

**PRAKTIKUM**  
**PENGEMBANGAN APLIKASI BERGERAK**

**MODUL 4**

**Design in React Native**

Disusun oleh:  
**ZULFA AZKA FARISADILAH**  
**1204220109**



**Universitas**  
**Telkom**

**PROGRAM STUDI SARJANA SISTEM INFORMASI**  
**DIREKTORAT KAMPUS SURABAYA**  
**UNIVERSITAS TELKOM**  
**SURABAYA**  
**2025**

## DAFTAR ISI

1	Membuat & Menjalankan Project React Native via Expo .....	3
2	Persiapan Struktur Project .....	4
3	Inisialisasi index.js pada Direktori /screens .....	4
4	Inisialisasi index.js pada Direktori /components .....	5
5	Inisialisasi App.js.....	6
6	Komponen Header .....	8
7	Komponen Button .....	9
8	Komponen Separator.....	10
9	Screen LotsOfStyles .....	11
10	Screen FixedDimensionsBasics .....	13
11	Screen FlexDimensionsBasics .....	14
12	Screen PercentageDimensionsBasics .....	15
13	Screen FlexBasic.....	17
14	Screen FlexDirectionBasics .....	19
15	Screen JustifyContentBasics .....	22
16	Screen AlignItemsLayout .....	25
17	Screen WidthHeightBasics .....	28
18	Screen PositionLayout .....	30
19	Screen DisplayAnImageWithStyle .....	32
20	[Update] Install Gluestack UI (Alternatif) .....	34
21	Tugas .....	35
22	Link Repository GitHub: .....	42

# 1 Membuat & Menjalankan Project React Native via Expo

## Input:

- Buat sebuah project **React Native** via **Expo** bernama **design-in-rn** dengan menjalankan command:
  - **npx create-expo-app design-in-rn --template blank**
- Masuk ke folder project **design-in-rn** yang sudah terbuat dengan menjalankan command:
  - **cd design-in-rn**
- Jalankan project react native anda dengan command:
  - **npx expo start** atau
  - **npx expo start --tunnel**
- Akan muncul sebuah **QRCode** pada command prompt anda

Buka aplikasi **Expo Go** di handphone, kemudian scan **QRCode** tersebut.

## Output:

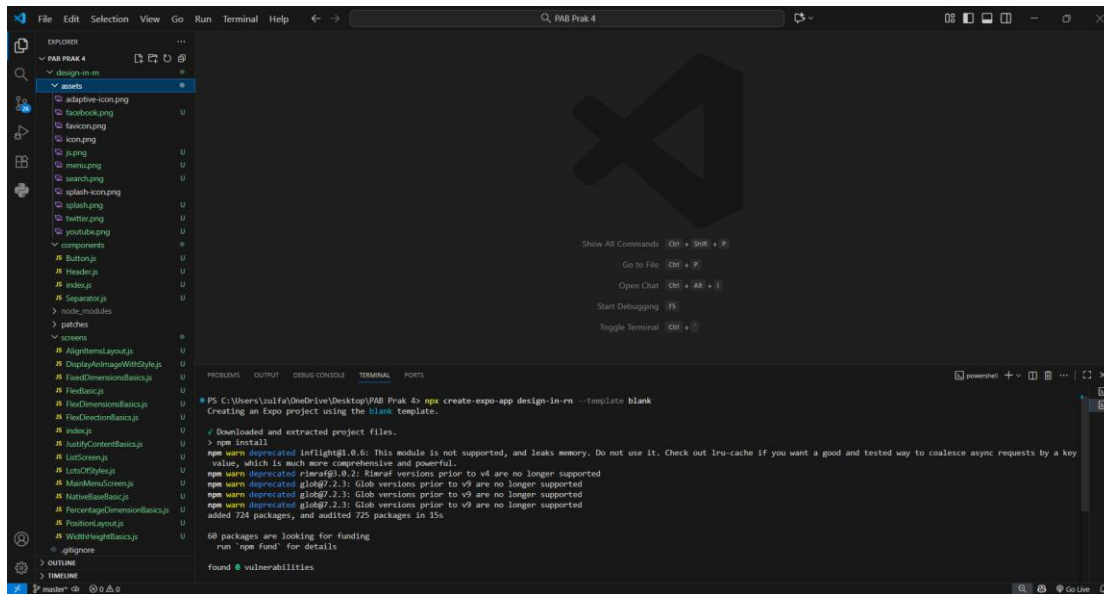
23:32

1.41 MB/s

Open up App.js to start working on your app!



## 2 Persiapan Struktur Project



## 3 Inisialisasi index.js pada Direktori /screens

Input:

```
import LotsOfStyles from "./LotsOfStyles";
import FixedDimensionsBasics from "./FixedDimensionsBasics";
import FlexDimensionsBasics from "./FlexDimensionsBasics";
import PercentageDimensionsBasics from "./PercentageDimensionBasic";
import FlexBasic from "./FlexBasic";
import FlexDirectionBasics from "./FlexDirectionBasics";
import JustifyContentBasics from "./JustifyContentBasics";
import AlignItemsLayout from "./AlignItemsLayout";
import WidthHeightBasics from "./WidthHeightBasics";
import PositionLayout from "./PositionLayout";
import DisplayAnImageWithStyle from "./DisplayAnImageWithStyle";
import NativeBaseBasic from "./NativeBaseBasic";
import GlueStackUIBasic from "./GlueStackUIBasic";

export {
  LotsOfStyles,
  FixedDimensionsBasics,
  FlexDimensionsBasics,
  PercentageDimensionsBasics,
  FlexBasic,
  FlexDirectionBasics,
  JustifyContentBasics,
  AlignItemsLayout,
  WidthHeightBasics,
  PositionLayout,
  DisplayAnImageWithStyle,
  NativeBaseBasic,
  GlueStackUIBasic,
};
```


### Penjelasan:

setiap baris import digunakan untuk mengambil komponen dari file lain di folder yang sama, misalnya `LotsOfStyles`, `FlexBasic`, atau `AlignItemsLayout`. Masing-masing file tersebut kemungkinan berisi contoh tata letak (layout) atau gaya tampilan (style) yang berbeda, seperti penggunaan flexbox, dimensi tetap atau persentase, dan penataan posisi elemen dalam React Native.

Bagian terakhir yang diawali dengan `export { ... }` berfungsi untuk menggabungkan semua komponen yang telah diimpor tadi menjadi satu objek ekspor bersama. Dengan cara ini, file lain bisa mengimpor komponen apa pun dari `/screens/index.js` tanpa harus menulis jalur file satu per satu. Misalnya, daripada menulis `import LotsOfStyles from './screens/LotsOfStyles'`, kita cukup menulis `import { LotsOfStyles } from './screens'`.

## 4 Inisialisasi `index.js` pada Direktori `/components`

### Input:



```
import Button from "./Button";
import Header from "./Header";
import Separator from "./Separator";

export { Button, Header, Separator };
```

### Penjelasan:

Pada bagian awal kode, terdapat tiga pernyataan import yang digunakan untuk memanggil komponen bernama `Button`, `Header`, dan `Separator` dari file masing-masing di dalam folder yang sama. Setiap komponen tersebut kemungkinan memiliki fungsi antarmuka pengguna tertentu, seperti tombol interaktif (`Button`), bagian kepala tampilan (`Header`), dan elemen pemisah antarbagian (`Separator`).

Selanjutnya, bagian `export { Button, Header, Separator };` digunakan untuk menyediakan ketiga komponen tersebut agar dapat diakses secara bersamaan dari file lain tanpa perlu mengimpor satu per satu dari jalur file aslinya.

## 5 Inisialisasi App.js

### Input:

```
import { useRef, useState } from "react";
import {
  View,
  DrawerLayoutAndroid,
  StatusBar,
  ScrollView,
  Text,
  StyleSheet,
} from "react-native";
import { Header, Button, Separator } from "../components";
import {
  LotsOfStyles,
  FixedDimensionsBasics,
  FlexDimensionsBasics,
  PercentageDimensionsBasics,
  FlexBasic,
  FlexDirectionBasics,
  JustifyContentBasics,
  AlignItemsLayout,
  WidthHeightBasics,
  PositionLayout,
  DisplayAnImageWithStyle,
  NativeBaseBasic,
  GlueStackUIBasic,
} from "../screens";

// Functional Component
const App = () => {
  // State Declaration
  const [page, setPage] = useState("Lots Of Styles");
  // Ref Declaration
  const drawer = useRef(null);
  // Array of Object Declaration
  const pageArr = [
    { name: "Lots Of Styles", comp: <LotsOfStyles /> },
    { name: "Fixed Dimensions Basics", comp: <FixedDimensionsBasics /> },
    { name: "Flex Dimensions Basics", comp: <FlexDimensionsBasics /> },
    {
      name: "Percentage Dimensions Basics",
      comp: <PercentageDimensionsBasics />,
    },
    { name: "Flex Basic", comp: <FlexBasic /> },
    { name: "Flex Direction Basics", comp: <FlexDirectionBasics /> },
    { name: "Justify Content Basics", comp: <JustifyContentBasics /> },
    { name: "Align Items Layout", comp: <AlignItemsLayout /> },
    { name: "Width Height Basics", comp: <WidthHeightBasics /> },
    { name: "Position Layout", comp: <PositionLayout /> },
    { name: "Display An Image With Style", comp: <DisplayAnImageWithStyle /> },
  ], { name: "Native Base Basic", comp: <NativeBaseBasic /> },
  { name: "Gluestack UI Basic", comp: <GlueStackUIBasic /> },
];

// Ref Declaration
const content = useRef(pageArr[0]);

// Find in Array of Object
content.current = pageArr.find((item) => item.name == page);

// Arrow Function inside Functional Component
const changePage = (drawer, pageName) => {
  // Close Drawer
  drawer.current.closeDrawer();
  // Change state value
  setPage(pageName);
};

// Arrow Function inside Functional Component
const navigationView = () => (
  <ScrollView style={styles.drawer}>
    <Text style={styles.textMenus}>MENUS:</Text>
    { /* Looping with map() */
      (pageArr.map((item, index) => {
        return (
          <View key={index}>
            <Button
              text={item.name}
              onPress={() => changePage(drawer, item.name)}
            />
            <Separator height={10} />
          </View>
        );
      })
    )
    <Button text="Close" onPress={() => drawer.current.closeDrawer()} />
    <Separator height={30} />
  </ScrollView>
);

return (
  <DrawerLayoutAndroid
    ref={drawer}
    drawerWidth={300}
    drawerPosition="left"
    renderNavigationView={navigationView}
  >
    <StatusBar style="light" backgroundColor="#AA0002" />
    <View style={{ flex: 1 }}>
      <Header drawer={drawer} />
      {content.current.comp}
    </View>
  </DrawerLayoutAndroid>
);
};

const styles = StyleSheet.create({
  drawer: { padding: 10, backgroundColor: "#222222", flex: 1 },
  textMenus: {
    color: "white",
    fontSize: 12,
    marginBottom: 10,
    fontWeight: "900",
  },
});

export default App;
```

**Penjelasan:**

Pada bagian awal, dilakukan import berbagai komponen React Native seperti View, Text, ScrollView, dan StatusBar, serta beberapa komponen khusus (Header, Button, dan Separator) dari direktori /components. Selain itu, kode juga mengimpor berbagai tampilan (screen) dari direktori /screens, seperti LotsOfStyles, FlexBasic, dan PositionLayout, yang masing-masing merepresentasikan halaman berbeda dalam aplikasi.

Di dalam fungsi utama App, digunakan useState untuk mendeklarasikan variabel status page, yang menentukan halaman aktif yang sedang ditampilkan, serta useRef untuk membuat referensi terhadap elemen DrawerLayoutAndroid agar dapat dikontrol secara langsung (misalnya membuka atau menutup drawer). Selanjutnya, terdapat array objek pageArr yang berisi daftar nama halaman (name) dan komponennya (comp), yang kemudian digunakan untuk menampilkan konten dinamis sesuai nilai status page. Proses pemilihan halaman dilakukan melalui fungsi changePage, yang menutup drawer dan memperbarui status halaman berdasarkan pilihan pengguna.

