Full Stack Mobile Application: IRIS(Rental and Selling Platform)

As this is a product was developed under GITSS PVT.LTD where we did our internship, we are restricted to add the code. We are adding the sample code snippets of some particular screens we have worked on for evaluation from our university.

We have also informed and taken permission from our Project guide- Prof. Janardhan Singh and Project Coordinator - Prof. Sandeep Kumar on the same.

Team members:
Mr. Mukund B(1CD18CS078)
Mr. M Arshad(1CD18CS076)
Mr. Varun B(1CD18CS025)
Mr. Nayan BA(1CD18CS086)
Mr. N Sudeep(1CD18CS082)
Mr. Syed Shoaib(1CD18CS183)

LOGIN SCREEN

```
import React, {useState} from 'react';
import {
  View,
  StyleSheet,
  Text,
  TextInput,
  TouchableOpacity,
} from 'react-native';
import {useLogin} from '_pages/login/LoginProvider';
import {SUCCESS_RESPONSE} from '_utils/constants';
```

```
import {postCall} from '_components/backendApiCall';
import {isValidPhoneNumber, isValidObjField, updateError} from './methods';
import {BASE_IP_OF_LOGIN_SIGNUP, LOGIN} from '_utils/api';
import {connect} from 'react-redux';
import {bindActionCreators} from 'redux';
import FormContainer from './FormContainer';
import FormInput from './FormInput';
import FormSubmitButton from './FormSubmitButton';
import * as Keychain from 'react-native-keychain';
import {storeTokens} from '_utils/commonFunctions';
import translate from '../../utils/language.utils';
const updateAccessToken=token =>(
  {
    type:"UPDATE_ACCESS_TOKEN",
    payload:token,
  }
);

const LoginForm = ({navigation}) => {
  const {setIsLoggedIn, setProfile} = useLogin();
  const [userInfo, setUserInfo] = useState({
    phone: '',
    password: '',
  });

  const [error, setError] = useState('');
  const [submitted,setSubmitted]=useState(false);
  const {phone, password} = userInfo;

  const handleOnChangeText = (value, fieldName) => {
    setUserInfo({...userInfo, [fieldName]: value});
  };

  const isValidForm = () => {
    if (!isValidObjField(userInfo))
      return updateError(translate('LOGINSCREEN_requireallFields'), setError);

    if (!isValidPhoneNumber(phone.trim()))
      return updateError(translate('LOGINSCREEN_phoneNumber'), setError);

    if (!password.trim() || password.length < 6)
      return updateError(translate('LOGINSCREEN_passwordisShort'), setError);

    return true;
  };
```

```
  const submitForm = async () => {
    setSubmitted(true);
    if (isValidForm()) {
      var res = await postCall(
        BASE_IP_OF_LOGIN_SIGNUP,
        LOGIN,
        {},
        {
          mobileNumber: userInfo.phone,
          password: userInfo.password,
        },
        false
      );
      if (res.status == SUCCESS_RESPONSE) {

        var response = res.value;
        setIsLoggedIn(storeTokens(response.data.username,response.data.refreshTok
en,response.data.type,response.data.token));
      }
      else{
        console.log(res.value);
      }
    }
    setSubmitted(false);
  };

  return (
    <FormContainer>
      {error ? (
        <Text style={{color: 'red', fontSize: 18, textAlign: 'center'}}>
          {error}
        </Text>
      ) : null}
      <FormInput
        value={phone}
        onChangeText={value => handleOnChangeText(value, 'phone')}
        label={translate('LOGINSCREEN_phoneNumber')}
        placeholder={'86******51'}
        autoCapitalize="none"
        keyboardType="numeric"
        maxLength={10}
      />
      <FormInput
        value={password}
```

```
        onChangeText={value => handleOnChangeText(value, 'password')}
        label={translate('LOGINSCREEN_password')}
        placeholder={'******'}
        autoCapitalize="none"
      />
      <FormSubmitButton
        onPress={submitForm}
        title={translate('LOGINSCREEN_Login')}
        submitting={submitted}
      />
      <View style={styles.forgotContainer}>
        <TouchableOpacity
          onPress={() => {
            navigation.navigate('ForgotPassword');
          }}>
          <Text style={styles.forgot}>
            {translate('LOGINSCREEN_forgotPassword')}
          </Text>
        </TouchableOpacity>
      </View>
    </FormContainer>
  );
};
import React, {useRef, useState} from 'react';
import {
  ScrollView,
  StyleSheet,
  View,
  Animated,
  Dimensions,
  TouchableWithoutFeedback,
} from 'react-native';
import colors from '_utils/Colors';
import FormHeader from './FormHeader';
import FormSelectorBtn from './FormSelectorBtn';
import SignupForm from './SignupForm';
import LoginForm from './LoginForm';
import {Keyboard} from 'react-native';
import translate from '../../utils/language.utils';

const {width} = Dimensions.get('window');

export default function Login({navigation}) {
  const animation = useRef(new Animated.Value(0)).current;
  const scrollView = useRef();
```

```
const rightHeaderOpacity = animation.interpolate({
  inputRange: [0, width],
  outputRange: [1, 0],
});

const leftHeaderTranslateX = animation.interpolate({
  inputRange: [0, width],
  outputRange: [0, 40],
});
const rightHeaderTranslateY = animation.interpolate({
  inputRange: [0, width],
  outputRange: [0, -20],
});
const loginColorInterpolate = animation.interpolate({
  inputRange: [0, width],
  outputRange: [colors.themeColor, colors.themeColor + '70'],
});
