|  |
| --- |
|  |
| React Native project |
| A Javascript developer technical test |
|  |
| **AFIR Mohamed Belkacem** |
| **5/4/2025** |

|  |
| --- |
| A React Native project made by using the code from the NextJS project to make a mobile application. The project is to be optimized and responsive to work fast and load all products from an API, similar to the NextJS project. |

Contents

[1. Introduction 3](#_Toc194929325)

[1.1 Introduction: 3](#_Toc194929326)

[1.2 Scope: 3](#_Toc194929327)

[1.3 Overview: 3](#_Toc194929328)

[2. General description 3](#_Toc194929329)

[2.1 Website function: 3](#_Toc194929330)

[2.2 User characterization: 3](#_Toc194929331)

[2.3 User objective: 3](#_Toc194929332)

[3. Test objectives 3](#_Toc194929333)

[3.1 Pages to implement: 4](#_Toc194929334)

[3.1.1 Mobile application: 4](#_Toc194929335)

[3.1.2 Adaptive User Interface: 4](#_Toc194929336)

[3.1.3 Performance Optimization: 4](#_Toc194929337)

[4. Project structure and needs 4](#_Toc194929338)

[4.1 Required libraries: 4](#_Toc194929339)

[4.1.1 Npm Install: 4](#_Toc194929340)

[4.1.2 React icons: 4](#_Toc194929341)

[4.1.3 Axios: 4](#_Toc194929342)

[4.1.4 asyncStorage: 4](#_Toc194929343)

[4.1.5 useNavigation: 5](#_Toc194929344)

[4.2 Running the project: 5](#_Toc194929345)

[4.3 Files structure: 5](#_Toc194929346)

[5. Code functions 6](#_Toc194929347)

[5.1 login.js: 6](#_Toc194929348)

[5.2 MainPage.js: 9](#_Toc194929349)

[5.2 MainPageAdmin.js: 12](#_Toc194929350)

[5.3 products/[id]/page.js: 15](#_Toc194929351)

[5.5 productsAdmin/[id]/page.js: 17](#_Toc194929352)

[5.6 app.js: 18](#_Toc194929353)

[6. Technical issues and constriction 19](#_Toc194929354)

[6.1 fakestoreapi: 19](#_Toc194929355)

[6.2 CSS properties: 20](#_Toc194929356)

[Conclusion 21](#_Toc194929357)

# Introduction

## 1.1 Introduction:

The purpose of this document is to define the requirements of the project, as well as describe the site’s functionalities and constraints.

## 1.2 Scope:

The Provider of this test and the projects it contains is the company of Weasydoo, It was provided for the purpose of testing the programmer’s abilities and consider him for a job opportunity as a Javascript developer. This project is due on the 8th of April 2025.

## 1.3 Overview:

This project is an E-commerce store mobile app, designed using the Javascript framework called React Native with the help of a mock API. It contains a Products main page for displaying products with a search bar, a page for displaying specific products in full detail, variants of those pages for Admin to make changes to the products, and a login page to allow the admin to access the admin pages.

# General description

## 2.1 Website function:

The app should use React Native to display the product list from fakestoreapi.com, mimicking an actual e-commerce store site, with also search features and login features for users.

## 2.2 User characterization:

The user will be the employees of Weasydoo for the purpose of evaluating the programmer’s Javascript skills as they use the functionalities of the site to navigate through it.

## 2.3 User objective:

The user is to evaluate the structure and functionality of the mobile app, ensure that each component works well, that the app is responsive and well optimized, and that all of the objectives provided by the user are present.

# Test objectives

In the React Native project, we'll be using the fakestoreapi.com API for testing and prototyping purposes. This API provides fictitious data for an online store, including products, categories and users.

## 3.1 Pages to implement:

3.1.1 Mobile application: Convert the existing Next.js application into a mobile application using React Native. Reproduce all the functions of the website faithfully.

3.1.2 Adaptive User Interface: Design an adaptive user interface for mobile devices, taking into account the specificities of navigation and interaction on smartphones and tablets.

3.1.3 Performance Optimization: Optimize the performance of the mobile application to ensure fast loading times and fluid responsiveness, even on weaker network connections.

# Project structure and needs

## 4.1 Required libraries:

4.1.1 Npm Install: React Native is a ReactJs variant for making mobile applications, and as such requires node\_modules folder to work.

