Mobile Applications

Lab 2

Styling, Flexbox, Safe Area, StatusBar, Components & Props

Mateusz Pawełkiewicz

Exercise goal

Master the layout and visual design of screens in React Native: Flexbox (justifyContent, alignItems, flex, gap, flexWrap), safe screen insets via react-native-safe-area-context, StatusBar control, creating components, and passing typed props with TypeScript.

Step 1: Safe Area + StatusBar (with react-native-safe-area-context)

Install the library:
npm i react-native-safe-area-context

Wrap the app in SafeAreaProvider at the nearest "root" (e.g., in App.tsx).

Render screen content inside SafeAreaView with specific edges so you have full control.

```
// App.tsx
import React, { JSX } from "react";
import { StatusBar } from "expo-status-bar";
import { View, Text, StyleSheet } from "react-native";
import { SafeAreaProvider, SafeAreaView } from "react-native-safe-area-context";

const BG = "#0f1420";
const FG = "#e8eaed";

export default function App(): JSX.Element {
  return (
    <SafeAreaProvider>
      <SafeAreaView style={styles.safe} edges={["top", "right", "bottom", "left"]}>
        <StatusBar style="light" />
        <View style={styles.container}>
          <Text style={styles.title}>Styling • Flexbox • Safe Area</Text>
          <Text style={styles.subtitle}>StatusBar: light icons on a dark background</Text>
        </View>
      </SafeAreaView>
    </SafeAreaProvider>
  );
}

const styles = StyleSheet.create({
  safe: { flex: 1, backgroundColor: BG },
  container: { flex: 1, backgroundColor: BG, padding: 20, justifyContent: "center" },
  title: { color: FG, fontSize: 22, fontWeight: "700", marginBottom: 6 },
  subtitle: { color: FG, opacity: 0.85 },
});
```

Why: SafeAreaProvider exposes the safe-edge dimensions context, and SafeAreaView automatically adds insets for the notch and rounded corners. Set the background color on both SafeAreaView and the container to avoid any "bleed-through".

Step 2: Flexbox — rows, columns, gap, wrap

Create components/FlexShowcase.tsx and add practical layouts.

```
// components/FlexShowcase.tsx
import React, { JSX } from "react";
import { View, Text, StyleSheet } from "react-native";

function Box({ label }: { label: string }): JSX.Element {
  return (
    <View style={styles.box}>
      <Text style={styles.boxText}>{label}</Text>
    </View>
  );
}

export default function FlexShowcase(): JSX.Element {
  return (
    <View style={styles.wrap}>
      {/* Row */}
      <View style={styles.row}>
        <Box label="A" /><Box label="B" /><Box label="C" />
      </View>

      {/* Column */}
      <View style={styles.col}>
        <Box label="1" /><Box label="2" />
      </View>

      {/* Wrapping into subsequent lines */}
      <View style={styles.wrapRow}>
        {Array.from({ length: 8 }, (_, i) => <Box key={i} label={`${i + 1}`} />)}
      </View>

      {/* Centering on both axes */}
      <View style={styles.centerBoth}>
        <Text style={{ color: "#cde4ff" }}>Centered on both axes</Text>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  wrap: { gap: 16, padding: 16 },
  row: { flexDirection: "row", gap: 8, alignItems: "center" },
  col: { flexDirection: "column", gap: 8, alignItems: "flex-start" },
  wrapRow: { flexDirection: "row", flexWrap: "wrap", gap: 8 },
  centerBoth: {
    height: 120, backgroundColor: "#12263a", justifyContent: "center",
    alignItems: "center", borderRadius: 12,
  },
  box: {
    backgroundColor: "#1e2a44", borderColor: "#2e3a58",
    borderWidth: 1, borderRadius: 10, paddingVertical: 12, paddingHorizontal: 16,
  },
  boxText: { color: "#e8eaed", fontWeight: "700" },
```

```
});
```

Enable the demo in App.tsx (inside the container, under the title):

```
import FlexShowcase from "./components/FlexShowcase";

// ...
<View style={styles.container}>
  <Text style={styles.title}>Styling • Flexbox • Safe Area</Text>
  <Text style={styles.subtitle}>StatusBar: light icons on a dark background</Text>
  <FlexShowcase />
</View>
```

Step 3: Components and props (TSX)

Extract a reusable tile with typed props (title, subtitle?, style?, titleStyle?) to keep things tidy and make appearance easy to tweak.

```
// components/Tile.tsx
import React, { JSX } from "react";
import { View, Text, StyleSheet, ViewStyle, TextStyle } from "react-native";

type TileProps = {
  title: string;
  subtitle?: string;
  style?: ViewStyle;
  titleStyle?: TextStyle;
  onPress?: () => void;
};

export default function Tile({ title, subtitle, style, titleStyle }: TileProps): JSX.Element {
  return (
    <View style={[styles.card, style]}>
      <Text style={[styles.title, titleStyle]}>{title}</Text>
      {!!subtitle && <Text style={styles.subtitle}>{subtitle}</Text>}
    </View>
  );
}

const styles = StyleSheet.create({
  card: {
    backgroundColor: "#162033",
    borderRadius: 14,
    padding: 16,
    borderWidth: 1,
    borderColor: "#253250",
  },
  title: { color: "#e8eaed", fontSize: 16, fontWeight: "700" },
  subtitle: { color: "#c9d4e6", marginTop: 6 },
});
```

Example usage in a tile layout:

```
// /App.tsx
import React, { JSX } from "react";
import { View, StyleSheet } from "react-native";
import Tile from "./components/Tile";

export default function Home(): JSX.Element {
  return (
    <View style={styles.container}>
      <View style={styles.row}>
        <Tile title="Profile" subtitle="Your data" style={{ flex: 1 }} />
        <Tile title="Settings" subtitle="Theme & more" style={{ flex: 1 }} />
      </View>
      <View style={styles.row}>
        <Tile title="Notifications" style={{ flex: 1 }} />
        <Tile title="Help" style={{ flex: 1 }} />
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { padding: 16, gap: 12 },
  row: { flexDirection: "row", gap: 12 },
});
```

Assignment

The task will be given by the instructor during the class and will require the use of:

- react-native-safe-area-context with SafeAreaProvider and SafeAreaView (full edges).
- StatusBar with a style appropriate for the background.
- Flexbox: rows/columns, gap, flexWrap, centering on both axes.
- At least one reusable component with well-typed props (TypeScript).