

Appice



**CUSTOMER  
DATA &  
ENGAGEMENT  
PLATFORM**





# Integration document

## React Native Development

*This document will help you integrate the Appice SDK in your Android/iOS Projects.*

## Table of Contents

<b>Setting up your app on //Appice.io</b>	<b>4</b>
Sign-up	4
Setup your App	4
<b>Initializing Appice in your project</b>	<b>7</b>
Appice sdkInitialization	7
Android (Update build.gradle files)	7
android->build.gradle	7
android->app->build.gradle	7
NOTE	7
iOS	8
<b>React Native JavaScript</b>	<b>8</b>
Import Appice Plugin	9
Get The Keys	10
Initialise SDK	10
Send Events	10
Set User	11
Get User	11
SetUserId	12
GetUserId	12
ApplInbox	12
GetInboxMessages	12
GetMessageCount	12
GetMessageForID	12
UpdateInboxMessage	13
ApplInbox - RichPush	13
Quick Action	14
isDeviceReady	16
PageView Event	16
Example Code for App.js	22
Setup Push Notification for Android	32
IOS App side Changes	35
Setup Push Notification	35
Setup DeepLink for url	37
Encryption/ Non-Encryption	38
<b>Verify Push Notifications</b>	<b>39</b>
<b>Install App on your Android/iOS phone</b>	<b>42</b>
<b>Verify integration on Appice panel</b>	<b>44</b>

# Setting up your app on //Appice.io

## Sign-up

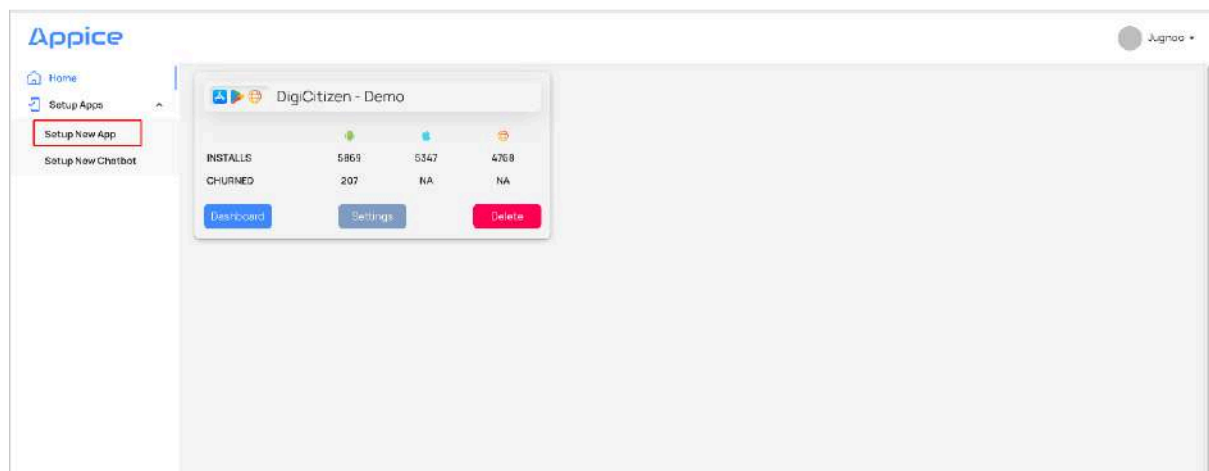
You need to sign-up and create an account with [www.Appice.io](https://www.Appice.io).

1. Visit <https://panel.Appice.io/signup> to Register.
2. This brings you to the “Sign up” page.
3. Provide your Name, Email address and your chosen password and create your account.
4. Login using your username and password credentials once your account is approved.

## Setup your App

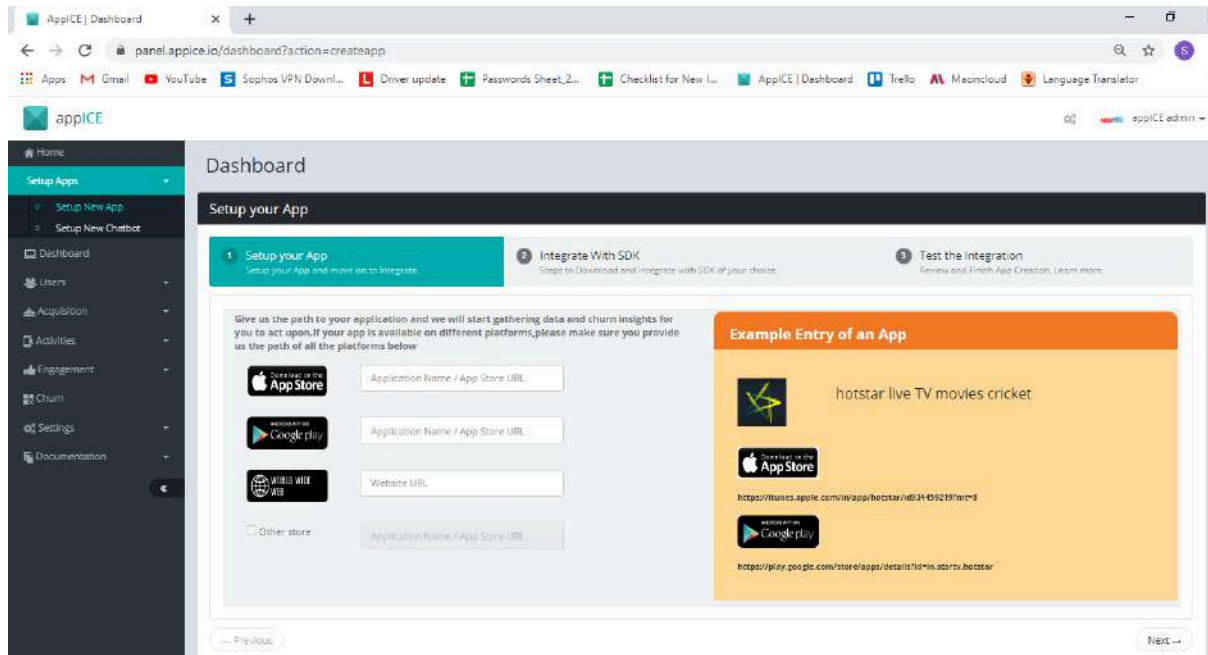
To start the process of integrating Appice in your app, you will first need to setup your app on the Appice dashboard.

1. Click on Setup New App in left panel.



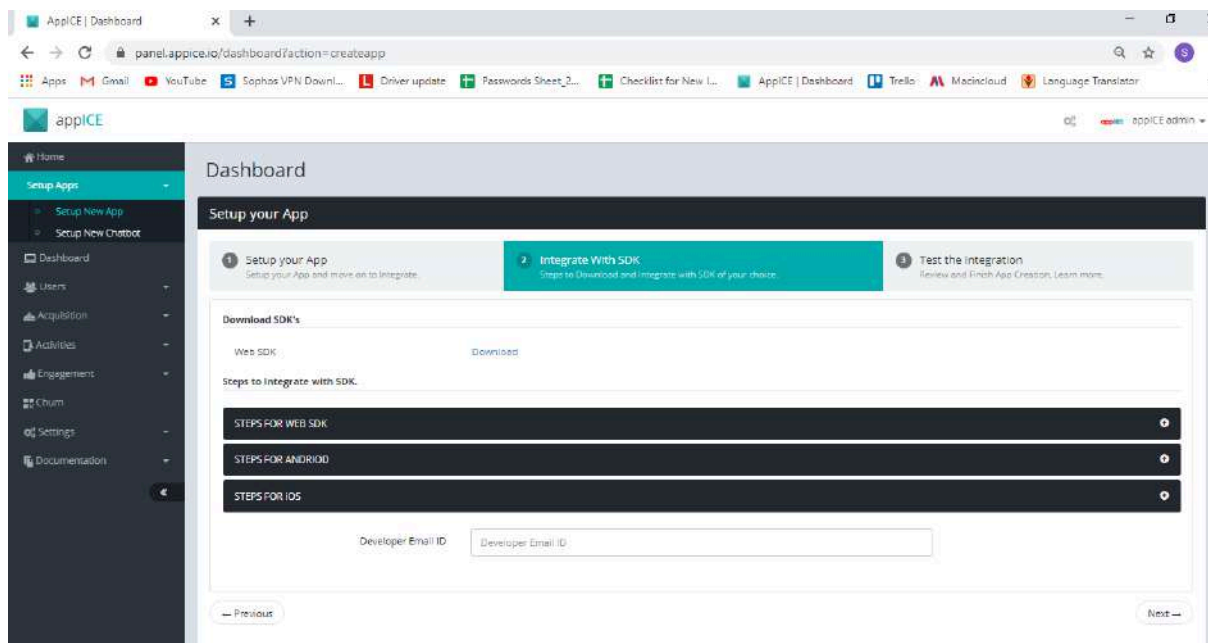
2. Provide the link to your mobile app on the Google Play Store as well as Apple store if you have both versions of your app.

Click Next to go to the next page.



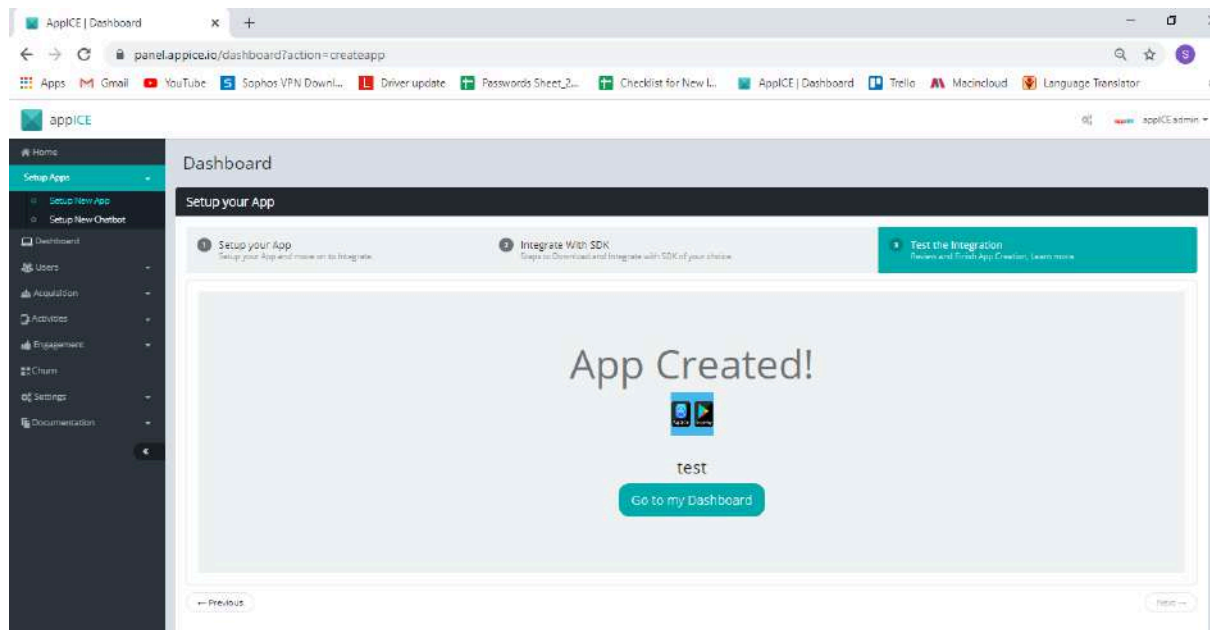
Click Next to go to next page.

