MOBILE APPLICATION USING REACT NATIVE

A project report submitted for the pre-final year of
Bachelor of Technology in Computer Science Engineering

By

P.N.J.Sirisha (N150838)

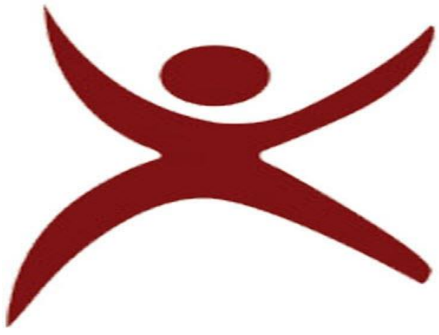
V.Srilakshmi (N150579)

Sk.Sameera (N150410)

J.Sravani (N150364)

Under the Guidance of

Mr.Krishna Kumar Singh



Guest Faculty of Department of Computer Science Engineering

Rajiv Gandhi University of Knowledge Technologies,Nuzvid

Nuzvid , Krishna – 521202

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P.Government Act 18 of 2008)

RGUKT-NUZVID,Krishna Dist -521202

CERTIFICATE OF PROJECT COMPLETION

This is to certify that the work entitled,"**MOBILE APPLICATION USING REACT NATIVE**" is the bonafield work of **P.N.J.Sirisha(N150838),V.Srilakshmi(N150579),Sk.Sameera(N150410), J.Sravani(N150364)** submitted in partial fulfilment of the requirement for the award of marks in pre-final year of Bachelor of Technology in the Department of Computer Science and Engineering under my guidance during the academic year 2019-2020. In my opinion, this project is worth of consideration for the award of marks in accordance with the Department and University regulations.

Date: 2 March 2021

Place: Nuzvid

.....	**.....**
Mr.Krishna Kumar Singh	**Mr.Kumar Anurupam**
Project Supervisor	**Head of Department**
Assistant Professor	**Assistant Professor**
Dept of CSE	**Dept of CSE**

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P.Government Act 18 of 2008)

RGUKT-NUZVID,Krishna Dist -521202

DECLARATION

We here declare that this summer project entitled "**MOBILE APPLICATION USING REACT NATIVE** " is a bonafield work done of **P.N.J.Sirisha (N150838**), **V.Srilakshmi (N150579)**, **Sk.Sameera (N150410)**, **J.Sravani (N150364)** and subtitled to **Department of Computer Science and Engineering** of **Rajiv Gandhi University of Knowledge Technologies** in the partial fulfilment of the requirement of degree in Bachelor of Technology is of our own and it is not submitted to any other university or has been published any time before.

Date: 2 FEB 2021

Place : NUZVID

P.N.J.Sirisha(N150838)

V. Srilaxmi(N150579)

Sk. Sameera(N150410)

J. Sravani(N150364)

ACKNOWLEDGEMENT

We would like to express our profound gratitude and deep regards to our guide Mr. Krishna Kumar Singh for his exemplary guidance, monitoring and constant encouragement throughout the course of the thesis.

At this juncture wee feel deeply honoured in expressing our sincere thanks to him for making the resources available at right time and providing valuable insights leading to the successful completion of our project.

We would lik to thank RGUKT Nuzvid Director, faculty and staff for their valuable suggestions and discussions.

P.N.J.Sirisha(N150838)

V. Srilaxmi(N150579)

Sk. Sameera(N150410)

J. Sravani(N150364)

Table of contents...

Chapters Page Number

1.ABSTRACT

Human ideas, tastes, opinions in every sector change day by day. As we are living in a digital world, we always fulfil our needs through mobiles with the help of internet. Making this as a motive, we have designed a Mobile application using React Native. It is a cross-platform so it can be accessible in each and every platform. This mobile application mainly focuses on shopping. We can select products according to our wish just with one click.

2.INTRODUCTION

1.1 Purpose:

The main purpose of our project is to create a Mobile Application using React Native regarding Shopping Which Attracts who usually prefer online Shopping

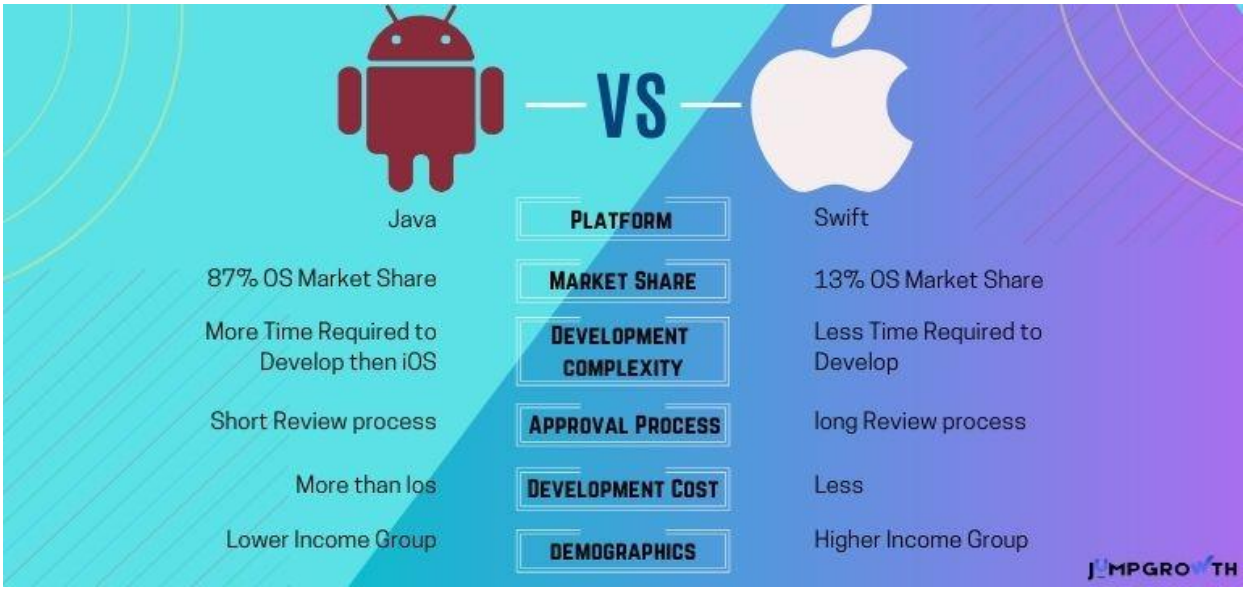
1.2 Scope:

As we prefer easy methods in this current situation, So this application to place the order according to our wish.

1.3 Overview:

Through this document we will get an idea regarding several technologies like React Native, Firebase, NoSQL , JSON .

3. DESCRIPTION



Apps are mainly two types, Android apps and iOS apps. Android apps are again divided into three types.

There are three basic types of mobile apps if we categorize them by the technology used to code them:

- Native apps : Created for one specific platform or operating system.
- Web apps : These apps are responsive versions of websites that can work on any mobile device or OS because they're delivered using a mobile browser.
- Hybrid apps : These apps are combinations of both native and web apps, but wrapped within a native app, giving it the ability to have its own icon or be downloaded from an app store.

2.1 Native Apps:

Native apps are built specifically for a mobile device's operating system (OS). Thus, you can have native Android mobile apps or native iOS apps, not to mention all the other platforms and devices.

Because they're built for just one platform, you cannot mix and match – say, use a Blackberry app on an Android phone or use an iOS app on a Windows phone.

Technology Used: Native apps are coded using a variety of programming languages. Some examples include: Java, Kotlin, Python, Swift, Objective-C, C++, and React.

Pros: Because of their singular focus, native apps have the advantage of being faster and more reliable in terms of performance. They're generally more efficient with the device's resources than other types of mobile apps. Native apps utilize the native device UI, giving users a more optimized customer experience. And because native apps connect with the device's hardware directly, they have access to a broad choice of device features like Bluetooth, phonebook contacts, camera roll, NFC, and more.

Cons: However, the problem with native apps lies in the fact that if you start developing them, you have to duplicate efforts for each of the different platforms. The code you create for one platform cannot be reused on another. This drives up costs. Not to mention the effort needed to maintain and update the codebase for each version.

And then, every time there's an update to the app, the user has to download the new file and reinstall it. This also means that native apps do take up precious space in the device's storage.

2.2 Web Apps:

Web apps behave similarly to native apps but are accessed via a web browser on your mobile device. They're not standalone apps in the sense of having to download and install code into your device. They're actually responsive websites that adapt its user interface to the device the user is on. In fact, when you come across the option to "install" a web app, it often simply bookmarks the website URL on your device.

One kind of web app is the progressive web app (PWA), which is basically a native app running inside a browser.

Technology Used: Web apps are designed using HTML5, CSS, JavaScript, Ruby, and similar programming languages used for web work.

Pros: Because it's web-based, there is no need to customize to a platform or OS. This cuts down on development costs. Plus, there's nothing to download. They won't take up space on your device memory like a native app, making maintenance easier – just push the update live over the web. Users don't need to download the update at the app store.

Cons: But this is also pertinent: web apps are entirely dependent on the browser used on the device. There will be functionalities available within one browser and not available on another, possibly giving users varying experiences.

And because they're shells for websites, they won't completely work offline. Even if they have an offline mode, the device will still need an internet connection to back up the data on your device, offer up any new data, or refresh what's on screen.

2.3 Hybrid Apps:

And then there are the hybrid apps. These are web apps that look and feel like native apps. They might have a home screen app icon, responsive design, fast performance, even be able to function offline, but they're really web apps made to look native.

Technology Used: Hybrid apps use a mixture of web technologies and native APIs. They're developed using: Ionic, Objective C, Swift, HTML5, and others.

Pros: Building a hybrid app is much quicker and more economical than a native app. As such, a hybrid app can be the minimum viable product – a way to prove the viability of building a native app. They

also load rapidly, are ideal for usage in countries with slower internet connections, and give users a consistent user experience. Finally, because they use a single code base, there is much less code to maintain.

Cons: Hybrid apps might lack in power and speed, which are hallmarks of native apps.

4.React

4.1 React:

React is a framework for building applications using JavaScript. React Native is an entire platform allowing you to build native, cross-platform mobile apps, and React.js is a JavaScript library you use for constructing a high performing UI layer. React.js is the heart of React Native, and it embodies all React's principles and syntax, so the learning curve is easy. The platform is what gave rise to their technical differences. Like the browser code in React is rendered through Virtual DOM while React Native uses Native API's to render components on mobile. React uses HTML and with React Native, you need to familiarize yourself with React Native syntax. React Native doesn't use CSS either. This means you'll have to use the animated API which comes with React Native to animate different components of your application.