Open cmd or powershell and go to the directory in which the project folder is. If the folder doesn’t contain (node\_modules) folder, then enter the folder and then type **npm install.**

$ npm install

4.1.2 React icons: The icons required for the login input icons can be installed using npm install react-icons.

$ npm install react-icons –save

4.1.3 Axios: Axios is an important library, it allows for the usage of API calls more easily. Type npm install axios.

$ npm install axios

### 4.1.4 asyncStorage:

This library is useful for storing strings such as the authentification token in the local storage and extracting it or deleting it. Type: npm install @react-native-async-storage/async-storage

$ npm install @react-native-async-storage/async-storage

4.1.5 useNavigation: this library is used to navigate between pages in react native. Type: npm install @react-navigation/native @react-navigation/native-stack

npm install @react-navigation/native @react-navigation/native-stack

## 4.2 Running the project:

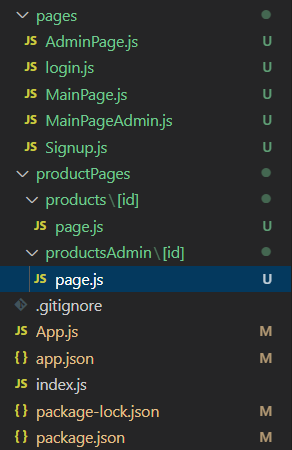
To run the project, go to the project folder and then type:

$ npm run web

Open <http://localhost:8081> with your browser to see the result.

## 4.3 Files structure:

The files are structured as similar to the NextJS project with minor difference; there is the apps folder and the pages folder.



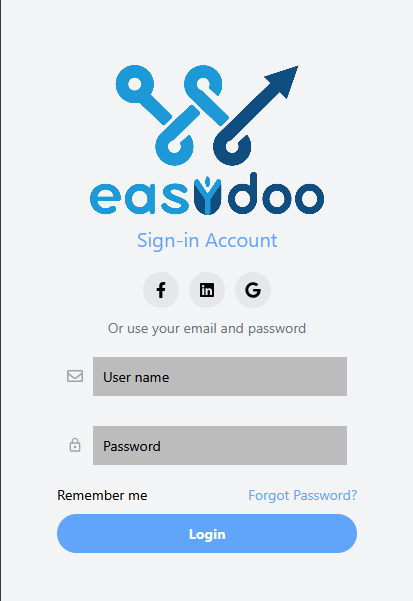
There are two folders that hold most of the code, the **pages** folder and the **productPages** folder, the **pages** folder holds the basic pages for the app, and the **productPages** folder holds the pages that display the product based on ID.

* **MainPage.js:** for the main page that displays the products without logging in.
* **Login.js:** for the login page.
* **MainPageAdmin.js:** for the admin version of the main page where editing can occur.
* **Signup.js & AdminPage.js:** obsolete files.

In the first picture there is also the **/productPage/products/[id]/page.js** and **/productPage /productsAdmin/[id]/page.js**, these two are for when the user clicks on one of the products to be directed to the page that contains more details of the product. The first one is for the normal page, and the second one for the admin to allow them to edit the product’s details.

# Code functions

## login.js:



import React, { useState, useEffect } from 'react';

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

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

import axios from 'axios';

import { FaFacebookF, FaLinkedin, FaGoogle, FaRegEnvelope } from 'react-icons/fa';

import { MdLockOutline } from 'react-icons/md';

import AsyncStorage from '@react-native-async-storage/async-storage';

export default function Login() {

  const navigation = useNavigation(); //for navigating between pages

  const [credentials, setCredentials] = useState({

    username: '',

    password: ''

  }); //variable that stores username and password

  const [error, setError] = useState(""); //error catching variable

The imports are for important hooks, **useState** for variables and **useEffect** for when the page loads. **Axios** for API call and **useNavigation** for managing links. The **icons** for icons, and **asyncStorage** to store strings in local storage.

  //function for handling the input from username and password inputs

  const handleChange = (name, value) => {

    setCredentials({ ...credentials, [name]: value });

  };

  //function that logs the user in

  const handleLogin = async () => {

    try {

      setError('');

      const response = await axios.post('https://fakestoreapi.com/auth/login', credentials, {

        headers: { 'Content-Type': 'application/json' }

      });

      const token = response.data.token; //takes the login token from the response

      if (token) {

        await AsyncStorage.setItem("authToken", token); //add the token to the local storage

        navigation.navigate("MainAdmin"); //navigate to the admin page

      }

    } catch (error) {

      setError('Username or Password incorrect');

    }

  };

**handleChange** is the function used to change the value of the input of the username and password input HTML in the credentials variable.

