

PRAKTIKUM
PENGEMBANGAN APLIKASI BERGERAK

MODUL 7

Case Based Learning 1

(ToDo List App)

Disusun oleh:
ZULFA AZKA FARISADILAH
1204220109



Universitas
Telkom

PROGRAM STUDI SARJANA SISTEM INFORMASI
DIREKTORAT KAMPUS SURABAYA
UNIVERSITAS TELKOM
SURABAYA

2025

DAFTAR ISI

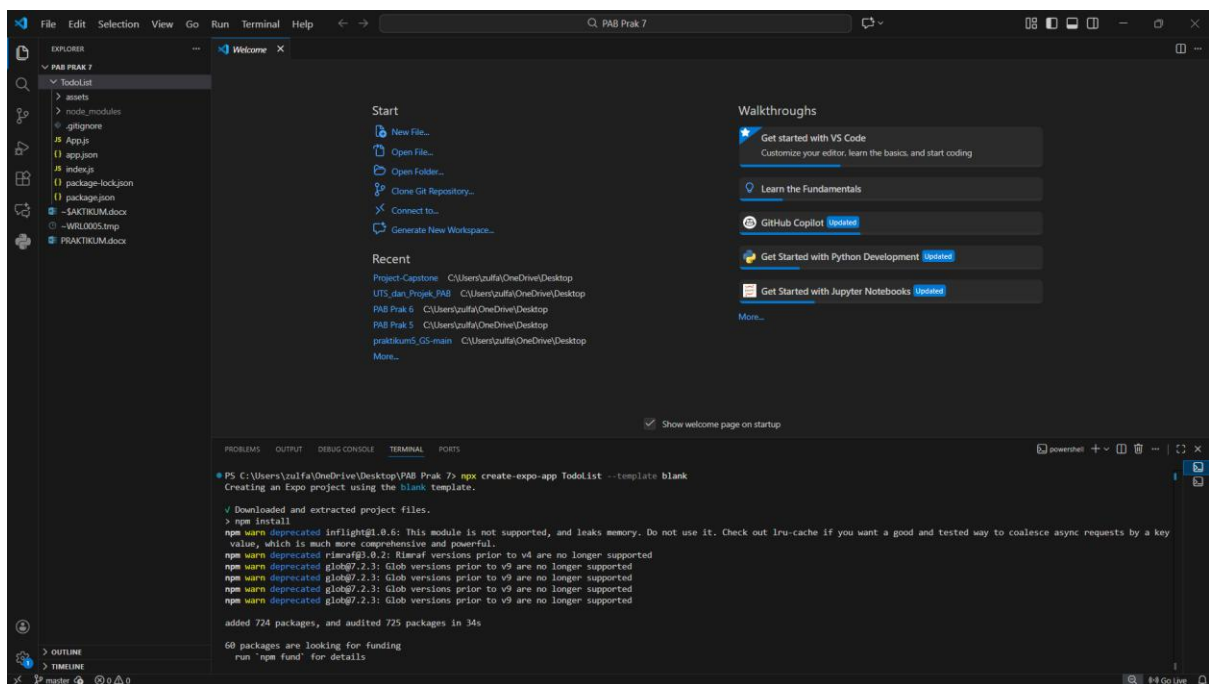
1	Membuat Project React Native (Expo)	3
2	Instalasi dan Setup	3
3	Persiapan Struktur Project	4
4	Membuat Splash Screen	4
5	Komponen Task.....	4
6	Komponen Index.....	6
7	Komponen About	6
8	Komponen Task All.....	11
9	Komponen Task Completed	14
10	Komponen Index.....	16
11	Komponen App.js.....	17
12	Output	20
13	Link Repository GitHub:	20

Project Gluestack dan Expo Router

1 Membuat Project React Native (Expo)

- Buat sebuah project React Native via Expo bernama TodoList dengan menjalankan command:
npx create-expo-app TodoList --template blank
- Masuk ke folder project TodoList yang sudah terbuat dengan menjalankan command:
cd TodoList

Output:



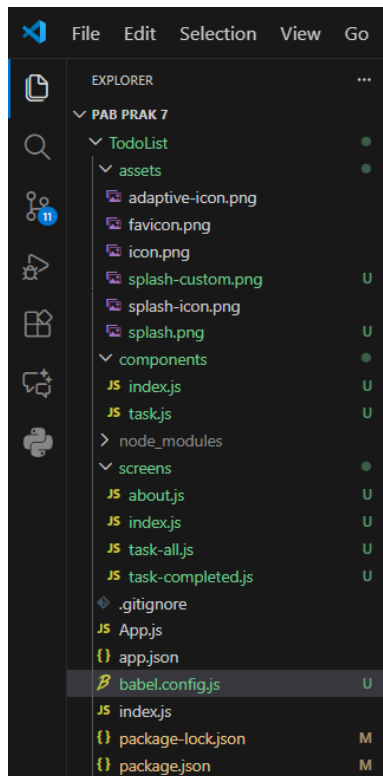
2 Instalasi dan Setup

- Install package-package yang diperlukan:
 - **npm install @react-navigation/native@6.1.9 --legacy-peer-deps**
 - **npm install @react-navigation/native-stack@6.9.17 --legacy-peer-deps**
 - **npm install react-native-screens react-native-safe-area-context**
 - **npm install @react-navigation/bottom-tabs@6.5.11 --legacy-peer-deps**
 - **npm install @react-native-async-storage/async-storage@^2.2.0 --legacy-peer-deps**
 - **npm install @gluestack-ui/themed-native-base react-native-svg@13.4.0 --legacy-peer-deps**
 - **npm install --save-dev babel-preset-expo --legacy-peer-deps**
 - **npm install native-base --legacy-peer-deps**
 - **npm install react-dom@19.1.0 --legacy-peer-deps**
- setelah install packagenya selanjutnya terapkan babel.config.js:

```
module.exports = function (api) {
  api.cache(true);
  return {
    presets: ['babel-preset-expo'],
  };
};
```

3 Persiapan Struktur Project

Sesuaikan struktur project yang ada seperti di bawah ini dengan cara menambahkan folder-folder dan file-file yang dibutuhkan. Download file assets pada link ini [assets - Google Drive](#):



4 Membuat Splash Screen

- Buka file app.json lalu terapkan kode JSON seperti di bawah ini untuk property splash:

```
"splash": {
  "image": "./assets/splash-custom.png",
  "resizeMode": "contain",
  "backgroundColor": "#ffffff"
},
```

Penjelasan:

Bagian kode “*splash*” tersebut merupakan konfigurasi untuk menampilkan layar pembuka (*splash screen*) ketika aplikasi pertama kali dijalankan. Properti **image** menentukan file gambar yang digunakan sebagai tampilan utama pada splash screen, yaitu *splash-custom.png* yang berada di folder *assets*. Selanjutnya, **resizeMode: "contain"** mengatur agar gambar tersebut

menyesuaikan ukuran layar tanpa terpotong, sehingga seluruh gambar tetap terlihat secara proporsional. Sementara itu, **backgroundColor: "#ffffff"** menentukan warna latar belakang splash screen, yaitu putih.

5 Komponen Task

- Terapkan pada file /components/task.js kode program React Native di bawah ini:

```
import { Text, HStack, IconButton, Icon, Box, Checkbox, Pressable } from "native-base";
import { FontAwesome5 } from "@expo/vector-icons";

const TaskList = (props) => {
  const { data, onChecked, onDeleted, deletedIcon, onItemPress } =
    props;
  return (
    <Pressable onPress={onItemPress}>
      <Box
        px={3}
        py={4}
        bg={data.isCompleted ? "primary.500" : "#fff"}
        my="7.5px"
        borderRadius="5"
      >
        <HStack w="100%" justifyContent="space-between"
          alignItems="center">
          <Checkbox
            isChecked={data.isCompleted}
            onChange={onChecked}
            accessibilityLabel="This is a dummy checkbox"
            value={data.title}
            aria-label="This is a dummy checkbox"
          />
          <Text
            width="100%"
            fontSize={16}
            flexShrink={1}
            textAlign="left"
            mx="10px"
            strikeThrough={data.isCompleted}
          >
            {data.title}
          </Text>
          {deletedIcon && (
            <IconButton
              size="sm"
              colorScheme="trueGray"
              icon={
                <Icon
                  as={FontAwesome5}

