Mobile Application Builder-Android Guide
Oracle Banking Digital Experience
Patchset Release 22.2.2.0.0

Part No. F72987-01

December 2023

ORACLE

Mobile Application Builder-Android Guide

December 2023

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax:+91 22 6718 3001

www.oracle.com/financialservices/

Copyright © 2006, 2022, Oracle and/or its affiliates. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Table of Contents

1. Preface 1–1

1.1 Intended Audience 1–1

1.2 Documentation Accessibility 1–1

1.3 Access to Oracle Support 1–1

1.4 Structure 1–1

1.5 Related Information Sources 1–1

2. OBDX Servicing Application 2–1

2.1 Prerequisites 2–1

2.2 Create project using Remote UI 2–3

2.3 Local UI by running on local machine or local server. 2–3

2.4 Importing in Android Studio 2–6

2.5 Widget Functionality 2–7

2.6 Scan to Pay from Application Icon – 2–8

2.7 Scan Card using Augmented Reality 2–8

2.8 Passkey (Passwordless login) 2–8

2.9 Deeplinking - To open reset password, claim money links with the application 2–11

2.10 Device Registration and Push Registration Functionality 2–13

3. Google Play Integrity 3–1

4. FCM Push Notifications 4–6

5. Build Release Artifacts 5–1

6. OBDX Authenticator Application 6–10

6.1 Authenticator UI (Follow any one step below) 6–10

6.2 Authenticator Application Workspace Setup 6–11

7. Application Security Configuration 7–1

8. Live Experience With Jumio Integration 8–1

9. Adding Custom Cordova Plugin 9–1

10. ODA Chatbot Inclusion 10–4

11. Live Experience Integration 11–7

12. Push Notification 2FA configuration 12–9

1. Preface

1.1 Intended Audience

This document is intended for the following audience:

- Customers
- Partners

1.2 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

1.3 Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1.4 Structure

This manual is organized into the following categories:

Preface gives information on the intended audience. It also describes the overall structure of the User Manual.

The subsequent chapters describes following details:

- Introduction
- Preferences & Database
- Configuration / Installation.

1.5 Related Information Sources

For more information on Oracle Banking Digital Experience Patchset Release 22.2.1.0.0, refer to the following documents:

- Oracle Banking Digital Experience Installation Manuals

2. OBDX Servicing Application

2.1 Prerequisites

OBDX Android App is supported only on versions n (current) and n-1 release.

a. Download and Install node Js (will be downloaded to default path)

b. Install node js from https://nodejs.org

c. DOWNLOAD AND INSTALL ANDROID STUDIO

d. Download and install Android Studio from https://developer.android.com/studio/index.html

e. Download and Install Android platforms

f. Update Android SDK to latest API Level.

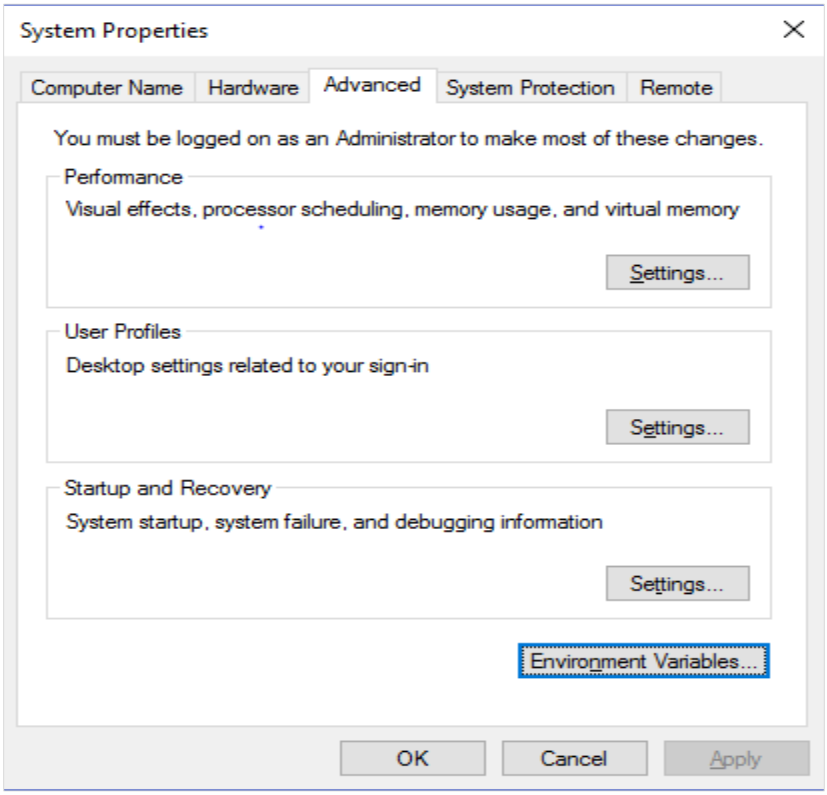
g. Gradle Version: gradle-4.6

h. Android Gradle Plugin Version (3.4.0): 'com.android.tools.build:gradle:3.4.0' or above

i. Set Environment variables

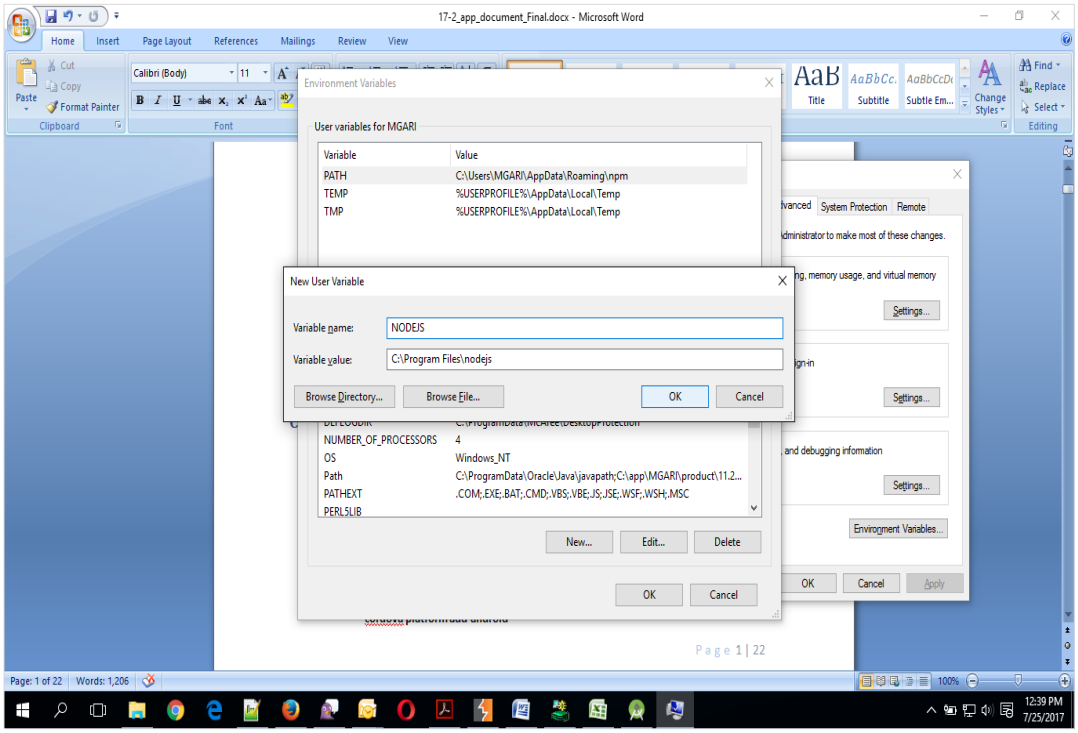
j. Set following system variables:

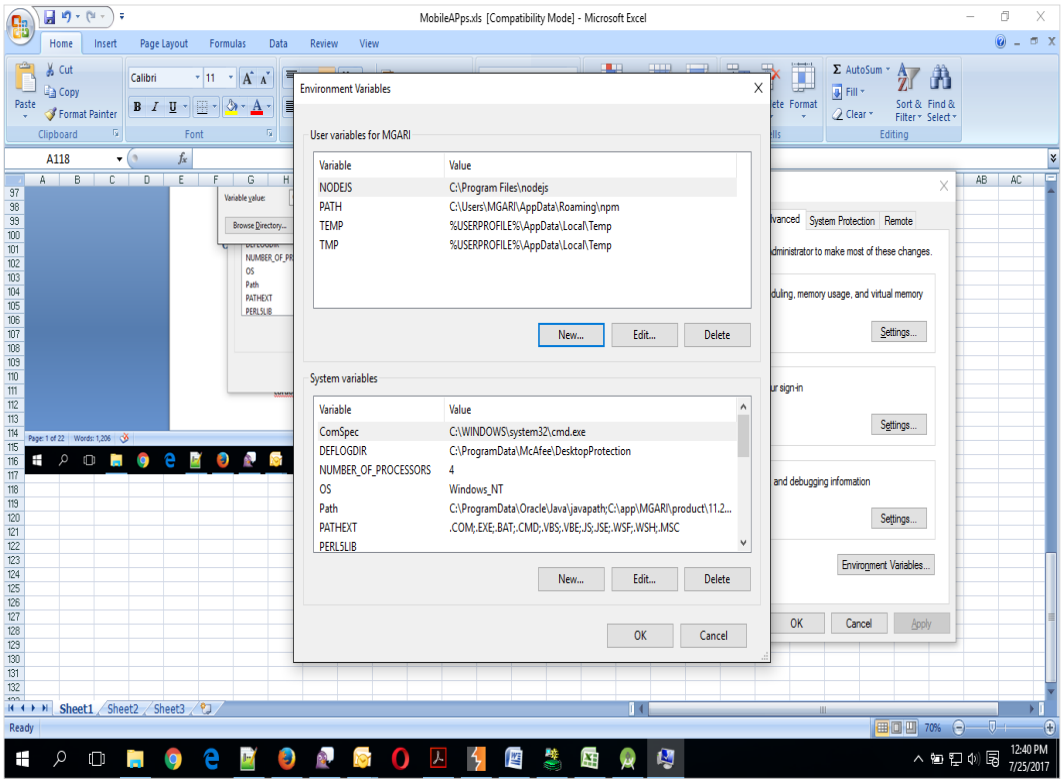
1. Click on Windows key and type Environment Variables.
2. A dialog box will appear. Click on the Environment Variables button as shown below



3. NODEJS <nodejs_path> Example: "*C:\Program Files\nodejs*".

k. Add the above variables in "*PATH*" system variable.





In 20.1, you can create app in two ways-using local UI or using remote UI (if want to create using remote go to section **Create project using Remote UI2.2** else directly to section **Local UI**)

2.2 Create project using Remote UI

a. Index.html changes(use Android Studio or any other editor)

- Update the server URL in app.properties against KEY_SERVER_URL key. This is the URL where the UI is also hosted.

  After this proceed to **2.4 Importing in Android Studio** directly.

2.3 Local UI by running on local machine or local server.

2.3.1 Adding UI to workspace

Use any 1 option below of a/b

a) Building un-built UI (required in case of customizations)

1. For this version, since the UI is built with webpack, the built UI cannot be modified from with the mobile workspace as it is minified code. Hence, either bank can hoist the UI is two ways:
   - Use local machine as local server and host the UI on local development machine and connect the application using localhost.
   - OR host the UI on local development server and point the application to that server URL

1. UI is same for internet and mobile, same build process of internet to be followed.

   Bank can follow the UI build steps from “Oracle Banking Digital Experience User Interface Guide”.

2. For building UI for mobile, Open scripts->webpack->webpack.dev.js and add below line in devServer object:

   as below:

```
headers: {
        "Access-Control-Allow-Origin": "*"
},
```

SAMPLE:

```
devServer: {
  static: path.join(__dirname,
                    "../../dist"),
  compress: true,
```

```
port: 4000,
hot: false,
client: false,
headers: {
        "Access-Control-Allow-Origin": "*"
},
```

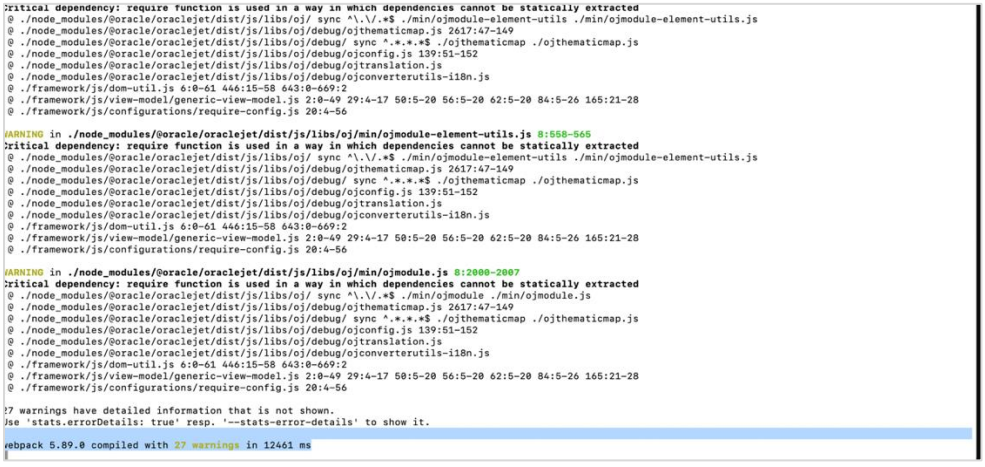
3. Also, in webpack.dev.js comment out below lines inside “entry” key.

```
entry: {
    //  main: "framework/js/configurations/require-config.js",
     // Runtime code for hot module replacement
     //hot: 'webpack/hot/dev-server.js',
     // Dev server client for web socket transport, hot and live reload logic
//client: 'webpack-dev-server/client/index.js?hot=true&live-//reload=true',
    },
```

4. Once the UI is built, run below command to start a local server on the development machine using below command:

   - npm run start

   - Once this server starts, below is the window which appears. This indicates local server is started.



   - Point the “key_server_url” to http://localhost:4000 and run the application on simulator. To run on device, the internet proxy should allow localhost domain to accept incoming requests.

If it is blocked, UI should be built and “npm start” command should be executed on a development server machine which is accessible in the network. They “key_server_url” will then point to that local server URL instead of localhost

b) Using built UI (out of box shipped with installer)

Available at --

OBDX_Installer/installables/ui/deploy (Main release, OBDX installer),
OBDX_Patch_Installer/installables/ui/deploy (Patchsets)

- There will be production enabled dist generated in the built UI.
- Bank can either directly deploy this dist to their server and point the application to that server as mentioned in point a above OR
- Bank can copy the dist folder in their workspace and follow steps from point 3in section 2.5.
- If bank wants to do any changes, point a) steps needs to be followed.

NOTE: If banks want to debug UI in production builds, then dist should be created with below configuration enabled in webpack.prod.js

devtool: 'eval',

- This will however increase the files deployed on server and reduce the proformance on production. Refer Webpack documentation https://webpack.js.org/configuration/devtool/ for more details.

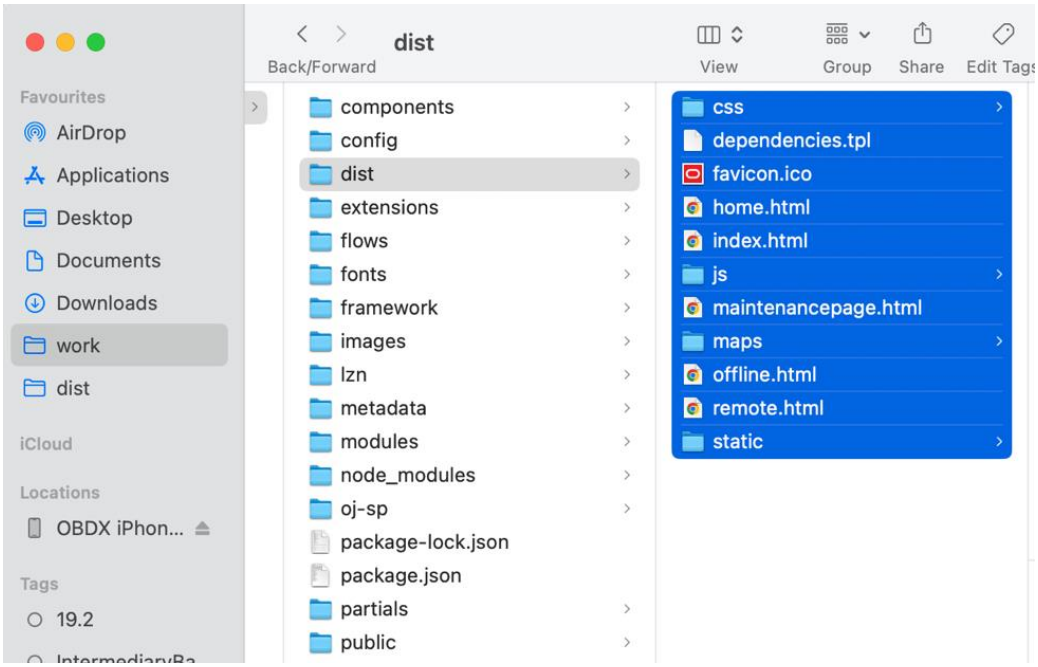
2.3.2 Create Project Using local UI within the workspace

1. Extract the unbuilt UI and follow steps up to 5 in the above section 2.4.
2. After step 4, run below command to generate dist folder.

   npm run webpack-dev – this will generate development enabled dist

   npm run webpack-build- this will generate production enabled dist
3. Once the dist folder is created, copy all files inside dist folder and save it in the

   workspace_installer/zigbank/platforms/android/app/src/main/assets/www/.