4.2 React Native:

React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It's based on React, Facebook's JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly "native," all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS.

Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque markup, known as JSX. Then, under the hood, the React Native "bridge" invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, *not* web views, and will look and feel like any other

mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user's location.

React Native currently supports both iOS and Android, and has the potential to expand to future platforms as well. In this book, we'll cover both iOS and Android. The vast majority of the code we write will be cross-platform.

Advantages of React Native:

The fact that React Native actually renders using its host platform's standard rendering APIs enables it to stand out from most existing methods of cross-platform application development, like Cordova or Ionic. Existing methods of writing mobile applications using combinations of JavaScript, HTML, and CSS typically render using webviews. While this approach can work, it also comes with drawbacks, especially around performance. Additionally, they do not usually have access to the host platform's set of native UI elements. When these frameworks do try to mimic native UI elements, the results usually "feel" just a little off; reverse-engineering all the fine details of things like animations takes an enormous amount of effort, and they can quickly become out of date.

In contrast, React Native actually translates your markup to real, native UI elements, leveraging existing means of rendering views on whatever platform you are working with. Additionally, React works separately from the main UI thread, so your application can maintain high performance without sacrificing capability. The update cycle in React Native is the same as in React: when props or state change, React Native re-renders the views. The major difference between React Native and React in the browser is that React Native does this by leveraging the UI libraries of its host platform, rather than using HTML and CSS markup.

4.3.Firebase:

Firebase is a mobile and web app development platform that provides developers with a plethora of tools and services to help them develop high-quality apps, grow their user base, and earn more profit.

Realtime Database:

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync between your users in realtime. The Realtime Database is really just one big JSON object that the developers can manage in realtime.

With just a single API, the Firebase database provides your app with both the current value of the data and any updates to that data.

Real-time syncing makes it easy for your users to access their data from any device, be it web or mobile. Real-time Database also helps your users collaborate with one another.

Another amazing benefit of Real-time Database is that it ships with mobile and web SDKs, allowing you to build your apps without the need for servers.

When your users go offline, the Real-time Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.

The Real-time Database can also integrate with Firebase Authentication to provide a simple and intuitive authentication process.

4.4. Design constraints:

Software constraints:

React Native, NoSql, JSON, Android Studio

Hardware constraints:

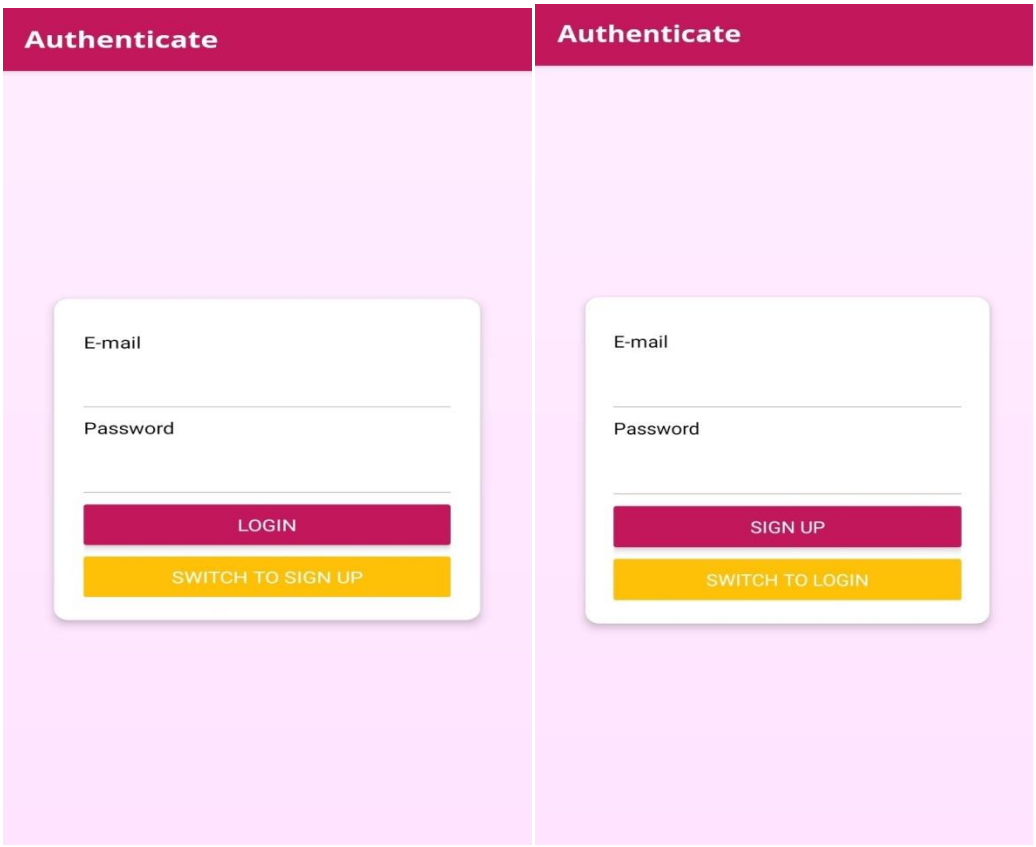
RAM: 8GB RAM

Processor: Intel(R) Core (TM) i5.2.50 GHz

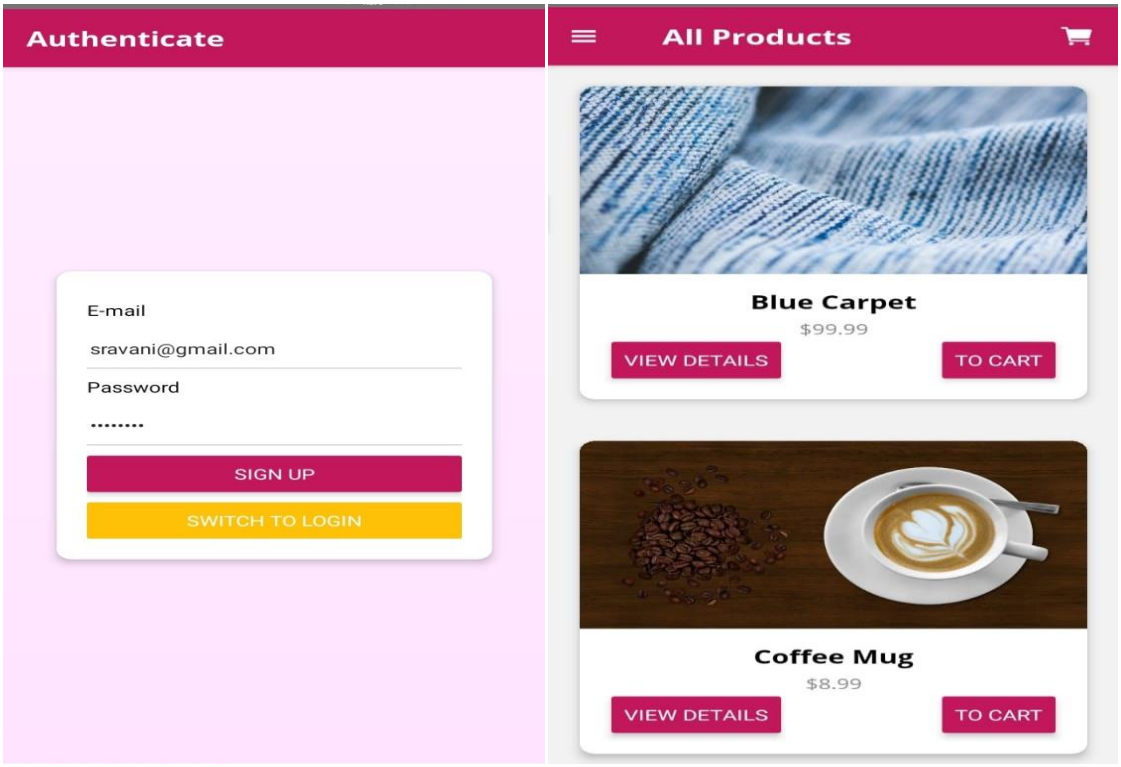
Operating System : Windows 7/8/10,Microsoft