The HTML for username and password inputs:

        <View style={styles.inputBox}>

          <FaRegEnvelope style={styles.iconInline} />

          <TextInput placeholder="User name" style={styles.input} onChangeText={text => handleChange('username', text)}/>

        </View>

        <View style={styles.inputBox}>

          <MdLockOutline style={styles.iconInline} />

          <TextInput placeholder="Password" secureTextEntry style={styles.input} onChangeText={text => handleChange('password', text)}/>

        </View>

**handleLogin** is the function where upon clicking on the login button, will take the credentials variable and send it to the API, receiving a login token in response that is stored in the **localStorage**, then going to the **MainPageAdmin** page. If the login fails then the error variable will display a message through the HTML to inform the user that the username or password is incorrect.

The HTML for **handleLogin**:

        {/\* if username or password is wrong \*/}

        {error ? <Text style={styles.error}>{error}</Text> : null}

        {/\* login button \*/}

        <TouchableOpacity style={styles.loginButton} onPress={handleLogin}>

          <Text style={styles.loginButtonText}>Login</Text>

        </TouchableOpacity>

**useEffect** checks to see if the page already has the login token, and if it does then to take the user out of the login page into the main admin page:

  //code that runs when the page loads

  useEffect(() => {

    //gets the list of users ex. username: johnd, password:m38rmF$

    // axios.get('https://fakestoreapi.com/users')

    // .then(response => console.log(response.data))

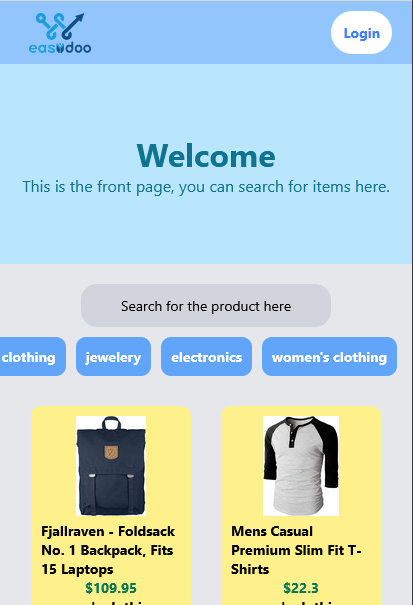
    AsyncStorage.getItem("authToken").then(token => {

      if (token) navigation.navigate("MainAdmin"); //prevents the user from accessing this page if they're already logged in.

    });

  }, []);

## 5.2 MainPage.js:



import React, { useEffect, useMemo, useState } from 'react';

import { View, Text, TextInput, Image, TouchableOpacity, ScrollView, StyleSheet } from 'react-native';

import axios from 'axios';

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

import AsyncStorage from '@react-native-async-storage/async-storage';

function MainPage() {

  const navigation = useNavigation(); //for navigating between pages

  const [items, setItems] = useState([]); // the products list

  const [Query, setQuery] = useState(""); //for the search bar

  const [selectedCategory, setSelectedCategory] = useState(""); //for the category filter

The important variables here are **items**, which is used to display and manage items retrieved from the API. **Query**, used for dealing with the search bar and filtering. **SelectedCategory**, used to filter items based on the category value.

  //code that runs when loading the page

  useEffect(() => {

    const checkToken = async () => {

      try {

        const token = await AsyncStorage.getItem('authToken'); //gets the login tokem form local storage

        if (token) {

          navigation.navigate('MainAdmin'); //navigate to the admin page if it exists

        } else {

          apiCall(); //calls this function

        }

      } catch (error) {

        console.error('error message:', error)

      }

    };

    checkToken();

  }, []);

**useEffect** calls the **apiCall** function, which is responsible for calling all the products from the API and **setItems** into the items variable. The use of **useEffect** is to have the API call when the page is loaded, it also checks if a login token exist and if it does, to navigate to the admin page.

  const desc = (productId) => {

    navigation.navigate('ProductDetails', { id: productId }); // You need to set up this screen

  };

//the HTML component

<TouchableOpacity onPress={() => desc(item.id)}>

    <Text style={styles.itemTitle}>{item.title}</Text>

</TouchableOpacity>

This function is an **onClick** function, where if the user clicks on the product’s name, it will take them to the product’s page.