```

```

        name="trash"
        size="sm"
        color={data.isCompleted ? "#fff" : "red.700"}
      />
    }
    onPress={onDeleted}
  />
)}
</HStack>
</Box>
</Pressable>
);
};

export default TaskList;

```

Penjelasan:

Komponen **TaskList** ini berfungsi untuk menampilkan satu item tugas dalam sebuah daftar menggunakan *NativeBase* sebagai pustaka UI. Komponen menerima beberapa properti, seperti **data** yang berisi informasi tugas, **onChecked** untuk menangani perubahan status tugas (selesai atau belum), **onDeleted** untuk menghapus tugas, **deletedIcon** untuk menentukan apakah ikon hapus ditampilkan, dan **onItemPress** untuk menangani ketika seluruh item ditekan. Tampilan utama dibungkus oleh komponen **Pressable**, sehingga item dapat merespons sentuhan. Di dalamnya terdapat **Box** yang mengatur ruang, warna latar, dan bentuk sudut, di mana warna latar otomatis berubah jika tugas sudah ditandai selesai. Elemen-elemen dalam baris disusun menggunakan **HStack**, termasuk **Checkbox** untuk menandai tugas, **Text** untuk menampilkan judul tugas (dengan efek coret jika sudah selesai), serta **IconButton** berisi ikon *trash* yang muncul jika fitur hapus diaktifkan.

6 Komponen Index

- Terapkan pada file `/components/index.js` kode program React Native di bawah ini:

```

import TaskList from "../task";
export { TaskList };

```

Penjelasan:

Kode `import TaskList from "../task";` digunakan untuk mengambil komponen **TaskList** dari file lain bernama *task.js* (atau *task/index.js*) yang berada pada folder yang sama. Setelah komponen tersebut diimpor, perintah `export { TaskList };` berfungsi untuk mengekspor kembali komponen tersebut sebagai bagian dari modul ini. Cara ini biasanya digunakan dalam struktur folder yang rapi, di mana sebuah file *index.js* mengumpulkan dan mengekspor beberapa komponen sehingga file lain dapat mengimpor komponen tersebut dengan lebih mudah.

7 Komponen About

- Terapkan pada file `/screens/about.js` kode program React Native di bawah ini:

```

import React, { useState } from "react";
import {
  Box,
  Button,
  Divider,
  Heading,
  HStack,
  ScrollView,
  Text,
  VStack,
  Center,
} from "native-base";
import { Modal, TouchableOpacity, StyleSheet, View } from "react-native";
import AsyncStorage from
  "@react-native-async-storage/async-storage";

const AboutScreen = () => {
  const [isClearDataOpen, setIsClearDataOpen] = useState(false);

  const handleClearData = async () => {
    try {
      await AsyncStorage.clear();
      setIsClearDataOpen(false);
    } catch (e) {
      console.log("Error clearing data in about.js");
      console.error(e);
    }
  };

  const AlertClearData = () => {
    return (
      <Modal
        transparent={true}
        visible={isClearDataOpen}
        onRequestClose={() => setIsClearDataOpen(false)}
        animationType="fade"
      >
        <View style={styles.modalOverlay}>
          <View style={styles.modalContent}>
            <Text style={styles.modalTitle}>Clear
              Data</Text>
            <Text style={styles.modalBody}>
              This action will delete all todo list
              data. Data that has been
              deleted cannot be restored!
            </Text>
            <View style={styles.modalFooter}>
              <TouchableOpacity