Fungsi navigationView membentuk struktur menu navigasi di dalam drawer menggunakan ScrollView dan melakukan iterasi dengan metode map() untuk menampilkan tombol-tombol yang mewakili setiap halaman. Tiap tombol (Button) akan memanggil fungsi changePage ketika ditekan. Pada bagian return, komponen DrawerLayoutAndroid menjadi wadah utama yang menampilkan Header di bagian atas dan konten halaman yang dipilih (content.current.comp) di bagian bawah. Selain itu, StatusBar digunakan untuk mengatur tampilan bar atas perangkat agar selaras dengan tema aplikasi.

Bagian terakhir dari kode berisi objek StyleSheet, yang mendefinisikan gaya tampilan untuk elemen-elemen dalam drawer, seperti warna latar belakang, warna teks, ukuran huruf, serta margin. Secara keseluruhan, kode ini membentuk struktur aplikasi dengan arsitektur modular dan dinamis, di mana setiap halaman dapat diakses melalui menu geser, serta mendukung pengelolaan navigasi yang efisien dan terorganisir.

## 6 Komponenten Header

Input:

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

// Functional Component with props
const Header = (props) => {
  return (
    <View style={styles.header}>
      <TouchableOpacity onPress={() => props.drawer.current.openDrawer()}>
        <Image
          source={require("../assets/menu.png")}
          style={{ width: 18, height: 18 }}
        />
      </TouchableOpacity>
      <View>
        <View style={styles.iconsView}>
          <Image
            source={require("../assets/facebook.png")}
            style={styles.icons}
          />
          <Image
            source={require("../assets/youtube.png")}
            style={styles.icons}
          />
          <Image
            source={require("../assets/twitter.png")}
            style={styles.icons}
          />
          <Image
            source={require("../assets/search.png")}
            style={styles.icons}
          />
        </View>
      </View>
    </View>
  );
};

// Styles
const styles = StyleSheet.create({
  header: {
    backgroundColor: "#AA0002",
    flexDirection: "row",
    justifyContent: "space-between",
    padding: 15,
  },
  iconsView: {
    flexDirection: "row",
    alignItems: "center",
    justifyContent: "center",
  },
  icons: {
    width: 36,
    height: 16,
    resizeMode: "contain",
  },
});

export default Header;
```



## Penjelasan:

Di dalam struktur return, elemen utama View diberi gaya melalui `styles.header` untuk mengatur tata letak dan warna latar belakang. Di sisi kiri terdapat `TouchableOpacity`, yaitu komponen yang memberikan efek sentuh (klik), dan ketika ditekan akan memanggil fungsi `props.drawer.current.openDrawer()` untuk membuka panel menu drawer. Di dalamnya terdapat ikon bergambar menu yang diambil dari file lokal `../assets/menu.png` melalui properti `require()`.

Selanjutnya, bagian kanan header menampilkan kumpulan ikon media sosial seperti Facebook, YouTube, Twitter, dan ikon pencarian (search), yang disusun secara horizontal di dalam View dengan gaya `styles.iconsView`. Masing-masing ikon diatur menggunakan elemen `Image` dengan ukuran dan mode tampilan yang telah ditentukan di bagian `StyleSheet`.

Bagian akhir kode mendefinisikan objek `styles` untuk mengatur gaya visual pada setiap elemen. Misalnya, header memiliki properti `backgroundColor` berwarna merah tua, tata letak `flexDirection: "row"` untuk menampilkan elemen sejajar secara horizontal, serta `justifyContent: "space-between"` agar ikon menu dan ikon media sosial berada di sisi berlawanan.

## 7 Komponen Button

### Input:

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

// Functional Component with props
const Button = (props) => {
  return (
    <TouchableOpacity style={styles.container} onPress={props.onPress}>
      <Text style={styles.text}>{props.text}</Text>
    </TouchableOpacity>
  );
};

// Styles
const styles = StyleSheet.create({
  container: {
    backgroundColor: "#ddddd",
    padding: 15,
    alignItems: "center",
  },
  text: {
    fontSize: 12,
    textTransform: "uppercase",
    fontWeight: "bold",
  },
});

export default Button;
```

### Penjelasan:

Struktur tampilan komponen menggunakan elemen `TouchableOpacity`, yaitu komponen `React Native` yang memungkinkan pengguna melakukan interaksi sentuhan (tap) dengan efek perubahan opasitas saat disentuh. Di dalamnya terdapat elemen `Text` yang menampilkan label tombol sesuai dengan nilai yang dikirim melalui `props.text`. Dengan demikian, komponen ini dapat digunakan secara fleksibel untuk berbagai kebutuhan dengan hanya mengubah teks dan fungsi yang dipanggil ketika tombol ditekan.

Bagian akhir kode berisi objek `StyleSheet`, yang mengatur tampilan visual tombol. Properti `container` memberikan latar belakang abu-abu muda (`#dddddd`), jarak dalam (`padding: 15`), dan penataan teks agar berada di tengah (`alignItems: "center"`). Sementara itu, gaya pada `text` mengatur ukuran huruf (`fontSize: 12`), menjadikan teks huruf besar semua melalui `textTransform: "uppercase"`, dan menebalkan tulisan dengan `fontWeight: "bold"`.

## 8 Komponen Separator

### Input:



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

// Functional Component with props
const Separator = (props) => {
  return <View style={{ height: props.height }}></View>;
};

export default Separator;
```

### Penjelasan:

Struktur komponen ini sangat ringkas karena hanya menggunakan satu elemen `View`, yang diatur secara dinamis melalui properti `style`. Nilai tinggi (`height`) pada gaya ditetapkan berdasarkan nilai yang dikirim dari komponen induk, sehingga komponen `Separator` dapat digunakan secara fleksibel di berbagai bagian aplikasi dengan ukuran spasi yang berbeda sesuai kebutuhan.

## 9 Screen LotsOfStyles

Input:

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

const LotsOfStyles = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.red}>just red</Text>
      <Text style={styles.bigBlue}>just bigBlue</Text>
      <Text style={[styles.bigBlue, styles.red]}>bigBlue, then red</Text>
      <Text style={[styles.red, styles.bigBlue]}>red, then bigBlue</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: 15,
  },
  bigBlue: {
    color: "blue",
    fontWeight: "bold",
    fontSize: 30,
  },
  red: {
    color: "red",
  },
});

export default LotsOfStyles;
```

Output:



just red

**just bigBlue**

**bigBlue, then red**

**red, then bigBlue**

**Penjelasan:**

Di dalam fungsi `LotsOfStyles`, terdapat sebuah elemen utama `View` yang menggunakan gaya `styles.container` untuk memberikan jarak dalam (`padding`) pada seluruh isi tampilan. Komponen ini menampilkan empat teks dengan variasi gaya: teks pertama hanya berwarna merah (`styles.red`), teks kedua berwarna biru dan berukuran besar (`styles.bigBlue`), sedangkan dua teks terakhir menunjukkan contoh penggabungan beberapa gaya melalui penggunaan array pada properti `style`, misalnya `[styles.bigBlue, styles.red]` dan `[styles.red, styles.bigBlue]`. Urutan penulisan gaya dalam array tersebut memengaruhi hasil akhir gaya yang ditulis terakhir akan menimpa gaya sebelumnya pada atribut yang sama.

Bagian akhir kode mendefinisikan objek `styles` menggunakan `StyleSheet.create()`. Di dalamnya, terdapat tiga gaya utama: `container` yang menambahkan jarak dalam (`padding: 15`), `bigBlue` yang memberikan warna biru, ukuran huruf besar (`fontSize: 30`), serta teks tebal (`fontWeight: "bold"`), dan `red` yang hanya mengatur warna teks menjadi merah.

## 10 Screen FixedDimensionsBasics

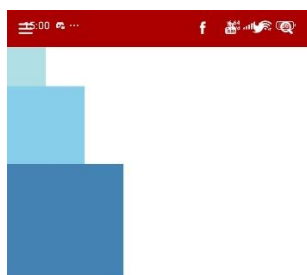
Input:

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

const FixedDimensionsBasics = () => {
  return (
    <View>
      <View
        style={{
          width: 50,
          height: 50,
          backgroundColor: "powderblue",
        }}
      />
      <View
        style={{
          width: 100,
          height: 100,
          backgroundColor: "skyblue",
        }}
      />
      <View
        style={{
          width: 150,
          height: 150,
          backgroundColor: "steelblue",
        }}
      />
    </View>
  );
};

export default FixedDimensionsBasics;
```

Output:



## Penjelasan:

Di dalam fungsi `FixedDimensionsBasics`, terdapat satu elemen utama `View` sebagai container yang membungkus tiga elemen `View` lain. Setiap `View` di dalamnya diberi lebar (`width`) dan tinggi (`height`) yang berbeda-beda, yaitu masing-masing `50x50`, `100x100`, dan `150x150` piksel. Selain itu, setiap `View` memiliki warna latar (`backgroundColor`) yang berbeda yaitu `powderblue`, `skyblue`, dan `steelblue` untuk memudahkan perbedaan visual antar elemen.

Kode ini tidak menggunakan `StyleSheet` terpisah seperti contoh lainnya, melainkan langsung menuliskan gaya di dalam properti `style` pada tiap elemen `View`. Cara ini dikenal sebagai `inline styling`, yang sering digunakan untuk contoh sederhana atau pengujian cepat.

## 11 Screen `FlexDimensionsBasics`

### Input:

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

const FlexDimensionsBasics = () => {
  return (
    <View style={{ flex: 1 }}>
      <View style={{ flex: 1, backgroundColor: "powderblue" }} />
      <View style={{ flex: 2, backgroundColor: "skyblue" }} />
      <View style={{ flex: 3, backgroundColor: "steelblue" }} />
    </View>
  );
};

export default FlexDimensionsBasics;
```

### Output:



## Penjelasan:

Di dalam komponen tersebut, terdapat satu elemen utama `<View>` yang berperan sebagai container dengan properti `style={{ flex: 1 }}`. Nilai `flex: 1` menunjukkan bahwa container ini akan mengambil seluruh ruang yang tersedia pada layar. Di dalam container utama tersebut, terdapat tiga elemen `<View>` lain yang masing-masing mewakili bagian atau blok tampilan dengan warna berbeda, yaitu `powderblue`, `skyblue`, dan `steelblue`. Masing-masing elemen memiliki nilai `flex` yang berbeda, yaitu 1, 2, dan 3. Nilai `flex` ini berfungsi untuk menentukan proporsi ruang yang akan diambil oleh setiap elemen dibandingkan dengan elemen lainnya di dalam container. Dengan demikian, blok pertama akan menempati satu bagian dari total ruang, blok kedua dua bagian, dan blok ketiga tiga bagian. Hasilnya, tinggi ketiga blok tersebut akan berbeda secara proporsional sesuai nilai `flex` yang diberikan.