- Now you have access to the SDK's for React Native including the instructions for integration. To make it easier, we have also provided the option to provide your developer's email address so that those instructions can be emailed to them directly.

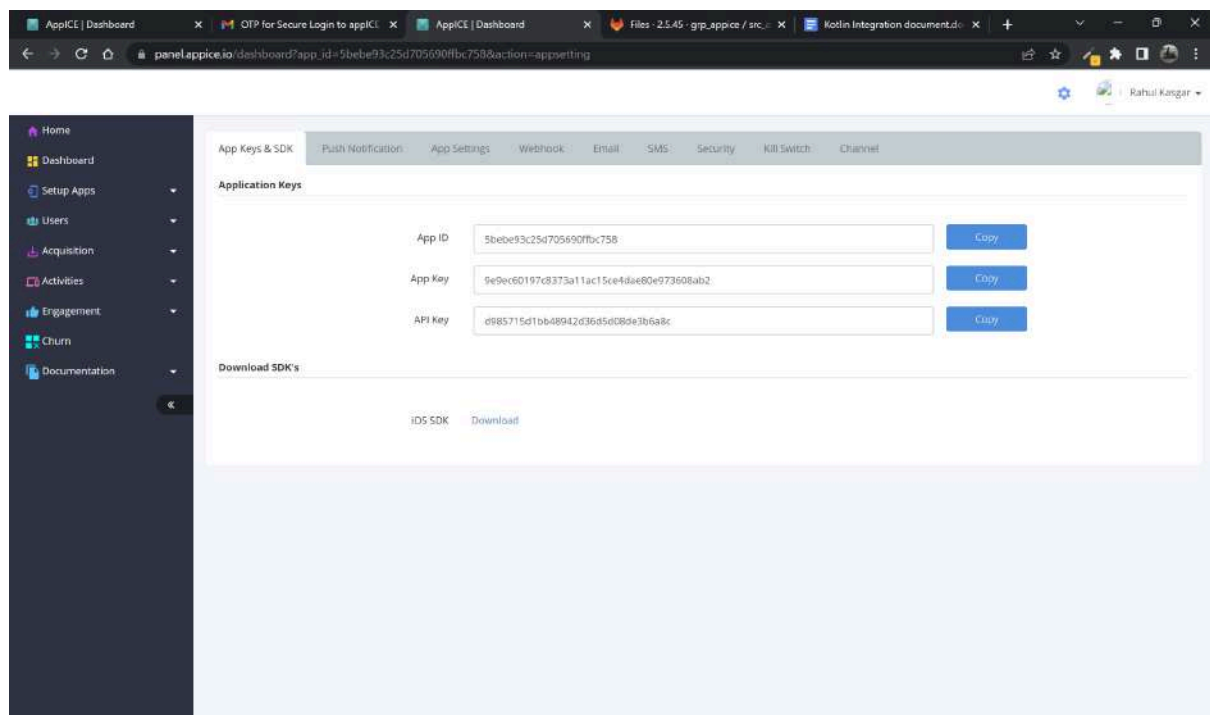


Click Next to go to next page.

- Now you are all ready to start receiving data from your mobile app once the developer completes the integration and publishes the app again on the app store.



5. Open your panel and select your app > App Settings  
You will find the App ID, App Key, API Key here.



# Initializing Appice in your project

## Appice sdkInitialization

To add Appice Plugin in your project, run this command:

```
npm i reactnative-plugin-Appice  
(or)  
npm i reactnative-plugin-Appice --legacy-peer-deps
```

## Android (Update build.gradle files)

android->build.gradle

```
allprojects{  
    repositories {  
        maven {  
            url "https://gitlab.com/api/v4/projects/10636887/packages/maven"  
            credentials(HttpHeaderCredentials) {  
                name = "Deploy-Token"  
                value = "PJMxxXdArqsmqDx4x5B6"  
            }  
            authentication {  
                header(HttpHeaderAuthentication)  
            }  
        }  
    }  
}
```

Now we have successfully added Appice SDK.

## NOTE

Apps targeting Android 13 and above, you must add the com.google.android.gms.permission.AD\_ID permission in the AndroidManifest.xml

***..\android\app\src\main\AndroidManifest.xml***

```
"<uses-permission android:name="com.google.android.gms.permission.AD_ID"/>"
```

## iOS

A. If your application has a podfile then skip this step, otherwise do the following:

1. Go to terminal and open ios folder from terminal (`cd projectName/ios`) and type the following command:
2. `pod init`
3. This will create a podfile.

B. Once you have a podfile, open the podfile and add these lines in the target section (used for building the application) of the podfile:

1. `pod 'reactnative-plugin-Appice',:path=>'../node_modules/reactnative-plugin-Appice'`

C. After adding these lines, do the following steps:

1. Go to terminal and open ios folder from terminal (`cd projectName/ios`) and type the following command:
2. `pod install`
3. Go to the Products section and do a clean build
4. Run your project (`Cmd+R`)

### Issue faced like :

Undefined symbols for architecture x86\_64:

"\_OBJC\_CLASS\_\$\_RCTAppDelegate", referenced from:

\_OBJC\_CLASS\_\$\_AppDelegate in AppDelegate.o

"\_OBJC\_CLASS\_\$\_RCTBundleURLProvider", referenced from:

objc-class-ref in AppDelegate.o

"\_OBJC\_METACLASS\_\$\_RCTAppDelegate", referenced from:

\_OBJC\_METACLASS\_\$\_AppDelegate in AppDelegate.o

ld: symbol(s) not found for architecture x86\_64

clang: error: linker command failed with exit code 1 (use -v to see invocation)

### Resolved by adding in podfile

```
installer.pods_project.build_configurations.each do |config|
  config.build_settings["EXCLUDED_ARCHS[sdk=iphonesimulator*]"] = "arm64"
end
```



```
Podfile

use_react_native!(
  :path => config[:reactNativePath],
  # Enables Flipper.
  #
  # Note that if you have use_frameworks! enabled, Flipper will not work and
  # you should disable the next line.
  :flipper_configuration => flipper_config,
  # An absolute path to your application root.
  :app_path => "#{Pod::Config.instance.installation_root}/.."
)

target 'MobileTests' do
  inherit! :complete
  # Pods for testing
end

post_install do |installer|
  installer.pods_project.build_configurations.each do |config|
    config.build_settings["EXCLUDED_ARCHS[sdk=iphonesimulator*]"] = "arm64"
  end
  # https://github.com/facebook/react-native/blob/main/packages/react-native/scripts/
  react_native_pods.rb#L197-L202
  react_native_post_install(
    installer,
    config[:reactNativePath],
    :mac_catalyst_enabled => false
  )
end
end
```

## Facing issue “error code 65” like Below

**error** Failed to build iOS project. We ran "xcodebuild" command but it exited with error code 65. To debug build logs further, consider building your app with Xcode.app, by opening UbiMobileApp.xcworkspace.

## Resolved by updating react native version in package.json and fresh npm install

"react-native": "0.73.2"

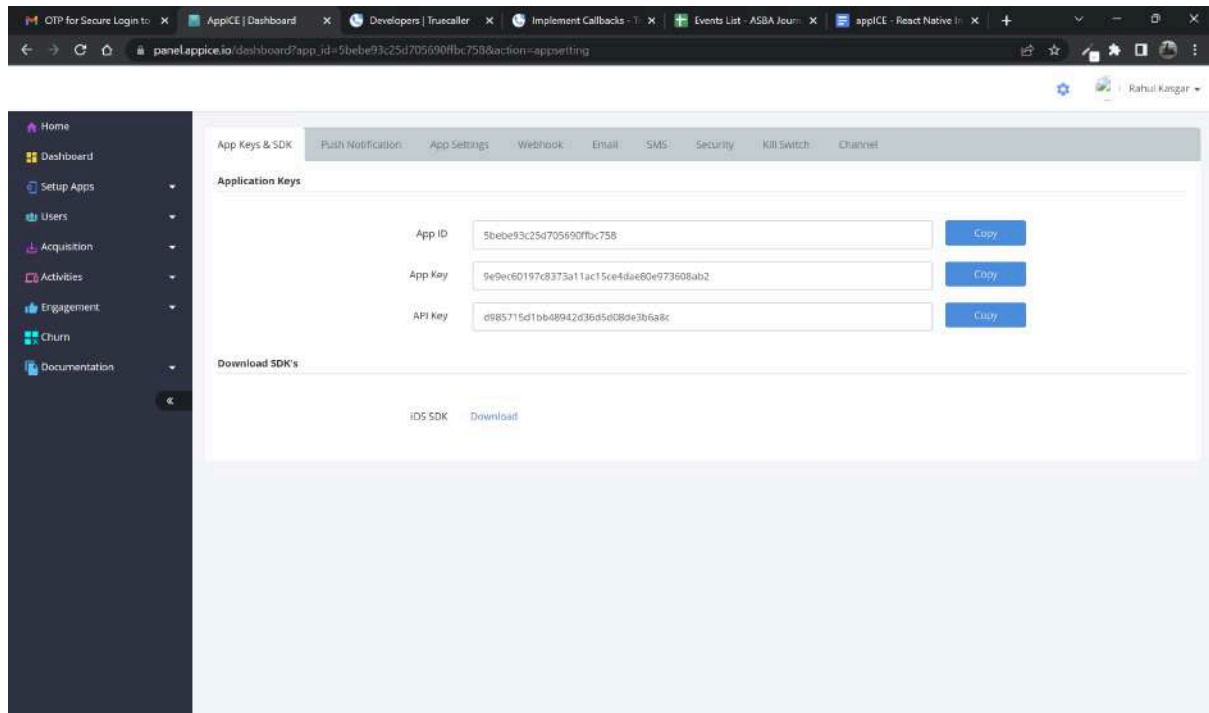
## React Native JavaScript

Once you follow above mentioned steps in your Android or iOS projects, open JAVASCRIPT file (say, App.js) of your project:

### Import Appice Plugin

```
import { AppiceReactPlugin } from 'reactnative-plugin-Appice';
```

### Get The Keys



Now Copy the keys and paste them into code to initialise the SDK

## Initialise SDK

```
if (Appice) {
  console.log('Appice is available');
  Appice.startContext(
    '6399cd6277e0a50e3a7f5a7e', //App id
    '0056374df1c9f17cf25c8dfc0ff17a8b1ed73cf6', //App Key
    '8fc46774553dca6939d02a6856900d63', //PI Key
    '',
    'US', //region
    'https://a.Appice.io', //base url
    certs,
  );
}
```