```

```

        style={[styles.button,
        styles.cancelButton]}
        onPress={() =>
            setIsClearDataOpen(false)}
    >
        <Text
            style={styles.cancelText}>Cancel</Text>
    </TouchableOpacity>
    <TouchableOpacity
        style={[styles.button,
        styles.deleteButton]}
        onPress={handleClearData}
    >
        <Text
            style={styles.deleteText}>Delete</Text>
    </TouchableOpacity>
    </View>
    </View>
    </View>
    </Modal>
    );
};
return (
    <Box flex={1}>
        <AlertClearData />
        <ScrollView px={3} py={5}>
            <Box>
                <Heading>App Description</Heading>
                <Text mt={3}>
                    This is a simple todo list application to
                    use for learning
                    AsyncStorage implementation in react
                    native.
                </Text>
            </Box>
            <Divider mt="20px" mb="15px" />
            <Box>
                <Heading mb="10px">App Info</Heading>
                <VStack>
                    <HStack justifyContent={"space-between"}>
                        <Text
                            fontWeight={"bold"}>Version</Text>
                        <Text>v0.0.1</Text>
                    </HStack>
                    <HStack justifyContent={"space-between"}>
                        <Text fontWeight={"bold"}>Update
                            on</Text>
                        <Text>December 11, 2022</Text>
                    </HStack>
                </VStack>
            </Box>
        </ScrollView>
    </Box>
);

```



```

        </HStack>
        <HStack justifyContent={"space-between"}>
          <Text fontWeight={"bold"}>Created
            By</Text>
          <Text>Muhamad Arsyad</Text>
        </HStack>
      </VStack>
    </Box>
    <Divider mt="20px" mb="15px" />
    <Box>
      <Button
        colorScheme="danger"
        onPress={() => setIsClearDataOpen(true)}
        mb={5}
      >
        Clear Data
      </Button>
    </Box>
  </ScrollView>
</Box>
);
};

```

```

const styles = StyleSheet.create({
  modalOverlay: {
    flex: 1,
    backgroundColor: 'rgba(0, 0, 0, 0.5)',
    justifyContent: 'center',
    alignItems: 'center',
  },
  modalContent: {
    backgroundColor: 'white',
    borderRadius: 8,
    padding: 20,
    width: '85%',
    maxWidth: 400,
  },
  modalTitle: {
    fontSize: 18,
    fontWeight: 'bold',
    marginBottom: 12,
    color: '#1f2937',
  },
  modalBody: {
    fontSize: 14,
    color: '#4b5563',
    marginBottom: 20,
    lineHeight: 20,
  },
});

```

```

    },
    modalFooter: {
      flexDirection: 'row',
      justifyContent: 'flex-end',
      gap: 10,
    },
    button: {
      paddingVertical: 8,
      paddingHorizontal: 16,
      borderRadius: 4,
      minWidth: 70,
      alignItems: 'center',
    },
    cancelButton: {
      backgroundColor: '#f3f4f6',
    },
    deleteButton: {
      backgroundColor: '#dc2626',
    },
    cancelText: {
      color: '#6b7280',
      fontWeight: '500',
    },
    deleteText: {
      color: 'fff',
      fontWeight: '500',
    },
  });
export default AboutScreen;

```

Penjelasan:

Kode **AboutScreen** ini merupakan sebuah komponen React Native yang menampilkan halaman informasi aplikasi sekaligus menyediakan fitur untuk menghapus seluruh data *Todo List* yang tersimpan di perangkat. Komponen ini menggunakan *state* `isClearDataOpen` untuk mengatur apakah modal konfirmasi penghapusan data sedang ditampilkan atau tidak. Fungsi `handleClearData` memanfaatkan **AsyncStorage**, yaitu penyimpanan lokal di React Native, untuk menghapus seluruh data aplikasi secara permanen, kemudian menutup modal setelah proses selesai. Elemen modal didefinisikan dalam komponen `AlertClearData`, yang menampilkan pesan peringatan, tombol pembatalan, dan tombol penghapusan data. Pada bagian tampilan utama, halaman ini menyajikan deskripsi aplikasi, informasi versi, tanggal pembaruan, dan pembuat aplikasi, semuanya disusun menggunakan komponen UI dari `NativeBase` seperti `Box`, `Heading`, `HStack`, dan `VStack`. Selain itu, tersedia tombol **Clear Data** yang ketika ditekan akan membuka modal konfirmasi. Seluruh tampilan modal menggunakan *StyleSheet* untuk mengatur overlay, tata letak konten, serta visual dari tombol-tombolnya.