const signupColorInterpolate = animation.interpolate({
  inputRange: [0, width],
  outputRange: [colors.themeColor + '70', colors.themeColor],
});

return (
  <TouchableWithoutFeedback
    style={{flex: 1}}
    onPress={Keyboard.dismiss}
    accessible={false}>
    <View style={{flex: 1, paddingTop: 120}}>
      <View style={{height: 80}}>
        <FormHeader
          leftHeading={translate('LOGINSCREEN_Welcome')}
          rightHeading={translate('LOGINSCREEN_Back')}
          subHeading={translate('LOGINSCREEN_toIris')}
          rightHeaderOpacity={rightHeaderOpacity}
          leftHeaderTranslateX={leftHeaderTranslateX}
          rightHeaderTranslateY={rightHeaderTranslateY}
        />
      </View>
      <View
        style={{
          flexDirection: 'row',
          paddingHorizontal: 20,
          marginBottom: 20,
        }}>
```

```
          <FormSelectorBtn
            style={styles.borderLeft}
            backgroundColor={loginColorInterpolate}
            title={translate('LOGINSCREEN_Login')}
            onPress={() => scrollView.current.scrollTo({x: 0})}
          />
          <FormSelectorBtn
            style={styles.borderRight}
            backgroundColor={signupColorInterpolate}
            title={translate('LOGINSCREEN_signUp')}
            onPress={() => scrollView.current.scrollTo({x: width})}
          />
        </View>
        <ScrollView
          ref={scrollView}
          horizontal
          pagingEnabled
          showsHorizontalScrollIndicator={false}
          scrollEventThrottle={16}
          onScroll={Animated.event(
            [{nativeEvent: {contentOffset: {x: animation}}}],
            {useNativeDriver: false},
          )}>
          <ScrollView>
            <LoginForm navigation={navigation} />
          </ScrollView>
          <ScrollView>
            <SignupForm navigation={navigation} />
          </ScrollView>
        </ScrollView>
      </View>
    </TouchableWithoutFeedback>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  borderLeft: {
    borderTopLeftRadius: 8,
    borderBottomLeftRadius: 8,
```

```
  },
  borderRight: {
    borderTopRightRadius: 8,
    borderBottomRightRadius: 8,
  },
});
```

My address

```
import React, {useState, useEffect} from 'react';
import {View, Text, FlatList, TouchableOpacity,Alert} from 'react-native';
import {Icon} from 'react-native-elements';
import colors from '_utils/Colors';
import axios from 'react-native-axios';
import {SUCCESS_RESPONSE} from '_utils/constants';
import {getCall,postCall} from '_components/backendApiCall';
import {BASE_IP_OF_MY_ACCOUNT, GET_ADDRESSES,DELETE_ADDRESS} from '_utils/api';
import SkeletonPlaceholder from 'react-native-skeleton-placeholder';
import Skeleton from './skeleton';

const MyAddress = props => {
  const [isLoading, setIsLoading] = useState(true);
  const [addresses, setAddresses] = useState([]);
  useEffect(() => {
    fetchData();
    const unsubscribe = props.navigation.addListener('focus', () => {
      fetchData();
    });
    return unsubscribe;
  }, [props.navigation]);

  const fetchData = async () => {
    var res = await getCall(BASE_IP_OF_MY_ACCOUNT,GET_ADDRESSES,{});
      if(res.status == SUCCESS_RESPONSE)
      {
        var response=res.value;
        setAddresses(response.data);
        setIsLoading(false);
      }else{
        console.log(res.value);
```

```
      };
  };
  const deleteAddress = async (id) => {
    var res = await
postCall(BASE_IP_OF_MY_ACCOUNT,DELETE_ADDRESS,{},{addressId:id});
      if(res.status == SUCCESS_RESPONSE)
      {
        setIsLoading(true);
        fetchData();
      }else {
        console.log(res.value);
      };
  };
  const locality = ({item}) => {
    switch (item.addressLabel) {
      case 'Home':
        var icon = 'home';
        var type = 'antdesign';
        break;
      case 'Work':
        var  icon  =  'suitcase';
        var type = 'font-awesome';
        break;
      default:
        var icon = 'location-sharp';
        var type = 'ionicon';
    }
    return (
      <View
        style={{
          backgroundColor: '#fff',
          borderRadius: 10,
          elevation: 2,
          paddingHorizontal: '2%',
          margin: '4%',
          paddingVertical: '4%',
        }}>
        <View style={{flexDirection: 'row'}}>
          <Icon
            name={icon}
            iconStyle={{paddingHorizontal: '3%', color: colors.primaryColor}}
            type={type}
            size={20}
          />
          <Text style={{fontWeight: 'bold', fontSize: 16, paddingBottom: '2%'}}>
```

```
            {item.addressLabel == 'Friends and Family'
              ? item.name
              : item.addressLabel}
          </Text>
        </View>
        <Text style={{paddingLeft: '12%'}}>
          {item.addressLineOne}
          {'\n'}
          {(item.addressLineTwo)?item.addressLineTwo+'\n':null}
          {item.location.address}
          {'\n'}
          Phone:{item.phoneNumber}
        </Text>

        <View
          style={{
            flexDirection: 'row',
            justifyContent: 'space-around',
            paddingTop: '2%',
          }}>
          <TouchableOpacity
            onPress={() => {
              props.navigation.navigate('commonScreens', {
                screen: 'map',
                params: {from: 'MyAddress', operation: 'edit', address: item},
