

Navigation

Navigation

❖ Navigation 관련 라이브러리

✓ react-navigation

- 리액트 네이티브 커뮤니티에서 관리하며 사용률이 가장 높은 라이브러리
- 리액트 공식 매뉴얼에서도 이 라이브러리로 화면을 전환하는 방법을 소개
- 이 라이브러리는 네비게이션 기능이 자바스크립트로 구현되어 있음

✓ react-native-navigation

- 홈페이지 제작 서비스 Wix에서 관리
- 이 라이브러리는 이미 만들어진 네이티브 앱에 리액트 네이티브를 적용하는 경우 사용하기에 더 적합하며 네비게이션 기능이 자바스크립트가 아닌 각 플랫폼의 네이티브 코드로 구현되어 있기 때문에 react-navigation보다 더욱 네이티브 스러운 사용 경험을 제공

Navigation

❖ 설치 및 적용

✓ 프로젝트 생성

- npx react-native init ReactNativeNavigation

✓ 설치

- cd ReactNativeNavigation
- yarn android
- yarn ios
- yarn add @react-navigation/native
- yarn add react-native-screens react-native-safe-area-context
- cd ios
- pod install

Navigation

❖ 설치 및 적용

✓ 라이브러리 적용

- 리액트 네이티브 프로젝트에 라이브러리를 적용하기 위해서는 @react-navigation/native에서 Navigationcontainer 컴포넌트를 불러와 사용해 앱 전체를 감싸주어야 함

✓ App.js 파일을 수정

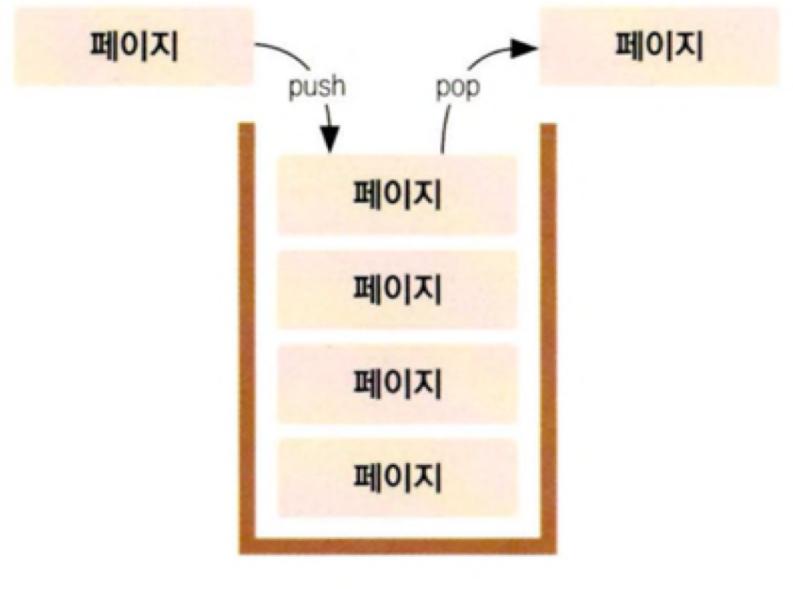
```
import React from 'react';
import {NavigationContainer} from '@react-navigation/native';
```

```
function App() {
  return (
    <NavigationContainer>
      </NavigationContainer>
    );
}

export default App;
```

기본적인 사용법

- ❖ 리액트 네이티브 앱에서는 화면을 전환할 때 브라우저의 History와 비슷한 사용성을 제공하기 위해 네이티브 스택 네비게이터(Native Stack Navigator)를 사용



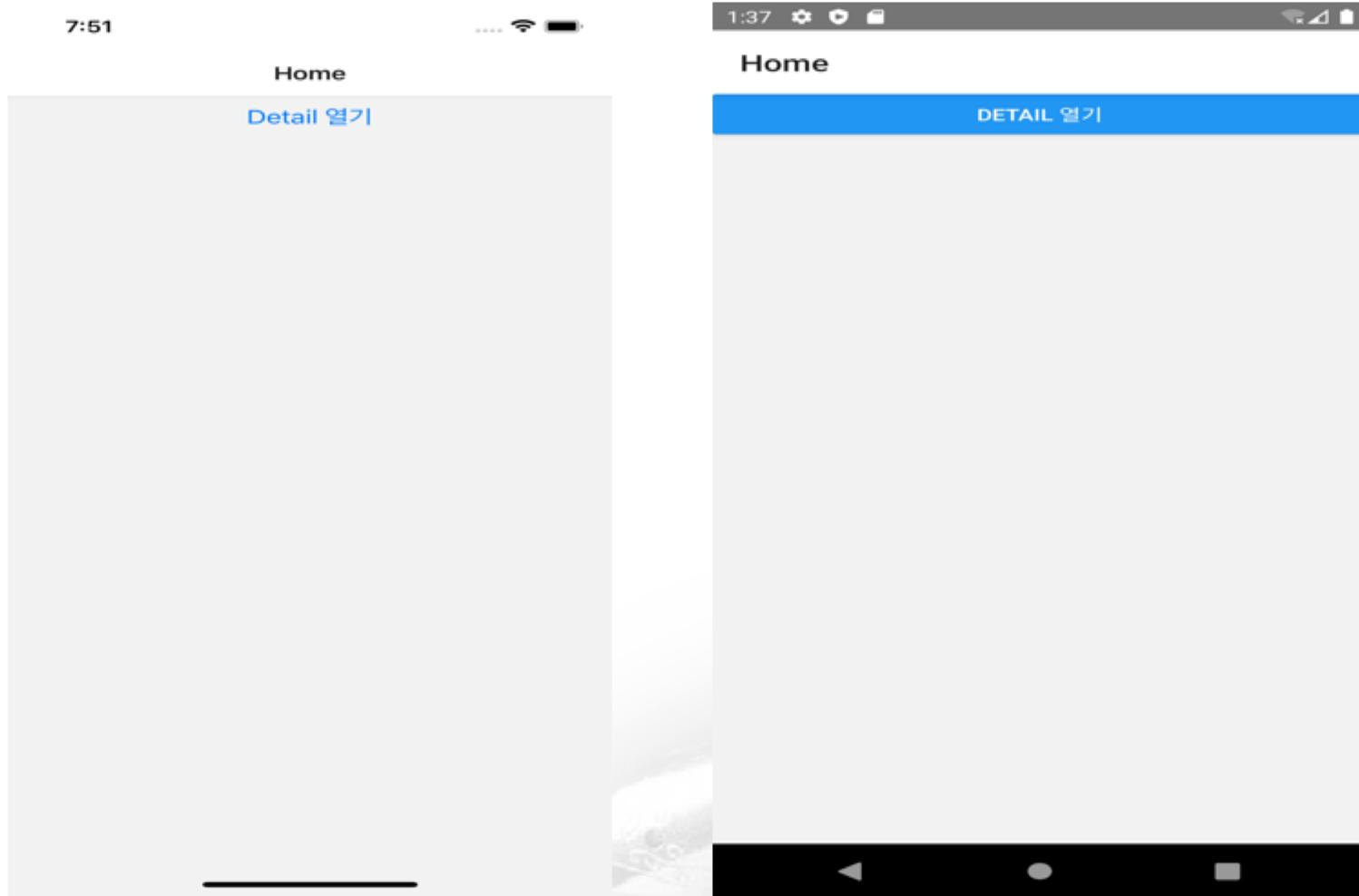
기본적인 사용법

❖ 네이티브 스택 네비게이터

- ✓ 리액트 네비게이션 라이브러리에는 다른 상황에 사용할 수 있는 다양한 네비게이터가 있음
- ✓ 네이티브 스택 네비게이터(Native Stack Navigator)는 가장 흔히 사용되며 안드로이드에서는 Fragment를 iOS에서는 UINavigationController를 사용해 일반 네이티브 앱과 정확히 동일한 방식으로 화면을 관리
- ✓ 네이티브 스택 네비게이터를 사용하기 위해서는 라이브러리를 추가적으로 설치해주어야 함
 - yarn add @react-navigation/native-stack
 - cd ios
 - pod install
- ✓ 도큐먼트: <https://reactnavigation.org/docs/native-stack-navigator/>

기본적인 사용법

❖ 화면 전환



기본적인 사용법

❖ 화면 전환

- ✓ screens 디렉토리를 생성하고 시작 화면으로 사용할 HomeScreen.js 파일을 생성하고 작성

```
import React from 'react';
import {View, Button} from 'react-native';

function HomeScreen({navigation}) {
  return (
    <View>
      <Button title="Detail 열기" />
    </View>
  );
}

export default HomeScreen;
```

기본적인 사용법

❖ 화면 전환

- ✓ screens 디렉토리에 상세 화면으로 사용할 DetailScreen.js 파일을 생성하고 작성

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

function DetailScreen({route, navigation}) {
  return (
    <View>
      <View>
        <Text>Detail</Text>
      </View>
    </View>
  );
}

export default DetailScreen;
```

기본적인 사용법

❖ 화면 전환

- ✓ 네이티브 스택 네비게이터를 생성하기 위해서 App.js 수정

```
import React from 'react';
import {NavigationContainer} from '@react-navigation/native';
import {createNativeStackNavigator} from '@react-navigation/native-stack';
import HomeScreen from './screens/HomeScreen';
import DetailScreen from './screens/DetailScreen';

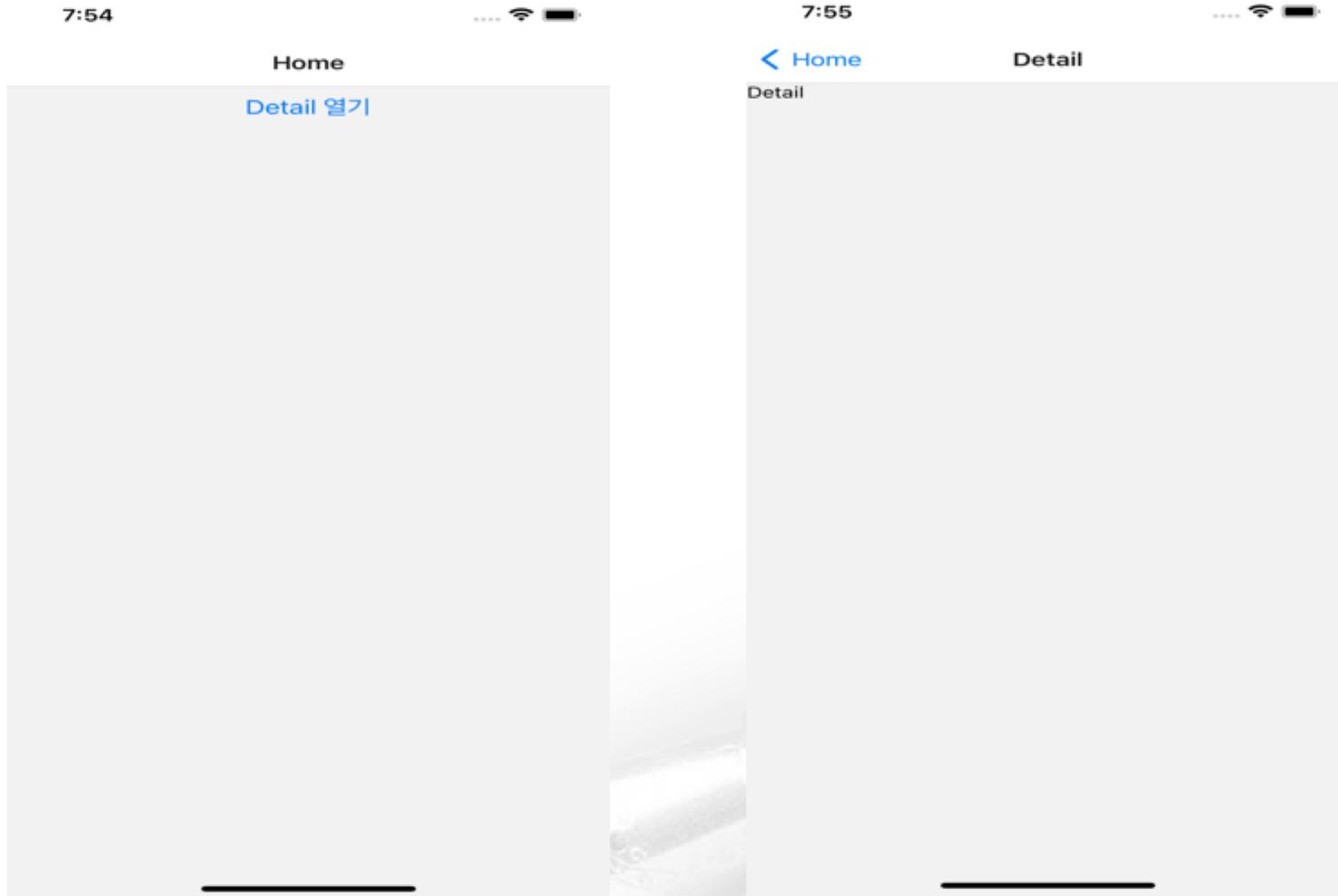
const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Detail" component={DetailScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default App;
```

기본적인 사용법

❖ 화면 전환



기본적인 사용법

❖ 화면 전환

- ✓ 상세보기 화면으로 이동하기 위한 이벤트 처리 코드를 HomeScreen.js 파일에 작성

