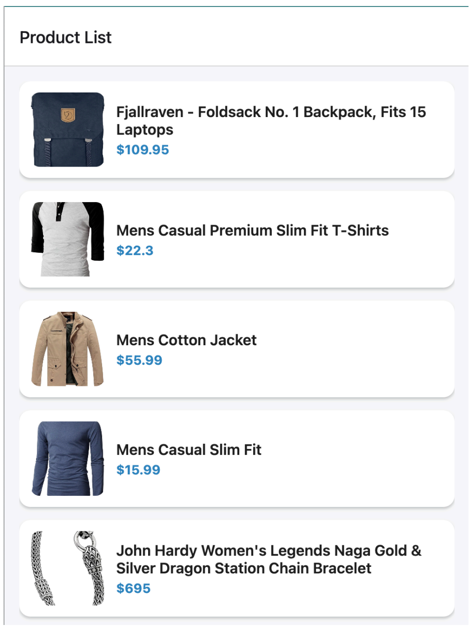
การเขียนโปรแกรมแสดงข้อมูลแบบ WebAPI ด้วย React Native

ผศ. ดร. ชนันท์กรณ์ จันแดง



Product List

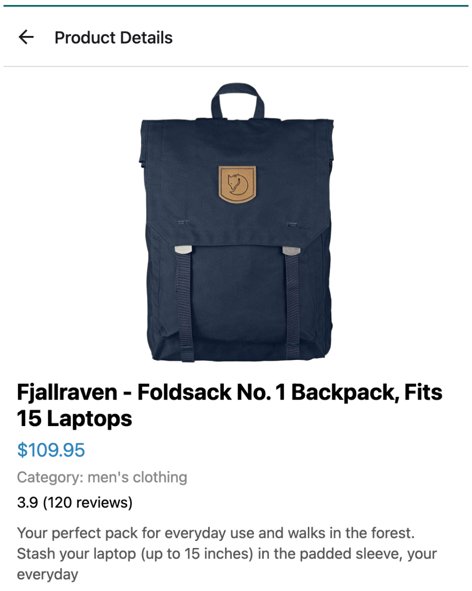
Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops
$109.95

Mens Casual Premium Slim Fit T-Shirts
$22.3

Mens Cotton Jacket
$55.99

Mens Casual Slim Fit
$15.99

John Hardy Women's Legends Naga Gold & Silver Dragon Station Chain Bracelet
$695



Product Details

Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops
$109.95
Category: men's clothing
3.9 (120 reviews)
Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday

Step 0 — Environment & Tooling

0.1 Create workspace

```
npx create-expo-app product-showcase —templates
cd product-showcase
```

Checkpoint: Folder contains `App.js` and `package.json`.

0.2 First run

```
npx expo start
```

Open in Expo Go or the web (`w`).
If you use web (w), you need to install:

```
npx expo install react-dom react-native-web @expo/metro-runtime
```

Checkpoint: Default Expo screen renders.

Step 1 — Baseline App (Hello)

1.1 Replace `App.js` with a minimal baseline

- Root functional component rendering a centered “Hello, Expo!”.
  Checkpoint: Text appears; no warnings.

1.2 Verify hot reload

- Edit the string; ensure live reload updates.
  Checkpoint: UI updates without a complete restart.

Step 2 — Navigation Skeleton

2.1 Install navigation dependencies

```
npm i @react-navigation/native @react-navigation/stack
expo install react-native-gesture-handler react-native-screens react-
native-safe-area-context
```

Checkpoint: Install completes; no peer-dep errors.

2.2 Wire the navigator

- Import NavigationContainer, createStackNavigator.

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

- Define Stack = createStackNavigator().

```
const Stack = createStackNavigator();
```

- Add two placeholder screens: `ListScreen` and `DetailsScreen`.

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

function ListScreen({ navigation }) {
  return (
    <View>
      <Text>Product List</Text>
    </View>
  );
}
```

```
function DetailsScreen() {
  return (
    <View>
      <Text>Product Details</Text>
    </View>
  );
}
```

- Wrap App in <NavigationContainer><Stack.Navigator>…</Stack.Navigator></NavigationContainer>.