              });
            }}>
            <Text style={{color: colors.primaryColor, fontWeight: 'bold'}}>
              EDIT
            </Text>
          </TouchableOpacity>
          <TouchableOpacity onPress={()=>{
            Alert.alert("Are you sure?","This address will be deleted
permanently",[{
              text:"Cancel",
              onPress:()=>{}
            },
            {
              text:"Ok",
              onPress:()=>{deleteAddress(item.addressId)}
            }
          ])
          }}>
            <Text style={{color: colors.primaryColor, fontWeight: 'bold'}}>
              DELETE
```

```
            </Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  };
```

```
```

HOME PAGE

```
import React from 'react';
import {View, Text, FlatList, TouchableOpacity} from 'react-native';
import {Icon} from 'react-native-elements';
import colors from '_utils/Colors';
import category from '../categories/categoriesData';
import translate from '../../utils/language.utils';

const CategoryList = props => {
  const Cards = ({item}) => {
    return (
      <TouchableOpacity
        style={{marginVertical: 5, alignItems: 'center'}}
        onPress={() => {
            paddingHorizontal:'2%',
            fontSize: 14,
            color: colors.black,
            fontWeight: 'bold',
            backgroundColor: colors.themeColor,
            borderRadius: 4,
          }}>
          {translate('HOMESCREEN_viewAll')}
        </Text>
      </View>
      <FlatList
        horizontal
        data={category}
        keyExtractor={item => item.key}
        renderItem={Cards}
        showsHorizontalScrollIndicator={false}
      />
    </View>
  );
};
```

```
import React, {useEffect, useState} from 'react';
import {
  View,
  StyleSheet,
  Text,
  FlatList,
  TouchableOpacity,
  Image,
  ScrollView,
  Dimensions,
  Alert,
  StatusBar,
} from 'react-native';
import Header from './header';
import colors from '_utils/Colors';
import CategoryList from './categoryList';
import category from '../categories/categoriesData';
import {requestLocationPermission} from '_components/permissions';
import NetInfo from '@react-native-community/netinfo';
import axios from 'react-native-axios';
import {connect} from 'react-redux';
import {SUCCESS_RESPONSE} from '_utils/constants';
import {postCall} from '_components/backendApiCall';
import {Icon} from 'react-native-elements';
import Swiper from 'react-native-swiper';
import {
  BASE_IP_OF_HOST,
  GET_HOME_PAGE_PRODUCTS,
  BASE_IP_OF_IMAGE_SERVER,
  GET_IMAGES,
} from '_utils/api';
import {RecyclerListView, DataProvider, LayoutProvider} from 'recyclerlistview';
import translate from '../../utils/language.utils';
import SkeletonPlaceholder from 'react-native-skeleton-placeholder';
import header from './header';

const HeaderInnov = props => {
  const [homePageData, setHomePageData] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const bufferViewCount = [1, 2, 3, 4, 5, 6];
  const {height, width} = Dimensions.get('window');
  const [dataProvider, setDataProvider] = useState({
    ELECTRONICS: new DataProvider((r1, r2) => r1 !== r2),
    SPORTS: new DataProvider((r1, r2) => r1 !== r2),
    FASHION: new DataProvider((r1, r2) => r1 !== r2),
```

```
    PROPERTIES: new DataProvider((r1, r2) => r1 !== r2),
    VEHICLES: new DataProvider((r1, r2) => r1 !== r2),
    FURNITURE: new DataProvider((r1, r2) => r1 !== r2),
    CONSTRUCTION: new DataProvider((r1, r2) => r1 !== r2),
    SOCIAL_GATHERING: new DataProvider((r1, r2) => r1 !== r2),
    GADGETS: new DataProvider((r1, r2) => r1 !== r2),
    OTHERS: new DataProvider((r1, r2) => r1 !== r2),
  });
  const [layoutProvider] = useState(
    new LayoutProvider(
      index => {
        return index;
      },
      (type, dim) => {
        dim.width = 165;
        dim.height = 220;
      },
    ),
  );

  useEffect(() => {
    fetchData();
    requestLocationPermission();
    const unsubscribe = NetInfo.addEventListener(state => {
      console.log('Connection type', state.type);

      console.log('Is connected?', state.isConnected);
      if (state.isConnected == false) {
        Alert.alert(
          translate('HOMESCREEN_no_internet'),
          translate('HOMESCREEN_make_sure_device_connected'),
        );
      }
    });
```

```
const categorySelector = key => {
    switch (key) {
      case 'ELECTRONICS':
        return dataProvider.ELECTRONICS;
      case 'SPORTS':
        return dataProvider.SPORTS;
      case 'FASHION':
        return dataProvider.FASHION;
      case 'PROPERTIES':
```

```
        return dataProvider.PROPERTIES;
      case 'VEHICLES':
        return dataProvider.VEHICLES;
      case 'FURNITURE':
        return dataProvider.FURNITURE;
      case 'CONSTRUCTION':
        return dataProvider.CONSTRUCTION;
      case 'SOCIAL_GATHERING':
        return dataProvider.SOCIAL_GATHERING;
      case 'GADGETS':
        return dataProvider.GADGETS;
      default:
        return dataProvider.OTHERS;
    }
  };

  const fetchData = async () => {
    console.log('requesting for home page data');
    var res = await postCall(
      BASE_IP_OF_HOST,
      GET_HOME_PAGE_PRODUCTS,
      {},
      {
        latitude: props.UserCurrentLocation.location.latitude,