## Send Events

```
const sendEvent = () => {
  console.log('inside sendEvent');
  var dataObj = {
    IsBalanceLow: 'true',
    IsUserVerified: 'true',
    AppName: 'RN Sample App',
    User: 'Hello User How are you',
  }
```

```

        TestRun: 'This is test run',
    };
    Appice.tagEvent('Event Test', dataObj);
};

```

## Set User

```

const setUser = () => {
    Appice.setUser({
        [Appice.name]: 'testUserA1',
        [Appice.phone]: '123456',
        [Appice.email]: 'test@test.com',
        [Appice.age]: 23,
    });
};

```

## Get User

```

const getUser = () =>
{
    return Appice.getUser((user:any) => {});
}

```

This function will return an object that encapsulates the following properties:

```

String name;
String email;
String phone;
String gender;
long dob;
boolean isEmployed;
String employmentType;
String educationType;
boolean isMarried;
int age;
String[] TOP_N_PRODUCTS_VIEWED;
long FIRST_SEEN;
long LAST_SEEN;
String TOTAL_ORDER_VALUE

```

## SetUserId

```

const setUserID = () => {

```

```
Appice.setUserId(['1232323', 'kuOEN8bCqdrROmJPVCeKSg==']);  
};
```

### GetUserId

```
const getUserId = () => {  
  Appice.getUserId((userIds: any[]) => {});  
};
```

### AppInbox

type = { All = 1, UNREAD = 2, READ = 3, VIEWED = 4, DELETED = 5 }

### GetInboxMessages

```
const synchronizeInbox = () => {  
  Appice.synchronizeInbox(10, (success: boolean) => {  
    Appice.getInboxMessages(type, ['1232323'], (res:any) =>  
    {}  
    });  
  });  
}
```

### GetMessageCount

```
const getMessageCount = () =>  
{  
  Appice.getMessageCount(type, ['1232323'], (count: number) =>  
  {});  
};
```

### GetMessageForID

```
const getInboxMessageForId = () => {
```

```

    Appice.getInboxMessageForId(inputValue, ['1232323'],
(inboxMessage: any) => {});

};

```

## UpdateInboxMessage

```

const updateInboxMessage = () => {

    Appice.updateInboxMessage(mid, 3, "1232323", (isSuccess:
boolean) => {});

};

```

## AppInboxRichPush

AppInbox for RichPush with 4 apis :

```

- Appice.getMediaData(message, mediaKey, (mediaData: any) =>
{});
- Appice.getMediaUrl(message, mediaData, (mediaUrl:any) => {});
- Appice.getMediaType(message, mediaData, (mediaType:any) =>
{});
- Appice.getMediaThumbnail(message, mediaData, (
mediaThumbnail:any) => {});

```

## User Profile - getUser :

- getUser which has the userObject like userDetails, Demographic, Quick Actions such as most frequently visited events etc.

### 1. getUser - without UserId

```

- Appice.getUser((user:any) => {}

```

### Request of getUser:

```

    const getUser = () => {
    console.log('GET USER:');
    const userData = Appice.getUser((res: any)=>{
    console.log("Receive getUser "+userData);

    if(res){
    var response;
    const resData = JSON.stringify(res);

```

```

        console.log("res "+resData);

        const data = JSON.parse(resData);
        response = data;
        const customProperties = response[Appice.DEMOGRAPHIC_INFO];
        console.log("customProperties "+customProperties);
    }
    });
}

```

### Response of getUser:

```

'Receive getUser:', [ { g: 'M',
  d: 631152000000,
  edt: '12th',
  emt: 'part time',
  p: '9000900006',
  e: 'gopalk2020@test.com',
  em: false,
  a: 30,
  m: false,
  n: 'Arthe' } ]

```

## 2. getUserForIds

- with UserId and without UserId returning Array.

**Ex:** Appice.getUserForIds(['181166344'], (res:any) => {})

**Ex:** Appice.getUserForIds([''], (res:any) => {})

### Request of getUserForIds:

```

Appice.synchronizeData(10,(success: boolean) => {

```

```

Appice.getUserForIds(['181166344'], (res:any) => {
  console.log('Receive getUser:', res);

  for (var index = 0; index < res.length; index++) {
    let userData = res[index];
    console.log('Received getUser details:', userData);
    const name = userData[Appice.name];
    const phone = userData[Appice.phone];
    const gender = userData[Appice.gender];
    const dob = userData[Appice.dob];

```

```

const edt = userData[Appice.educationType];
const email = userData[Appice.email];
const employment = userData[Appice.employmentType];
const age = userData[Appice.age];
const marriage = userData[Appice.married];
const isEmployed = userData[Appice.isEmployed];
const customProperties = userData[Appice.DEMOGRAPHIC_INFO];
const customPropertiesObj = JSON.parse(customProperties);
console.log("user Profile, top_n_product_viewed =
"+customPropertiesObj[Appice.TOP_N_PRODUCTS_VIEWED]);
}
});
});
}

```

### Response of getUserForIds:

```

'Receive getUser:', { customProperties: '{\n "sb": "5000",\n "nc"
: 3,\n "cs": "Excellent",\n "pl": "Desktop",\n "tn": [\n "SupportVisit",\n
"ProfileVisit"\n ],\n "rc": "Referral456",\n "cb": "1000",\n "at": "200",\n "tt":
"Investments",\n "ft": "Monthly",\n "di": "0.2"\n}',
n: 'freddy smith',
g: 'Male',
d: 19851215,
edt: 'Master\'s',
p: '987-654-3210',
emt: 'Part-time',
e: 'fred@example.com',
em: true,
a: 35,
m: true }

```

**NOTE:** before calling getUserIds we have to synchronizeData to fetch the latest activities

**UserDetails** : such as Name, Phone, Gender, Dob, EducationType, Email, EmploymentType, Age, Married, IsEmployed.

**Demographic:** such as FIRST\_SEEN, LAST\_SEEN, TOP\_N\_PRODUCTS\_VIEWED, N\_COMPLAINTS\_RAISED, PREF\_LOGIN\_DEVICE, REFERRAL\_CAMPAIGN, TOTAL\_ORDER\_VALUE, ADD\_TO\_CART\_N\_DAYS, CREDIT\_SCORE, DEBT\_TO\_INCOME\_RATIO, SAVINGS\_BALANCE, CHECKING\_BALANCE

## isDeviceReady:

Handle Push DeepLink in all 3 state(FG ,BG, Kill state)

- *Appice.isDeviceReady(true)* - when the App is in active state.
- *Appice.isDeviceReady(false)* - when the App is in background state.

## PageView Event:

PageView Event is triggered when a screen is displayed.

1. When the user Views the Page/Screen

```
var pageViewAttributes = {"view": "Emi & More" };  
Appice.tagEvent("PageView", pageViewAttributes);
```

2. When the user Leaves the Page/Screen

```
var pageDurationAttributes = {"name": "Emi & More",  
                             "duration":durationInSeconds };  
Appice.tagEvent("PageDuration", pageDurationAttributes);
```

## **Example for pageViewEvent**

```
import { useState, useEffect, useRef } from 'react';  
import { AppState, AppStateStatus } from 'react-native';  
import Appice from 'reactnative-plugin-Appice';  
  
export const usePageTracking = (pageName: string, navigation: any) => {  
  const [startTime, setStartTime] = useState<number | null>(null);  
  const appState = useRef<AppStateStatus>(AppState.currentState);  
  const isPageFocused = useRef(false); // Track if the page is currently focused  
  const isForeground = useRef(true); // Track if the app is in the foreground  
  useEffect(() => {  
    // Function to trigger PageView event  
    const triggerPageView = () => {  
      if (!isPageFocused.current) { // Ensure the PageView is triggered only once  
        setStartTime(Date.now());  
        const dataObj = { view: pageName };  
        console.log(`PageView event triggered for page: ${pageName}`);  
        Appice.tagEvent('PageView', dataObj);  
        isPageFocused.current = true;  
      }  
    }  
  });  
}
```



```

    }
  };
  // Function to trigger PageDuration event
  const triggerPageDuration = () => {
    if (isPageFocused.current && startTime) {
      const endTime = Date.now();
      const totalSeconds = Math.floor((endTime - startTime) / 1000);
      const dataObj = { name: pageName, duration: totalSeconds };
      console.log(`PageDuration event triggered for page: ${pageName}, duration:
${totalSeconds}s`);
      Appice.tagEvent('PageDuration', dataObj);
      setStartTime(null); // Reset the start time
      isPageFocused.current = false;
    }
  };
  // Handle focus and blur events
  const handleFocus = () => {
    console.log(`Page ${pageName} focused`);
    if (isForeground.current) {
      triggerPageView(); // Trigger PageView only when the app is in the foreground
    }
  };
  const handleBlur = () => {
    console.log(`Page ${pageName} lost focus`);
    triggerPageDuration(); // Trigger PageDuration when leaving the page
  };
  const unsubscribeFocus = navigation.addListener('focus', handleFocus);
  const unsubscribeBlur = navigation.addListener('blur', handleBlur);
  // Handle app state changes (background/foreground)
  const handleAppStateChange = (nextAppState: AppStateStatus) => {
    console.log(`AppState changed from ${appState.current} to ${nextAppState}`);
    if (appState.current.match(/active|inactive/) && nextAppState === 'background') {
      triggerPageDuration(); // App went to background
      isForeground.current = false;
    } else if (appState.current === 'background' && nextAppState === 'active') {
      isForeground.current = true;
      if (isPageFocused.current) {
        console.log(`Re-triggering PageView for page: ${pageName} after returning
from background`);
        triggerPageView(); // Re-trigger PageView when coming back to foreground
      }
    }
    appState.current = nextAppState; // Update app state
  };