## 12 Screen PercentageDimensionsBasics

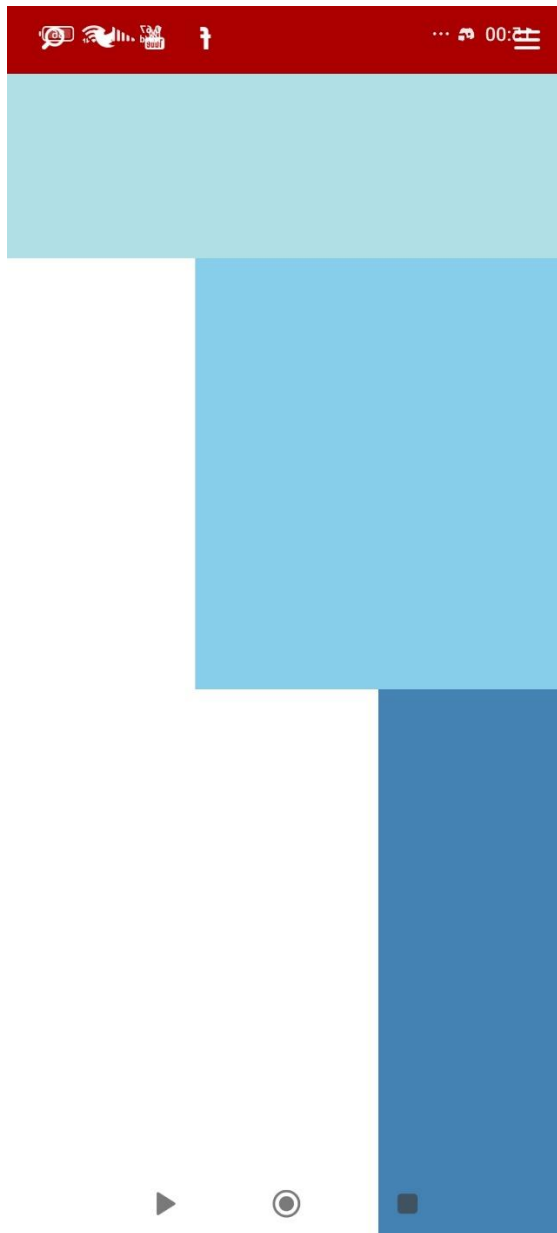
### Input:

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

const PercentageDimensionsBasics = () => {
  return (
    <View style={{ height: "100%" }}>
      <View
        style={{
          height: "15%",
          backgroundColor: "powderblue",
        }}
      />
      <View
        style={{
          width: "66%",
          height: "35%",
          backgroundColor: "skyblue",
        }}
      />
      <View
        style={{
          width: "33%",
          height: "50%",
          backgroundColor: "steelblue",
        }}
      />
    </View>
  );
};

export default PercentageDimensionsBasics;
```

## Output:



## Penjelasan:


Di dalam komponen ini, terdapat satu elemen utama `<View>` yang berperan sebagai wadah (container) dengan properti `style={{ height: "100%" }}`, yang berarti tinggi elemen tersebut akan memenuhi seluruh tinggi layar perangkat.

Di dalam container utama tersebut terdapat tiga elemen `<View>` lainnya yang masing-masing memiliki ukuran lebar (width) dan tinggi (height) dalam satuan persentase (%). Elemen pertama memiliki `height: "15%"` dan diberi warna `powderblue`, yang berarti hanya menempati 15% dari tinggi container utama. Elemen kedua memiliki `width: "66%"` dan `height: "35%"` dengan warna `skyblue`, sehingga menempati 66% lebar dan 35% tinggi dari container. Sementara itu, elemen ketiga memiliki `width: "33%"` dan `height: "50%"` dengan warna `steelblue`, yang berarti mengisi sepertiga lebar dan setengah tinggi dari ruang utama.



## 13 Screen FlexBasic

Input:



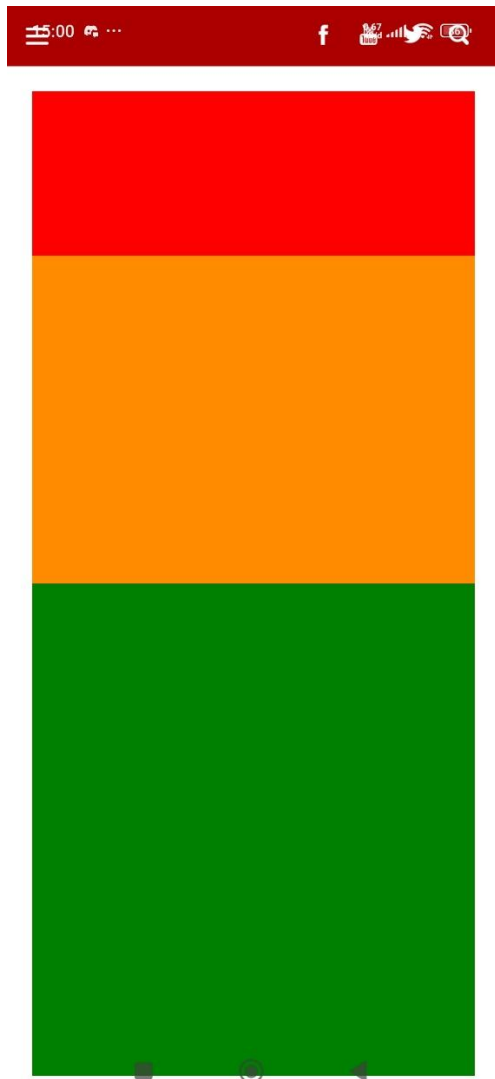
```
import React from "react";
import { StyleSheet, View } from "react-native";

const FlexBasic = () => {
  return (
    <View style={[styles.container, { flexDirection: "column" }]}>
      <View style={{ flex: 1, backgroundColor: "red" }} />
      <View style={{ flex: 2, backgroundColor: "darkorange" }} />
      <View style={{ flex: 3, backgroundColor: "green" }} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
  },
});

export default FlexBasic;
```

## Output:



## Penjelasan:

Di dalam fungsi FlexBasic, terdapat elemen utama `<View>` yang berperan sebagai container, dengan gaya gabungan dari `styles.container` dan properti tambahan `{ flexDirection: "column" }`. Nilai `flexDirection: "column"` menunjukkan bahwa susunan elemen anak diatur secara vertikal dari atas ke bawah. Di dalam container tersebut terdapat tiga elemen `<View>` anak yang masing-masing memiliki nilai properti `flex` berbeda, yaitu `flex: 1`, `flex: 2`, dan `flex: 3`, serta diberi warna latar berbeda: merah, oranye tua, dan hijau. Nilai properti `flex` ini menentukan proporsi ruang yang ditempati oleh setiap elemen terhadap total ruang yang tersedia dalam container. Dengan demikian, elemen hijau akan menempati ruang paling besar, diikuti oleh elemen oranye, dan yang paling kecil adalah elemen merah.

Selanjutnya, pada bagian bawah kode, didefinisikan objek `StyleSheet` bernama `styles` yang berisi properti container dengan nilai `flex: 1` dan `padding: 20`. Properti `flex: 1` pada container memastikan bahwa komponen utama mengisi seluruh ruang layar yang tersedia, sedangkan `padding: 20` memberikan jarak antara tepi layar dan elemen-elemen di dalamnya.

## 14 Screen FlexDirectionBasics

### Input:

```
import React, { useState } from "react";
import { StyleSheet, Text, TouchableOpacity, View } from "react-native";

const FlexDirectionBasics = () => {
  const [flexDirection, setflexDirection] = useState("column");

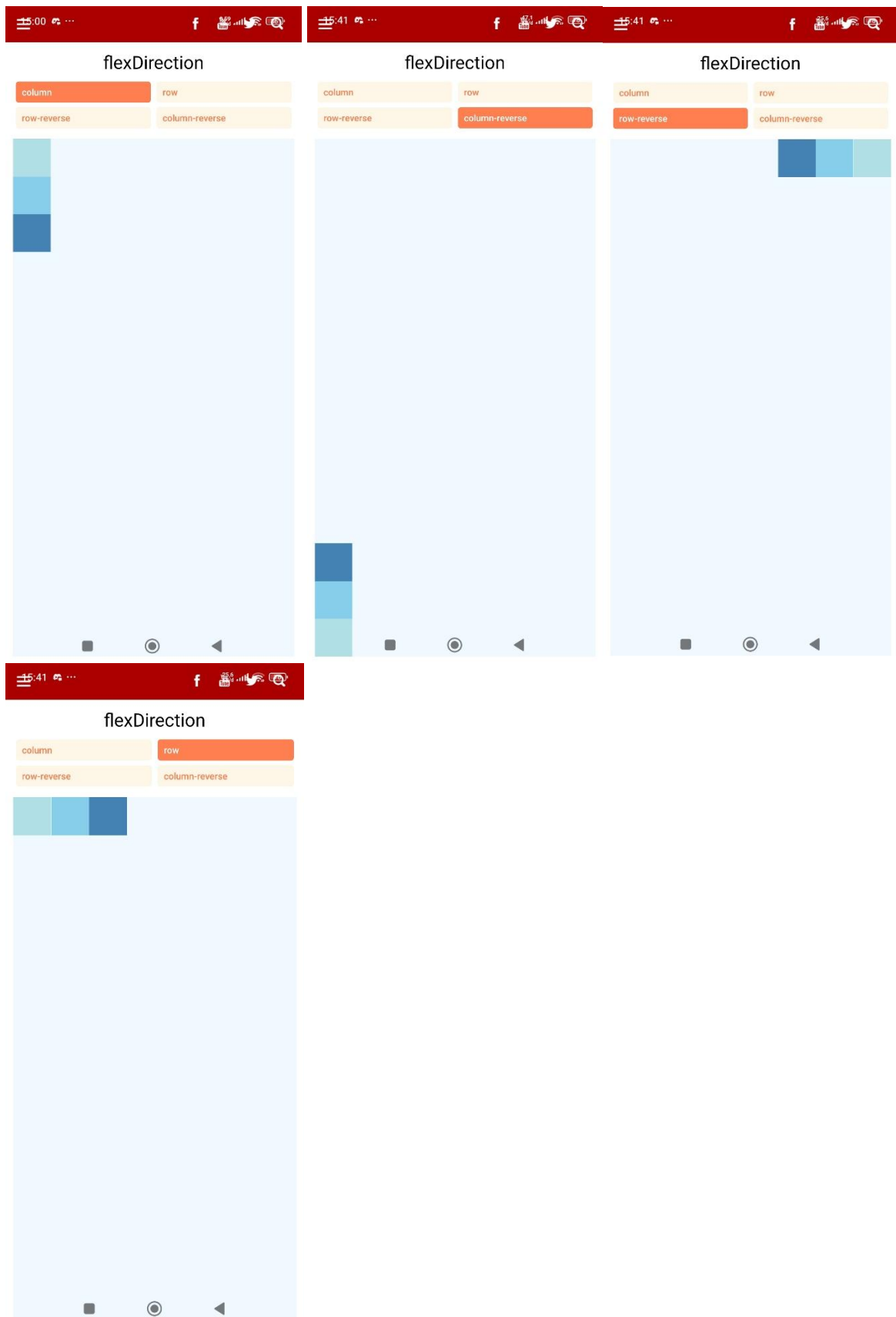
  return (
    <PreviewLayout
      label="flexDirection"
      values={["column", "row", "row-reverse", "column-reverse"]}
      selectedValue={flexDirection}
      setSelectedValue={setflexDirection}
    >
      <View style={styles.box, { backgroundColor: "powderblue" }} />
      <View style={styles.box, { backgroundColor: "skyblue" }} />
      <View style={styles.box, { backgroundColor: "steelblue" }} />
    </PreviewLayout>
  );
};

const PreviewLayout = ({
  label,
  children,
  values,
  selectedValue,
  setSelectedValue,
}) => {
  <View style={{ padding: 10, flex: 1 }}>
    <Text style={styles.label}>{label}</Text>
    <View style={styles.row}>
      {values.map((value) => (
        <TouchableOpacity
          key={value}
          onPress={() => setSelectedValue(value)}
          style={styles.button, selectedValue === value && styles.selected}
        >
          <Text
            style={
              styles.buttonLabel,
              selectedValue === value && styles.selectedLabel,
            }
          >
            {value}
          </Text>
        </TouchableOpacity>
      ))}
    </View>
    <View style={styles.container, { [label]: selectedValue }}>
      {children}
    </View>
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: 8,
    backgroundColor: "aliceblue",
  },
  box: {
    width: 50,
    height: 50,
  },
  row: {
    flexDirection: "row",
    flexWrap: "wrap",
  },
  button: {
    paddingHorizontal: 8,
    paddingVertical: 6,
    borderRadius: 4,
    backgroundColor: "oldlace",
    alignSelf: "flex-start",
    marginHorizontal: "1%",
    marginBottom: 6,
    minWidth: "48%",
    textAlign: "center",
  },
  selected: {
    backgroundColor: "coral",
    borderWidth: 0,
  },
  buttonLabel: {
    fontSize: 12,
    fontWeight: "500",
    color: "coral",
  },
  selectedLabel: {
    color: "white",
  },
  label: {
    textAlign: "center",
    marginBottom: 10,
    fontSize: 24,
  },
});