```
import React from 'react';
import { View, Button } from 'react-native';

function HomeScreen({ navigation }) {
  return (
    <View>
      <Button title="Detail 열기" onPress={() => navigation.navigate("Detail")} />
    </View>
  );
}

export default HomeScreen;
```

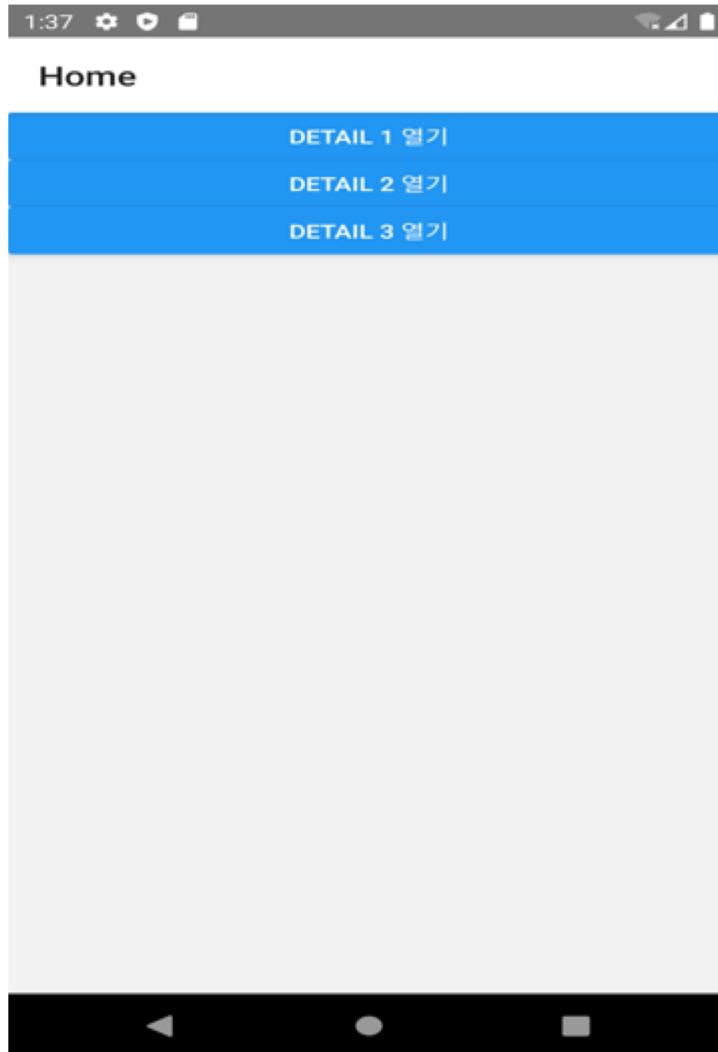
기본적인 사용법

❖ Route Parameter

- ✓ 새로운 화면을 보여줄 때 의존해야 하는 어떤 값이 있으면 Route Parameter를 이용
- ✓ Route Parameter는 객체 타입으로 설정
- ✓ Route Parameter를 전달하면서 화면을 전환할 때는 navigate 또는 push 함수의 두 번째 인자로 객체를 설정해주면 됨

기본적인 사용법

❖ Route Parameter



기본적인 사용법

❖ Route Parameter

- ✓ Route Parameter 전달을 위해서 HomeScreen.js 파일을 수정

```
import React from 'react';
import {View, Button} from 'react-native';

function HomeScreen({navigation}) {
  return (
    <View>
      <Button title="Detail 1 열기"
        onPress={() => navigation.navigate("Detail", {id:1})}/>
      <Button title="Detail 2 열기"
        onPress={() => navigation.navigate("Detail", {id:2})}/>
      <Button title="Detail 3 열기"
        onPress={() => navigation.navigate("Detail", {id:3})}/>
    </View>
  );
}

export default HomeScreen;
```

기본적인 사용법

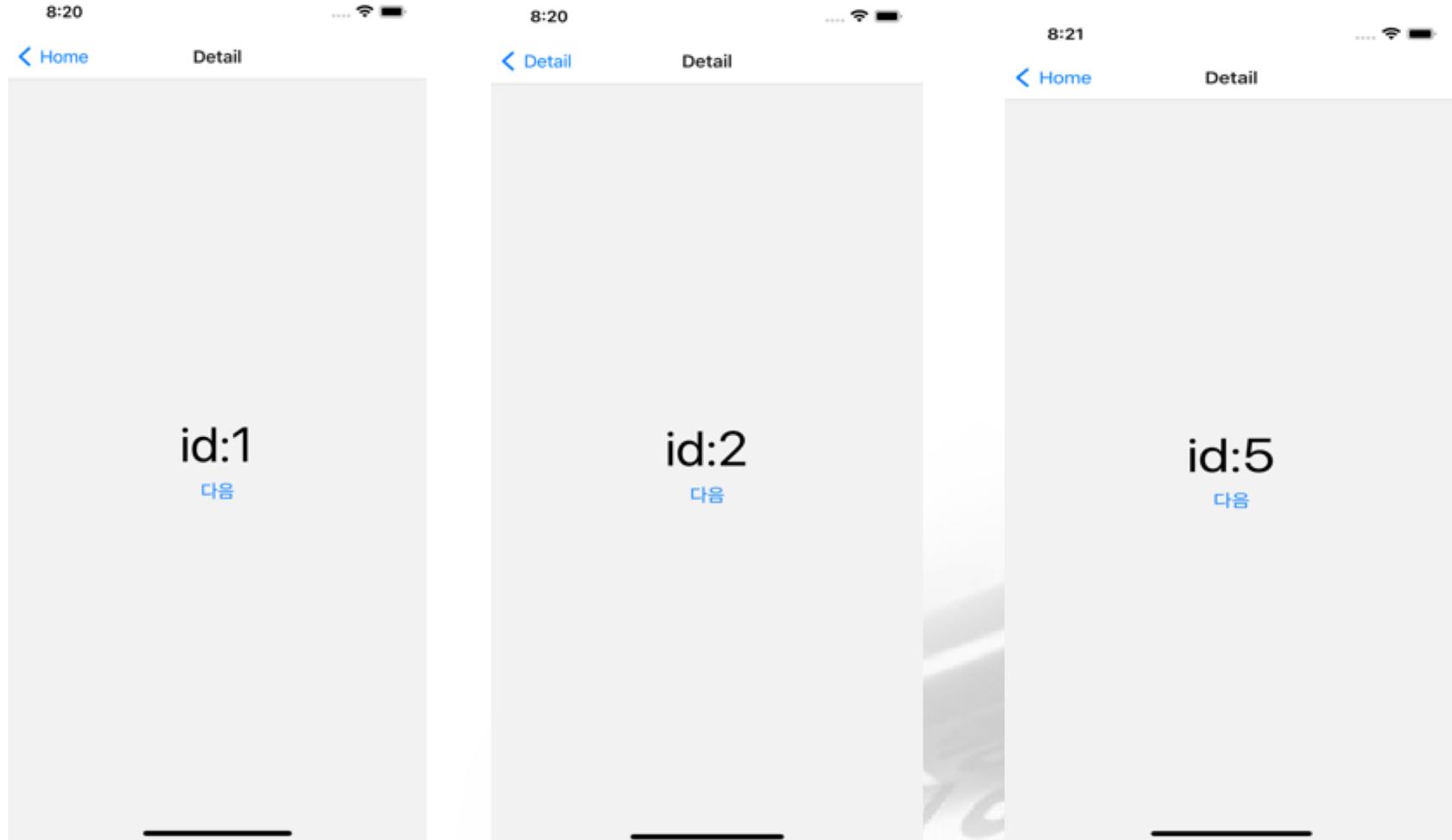
❖ Route Parameter

- ✓ Route Parameter를 사용하기 위해서 DetailScreen.js 파일을 수정

```
import React from 'react';
import {View, Text, StyleSheet} from 'react-native';
function DetailScreen({route, navigation}) {
  return (
    <View style={styles.block}>
      <Text style={styles.text}>id:{route.params.id}</Text>
    </View>
  );
}
const styles = StyleSheet.create({
  block: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center'
  },
  text: {
    fontSize: 48
  }
});
export default DetailScreen;
```

기본적인 사용법

❖ push



기본적인 사용법

❖ push

- ✓ 화면 전환을 위한 함수로 새로 이동할 화면이 현재 화면과 같더라도 스택에 추가
- ✓ navigate는 push와 달리 새로 이동할 화면이 현재 화면과 같으면 새로운 화면을 쌓지 않고 파라미터만 변경

기본적인 사용법

- ❖ push

- ✓ HomeScreen.js 파일을 수정

```
import React from 'react';
import {View, Button} from 'react-native';

function HomeScreen({navigation}) {
  return (
    <View>
      <Button title="Detail 1 열기"
        onPress={() => navigation.push("Detail", {id:1})}/>
      <Button title="Detail 2 열기"
        onPress={() => navigation.push("Detail", {id:2})}/>
      <Button title="Detail 3 열기"
        onPress={() => navigation.push("Detail", {id:3})}/>
    </View>
  );
}

export default HomeScreen;
```

기본적인 사용법

❖ push

- ✓ push 와 navigator를 차이를 알아보기 위해서 DetailScreen.js 파일을 수정

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

function DetailScreen({route, navigation}) {
  return (
    <View style={styles.block}>
      <Text style={styles.text}>id:{route.params.id}</Text>
      <Button title="다음" onPress={()=> navigation.push("Detail", {id:route.params.id + 1 })}>
    />
    </View>
  );
}
```

기본적인 사용법

❖ push

- ✓ push 와 navigator를 차이를 알아보기 위해서 DetailScreen.js 파일을 수정

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

function DetailScreen({ route, navigation }) {
  return (
    <View style={styles.block}>
      <Text style={styles.text}>id:{route.params.id}</Text>
      <Button title="다음" onPress={() => navigation.navigate("Detail", { id: route.params.id + 1 })} />
    </View>
  );
}
```

기본적인 사용법

- ❖ 뒤로 가기
 - ✓ 뒤로 가기를 할 때는 `navigation.pop()` 함수를 호출
 - ✓ 첫 번째 화면으로 이동하고 싶을 때는 `nvgiation.popToTop()` 함수를 호출

기본적인 사용법

❖ 뒤로 가기

9:01



Home

Detail 1 열기
Detail 2 열기
Detail 3 열기

9:02



< Detail

Detail

id:4

다음

뒤로가기

처음으로

9:02



Home

Detail 1 열기
Detail 2 열기
Detail 3 열기

기본적인 사용법

❖ 뒤로 가기

- ✓ DetailScreen.js 파일을 수정

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

function DetailScreen({ route, navigation }) {
  return (
    <View style={styles.block}>
      <Text style={styles.text}>id:{route.params.id}</Text>
      <View style={styles.buttons}>
        <Button title="다음" onPress={() => navigation.push("Detail", { id: route.params.id + 1 })} />
        <Button title="뒤로가기" onPress={() => navigation.pop()} />
        <Button title="처음으로" onPress={() => navigation.popToTop()} />
      </View>
    </View>
  );
}
```

기본적인 사용법

❖ 뒤로 가기

- ✓ DetailScreen.js 파일을 수정

```
const styles = StyleSheet.create({
  block: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center'
  },
  text: {
    fontSize: 48
  },
  buttons: {
    flexDirection: 'row'
  }
});

export default DetailScreen;
```

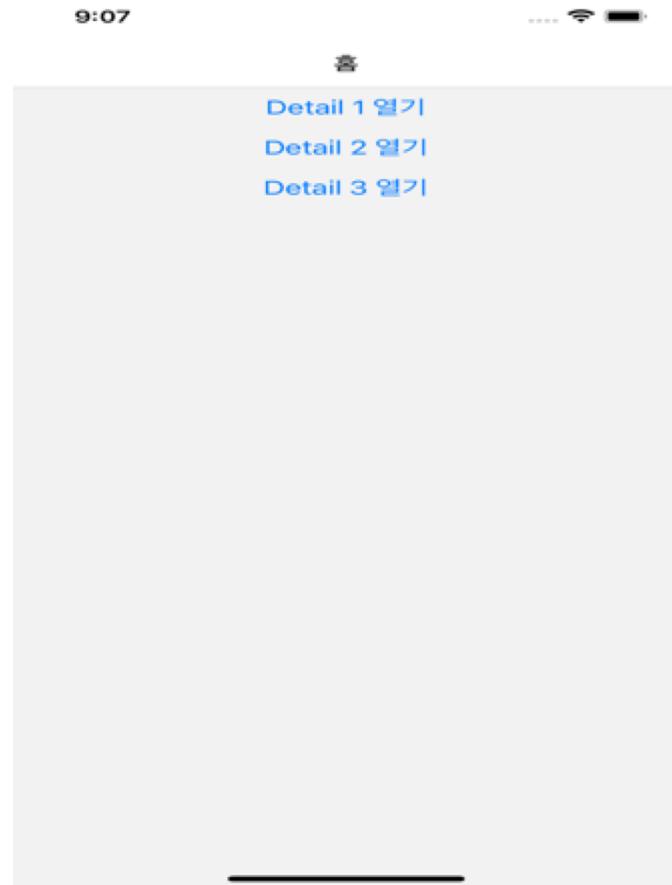
기본적인 사용법

❖ 헤더 커스터마이징

- ✓ react-navigation에서는 타이틀 영역을 헤더(Header)라고 부름
- ✓ 헤더를 커스터마이징 하는 방법
 - Stack.Screen의 Props로 설정
 - navigation.setOptions 함수를 사용하는 것