```
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Product List" component={ListScreen} />
        <Stack.Screen name="Product Details" component={DetailsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
```

Checkpoint: App runs with a header titled “Product List”.

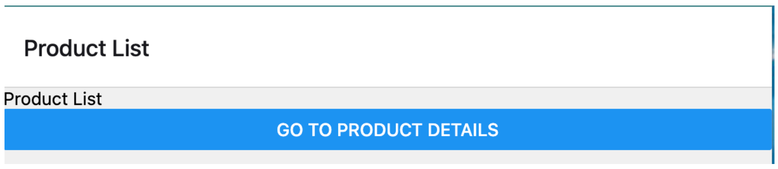
2.3 Test navigation action

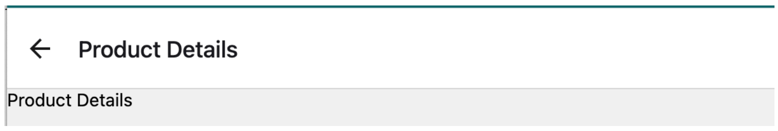
- Add a button on `ListScreen` that navigates to `DetailsScreen` with a mock `product` param.

```
    <View>
      <Text>Product List</Text>

      <Button
        title="Go to Product Details"
        onPress={() => navigation.navigate('Product Details')}
      />
    </View>
```

- **Checkpoint:** Tap → navigates; back button returns.





Step 3 — Static Data & List/Detail Flow

3.1 Introduce domain model (static)

- Define `products` array (id, title, price, image, description, category, rating).
  Checkpoint: Data object exists in code.

```
products = [
    {
        "id": 1,
        "title": "iPhone 15 Pro",
        "price": 1199.99,
        "image": "https://via.placeholder.com/300",
        "description": "The latest iPhone 15 Pro with A17 Bionic chip and
Titanium build.",
        "category": "Electronics",
        "rating": {
            "rate": 4.8,
            "count": 254
        }
    },
    {
        "id": 2,
        "title": "Nike Air Zoom Pegasus",
        "price": 129.99,
        "image": "https://via.placeholder.com/300",
        "description": "High-performance running shoes designed for
comfort and durability.",
        "category": "Fashion",
        "rating": {
            "rate": 4.5,
            "count": 102
        }
    },
    {
        "id": 3,
        "title": "Sony WH-1000XM5",
        "price": 349.99,
        "image": "https://via.placeholder.com/300",
        "description": "Industry-leading wireless noise-canceling
headphones with premium sound quality.",
        "category": "Electronics",
        "rating": {
            "rate": 4.7,
            "count": 187
        }
    }
]
```

3.2 **Implement `ProductList`**

- Add new components named `ProductList`
  - Create file named components/productlist.js

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

function ProductList({ navigation }) {
  const renderItem = ({item}) => (<View></View>);
  return (
    <View style={{ flex: 1, padding: 16 }}>
      <FlatList
        data={products}
        keyExtractor={(item) => item.id.toString()}
        renderItem={renderItem}
        showsVerticalScrollIndicator={false}
      />
    </View>
  );
}
export default ProductList;
```

- Render `FlatList` or a mapped `ScrollView` of cards (image, title, price).

```
const renderItem = ({ item }) => (
    <TouchableOpacity
     style={{
        backgroundColor: '#fff',
        marginBottom: 12,
        padding: 10,
        borderRadius: 8,
        flexDirection: 'row',
        alignItems: 'center',
        elevation: 3,
        shadowColor: '#000',
        shadowOffset: { width: 0, height: 1 },
        shadowOpacity: 0.2,
        shadowRadius: 1,
      }}
    >
      <Image
        source={{ uri: item.image }}
        style={{ width: 80, height: 80, marginRight: 12,
                      borderRadius: 6 }}
      />
      <View style={{ flex: 1 }}>
        <Text style={{ fontSize: 16, fontWeight: 'bold' }}>
                      {item.title}</Text>
        <Text style={{ color: '#888', marginTop: 4 }}>
                           ${item.price}</Text>
      </View>
    </TouchableOpacity>
  );
```

- On press → navigation.navigate('Product Details', { product }).

```
onPress={() => navigation.navigate('Product Details', { product: item })}
```

Checkpoint: Scrollable list renders; items tappable.