4. Open Index.html and home.html and add below line inside head section below meta tag

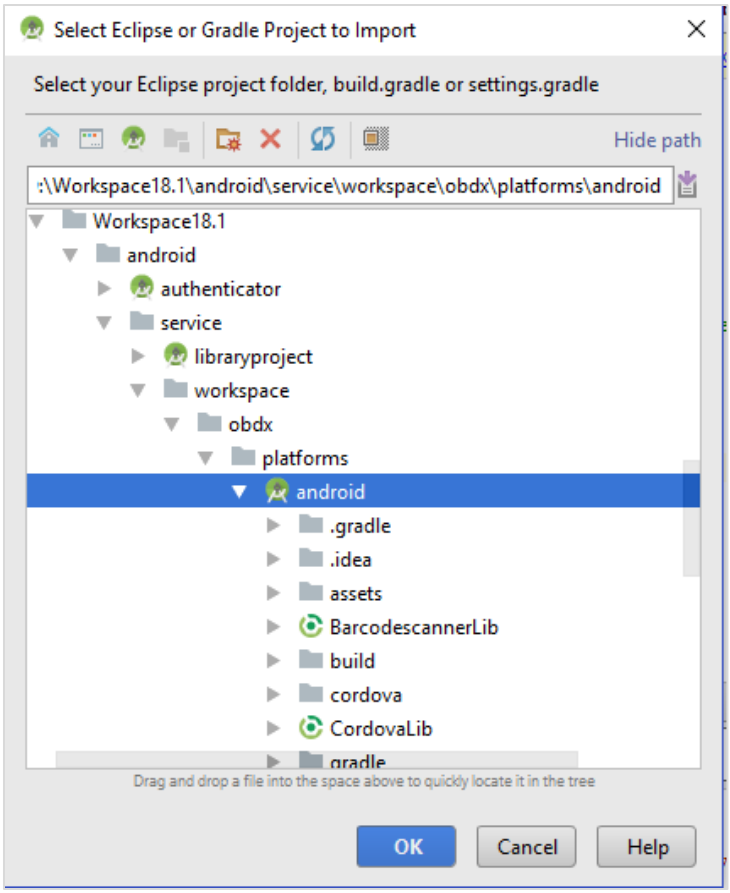
   <script src="cordova.js" type=”text/javascript”></script>

5. Set the server URL in app.properties against key_server_url. This is the URL where backend services are hosted.
6. With this setup, since the files generated in dist folder are minified format we cannot change the code. If any change needs to be done in any UI file, then the changes must be done in the UI folder, built it again to generate dist and copy the files to workspace again. Since this is tedious process, we recommend to setup local server and host UI there for development.

2.4 Importing in Android Studio

Open Android Studio

1. Import zigbank**/**platforms**/**android in android studio by clicking on Open an Existing Project.



2.5 Widget Functionality

Widgets are Android native feature. Below widgets are available in the application

1. All Accounts Widgets – Widget, showing all accounts balances & account numbers.
2. Account Details Widget - Widget, showing account balance of default account and last 5 transactions of the same account, can be added to the phone home screen. If default account is not set, then the details of the account fetched first is shown.
3. Multi-Functional Widget – Widget showing default account balance. If default account is not present, it shows details of account fetched first. Additionally, it has option to scan to pay feature
4. Scan to Pay Widget – Widget which allows to scan to pay.

Prerequisite:

Quick Snapshot feature needs to be enabled in the app application from the login screen. (Refer function doc - User Manual Oracle Banking Digital Experience Quick Snapshot.docx)

Please enable below property in app.properties file

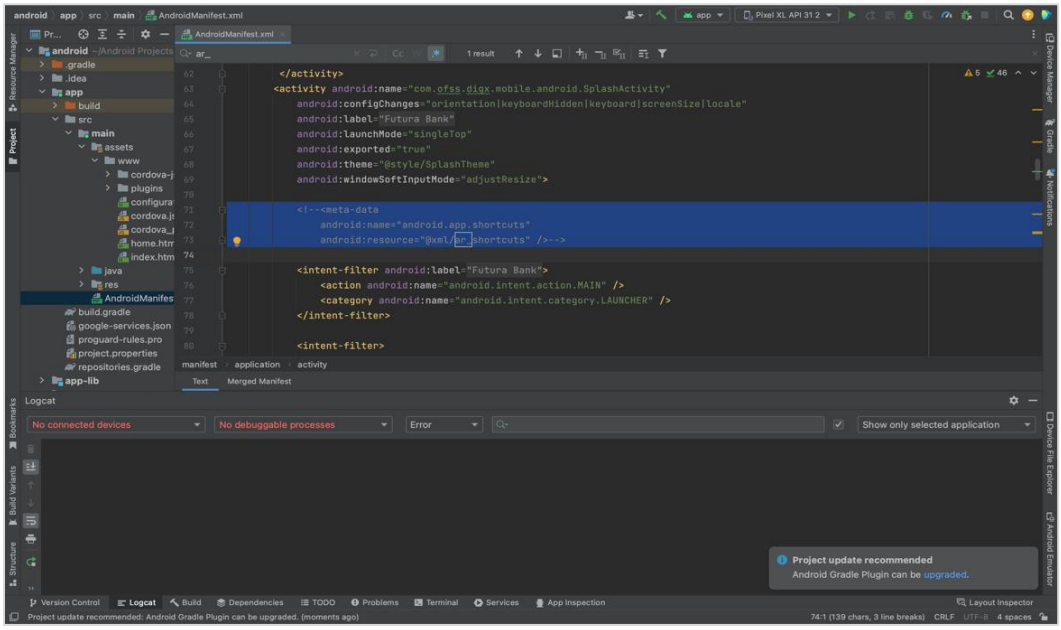
<bool name="ENABLE_WIDGET">true</bool>

If bank does not want this feature, then they can disable this by making above flag to false.

2.6 Scan to Pay from Application Icon –

Users can long press on bank's application icon on home screen and click on scan-to-pay option to scan QR and make payments.

To enable this feature uncomment below from app's AndroidManifest.xml



2.7 Scan Card using Augmented Reality

Users can scan card and view account details and transactions of the account associated with the card.

To enable this feature, do the same step which is mentioned on 2.6 section.

2.8 Passkey (Passwordless login)

Passkeys are a safer and easier replacement for passwords. With passkeys, users can sign in to apps and websites using a biometric sensor (such as a fingerprint or facial recognition), PIN, or pattern. This provides a seamless sign-in experience, freeing your users from having to remember usernames or passwords.

Passkeys are supported only on devices that run Android 9 (API level 28) or higher

TO DISBALE THIS OPTION:

By doing this, passkey option will not be available to users withing the application. User will not be able to register for passkey and also will not be able to login using passkey. Follow below steps

a. Remove RTM access from Client Servicing -> Authentication - > Passkey Setup for Mobile Application/Mobile (Responsive) and Internet touch points

b. Set this flag in channel-framework-js-configurations-config..js to false

thirdPartyAPIs -> passkey -> required -> false

TO ENABLE THIS OPTION:

1. Add RTM access from Client Servicing -> Authentication - > Passkey Setup for Mobile Application,Mobile (Responsive) and Internet touch points

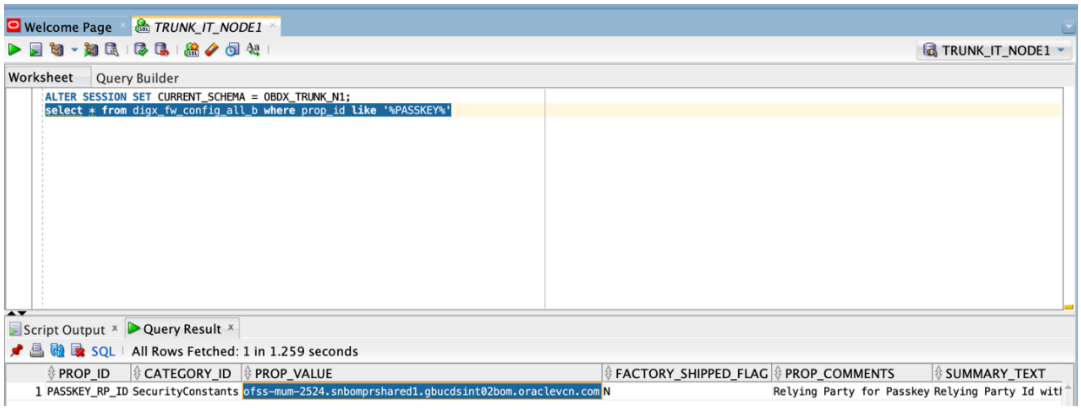
2. Set this flag in channel-framework-js-configurations-config.js to true

   thirdPartyAPIs -> passkey -> required -> true

3. Along with above, we need below server side and application side settup

Server-Side Setup:

1. Update the relying party in below property select prop_value from digx_fw_config_all_b where prop_id='PASSKEY_RP_ID'



2. Note – Relying partId is the domain name if the website to which credentials will be associated. (Eg google.com, example.com etc)

   Relying party origin is the relying party of website prefixed with protocol without the port.