```

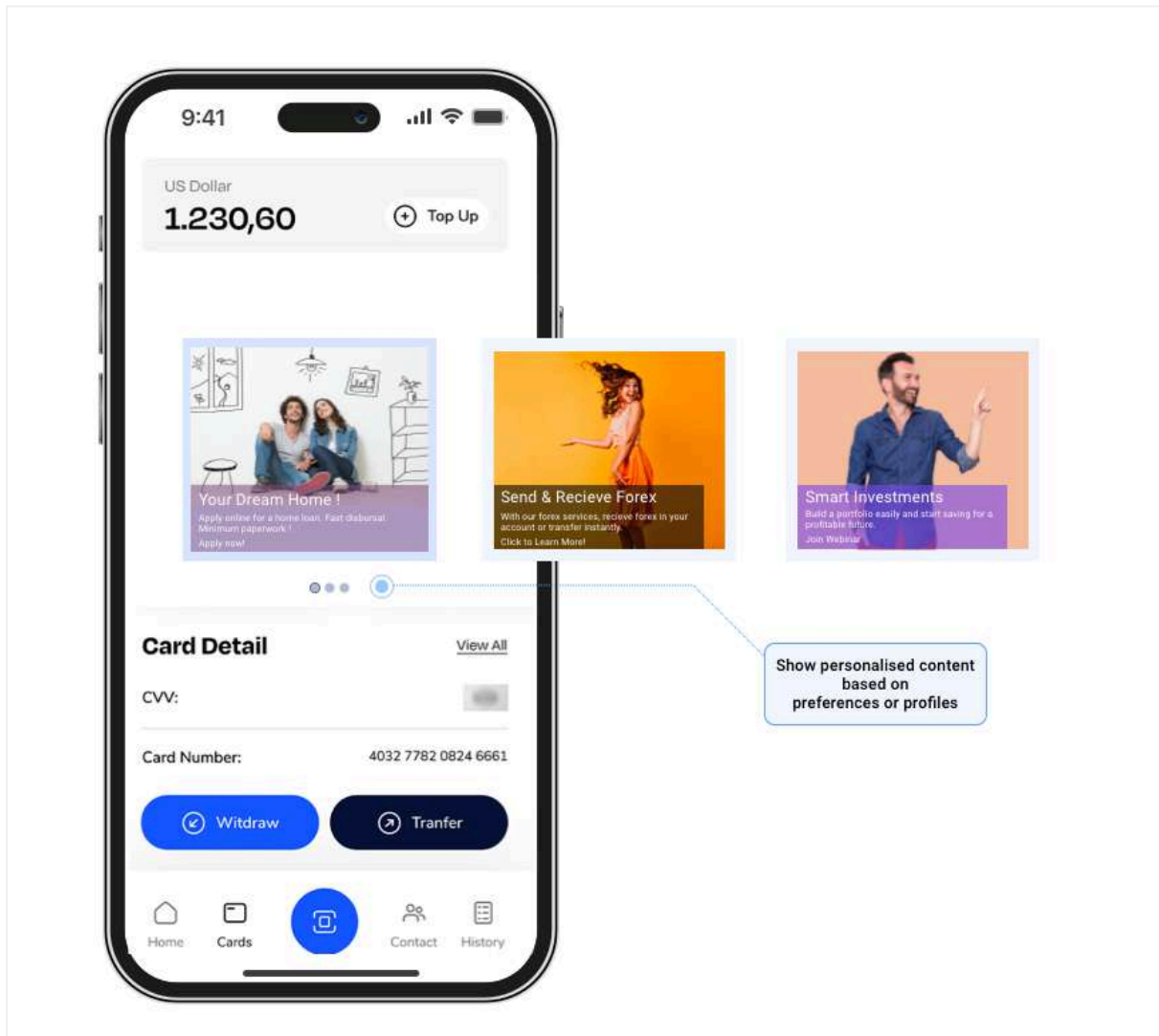
```
const appStateListener = AppState.addListener('change',
handleAppStateChange);
// Cleanup listeners on unmount
return () => {
  unsubscribeFocus();
  unsubscribeBlur();
  appStateListener.remove();
  triggerPageDuration(); // Trigger duration if unmounted
};
}, [navigation, pageName, startTime]);
};
```

**Calling function as**

```
usePageTracking('Emi & More', navigation);
```

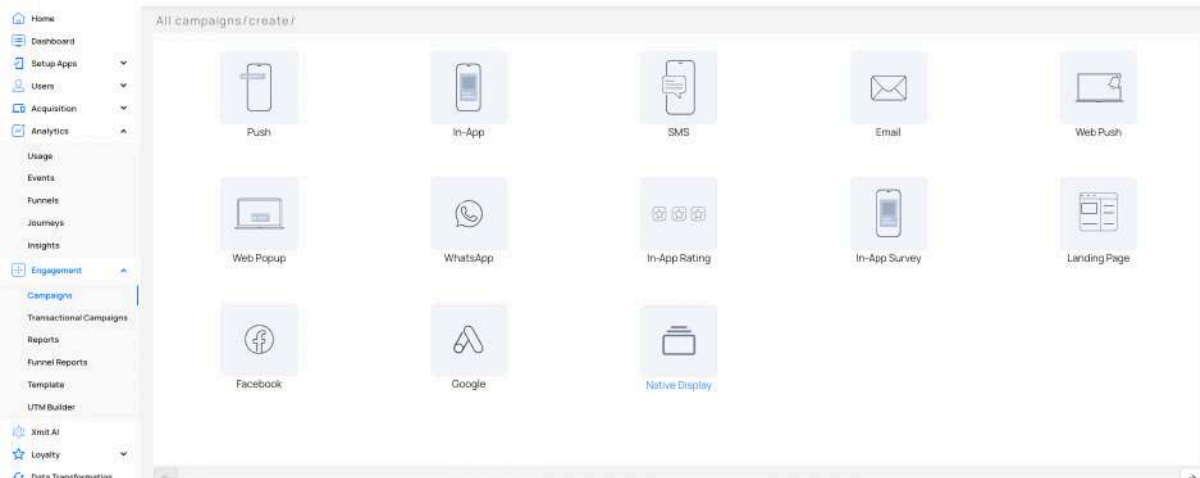
## Native Display

Native Display helps to display content natively within your app - deliver relevant and contextual content based on customer preferences or profile.



## Steps to setup Native Display

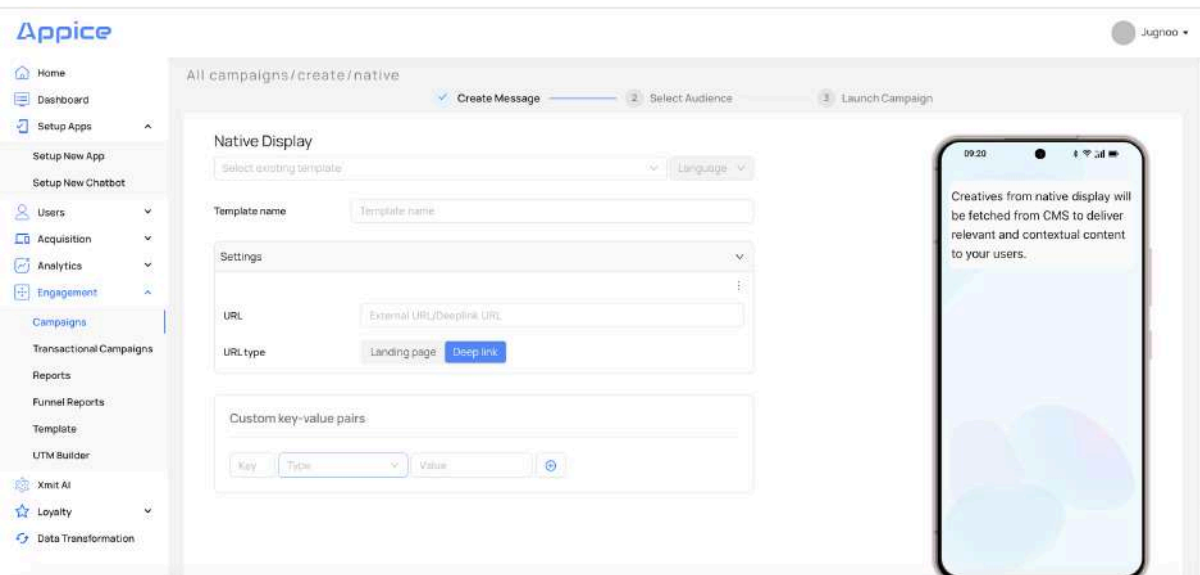
### Select the option of Native Display from Campaigns:



### Populate the values:

The custom key-value can have any value.

For example, the key could be the title and value could be the header of the content to be displayed. Or you can take Offerid as the key with a value of the particular offer. The app can take the Offerid and fetch the additional details of the offer from a backend system.



Select the Audience that this will be targeted to:

The screenshot shows the 'Select Audience' step in the campaign creation process. At the top, there are three progress indicators: 'Create Message' (checked), 'Select Audience' (checked), and 'Launch Campaign' (unchecked). Below these, there is a search bar and a 'Create New Audience Segment' button. A list of audience segments is displayed, with 'VIVO Y50 version device' selected and highlighted in blue. Other segments include 'CHNaid', 'SAMSUNG F42 v13', 'OppoA7 pu', 'ONEPLUS 11R version', 'GOOGLEPixel 6 npush', and 'GOOGLEPixel 6'.

Launch the Campaign:

The screenshot shows the 'Launch Campaign' step in the campaign creation process. At the top, there are three progress indicators: 'Create Message' (checked), 'Select Audience' (checked), and 'Launch Campaign' (checked). Below these, there are several form fields: 'Campaign Name' (NativeCampaignDisplay01), 'Delivery Type' (Direct Push), 'Category' (Engagement), 'Data Range' (2024-09-18 to 2024-09-19), 'Display' (Now), 'Conversion' (Yes), and 'Live Event' (True). There is also a 'Frequency Capping' section at the bottom.

Rendering the Campaign:

The responsibility for rendering lies with the App

## Integration Steps

**getCampaigns**(type: string):

Fetches all campaigns of a specific type (e.g., "NATIVE", "IN-APP")

```
const getCampaigns = () => {
  const type = AppICE.CampaignType.NATIVE;
  AppICE.getCampaigns(type, (campaigns: any[]) => {
    campaigns.forEach((campaign) => {

      console.log('getCampaigns of Campaign ID:', campaign.getCampaignId());
      console.log('Action URL is :', campaign.getActionUrl());
      console.log('Action Type is :', campaign.getActionType());
      console.log('Custom Data is :', campaign.getCustomData());
    });
  });
};
```

```
};
```

**getCampaignById**(cmpId: string):

Retrieves a campaign by its ID. Uses a generic getProperty method to access campaign properties like ID, action URL etc

```
const getCampaignById = () => {  
  const cmpId = '1234';  
  AppICE.getCampaignById(cmpId, (campaign: any) => {  
  
    const campaignId = campaign.getCampaignId();  
    const actionUrl = campaign.getActionUrl();  
    const actionType = campaign.getActionType();  
    const customData = campaign.getCustomData();  
  
  });  
};
```

## Example Code for App.js

Open App.js and add the code below

```
import React from 'react';  
import {  
  Alert,  
  Text,  
  Platform,  
  Image,  
  Linking,  
  Button,  
  View,  
} from 'react-native';  
  
const Appice = require('reactnative-plugin-Appice');  
const HelloWorldApp = () => {  
  
  const certs = [];  
  if (Appice) {  
  
    console.log('Appice is available');  
    Appice.startContext(  