3.3 Implement `ProductDetails`

- Create file named components/productdetails.js

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

function ProductDetails({ route }) {
  return (
    <ScrollView style={{ flex: 1, padding: 16,
                         backgroundColor: '#fff' }}>
       <Text> Product Details </Text>
    </ScrollView>
  );
}

export default ProductDetails;
```

- Read const { product } = route.params.

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

function ProductDetails({ route }) {
  const { product } = route.params;
  return (
    <ScrollView style={{ flex: 1, padding: 16,
                         backgroundColor: '#fff' }}>
       <Text>         {product.title}  </Text>
    </ScrollView>
  );
}

export default ProductDetails;
```

- Render image, title, price, description, category, rating.

```
return (
    <ScrollView style={{ flex: 1, padding: 16, backgroundColor: '#fff' }}
>
      <Image
        source={{ uri: product.image }}
        style={{
          width: '100%',
          height: 300,
          borderRadius: 8,
          marginBottom: 16,
        }}
        resizeMode="contain"
      />
      <Text style={{ fontSize: 24, fontWeight: 'bold',
                                  marginBottom: 8 }}>
        {product.title}
      </Text>
      <Text style={{ fontSize: 20, color: '#2E86C1', marginBottom: 8 }}>
        ${product.price}
      </Text>
      <Text style={{ fontSize: 16, color: '#888', marginBottom: 8 }}>
        Category: {product.category}
      </Text>
      <Text style={{ fontSize: 16, marginBottom: 12 }}>
         {product.rating.rate} ({product.rating.count} reviews)
      </Text>

      <Text style={{ fontSize: 16, lineHeight: 22, color: '#555' }}>
        {product.description}
      </Text>
    </ScrollView>
```

Checkpoint: Details screen shows correct item data.

3.4 Styling pass (v1)

- Add `StyleSheet` for card shadows, radii, paddings; detail text hierarchy.

```
import React from 'react';
import {
  View,
  Text,
  FlatList,
  Image,
  TouchableOpacity,
  StyleSheet,
} from 'react-native';

function ProductList({ navigation }) {
  const renderItem = ({ item }) => (
    <TouchableOpacity
      onPress={() => navigation.navigate('Product Details', { product:
item })}
      style={styles.card}
    >
      <Image source={{ uri: item.image }} style={styles.cardImage} />
      <View style={styles.cardContent}>
        <Text style={styles.cardTitle}>{item.title}</Text>
        <Text style={styles.cardPrice}>${item.price}</Text>
      </View>
    </TouchableOpacity>
  );

  return (
    <View style={styles.container}>
      <FlatList
        data={products}
        keyExtractor={(item) => item.id.toString()}
        renderItem={renderItem}
        showsVerticalScrollIndicator={false}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
    backgroundColor: '#F8F9FA',
  },
  card: {
    backgroundColor: '#fff',
    marginBottom: 14,
    padding: 12,
    borderRadius: 12,
    flexDirection: 'row',
    alignItems: 'center',

    // Shadows for Android & iOS
    elevation: 4,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.2,
    shadowRadius: 3,
  },
  cardImage: {
    width: 80,
    height: 80,
    marginRight: 12,
    borderRadius: 10,
  },
  cardContent: {
    flex: 1,
  },
  cardTitle: {
    fontSize: 16,
    fontWeight: '600',
    marginBottom: 4,
    color: '#222',
  },
  cardPrice: {
    fontSize: 14,
    color: '#2E86C1',
    fontWeight: 'bold',
  },
});

export default ProductList;
```

Checkpoint: Visual separation is clear; text is legible.

Step 4 — Remote Data & Asynchrony
4.1 State & lifecycle

- In ProductList, add products, loading via useState.
- Fetch once in `useEffect` from `https://fakestoreapi.com/products`.

```
function ProductList({ navigation }) {
    const [products, setProducts] = useState([]);

    useEffect(() => {
        const fetchProducts = async () => {
            try {
               const response = await fetch('https://fakestoreapi.com/
products');
                const data = await response.json();
                setProducts(data);
            } catch (error) {
                console.error('Error fetching products:', error);
            } finally {
                setLoading(false);
            }
        };
        fetchProducts();
    }, []);
```

4.2 Loading UI

- While `loading`, render `ActivityIndicator` + message.

```
const [loading, setLoading] = useState(true);

if (loading) {
      return (
        <View style={{ flex: 1, alignItems: 'center',
                          justifyContent: 'center', padding: 16 }}>
          <ActivityIndicator size="large" />
          <Text style={{ marginTop: 8 }}>Loading products…</Text>
         </View>
        );
    }
```