(E,g, https://google.com, https://example.com)

   a. Create assetlinks file (assetlinks.json) -

      A Digital Asset Links JSON file must be published on your website to indicate the Android apps that are associated with the website and verify the app's URL intents.

The following example assetlinks.json file grants link-opening rights to a com.example Android app:


```
[{
    "relation": ["delegate_permission/common.handle_all_urls"],
    "target": {
      "namespace": "android_app",
        "package_name": "com.example",
"sha256_cert_fingerprints":["14:6D:E9:83:C5:73:06:50:D8:EE:B9:95:2F:34:FC:64:16:A0:83:42:E6:1D:BE:A8:8A:04:96:B2:3F:CF:44:E5"]
    }
}]
```


The JSON file uses the following fields to identify associated apps:

package_name: The application ID declared in the app's build.gradle file.

sha256_cert_fingerprints: The SHA256 fingerprints of your app’s signing certificate. You can use the following command to generate the fingerprint via the Java keytool:

```
keytool -list -v -keystore my-release-key.keystore
```

b. Publish assestlinks.json file-

This file needs to be on https server with valid SSL certificate

You must publish your JSON verification file at the following location:

https://domain.name/.well-known/assetlinks.json

For example, if your sign-in domain is signin.example.com, host the JSON file at https://signin.example.com/.well-known/assetlinks.json.

Verify your assetlink json on below statement list tester-

https://developers.google.com/digital-asset-links/tools/generator

The MIME type for the Digital Assets Link file needs to be JSON. Make sure the server sends a Content-Type: application/json header in the response.

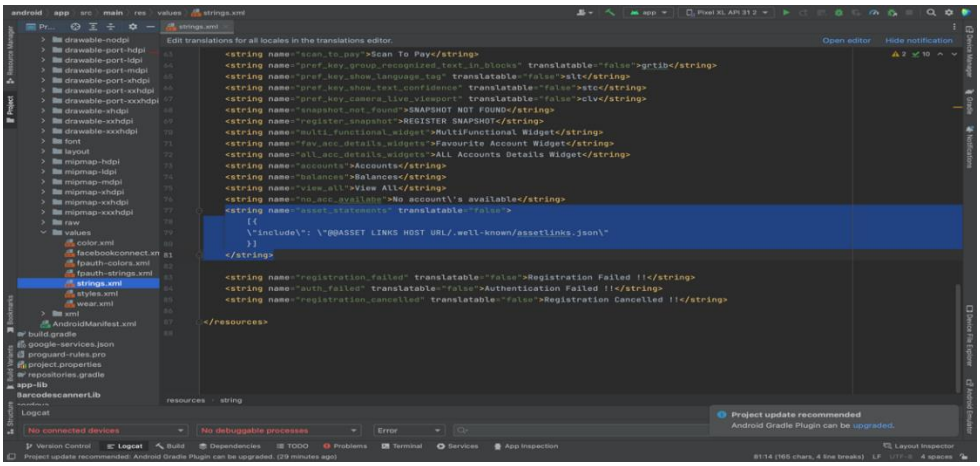
Need to change host and port in Obdx.conf as,

ProxyPass "/.well-known" "http://100.76.157.55:7003/digx-admin/sms/v1/.well-known"

ProxyPassReverse "/.well-known" "http://100.76.157.55:7003/digx-admin/sms/v1/.well-known"

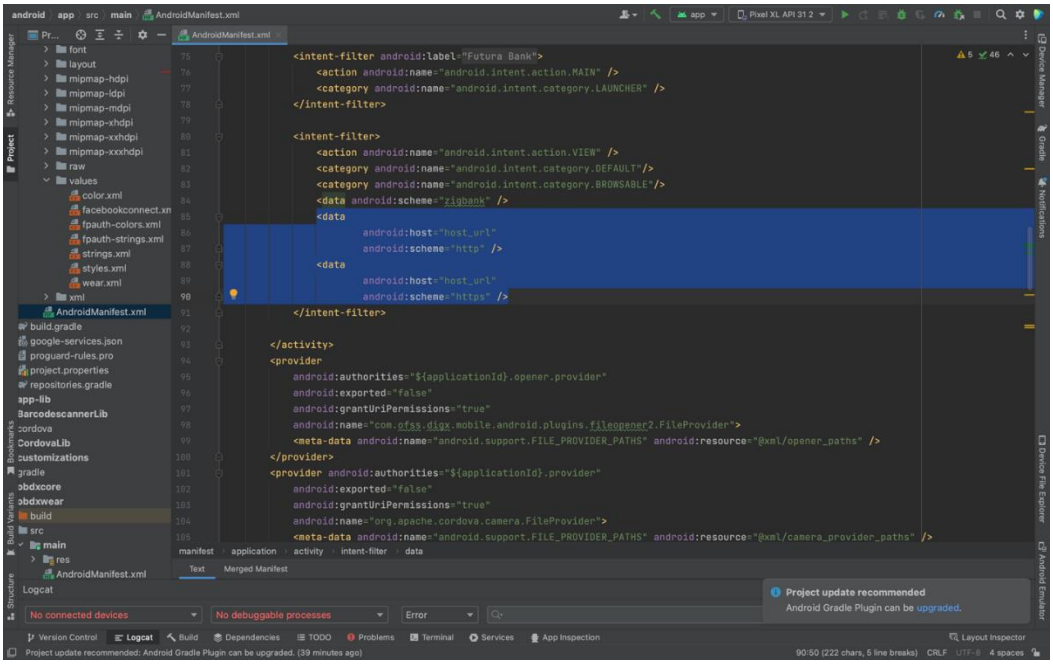
After the setup is done, this file must be accessible on mobile browser with this url. There should not by any redirects for accessing this file.

c. Add assetlinks.json file host in app's strings.xml file.



2.9 Deeplinking - To open reset password, claim money links with the application

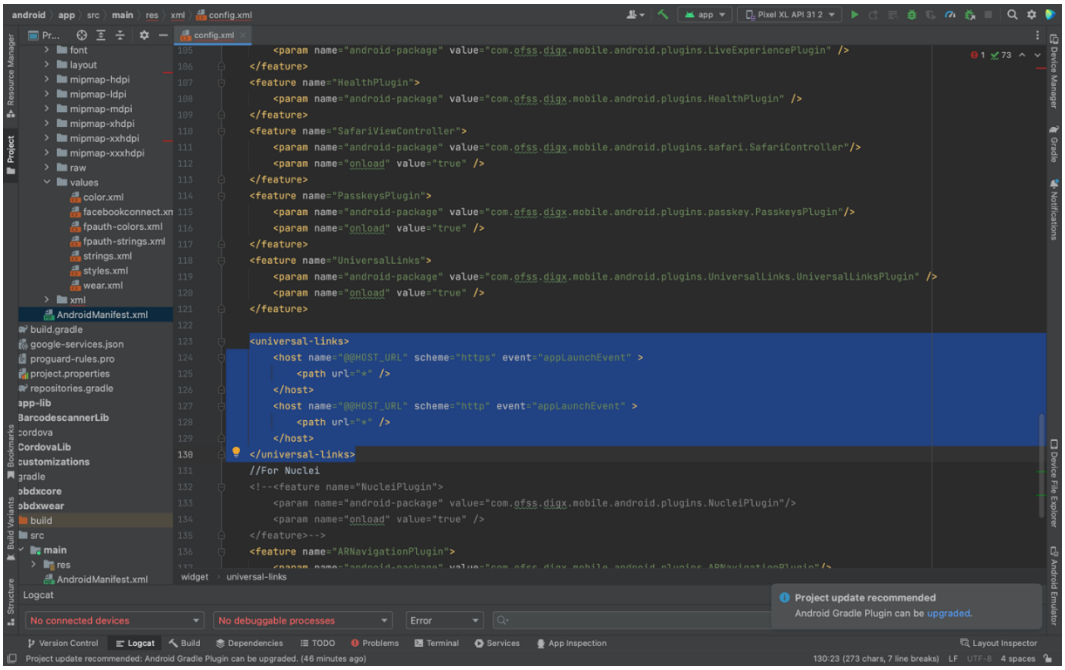
Please add host url under data tag in app's AndroidManifest.xml as,



Note – Please add host url without https or http.