```

```

    '5bebe93c25d705690ffbc758', //appid
    '9e9ec60197c8373a11ac15ce4dae80e973608ab2', //app key
    '844770e0edf3b795cd7508fa71789572', //api key
    '',
    'US', //region
    'https://a.Appice.io', //base url

    certs,
  );
}
addAppiceAPIListeners();
return (
  <View
    style={{
      flex: 1,
      justifyContent: 'center',
      alignItems: 'center',
      marginBottom: 10,
    }}>
    <Text>Appice React Demo</Text>
    <Button onPress={sendEvent} title="Send Event1"
color="#841584" />
    <View />

    <Button onPress={sendEvent2} title="Send Event 2"
color="#841584" />

    <Button
      onPress={setUserProfile}
      title="set User Profile"
      color="#841584"
    />

    <Button
      onPress={setSampleUserID}
      title="set Sample User ID"
      color="#841584"
    />
    <Button
onPress={setLanguage}
title="set Language"
color="#841584"
/>

    <Button
onPress={sendView}
title="Send View"
color="#841584"

```

```

/>
<Button
onPress={getUserId}
title="getUserId"
color="#841584"
/>
<Button
onPress={getInboxMessage}
title="getInboxMessage"
  color="#841584"
/>
<Button
onPress={getMessageCount}
title="getMessageCount"
color="#841584"
/>
<Button
onPress={getInboxMessageId} title="getInboxMessageId"
  color="#841584"
/>
<Button
onPress={updateInbox}
  title="updateInbox"
color="#841584" />
  </View>
<Button
onPress={getUser}
  title="getUser"
color="#841584" />

<Button
onPress={getCampaignById}
title="getCampaignById"
color="#841584" />

<Button
onPress={getCampaigns}
title="getCampaigns"
color="#841584" />

);
};

function addAppiceAPIListeners() {

console.log('handleAppiceTapEvent');

```



```

    Appice.addListener(Appice.AppicePushNotificationClicked,
(event : any) => {
    if (typeof event !== 'undefined'){
        console.log('handleAppiceTapEvent',
Appice.AppicePushNotificationClicked,
event.AppicePushNotificationClicked);

Appice.pushNotificationClicked(event.AppicePushNotificationCli
cked);

        //fetching custom data from push payload
        var data =
Appice.getCustomDataFromPayload(event.AppicePushNotificationCl
icked);

        console.log(data);
    }
});
Appice.addListener(Appice.AppiceInAppClicked, (event: {
AppiceInAppClicked: any; }) => {
    if (typeof event !== 'undefined') {
        console.log("event AppiceInAppClicked: " + event);
console.log("event is from AppiceInAppClicked is " +
event.AppiceInAppClicked);
console.log('handleAppiceTapEvent',
Appice.AppiceInAppClicked);

        callbackInApp(event.AppiceInAppClicked)
    }
});
}

function callbackInApp(payload: any) {

    console.log('Call Back payload:', payload);

    if (payload) {
        const screen = JSON.parse(payload)
        console.log('Call BACK Screen:',screen);
    }

};

const sendEvent = () => {
    console.log('inside sendEvent 1');
    var dataObj = {
        IssueType: 'IPO',

```

```

    ApplyForIPO: 'true',
    FailureMessage: ' PAN not linked',
    ApplyPAN: 'true',
    LinkPAN: 'false',
    BankAccount: 'BOI',
    DematAccount: 'NON BOI',
  };
  Appice.tagEvent('ReactSampleApp', dataObj);
  console.log(JSON.stringify(dataObj));
  //Recording an Event
};

const sendEvent2 = () => {
  console.log('inside sendEvent 2');
  var dataObj = {
    BidAmount: 'IPO',
    Quantity: 'true',
    TotalValue: ' PAN not linked',
    HighBidValue: 'true',
    InvestorCategory: 'false',
  };
  Appice.tagEvent('ReactSampleApp', dataObj);
  console.log(JSON.stringify(dataObj));

  //Recording an Event
};

const setUserProfile = () => {
  Appice.setUser({
    [Appice.name]: 'testUserA1',
    [Appice.phone]: '123456',
    [Appice.email]: 'test@test.com',
    [Appice.age]: 23,
  });
};

const setSampleUserID = () => {
  Appice.setUserId(['1232323', 'kuOEN8bCqdrROmJPVCeKSg==']);
};

const setLanguage = () => {
  Appice.setCustomVariable('_lang', 'ar-AE');
};

const getUserId = () => {
  Appice.getUserId((userIds: any[]) => {
    console.log('User IDs:', userIds);
  });
};

```

```

        const inboxMessageString = JSON.stringify(userIds, null,
2);

        Alert.alert(`getInboxMessageForId: \n
${inboxMessageString}`);
    });
};

/*-----
getInboxMessages
-----*/
const synchronizeInbox = () => {
    Appice.synchronizeInbox(10, (success: boolean) => {
        const userIds = ['1232323'];
        Appice.getInboxMessages(type, userIds, (res:any) => {
            var inboxVar = [Appice.INBOX_MESSAGE];
            for (var index = 0; index < res.length; index++) {
                let inboxmsg = res[index];
                console.log('Received Inbox:', inboxmsg);
                const messageIds: string[] =
inboxmsg[Appice.INBOX_MESSAGE];
                console.log('Message inbox is:', messageIds);

                const inboxMessageString = JSON.stringify(res, null, 2);
            }
        });
    });
}

/*-----
getMessageCount
-----*/
const getMessageCount = () =>
{
    Appice.synchronizeInbox(timeout, (res)=>{
        if(res){

Appice.getMessageCount(type, userIds, (count: number) => {
            console.log('Messages count :', count);
            Alert.alert(`Messages count : \n ${count}`);
        });
    }else{
        console.log("inbox is not synced with server, showing old
stored data")
        Appice.getMessageCount(type, userids, (res)=>{

```

```

        if(res){
            return res;
        }
    });
}
});
}

/*-----
getInboxMessageForId
-----*/

const getInboxMessageForId = () => {
    const messageId = "8ce910b0-9e21-4782-801d-b14e60ae174e";
    const userIds = ['1232323' , 'kuOEN8bCqdrROmJPVCeKSg=='];

    Appice.getInboxMessageForId(messageId, userIds,
(inboxMessage: any) => {
        console.log('getInboxMessageForId in JS:',
inboxMessage);
    });
};

/*-----
UpdateInboxMessage
* type      = { All = 1, UNREAD = 2, READ = 3, VIEWED = 4,
DELETED = 5 },
-----*/
const updateInboxMessage = () => {
    const messageId = "96016d77-720b-4b12-a8ae-5a1e7b7efdcb";
    const userId = "1232323";
    Appice.updateInboxMessage(messageId, 3, userId,
(isSuccess: boolean) => {
        console.log('Update Inbox Message Success:', isSuccess);
    });
};

/*-----
AppInbox RichPush
-----*/
function getMediaDataKeys(inboxMessages: any) {

    const newInboxMessageData: any[] = [];
    // Iterate over all messages in inboxMessages array
    inboxMessages.forEach((message: any) => {
        const mediaKey = 'imageUrl'; // Adjust media key as needed
        const singleInboxMessage = message;

```

```

    Appice.getMediaData(message, mediaKey, async (mediaData:
any) => {
        console.log('Media Data:', mediaData);
        // Use the mediaData value within the callback scope
        Appice.getMediaUrl(message, mediaData, (mediaUrl: any)
=> {
            console.log('getMediaUrl is:', mediaUrl);
            singleInboxMessage.mediaUrl = mediaUrl;

        });
        Appice.getMediaType(message, mediaData, (mediaType:
any) => {
            console.log('getMediaType is:', mediaType);
            singleInboxMessage.mediaType = mediaType;
        });
        Appice.getMediaThumbnail(message, mediaData,
(mediaThumbnail: any) => {
            console.log('Media Thumbnail:', mediaThumbnail);
            singleInboxMessage.mediaThumbnail = mediaThumbnail;
        });
        newInboxMessageData.push(singleInboxMessage);
        console.log('Media newInboxMessageData pushh:',
newInboxMessageData);
    });
});
// Wait for all asynchronous operations to complete before
returning
return new Promise((resolve) => {
    setTimeout(() => {
        resolve(newInboxMessageData);
        console.log('Media newInboxMessageData promise:',
newInboxMessageData);
    }, 500); //millisecond

});
}

```

```

/*-----
getUser
-----*/
const getUser = () => {
    console.log('GET USER:');
    Appice.getUser((user:any) => {
        // Handle the user object received from the native side.
    })
}

```

```

        console.log(user[Appice.name]);
        console.log(user[Appice.email]);
        console.log(user[Appice.phone]);
        console.log(user[Appice.gender]);
        console.log(user[Appice.dob]);
        console.log(user[Appice.isEmployed]);
        console.log(user[Appice.employmentType]);
        console.log(user[Appice.educationType]);
        console.log(user[Appice.isMarried]);
        console.log(user[Appice.age]);
        var jsonVal:any;
    if (user[Appice.DEMOGRAPHIC_INFO])
    {
        jsonVal= JSON.parse(Appice.DEMOGRAPHIC_INFO);
        console.log('Tota Order:',jsonVal
[Appice.TOTAL_ORDER_VALUE]);
const topnProd = Appice.TOP_N_PRODUCTS_VIEWED;
        console.log('Top N prod:',jsonVal[topnProd]);
    }

    });
};

/*-----
getUserForIds
-----*/

const getUserData = () => {
    Appice.synchronizeData(10,(success: boolean) => {
        console.log('Profile synchronization success:', success);
        const userIds = ['160840156'];
        Appice.getUserForIds(userIds, (res:any[]) => {
            console.log('Receive getUser:', res);
            for (var index = 0; index < res.length; index++) {
                let userData = res[index];
                const customProperties =
userData[Appice.DEMOGRAPHIC_INFO];
                const customPropertiesObj =
JSON.parse(customProperties);
                console.log("user Profile, top_n_product_viewed =
"+customPropertiesObj[Appice.TOP_N_PRODUCTS_VIEWED]);

            }
        });
    });
};
}

```

```

/*-----
isDeviceReady
-----*/

if (nextAppState === 'active') {
  Appice.isDeviceReady(true);
}
else if (nextAppState === 'background'){
  Appice.isDeviceReady(false);
}

/*-----
getCampaigns
-----*/

const getCampaigns = () => {

  const type = AppICE.CampaignType.NATIVE;

  AppICE.getCampaigns(type, (campaigns: any[]) => {
    console.log('Campaigns:', campaigns);

    campaigns.forEach((campaign) => {

      console.log('getCampaigns of Campaign ID:',
        campaign.getCampaignId());
      console.log('Action URL is :',
        campaign.getActionUrl());
      console.log('Action Type is :',
        campaign.getActionType());
      console.log('Custom Data is :',
        campaign.getCustomData());

    });
  });
};

const getCampaignById = () => {

  const cmpId = '1234';
  AppICE.getCampaignById(cmpId, (campaign: any) => {

    console.log('getCampaignById of Campaign ID:',
      campaign.getCampaignId());
  });
};