export default FlexDirectionBasics;
```

## Output:



**Penjelasan:**

Pada awalnya, nilai `flexDirection` diinisialisasi dengan “column”, yang berarti elemen-elemen akan tersusun secara vertikal dari atas ke bawah. Komponen ini memanfaatkan komponen turunan bernama `PreviewLayout`, yang berfungsi sebagai wadah (layout preview) untuk menampilkan perubahan arah tata letak berdasarkan pilihan pengguna. Komponen `PreviewLayout` menerima beberapa properti, yaitu `label`, `values`, `selectedValue`, dan `setSelectedValue`. Di dalamnya, terdapat beberapa tombol (diciptakan menggunakan `TouchableOpacity`) yang menampilkan pilihan arah tata letak seperti “column”, “row”, “row-reverse”, dan “column-reverse”. Ketika pengguna menekan salah satu tombol, fungsi `setSelectedValue` akan dipanggil untuk memperbarui nilai `flexDirection` sesuai dengan pilihan tersebut.

Setelah nilai `flexDirection` berubah, container utama akan menyesuaikan arah penataan tiga elemen `<View>` berwarna `powderblue`, `skyblue`, dan `steelblue` secara otomatis sesuai dengan nilai properti `flexDirection` yang sedang aktif. Dengan demikian, pengguna dapat secara langsung melihat bagaimana perubahan nilai `flexDirection` memengaruhi susunan elemen di layar.

Selain itu, objek `StyleSheet` digunakan untuk mengatur gaya tampilan seluruh komponen, seperti pengaturan ukuran kotak (`width` dan `height`), warna latar tombol, teks label, serta tata letak baris tombol menggunakan `flexDirection: "row"`. Properti `selected` dan `selectedLabel` digunakan untuk memberikan efek visual berbeda pada tombol yang sedang dipilih, yaitu perubahan warna latar menjadi coral dan warna teks menjadi putih.

## 15 Screen JustifyContentBasics

Input:

```
import React, { useState } from "react";
import { View, TouchableOpacity, Text, StyleSheet } from "react-native";

const JustifyContentBasics = () => {
  const [justifyContent, setJustifyContent] = useState("flex-start");

  return (
    <PreviewLayout
      label="JustifyContent"
      selectedValue={justifyContent}
      values={[
        "flex-start",
        "flex-end",
        "center",
        "space-between",
        "space-around",
        "space-evenly",
      ]}
      setSelectedValue={setJustifyContent}
    >
      <View style={([styles.box, { backgroundColor: "powderblue" }])} />
      <View style={([styles.box, { backgroundColor: "skyblue" }])} />
      <View style={([styles.box, { backgroundColor: "steelblue" }])} />
    </PreviewLayout>
  );
};

const PreviewLayout = ({
  label,
  children,
  values,
  selectedValue,
  setSelectedValue,
}) => (
  <View style={{ padding: 10, flex: 1 }}>
    <Text style={styles.label}>{label}</Text>
    <View style={styles.row}>
      {values.map((value) => (
        <TouchableOpacity
          key={value}
          onPress={() => setSelectedValue(value)}
          style={([styles.button, selectedValue === value && styles.selected])}
        >
          <Text
            style={([
              styles.buttonLabel,
              selectedValue === value && styles.selectedLabel,
            ])}
          >
            {value}
          </Text>
        </TouchableOpacity>
      ))}
    </View>
    <View style={styles.container, { [label]: selectedValue }}>
      {children}
    </View>
  </View>
);

const styles = StyleSheet.create({
  container: {
    flexDirection: "column",
    flex: 1,
    marginTop: 8,
    backgroundColor: "aliceblue",
  },
  box: {
    width: 50,
    height: 50,
  },
  row: {
    flexDirection: "row",
    flexWrap: "wrap",
  },
  button: {
    paddingHorizontal: 8,
    paddingVertical: 6,
    borderRadius: 4,
    backgroundColor: "oldlace",
    alignSelf: "flex-start",
    marginHorizontal: "1%",
    marginBottom: 6,
    minWidth: "48%",
    textAlign: "center",
  },
  selected: {
    backgroundColor: "coral",
    borderWidth: 0,
  },
  buttonLabel: {
    fontSize: 12,
    fontWeight: "500",
    color: "coral",
  },
  selectedLabel: {
    color: "white",
  },
  label: {
    textAlign: "center",
    marginBottom: 10,
    fontSize: 24,
  },
});

export default JustifyContentBasics;
```

Output:

justifyContent

flex-start


flex-end

center

space-between

space-around

space-evenly



justifyContent

flex-start


flex-end

center

space-between

space-around

space-evenly



justifyContent

flex-start


flex-end

center

space-between

space-around

space-evenly



justifyContent

flex-start

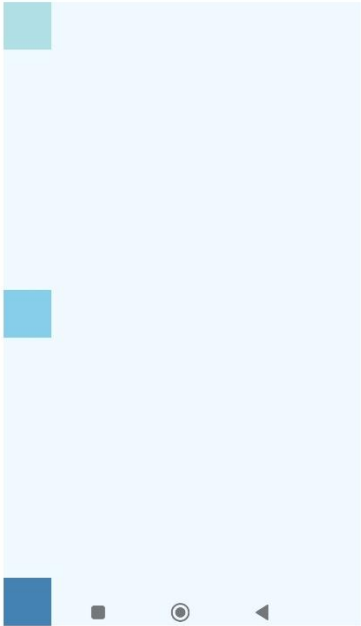
flex-end

center

space-between

space-around

space-evenly



justifyContent

flex-start


flex-end

center

space-between

space-around

space-evenly



justifyContent

flex-start

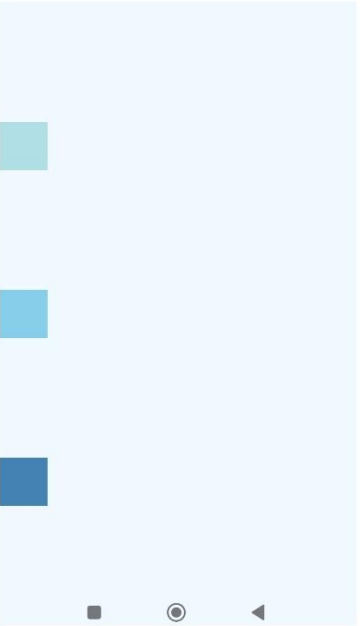
flex-end

center

space-between

space-around

space-evenly



**Penjelasan:**

Secara default, nilai awal dari `justifyContent` adalah "flex-start", yang berarti elemen-elemen anak akan ditempatkan di awal area container. Komponen ini menggunakan komponen turunan bernama `PreviewLayout`, yang berperan sebagai wadah interaktif untuk menampilkan efek perubahan tata letak berdasarkan nilai properti yang dipilih. Komponen `PreviewLayout` menerima beberapa properti, antara lain `label`, `values`, `selectedValue`, dan `setSelectedValue`. Melalui elemen `TouchableOpacity`, pengguna dapat memilih salah satu dari beberapa opsi pengaturan `justifyContent`, yaitu "flex-start", "flex-end", "center", "space-between", "space-around", dan "space-evenly". Setiap kali pengguna menekan tombol tertentu, fungsi `setSelectedValue` akan memperbarui nilai `justifyContent`, sehingga tampilan tata letak langsung berubah secara dinamis di layar.

Dalam area pratinjau (View dengan gaya `styles.container`), terdapat tiga elemen berbentuk persegi kecil dengan warna berbeda `powderblue`, `skyblue`, dan `steelblue` yang posisinya berubah sesuai dengan nilai `justifyContent` yang dipilih. Misalnya, ketika pengguna memilih "center", ketiga kotak akan berada di tengah secara vertikal; sedangkan jika memilih "space-between", jarak antar elemen akan merata dari atas hingga bawah. Hal ini membantu pengguna memahami perbedaan setiap nilai properti `justifyContent` secara visual dan langsung.



## 16 Screen AlignItemsLayout

### Input:

```
import React, { useState } from "react";
import { View, TouchableOpacity, Text, StyleSheet } from "react-native";

const AlignItemsLayout = () => {
  const [alignItems, setAlignItems] = useState("stretch");

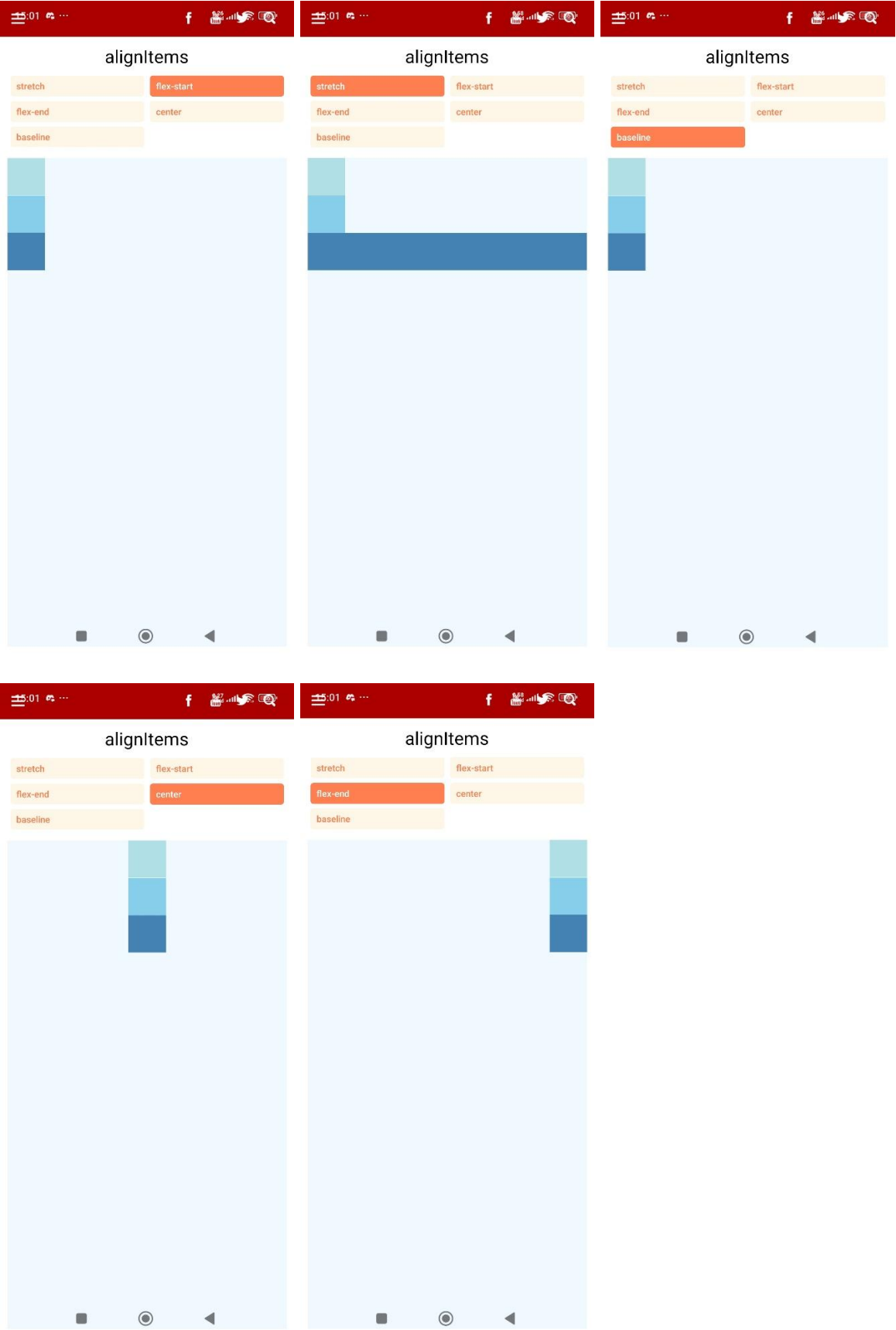
  return (
    <PreviewLayout
      label="alignItems"
      selectedValue={alignItems}
      values={["stretch", "flex-start", "flex-end", "center", "baseline"]}
      setSelectedValue={setAlignItems}
    >
      <View style={styles.box, { backgroundColor: "powderblue" }} />
      <View style={styles.box, { backgroundColor: "skyblue" }} />
      <View
        style={[
          styles.box,
          {
            backgroundColor: "steelblue",
            width: "auto",
            minWidth: 50,
          },
        ]
      />
    </PreviewLayout>
  );
};