기본적인 사용법

- ❖ 타이틀 텍스트 변경



기본적인 사용법

- ❖ 타이틀 텍스트 변경

- ✓ App.js 파일을 수정

```
import React from 'react';
import {NavigationContainer} from '@react-navigation/native';
import {createNativeStackNavigator} from '@react-navigation/native-stack';
import HomeScreen from './screens/HomeScreen';
import DetailScreen from './screens/DetailScreen';

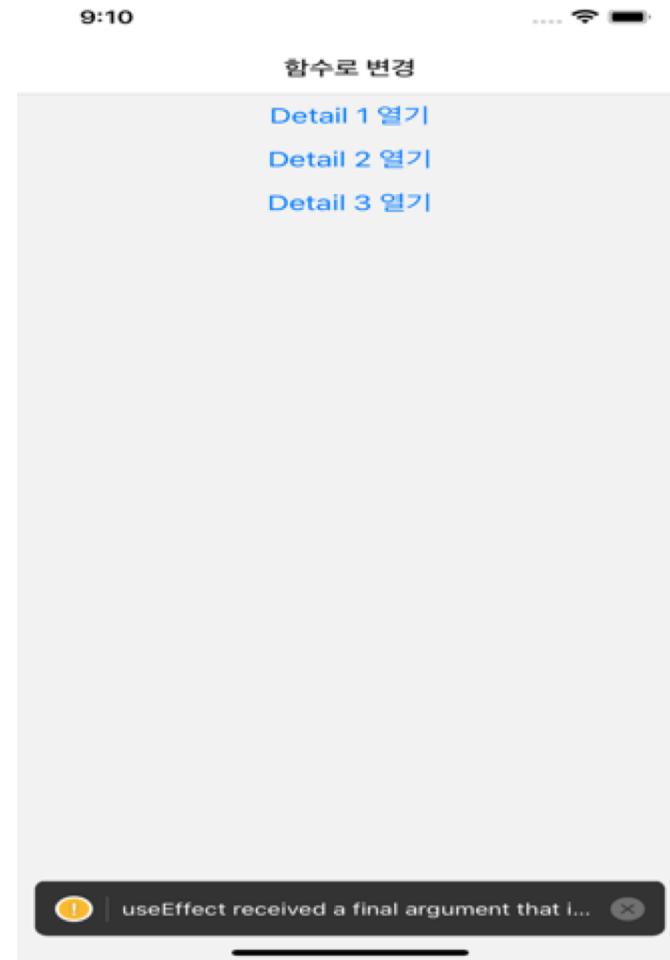
const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen} options={{title:'홈'}} />
        <Stack.Screen name="Detail" component={DetailScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default App;
```

기본적인 사용법

- ❖ 타이틀 텍스트 변경



기본적인 사용법

- ❖ 타이틀 텍스트 변경

- ✓ HomeScreen.js 파일을 수정

```
import React, { useEffect } from 'react';
import { View, Button } from 'react-native';

function HomeScreen({ navigation }) {
    //navigation의 값이 변경되었을 때 호출
    useEffect(() => {
        navigation.setOptions({ title: "함수로 변경" });
    }, [navigation])

    return (
        <View>
            <Button title="Detail 1 열기"
                onPress={() => navigation.push("Detail", { id: 1 })} />
            <Button title="Detail 2 열기"
                onPress={() => navigation.push("Detail", { id: 2 })} />
            <Button title="Detail 3 열기"
                onPress={() => navigation.push("Detail", { id: 3 })} />
        </View>
    );
}

export default HomeScreen;
```

기본적인 사용법

- ❖ 타이틀 텍스트 변경



기본적인 사용법

- ❖ 타이틀 텍스트 변경

- ✓ App.js 파일을 수정

```
import React from 'react';
import {NavigationContainer} from '@react-navigation/native';
import {createNativeStackNavigator} from '@react-navigation/native-stack';
import HomeScreen from './screens/HomeScreen';
import DetailScreen from './screens/DetailScreen';

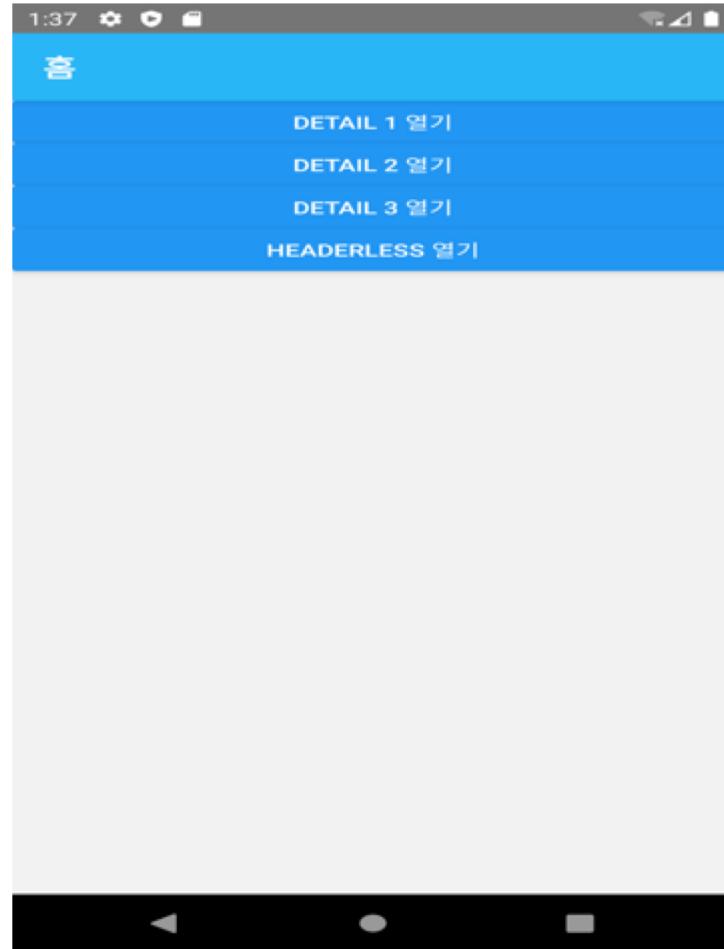
const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen} options={{title:'홈'}} />
        <Stack.Screen name="Detail" component={DetailScreen} options={({route}) => ({
          title: `상세 정보 - ${route.params.id}`
        })}/>
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default App;
```

기본적인 사용법

- ❖ 헤더 스타일 변경



기본적인 사용법

❖ 헤더 스타일 변경

- ✓ HomeScreen 의 헤더를 변경하기 위해서 App.js 파일을 수정

```
function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen}>
          options={{
            title: '홈',
            headerStyle: {
              //헤더 배경색
              backgroundColor: '#29b6f6' },
              //헤더의 텍스트 와 버튼 색상
              headerTintColor: '#ffffff',
              //타이틀 텍스트의 스타일
              headerTitleStyle: { fontWeight: 'bold', fontSize: 20 }
            } />
```

기본적인 사용법

❖ 헤더 스타일 변경



기본적인 사용법

❖ 헤더 스타일 변경

- ✓ DetailScreen의 헤더 스타일을 변경하기 위해서 App.js 파일을 수정

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/native-stack';

import HomeScreen from './screens/HomeScreen';
import DetailScreen from './screens/DetailScreen';

import {View, Text, TouchableOpacity} from 'react-native'

const Stack = createStackNavigator();
```

기본적인 사용법

❖ 헤더 스타일 변경

- ✓ DetailScreen의 헤더 스타일을 변경하기 위해서 App.js 파일을 수정

```
function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen}>
          options={{
            title: '홈',
            headerStyle: {
              //헤더 배경색
              backgroundColor: '#29b6f6' },
              //헤더의 텍스트 와 버튼 색상
              headerTintColor: '#ffffff',
              //타이틀 텍스트의 스타일
              headerTitleStyle: { fontWeight: 'bold', fontSize: 20 }
            }} />
```

기본적인 사용법

❖ 헤더 스타일 변경

- ✓ DetailScreen의 헤더 스타일을 변경하기 위해서 App.js 파일을 수정

```
<Stack.Screen
  name="Detail"
  component={DetailScreen}
  options={{
    headerBackVisible: false,
    headerLeft: ({ onPress }) => (
      <TouchableOpacity onPress={onPress}>
        <Text>Left</Text>
      </TouchableOpacity>
    ),
    headerTitle: ({ children }) => (
      <View>
        <Text>{children}</Text>
      </View>
    ),
    headerRight: () => (
      <View>
        <Text>Right</Text>
      </View>
    ),
  }}
/>
```

기본적인 사용법

- ❖ 헤더 스타일 변경

- ✓ DetailScreen의 헤더 스타일을 변경하기 위해서 App.js 파일을 수정

```
</Stack.Navigator>
</NavigationContainer>
);
}

export default App;
```

기본적인 사용법

- ❖ 헤더 스타일 변경
 - ✓ HomeScreen.js 파일을 수정

```
import React, { useEffect } from 'react';
import { View, Button } from 'react-native';
```

```
function HomeScreen({ navigation }) {
  useEffect(() => {
    navigation.setOptions({ title: '홈' });
  }, [navigation]);
```

기본적인 사용법

- ❖ 헤더 스타일 변경

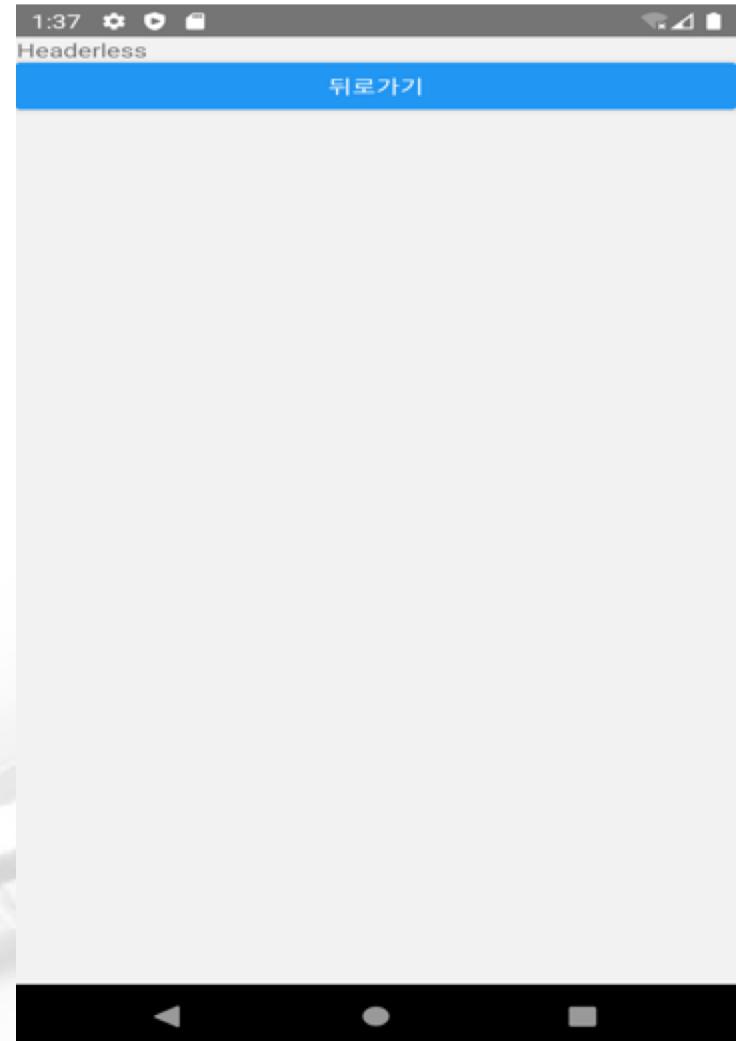
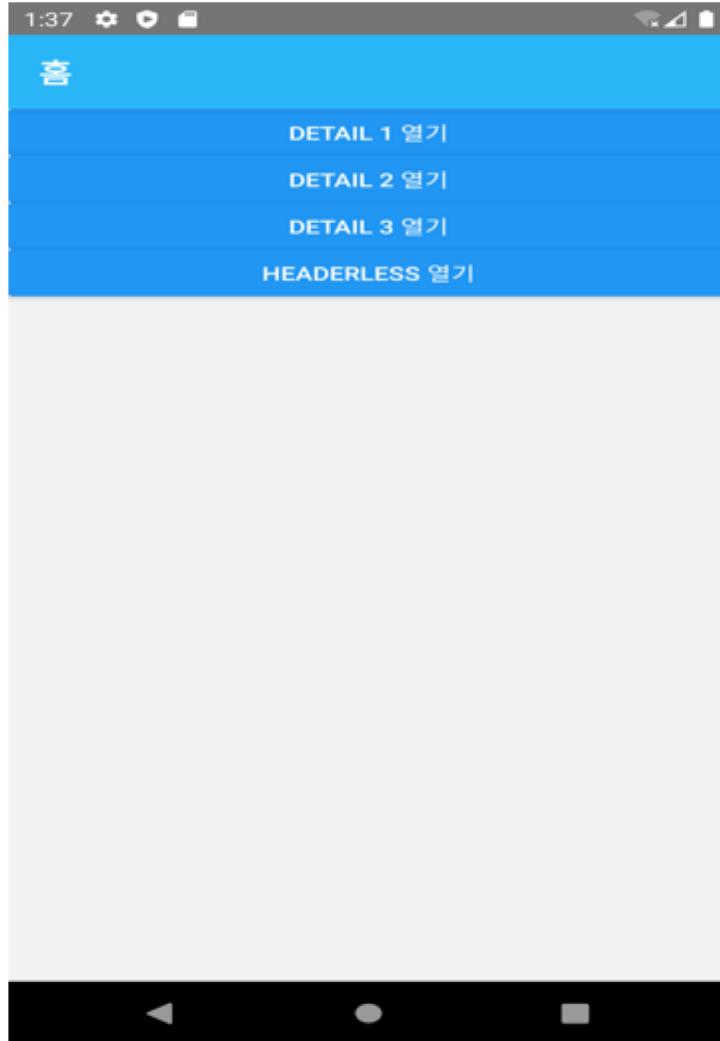
- ✓ HomeScreen.js 파일을 수정

```
return (
  <View>
    <Button
      title="Detail 1 열기"
      onPress={() => navigation.push('Detail', { id: 1 })}
    />
    <Button
      title="Detail 2 열기"
      onPress={() => navigation.push('Detail', { id: 2 })}
    />
    <Button
      title="Detail 3 열기"
      onPress={() => navigation.push('Detail', { id: 3 })}
    />
  </View>
);
}

export default HomeScreen;
```

기본적인 사용법

❖ 헤더 숨기기



기본적인 사용법

❖ 헤더 숨기기

- ✓ screens 디렉토리에 헤더가 없는 UI를 가진 HeaderlessScreen.js 파일을 생성하고 작성

```
import React from 'react';
import {View, Text, Button} from 'react-native';
import {SafeAreaView} from 'react-native-safe-area-context';

function HeaderlessScreen({navigation}) {
  return (
    <SafeAreaView>
      <View>
        <Text>Headerless</Text>
        <Button onPress={() => navigation.pop()} title="뒤로가기" />
      </View>
    </SafeAreaView>
  );
}

export default HeaderlessScreen;
```

기본적인 사용법

- ❖ 헤더 숨기기
 - ✓ App.js 파일을 수정

```
import React from 'react';
import {NavigationContainer} from '@react-navigation/native';
import {createNativeStackNavigator} from '@react-navigation/native-stack';
import HomeScreen from './screens/HomeScreen';
import DetailScreen from './screens/DetailScreen';

import {View, Text, TouchableOpacity} from 'react-native';

import HeaderlessScreen from './screens/HeaderlessScreen';

const Stack = createNativeStackNavigator();
```

기본적인 사용법

- ❖ 헤더 숨기기
 - ✓ App.js 파일을 수정

