**Example of Infinite Loading Listview in React Native using FlatList**

Paging in ListView

This is an Example of Infinite Loading Listview in React Native using FlatList. A React Native list view that supports infinite scroll. In mobile application development, ListView has a very important part as we use ListView in almost all applications. If we have to load a huge amount of data in a listview we have to use pagination for the seamless performance. A ListView with Load More Button in the bottom to load data can be an option but what if we load the data automatically when the user reaches the end of the list? This feature will give a good user experience to your application users.

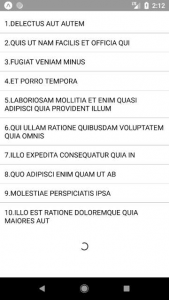
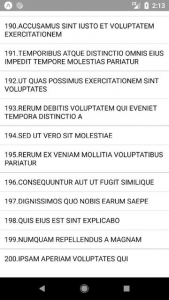
What is pagination?

Let’s take a case where you can simply imagine what if Facebook loads all the Facebook posts from the Facebook database on your Facebook wall? Doesn’t it sound nasty? It will take hundreds of hours even more than that so what is the best way to load the data? Simply paging. When you open the Facebook they just show you 10-20 post and while you scroll the Facebook they load the next 10-20 posts in background. That is what the paging is.

In our example,

1. We are loading the first 10 posts from the web API call in componentDidMount.
2. While the user reaches the bottom of the list we call the Web API again to get the next 10 posts.

We are using a variable offset to manage the index on upcoming Data. We will increase the offset by 1 after the successful call of the web API so that when we call the web API next time we will get the next data-set. So let’s get started with the example.

//This is an example of React Native

//Pagination to Load More Data dynamically - Infinite List

import React, { Component } from 'react';

//import react in our code.

import {

View,

Text,

StyleSheet,

FlatList,

ActivityIndicator

} from 'react-native';

//import all the components we are going to use.

export default class App extends Component {

constructor() {

super();

this.state = {

loading: false,

isListEnd: false,

//Loading state used while loading the data for the first time

serverData: [],

//Data Source for the FlatList

fetching\_from\_server: false,

//Loading state used while loading more data

};

this.offset = 1;

//Index of the offset to load from web API

}

componentDidMount() {

this.loadMoreData();

}

loadMoreData = () => {

if (!this.state.fetching\_from\_server && !this.state.isListEnd) {

//On click of Load More button We will call the web API again

this.setState({ fetching\_from\_server: true }, () => {

fetch('https://aboutreact.herokuapp.com/getpost.php?offset=' + this.offset)

//Sending the currect offset with get request

.then(response => response.json())

.then(responseJson => {

if (responseJson.results.length > 0) {

//Successful response from the API Call

this.offset = this.offset + 1;

//After the response increasing the offset for the next API call.

this.setState({

serverData: [...this.state.serverData, ...responseJson.results],

//adding the new data with old one available

fetching\_from\_server: false,

//updating the loading state to false

});

} else {

this.setState({

fetching\_from\_server: false,

isListEnd: true,

});

}

})

.catch(error => {

console.error(error);

});

});

}

};

renderFooter() {

return (

<View style={styles.footer}>

{this.state.fetching\_from\_server ? (

<ActivityIndicator color="black" style={{ margin: 15 }} />

) : null}

</View>

);

}

render() {

return (

<View style={styles.container}>

{this.state.loading ? (

<ActivityIndicator size="large" />

) : (

<FlatList

style={{ width: '100%' }}

keyExtractor={(item, index) => index.toString()}

data={this.state.serverData}

onEndReached={() => this.loadMoreData()}

onEndReachedThreshold={0.5}

renderItem={({ item, index }) => (

<View style={styles.item}>

<Text style={styles.text}>

{item.id}

{'.'}

{item.title.toUpperCase()}

</Text>

</View>

)}

ItemSeparatorComponent={() => <View style={styles.separator} />}

ListFooterComponent={this.renderFooter.bind(this)}

//Adding Load More button as footer component

/>

)}

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

paddingTop: 30,

},

item: {

padding: 10,

},

separator: {

height: 0.5,

backgroundColor: 'rgba(0,0,0,0.4)',

},

text: {

fontSize: 15,

color: 'black',

},

footer: {

padding: 10,

justifyContent: 'center',

alignItems: 'center',

flexDirection: 'row',

},

});