const PreviewLayout = ({
  label,
  children,
  values,
  selectedValue,
  setSelectedValue,
}) => (
  <View style={{ padding: 10, flex: 1 }}>
    <Text style={styles.label}>{label}</Text>
    <View style={styles.row}>
      {values.map((value) => (
        <TouchableOpacity
          key={value}
          onPress={() => setSelectedValue(value)}
          style={styles.button, selectedValue === value && styles.selected}
        >
          <Text
            style={styles.buttonLabel,
              selectedValue === value && styles.selectedLabel,
            >
            {value}
          </Text>
        </TouchableOpacity>
      ))}
    </View>
    <View style={styles.container, { [label]: selectedValue }}>
      {children}
    </View>
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: 8,
    backgroundColor: "aliceblue",
    minHeight: 200,
  },
  box: {
    width: 50,
    height: 50,
  },
  row: {
    flexDirection: "row",
    flexWrap: "wrap",
  },
  button: {
    paddingHorizontal: 8,
    paddingVertical: 6,
    borderRadius: 4,
    backgroundColor: "oldlace",
    alignSelf: "flex-start",
    marginHorizontal: "1%",
    marginBottom: 6,
    minWidth: "48%",
    textAlign: "center",
  },
  selected: {
    backgroundColor: "coral",
    borderWidth: 0,
  },
  buttonLabel: {
    fontSize: 12,
    fontWeight: "500",
    color: "coral",
  },
  selectedLabel: {
    color: "white",
  },
  label: {
    textAlign: "center",
    marginBottom: 10,
    fontSize: 24,
  },
});

export default AlignItemsLayout;
```

Output:



### **Penjelasan:**

Nilai awal dari `alignItems` ditetapkan sebagai "stretch", yang berarti seluruh elemen anak di dalam container akan menyesuaikan lebarnya agar memenuhi ruang yang tersedia secara vertikal maupun horizontal tergantung pada orientasi flex container.

Komponen ini memanfaatkan komponen turunan bernama `PreviewLayout`, yang berfungsi sebagai wadah interaktif untuk menampilkan hasil perubahan tata letak berdasarkan nilai properti yang sedang dipilih. Melalui beberapa tombol yang ditampilkan (`TouchableOpacity`), pengguna dapat memilih satu dari lima opsi nilai `alignItems`, yaitu "stretch", "flex-start", "flex-end", "center", dan "baseline". Setiap kali tombol ditekan, fungsi `setAlignItems` akan memperbarui nilai state sehingga tata letak elemen-elemen di area pratinjau berubah secara langsung, memperlihatkan bagaimana setiap nilai `alignItems` memengaruhi perataan elemen secara vertikal dalam container.

Dalam area pratinjau (View dengan gaya `styles.container`), terdapat tiga elemen berbentuk kotak dengan warna berbeda `powderblue`, `skyblue`, dan `steelblue` yang digunakan untuk menunjukkan efek visual dari berbagai pengaturan perataan. Misalnya, ketika pengguna memilih "flex-start", semua kotak akan sejajar di bagian atas container; sedangkan jika "center", kotak akan berada di tengah secara vertikal. Opsi "baseline" digunakan untuk menyelaraskan elemen berdasarkan garis dasar teks di dalamnya, meskipun dalam contoh ini efeknya minimal karena elemen tidak berisi teks.

# 17 Screen WidthHeightBasics

## Input:

```
import React, { useState } from "react";
import {
  View,
  SafeAreaView,
  TouchableOpacity,
  Text,
  StyleSheet,
} from "react-native";

const WidthHeightBasics = () => {
  const [widthType, setWidthType] = useState("auto");
  const [heightType, setHeightType] = useState("auto");

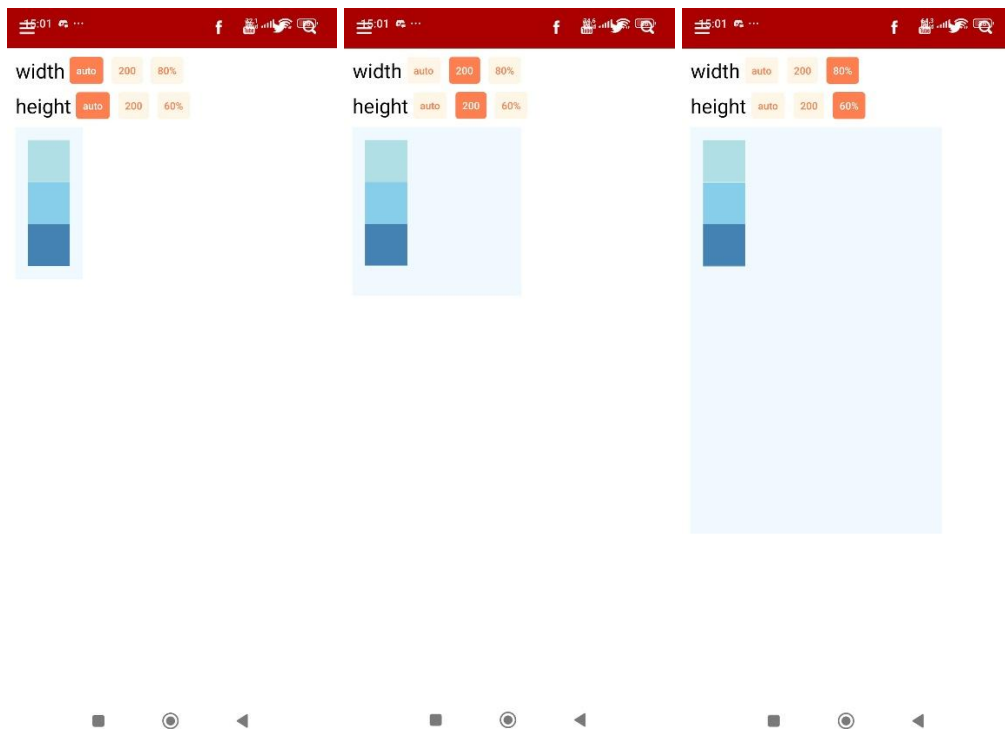
  return (
    <PreviewLayout
      widthType={widthType}
      heightType={heightType}
      widthValues={['auto', 200, '80%']}
      heightValues={['auto', 200, '60%']}
      setWidthType={setWidthType}
      setHeightType={setHeightType}
    >
      <View
        style={{
          alignSelf: "flex-start",
          backgroundColor: "aliceblue",
          height: heightType,
          width: widthType,
          padding: 15,
        }}
      >
        <View style={styles.box, { backgroundColor: "powderblue" }} />
        <View style={styles.box, { backgroundColor: "skyblue" }} />
        <View style={styles.box, { backgroundColor: "steelblue" }} />
      </View>
    </PreviewLayout>
  );
};

const PreviewLayout = ({
  children,
  widthType,
  heightType,
  widthValues,
  heightValues,
  setWidthType,
  setHeightType,
}) => {
  <SafeAreaView style={{ flex: 1, padding: 10 }}>
    <View style={styles.row}>
      <Text style={styles.label}>width </Text>
      {widthValues.map((value) => {
        <TouchableOpacity
          key={value}
          onPress={() => setWidthType(value)}
          style={([styles.button, widthType === value && styles.selected])}
        >
          <Text
            style={([
              styles.buttonLabel,
              widthType === value && styles.selectedLabel,
            ])}
          >
            {value}
          </Text>
        </TouchableOpacity>
      })}
    </View>
    <View style={styles.row}>
      <Text style={styles.label}>height </Text>
      {heightValues.map((value) => {
        <TouchableOpacity
          key={value}
          onPress={() => setHeightType(value)}
          style={([styles.button, heightType === value && styles.selected])}
        >
          <Text
            style={([
              styles.buttonLabel,
              heightType === value && styles.selectedLabel,
            ])}
          >
            {value}
          </Text>
        </TouchableOpacity>
      })}
    </View>
  </SafeAreaView>
);

const styles = StyleSheet.create({
  box: {
    width: 50,
    height: 50,
  },
  row: {
    flexDirection: "row",
    flexWrap: "wrap",
  },
  button: {
    padding: 8,
    borderRadius: 4,
    backgroundColor: "oldlace",
    alignSelf: "flex-start",
    marginRight: 10,
    marginBottom: 10,
  },
  selected: {
    backgroundColor: "coral",
    shadowOpacity: 0,
    borderWidth: 0,
  },
  buttonLabel: {
    fontSize: 12,
    fontWeight: "500",
    color: "coral",
  },
  selectedLabel: {
    color: "white",
  },
  label: {
    textAlign: "center",
    marginBottom: 10,
    fontSize: 24,
  },
});

export default WidthHeightBasics;
```

## Output:



## Penjelasan:

Dua state `widthType` dan `heightType` menyimpan pilihan lebar dan tinggi saat ini, dengan nilai awal "auto". Nilai "auto" memungkinkan elemen menyesuaikan ukurannya secara otomatis berdasarkan isi di dalamnya, sedangkan nilai numerik seperti 200 dan nilai persentase seperti "80%" menunjukkan ukuran tetap atau relatif terhadap container induk.

Komponen `WidthHeightBasics` memanfaatkan komponen turunan `PreviewLayout` sebagai wadah interaktif untuk menampilkan perubahan visual berdasarkan pilihan pengguna. Melalui baris tombol (`TouchableOpacity`) yang diatur dalam dua kelompok, pengguna dapat memilih variasi nilai untuk properti `width` dan `height`. Ketika tombol ditekan, fungsi `setWidthType` atau `setHeightType` akan memperbarui nilai state, sehingga tampilan pratinjau berubah secara langsung. Hal ini memberikan ilustrasi bagaimana perbedaan satuan ukuran otomatis, absolut (dalam satuan piksel), dan relatif (dalam persen) memengaruhi dimensi tampilan elemen pada layar.

Bagian utama dari area pratinjau adalah sebuah elemen `View` dengan latar belakang `aliceblue` yang menyesuaikan lebar dan tinggi sesuai pilihan pengguna. Di dalamnya terdapat tiga kotak kecil berwarna `powderblue`, `skyblue`, dan `steelblue` sebagai konten visual untuk memperlihatkan perubahan ruang yang dihasilkan oleh pengaturan dimensi. Ketika pengguna memilih nilai lebar "80%", misalnya, kontainer akan menempati 80% dari lebar layar; sedangkan jika memilih 200, lebarnya menjadi tetap sebesar 200 piksel. Sementara itu, elemen `SafeAreaView` digunakan untuk memastikan tampilan tetap aman dari area layar yang sensitif seperti notch atau status bar pada perangkat modern.

# 18 Screen PositionLayout

## Input:

```
import React, { useState } from "react";
import {
  View,
  SafeAreaView,
  TouchableOpacity,
  Text,
  StyleSheet,
} from "react-native";

const PositionLayout = () => {
  const [position, setPosition] = useState("relative");