```

```

        console.log('Action URL is :',
campaign.getActionUrl());
        console.log('Action Type is :',
campaign.getActionType());
        console.log('Custom Data is :',
campaign.getCustomData());

    });
};

```

```
export default HelloWorldApp;
```

## Setup Push Notification for Android

In order to setup push we need to Integrate Firebase in our application

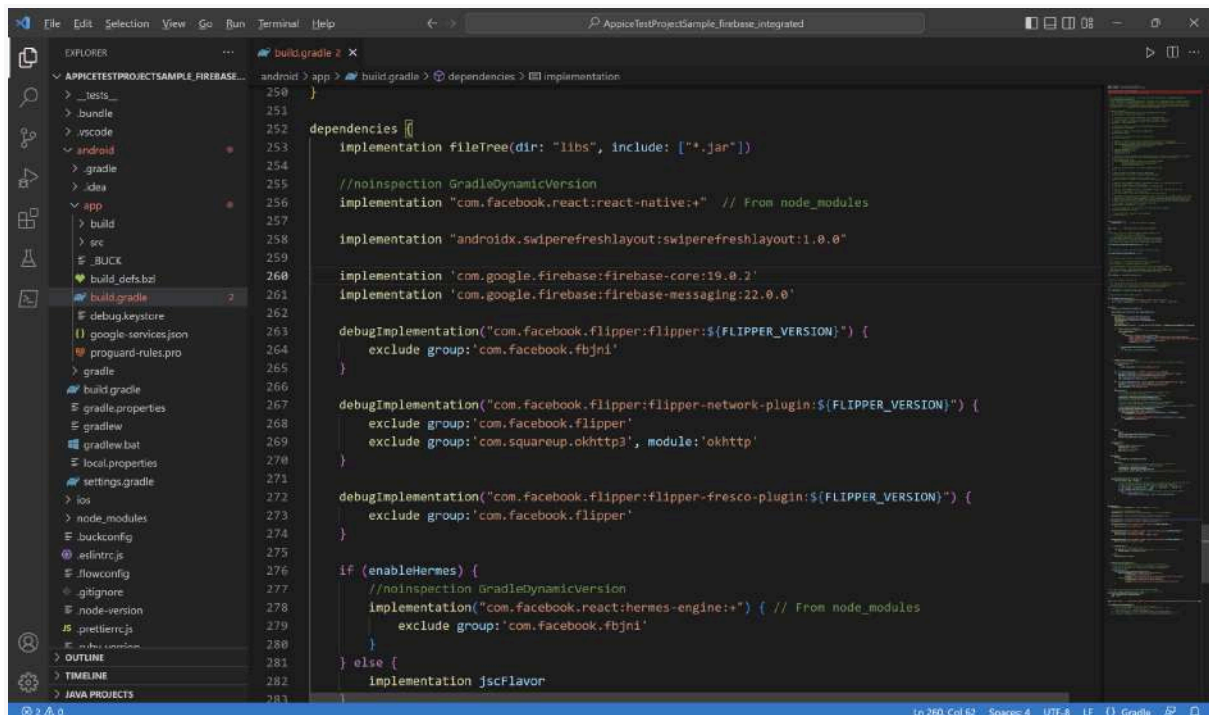
Add Firebase dependencies in Android build.gradle file

1. Open android/app/build.gradle and add

```

implementation 'com.google.firebase:firebase-core:19.0.2'
implementation 'com.google.firebase:firebase-messaging:22.0.0'

```



2. Apply Plugin



```
apply plugin: 'com.google.gms.google-services'
```

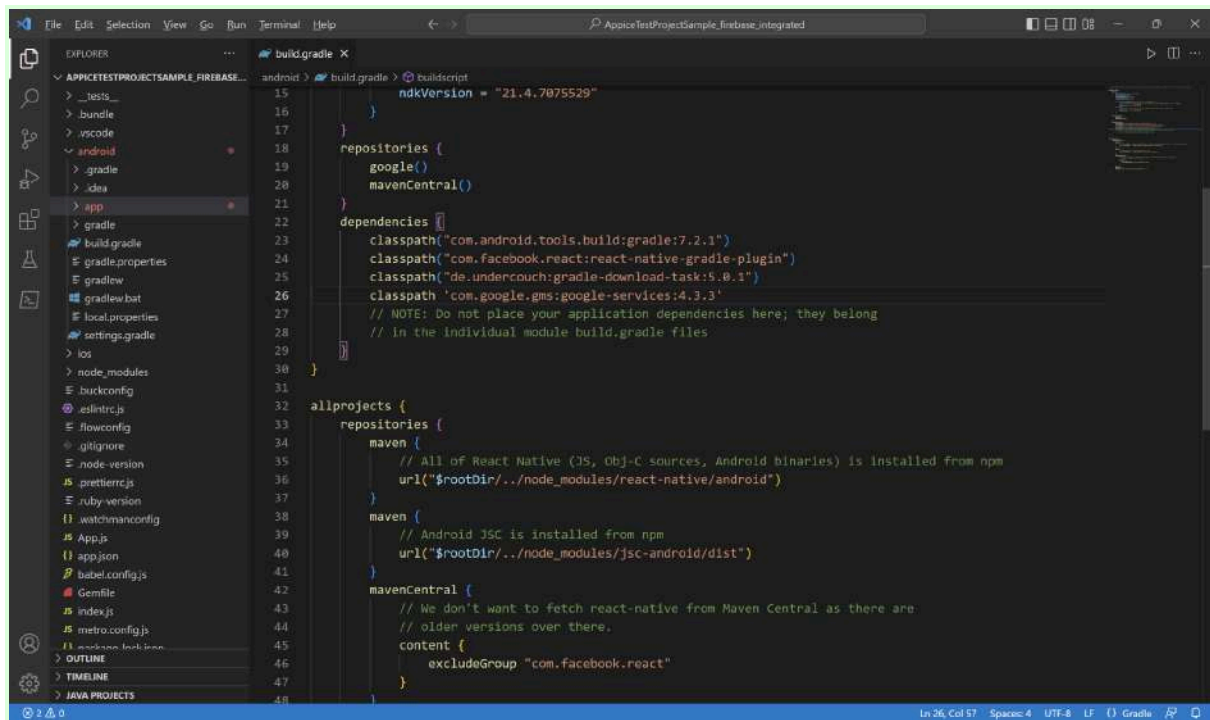


*Project/android/app/google-services.json*



Open android/build.gradle and add

```
classpath 'com.google.gms:google-services:4.3.3'
```



5. Add SDKFcmListener Service in android's manifest

open Android Manifest and add following service

```
\android\app\src\main\AndroidManifest.xml
```

```
<service
```

```
android:name="semusi.ruleengine.pushmanager.SdkFcmListenerService"
```

```
android:exported="false"
```

```
android:protectionLevel="signature">
```

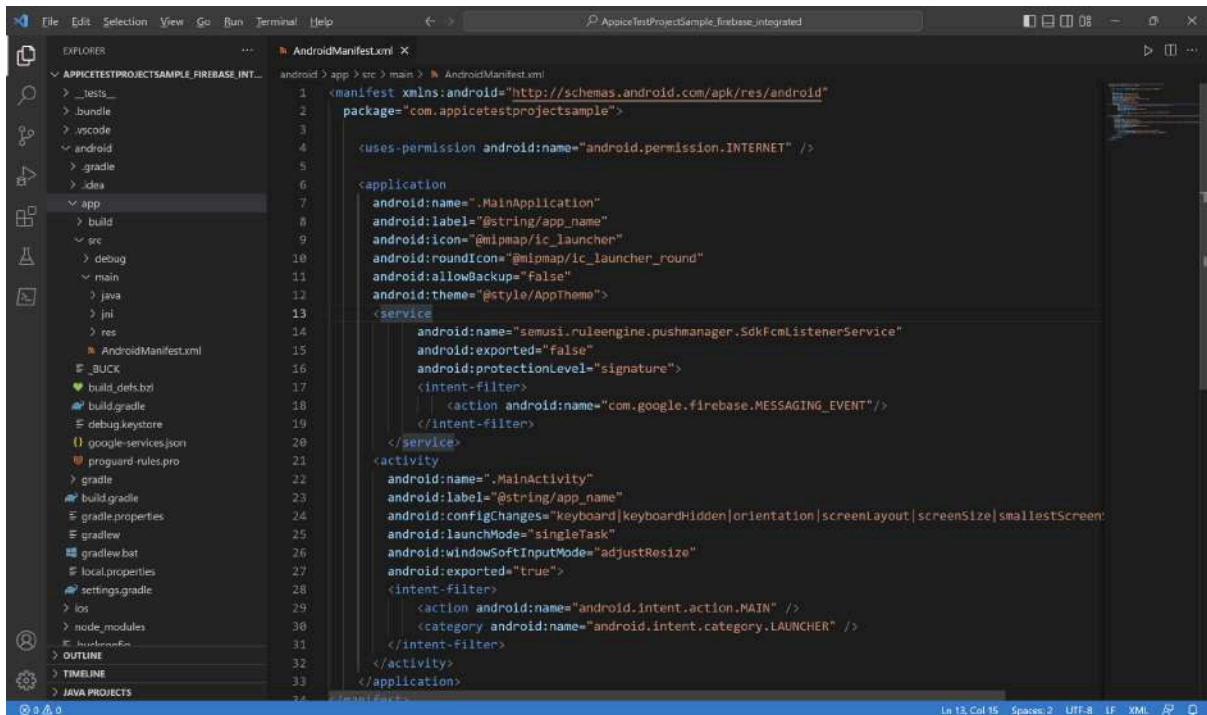
```
<intent-filter>
```

```
<action
```

```
android:name="com.google.firebase.MESSAGING_EVENT"/>
```

```
</intent-filter>
```

```
</service>
```



## IOS App side Changes

### Setup Push Notification

#### 1. App Side changes:

Add below code in **AppDelegate.h** file:

```
#import<UserNotifications/UserNotifications.h>
```

```
@interface AppDelegate :
```

```
RCTAppDelegate<UIApplicationDelegate, RCTBridgeDelegate, UNUserN  
otificationCenterDelegate>
```

```
@end
```