For e.g. If your deeplink url is https://exmple.com/test then you can add only example.com in the data tag

Similary you can add the same host url in app's config.xml under universal-links tag as,



2.10 Device Registration and Push Registration Functionality

In this version, only one device is allowed to be registered for alternate login for the same username. If user tries to register another device with same username for alternate login, then the previous registration on other devices will be removed. User will get an error message if he/she tries to use PIN/PATTERN/BIOMETRIC on the de-registered devices.

While user registers his second device or same device again (by re-installing the application), a popup will appear to notify the same.

If user confirms, then the current device will be registered, and all previous registrations will be removed.

If user cancel, the process is exited.

Also, in this version, only one device is allowed to be registered for push.

Bank can allow multiple devices to be registered for same username in their setup by setting below two configurations:

ALLOWED_DEVICE_COUNT to any value between than 1 and 100.

- 1 will allow on one device registration.
- 100 will allow more than one device registration

ALLOWED_PUSH_DEVICE_COUNT any value between 1 and -1

- 1 will only one one device to be registered for push.
- -1 will only multiple devices to be registered for push

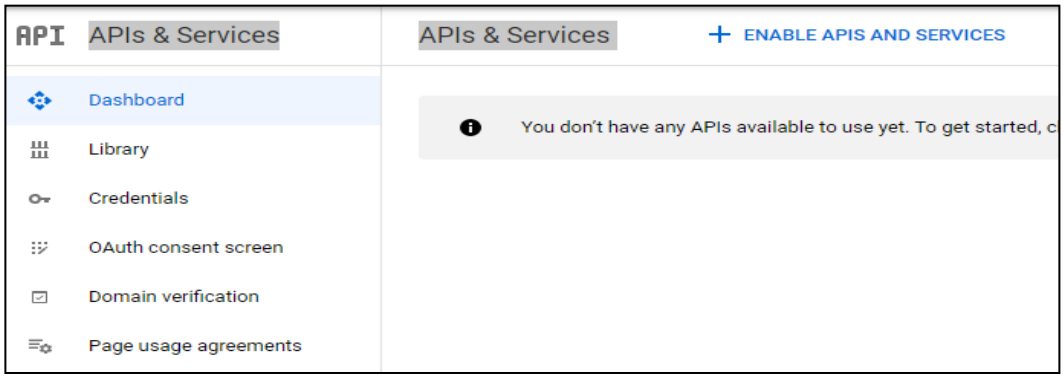
3. Google Play Integrity

a. Go to URL https://console.developers.google.com/

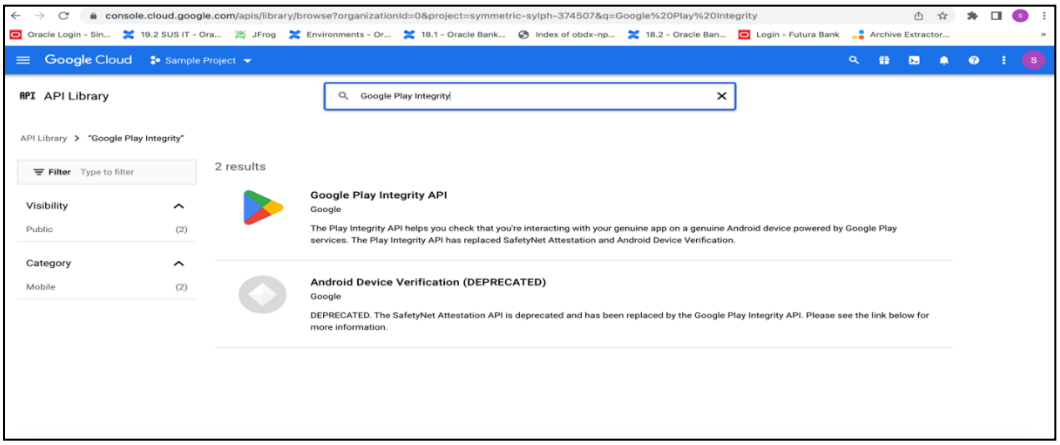
b. Create a new Project and set name of you project

c. Choose **'API's & Services'** option from side bar.

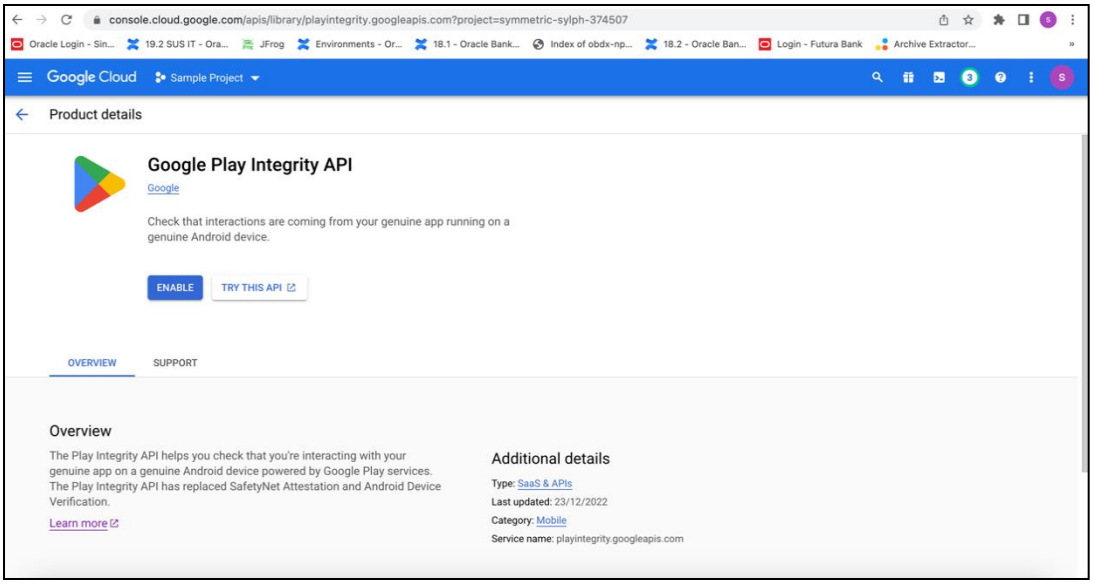
d. In API's & Services > Dashboard > Choose **'Enable APIS AND SERVICES'**.



e. This will redirect to **'Library'** where we need to search **'Google Play Integrity API'.**