```
const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen}
          options={{title:'홈', headerStyle:{backgroundColor:'#29b6f6'},
          headerTintColor:'#ffffff',
          headerTitleStyle:{fontWeight:'bold', fontSize:20}}} />
```

기본적인 사용법

- ❖ 헤더 숨기기
 - ✓ App.js 파일을 수정

```
<Stack.Screen
  name="Detail"
  component={DetailScreen}
  options={{
    headerBackVisible: false,
    headerLeft: ({ onPress }) => (
      <TouchableOpacity onPress={onPress}>
        <Text>Left</Text>
      </TouchableOpacity>
    ),
    headerTitle:({ children }) => (
      <View>
        <Text>{children}</Text>
      </View>
    ),
    headerRight: () => (
      <View>
        <Text>Right</Text>
      </View>
    ),
  }}
/>
```

기본적인 사용법

- ❖ 헤더 숨기기
 - ✓ App.js 파일을 수정

```
<Stack.Screen  
  name="Headerless"  
  component={HeaderlessScreen}  
  options={{  
    headerShown: false,  
  }}  
/>  
  
</Stack.Navigator>  
</NavigationContainer>  
);  
}  
  
export default App;
```

기본적인 사용법

❖ 헤더 숨기기

- ✓ HomeScreen.js 파일을 수정

```
import React, { useEffect } from 'react';
import { View, Button } from 'react-native';
```

```
function HomeScreen({ navigation }) {
  useEffect(() => {
    navigation.setOptions({ title: '홈' });
  }, [navigation]);
```

기본적인 사용법

❖ 헤더 숨기기

- ✓ HomeScreen.js 파일을 수정

```
return (
  <View>
    <Button
      title="Detail 1 열기"
      onPress={() => navigation.push('Detail', { id: 1 })}>
    />
    <Button
      title="Detail 2 열기"
      onPress={() => navigation.push('Detail', { id: 2 })}>
    />
    <Button
      title="Detail 3 열기"
      onPress={() => navigation.push('Detail', { id: 3 })}>
    />
    <Button
      title="Headerless 열기"
      onPress={() => navigation.push('Headerless')} />
  </View>
);
}

export default HomeScreen;
```

기본적인 사용법

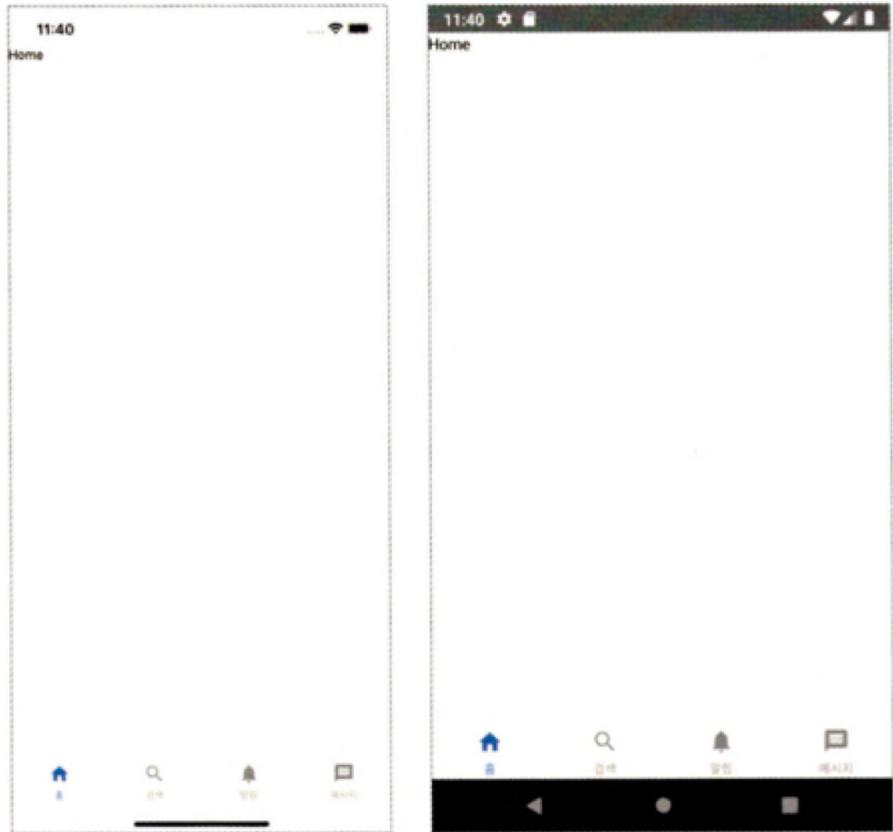
- ❖ 헤더 숨기기

- ✓ 전체 영역에서 헤더 숨기기

```
<Stack.Navigator  
initialRouteName="Home"  
screenOptions={{  
    headerShown: false,  
}}>
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터
- ✓ 하단에 탭을 보여주는 네비게이터



다양한 네비게이터

❖ 하단 탭 네비게이터

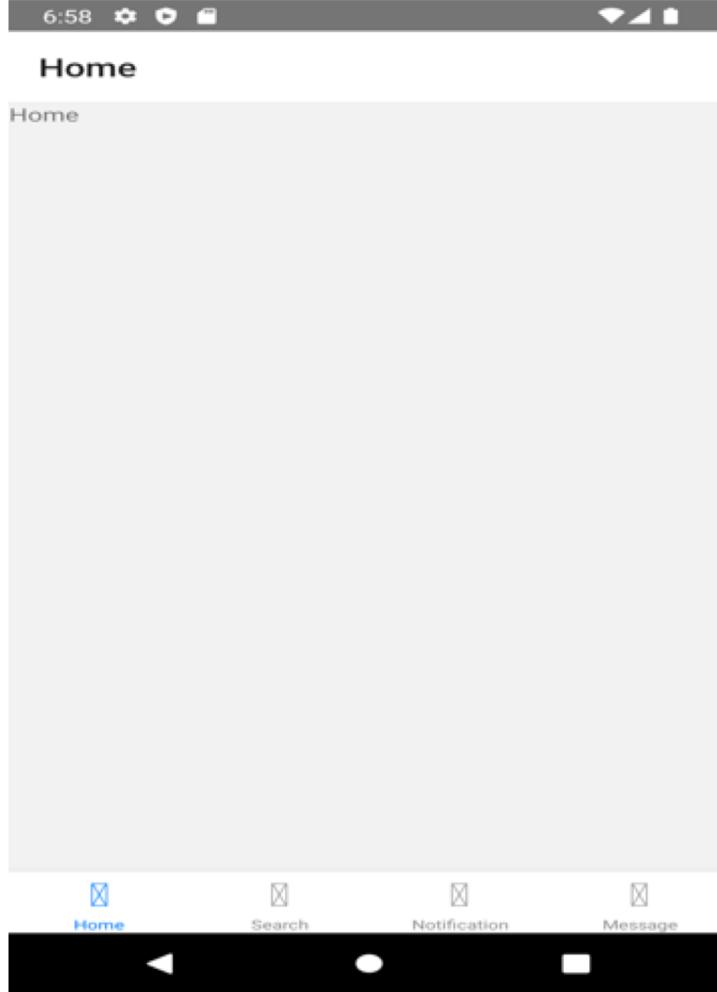
✓ 도큐먼트:

<https://reactnavigation.org/docs/bottom%25e2%2580%2594tab%25e2%2580%2594navigator설치>

- yarn add @react-navigation/bottom-tabs react-native-vector-icons
- cd ios
- pod install
- cd ..
- yarn android
- yarn ios

다양한 네비게이터

- ❖ 하단 탭 네비게이터



다양한 네비게이터

- ❖ 하단 탭 네비게이터
 - ✓ 텍스트만 존재하는 하단 탭을 위해서 App.js 수정

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Text } from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';

const Tab = createBottomTabNavigator();
```

다양한 네비게이터

❖ 하단 탭 네비게이터

- ✓ 텍스트만 존재하는 하단 탭을 위해서 App.js 수정

```
//탭으로 사용할 컴포넌트 생성
```

```
function HomeScreen() {  
  return <Text>Home</Text>;  
}
```

```
function SearchScreen() {  
  return <Text>Search</Text>;  
}
```

```
function NotificationScreen() {  
  return <Text>Notification</Text>;  
}
```

```
function MessageScreen() {  
  return <Text>Message</Text>;  
}
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터
 - ✓ 텍스트만 존재하는 하단 탭을 위해서 App.js 수정

```
//탭 생성
function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator
        initialRouteName="Home">
        <Tab.Screen
          name="Home"
          component={HomeScreen}>
        />
        <Tab.Screen
          name="Search"
          component={SearchScreen}>
        />
        <Tab.Screen
          name="Notification"
          component={NotificationScreen}>
        />
    </Tab.Navigator>
  )
}
```

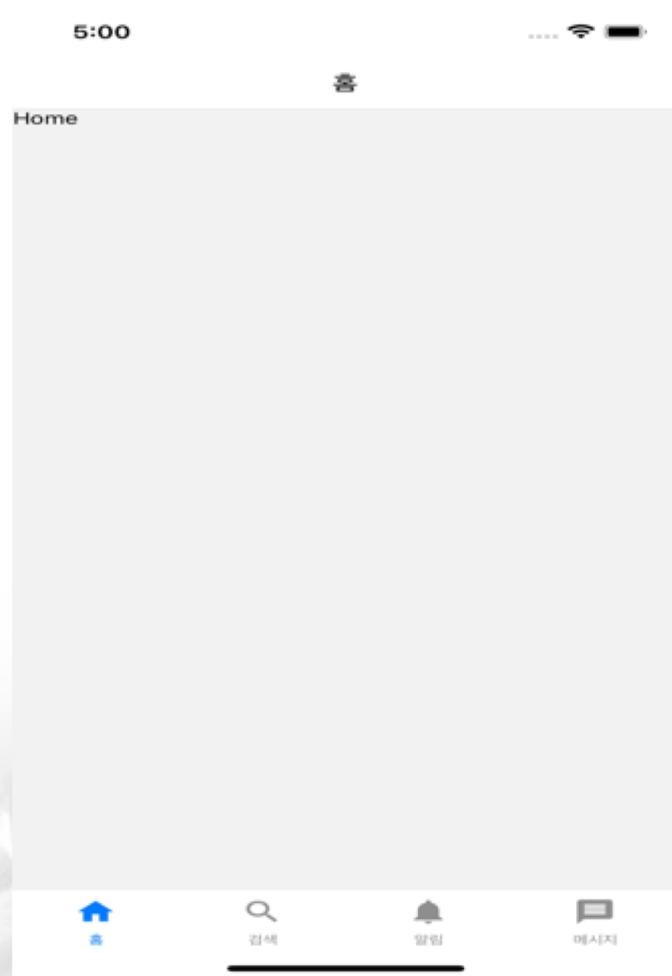
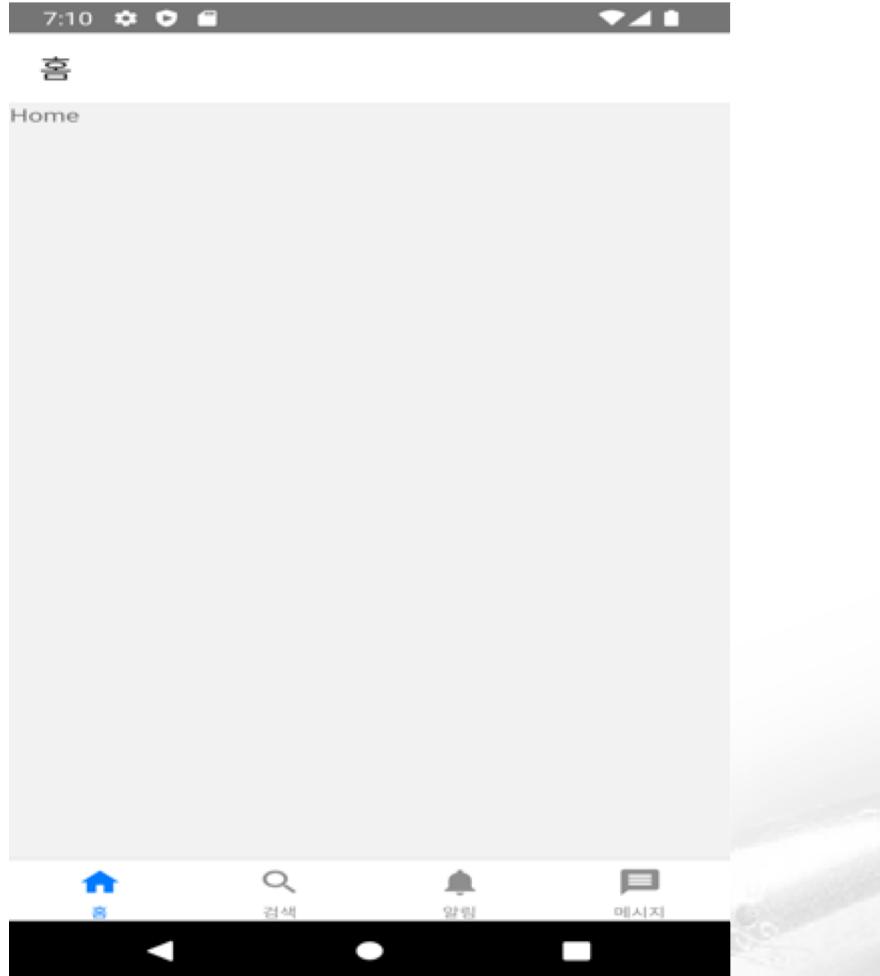
다양한 네비게이터