8 Komponen Task All

- Terapkan pada file /screens/task-all.js kode program React Native di bawah ini:

```
import React, { useState, useEffect } from "react";
import {
  Box,
  HStack,
  IconButton,
  Icon,
  Center,
  Toast,
  ScrollView,
  Spinner,
} from "native-base";
import { TextInput, StyleSheet } from "react-native";
import { Feather } from "@expo/vector-icons";
import AsyncStorage from "@react-native-async-storage/async-storage";
import { TaskList } from "../components";

const TaskScreen = () => {
  const [list, setList] = useState([]);
  const [inputValue, setInputValue] = useState("");
  const [isLoading, setIsLoading] = useState(true);
  const toastID = "toast-add-task";
  const handleAddTask = (data) => {
    if (data === "") {
      if (!Toast.isActive(toastID)) {
        Toast.show({
          id: toastID,
          title: "Masukan nama task",
        });
      }
    }
    return;
  }

  setList((prevList) => [...prevList, { title: data, isCompleted: false }]);
  setInputValue("");

  try {
    AsyncStorage.setItem(
      "@task-list",
      JSON.stringify([...list, { title: data, isCompleted: false }])
    );
  } catch (e) {
    console.log("Error add task: in task-all.js");
    console.error(e.message);
  }
};
```

```

const handleDeleteTask = (index) => {
  const deletedList = list.filter((_, listIndex) => listIndex !== index);
  setList(deletedList);

  try {
    AsyncStorage.setItem("@task-list", JSON.stringify(deletedList));
  } catch (e) {
    console.log("Error delete task: in task-all.js");
    console.error(e.message);
  }
};

const handleStatusChange = (index) => {
  setList((prevList) => {
    const newList = [...prevList];
    newList[index].isCompleted = !newList[index].isCompleted;

    try {
      AsyncStorage.setItem("@task-list", JSON.stringify(newList));
    } catch (e) {
      console.log("Error update status task: in task-all.js");
      console.error(e.message);
    }

    return newList;
  });
};

const getTaskList = async () => {
  try {
    const value = await AsyncStorage.getItem("@task-list");
    if (value !== null) {
      console.log(value);
      setList(JSON.parse(value));
    } else {
      console.log("No Tasks");
    }
  } catch (e) {
    console.log("Error get task: in task-all.js");
    console.error(e);
  } finally {
    setIsLoading(false);
  }
};

useEffect(() => {
  getTaskList();
}, []);

```

```

return (
  <Box flex={1}>
    <Box mt="15px" mx="15px" mb="7.5px">
      <HStack space="15px">
        <TextInput
          style={styles.input}
          onChangeText={(char) => setInputValue(char)}
          value={inputValue}
          placeholder="Add Task"
          placeholderTextColor="#9ca3af"
        />
        <IconButton
          flex={1}
          borderRadius="sm"
          variant="solid"
          icon={
            <Icon as={Feather} name="plus" size="lg" color="warmGray.50" />
          }
          onPress={() => {
            handleAddTask(inputValue);
          }}
        />
      </HStack>
    </Box>
    {isLoading ? (
      <Center flex={1}>
        <Spinner size="lg" />
      </Center>
    ) : (
      <ScrollView>
        <Box mb="15px" mx="15px">
          {list.map((item, index) => (
            <Box key={item.title + index.toString()}>
              <TaskList
                data={item}
                index={index}
                deletedIcon={true}
                onItemPress={() => handleStatusChange(index)}
                onChecked={() => handleStatusChange(index)}
                onDelete={() => handleDeleteTask(index)}
              />
            </Box>
          ))}
        </Box>
      </ScrollView>
    )}
  </Box>

```

```

    );
  });

const styles = StyleSheet.create({
  input: {
    flex: 6,
    height: 48,
    borderWidth: 1,
    borderColor: "#0891b2",
    borderRadius: 4,
    paddingHorizontal: 15,
    fontSize: 16,
    backgroundColor: "#fff",
  },
});

export default TaskScreen;

```

Penjelasan:

Komponen **TaskScreen** ini berfungsi sebagai halaman utama untuk mengelola daftar tugas (todo list) pada aplikasi. Komponen ini menggunakan beberapa *state* untuk menyimpan data, yaitu list sebagai daftar tugas, *inputValue* sebagai teks input pengguna, dan *isLoading* untuk menandai apakah proses pengambilan data sedang berlangsung. Data tugas disimpan secara permanen menggunakan **AsyncStorage**, sehingga daftar tugas tidak hilang meskipun aplikasi ditutup. Fungsi *handleAddTask* digunakan untuk menambahkan tugas baru, dengan validasi agar pengguna tidak memasukkan teks kosong. Fungsi *handleDeleteTask* menghapus tugas berdasarkan indeks, sedangkan *handleStatusChange* mengubah status tugas menjadi selesai atau belum selesai. Semua perubahan ini langsung diperbarui ke **AsyncStorage** agar tetap sinkron. Pada saat komponen pertama kali dijalankan, fungsi *getTaskList* dipanggil melalui *useEffect* untuk mengambil data yang sebelumnya sudah tersimpan.

9 Komponen Task Completed

- Terapkan pada file `/screens/task-completed.js` kode program React Native di bawah ini:

```

import React, { useState, useEffect } from "react";
import { Center, Text, Box, ScrollView, Icon, Spinner } from "native-base";
import { AntDesign } from "@expo/vector-icons";
import AsyncStorage from "@react-native-async-storage/async-storage";
import { TaskList } from "../components";

const TaskCompletedScreen = () => {
  const [completedListLength, setCompletedListLength] =
    useState(0);
  const [allList, setAllList] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const handleStatusChange = (index) => {

```

```

const newList = [...allList];
newList[index].isCompleted = !newList[index].isCompleted;
setAllList(newList);
try {
  AsyncStorage.setItem("@task-list", JSON.stringify(newList));
} catch (e) {
  console.log("Error update status task: in task-completed.js");
  console.error(e.message);
} finally {
  setCompletedListLength(newList.filter((item) =>
    item.isCompleted).length);
}
};

```

```

const getTaskList = async () => {
  try {
    const value = await AsyncStorage.getItem("@task-list");
    if (value !== null) {
      const allData = JSON.parse(value);
      const completedData = allData.filter((item) =>
        item.isCompleted).length;
      setAllList(allData);
      setCompletedListLength(completedData);
    } else {
      console.log("No tasks");
    }
  } catch (e) {
    console.log("Error get task: in task-completed.js");
    console.error(e);
  } finally {
    setIsLoading(false);
  }
};

```

```

useEffect(() => {
  getTaskList();
}, []);

```

```

return (
  <Box mx={3} mt={3} flex={1}>
    {isLoading ? (
      <Center flex={1}>
        <Spinner size="lg" />
      </Center>
    ) : completedListLength === 0 ? (
      <Center flex={1}>
        <Icon as={AntDesign} name="frown" size={82}
          color="primary.600" mb={2} />

```

```

        <Text fontSize={16} bold={true}>
          No completed listings yet
        </Text>
        <Text fontSize={16}>Hurry up your list!</Text>
      </Center>
    ): (
      <ScrollView>
        {allList.map((item, index) => {
          if (item.isCompleted) {
            return (
              <Box key={item.title + index.toString()}>
                <TaskList
                  data={item}
                  onChecked={() => handleStatusChange(index)}
                  onItemPress={() => handleStatusChange(index)}
                />
              </Box>
            );
          }
          return null;
        })}
      </ScrollView>
    )}
  </Box>
);
};

export default TaskCompletedScreen;

```

Penjelasan:

Komponen **TaskCompletedScreen** digunakan untuk menampilkan daftar tugas yang sudah diselesaikan oleh pengguna. Komponen ini menggunakan tiga state utama, yaitu `allList` untuk menyimpan seluruh data tugas, `completedListLength` untuk menghitung jumlah tugas yang sudah selesai, dan `isLoading` untuk menandai proses pengambilan data. Data tugas diambil dari **AsyncStorage** melalui fungsi `getTaskList`, lalu dipisahkan antara seluruh daftar dan jumlah yang sudah selesai. Fungsi `handleStatusChange` digunakan untuk mengubah status sebuah tugas ketika pengguna menekan itemnya, kemudian memperbarui penyimpanan dan menghitung ulang jumlah tugas selesai. Pada bagian tampilan, jika data masih dimuat maka aplikasi menampilkan spinner; jika tidak ada tugas selesai maka tampil pesan kosong; dan jika ada tugas selesai, setiap item ditampilkan menggunakan komponen `TaskList`.