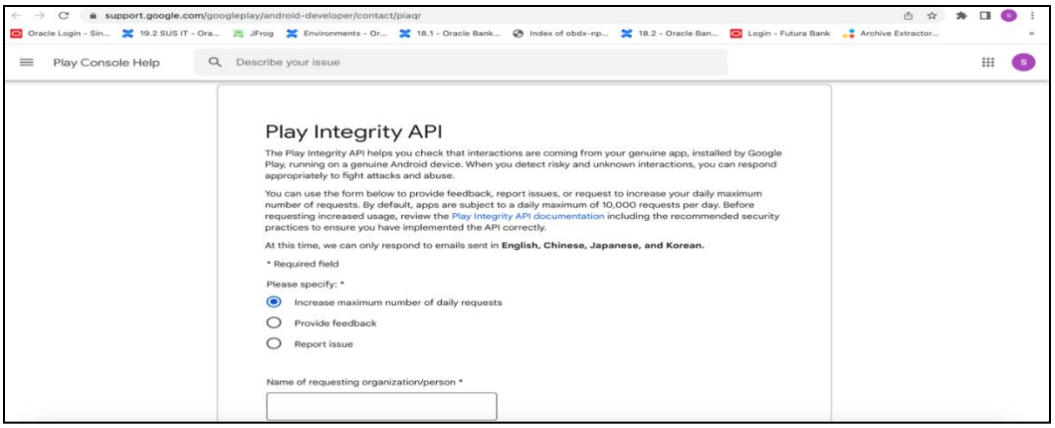


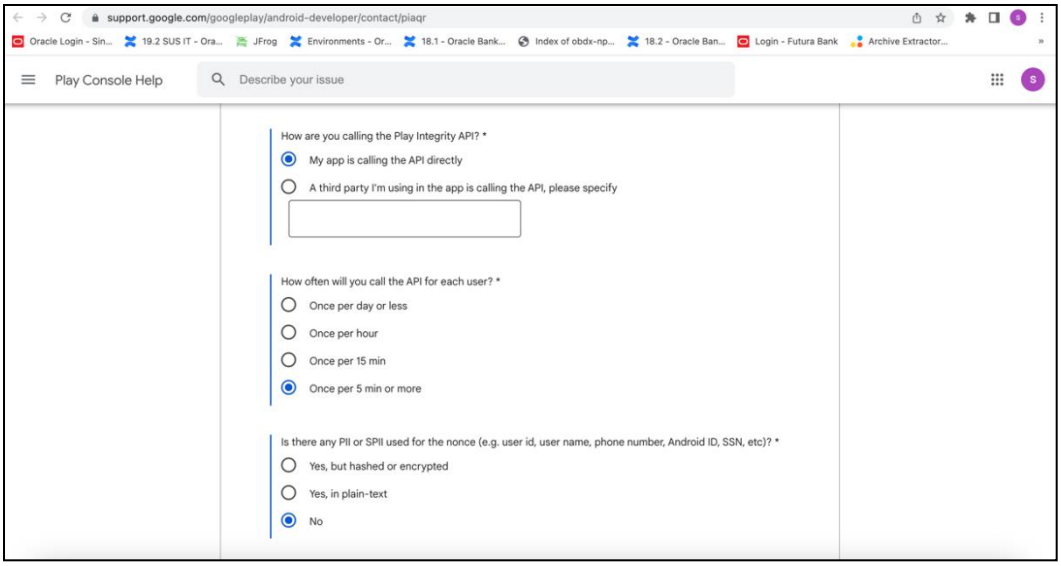
f. Click on Google Play Integrity API and enable it

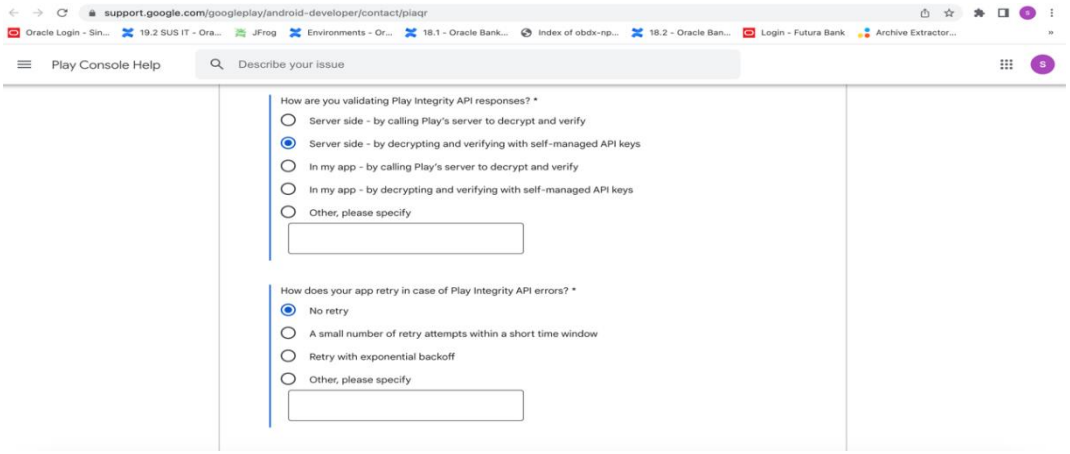


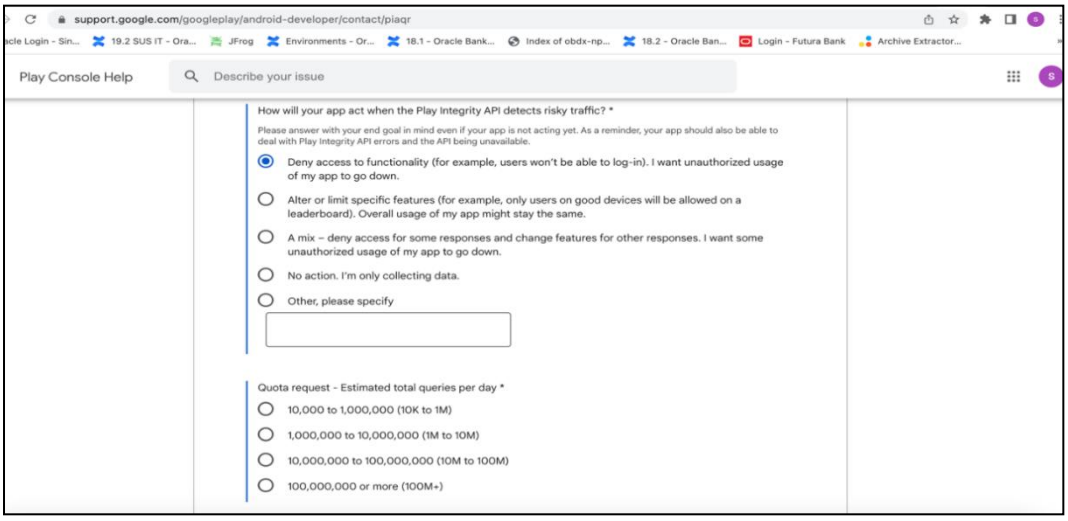
g. If the application usage is high, the quota request form needs to be submitted. Please fill quota request form from below site. Also select below options.

https://support.google.com/googleplay/android-developer/contact/piaqr







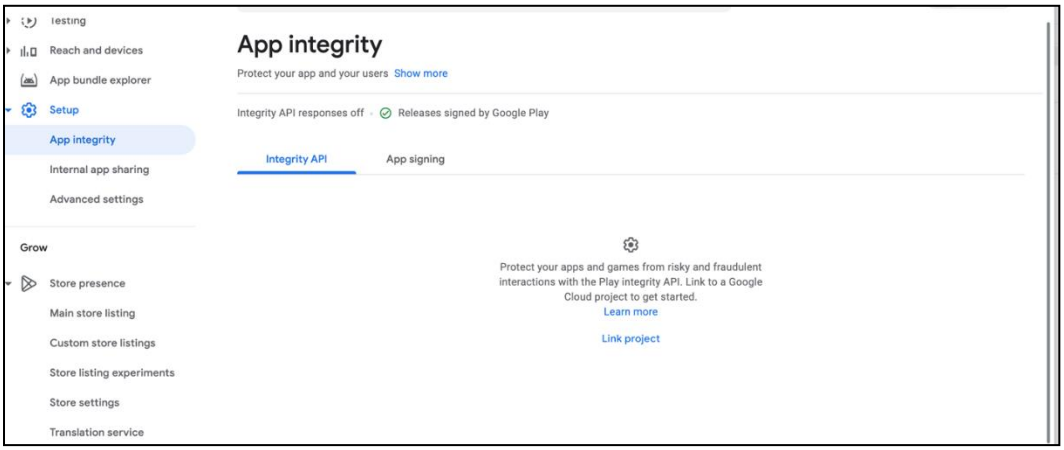


Quota request - Estimated total queries per day * → The approximate load, Play Integrity API is called once each time the app in opened

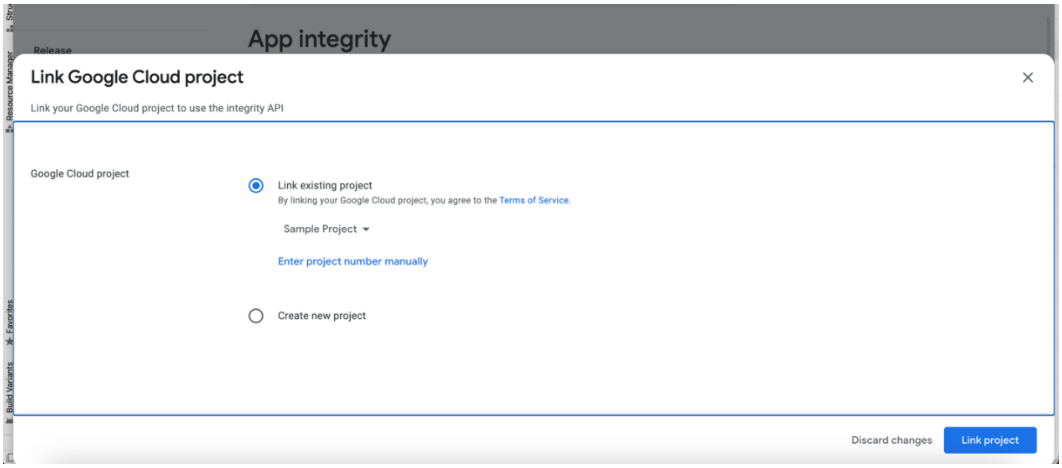
Quota request - Estimated peak queries per second → Leave blank