- ❖ 하단 탭 네비게이터
 - ✓ 텍스트만 존재하는 하단 탭을 위해서 App.js 수정

```
<Tab.Screen  
  name="Message"  
  component={MessageScreen}  
  
  />  
</Tab.Navigator>  
</NavigationContainer>  
);  
}  
  
export default App;
```

다양한 네비게이터

❖ 하단 탭 네비게이터



다양한 네비게이터

❖ 하단 탭 네비게이터

- ✓ icon 사용을 위해서 ios/앱이름/info.plist 파일에 아이콘 사용을 위한 설정을 추가

```
<key>UIAppFonts</key>
<array>
    <string>MaterialIcons.ttf</string>
</array>
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터

- ✓ icon 사용을 위해서 android/app/build.gradle 파일에 아이콘 사용을 위한 설정을 추가

```
project.ext.vectoricons = [  
    iconFontNames: [ 'MaterialIcons.ttf' ]  
]
```

```
apply from: "../../node_modules/react-native-vector-icons/fonts.gradle"
```

다양한 네비게이터

❖ 하단 탭 네비게이터

- ✓ icon 사용을 위해서 App.js 파일 수정

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Text } from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';
```

```
const Tab = createBottomTabNavigator();
```

```
function HomeScreen() {
  return <Text>Home</Text>;
}
```

```
function SearchScreen() {
  return <Text>Search</Text>;
}
```

```
function NotificationScreen() {
  return <Text>Notification</Text>;
}
```

```
function MessageScreen() {
  return <Text>Message</Text>;
}
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터

- ✓ icon 사용을 위해서 App.js 파일 수정

```
function App() {  
  return (  
    <NavigationContainer>  
      <Tab.Navigator  
        initialRouteName="Home">  
        <Tab.Screen  
          name="Home"  
          component={HomeScreen}  
          options={({  
            title: '홈',  
            tabBarIcon: ({ color, size }) => (  
              <Icon name="home" color={color} size={size} />  
            ),  
          })  
      />
```

다양한 네비게이터

❖ 하단 탭 네비게이터

- ✓ icon 사용을 위해서 App.js 파일 수정

```
<Tab.Screen
  name="Search"
  component={SearchScreen}
  options={{
    title: '검색',
    tabBarIcon: ({ color, size }) => (
      <Icon name="search" color={color} size={size} />
    ),
  }}
/>
<Tab.Screen
  name="Notification"
  component={NotificationScreen}
  options={{
    title: '알림',
    tabBarIcon: ({ color, size }) => (
      <Icon name="notifications" color={color} size={size} />
    ),
  }}
/>
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터

- ✓ icon 사용을 위해서 App.js 파일 수정

```
<Tab.Screen  
  name="Message"  
  component={MessageScreen}  
  options={({  
    title: '메시지',  
    tabBarIcon: ({ color, size }) => (  
      <Icon name="message" color={color} size={size} />  
    ),  
  })  
  />  
</Tab.Navigator>  
</NavigationContainer>  
);  
}  
  
export default App;
```

다양한 네비게이터

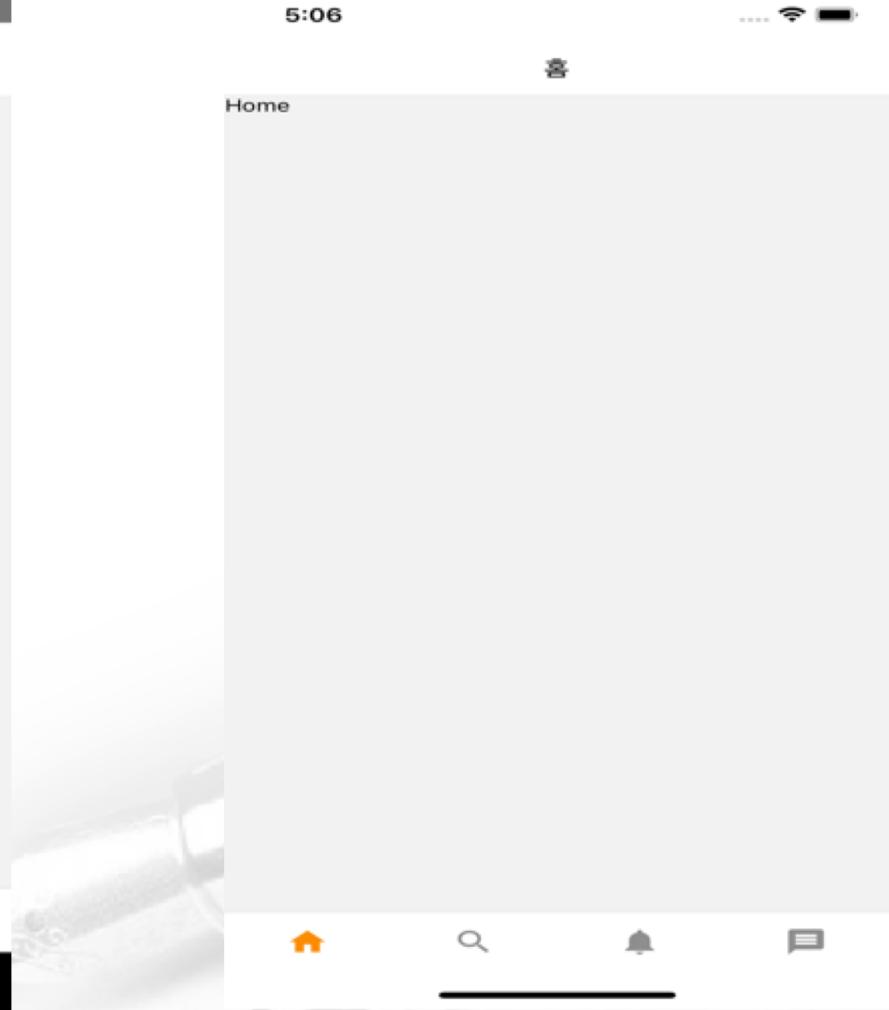
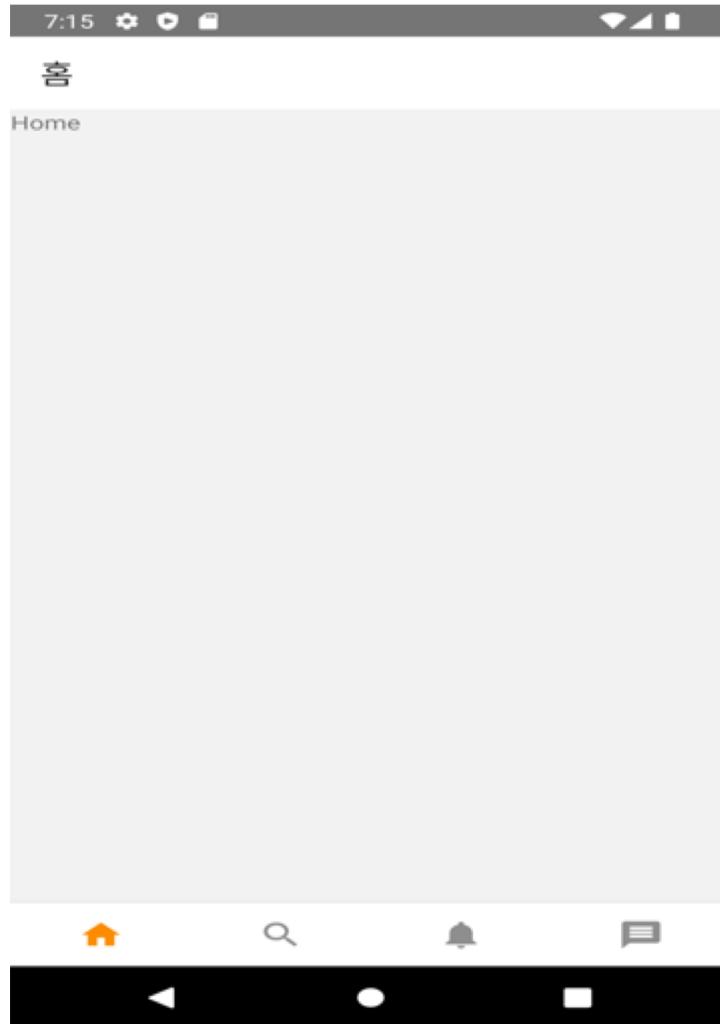
❖ 하단 탭 네비게이터

✓ 탭 커스터마이징

- tabBarActiveTintColor: 활성화된 항목의 아이콘과 텍스트 색상
- tabBarActiveBackgroundColor: 활성화된 항목의 배경색
- tabBarInactiveTintColor: 비활성화된 항목의 아이콘과 텍스트 색상
- tabBarInactiveBackgroundColor: 비활성화된 항목의 배경색
- tabBarShowLabel: 항목에서 텍스트의 가시성 설정(기본값: true)
- tabBarShowIcon: 항목에서 아이콘의 가시성 설정(기본값: false)
- tabBarStyle: 하단 탭 스타일
- tabBarLabelStyle: 텍스트 스타일
- tabBarItemStyle: 항목 스타일
- tabBarLabelPosition: 텍스트 위치 'beside-icon1 (아이콘 우측) / 'below-icon'(아이콘 하단)
- tabBarAllowFontScaling: 시스템 폰트 크기에 따라 폰트 크기를 키울지 결정(기본값: true)
- tabBarSafeAreaInset: SafeAreaView의 forceinset 덮어쓰는 객체(기본값: {bottom:'always', top: 'never'})
- tabBarKeyboardHidesTabBar: 키보드가 나타날 때 하단 탭을 가릴지 결정(기본값: false)

다양한 네비게이터

❖ 하단 탭 네비게이터



다양한 네비게이터

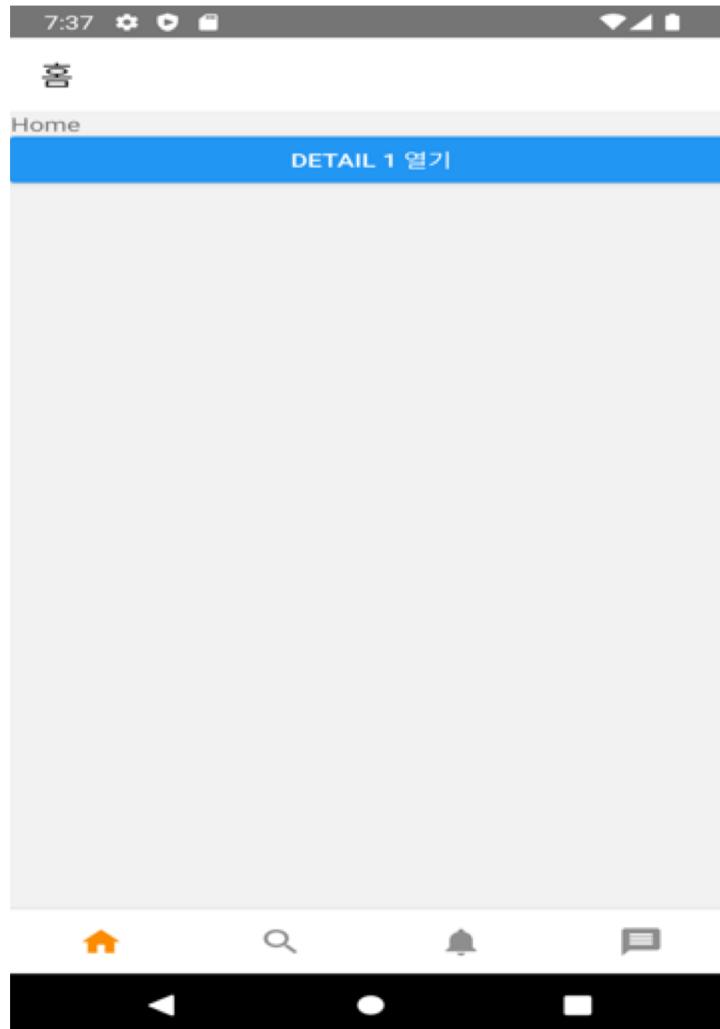
- ❖ 하단 탭 네비게이터

- ✓ 탭 커스터마이징을 위해서 App.js 파일 수정

```
<Tab.Navigator  
    initialRouteName="Home"  
    screenOptions={{  
        tabBarActiveTintColor: '#fb8c00',  
        tabBarShowLabel: false,  
    }}>
```

다양한 네비게이터

❖ 하단 탭 네비게이터



다양한 네비게이터

- ❖ 하단 탭 네비게이터

- ✓ screens 디렉토리에 MainScreen.js 파일 생성

```
import React from 'react';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Text, View, Button } from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';

const Tab = createBottomTabNavigator();

function HomeScreen({ navigation }) {
  return (
    <View>
      <Text>Home</Text>
      <Button
        title="Detail 1 열기"
        onPress={() =>
          navigation.push('Detail', {
            id: 1,
          })
        }
      />
    </View>
  );
}
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터
 - ✓ screens 디렉토리에 MainScreen.js 파일 생성

```
function SearchScreen() {  
  return (  
    <View>  
      <Text>Search</Text>  
    </View>  
  );  
}
```

```
function NotificationScreen() {  
  return (  
    <View>  
      <Text>Notification</Text>  
    </View>  
  );  
}
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터
 - ✓ screens 디렉토리에 MainScreen.js 파일 생성

```
function MessageScreen() {  
  return (  
    <View>  
      <Text>Message</Text>  
    </View>  
  );  
}
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터

- ✓ screens 디렉토리에 MainScreen.js 파일 생성

```
function MainScreen() {
  return (
    <Tab.Navigator
      initialRouteName="Home"
      screenOptions={{
        tabBarActiveTintColor: '#fb8c00',
        tabBarShowLabel: false,
      }}>
      <Tab.Screen
        name="Home"
        component={HomeScreen}
        options={{
          title: '홈',
          tabBarIcon: ({ color, size }) => (
            <Icon name="home" color={color} size={size} />
          ),
        }}
    />
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터