System Type : 64-bit operating system

5.DESIGN OVERVIEW

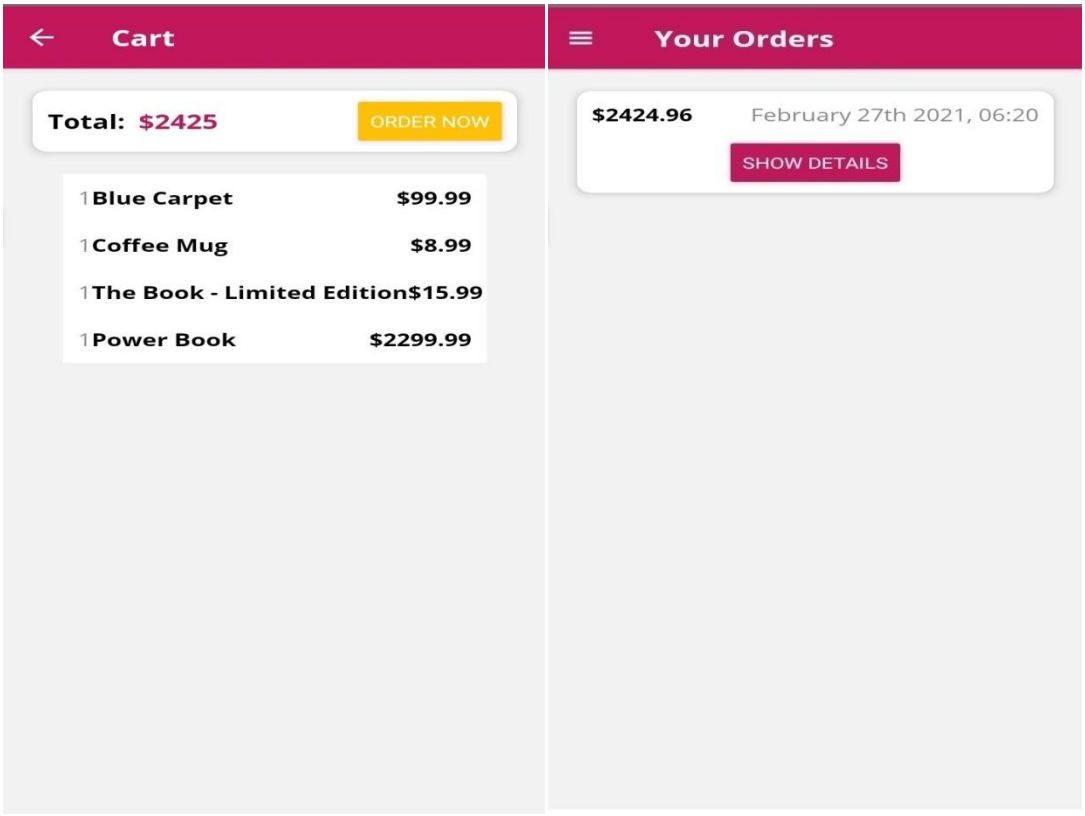


The first interface of the mobile application displays the Authentication. Authentication deals with sign in and login in details.Here we need to enter our credentials like Mail id and need to create our own password.These details are entered for accessing the app permission. If you don't logout from your account after using the mobile application, there is no need to enter the credentials foe thr next time.If you log out from your account, you need to enter the required details for successful login.

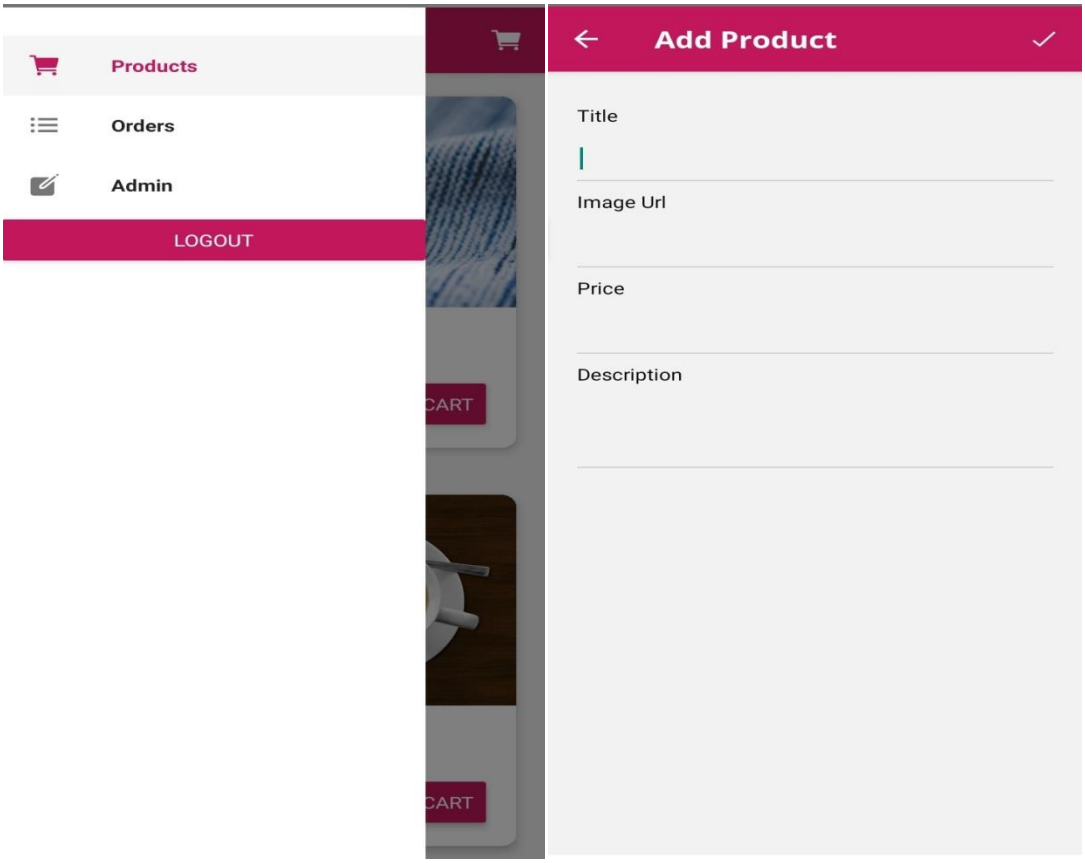


These images gives an idea what happens in the next interface.When you login into to your acount, it directs you to the page,where you can see all kind's of products namely shirts, tops,watches, scarfs and jewellery. You can see the cost,description and model of a particular product by clicking View details button. It gives a clear idea about product which you choose to buy based on yor intrest.There is another option named Add to cart. This helps you to save the group products,so that you can buy all those items at a time. You can delete the items in the cart, if you are not interested to buy them

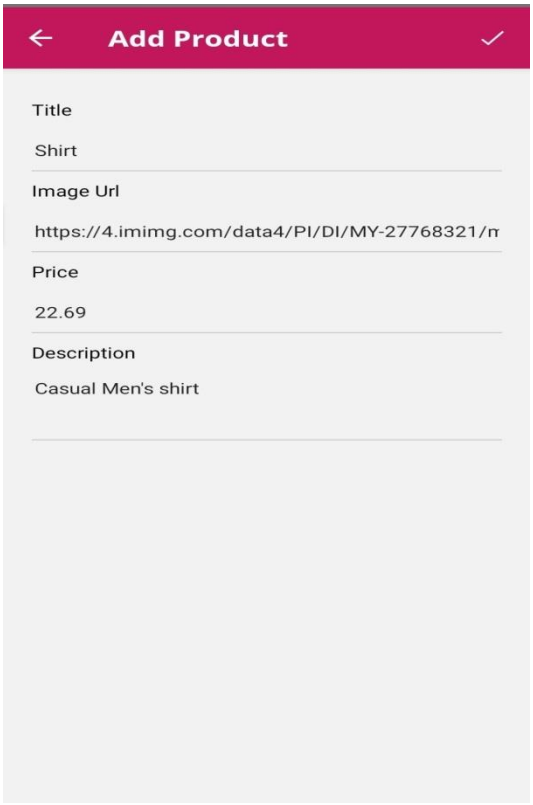
.