h. To enable Play Integrity responses please follow below steps-

Go to Google Play Console->Side Menu->Setup->App Integrity



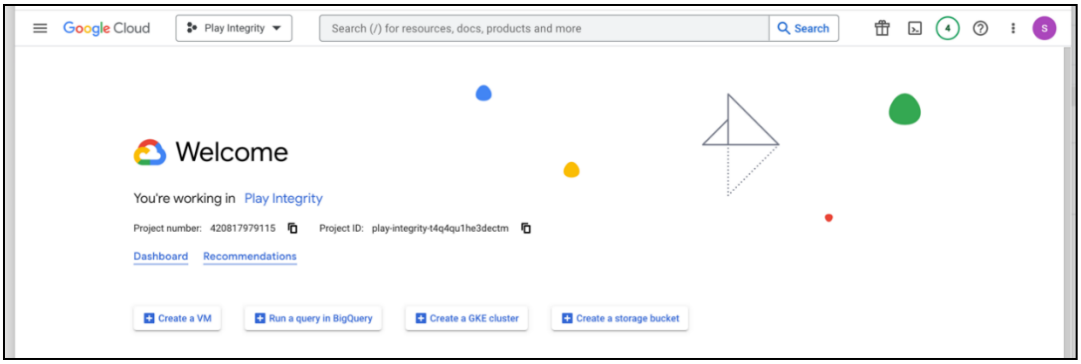
Click on **Link project** and then link your existing google cloud project. If it is not created then create new and link the same.



i. Add project number in below property of app.properties

```
<string name="GOOGLE_CLOUD_PROJECT_NO">@@GOOGLE_CLOUD_PROJECT NO</string>
```

You will get the project number on google cloud console project



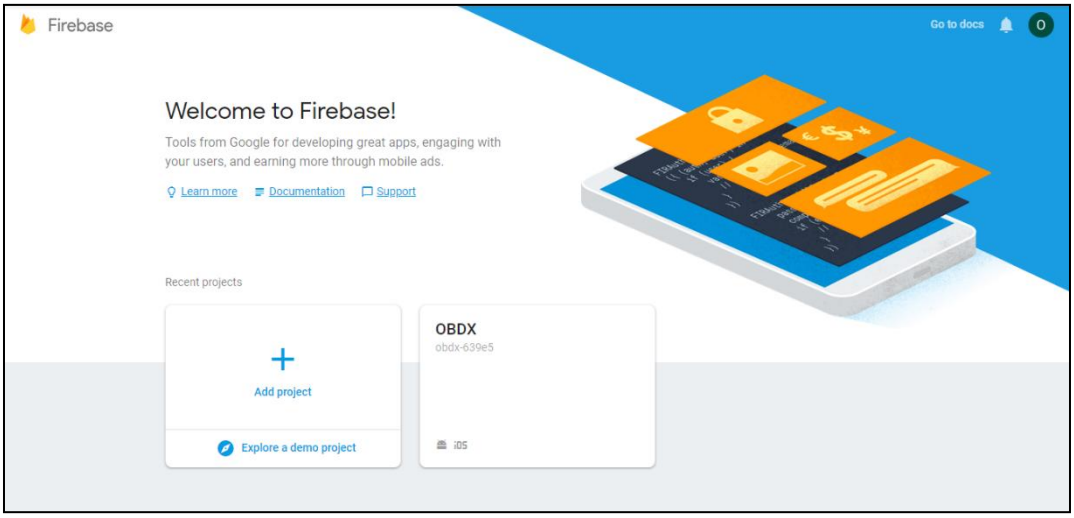
j. Mention the time in seconds to which app can hit the play integrity api. By default it is 300seconds but you can configure as per the requirement. Please use below property in RootCheckFlags.java(workspace_installer/zigbank/platforms/android/app/src/main/java/com/ofss/digx/mobile/android/)

long playIntegrityAPICallTime = your_time_in_seconds;

4. FCM Push Notifications

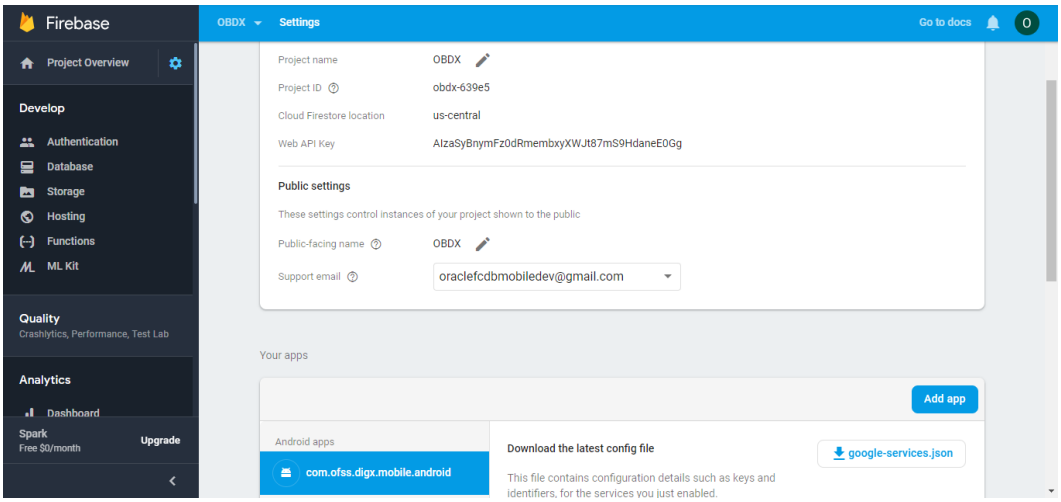
a. Go to URL https://firebase.google.com/

b. Traverse to console and create a project

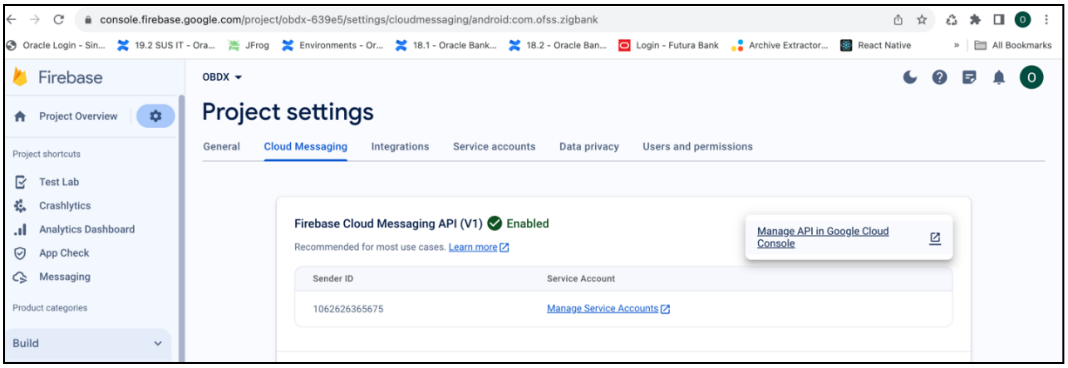


c. Download google-services.json from below page and save to (zigbank\platforms\android\app) directory.