        longitude: props.UserCurrentLocation.location.longitude,
      },
    );
    console.log(res);
    if (res.status == SUCCESS_RESPONSE) {
      var response = res.value;
      setHomePageData(response.data);
      setDataProvider({
        ELECTRONICS: dataProvider.ELECTRONICS.cloneWithRows(
          response.data.ELECTRONICS.results,
        ),
        SPORTS: dataProvider.SPORTS.cloneWithRows(response.data.SPORTS.results),
        FASHION: dataProvider.FASHION.cloneWithRows(
          response.data.FASHION.results,
        ),
        PROPERTIES: dataProvider.PROPERTIES.cloneWithRows(
          response.data.PROPERTIES.results,
        ),
        VEHICLES: dataProvider.VEHICLES.cloneWithRows(
          response.data.VEHICLES.results,
        ),
```

```
        FURNITURE: dataProvider.FURNITURE.cloneWithRows(
          response.data.FURNITURE.results,
        ),
        CONSTRUCTION: dataProvider.CONSTRUCTION.cloneWithRows(
          response.data.CONSTRUCTION.results,
        ),
        SOCIAL_GATHERING: dataProvider.SOCIAL_GATHERING.cloneWithRows(
          response.data.SOCIAL_GATHERING.results,
        ),
        GADGETS: dataProvider.GADGETS.cloneWithRows(
          response.data.GADGETS.results,
        ),
        OTHERS: dataProvider.OTHERS.cloneWithRows(response.data.OTHERS.results),
      });
      console.log(
        'data has been successfully recieved in Home Page',
        response.data,
      );
      setIsLoading(false);
    } else {
      console.log('Home Page Data Fetching Error:' + res.value);
    }
  };
```

```
import {connect} from 'react-redux';
import {bindActionCreators} from 'redux';
import SkeletonPlaceholder from 'react-native-skeleton-placeholder';
import Modal from 'react-native-modal';

const ProductScreen = props => {
  const [showCalender, setShowCalender] = useState({
    show: false,
    selectedStartDate: null,
    selectedEndDate: null,
  });
  const [toggleFavBtn, setToggleFavBtn] =
useState(props.wishList.filter((item)=>{return item.productId ==
props.route.params.productData.productId && item.productCategory ==
props.route.params.productData.prouctCategory}).length > 0);
  const [wishListUpdating,setWishListUpdating]=useState(false)
  const [product, setProduct] = useState({});
  const [similarProduct, setSimilarProduct] = useState([]);
  const [selectedValue, setSelectedValue] = useState(0);
  const bufferViewCount = [1, 2];
```

```
  const [isLoading, setIsLoading] = useState(true);
  const [imageCurrentIndex, setImageCurrentIndex] = useState(0);
  const [addresses, setAddresses] = useState([]);
  const [mobileNumber, setmobileNumber] = useState([]);
  const [buyNowvalue, buyNowsetValue] = useState(0);
  const [specialPriceValue, specialPricesetValue] = useState(0);
  // const specialPriceColors = ['#0760f0', '#f09a07'];

  const RatingColorSelector = rating => {
    switch (true) {
      case rating >= 3:
        return '#00a832';
      case rating >= 2:
        return '#f75f23';
      case rating >= 0:
        return '#ff0000';
      default:
        return '#00a832';
    }
  };
  const wishListHandler=async()=>{
    setWishListUpdating(true);
    if(props.wishList.filter((item)=>{return item.productId ==
props.route.params.productData.productId && item.productCategory ==
props.route.params.productData.prouctCategory}).length > 0){
      var res = await postCall(
        BASE_IP_OF_MY_ACCOUNT,
        DELETE_WISHLIST,
        {},
        {wishlistId: props.wishList.filter((item)=>{return item.productId ==
props.route.params.productData.productId && item.productCategory ==
props.route.params.productData.prouctCategory})[0].wishlistId},
      );
      if (res.status == SUCCESS_RESPONSE) {
        setToggleFavBtn(false);
        refreshWishList();
        Toast.show('Product removed from your Wishlist ×', Toast.LONG);
      } else {
        console.log(res.value);
      }
    }else{
      var res = await postCall(
        BASE_IP_OF_MY_ACCOUNT,
        ADD_WISHLIST,
        {},
```

```
        {productId: props.route.params.productData.productId, productCategory:
props.route.params.productData.prouctCategory},
      );
      if (res.status == SUCCESS_RESPONSE) {
        setToggleFavBtn(true);
        refreshWishList();
        Toast.show('Product added to your Wishlist ✓', Toast.LONG);
      } else {
        console.log(res.value);
      }
    }
    setWishListUpdating(false);
  }
  const ProductCard = ({item}) => {
    let ratingColor = '#00a832';
    switch (true) {
      case item.rating >= 3:
        ratingColor = '#00a832';
        break;
      case item.rating >= 2:
        ratingColor = '#f75f23';
        break;
      case item.rating >= 0:
        ratingColor = '#ff0000';
        break;
      default:
        ratingColor = '#00a832';
    }

    return (
      <TouchableOpacity
        onPress={() => {
          console.log({productId: item.id, prouctCategory: item.category});
          props.navigation.push('productScreen', {
            productData: {productId: item.id, prouctCategory: item.category},
          });
        }}
        style={{
          height: 220,
          width: 170,
          backgroundColor: colors.white,
          borderRadius: 10,
          marginHorizontal: 5,
          elevation: 2,
          marginBottom: 5,
```

```
        }}>