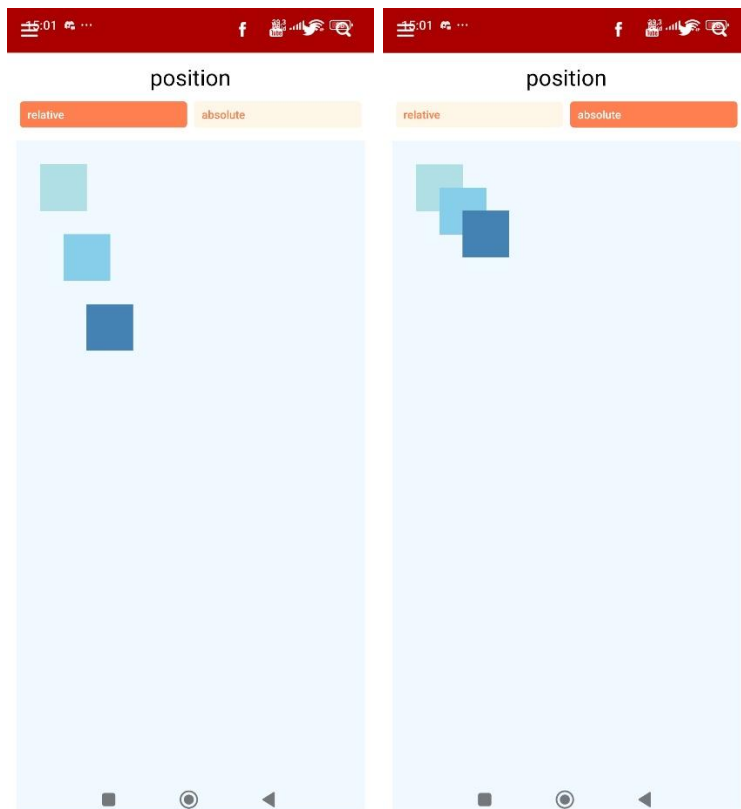
  return (
    <PreviewLayout
      label="position"
      selectedValue={position}
      values={["relative", "absolute"]}
      setSelectedValue={setPosition}
    >
      <View
        style={[
          styles.box,
          {
            top: 25,
            left: 25,
            position,
            backgroundColor: "powderblue",
          },
        ]
      />
      <View
        style={[
          styles.box,
          {
            top: 50,
            left: 50,
            position,
            backgroundColor: "skyblue",
          },
        ]
      />
      <View
        style={[
          styles.box,
          {
            top: 75,
            left: 75,
            position,
            backgroundColor: "steelblue",
          },
        ]
      />
    </PreviewLayout>
  );
};

const PreviewLayout = ({
  label,
  children,
  values,
  selectedValue,
  setSelectedValue,
}) => {
  <view style={{ padding: 10, flex: 1 }}>
    <text style={styles.label}>{label}</text>
    <view style={styles.row}>
      {values.map((value) => (
        <TouchableOpacity
          key={value}
          onPress={() => setSelectedValue(value)}
          style={styles.button, selectedValue === value && styles.selected}
        >
          <text
            style={[
              styles.buttonLabel,
              selectedValue === value && styles.selectedLabel,
            ]}
          >
            {value}
          </text>
        </TouchableOpacity>
      ))}
    </view>
    <view style={styles.container}>{children}</view>
  </view>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    margin: 10,
    backgroundColor: "aliceblue",
    minHeight: 200,
  },
  box: {
    width: 50,
    height: 50,
  },
  row: {
    flexDirection: "row",
    flexWrap: "wrap",
  },
  button: {
    padding: 8,
    paddingVertical: 6,
    borderRadius: 4,
    backgroundColor: "oldlace",
    alignSelf: "flex-start",
    margin: 10,
    marginHorizontal: 6,
    minWidth: 40,
    textAlign: "center",
  },
  selected: {
    backgroundColor: "coral",
    borderWidth: 0,
  },
  buttonLabel: {
    fontSize: 12,
    fontWeight: "500",
    color: "coral",
  },
  selectedLabel: {
    color: "white",
  },
  label: {
    textAlign: "center",
    margin: 10,
    fontSize: 24,
  },
});

export default PositionLayout;
```

## Output:



## Penjelasan:

Komponen `PreviewLayout` bertugas menampilkan label, deretan tombol pilihan posisi, dan area tampilan (container) yang berisi tiga buah kotak (View) berwarna berbeda. Tiap kotak memiliki ukuran tetap (50x50 piksel) dan diberikan nilai `top` serta `left` yang berbeda untuk menunjukkan efek dari perubahan properti `position`. Ketika posisi diset ke `relative`, setiap kotak akan bergeser relatif terhadap posisi normalnya dalam aliran layout. Namun, jika diset ke `absolute`, setiap kotak akan ditempatkan berdasarkan koordinat absolut relatif terhadap container induknya.

Bagian `StyleSheet` mendefinisikan gaya visual elemen-elemen seperti tata letak baris (`row`), tombol (`button`), label (`label`), dan kontainer (`container`). Warna latar belakang seperti `aliceblue`, `coral`, serta variasi warna biru digunakan untuk membedakan elemen secara visual.

## 19 Screen DisplayAnImageWithStyle

### Input:

```
import React from "react";
import { View, Image, Text, StyleSheet, ScrollView } from "react-native";
import { Separator } from "../components";

const DisplayAnImageWithStyle = () => {
  return (
    <ScrollView>
      <View style={styles.container}>
        <View>
          <Image
            style={{
              resizeMode: "cover",
              ...styles.image,
            }}
            source={require("../assets/js.png")}
          />
          <Text>resizeMode : cover</Text>
        </View>

        <Separator height={30} />

        <View>
          <Image
            style={{
              resizeMode: "contain",
              ...styles.image,
            }}
            source={require("../assets/js.png")}
          />
          <Text>resizeMode : contain</Text>
        </View>

        <Separator height={30} />

        <View>
          <Image
            style={{
              resizeMode: "stretch",
              ...styles.image,
            }}
            source={require("../assets/js.png")}
          />
          <Text>resizeMode : stretch</Text>
        </View>

        <Separator height={30} />

        <View>
          <Image
            style={{
              resizeMode: "repeat",
              ...styles.image,
            }}
            source={require("../assets/js.png")}
          />
          <Text>resizeMode : repeat</Text>
        </View>

        <Separator height={30} />

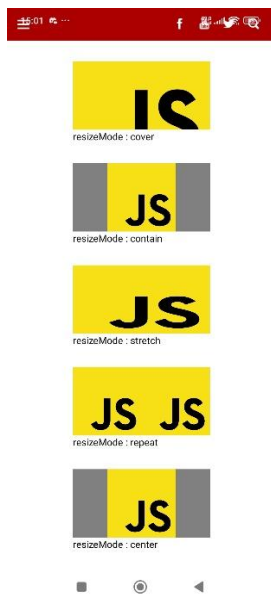
        <View>
          <Image
            style={{
              resizeMode: "center",
              ...styles.image,
            }}
            source={require("../assets/js.png")}
          />
          <Text>resizeMode : center</Text>
        </View>
      </ScrollView>
    );
  };

  const styles = StyleSheet.create({
    container: {
      justifyContent: "center",
      alignItems: "center",
      padding: 30,
    },
    image: {
      height: 100,
      width: 200,
      backgroundColor: "grey",
    },
  });
});

export default DisplayAnImageWithStyle;
```



## Output:



## Penjelasan:

Komponen utama bernama `DisplayAnImageWithStyle` menampilkan beberapa variasi gambar yang sama, namun masing-masing memiliki nilai `resizeMode` berbeda seperti `cover`, `contain`, `stretch`, `repeat`, dan `center`. Tujuan dari kode ini adalah untuk memperlihatkan bagaimana setiap mode tersebut memengaruhi cara gambar menyesuaikan diri terhadap ukuran wadah (container) yang telah ditentukan.

Struktur komponen diawali dengan `ScrollView` agar seluruh konten dapat digulir ke bawah jika melebihi tinggi layar. Di dalamnya terdapat elemen `View` dengan gaya `styles.container`, yang mengatur posisi elemen secara terpusat melalui properti `justifyContent` dan `alignItems`. Setiap gambar ditampilkan menggunakan komponen `Image` yang sumbernya berasal dari direktori lokal (`../assets/js.png`). Properti `style` pada setiap gambar menggabungkan pengaturan dari `styles.image` yang menentukan tinggi (100 piksel), lebar (200 piksel), dan latar belakang abu-abu dengan nilai `resizeMode` yang bervariasi.

Nilai `resizeMode` memiliki fungsi berbeda:

- `cover` memperbesar gambar hingga memenuhi seluruh ruang tanpa mengubah rasio aspek, sehingga sebagian gambar bisa terpotong.
- `contain` menyesuaikan gambar agar seluruh bagian terlihat tanpa memotong, dengan menjaga rasio aslinya.
- `stretch` mengubah ukuran gambar agar mengisi seluruh area meski rasio aslinya berubah.
- `repeat` menggandakan gambar secara berulang untuk mengisi area.
- `center` menampilkan gambar di tengah tanpa mengubah ukuran aslinya.

Selain itu, komponen `Separator` digunakan untuk memberi jarak antar tampilan gambar, menjaga keterbacaan dan kerapian tata letak.

## 20 [Update] Install Gluestack UI (Alternatif)

### Input:

Install Gluestack dan dependencies nya dengan mengeksekusi command berikut ini pada terminal / command prompt:

```
npm install @gluestack-style/legend-motion-animation-driver @gluestack-style/react @gluestack-ui/config @gluestack-ui/themed @react-native-aria/focus @react-native-aria/interactions @react-native-aria/overlays --legacy-peer-deps
```

atau

```
npm i @gluestack-ui/themed @gluestack-style/react react-native-svg@13.4.0 --legacy-peer-deps
```

```
import React from "react";
import { GluestackUIProvider, Heading, Center } from "@gluestack-ui/theme";
import { config } from "@gluestack-ui/config";

const GlueStackUIBasic = () => {
  return (
    <GluestackUIProvider config={config}>
      <Center flex={1}>
        <Heading>Gluestack UI</Heading>
      </Center>
    </GluestackUIProvider>
  );
};

export default GlueStackUIBasic;
```

### Output:



Gluestack UI

## Penjelasan:

Pada bagian awal, dilakukan impor beberapa elemen penting, yaitu GluestackUIProvider, Heading, dan Center dari pustaka `@gluestack-ui/themed`, serta config dari `@gluestack-ui/config`. Komponen GluestackUIProvider berfungsi sebagai penyedia konteks tema global (theme provider) yang memungkinkan seluruh komponen di dalamnya menggunakan konfigurasi gaya dari Gluestack secara konsisten. Di dalamnya, digunakan komponen Center untuk menempatkan konten tepat di tengah layar dengan properti `flex={1}`, yang menandakan elemen ini mengisi seluruh ruang yang tersedia. Sementara itu, komponen Heading digunakan untuk menampilkan teks “Gluestack UI” sebagai elemen judul utama.

Selanjutnya, file `index.js` pada folder `/screens` diperbarui dengan menambahkan ekspor (export) untuk komponen `GlueStackUIBasic`, agar dapat digunakan di seluruh bagian aplikasi. Kemudian, pada file `App.js`, komponen tersebut dimasukkan ke dalam daftar `pageArr`, yaitu array of objects yang berisi kumpulan halaman atau tampilan yang dapat dipilih dalam aplikasi. Setiap objek di dalam `pageArr` memiliki dua atribut: `name` sebagai judul tampilan dan `comp` yang berisi elemen React yang akan dirender di layar. Dengan menambahkan `{ name: "Gluestack UI Basic", comp: <GlueStackUIBasic /> }`, pengguna kini dapat mengakses tampilan demo dari Gluestack UI secara langsung melalui aplikasi.

## 21 Tugas

### Input:

- App.js

```
import { useRef, useState } from "react";
import { DrawerLayoutAndroid, StatusBar } from "react-native";
import { GluestackUIProvider } from "@gluestack-ui/themed";
import { config } from "@gluestack-ui/config";
import { Box, VStack } from "@gluestack-ui/themed";
import Header from "../components/header";
import Button from "../components/button";
import Separator from "../components/separator";
import List from "../screens/list";
import Article from "../screens/article";