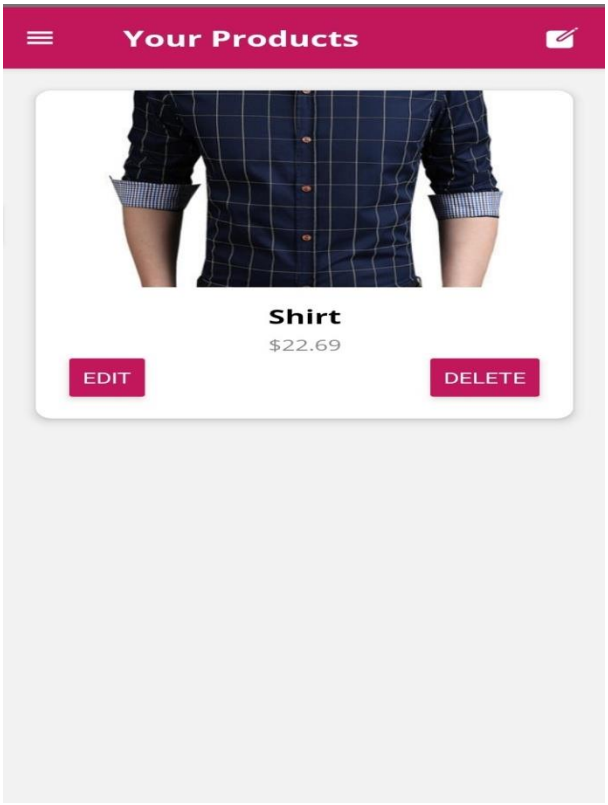
These images shows the total amount of products,which you have added to the cart. We can check the items that we have added to the cart anytime. It also shows the details of the items that we have added to the cart. In future, we can place the orders of the items that are added to the cart.



This layout shows how to add products by admin. While adding a product we have some specifications like adding the title of an item, and url of the image, price of the image and description of the image. Descrciption refers to the type and model of the particular item. Here we are taking the price in dollar form.



It shows errors when any of these details are entered incorrectly. After adding all the item details the result is as shown as follows



This image appears after adding a new product successfully.

6.CODE IMPLEMENTATION

CartScreen.js

```
import React,{useState} from 'react';
import {View,Text,Button,StyleSheet,FlatList,ActivityIndicator} from 'react-native';
import {useSelector, useDispatch} from 'react-redux';

import Colors from '../../constants/Colors';
import CartItem from '../../components/shop/CartItem';
import * as cartActions from '../../store/actions/cart';
import * as orderActions from '../../store/actions/orders';
import Card from '../../components/UI/Card'

const CartScreen = props => {
    const [isLoading,setIsLoading] = useState(false)
    const cartTotalAmount = useSelector(state => state.cart.totalAmount);
    console.log("@total",cartTotalAmount)
    const cartItems = useSelector(
        state => {
            const transformedCartIems = [];
            for(const key in state.cart.items){
                transformedCartIems.push({
                    productId: key,
                    productTitle: state.cart.items[key].productTitle,
                    productPrice: state.cart.items[key].productPrice,
                    quantity: state.cart.items[key].quantity,
                    sum: state.cart.items[key].sum
                })
            }
            return transformedCartIems.sort((a,b) =>
                a.productId > b.productId ? 1 : -1);
        }
    )
    const sendOrderHandler = async () => {
        setIsLoading(true)
       await dispatch(orderActions.addOrder(cartItems,cartTotalAmount))
        setIsLoading(false)
    }
    const dispatch = useDispatch()
    return(
        <View style={styles.screen}>
            <Card style={styles.summary}>
                <Text style={styles.summaryText}>
                    Total: {' '}
                    <Text style={styles.amount}>
                    ${Math.round(cartTotalAmount.toFixed(2) * 100 /100 )}</Text>
                </Text>
```

```
                {isLoading ? <ActivityIndicator size='small' color={Colors.primary}/> :
                <Button
                color={Colors.accent}
                title="Order Now"
                disabled={cartItems.length === 0}
                onPress={sendOrderHandler}
                />
                }
            </Card>
            <FlatList
                data={cartItems}
                keyExtractor={item => item.productId}
                renderItem={itemData => (
                    <CartItem
                        quantity= {itemData.item.quantity}
                        title={itemData.item.productTitle}
                        amount={itemData.item.sum}
                        onRemove={() => {dispatch(cartActions.removeFromCart(itemData.item.productId))}}/>
                )}/>
        </View>
    )
}

const styles = StyleSheet.create({
    screen: {
        margin: 20
    },
    summary: {
        flexDirection: 'row',
        alignItems: 'center',
        justifyContent: 'space-between',
        marginBottom: 20,
        padding: 10,
    },
    summaryText: {
        fontFamily: 'OpenSans-Bold',
        fontSize: 18
    },
    amount: {
        color: Colors.primary
    }
})

export default CartScreen;
```