        <Image
          style={{height: '65%', width: '100%', resizeMode: 'contain'}}
          source={{
            uri:
              (item?.images?.length != 0 &&
                BASE_IP_OF_IMAGE_SERVER + GET_IMAGES + '/' + item?.images[0]) ||
              'https://www.freeiconspng.com/thumbs/no-image-icon/no-image-icon-
6.png',
          }}
        />
        <View style={{paddingHorizontal: '3%', paddingVertical: '1%'}}>
          <Text numberOfLines={1} style={{fontWeight: 'bold'}}>
            {item.title}
          </Text>
          <Text numberOfLines={1} style={{fontWeight: 'bold'}}>
            {'₹ ' +
              item.rentPrice[0] +
              ' / ' +
              (item.rentType === 'shortTerm' ? 'day' : 'month')}
          </Text>
          <Text numberOfLines={1}>{item.loc.address}</Text>
          {item.rating > 0 ? (
            <View
              style={{
                flexDirection: 'row',
                alignItems: 'center',
                width: '100%',
              }}>
              <Text
                style={{
                  color: '#fff',
                  paddingHorizontal: '2%',
                  backgroundColor: ratingColor,
                  borderRadius: 5,
                }}>
                {item.rating} ★
              </Text>
              <Text numberOfLines={1} style={{width: '65%'}}>
                {' '}
                ({item.totalRatings})
              </Text>
            </View>
          ) : null}
        </View>
```

```
      </TouchableOpacity>
    );
  };

  useEffect(async () => {
    await fetchData();
    fetchSimilarProducts();
  }, []);

  const fetchData = async () => {
    console.log(JSON.stringify(props.route.params.productData.productId));
    await fetchAdresses().then(async () => {
      var res = await postCall(
        BASE_IP_OF_HOST,
        GET_PRODUCT_BY_ID,
        {},
        {
          id: props.route.params.productData.productId,
          category: props.route.params.productData.prouctCategory,
        },
      );
      if (res.status == SUCCESS_RESPONSE) {
        var response = res.value;
        setProduct(response.data);
        setIsLoading(false);
      } else {
        console.log(res.value);
      }
    });
  };
  const addProductToWishlistById = async (id, category) => {
    var res = await postCall(
      BASE_IP_OF_MY_ACCOUNT,
      ADD_WISHLIST,
      {},
      {productId: props.route.params.productData.productId, productCategory:
props.route.params.productData.prouctCategory},
    );
  };
  const fetchAdresses = async () => {
    var res = await getCall(BASE_IP_OF_MY_ACCOUNT, GET_ADDRESSES, {});
    if (res.status == SUCCESS_RESPONSE) {
      var response = res.value;
      setAddresses(response.data);
    } else {
```

```
      console.log(error);
    }
  };
  const fetchSimilarProducts = async () => {
    console.log(
      'requesting for similar products for product id :' +
        props.route.params.productData.productId,
    );
    var res = await postCall(
      BASE_IP_OF_HOST,
      GET_SIMILAR_PRODUCTS,
      {},
      {
        id: props.route.params.productData.productId,
        category: props.route.params.productData.prouctCategory,
        location: {
          latitude: props.UserCurrentLocation.location.latitude,
          longitude: props.UserCurrentLocation.location.longitude,
        },
        page: 0,
        size: 10,
      },
```

MY ORDERS

RENTINS

```
import React, {useState, useEffect,useCallback} from 'react';
import {
  Text,
  View,
  Image,
  TouchableOpacity,
  StyleSheet,
  Dimensions,
  FlatList,
} from 'react-native';
import {
  BASE_IP_OF_MY_ACCOUNT,
  GET_RENTINS,
  BASE_IP_OF_IMAGE_SERVER,
  GET_IMAGES,
} from '_utils/api';
```

```
import {getCall} from '_components/backendApiCall'
import {SUCCESS_RESPONSE} from '_utils/constants';
import CardViewList from '_pages/myorders/cardViewList';
import { DataProvider, LayoutProvider} from 'recyclerlistview';
import translate from '../../utils/language.utils';
import Skeleton from './skeleton';
import { RENTIN_NO_PRODUCTS} from '_utils/appImages'
import NoProductsFound from '_components/noProductsFound';

import axios from 'react-native-axios';
import { colors } from 'react-native-elements';

const Rentin = props => {
  const [isLoading, setIsLoading] = useState(true);
  const [currentOrders, setCurrentOrders] = useState([]);
  const [previousOrders, setPreviousOrders] = useState([]);
  const [refreshing, setRefreshing] = React.useState(false);
  const onRefresh = useCallback(() => {
    setRefreshing(true);
    fetchData().then(() => {
      setRefreshing(false);
    });
  }, []);
  const [dataProvider, setDataProvider] = useState({
    'currentOrders': new DataProvider((r1, r2) => r1 !== r2),
    'previousOrders': new DataProvider((r1, r2) => r1 !== r2),
  }
  );
  useEffect(() => {
    fetchData();
  }, []);
  const fetchData = async () => {
    var res = await getCall(BASE_IP_OF_MY_ACCOUNT,GET_RENTINS,{});
    if(res.status == SUCCESS_RESPONSE)
      {
        var response=res.value;
        console.log(response.data);
        let cur = [],
          prev = [];
        response.data.forEach(element => {
          const t = new Date();
          const date = ('0' + t.getDate()).slice(-2);
          const month = ('0' + (t.getMonth() + 1)).slice(-2);