  //code for handling filtration, both search bar and category

  const filterItems = useMemo(() => { //useMemo so that the API is not called upon multiple times

    return items.filter((item) => {

      const matchesQuery = item.title.toLowerCase().includes(Query.toLowerCase()); //converts the input to lowercase

      const matchesCategory = selectedCategory ? item.category.toLowerCase() === selectedCategory.toLowerCase() : true;

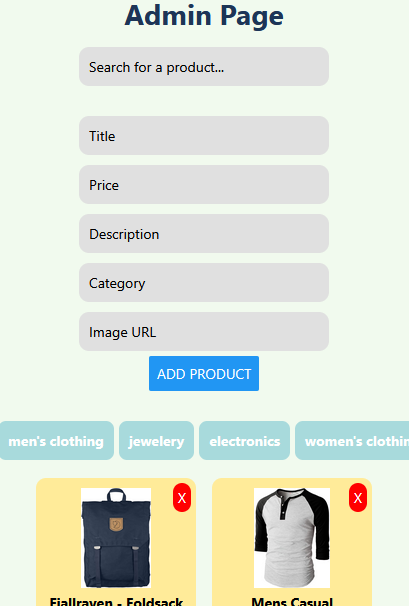
      return matchesQuery && matchesCategory;

    });

  }, [items, Query, selectedCategory]);

This function is for filtering products depending on what was typed in the search bar. **useMemo** is to cache the values so that they’re not recalculated. **toLowerCase** allows for the search feature to not care about the casing of the letters.

## MainPageAdmin.js:



import React, { useEffect, useState, useMemo } from 'react';

import { View, Text, TextInput, Button, ScrollView, Image, TouchableOpacity, StyleSheet,

} from 'react-native';

import axios from 'axios';

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

import AsyncStorage from '@react-native-async-storage/async-storage';

const MainPageAdmin = () => {

  const navigation = useNavigation(); //for navigating between pages

  const [items, setItems] = useState([]); // the products list

  const [add, setAdd] = useState({

    title: '',

    price: '',

    description: '',

    category: '',

    image: '',

  }); //the variable for adding a new product

  const [query, setQuery] = useState(''); //for the search bar

  const [selectedCategory, setSelectedCategory] = useState(''); //for the category filter

  const [loading, setLoading] = useState(true); //for loading before the products list is loaded

This page is similar to the **MainPage** while adding the functionalities of adding products through the **add** variable, a **delete** button to remove products, and **isAuthenticated** variable to ensure the user has a login token to access this page.

  //handles the code for logging out from the admin page

  const handleLogout = async () => {

    try {

      await AsyncStorage.removeItem("authToken"); //removes the login token from local storage

      //use this to check if the token was removed

      // console.log("Logged out successfully");

      navigation.navigate('Main'); //go to the main page

    } catch (error) {

      console.error("Logout failed:", error)

    }

  };

**HandleLogout** removes the token from local storage and takes the user back to the original **MainPage**.

  //the function that adds a new product

  const handleAdd = async () => {

    try {

        const response = await axios.post('https://fakestoreapi.com/products', add);

        setItems(prev => [...prev, response.data]); //modifies items to add this new item

        setAdd({ title: '', price: 0.0, description: '', category: '', image: '' }); //return the add variable to default values

    } catch (error) {

        console.error("Error adding product:", error);

    }

};

      {/\* Add Product Form \*/}

      <View style={styles.form}>

        <TextInput placeholder="Title" value={add.title} onChangeText={text => handleChange('title', text)} style={styles.input}/>

        <TextInput placeholder="Price" value={add.price} onChangeText={text => handleChange('price', text)} keyboardType="numeric" style={styles.input}/>

        <TextInput placeholder="Description" value={add.description} onChangeText={text => handleChange('description', text)} style={styles.input}/>

        <TextInput placeholder="Category" value={add.category} onChangeText={text => handleChange('category', text)} style={styles.input}/>

        <TextInput placeholder="Image URL" value={add.image} onChangeText={text => handleChange('image', text)} style={styles.input}/>