Orders Screen.js

```
import React,{useState,useEffect} from 'react';
import {View,ActivityIndicator,FlatList,Text,Platform,StyleSheet} from 'react-native';
import {useSelector,useDispatch} from 'react-redux';
import {HeaderButtons,Item} from 'react-navigation-header-buttons';
import HeaderButton from '../../components/UI/HeaderButton';
import OrderItem from '../../components/shop/OrderItem';
import * as orderActions from '../../store/actions/orders';
import Colors from '../../constants/Colors';

const OrdersScreen = props => {
    const [isLoading,setIsLoading] = useState(false);
    const orders = useSelector(state => state.orders.orders);
    console.log(orders)
    const dispatch = useDispatch()
    useEffect(() => {
        setIsLoading(true)
        dispatch(orderActions.fetchOrders()).then(() => {
        setIsLoading(false)
        })
    },[dispatch])

    if(isLoading){
        return(
            <View style={styles.centered}>
                <ActivityIndicator size="large" color={Colors.primary}/>
            </View>
        )
    }
    if(orders.length === 0){
        return(
        <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
            <Text>No order found, maybe start ordering some products?</Text>
      </View>
        )
    }
    return(
        <FlatList
            data = {orders}
            keyExtractor={item => item.id}
            renderItem={itemData => (
                <OrderItem
                    amount = {itemData.item.totalAmount}
                    date = {itemData.item.readableDate}
```

```
                    items = {itemData.item.items}
                    />
            )}/>
    )
}

OrdersScreen.navigationOptions = navData => {
    return{
        headerTitle: 'Your Orders',
        headerLeft: (
            <HeaderButtons HeaderButtonComponent={HeaderButton}>
                <Item
                    title="Menu"
                    iconName='menu'
                    onPress={() => {
                        navData.navigation.toggleDrawer()
                    }}
                    />
            </HeaderButtons>
        )
    }
}

const styles = StyleSheet.create({
    centered: {
        flex: 1,
        alignItems: 'center',
        justifyContent: 'center'
    }
})

export default OrdersScreen;
```

ProductDetailsScreen.js

```
import React from 'react';
import {
    ScrollView,
    View,
    Text,
    Button,
    Image,
    StyleSheet
} from 'react-native';
import {useSelector,useDispatch} from 'react-redux';
import Colors from '../../constants/Colors';
import * as cartActions from '../../store/actions/cart';


const ProductDetailScreen = props => {
    const productId = props.navigation.getParam('productId')
    const selectedProduct = useSelector(state =>
        state.products.availableProducts.find(prod => prod.id === productId))
    const dispatch = useDispatch()
    return(
        <View>
            <Image style={styles.image} source={{uri: selectedProduct.imageUrl}}/>
            <View style={styles.actions}>
                <Button
                    style= {styles.actions}
                    color={Colors.primary}
                    title="Add to Cart"
                    onPress={() => {
                        dispatch(cartActions.addToCart(selectedProduct))
                    }}/>
            </View>
            <Text style={styles.price}>${selectedProduct.price}</Text>
            <Text style={styles.description}>{selectedProduct.description}</Text>
        </View>
    )
}

ProductDetailScreen.navigationOptions = navData => {
    return{
        headerTitle: navData.navigation.getParam('productTitle')
    }
}

const styles = StyleSheet.create({
```

```
    image: {
        width: '100%',
        height: 300
    },
    actions: {
        marginVertical: 10,
        alignItems: 'center'
    },
    price: {
        fontFamily: 'OpenSans-Bold',
        fontSize: 20,
        color: '#888',
        alignSelf: 'center',
        marginVertical: 20
    },
    description: {
        fontFamily: 'OpenSans-Regular',
        fontSize: 14,
        textAlign: 'center',
        marginHorizontal: 20
    },

})

export default ProductDetailScreen;
```