- ✓ screens 디렉토리에 MainScreen.js 파일 생성

```
<Tab.Screen
  name="Search"
  component={SearchScreen}
  options={{
    title: '검색',
    tabBarIcon: ({ color, size }) => (
      <Icon name="search" color={color} size={size} />
    ),
  }}
/>
<Tab.Screen
  name="Notification"
  component={NotificationScreen}
  options={{
    title: '알림',
    tabBarIcon: ({ color, size }) => (
      <Icon name="notifications" color={color} size={size} />
    ),
  }}
/>
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터

- ✓ screens 디렉토리에 MainScreen.js 파일 생성

```
<Tab.Screen
  name="Message"
  component={MessageScreen}
  options={{
    title: '메시지',
    tabBarIcon: ({ color, size }) => (
      <Icon name="message" color={color} size={size} />
    ),
  }}
/>
</Tab.Navigator>
);
}

export default MainScreen;
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터

- ✓ App.js 수정

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/native-stack';
import MainScreen from './screens/MainScreen';
import DetailScreen from './screens/DetailScreen';

const Stack = createStackNavigator();
```

다양한 네비게이터

- ❖ 하단 탭 네비게이터

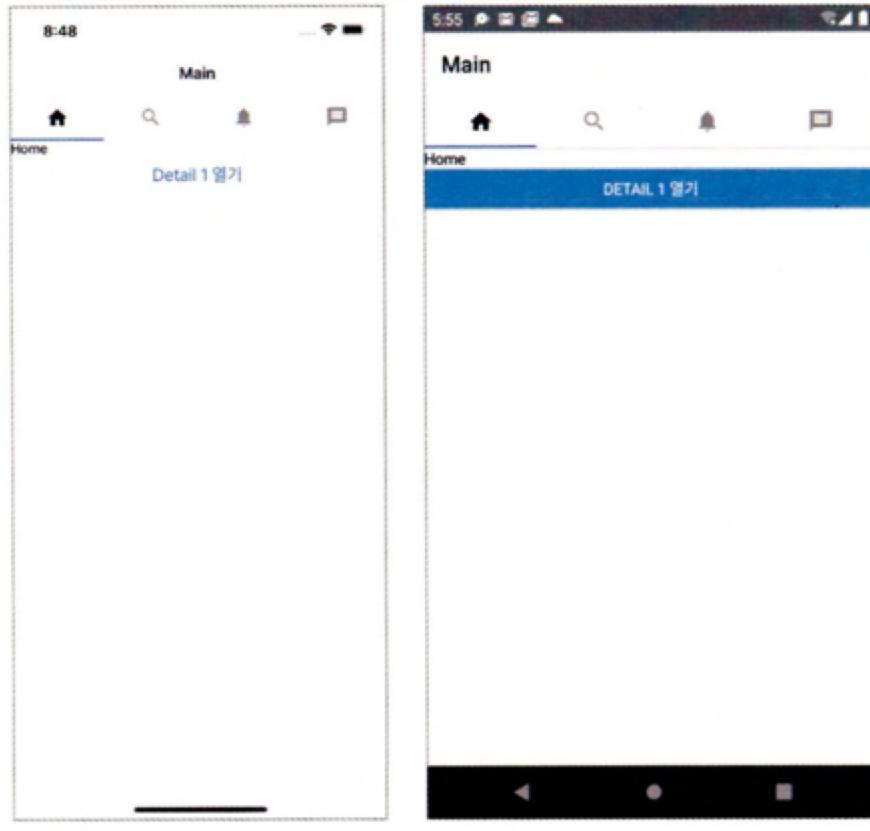
- ✓ App.js 수정

```
function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen
          name="Main"
          component={MainScreen}
          options={{ headerShown: false }}
        />
        <Stack.Screen name="Detail" component={DetailScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default App;
```

다양한 네비게이터

- ❖ 머터리얼 상단 탭 네비게이터
 - ✓ 상단에 탭을 보여주는 네비게이터



다양한 네비게이터

❖ 머터리얼 상단 탭 네비게이터

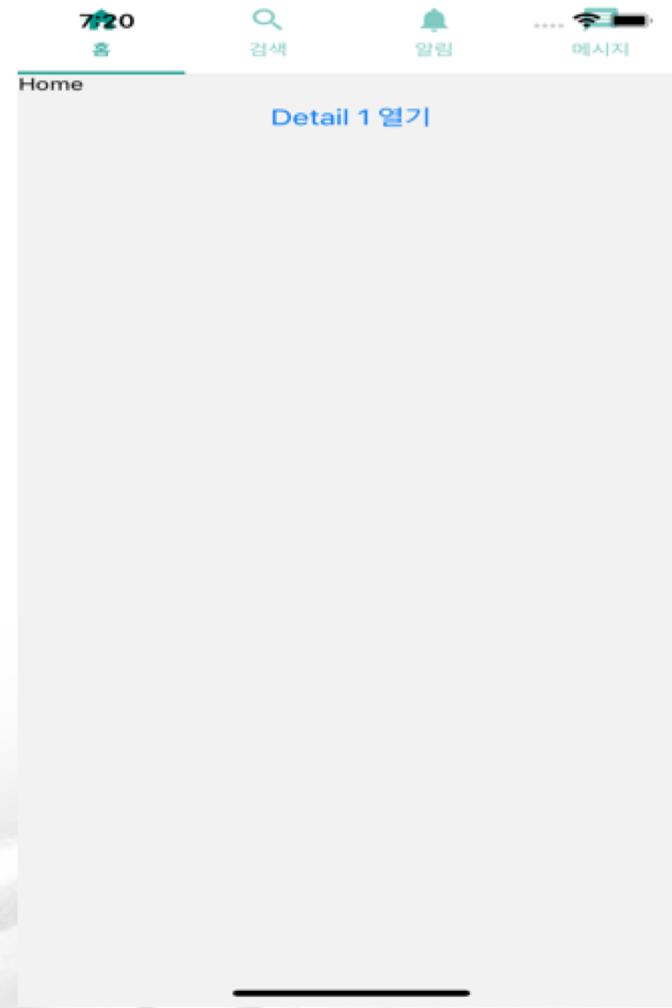
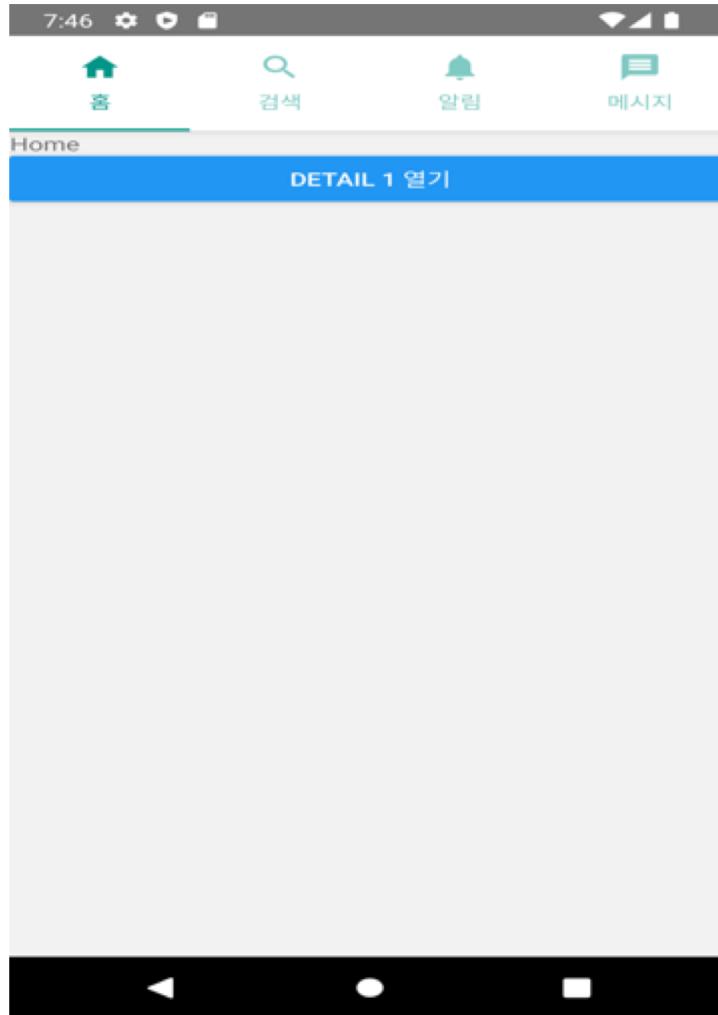
✓ 도큐먼트: <https://reactnavigation.org/docs/material-top-tab-navigator/>

✓ 설치

- yarn add @react-navigation/material-top-tabs react-native-tab-view react-native-pager-view
- cd ios
- pod install
- cd ..
- yarn android
- yarn ios

다양한 네비게이터

- ❖ 머터리얼 상단 탭 네비게이터



다양한 네비게이터

- ❖ 머터리얼 상단 탭 네비게이터
 - ✓ MainScreen.js 파일을 수정

```
import React from 'react';
import { createMaterialTopTabNavigator } from '@react-navigation/material-top-tabs';
import { Text, View, Button } from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';

const Tab = createMaterialTopTabNavigator();

function HomeScreen({ navigation }) {
  return (
    <View>
      <Text>Home</Text>
      <Button
        title="Detail 1 열기"
        onPress={() =>
          navigation.push('Detail', {
            id: 1,
          })
        }
      />
    </View>
  );
}
```

다양한 네비게이터

- ❖ 머터리얼 상단 탭 네비게이터
 - ✓ MainScreen.js 파일을 수정

```
function SearchScreen() {  
    return (  
        <View>  
            <Text>Search</Text>  
        </View>  
    );  
}  
  
function NotificationScreen() {  
    return (  
        <View>  
            <Text>Notification</Text>  
        </View>  
    );  
}
```

다양한 네비게이터

- ❖ 머터리얼 상단 탭 네비게이터

- ✓ MainScreen.js 파일을 수정

```
function MessageScreen() {  
  return (  
    <View>  
      <Text>Message</Text>  
    </View>  
  );  
}
```

다양한 네비게이터

- ❖ 머터리얼 상단 탭 네비게이터
 - ✓ MainScreen.js 파일을 수정

```
function MainScreen() {
  return (
    <Tab.Navigator
      initialRouteName="Home"
      screenOptions={{
        tabBarIndicatorStyle: {
          backgroundColor: '#009688',
        },
        tabBarActiveTintColor: '#009688',
      }}>
    <Tab.Screen
      name="Home"
      component={HomeScreen}
      options={{
        tabBarLabel: '홈',
        tabBarIcon: ({ color }) => <Icon name="home" color={color} size={24} />,
      }}
    />
```

다양한 네비게이터

- ❖ 머터리얼 상단 탭 네비게이터
 - ✓ MainScreen.js 파일을 수정

```
<Tab.Screen
  name="Search"
  component={SearchScreen}
  options={{
    tabBarLabel: '검색',
    tabBarIcon: ({ color }) => (
      <Icon name="search" color={color} size={24} />
    ),
  }}
/>
<Tab.Screen
  name="Notification"
  component={NotificationScreen}
  options={{
    tabBarLabel: '알림',
    tabBarIcon: ({ color }) => (
      <Icon name="notifications" color={color} size={24} />
    ),
  }}
/>
```

다양한 네비게이터

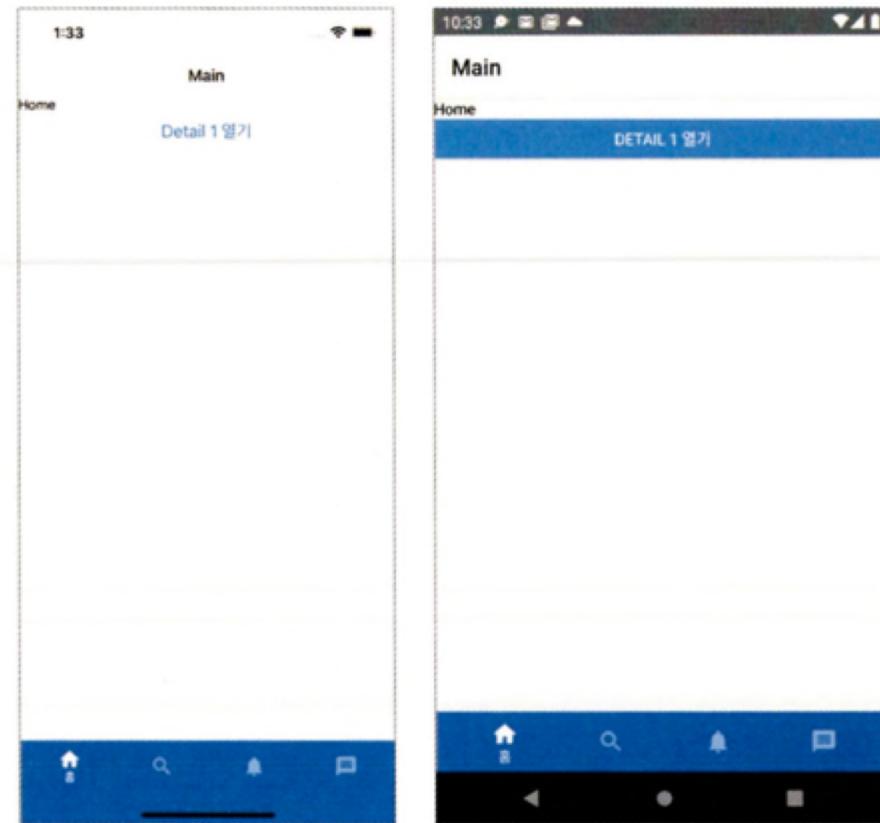
- ❖ 머터리얼 상단 탭 네비게이터
 - ✓ MainScreen.js 파일을 수정