        <Button title="Add Product" onPress={handleAdd} style={styles.button}/>

      </View>

**HandleAdd** allows for adding a new product with the data inputted in the inputs HTML.

      {/\* Category Filters \*/}

      <ScrollView horizontal style={styles.categoryRow}>

        {[...new Set(items.map(item => item.category))].map((category, index) => (

          <TouchableOpacity key={index} onPress={() => setSelectedCategory(category)} style={[ styles.categoryBtn, selectedCategory === category && styles.activeCategory,]}>

            <Text style={styles.categoryText}>{category}</Text>

          </TouchableOpacity>

        ))}

        {selectedCategory !== '' && (

          <TouchableOpacity onPress={() => setSelectedCategory('')} style={styles.clearBtn}>

            <Text style={styles.categoryText}>Clear</Text>

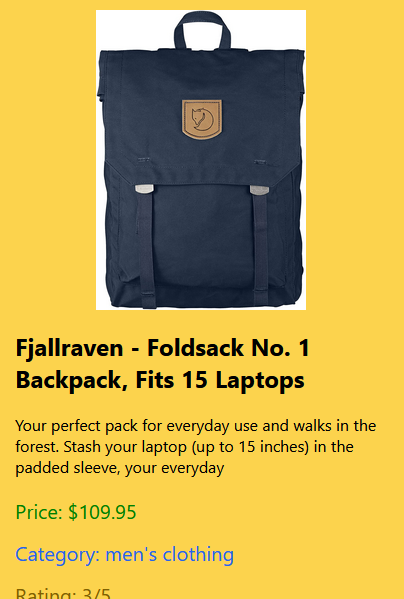
          </TouchableOpacity>

        )}

      </ScrollView>

This HTML allows for filtering the products by categories. The second button clears the category and displays the full list with clicked.

## products/[id]/page.js:



import React, { useEffect, useState } from 'react';

import { View, Text, Image, StyleSheet, TouchableOpacity, ScrollView, ActivityIndicator } from 'react-native';

import axios from 'axios';

function ProductScreen({ route, navigation }) {

  const { id } = route.params || {}; //gets the id from the url

  const [product, setProduct] = useState(null); //holds the product's information

This page contains all the details of the product including title, description, price, category and image.

**Params** gets the product’s id from either **MainPage** or **MainPageAdmin**.

  //code that runs when the page is called

  useEffect(() => {

    if (id) {

      fetchProduct(); //get this function if there is an id

    }

  }, [id]);

  //function that fetches product information

  const fetchProduct = async () => {

    try {

      const response = await axios.get(`https://fakestoreapi.com/products/${id}`);

      setProduct(response.data); //puts the product's properties in the product variable.

    } catch (error) {

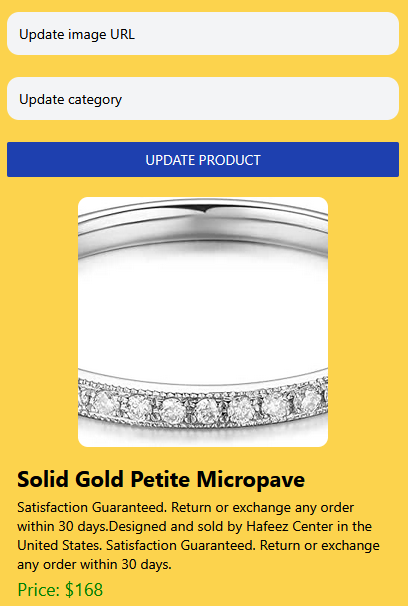
      console.error('Error fetching product:', error);

    }

  };

This function fetches the specific product using the **id** from the **url** and stores it in the **product** variable, it runs when the page loads.

## 5.5 productsAdmin/[id]/page.js:



Similar to the above page, with added input HTML for updating the product properties.