// Functional Component
const App = () => {
  // State Declaration
  const [page, setPage] = useState("list");
  // Ref Declaration
  const drawer = useRef(null);

  // Arrow Function inside Functional Component
  const changePage = (drawer, pageName) => {
    // Close Drawer
    drawer.current.closeDrawer();
    // Change state value
    setPage(pageName);
  };

  // Arrow Function inside Functional Component
  const navigationView = () => (
    <Box bg="coolGray900" flex={1} p="8">
      <VStack space="md">
        <Button text="List" onPress={() => changePage(drawer, "list")} />
        <Separator height={30} />
        <Button text="Article" onPress={() => changePage(drawer, "article")} />
        <Separator height={30} />
        <Button text="Close" onPress={() => drawer.current.closeDrawer()} />
      </VStack>
    </Box>
  );

  return (
    <GluestackUIProvider config={config}>
      <DrawerLayoutAndroid
        ref={drawer}
        drawerWidth={300}
        drawerPosition="left"
        renderNavigationView={navigationView}
      >
        <StatusBar style="light" backgroundColor="#0C143C" />
        <Box flex={1}>
          <Header drawer={drawer} />
          {page == "list" ? <List /> : page == "article" ? <Article /> : null}
        </Box>
      </DrawerLayoutAndroid>
    </GluestackUIProvider>
  );
};

export default App;
```

- **gluestack-ui.config.js**

```
import { createConfig } from '@gluestack-style/react';
export const config = createConfig({
  aliases: {
    bg: 'backgroundColor',
    bgColor: 'backgroundColor',
    h: 'height',
    w: 'width',
    p: 'padding',
    px: 'paddingHorizontal',
    py: 'paddingVertical',
    pt: 'paddingTop',
    pb: 'paddingBottom',
    pl: 'paddingLeft',
    pr: 'paddingRight',
    m: 'margin',
    mx: 'marginHorizontal',
    my: 'marginVertical',
    mt: 'marginTop',
    mb: 'marginBottom',
    ml: 'marginLeft',
    mr: 'marginRight',
  },
  tokens: {
    colors: {
      // Crimson theme colors
      primary0: '#FFF5F5',
      primary50: '#FFE5E5',
      primary100: '#FFC2C2',
      primary200: '#FF9999',
      primary300: '#FF6868',
      primary400: '#FF4747',
      primary500: '#DC143C', // Crimson
      primary600: '#C41230',
      primary700: '#AA0028',
      primary800: '#8B001F',
      primary900: '#6E0019',
      primary950: '#4A0011',

      // Gray colors
      white: 'FFFFFF',
      black: '000000',
      coolGray50: 'F9F9FB',
      coolGray100: 'F3F4F6',
      coolGray200: 'E5E7EB',
      coolGray300: 'D1D5DB',
      coolGray400: '9CA3AF',
      coolGray500: '6B7280',
      coolGray600: '4B5563',
      coolGray700: '374151',
      coolGray800: '1F2937',
      coolGray900: '111827',
    },
    space: {
      0: 0,
      1: 4,
      2: 8,
      3: 12,
      4: 16,
      5: 20,
      6: 24,
      7: 28,
      8: 32,
      9: 36,
      10: 40,
      xs: 4,
      sm: 8,
      md: 12,
      lg: 16,
      xl: 20,
      '2xl': 24,
    },
    fontSizes: {
      xs: 12,
      sm: 14,
      md: 16,
      lg: 18,
      xl: 20,
      '2xl': 24,
    },
    lineHeights: {
      xs: 16,
      sm: 20,
      md: 22,
      lg: 24,
      xl: 28,
      '2xl': 32,
    },
    fontWeights: {
      normal: '400',
      medium: '500',
      semibold: '600',
      bold: '700',
    },
    radii: {
      none: 0,
      sm: 4,
      md: 8,
      lg: 12,
      xl: 16,
      '2xl': 20,
      full: 9999,
    },
  },
});
```

- **button.js**

```
import { Button as GluestackButton, ButtonText } from "@gluestack-ui/theme";
// Functional Component with props
const Button = (props) => {
  return (
    <GluestackButton
      onPress={props.onPress}
      bg="$primary500"
      borderRadius="$full"
      py="$4"
      px="$6"
      $hover={{
        bg: "$primary600",
      }}
      $active={{
        bg: "$primary700",
      }}
      shadowColor="$primary900"
      shadowOffset={{ width: 0, height: 2 }}
      shadowOpacity={0.25}
      shadowRadius={3.84}
      elevation={5}
    >
      <ButtonText
        color="$white"
        fontSize="$md"
        fontWeight="$bold"
        textTransform="uppercase"
      >
        {props.text}
      </ButtonText>
    </GluestackButton>
  );
};

export default Button;
```

- **header.js**

```
import { Image } from "react-native";
import { Box, HStack, Pressable } from "@gluestack-ui/themed";

// Functional Component with props
const Header = (props) => {
  return (
    <Box bg="$primary500" p="$4">
      <HStack
        justifyContent="space-between"
        alignItems="center"
        flexDirection="row"
        w="$full"
      >
        <Pressable onPress={() => props.drawer.current.openDrawer()}>
          <Box w={18} h={18}>
            <Image
              source={require("../assets/menu.png")}
              style={{ width: 18, height: 18, tintColor: '#fff' }}
            />
          </Box>
        </Pressable>
        <HStack
          alignItems="center"
          justifyContent="center"
          space="md"
          flexDirection="row"
        >
          <Box w={36} h={16}>
            <Image
              source={require("../assets/facebook.png")}
              style={{ width: 36, height: 16, resizeMode: "contain", tintColor: '#fff' }}
            />
          </Box>
          <Box w={36} h={16}>
            <Image
              source={require("../assets/youtube.png")}
              style={{ width: 36, height: 16, resizeMode: "contain", tintColor: '#fff' }}
            />
          </Box>
          <Box w={36} h={16}>
            <Image
              source={require("../assets/twitter.png")}
              style={{ width: 36, height: 16, resizeMode: "contain", tintColor: '#fff' }}
            />
          </Box>
          <Box w={36} h={16}>
            <Image
              source={require("../assets/search.png")}
              style={{ width: 36, height: 16, resizeMode: "contain", tintColor: '#fff' }}
            />
          </Box>
        </HStack>
      </HStack>
    </Box>
  );
};

export default Header;
```

- **separator.js**



```
import { Box } from "@gluestack-ui/themed";

// Functional Component with props
const Separator = (props) => {
  return <Box h={props.height} />;
};

export default Separator;
```

- **list.js**

```

1  #!/usr/bin/perl
2  #
3  # Copyright (c) 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 2680, 2681,
```

- article.js

```
import { Alert, Image as RNImage } from "react-native";
import {
  ScrollView,
  Box,
  VStack,
  Text,
  Heading,
  Divider,
} from "@gluestack-ui/themed";
import Separator from "../components/separator";
import Button from "../components/button";

// Functional Component
const Article = () => {
  // Arrow Function
  const buttonHandler = () => {
    Alert.alert("Button Handler");
  };

  return (
    <ScrollView bg="white">
      <VStack space="md">
        <Box alignItems="center" py="$4" px="$4">
          <Box w="80%" h="100">
            <RNImage
              source={{
                uri: "https://b3338070.smushcdn.com/3338070/wp-content/uploads/2024/03/logo-telkom-university-surabaya-color-300x133.png?lossy=2&strip=1&webp=1",
              }}
              style={{ height: 100, width: "100%", resizeMode: "contain" }}
            />
          </Box>
        </Box>

        <Box bg="$coolGray100" px="$4" py="$5">
          <Heading size="xl" color="$coolGray900" lineHeight="$2xl">
            Solusi Inovatif Penurunan Angka Stunting, Mahasiswa Telkom University
            Sabat Penghargaan di Innovillage 2023
          </Heading>
        </Box>

        <VStack space="md" px="$4">
          <Box borderRadius="$xl" overflow="hidden" h="(220)">
            <RNImage
              source={{
                uri: "https://b3338070.smushcdn.com/3338070/wp-content/uploads/2024/03/Innovillage-2023-1200x600.jpeg?lossy=2&strip=1&webp=1",
              }}
              style={{
                height: 220,
                width: "100%",
                resizeMode: "cover"
              }}
            />
          </Box>

          <Separator height={10} />

          <Text size="md" color="$coolGray800" lineHeight="$xl">
            <Text fontWeight="bold" color="$coolGray900">Surabaya, Maret 2024</Text>
            ("")- Gelombang kebahagiaan kembali datang bagi Telkom University Surabaya,
            tim mahasiswa Telkom University Surabaya yang menamakan diri mereka
            Connect Care Pediatrics berhasil meraih prestasi gemilang dalam
            kompetisi Innovillage 2023 yang diumumkan pada tanggal 09 Maret 2024,
            bertempat di Auditorium Gedung Damar, Telkom University. Sebuah
            kompetisi sosial project bergengsi ini diselenggarakan untuk memwadihi
            mahasiswa untuk melakukan kegiatan sosial di tengah-tengah masyarakat
            yang dapat mengatasi permasalahan yang dihadapi. Innovillage yang
            telah memasuki tahun ketiga ini mengangkat tema "Empowering Young
            Sociopreneur for National Development."
          </Text>

          <Text size="md" color="$coolGray800" lineHeight="$xl">
            Melalui inovasi proyek berjudul "Pembuatan IoT dan Perancangan Website
            'Connect Pediatrics' Untuk Monitoring Kesehatan Balita pada Posyandu",
            tim yang digawangi oleh Rendy Adi Fatma Saputra, Daffa Fakhuddin
            Arroyo, dan Johannes Abner Sebastian, bersama dengan dosen pembimbing
            Ika Sari Oktarina S.Kw, M.Ting, berhasil menciptakan inovasi luar
            biasa yang mendukung tujuan pembangunan berkelanjutan. Dan
            mengantarkan mereka meraih predikat "Best Stunting Reduction
            Solution".
          </Text>

          <Text size="md" color="$coolGray800" lineHeight="$xl">
            "Proyek ini fokus pada penurunan angka stunting dengan menghadirkan
            solusi inovatif berupa alat IoT yang terintegrasi dengan website. Alat
            ini tidak hanya mengukur berat badan dan tinggi badan balita, tetapi
            juga merekam data secara otomatis ke dalam database yang dapat diakses
            oleh orang tua. Informasi ini juga mempermudah pekerjaan bidan di
            Posyandu dalam penginputan data ke EPPGBM." jelas Rendy
          </Text>

          <Text size="md" color="$coolGray800" lineHeight="$xl">
            Sementara itu, anggota tim lainnya berbagi pengalaman yang menurutnya
            berkesan selama menjalankan proyek, "Program Innovillage ini seru dan
            memotivasi. Kami banyak keluar dari zona nyaman, menerapkan materi
            kuliah ke dalam kehidupan nyata masyarakat." Ujar Daffa, di tempat
            yang sama Johannes menambahkan, "Kami belajar bagaimana berinteraksi
            dengan masyarakat yang jarang terpapar teknologi, serta melihat
            permasalahan dari sudut pandang baru."
          </Text>

          <Text size="md" color="$coolGray800" lineHeight="$xl">
            Keberhasilan tim mahasiswa Connect Care Pediatrics ini bukan hanya
            sekadar pencapaian dalam sebuah kompetisi, melainkan bukti nyata
            kemampuan mereka dalam menerapkan pengetahuan dan keterampilan untuk
            memecahkan permasalahan sehari-hari. Prestasi ini diharapkan menjadi
            sumber inspirasi bagi mahasiswa lainnya untuk mengambil risiko,
            menerapkan ilmu yang mereka pelajari, dan menjadi agen perubahan dalam
            membangun bangsa.
          </Text>

          <Text size="md" color="$coolGray800" lineHeight="$xl">
            Detail menarik dari proyek innovillage ini bisa langsung dilihat di
            Instagram connectcare.pediatrics
          </Text>

          <Text size="md" fontWeight="bold" color="$coolGray900">
            Surabaya, Maret 2024
          </Text>