```
<Tab.Screen
  name="Message"
  component={MessageScreen}
  options={{
    tabBarLabel: '메시지',
    tabBarIcon: ({ color }) => (
      <Icon name="message" color={color} size={24} />
    ),
  }}
/>
</Tab.Navigator>
);
}

export default MainScreen;
```

다양한 네비게이터

- ❖ 머터리얼 하단 탭 네비게이터
 - ✓ 하단에 탭을 보여주는 네비게이터

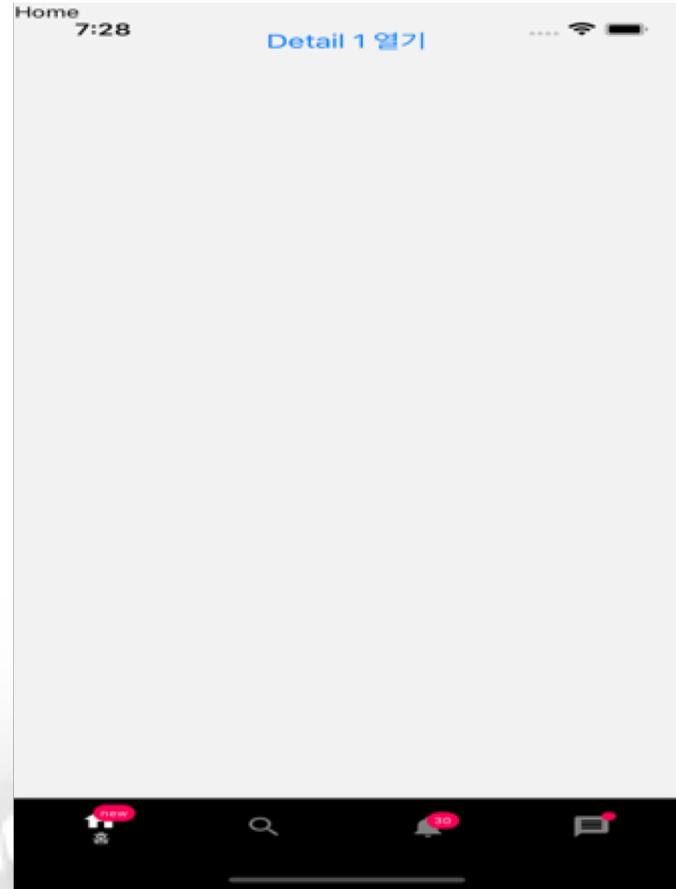
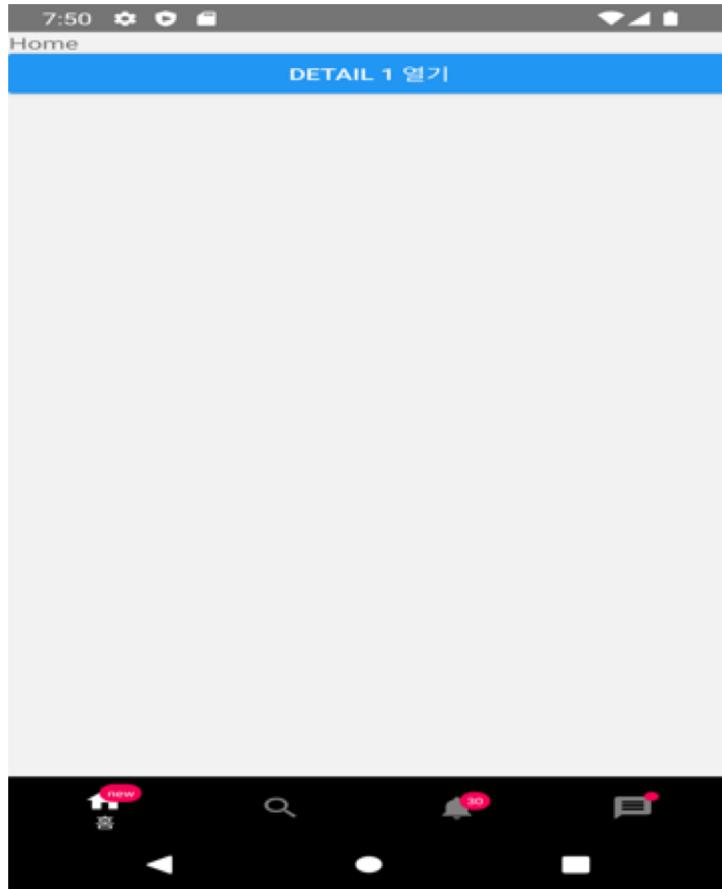


다양한 네비게이터

- ❖ 머터리얼 하단 탭 네비게이터
 - ✓ 도큐먼트: <https://callstack.github.io/react-native-paper/>
 - ✓ 설치
 - yarn add @react-navigation/material-bottom-tabs react-native-paper
 - cd ios
 - pod install
 - cd ..
 - yarn android
 - yarn ios

다양한 네비게이터

- ❖ 머터리얼 하단 탭 네비게이터



다양한 네비게이터

- ❖ 머터리얼 하단 탭 네비게이터
 - ✓ MainScreen.js 파일 수정

```
import React from 'react';
import { createMaterialBottomTabNavigator } from '@react-navigation/material-bottom-tabs';
import { Text, View, Button } from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';

const Tab = createMaterialBottomTabNavigator();

function HomeScreen({ navigation }) {
  return (
    <View>
      <Text>Home</Text>
      <Button
        title="Detail 1 열기"
        onPress={() =>
          navigation.push('Detail', {
            id: 1,
          })
        }
      />
    </View>
  );
}
```

다양한 네비게이터

- ❖ 머터리얼 하단 탭 네비게이터

- ✓ MainScreen.js 파일 수정

```
function SearchScreen() {  
    return (  
        <View>  
            <Text>Search</Text>  
        </View>  
    );  
}  
  
function NotificationScreen() {  
    return (  
        <View>  
            <Text>Notification</Text>  
        </View>  
    );  
}  
  
function MessageScreen() {  
    return (  
        <View>  
            <Text>Message</Text>  
        </View>  
    );  
}
```

다양한 네비게이터

- ❖ 머터리얼 하단 탭 네비게이터

- ✓ MainScreen.js 파일 수정

```
function MainScreen() {
  return (
    <Tab.Navigator initialRouteName="Home">
      <Tab.Screen
        name="Home"
        component={HomeScreen}
        options={{
          tabBarLabel: '홈',
          tabBarIcon: ({ color }) => <Icon name="home" color={color} size={24} />,
          tabBarColor: 'black',
          tabBarBadge: 'new',
        }}
    />
```

다양한 네비게이터

- ❖ 머터리얼 하단 탭 네비게이터

- ✓ MainScreen.js 파일 수정

```
<Tab.Screen
  name="Search"
  component={SearchScreen}
  options={{
    tabBarLabel: '검색',
    tabBarIcon: ({ color }) => (
      <Icon name="search" color={color} size={24} />
    ),
    tabBarColor: 'gray',
  }}
/>
<Tab.Screen
  name="Notification"
  component={NotificationScreen}
  options={{
    tabBarLabel: '알림',
    tabBarIcon: ({ color }) => (
      <Icon name="notifications" color={color} size={24} />
    ),
    tabBarColor: 'green',
    tabBarBadge: 30,
  }}
/>
```

다양한 네비게이터

- ❖ 머터리얼 하단 탭 네비게이터

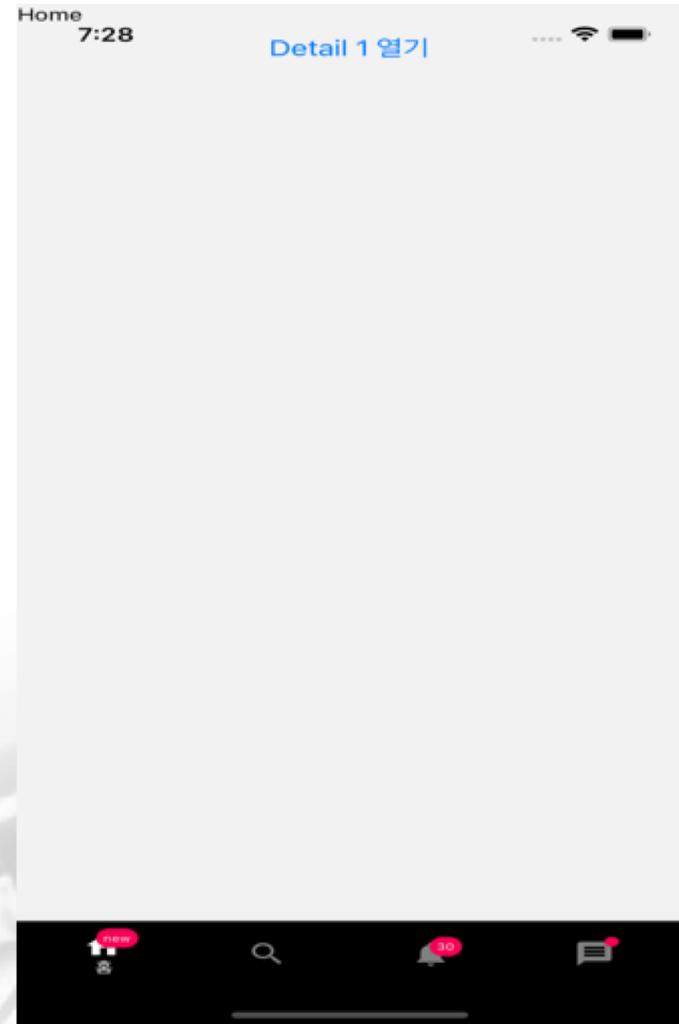
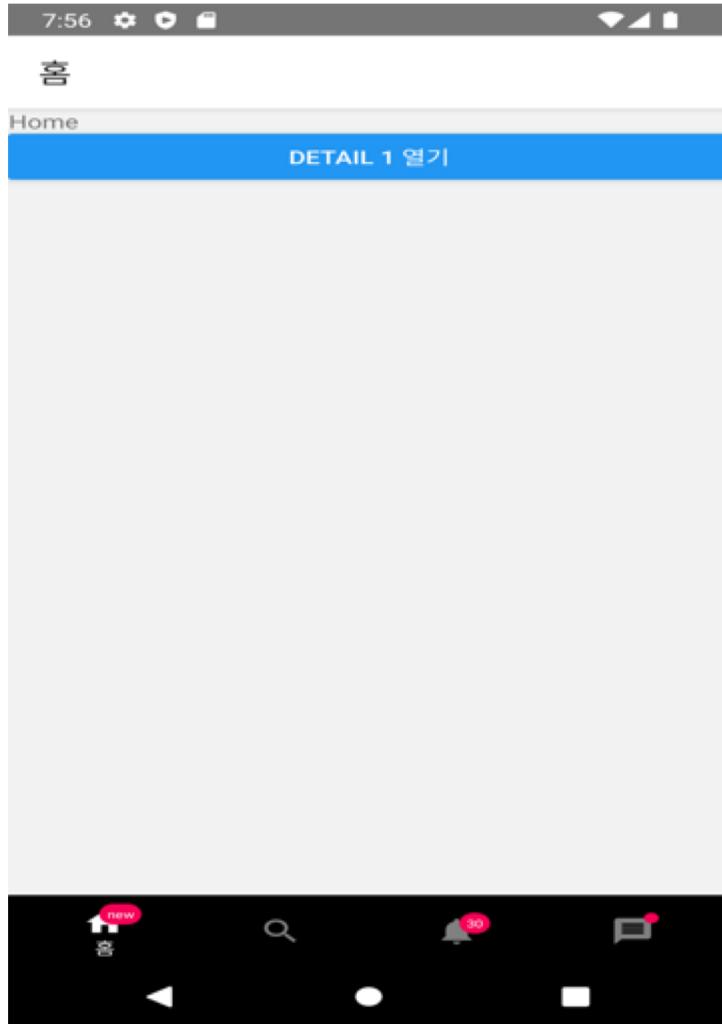
- ✓ MainScreen.js 파일 수정

```
<Tab.Screen
  name="Message"
  component={MessageScreen}
  options={{
    tabBarLabel: '메시지',
    tabBarIcon: ({ color }) => (
      <Icon name="message" color={color} size={24} />
    ),
    tabBarColor: 'blue',
    tabBarBadge: true,
  }}
/>
</Tab.Navigator>
);
}

export default MainScreen;
```

다양한 네비게이터

- ❖ 머티리얼 하단 탭 내비게이터 헤더 타이틀 동기화



다양한 네비게이터

- ❖ 머티리얼 하단 탭 내비게이터 헤더 타이틀 동기화

- ✓ App.js 파일 수정

```
import React from 'react';
import {
  getFocusedRouteNameFromRoute,
  NavigationContainer,
} from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/native-stack';
import MainScreen from './screens/MainScreen';
import DetailScreen from './screens/DetailScreen';

const Stack = createStackNavigator();

function getHeaderTitle(route) {
  const routeName = getFocusedRouteNameFromRoute(route) ?? 'Home';
  const nameMap = {
    Home: '홈',
    Search: '검색',
    Notification: '알림',
    Message: '메시지',
  };
  return nameMap[routeName];
}


```

다양한 네비게이터

- ❖ 머티리얼 하단 탭 네비게이터 헤더 타이틀 동기화

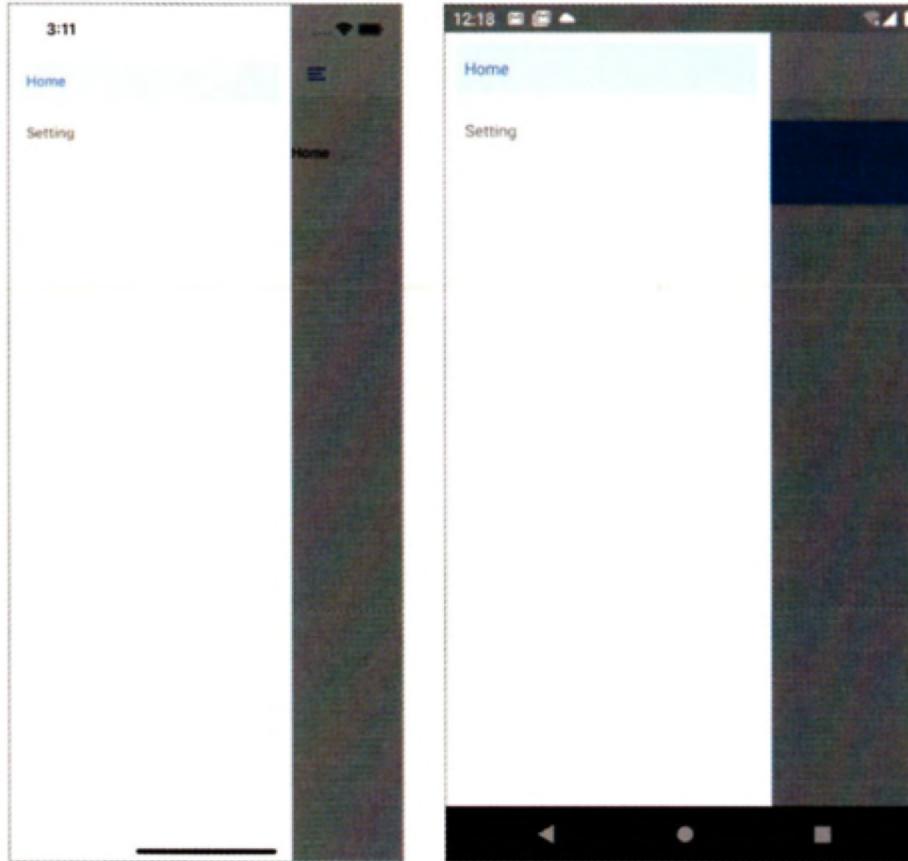
- ✓ App.js 파일 수정

