LINCOLN UNIVERSITY COLLEGE DKU016 (B) 5 SETARA-2017 University College EXCELLENT ISO 9001:2015 Certified

# WEEK 3: Styling & Layout in React Native

Course Title: Android Programming (React Native II)

Course Code: BIT 6294

Target Audience: (300 Level Lincoln University (NSUK) Campus Students)

Credit: 4

Topic: API Integration

Lecturer: Mr. Vincent

## Main Goal:

The main goal of this lesson is to equip students with the ability to connect a React Native application to external APIs.

Learning Objectives:

- Understand how to fetch data from external APIs
- Display fetched data using FlatList or ScollView
- Handle errors gracefully

## API Integration

API (Application Programming Interface) integration is a fundamental skill in mobile development. It allows your React Native app to communicate with external servers, fetch data, and provide dynamic content to users.

In React Native, API integration is usually done using HTTP requests with tools like:

- fetch() (built-in)
- axios (third-party library)

## 1. Fetching Data Using fetch()

fetch() is a built-in JavaScript function used to make network requests.

```
fetch(‘https://api.example.com/data’)
    .then(response => response.json())
    .then(data  => console.log (data))
    .catch(error  => console.error (error));
```

### Example in React Native

```
import { useEffect, useState } from "react";
import { StyleSheet, View } from "react-native";
import { SafeAreaView } from "react-native-safe-area-context";

export default function Index() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/users")
      .then((res) => res.json())
      .then((data) => setUsers(data))
      .catch((err) => console.log(err));
  }, []);

  return (
    <SafeAreaView style={styles.conatiner}>
      <View></View>
    </SafeAreaView>
  );
}
```

**What is useEffect?**

useEffect is a React Hook that allows function components to perform side effects after rendering.it runs after rendering and uses a dependency array to control when it executes. Proper use of useEffect prevents infinite loops, improves performance, and ensure clean resource management.

## 2. Fetching Data Using Axios

Axios is a popular third-party library that provides a more feature-rich alternative to fetch(). It automatically transforms JSON data and provides better error handling.

- npm install axios

**Basic Axios Example**

```
import axios from 'axios';


axios.get('https://jsonplaceholder.typicode.com/users')
.then(response => console.log(response.data))

.catch(error => console.log(error));
```

## 3. Displaying Data with FlatList

FlatList is a performant component for rendering lists in React Native. It only renders items that are currently visible on screen, making it ideal for large datasets.

Basic FlatList Usage

Key Props:

- data: Array of items to render

- renderItem: Function that renders each item
- keyExtractor: Function to extract unique keys for each item

## Example Using ScrollView

```
<SafeAreaView style={styles.container}>
      <ScrollView>
        <View>
          {users.map((item) => (
            <View key={item.id}>
              <Text>Name: {item.name}</Text>
              <Text>Username: {item.username}</Text>
              <Text>Email: {item.email}</Text>
              <Text>Street: {item.address.street}</Text>
              <Text>City: {item.address.city}</Text>
            </View>
          ))}
        </View>
      </ScrollView>
    </SafeAreaView>
```

## Example Simple FlatList

```
import axios from "axios";
import { useEffect, useState } from "react";
import { FlatList, StyleSheet, Text, View } from "react-native";
import { SafeAreaView } from "react-native-safe-area-context";
```

```
export default function Index() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    axios
      .get("https://jsonplaceholder.typicode.com/users")
      .then((res) => setUsers(res.data))
      .catch((err) => console.log(err));
  }, []);

  return (
    <SafeAreaView style={styles.conatiner}>
      <View>
        <FlatList
          data={users}
          keyExtractor={(item) => item.id.toString()}
          renderItem={({ item }) => (
            <View>
              <Text>Name:{item.name}</Text>
              <Text>UserNmae:{item.username}</Text>
              <Text>email:{item.email}</Text>
              <Text>Street:{item.address.street}</Text>
              <Text>City:{item.address.city}</Text>
            </View>
          )}
        ></FlatList>
      </View>
    </SafeAreaView>
  );
}
```

**ScrollView:** Renders all items at once, uses more memory.

**Use scrollView when:**

- List is small
- Content is mostly static
- You want quick layout

**FlatList:** Renders only visible items, Optimized for large data

**Use FlatList when:**

- List is large
- Data changes often
- Performance matters

**Error Handling in API Calls**

**Why Error Handing is Important**

- Network issues
- Server errors
- Invalid responses

Basic Error Handing Example with axios

```
useEffect(() => {

   axios

     .get("https://jsonplaceholder.typicode.com/users")

     .then((res) => setUsers(res.data))

     .catch((err) => {

       if (err.response) {

         console.log("Server erro", err.response.data);
```

```
        } else if (err.request) {
          console.log("Network error,:No response from server");
        } else {
          console.log("Error:", err.message);
        }
      });
  }, []);
```