Add below code in **AppDelegate.m** file:

```
#import "AppiceReactPlugin.h"
```

```
- (BOOL)application:(UIApplication *)application  
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions  
{  
    center = [UNUserNotificationCenter  
currentNotificationCenter];  
    center.delegate = self;  
}
```

```

- (void)application:(UIApplication*)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData*)deviceToken
{
    NSLog(@"token is : %@",deviceToken);
    [AppiceReactPlugin
didRegisterForRemoteNotificationsWithDeviceToken:deviceToken];
}

- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
{
    NSLog(@"%@ = %@", NSStringFromSelector(_cmd), error);
    NSLog(@"Failed to get token, Error: %@", error);
}

- (void)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo
fetchCompletionHandler:(void
(^) (UIBackgroundFetchResult))completionHandler{
    NSLog(@"received payload =%@",userInfo);
    [AppiceReactPlugin pushNotificationReceived:userInfo];
}

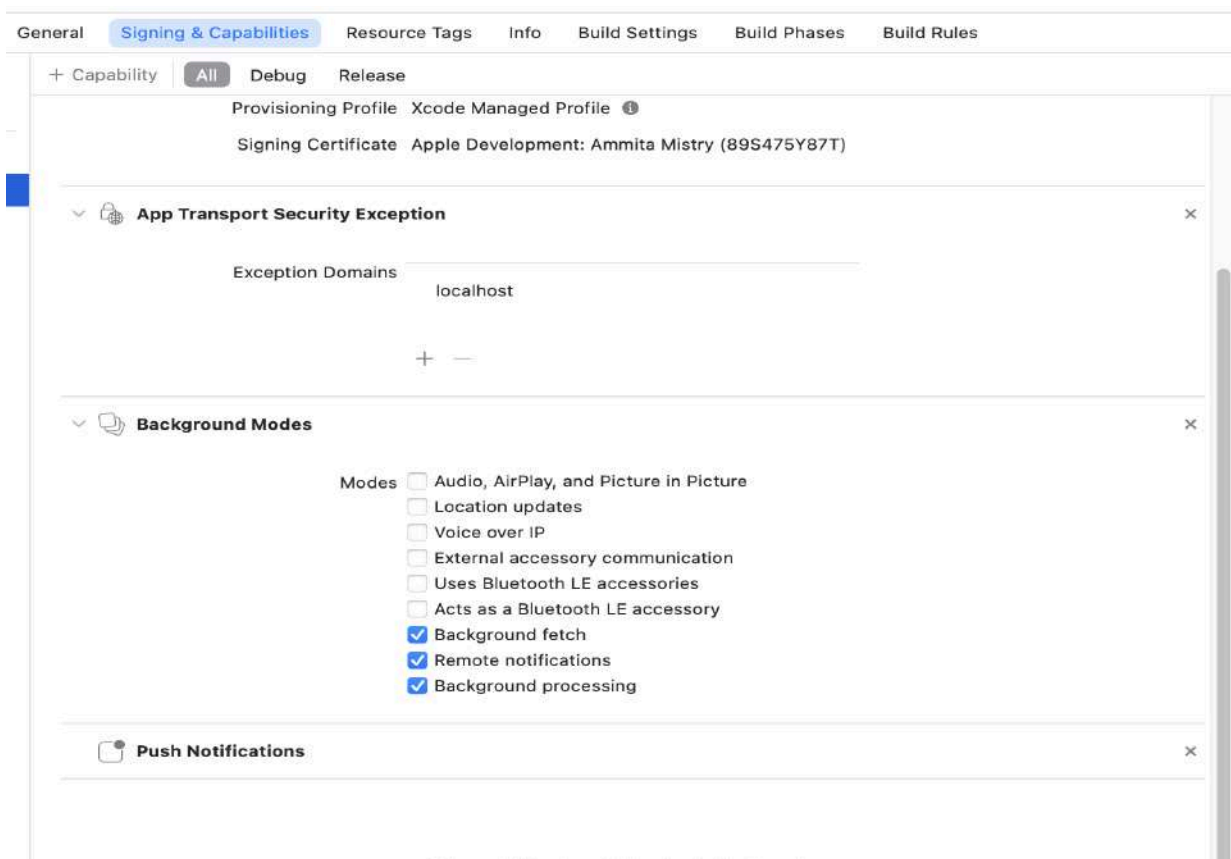
- (void)userNotificationCenter:(UNUserNotificationCenter
*)center
didReceiveNotificationResponse:(UNNotificationResponse
*)response withCompletionHandler:(void
(^) (void))completionHandler {

    [AppiceReactPlugin
pushNotificationClicked:response.notification.request.content.
userInfo];

    completionHandler();
}

```

2. Enable PushNotification and Background Modes as Below ScreenShot



## Setup DeepLink for url

### AppDelegate.m

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url
options:(NSDictionary<UIApplicationOpenURLOptionsKey,id>
*)options {

    return [RCTLinkingManager application:app openURL:url
sourceApplication:options[UIApplicationOpenURLOptionsSourceApp
licationKey]
annotation:options[UIApplicationOpenURLOptionsAnnotationKey]];
}

// Universal links
- (BOOL)application:(UIApplication *)application
continueUserActivity:(NSUserActivity *)userActivity
restorationHandler:(void
(^)(NSArray<id<UIUserActivityRestoring>> *
_Nullable))restorationHandler {
return [RCTLinkingManager application:application
continueUserActivity:userActivity
```

```
restorationHandler:restorationHandler];  
}
```

## Encryption/ Non-Encryption

### Android:

In the Application side of Manifest.xml add the below TAG