          const year = t.getFullYear();
          if (
```

```
            element.toDate.split('-').reverse().join('-') <=
            `${date}-${month}-${year}`
          )
            prev.push(element);
          else cur.push(element);
        });
        setDataProvider({
          'currentOrders': dataProvider.currentOrders.cloneWithRows(cur),
          'previousOrders': dataProvider.previousOrders.cloneWithRows(prev),
        })
        setCurrentOrders(cur);
        setPreviousOrders(prev);
        setIsLoading(false);
      }
    else {
        console.log(res.value);
      };
  };
  return (
    <View style={{width:
Dimensions.get('window').width,backgroundColor:colors.white}}>
      {isLoading ? (
        <Skeleton />
      ) : currentOrders.length != 0 || previousOrders.length != 0 ? (
        <CardViewList
          currentOrders={currentOrders}
          previousOrders={previousOrders}
          navigation={props.navigation}
          dataProvider={dataProvider}
          onRefresh={onRefresh}
          refreshing={refreshing}
        />
      ) : (
        <NoProductsFound text={translate("RENTALSTACK_sryYouHaveNotPlacedOrder")}
image={BASE_IP_OF_IMAGE_SERVER+GET_IMAGES+RENTIN_NO_PRODUCTS}    />
      )}
    </View>
  );
```

RENTOUTS

```
import React,{useState,useEffect,useCallback} from 'react';
import {
  Text,
  View,
  Image,
  TouchableOpacity,
  StyleSheet,
  Dimensions,
  FlatList,
} from 'react-native';
import {BASE_IP_OF_MY_ACCOUNT,GET_RENTOUTS,
  BASE_IP_OF_IMAGE_SERVER,
  GET_IMAGES,} from '_utils/api';
import CardViewList from '_pages/myorders/cardViewList';
import axios from 'react-native-axios';
import Skeleton from './skeleton';
import {SUCCESS_RESPONSE} from '_utils/constants';
import {getCall} from '_components/backendApiCall'
import { DataProvider, LayoutProvider} from 'recyclerlistview';
import NoProductsFound from '_components/noProductsFound';
import translate from '../../utils/language.utils';
import { colors } from 'react-native-elements';
import { RENTOUT_NO_PRODUCTS} from '_utils/appImages'

const Rentout = props => {
  const [isLoading, setIsLoading] = useState(true);
  const [currentOrders, setCurrentOrders] = useState([]);
  const [previoustOrders, setPreviousOrders] = useState([]);
  const [refreshing, setRefreshing] = React.useState(false);
  const [dataProvider, setDataProvider] = useState({
    'currentOrders': new DataProvider((r1, r2) => r1 !== r2),
    'previousOrders': new DataProvider((r1, r2) => r1 !== r2),
  }
  );
  const onRefresh = useCallback(() => {
    setRefreshing(true);
    fetchData().then(() => {
      setRefreshing(false);
    });
  }, []);
  useEffect(() => {
    fetchData();
  }, []);
  const fetchData = async () => {
    var res = await getCall(BASE_IP_OF_MY_ACCOUNT,GET_RENTOUTS,{});
```

```
    if(res.status == SUCCESS_RESPONSE)
      {
        var response=res.value;
        console.log(response.data);
        let cur=[],prev=[];
        response.data.forEach(element => {
          const t = new Date();
          const date = ('0' + t.getDate()).slice(-2);
          const month = ('0' + (t.getMonth() + 1)).slice(-2);
          const year = t.getFullYear();
          if (element.toDate.split('-').reverse().join('-') <= `${date}-${month}-
${year}`) prev.push(element);
          else cur.push(element);
        });
        setDataProvider({
          'currentOrders': dataProvider.currentOrders.cloneWithRows(cur),
          'previousOrders': dataProvider.previousOrders.cloneWithRows(prev),
        })
        setCurrentOrders(cur);
        setPreviousOrders(prev);
        setIsLoading(false);
      }
      else {
          console.log(res.value);
      };
  };
  return (
    <View style={{width:
Dimensions.get('window').width,backgroundColor:colors.white}}>
      {isLoading ? (
        <Skeleton />
      ) : currentOrders.length != 0 || previoustOrders.length != 0 ? (
        <CardViewList
          currentOrders={currentOrders}
          previousOrders={previoustOrders}
          navigation={props.navigation}
          dataProvider={dataProvider}
          onRefresh={onRefresh}
          refreshing={refreshing}
        />
      ) : (
        <NoProductsFound
text={translate("RENTALSTACK_sryYouHaveNotGivenAnyProductForRent")}
image={BASE_IP_OF_IMAGE_SERVER+GET_IMAGES+RENTOUT_NO_PRODUCTS}  />
      )}
```

```
    </View>
  );
```

ORDER SUMMARY CHECKOUT

```
import RNPgReactNativeSDK from 'react-native-pg-react-native-sdk';
import colors from '_utils/Colors';
import {Icon} from 'react-native-elements/dist/icons/Icon';
import orderSummaryStyles from './orderSummaryStyles';
import axios from 'react-native-axios';
import {SUCCESS_RESPONSE} from '_utils/constants';
import {postCall, getCall} from '_components/backendApiCall';
import SkeletonPlaceholder from 'react-native-skeleton-placeholder';
import StepIndicatorHeader from '_components/stepIndicator';
import {
  BASE_IP_OF_IMAGE_SERVER,
  GET_IMAGES,
  BASE_IP_OF_ORDER_SUMMARY,
  POST_PLACE_ORDER,
  GET_ORDER_SUMMARY,
  PLACE_ORDER,
  VERIFY_PAYMENT
} from '_utils/api';

const OrderSummary = props => {
  const [couponCode, setCouponCode] = useState(null);
  const [orderSummary, setOrderSummary] = useState({});
  const [isLoading, setIsLoading] = useState(true);