          <Divider bg="$coolGray300" />

          <Separator height={10} />

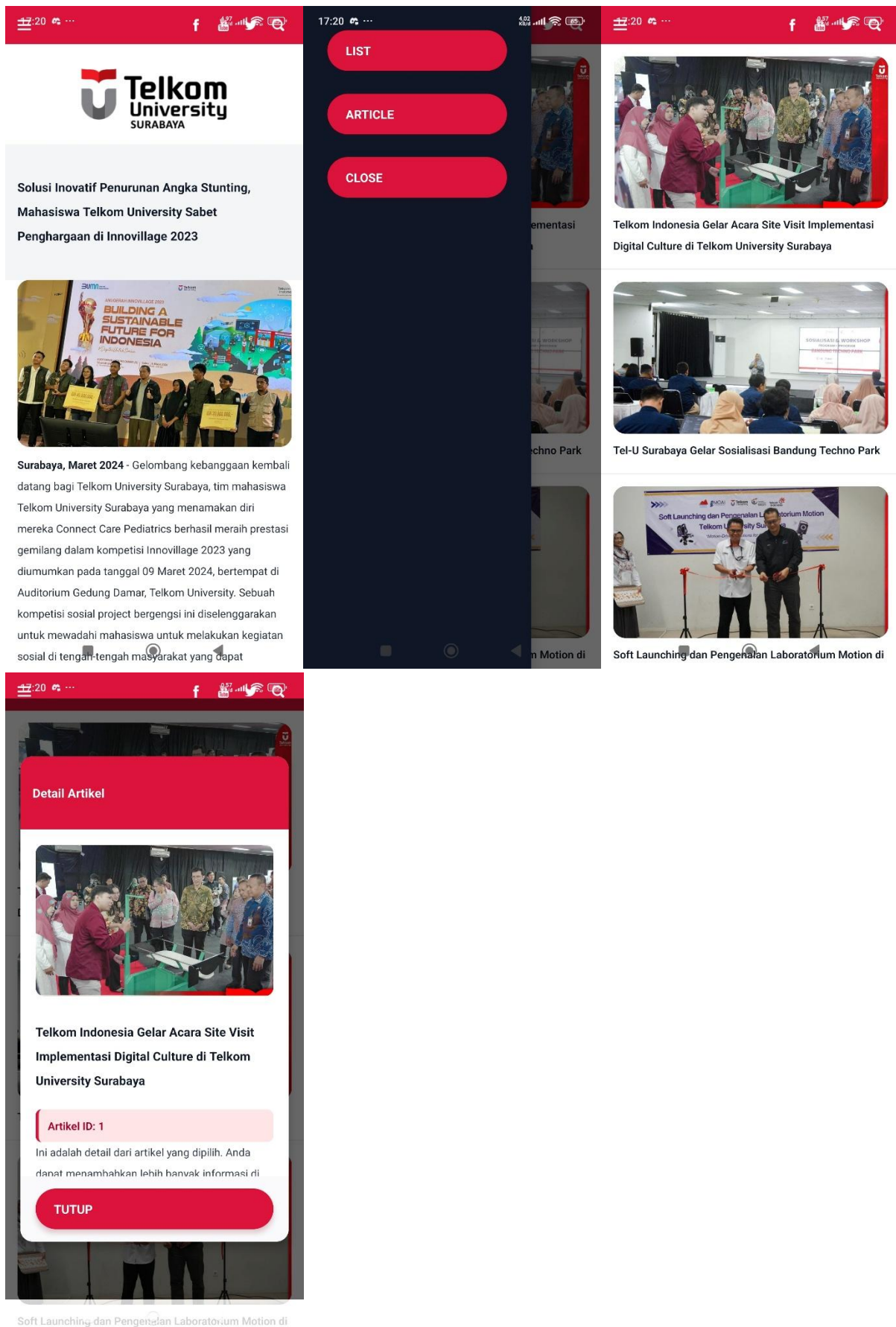
          <Button text="Share" onPress={buttonHandler} />

          <Separator height={70} />
        </VStack>
      </ScrollView>
    </ScrollView>
  );
};

export default Article;
```



## Output:



## Penjelasan:

Perbedaan mendasar pertama terletak pada dependency dan konfigurasi proyek. Project yang menggunakan gluestack memiliki dependensi tambahan berupa package `@gluestack-style/react`, `@gluestack-ui/config`, dan `@gluestack-ui/themed` yang terinstal di dalam file `package.json`. Selain itu, folder ini juga memiliki file konfigurasi khusus `gluestack-ui.config.js` yang berfungsi untuk mendefinisikan custom theme seperti sistem warna (color tokens), spacing, typography, dan border radius secara terpusat. Sebaliknya, project `Style/StyleSheet` hanya bergantung pada dependensi standar React Native tanpa library UI tambahan, sehingga ukuran proyek lebih ringan dan tidak memerlukan file konfigurasi tema eksternal.

Dari aspek implementasi komponen, terdapat perbedaan signifikan dalam cara penulisan kode. Pada project gluestack yang menggunakan GlueStack UI, komponen-komponen seperti Button, Header, dan layout container mengimport komponen pre-built dari `@gluestack-ui/themed` seperti Box, VStack, HStack, Pressable, Text, dan Heading. Styling diterapkan langsung pada komponen tersebut melalui props dengan sintaks yang lebih deklaratif, misalnya `bg="$primary500"`, `p="$4"`, atau `borderRadius="$x1"` yang mengacu pada token yang telah didefinisikan dalam file konfigurasi. Pendekatan ini mengadopsi konsep design tokens yang memungkinkan konsistensi tema di seluruh aplikasi. Sementara itu, project `Style/StyleSheet` menggunakan komponen core React Native seperti View, Text, TouchableOpacity, dan Image, dengan styling yang didefinisikan melalui object StyleSheet menggunakan method `StyleSheet.create()`. Setiap komponen memiliki konstanta styles terpisah di bagian bawah file yang berisi properti CSS-like dengan sintaks camelCase seperti `backgroundColor`, `paddingHorizontal`, dan `flexDirection`.

Perbedaan ketiga terletak pada struktur kode dan wrapper provider. File `App.js` pada project gluestack diawali dengan wrapper `<GluestackUIProvider config={config}>` yang membungkus seluruh aplikasi, memungkinkan semua child components mengakses tema dan konfigurasi global. Provider ini juga memfasilitasi fitur-fitur advanced seperti responsive design dan dark mode support. Komponen layout menggunakan Box dan VStack dari GlueStack sebagai pengganti View, yang memberikan API styling yang lebih intuitif dan konsisten. Di sisi lain, `App.js` pada project `Style/StyleSheet` tidak memerlukan provider khusus dan langsung menggunakan komponen View dengan inline styles atau reference ke StyleSheet object, yang merupakan pendekatan konvensional dalam pengembangan React Native.

Dari segi penerapan styling pada komponen Button, perbedaan implementasi sangat terlihat. Button pada project gluestack menggunakan komponen `GluestackButton` dan `ButtonText` yang telah memiliki state management bawaan untuk interaksi seperti hover dan active states melalui props `$hover` dan `$active`. Styling seperti shadow dan elevation juga langsung diterapkan sebagai props dengan nama yang semantik. Sebaliknya, Button pada project `Style/StyleSheet` menggunakan `TouchableOpacity` dengan komponen Text di dalamnya, dan seluruh styling didefinisikan dalam object styles yang terpisah dengan properti seperti container dan text, memerlukan penulisan kode yang lebih verbose untuk mencapai hasil visual yang sama.

Pada komponen Header, project gluestack menggunakan kombinasi Box, HStack, dan Pressable dari GlueStack UI dengan props styling yang didefinisikan inline seperti

`bg="$primary500", p="$4", justifyContent="space-between", dan alignItems="center". Props space="md" pada HStack secara otomatis memberikan spacing antar child elements berdasarkan token spacing yang telah dikonfigurasi. Project Style/StyleSheet menggunakan View dan TouchableOpacity native dengan StyleSheet terpisah yang mendefinisikan header, iconsView, dan icons styles, di mana spacing harus diatur manual menggunakan properti seperti padding dengan nilai numerik eksplisit.`

Implementasi screen `List` dan `Article` juga menunjukkan perbedaan paradigma styling yang kontras. Pada project `gluestack`, komponen-komponen layout seperti `ScrollView`, `VStack`, `Heading`, dan `Divider` berasal dari `GlueStack UI` dengan props styling bawaan yang mendukung responsive design tokens. Modal dialog pada screen `List` menggunakan nested `Box components` dengan props seperti `shadowColor`, `shadowOffset`, `shadowOpacity`, dan `elevation` yang lebih readable. Sebaliknya, project `Style/StyleSheet` menggunakan komponen `React Native` native seperti `ScrollView`, `View`, `Text`, dan `Modal` dengan `StyleSheet` yang mendefinisikan multiple style objects seperti `modalOverlay`, `modalContent`, `modalImage`, dan `modalTitle`, yang memerlukan lebih banyak kode boilerplate untuk mencapai hasil yang sama.

Aspek sistem desain dan maintainability juga membedakan kedua pendekatan. `GlueStack UI` pada project `gluestack` menerapkan design system yang terstruktur dengan centralized theme configuration, memungkinkan perubahan global pada warna, spacing, atau typography hanya dengan memodifikasi file `gluestack-ui.config.js`. Penggunaan token seperti `$primary500`, `$coolGray900`, atau `$md` memastikan konsistensi visual dan memudahkan kolaborasi tim dalam proyek berskala besar. Project `Style/StyleSheet` dengan `StyleSheet` native memiliki styling yang tersebar di setiap file komponen, sehingga perubahan tema global memerlukan modifikasi di multiple files, yang berpotensi menimbulkan inconsistency dan meningkatkan maintenance overhead.

Menurut saya, kedua pendekatan memiliki trade-off yang berbeda. `StyleSheet` native pada project `Style/StyleSheet` lebih mudah dipahami oleh developer pemula karena tidak memerlukan pengetahuan tentang library eksternal dan konsepnya mirip dengan `CSS` tradisional. Namun, seiring bertambahnya kompleksitas aplikasi, penulisan kode menjadi repetitif dan kurang scalable. `GlueStack UI` pada project `gluestack` memerlukan pembelajaran awal tentang design tokens, component API, dan configuration system, namun memberikan productivity boost yang signifikan untuk proyek jangka panjang dengan fitur-fitur seperti theme switching, responsive variants, dan accessible components yang sudah terintegrasi.

Secara performa runtime, kedua pendekatan memiliki karakteristik yang berbeda. `StyleSheet` native menghasilkan bundle size yang lebih kecil karena tidak ada overhead dari library UI tambahan, dan styling object di-optimize secara internal oleh `React Native`. `GlueStack UI` menambahkan bundle size karena dependency tambahan, namun library ini telah dioptimasi dengan tree-shaking dan code-splitting, serta menyediakan fitur-fitur advanced seperti animation driver dan style resolution yang efisien. Untuk aplikasi sederhana, perbedaan performa mungkin tidak signifikan, namun untuk aplikasi enterprise dengan kompleksitas tinggi, `GlueStack UI` dapat memberikan development velocity yang lebih baik.

Kesimpulannya, project gluestack dengan GlueStack UI mewakili pendekatan modern yang mengadopsi component-based styling dengan design system yang robust, ideal untuk proyek yang membutuhkan scalability, consistency, dan maintainability tinggi. Project Style/StyleSheet dengan StyleSheet native mewakili pendekatan fundamental yang lebih lightweight, cocok untuk proyek sederhana, prototyping cepat, atau developer yang menginginkan full control atas styling tanpa abstraksi library. Pemilihan antara kedua pendekatan bergantung pada kebutuhan spesifik proyek, ukuran tim, timeline development, dan requirement untuk design consistency serta future extensibility.

## **22 Link Repository GitHub:**

[BlazerKers354/PRAKTIKUM-PENGEMBANGAN-APLIKASI-BERGERAK-MODUL-4-Design-in-React-Native](#)