```
function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen
          name="Main"
          component={MainScreen}
          options={({ route }) => ({
            title: getHeaderTitle(route),
          })}
        />
        <Stack.Screen name="Detail" component={DetailScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default App;
```

다양한 네비게이터

- ❖ Drawer Navigator
 - ✓ 우측에 사이드 바를 만들고자 하는 경우 사용하는 네비게이터



다양한 네비게이터

❖ Drawer Navigator

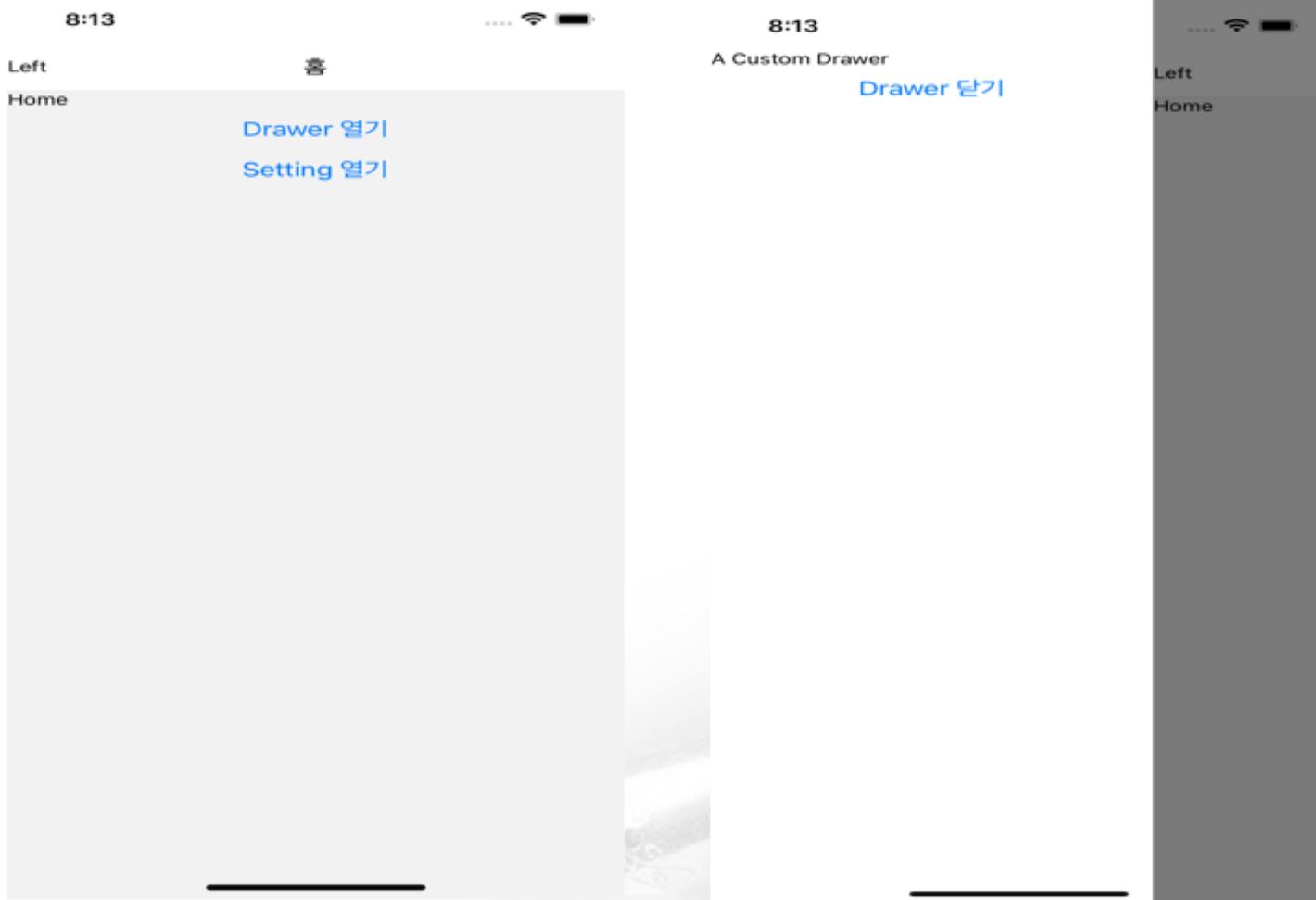
✓ 도큐먼트: <https://reactnavigation.org/docs/drawer-navigator/>

✓ 설치

- yarn add @react-navigation/drawer react-native-gesture-handler react-native-reanimated
- cd ios
- pod install
- cd ..
- yarn android
- yarn ios

다양한 네비게이터

❖ Drawer Navigator



다양한 네비게이터

- ❖ Drawer Navigator

- ✓ app.js 파일 수정

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createDrawerNavigator } from '@react-navigation/drawer';
import { View, Text, Button } from 'react-native';
import { SafeAreaView } from 'react-native-safe-area-context';

const Drawer = createDrawerNavigator();

function HomeScreen({ navigation }) {
  return (
    <View>
      <Text>Home</Text>
      <Button title="Drawer 열기" onPress={() => navigation.openDrawer()} />
      <Button
        title="Setting 열기"
        onPress={() => navigation.navigate('Setting')}
      />
    </View>
  );
}
```

다양한 네비게이터

- ❖ Drawer Navigator

- ✓ app.js 파일 수정

```
function SettingScreen({ navigation }) {  
  return (  
    <View>  
      <Text>Setting</Text>  
      <Button title="뒤로가기" onPress={() => navigation.goBack()} />  
    </View>  
  );  
}
```

다양한 네비게이터

- ❖ Drawer Navigator

- ✓ app.js 파일 수정

```
function App() {
  return (
    <NavigationContainer>
      <Drawer.Navigator
        initialRouteName="Home"
        drawerPosition="left"
        backBehavior="history"
        drawerContent={({ navigation }) => (
          <SafeAreaView>
            <Text>A Custom Drawer</Text>
            <Button
              onPress={() => navigation.closeDrawer()}
              title="Drawer 닫기"
            />
          </SafeAreaView>
        )}>
        <Drawer.Screen
          name="Home"
          component={HomeScreen}
          options={{ title: '홈', headerLeft: () => <Text>Left</Text> }}
        />
      </Drawer.Navigator>
    </NavigationContainer>
  )
}
```

다양한 네비게이터

- ❖ Drawer Navigator

- ✓ app.js 파일 수정

```
<Drawer.Screen  
  name="Setting"  
  component={SettingScreen}  
  options={{ title: '설정' }}  
/>  
</Drawer.Navigator>  
</NavigationContainer>  
);  
}
```

```
export default App;
```

다양한 네비게이터

- ❖ Drawer Navigator

- ✓ babel.config.js 파일 수정

```
module.exports = {  
  presets: ['module:metro-react-native-babel-preset'],  
  plugins: ['react-native-reanimated/plugin'],  
};
```

다양한 네비게이터

❖ Drawer Navigator

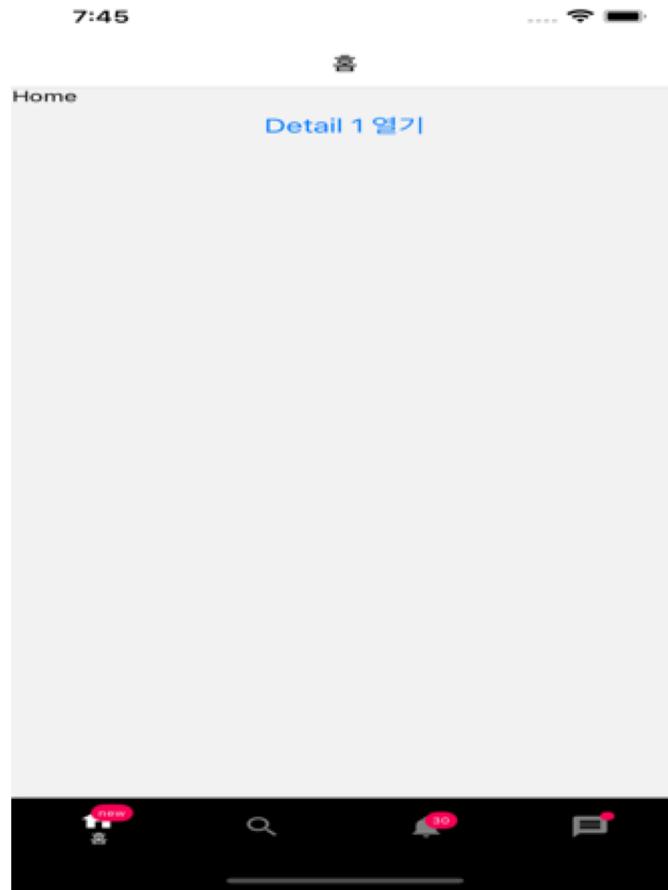
✓ 터미널에서 아래 명령 수행

```
npx react-native start --reset-cache
```

✓ 에뮬레이터 와 시뮬레이터를 모두 종료하고 다시 시작

네비게이션 Hooks

❖ 네비게이션 Hooks



This is a development-only warning and won't be shown in production.

LOG 이 화면을 보고 있어요.
LOG 이 화면을 보고 있어요.
LOG 다른 화면으로 넘어갔어요.
LOG 이 화면을 보고 있어요.

네비게이션 Hooks

❖ 네비게이션 Hooks



This is a development-only warning and won't be shown in production.

LOG 이 화면을 보고 있어요 .
LOG 이 화면을 보고 있어요 .
LOG 다른 화면으로 넘어갔어요 .
LOG 이 화면을 보고 있어요 .
LOG 다른 화면으로 넘어갔어요 .

네비게이션 Hooks

- ❖ 네비게이션 Hooks

- ✓ MainScreen.js 파일을 수정

```
import React, { useCallback } from 'react';
import { createMaterialBottomTabNavigator } from '@react-navigation/material-bottom-tabs';
import { Text, View, Button } from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';
import { useFocusEffect, useNavigation } from '@react-navigation/native';

const Tab = createMaterialBottomTabNavigator();

function OpenDetailButton() {
    const navigation = useNavigation();

    return (
        <Button
            title="Detail 1 열기"
            onPress={() => navigation.push('Detail', { id: 1 })}
        />
    );
}
```

네비게이션 Hooks

- ❖ 네비게이션 Hooks

- ✓ MainScreen.js 파일을 수정

```
function HomeScreen() {
  useFocusEffect(
    useCallback(() => {
      console.log('이 화면을 보고 있어요.');
      return () => {
        console.log('다른 화면으로 넘어갔어요.');
      };
    }, []),
  );

  return (
    <View>
      <Text>Home</Text>
      <OpenDetailButton />
    </View>
  );
}
```

네비게이션 Hooks

- ❖ 네비게이션 Hooks

- ✓ MainScreen.js 파일을 수정

```
function SearchScreen() {  
    return (  
        <View>  
            <Text>Search</Text>  
        </View>  
    );  
}  
  
function NotificationScreen() {  
    return (  
        <View>  
            <Text>Notification</Text>  
        </View>  
    );  
}  
  
function MessageScreen() {  
    return (  
        <View>  
            <Text>Message</Text>  
        </View>  
    );  
}
```

네비게이션 Hooks

❖ 네비게이션 Hooks

- ✓ MainScreen.js 파일을 수정

```
function MainScreen() {
  return (
    <Tab.Navigator initialRouteName="Home">
      <Tab.Screen
        name="Home"
        component={HomeScreen}
        options={{
          tabBarLabel: '홈',
          tabBarIcon: ({ color }) => <Icon name="home" color={color} size={24} />,
          tabBarColor: 'black',
          tabBarBadge: 'new',
        }}
    />
```

네비게이션 Hooks

❖ 네비게이션 Hooks

- ✓ MainScreen.js 파일을 수정

```
<Tab.Screen
  name="Search"
  component={SearchScreen}
  options={{
    tabBarLabel: '검색',
    tabBarIcon: ({ color }) => (
      <Icon name="search" color={color} size={24} />
    ),
    tabBarColor: 'gray',
  }}
/>
<Tab.Screen
  name="Notification"
  component={NotificationScreen}
  options={{
    tabBarLabel: '알림',
    tabBarIcon: ({ color }) => (
      <Icon name="notifications" color={color} size={24} />
    ),
    tabBarColor: 'green',
    tabBarBadge: 30,
  }}
/>
```

네비게이션 Hooks

- ❖ 네비게이션 Hooks

- ✓ MainScreen.js 파일을 수정

```
<Tab.Screen
  name="Message"
  component={MessageScreen}
  options={{
    tabBarLabel: '메시지',
    tabBarIcon: ({ color }) => (
      <Icon name="message" color={color} size={24} />
    ),
    tabBarColor: 'blue',
    tabBarBadge: true,
  }}
/>
</Tab.Navigator>
);
}

export default MainScreen;
```