  const bufferViewCount = [1, 2, 3, 4, 5, 6];
  const scrollViewRef = useRef();
  useEffect(() => {
    fetchData();
  }, [couponCode, props.route.params]);

  const fetchData = async () => {
    var res = await postCall(
      BASE_IP_OF_ORDER_SUMMARY,
      GET_ORDER_SUMMARY,
      {},
      {
        productId: props.route.params.productData.productId,
        category: props.route.params.productData.productCategory,
        dateRange: {
          fromDate: props.route.params.orderData.dateRange.fromDate,
```

```
          toDate: props.route.params.orderData.dateRange.toDate,
        },
        location: props.route.params.orderData.deliveryAddress.location,
        couponCode: couponCode,
        ownermobileNumber: props.route.params.productData.mobileNumber,
      },
    );
    if (res.status == SUCCESS_RESPONSE) {
      var response = res.value;
      setOrderSummary(response.data);
      setIsLoading(false);
    } else {
      console.log(res.value);
    }
  };
  const proceedToPay = async () => {
    var res1 = await postCall(
      BASE_IP_OF_ORDER_SUMMARY,
      PLACE_ORDER,
      {},
      {
        productId: props.route.params.productData.productId,
        category: props.route.params.productData.productCategory,
        dateRange: {
          fromDate: props.route.params.orderData.dateRange.fromDate,
          toDate: props.route.params.orderData.dateRange.toDate,
        },
        location: props.route.params.orderData.deliveryAddress.location,
        couponCode: couponCode,
      },
    );
    if (res1.status == SUCCESS_RESPONSE) {
      var response = res1.value;
      var map = {
        orderId: response.data.orderId.toString(),
        orderAmount: response.data.orderAmount.toString(),
        appId: response.data.appId.toString(),
        tokenData: response.data.tokenData.toString(),
        orderCurrency: response.data.orderCurrency.toString(),
        // "orderNote": response.data.orderNote.toString(),
        notifyUrl: response.data.notifyUrl.toString(),
        customerName: response.data.customerName.toString(),
        customerPhone: response.data.customerPhone.toString(),
        customerEmail: response.data.customerEmail.toString(),
      };
```

```
      RNPgReactNativeSDK.startPaymentWEB(map, 'TEST',async result  => {
        console.log(result);
        var res2 = await postCall(BASE_IP_OF_WALLET,VERIFY_PAYMENT , {},result);
        if (res2.status == SUCCESS_RESPONSE) {
          var response = res2.value;
          // props.navigation.navigate();
        }
        else{
          console.log(res2.value);
        }
      });
    } else {
      console.log(res1.value);
    }
  };
  return (
    <>
      {isLoading ? (
        <SkeletonPlaceholder
          backgroundColor={colors.skeletonBackgroundColor}
          highlightColor={colors.skeletonHighlightColor}>
          <View style={{marginHorizontal: 2}}>
            <View
              style={{
                width: '100%',
                height: 60,
                borderBottomLeftRadius: 20,
                    source={{
                      uri:
                        props.route.params.productData.productImage != null
                          ? BASE_IP_OF_IMAGE_SERVER +
                            GET_IMAGES +
                            '/' +
                            props.route.params.productData.productImage
                          : 'https://www.freeiconspng.com/thumbs/no-image-
icon/no-image-icon-6.png',
                    }}></Image>
              </View>
            </View>
          </View>
          <Text style={{fontWeight: 'bold', paddingHorizontal: '3%'}}>
            {' '}
            Offer & Benefits{' '}
          </Text>
```

PAYMENT

```
import colors from '_utils/Colors';
import {Icon} from 'react-native-elements/dist/icons/Icon';
import orderSummary1Styles from './orderSummary1Styles';
import axios from 'react-native-axios';
import SkeletonPlaceholder from 'react-native-skeleton-placeholder';
import StepIndicatorHeader from '_components/stepIndicator';
import {
  BASE_IP_OF_IMAGE_SERVER,
  GET_IMAGES,
  BASE_IP_OF_ORDER_SUMMARY,
  GET_ORDER_SUMMARY,
} from '_utils/api';

const Payment = props => {
  const [isLoading, setIsLoading] = useState(false);
  const bufferViewCount = [1, 2, 3, 4, 5, 6];

  return (
    <>
      {isLoading ? (
        <SkeletonPlaceholder
          backgroundColor={colors.skeletonBackgroundColor}
          highlightColor={colors.skeletonHighlightColor}>
          <View style={{marginHorizontal: 2}}>
            <View
```

NOTIFICATION

```
import React, {useEffect, useState} from 'react';
import {
  View,
  Text,
  ScrollView,
  Dimensions,
  TouchableOpacity,
} from 'react-native';
import {Icon} from 'react-native-elements';
```

```
import colors from '_utils/Colors';
import {RecyclerListView, DataProvider, LayoutProvider} from 'recyclerlistview';
import {SUCCESS_RESPONSE} from '_utils/constants';
import notificationsStyles from './notificationsStyles';
import {
  BASE_IP_OF_NOTIFICATION_SERVICE,
  MARK_NOTIFICATION_AS_READ,
  GET_NOTIFICATIONS,
} from '_utils/api';
const {height, width} = Dimensions.get('window');
import {postCall, getCall} from '_components/backendApiCall';

const NotificationsScreen = props => {