```
<meta-data  
    android:name="APPICE_ENCRYPTION"  
    android:value="1" />
```

0 Encryption is disabled

1 Encryption is enabled

### iOS:

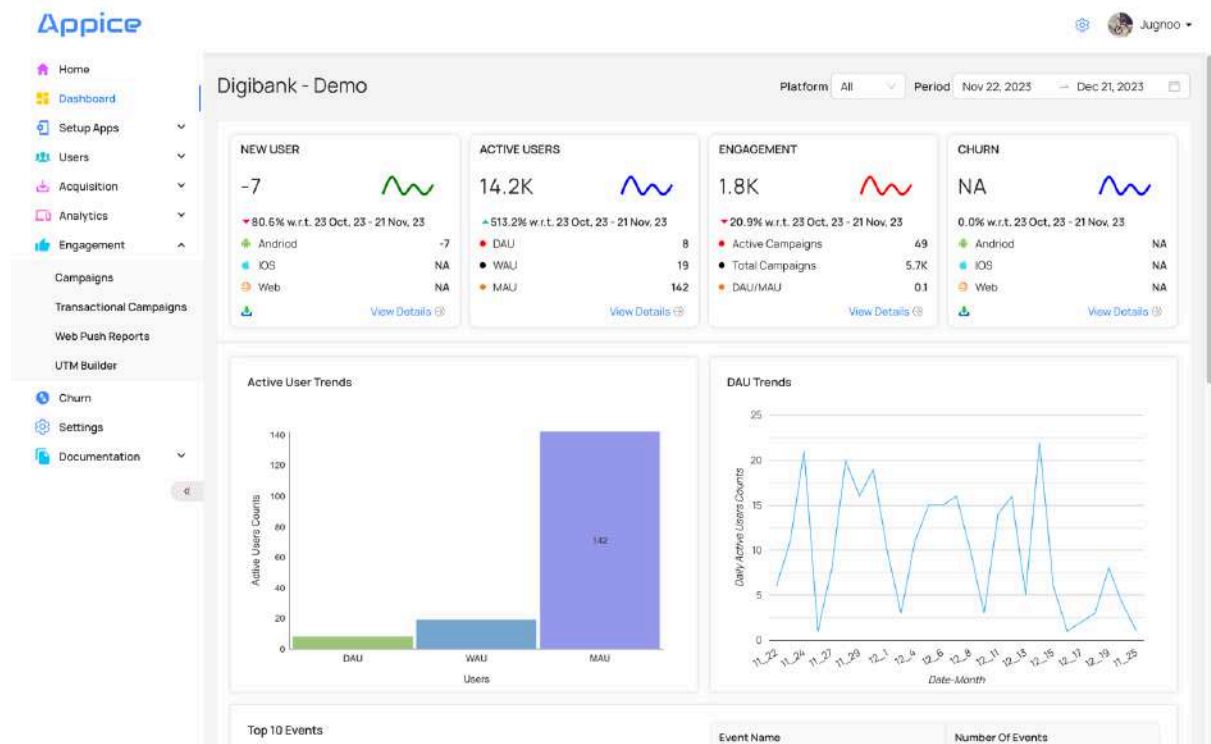
In the **Info.plist** of your iOS project, you can add the following key to control session encryption:

1. Open your project's **Info.plist** file.
2. Add a new key named **SESSION\_ENCRYPTION** with the value type set to **Boolean**.
3. Set the value to **YES** to enable encryption or **NO** to disable encryption.

**SESSION\_ENCRYPTION - Boolean - YES**

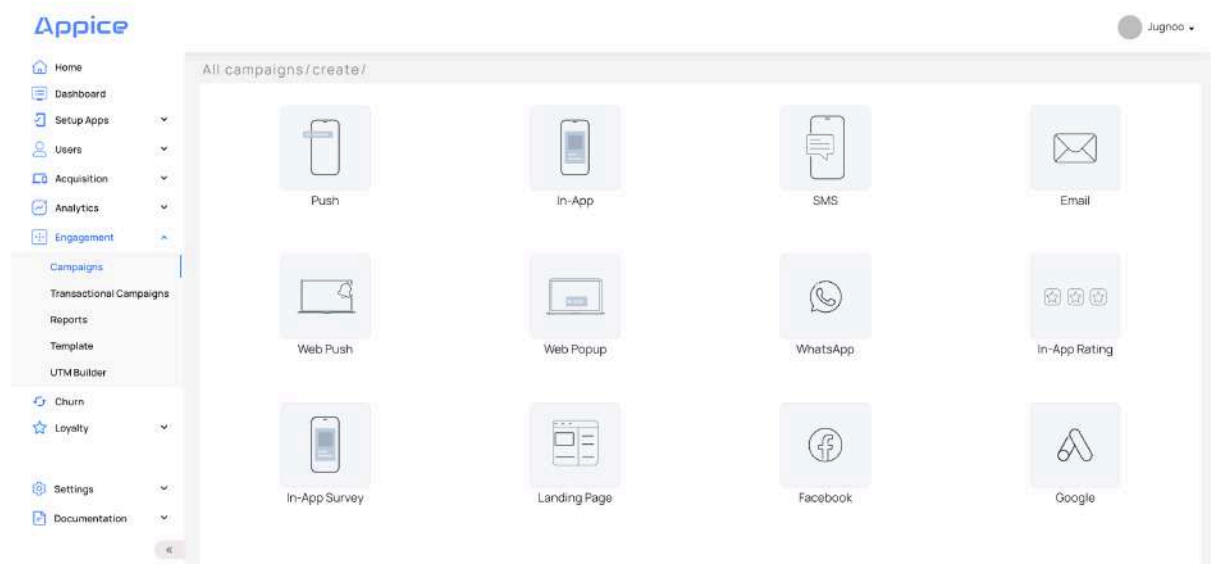
## Verify Push Notifications

<https://panel.Appice.io/login>



Now Select your application and go to

Engagement > Campaigns > Create New Campaign > PUSH NOTIFICATION



Select from existing template, select details and click on Update button:



Appice

All campaigns/create/push

1 Create Message 2 Select Audience 3 Launch Campaign

Push campaign

UBI Platinum Card

Variant 1 x Hindi

Template name: UBI Platinum Card

Header: Amazing offer! Platinum Card (28 / 50)

Description: Invest and save with our New Platinum Card offer 🎉 (53 / 100)

Expanded text: Yes No Expanded image: Yes No

Buttons: Yes No

Advance settings

URL: External URL/DeepLink: URL

URL type: Landing page Deep link

Special Attributes: Sound Yes No Vibrate Yes No Increment badge Yes No

Update Close

Mobile Preview: Digibank - Demo, Amazing offer! Platinum Card, Invest and save with our New Platinum Card offer 🎉

Android iOS

Click on Create New Audience Segment and fill details according to your need.

Appice

All campaigns/create/in-app

1 Create Message 2 Select Audience 3 Launch Campaign

Create New Audience Segments

Search

who what	
who what test	
mmmm	
Google Play	
Flamingo What's App	
Android iOS N Phase 23 July	
test23civ	
ph test	
phone test	
test test	
civ test	
test test	
motorola new	
test geo jul 23	

In this example we are targeting an app to specific device id for that we will need to enter our device id and click on Get Reach Count to see the reachable count for the device.

Enter segment name and description to save

Click on Save button to go next



Select your newly created Audience segment and click next

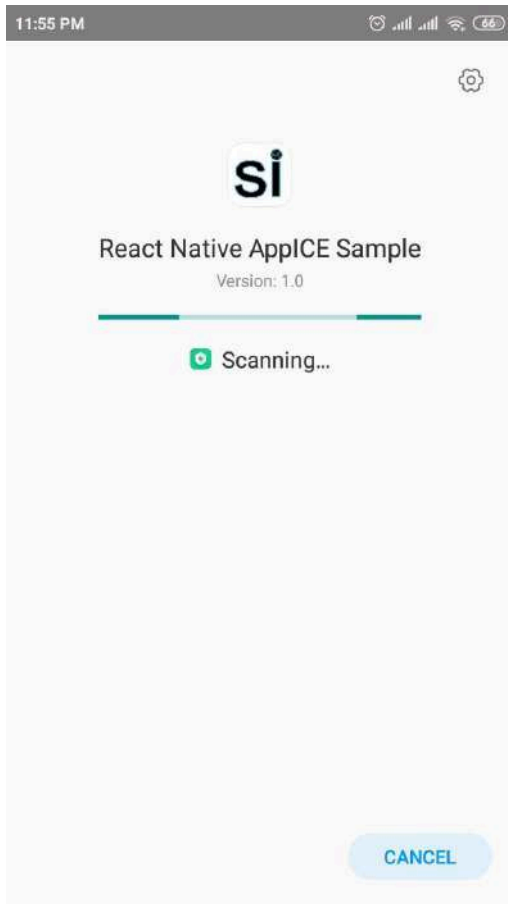
Now Enter your campaign name, select dates and campaign details according to your need.

Click on Submit button to create campaign



## Install App on your Android/iOS phone

1. Download & install app on your Android/iOS phone.



2. open the sample App and you will see 4 buttons

SEND EVENT

SET SAMPLE USER ID

SET USER PROFILE

SET CUSTOM VARIABLE

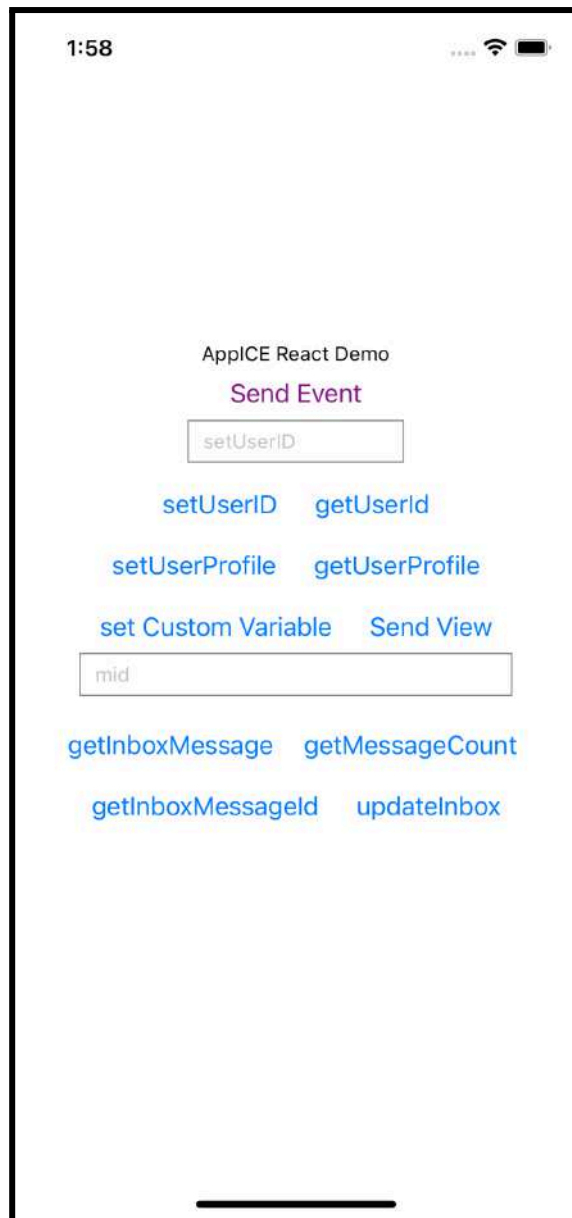
SEND VIEW

GET INBOXMESSAGE

GET MESSAGECOUNT

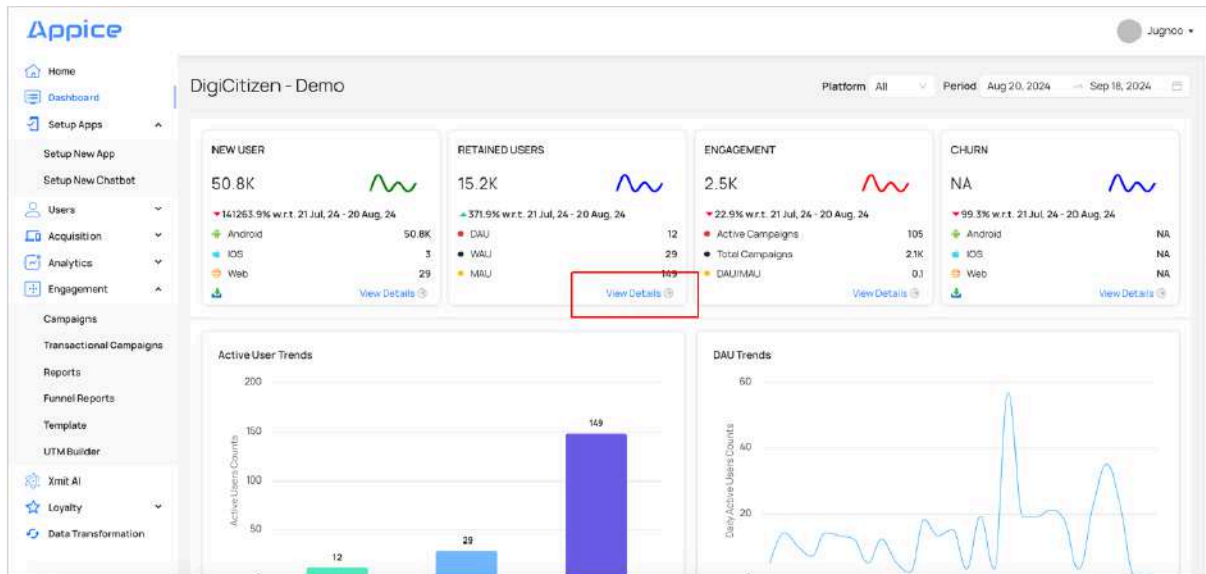
GET INBOXMESSAGEID

UPDATEINBOX



## Verify integration on Appice panel

1. Login to panel.Appice.io to see these values. Click on your app and go to its dashboard.



2. Click on 'Active Users View Detail' to see details of your phone app.

3. Clicking on 'View Detail' takes to App Users page. Search your phone Advertising Id in Search box to search for your phone Ads Id.

The screenshot shows the 'App User' page in Appice. The top navigation bar includes 'Platform' (All) and 'Period' (Aug 20, 2024 - Sep 18, 2024). The main content area displays three metrics: DAU (12), WAU (29), and MAU (149). Below these metrics is a search bar and a table of users. A red box highlights the search bar.

User	User Since	Last Seen	# of Sessions	Average Duration	Client ID
p6d99e7-64ba-4707-8495-309754de4205	Sep 21, 2024	Sep 18, 2024	1		
de65e531-454c-4409-8955-82d892d7055	Sep 19, 2024	Sep 19, 2024	2		
87d6b07-15ca-4848-8968-7e5c1f201015	Sep 19, 2024	Sep 19, 2024			
a2c7849e-4ced-431b-8fc1-8890e0e0e92	Sep 19, 2024	Sep 19, 2024	1		
5de9f85-e970-48a7-9324-5e3529e0d428	Sep 19, 2024	Sep 18, 2024	1		
778ba739-1a01-448d-bbb9-859d9e28eb95	Sep 19, 2024	Sep 19, 2024	12		

Showing 1 to 6 of 1519 entries

4. Click on the searched Ad Id to see details:

WhatsApp

appICE - React Native Integratio

AppICE | Dashboard

OTP for Secure Login to appICE

panel.appice.io/dashboard?actor=users#/users/details/c56b9cb0da0c9398b009a8b46c34066383b8482

Rahul K. Singh

Home

Dashboard

Setup Apps

Users

Acquisition

Activities

Engagement

Churn

Documentation

Reports

a0b10608-7378-486a-b303-18457a093a2c

Demographics

Gender : N/A

User Attributes

Variable	Value
Model	Xiaomi MI A3
Carrier	Vodafone IN
App Version	1.0

Competing Apps

Interests

Recent Activity

02/16/23 12:11:11

Event Test

ClickedMenuBtn : true  
User : Hello User How are you  
ClickedSubmitBtn : true  
TestRun : This is test run  
AppName : RN Sample App

02/16/23 12:09:20

Session\_Start

03/16/23 12:07:00

Session\_End

02/16/23 12:06:33

Session\_Start

02/16/23 11:55:25

Campaign\_Clicked

campid : 63edcc48803984fc72a20bc5  
mid : f22ad5e2-4f10-49fb-b7af-  
a1c3cd516bc

02/16/23 11:55:16

Campaign\_Received

campid : 63edcc48803984fc72a20bc5  
mid : f22ad5e2-4f10-49fb-b7af-  
a1c3cd516bc

02/16/23

Session\_End

Recency

Seen 8 secs ago

Monetary

0

Campaign Engagement

0

CAC / LTV

0