10 Komponen Index

- Terapkan pada file `/screens/index.js` kode program React Native di bawah ini:

```

import AboutScreen from "../about";
import TaskScreen from "../task-all";
import TaskCompletedScreen from "../task-completed";

```



```
export { AboutScreen, TaskCompletedScreen, TaskScreen };
```

Penjelasan:

Kode tersebut berfungsi sebagai *index module* yang mengumpulkan beberapa komponen layar (**AboutScreen**, **TaskScreen**, dan **TaskCompletedScreen**) dari file terpisah di dalam folder yang sama. Setiap layar di-*import* dari file masing-masing, lalu diekspor kembali dalam satu objek menggunakan bentuk **named export**. Pendekatan seperti ini umum digunakan dalam struktur proyek React Native untuk menjaga kerapian dan memudahkan proses impor pada bagian lain dari aplikasi. Dengan adanya file pengumpul seperti ini, modul lain cukup mengimpor satu pintu masuk (`import { TaskScreen } from "../screens"`) tanpa perlu mengetahui lokasi masing-masing file secara spesifik.

11 Komponen App.js

- Terapkan pada file App.js kode program React Native di bawah ini:

```
import { NativeBaseProvider, Icon, Text } from "native-base";
import { NavigationContainer } from "@react-navigation/native";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import { FontAwesome5, MaterialIcons } from "@expo/vector-icons";
import { SafeAreaProvider, useSafeAreaInsets } from "react-native-safe-area-context";
import { AboutScreen, TaskCompletedScreen, TaskScreen } from "../screens";
const Tab = createBottomTabNavigator();

const AppContent = () => {
  const insets = useSafeAreaInsets();

  return (
    <NativeBaseProvider>
      <NavigationContainer>
        <Tab.Navigator
          screenOptions={({ route }) => ({
            tabBarIcon: ({ focused, color, size }) => {
              let iconName;
              let IconComponent = FontAwesome5;

              if (route.name === "All Task") {
                iconName = "tasks";
                IconComponent = FontAwesome5;
              } else if (route.name === "Completed") {
                iconName = "check-circle";
                IconComponent = MaterialIcons;
              } else if (route.name === "About") {
                iconName = "info-circle";
                IconComponent = FontAwesome5;
              }
            }
          })
        >
          <TaskScreen />
          <TaskCompletedScreen />
          <AboutScreen />
        </Tab.Navigator>
      </NavigationContainer>
    </NativeBaseProvider>
  )
}
```

```

        <Icon
          as={IconComponent}
          name={iconName}
          size={6}
          color={focused ? "#0891b2" : color}
        />
      );
    },
    tabBarIconStyle: { marginTop: 5 },
    tabBarLabel: ({ children, color, focused }) => {
      return (
        <Text color={focused ? "#0891b2" : color} mb={1}
          fontSize={9} fontWeight={focused ? "600" : "400"}
          numberOfLines={1}>
          {children}
        </Text>
      );
    },
    tabBarStyle: {
      height: 58 + insets.bottom,
      paddingBottom: Math.max(insets.bottom, 8),
      borderTopWidth: 0,
      elevation: 8,
      shadowColor: "#000",
      shadowOffset: { width: 0, height: -2 },
      shadowOpacity: 0.1,
      shadowRadius: 3,
      paddingTop: 5,
    },
    tabBarActiveTintColor: "#0891b2",
    tabBarInactiveTintColor: "#6b7280",
  ))}
<
<Tab.Screen
  name="All Task"
  component={TaskScreen}
  options={{
    headerStyle: {
      backgroundColor: "#0891b2",
    },
    headerTintColor: "#fff",
    headerTitleStyle: {
      fontWeight: "bold",
    },
  }}
/>
<Tab.Screen
  name="Completed"

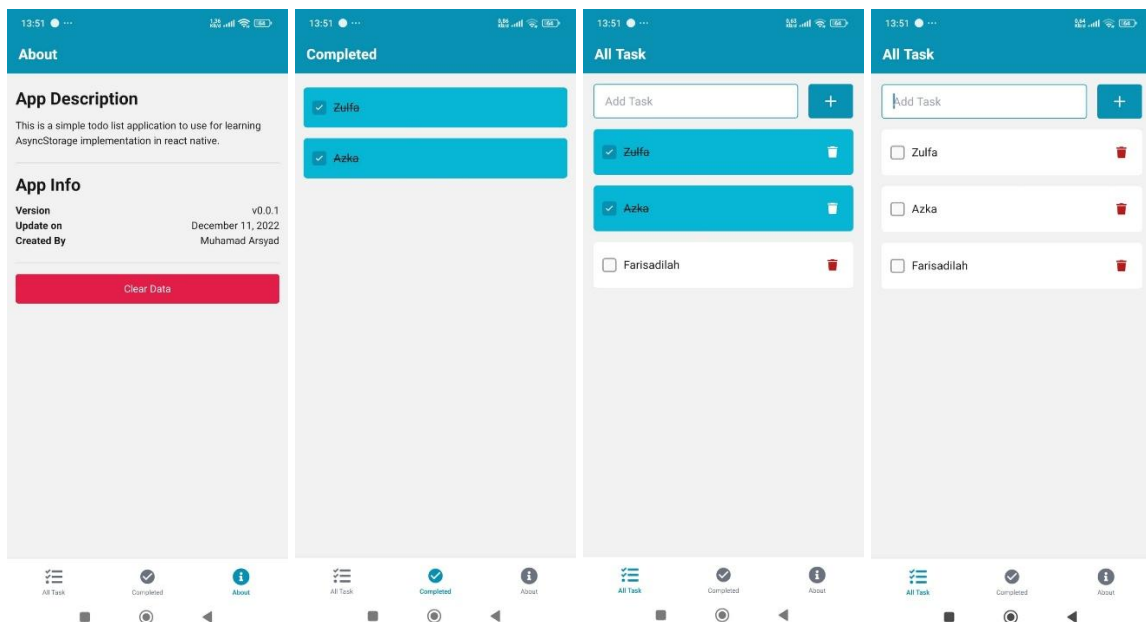
```

```
        component={TaskCompletedScreen}
        options={{
          headerStyle: {
            backgroundColor: "#0891b2",
          },
          headerTintColor: "#fff",
          headerTitleStyle: {
            fontWeight: "bold",
          },
        }}
      />
    <Tab.Screen
      name="About"
      component={AboutScreen}
      options={{
        headerStyle: {
          backgroundColor: "#0891b2",
        },
        headerTintColor: "#fff",
        headerTitleStyle: {
          fontWeight: "bold",
        },
      }}
    />
  </Tab.Navigator>
</NavigationContainer>
</NativeBaseProvider>
);
};
const App = () => {
  return (
    <SafeAreaProvider>
      <AppContent />
    </SafeAreaProvider>
  );
};
export default App;
```

