: {
    padding: 16,
    paddingBottom: 32,
    gap: 16,
  },
  screenTitle: {
    color: COLORS.text,
    fontSize: 22,
    fontWeight: "800",
  },
  sectionHint: {
    color: COLORS.subtext,
    marginTop: -4,
    marginBottom: 4,
  },

  card: {
    backgroundColor: COLORS.card,
    borderRadius: 16,
    borderWidth: 1,
    borderColor: COLORS.border,
    padding: 12,
  },
  cardTitle: {
    color: COLORS.subtext,
    fontSize: 12,
    letterSpacing: 0.4,
    textTransform: "uppercase",
    marginBottom: 8,
  },
  cardBody: {
    borderRadius: 12,
    overflow: "hidden",
  },

  row: {
    minHeight: 54,
    paddingHorizontal: 12,
```

```
    backgroundColor: "transparent",
    flexDirection: "row",
    alignItems: "center",
    justifyContent: "space-between",
  },
  rowLeft: { flexDirection: "row", alignItems: "center", gap: 12 },
  rowIconWrap: {
    width: 28,
    height: 28,
    borderRadius: 8,
    backgroundColor: "rgba(76,154,255,0.12)",
    alignItems: "center",
    justifyContent: "center",
    borderWidth: 1,
    borderColor: "rgba(76,154,255,0.25)",
  },
  rowLabel: { color: COLORS.text, fontSize: 16, fontWeight: "600" },
  rowCaption: { color: COLORS.subtext, fontSize: 12, marginTop: 2 },
  rowRight: { flexDirection: "row", alignItems: "center", gap: 8 },
  rowValue: { color: COLORS.subtext, fontSize: 14 },

  divider: {
    height: 1,
    backgroundColor: "rgba(255,255,255,0.06)",
    marginHorizontal: 12,
  },

  modalOverlay: {
    flex: 1,
    backgroundColor: COLORS.overlay,
    alignItems: "center",
    justifyContent: "center",
    padding: 24,
  },
  modalCard: {
    width: "100%",
    backgroundColor: COLORS.card,
    borderRadius: 16,
    borderWidth: 1,
    borderColor: COLORS.border,
    padding: 16,
    gap: 8,
  },
  modalTitle: {
    color: COLORS.text,
    fontSize: 16,
    fontWeight: "800",
    marginBottom: 8,
  },
  modalOption: {
```

```
    minHeight: 44,
    paddingVertical: 8,
    paddingHorizontal: 4,
    borderRadius: 10,
    flexDirection: "row",
    alignItems: "center",
    justifyContent: "space-between",
  },
  modalOptionText: {
    color: COLORS.text,
    fontSize: 15,
  },
  modalClose: {
    marginTop: 8,
    alignSelf: "flex-end",
    paddingHorizontal: 12,
    paddingVertical: 8,
    borderRadius: 10,
    backgroundColor: "rgba(76,154,255,0.12)",
    borderWidth: 1,
    borderColor: "rgba(76,154,255,0.25)",
  },
  modalCloseText: { color: COLORS.primary, fontWeight: "700" },
});
```

Step 5: Header configuration, hiding the header, and custom titles

- Hide the header for the entire tabs container: `options={{ headerShown: false }}` on the `AppTabs` screen.
- Custom title: `options={{ title: "My title" }}`.
- Dynamic title from a route param (at Stack screen level):

```
<Stack.Screen
 name="ProfileDetails"
 component={ProfileDetailsScreen}
 options={({ route }) => ({ title: `Profile: ${(route.params as any).userId}` })}
/>
```

- Custom header color/appearance via `screenOptions` in both `Stack.Navigator` and `Tabs.Navigator` (shown above).

Step 6 (optional): Reset/replace, going back, and simple route guards

- Go back: `navigation.goBack()`.
- Replace (e.g., after login): `navigation.replace("AppTabs")`.
- Reset the stack (fresh start, e.g., logout):

```
navigation.reset({
  index: 0,
  routes: [{ name: "Login" }],
});
```

- Simple guard: keep isAuthenticated in a global place (e.g., context). In RootNavigator show Login when !isAuthenticated, and AppTabs when isAuthenticated.

Task to complete

The exact task will be given by the instructor during the class. Prepare the project so it meets the following conditions:

- Working Stack and Tabs with a sensible structure (e.g., Login → AppTabs (Home, Settings) → ProfileDetails).
- Correctly typed routes and params (RootStackParamList, AppTabsParamList) and usage of useNavigation, useRoute.
- Example of passing a parameter (e.g., userId) and header configuration (static and/or dynamic).
- Consistent theme and headers (colors, headerTintColor), working with react-native-safe-area-context.
- Nice to have: a simple “guard” (simulated login state) and using replace after login.