ProductOverviewScreen.js

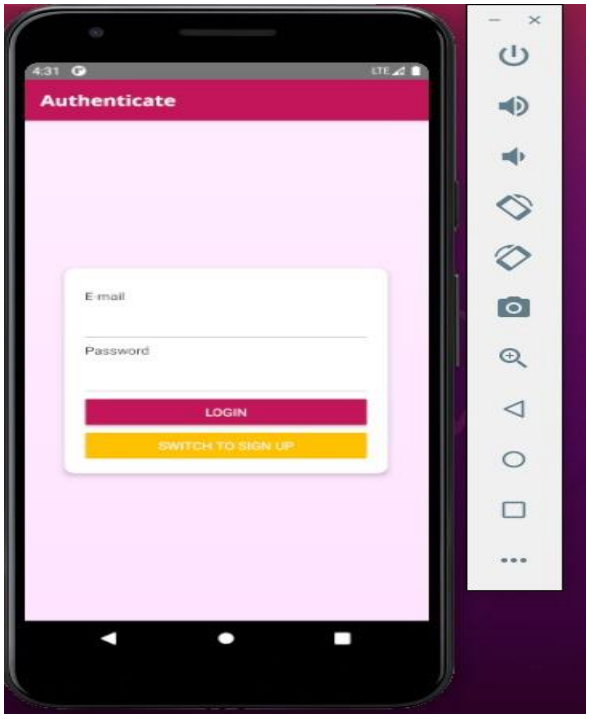
```
import React,{useState,useEffect,useCallback} from 'react';
import {View,
        FlatList,
        Platform,
        ActivityIndicator,
        StyleSheet,
        Text,
        Button} from 'react-native';
import {useSelector,useDispatch} from 'react-redux';

import ProductItem from '../../components/shop/ProductItem';
import {HeaderButtons, Item} from 'react-navigation-header-buttons';
import * as cartActions from '../../store/actions/cart';
import * as productActions from '../../store/actions/products';
import HeaderButton from '../../components/UI/HeaderButton';

import Colors from '../../constants/Colors';


const ProductsOverviewScreen = props => {

    const [isLoading,setIsLoading] = useState(false)
    const [isRefreshing,setIsRefreshing] = useState(false)
    const [error,setError] = useState(false)
    const products = useSelector(state => state.products.availableProducts);
    const dispatch = useDispatch();

    const loadProducts = useCallback(async () => {
      setError(null)
      setIsRefreshing(true)
      try {
        await dispatch(productActions.fetchProducts())
      }
      catch(err){
        setError(err.message)
      }
      setIsRefreshing(false)
    },[dispatch,setIsLoading,setError])

    useEffect(() => {
      const willFocusSub = props.navigation.addListener(
        'willFocus',
        loadProducts
      )
```

```
45          return () => {
46            willFocusSub.remove()
47          }
48        },[loadProducts])
49
50
51        useEffect(() => {
52          setIsLoading(true)
53          loadProducts().then(() => {
54            setIsLoading(false)
55          })
56        },[dispatch,loadProducts])
57
58        const selectItemHandler = (id,title) => {
59          props.navigation.navigate('ProductDetail',{
60            productId: id,
61            productTitle: title
62          })
63        }
64
65        if(error){
66          return(
67            <View style={styles.centered}>
68              <Text>An error occurred</Text>
69              <Button
70                title="Try again"
71                color={Colors.primary}
72                onPress={loadProducts}/>
73            </View>
74          )
75        }
76
77        if(isLoading) {
78          return (
79            <View style={styles.centered}>
80              <ActivityIndicator size="large" color={Colors.primary}/>
81            </View>
82          )
83        }
84
85        if(!isLoading && products.length === 0){
86          return(
87            <View style={styles.centered}>
```

```
89          </View>
90        )
91      }
92      return(
93          <FlatList
94              onRefresh={loadProducts}
95              refreshing={isRefreshing}
96              data={products}
97              keyExtractor={item => item.id}
98              renderItem={itemData => (
99              <ProductItem
100                 image={itemData.item.imageUrl}
101                 title={itemData.item.title}
102                 price={itemData.item.price}
103                 onSelect={() => {
104                    selectItemHandler(itemData.item.id,
105                     itemData.item.title)
106                 }}
107                 >
108                   <Button
109                     title="View Details"
110                     color={Colors.primary}
111                     onPress={() => {
112                       selectItemHandler(itemData.item.id,
113                         itemData.item.title)
114                     }}
115                     />
116                     <Button
117                       title="To Cart"
118                       color={Colors.primary}
119                       onPress={() => {
120                         dispatch(cartActions.addToCart(itemData.item))
121                       }}
122                       />
123             </ProductItem>
124         )}
125         />
126     )
127 }
128
129
130 ProductsOverviewScreen.navigationOptions = navData => {
131     return {
132       headerTitle: 'All Products',
```

```
133         headerLeft: (
134           <HeaderButtons HeaderButtonComponent={HeaderButton}>
135               <Item
136                 title="Menu"
137                 iconName="menu"
138                 onPress={() => {
139                   navData.navigation.toggleDrawer()
140                 }}/>
141           </HeaderButtons>
142         ),
143         headerRight: (
144           <HeaderButtons HeaderButtonComponent={HeaderButton}>
145             <Item
146               title="Cart"
147               iconName={Platform.OS === 'android' ? 'md-cart' : 'ios-cart'}
148               onPress={() => {
149                   navData.navigation.navigate('Cart')
150               }}
151             />
152           </HeaderButtons>
153         )
154       };
155     };
156
157   const styles = StyleSheet.create({
158     centered: {
159       flex: 1,
160       alignItems: 'center',
161       justifyContent: 'center'
162     },
163
164   })
165
166   export default ProductsOverviewScreen;
```

OUTPUT :



7.CONCLUSION

While years passes, ideas and opinions also change. We always prefer easy methods to survive. As we are living in a digital world , we always fulfil our needs through mobiles with the help of internet. Making this as a motive, we have designed a Mobile application using React Native. It is a cross-platform so it can be accessible in each and every platform. This mobile application mainly focuses on shopping. We can select products according to our wish just with one click.

8.REFERENCES

For further information, refer to the following websites.

https://reactnative.dev/

https://hackernoon.com/introduction-to-firebase-218a23186cd7

https://www.couchbase.com/resources/why-nosql