Penjelasan:

Kode tersebut membentuk struktur utama aplikasi React Native yang menggunakan navigasi berbasis *Bottom Tab* melalui library **React Navigation**. Aplikasi dibungkus oleh **SafeAreaProvider**, yang memastikan tampilan menyesuaikan area aman perangkat, seperti notch dan gesture bar. Di dalamnya, komponen **AppContent** mengatur penyedia NativeBase dan container navigasi. Komponen ini mendefinisikan *tab navigator* dengan tiga layar: **TaskScreen** untuk seluruh tugas, **TaskCompletedScreen** untuk tugas selesai, dan **AboutScreen** untuk informasi aplikasi. Setiap tab diberi ikon yang ditentukan secara dinamis berdasarkan nama rute, menggunakan ikon dari FontAwesome5 atau MaterialIcons. Selain itu, terdapat kustomisasi tampilan tab seperti tinggi, padding, warna aktif dan tidak aktif, serta gaya label. Header untuk setiap layar juga dikustomisasi agar konsisten, dengan warna latar biru dan teks putih tebal.

12 Output



13 Link Repository GitHub:

[BlazerKers354/PRAKTIKUM-PENGEMBANGAN-APLIKASI-BERGERAK-MODUL-7-Case-Based-Learning-1-ToDo-List-App-](#)