      <View style={styles.inputContainer}>

        <TextInput placeholder="Update title" onChangeText={text => handleChange('title', text)} style={styles.input} />

        <TextInput placeholder="Update price" onChangeText={text => handleChange('price', text)} keyboardType="numeric" style={styles.input} />

        <TextInput placeholder="Update description" onChangeText={text => handleChange('description', text)} multiline style={styles.input} />

        <TextInput placeholder="Update image URL" onChangeText={text => handleChange('image', text)} style={styles.input} />

        <TextInput placeholder="Update category" onChangeText={text => handleChange('category', text)} style={styles.input} />

        <Button title="Update Product" onPress={handleUpdate} color="#1e40af" />

      </View>

The code below handles updating the product’s parameters using the API put function. The second **setUpdate** is to clear the update inputs after updating.

  const handleUpdate = async () =>{

    if (!product || Object.keys(update).length === 0) return; // Don't update if no changes

    try{

      const response = await axios.put(`https://fakestoreapi.com/products/${id}`, {...product,...update});

      setProduct(response.data);

      setUpdate({});

    }catch (error) {

      console.error("Error fetching product:", error);

    }

  }

## 5.6 app.js:

import MainPage from './pages/MainPage';

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

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

import Login from './pages/login';

import ProductScreen from './productPages/products/[id]/page';

import MainPageAdmin from './pages/MainPageAdmin';

import ProductPage from './productPages/productsAdmin/[id]/page';

export default function App() {

  const Stack = createNativeStackNavigator();

  return (

    <NavigationContainer>

      <Stack.Navigator initialRouteName='Main' >

        <Stack.Screen name="Main" component={MainPage}/>

        <Stack.Screen name="login" component={Login}/>

        <Stack.Screen name="MainAdmin" component={MainPageAdmin}/>

        <Stack.Screen name="ProductDetails" component={ProductScreen}/>

        <Stack.Screen name="ProductAdmin" component={ProductPage}/>

      </Stack.Navigator>

    </NavigationContainer>

// add options={{ headerShown: false }} to the stacks if you want to remove the header.

  );}

The main file that displays the project in the app; It contains the navigation code incased in the **NavigationContainer** component; further pages can be added here.

# Technical issues and constriction

Most constraints came from either the API being a **fakestoreapi**, and as such would not actually commit the **CRUD** operations when called, or time being limited as to not allow for different approaches to the coding or **CSS**.

## fakestoreapi:

Since the API used in the project isn’t a real API, it does not truly update itself when using CRUD operations.

  //function that handles updating the product's properties

  const handleUpdate = async () => {

    if (!product || Object.keys(update).length === 0) return; //if there is no product or the inputs are empty, do nothing

    try {

      const response = await axios.put(`https://fakestoreapi.com/products/${id}`, {

        ...product,

        ...update,

      });

      setProduct(response.data); //update the product

      setUpdate({}); //set the update variable to empty object

    } catch (error) {

      console.error("Error updating product:", error);

    }

  };

This code in the product page will only update the product’s properties in the specific page, but it will not update it in the main page. The **update** function of **CRUD** is in the product page as it allows for the all properties to be changed.

  //the function that adds a new product

  const handleAdd = async () => {

    try {

        const response = await axios.post('https://fakestoreapi.com/products', add);

        setItems(prev => [...prev, response.data]); //modifies items to add this new item

        setAdd({ title: '', price: 0.0, description: '', category: '', image: '' }); //return the add variable to default values

    } catch (error) {

        console.error("Error adding product:", error);

    }

};

These two functions in the **MainPageAdmin** also have the same problem, it was determined that this is a test and it’s more important to showcase that these **CRUD** functions work properly.

## CSS properties:

A sudden change in design between NextJS and React Native is that tailwind is slightly different in Native, and as such using normal CSS syntax was preferable.

EX:

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#f3f4f6',

    alignItems: 'center',

    justifyContent: 'center',

    padding: 20,

  },

  logo: {

    fontSize: 24,

    fontWeight: 'bold',

    marginBottom: 10,

    cursor: "pointer",

  },

  header: {

    fontSize: 20,

    color: '#60a5fa',

    marginBottom: 10,

  },

  socialIcons: {

    flexDirection: 'row',

    marginVertical: 10,

  },

  icon: {

    padding: 10,

    marginHorizontal: 5,

    backgroundColor: '#e5e7eb',

    borderRadius: 50,

  },

# Conclusion

This product showcases the many differences between NextJS and React Native, which caused different approach to some of the common Javascript practices going from one project to another.

Further improvements:

* Improving CSS usage.
* Further usage of local storage.
* Performance can be improved further.

Acknowledgement:

* (https://fakestoreapi.com/) for dummy product data.
* React Native and Axios for the core foundation of this project.
* Weasydoo for providing this test.