Version: 1.0

Date: 23 June 2023

Author: Tuqa Al Qadi

Table of Contents:

1. Introduction

The iRet tool is a project that is intended to provide users the ability to create a building sketch in 2D and be able to see that sketch in 3D, and see the energy plus results on the building sketch that was provided.

This project is built on React Native Expo and tailwindCSS for frontend, Nodejs and mongoDB for backend.

This Product is mainly designed for Android users, and supposed to be running on tablets.

2. Getting Started

2.1. System Requirements

- Expo
- NPM
- Node version 16.8.0

2.2. Installation

Download the code from the github repo:
Github Link Frontend

Github Link Backend

APK Link

2.3. Environment Variables

- Frontend:

```
BASE_URL= ** URL OF THE BACKEND ON BCU SERVER **
```

- Backend:

```
SERVER_PORT=**PORT**
DATABASE_URL=**DB URL **
TOKEN_KEY= **TOKEN KEY**
```

3. Developer Guide

3.1. Folder Structure

- **App.js file**: has the navigation of the app
- **assets folder**: has all of the icons required in the project
- **src folder**: has all of the code within it
- **apis folder**: has all energy plus APIs
- **components folder**: has reusable components needed in the app
- **context folder**: has functionalities that are required to be accessible within the whole app screens
- **custom hooks folder**: has some custom hooks for location and simulation
- **screens folder**: has all of the screens within the application