  const [notifications, setNotifications] = useState([]);

  const [dataProvider, setDataProvider] = useState(
    new DataProvider((r1, r2) => r1 !== r2),
  );
  const [layoutProvider] = useState(
    new LayoutProvider(
      index => {
        return index;
      },
      (type, dim) => {
        dim.width = width;
        dim.height = 69;
      },
    ),
  );

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

  const fetchData = async () => {
    var res = await getCall(
      BASE_IP_OF_NOTIFICATION_SERVICE,
      GET_NOTIFICATIONS,
      {},
    );
    if (res.status == SUCCESS_RESPONSE) {
      var response = res.value;
      setDataProvider(dataProvider.cloneWithRows(response.data));
      setProducts(response.data);
      setIsLoading(false);
```

```
    } else {
      console.log(res.value);
    }
const history = (type, item) => {
    return (
      <View style={{paddingHorizontal: 10, paddingTop: 6}}>
        <TouchableOpacity
          style={{
            backgroundColor: '#fff',
            borderRadius: 10,
            paddingVertical: 8,
            elevation: 2,
          }}>
            <View style={notificationsStyles.container}>
              <Icon
                name="reload-alert"
                type="material-community"
                size={26}
                color={'#fa7d00'}
              />
              <Text
            style={{
              fontSize: 19,
              paddingHorizontal:'10%'
            }}>
            {item.title}{' '}
          </Text>
            </View>
        </TouchableOpacity>
      </View>
    );
```

PRODUCTS

```
import React, {useState, useEffect, useCallback} from 'react';
import {
  Text,
  View,
  TouchableOpacity,
  FlatList,
  Alert,
  ScrollView,
```

```
  Dimensions,
  Image,
  RefreshControl,
} from 'react-native';
import colors from '_utils/Colors';
import {Icon} from 'react-native-elements/dist/icons/Icon';
import {
  BASE_IP_OF_IMAGE_SERVER,
  GET_IMAGES,
  BASE_IP_OF_MY_ACCOUNT,
  GET_PRODUCTS,
  DELETE_PRODUCT,
  BASE_IP_OF_HOST,
  GET_AVAILABLE_DATES,
} from '_utils/api';
import Modal from 'react-native-modal';
import DatePicker from '_components/datePicker';
import {SUCCESS_RESPONSE} from '_utils/constants';
import {getCall, postCall} from '_components/backendApiCall';
import {RecyclerListView, DataProvider, LayoutProvider} from 'recyclerlistview';
import NoProductsFound from '_components/noProductsFound';
import translate from '../../utils/language.utils';
import { PRODUCTS_NO_PRODUCTS} from '_utils/appImages'
import Skeleton from './skeleton';
const MyProducts = props => {
  const [products, setProducts] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const [deleteCount, setDeleteCount] = useState([1]);
  const [refreshing, setRefreshing] = React.useState(false);
  const [showCalender, setShowCalender] = useState({
    show: false,
    id: null,
    category: null,
  });
  const [bookedDates, setBookedDates] = useState([]);
  const {height, width} = Dimensions.get('window');
  const [dataProvider, setDataProvider] = useState({
    LIV: new DataProvider((r1, r2) => r1 !== r2),
    UNV: new DataProvider((r1, r2) => r1 !== r2),
    REJ: new DataProvider((r1, r2) => r1 !== r2),
  });
  const [layoutProvider] = useState(
    new LayoutProvider(
      index => {
        return index;
```

```
      },
      (type, dim) => {
        dim.width = width;
        dim.height = 160;
      },
    ),
  );

  const onRefresh = useCallback(() => {
    setRefreshing(true);
    fetchData().then(() => {
      setRefreshing(false);
    });
  }, []);

  useEffect(() => {
    fetchData();
    const unsubscribe = props.navigation.addListener('focus', () => {
      fetchData();
    });
    return unsubscribe;
  }, [props.navigation]);

  const fetchData = async () => {
    var res = await getCall(BASE_IP_OF_MY_ACCOUNT, GET_PRODUCTS, {});
    if (res.status == SUCCESS_RESPONSE) {
      var response = res.value;
      setDataProvider({
        LIV: dataProvider.LIV.cloneWithRows(response.data[0].products),
        UNV: dataProvider.UNV.cloneWithRows(response.data[1].products),
        REJ: dataProvider.REJ.cloneWithRows(response.data[2].products),
      });
      setProducts(response.data);
      setIsLoading(false);
    } else {
      console.log(res.value);
    }
  };

  const deleteProductById = async (id, category) => {
    setIsLoading(true);
    var res = await postCall(
      BASE_IP_OF_HOST,
      DELETE_PRODUCT,
      {},
```

```
      {id: id, category: category},
    );
    if (res.status == SUCCESS_RESPONSE) {
      fetchData();
    } else {
      console.log(res.value);
      setIsLoading(false);
    }
  };

  const getAvailableDatesById = async (id, category) => {
    var res = await postCall(
      BASE_IP_OF_HOST,
      GET_AVAILABLE_DATES,
      {},
      {id: id, category: category},
    );
    if (res.status == SUCCESS_RESPONSE) {
      var response = res.value;
      setBookedDates(response.data);
      setShowCalender({
        show: true,
        id: id,
        category: category,
      });
    } else {
      console.log(res.value);
    }
  };
const statusColor = status => {
    switch (status) {
      case 'LIV':
        return colors.green;
      case 'UNV':
        return colors.yellow;
      case 'REJ':
        return colors.red;
    }
```