3.2. Core Functionalities

3.2.1. 2D playground / 3D simulator

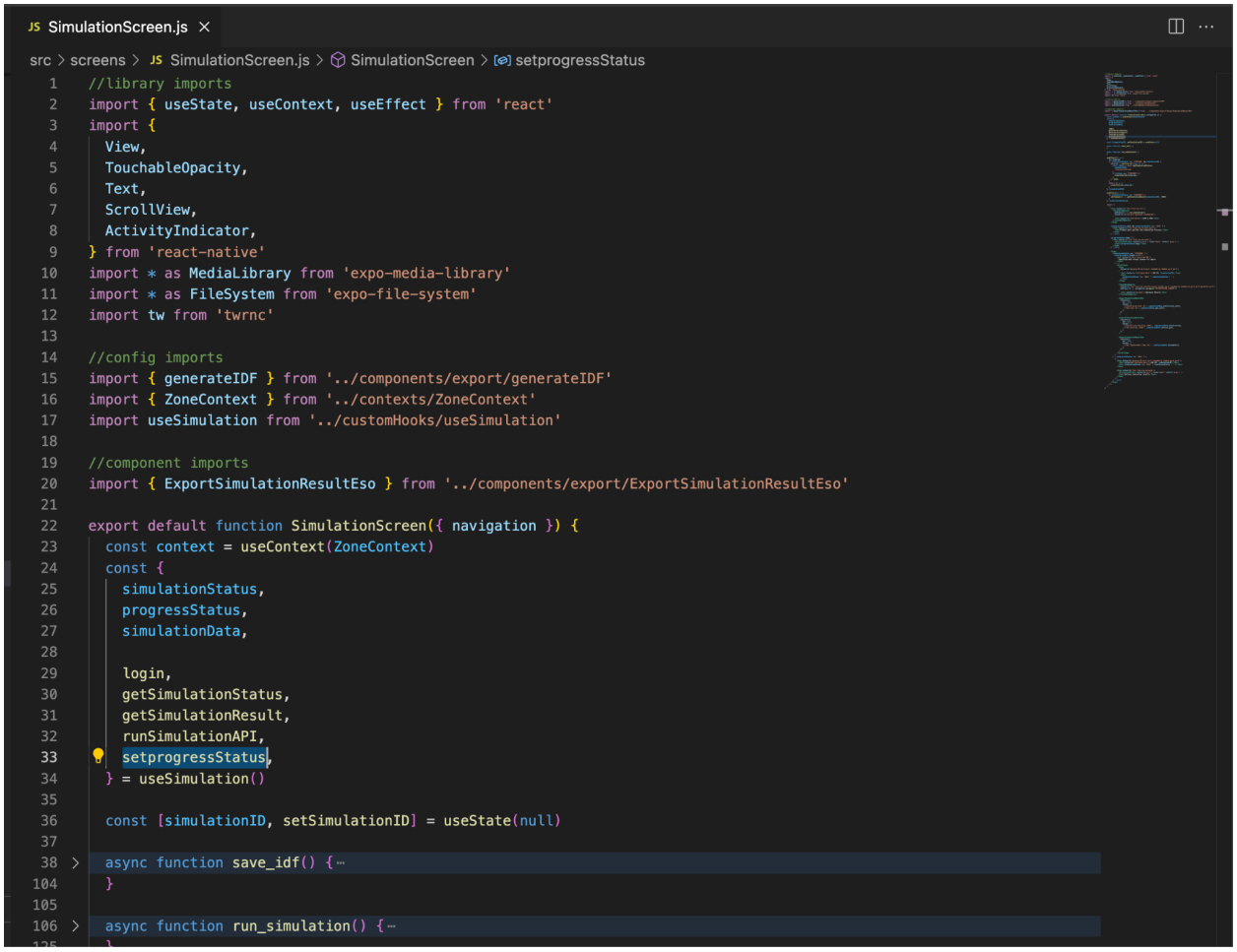
Src > screens > ProjectDetailsScreen.js

Here you will find the start of the 2D and 3D code. In the components folder you will also find a Twod folder and a Threed folder that has all of the required 2D and 3D components that are being used within the playground.

```
return (
  <View style={tw`relative flex-1 flex-col justify-between`}>
    {show3Delement ? (
      <>
        <View style={tw`flex relative h-[90%]`}>
          <Threed />
        </View>

        <TouchableHighlight
          underlayColor="gray"
          onPress={() => {
            setshow3Delement(!show3Delement)
          }}
        >
          <SafeAreaView
            style={tw`absolute bottom-9 left-2 flex flex-row bg-black justify-between rounded-lg h-9 w-9 shadow`}
          >
            <WithLocalSvg
              asset={ThreeDIcon}
              onPress={() => {
                setshow3Delement(!show3Delement)
              }}
              style={tw`m-auto text-white h-6 w-6`}
            />
          </SafeAreaView>
        </TouchableHighlight>
      </>
    ) : (
      <>
        <View style={tw`flex relative h-[90%]`}>
          <View…
          </View>

          <FloorView />
        </View>

        <TouchableHighlight…
        </TouchableHighlight>

        {showFloorsStack ? (…
        ) : null}
```

3.2.2. IDF - Simulation

Src > screens > SimulationScreen.js
Here you will find the start of the simulation screen code. In the components folder you will see a folder called export that has all of the components being used within the simulation screen.


SimulationScreen.js
src > screens > SimulationScreen.js > SimulationScreen > setprogressStatus
1 //library imports
2 import { useState, useContext, useEffect } from 'react'
3 import {
4 View,
5 TouchableOpacity,
6 Text,
7 ScrollView,
8 ActivityIndicator,
9 } from 'react-native'
10 import * as MediaLibrary from 'expo-media-library'
11 import * as FileSystem from 'expo-file-system'
12 import tw from 'twrnc'
13
14 //config imports
15 import { generateIDF } from '../components/export/generateIDF'
16 import { ZoneContext } from '../contexts/ZoneContext'
17 import useSimulation from '../customHooks/useSimulation'
18
19 //component imports
20 import { ExportSimulationResultEso } from '../components/export/ExportSimulationResultEso'
21
22 export default function SimulationScreen({ navigation }) {
23 const context = useContext(ZoneContext)
24 const {
25 simulationStatus,
26 progressStatus,
27 simulationData,
28
29 login,
30 getSimulationStatus,
31 getSimulationResult,
32 runSimulationAPI,
33 setprogressStatus,
34 } = useSimulation()
35
36 const [simulationID, setSimulationID] = useState(null)
37
38 async function save_idf() {
104 }
105
106 async function run_simulation() {