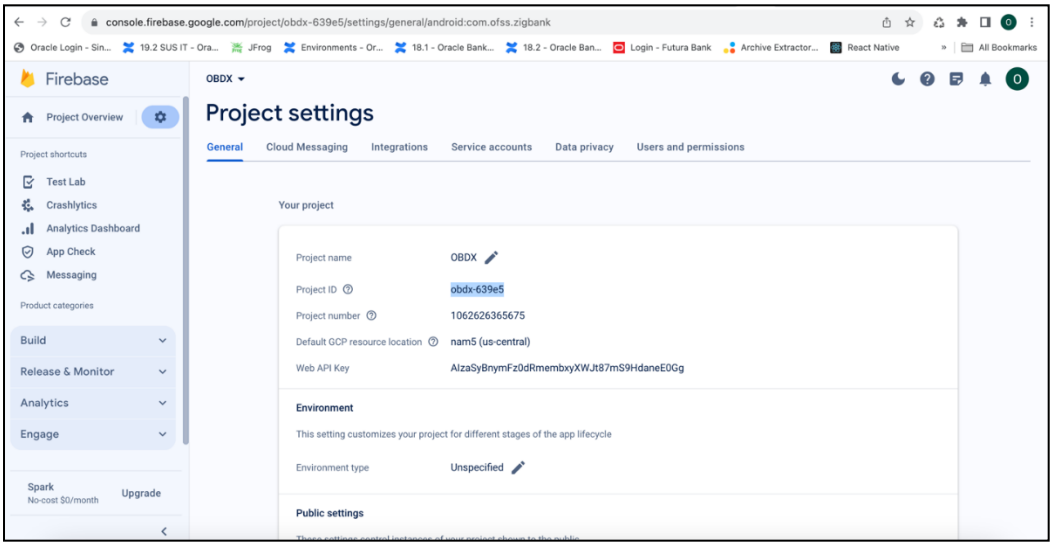
d. Remember to keep the projects package name and firebase package name same.



e. Traverse to cloud messaging tab Enable Firebase Cloud Messaging API(V1) by clicking on Manage API in Google Cloud Console.



f. Get the Project ID from Project Setting in Firebase console



g. Update FCM URL in below table as-

update DIGX_FW_CONFIG_ALL_B set prop_value = 'https://fcm.googleapis.com/v1/projects/YOUR_PROJECT_ID/messages:send' where prop_id = 'FCM_URL';

Add YOUR_PROJECT_ID in url which is captured on above step

h. If proxy address is to be used, provide the same in database as mentioned in point 3.

i. Generate private key for your service account by using below steps-

- In the Firebase console, open **Settings >** Service Accounts

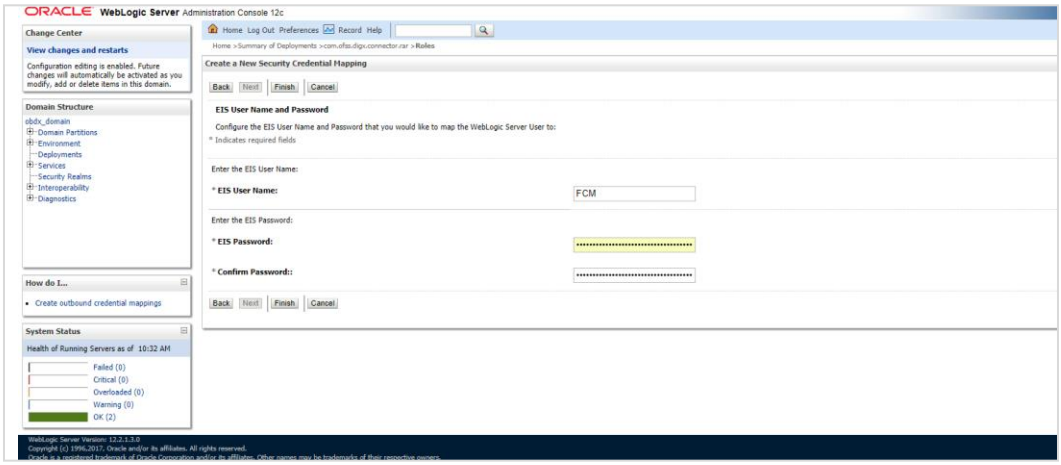
- Click **Generate New Private Key**, then confirm by clicking **Generate Key**

You can also follow below google doc -

https://firebase.google.com/docs/cloud-messaging/auth-server#provide-credentials-manually

Sr. No.	Table	PROP_ID	CATEGORY_ID	PROP_VALUE	Purpose
1	DIGX_FW_CONFIG_VAR_B	FCM	DispatchDetails	<Server_Key>	Service account json file content captured in above step
2	DIGX_FW_CONFIG_ALL_B	FCMKeyStore	DispatchDetails	DATABASE or CONNECTOR	Specifies whether to pick server key from database or from connector. Default DB (No change)
3	DIGX_FW_CONFIG_ALL_B	Proxy	DispatchDetails	<protocol,proxy_address>	Provides proxy address, if any, to be provided while connecting to APNS server. Delete row if proxy not required. Example: HTTP,148.50.60.8

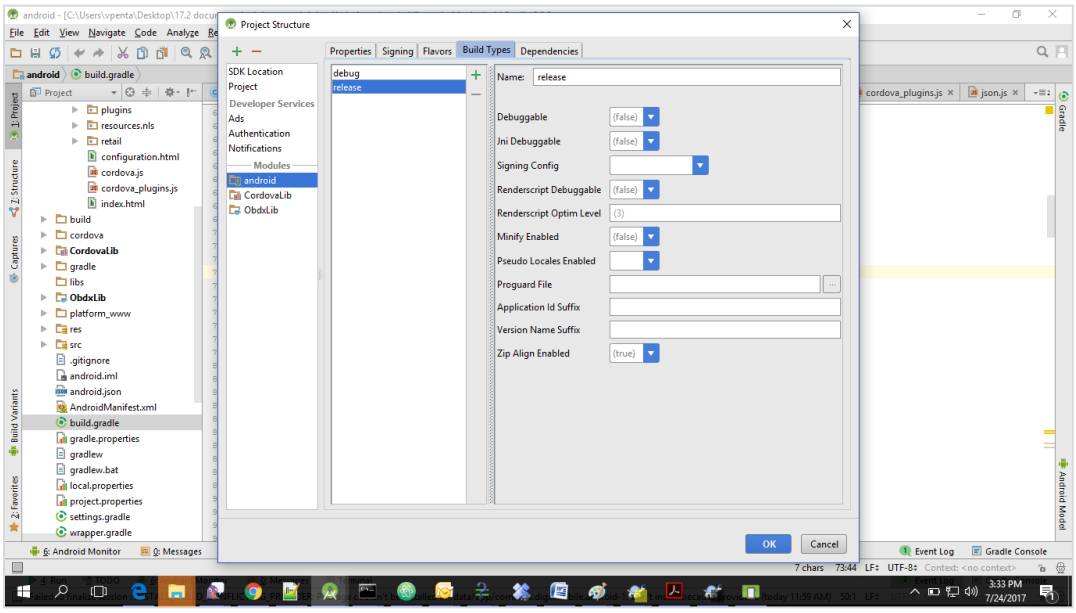
If CONNECTOR is selected in Step 2 update password as below



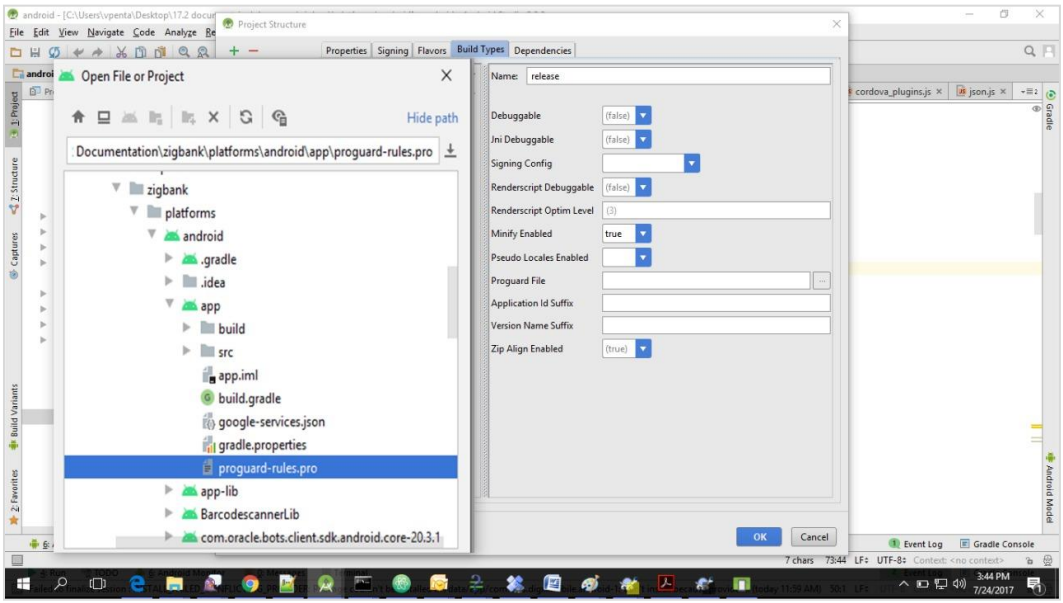
Home

5. Build Release Artifacts

1. Clean and Rebuild your project in Android Studio.
2. In Android Studio, on the menu bar Click on **Build -> Edit Build Types ->** select **release**



3. Set Minify Enabled -> True & click on Proguard File selection -> Navigate to proguard-rules.pro (zigbank\platforms\android\app)



4. Click on OK -> again click on OK.
5. Adding URLs to app.properties.xml (customizations/src/main/res/values/)
   a. NONOAM (DB Authenticator setup)

SERVER_TYPE	NONOAM
KEY_SERVER_URL	Eg. https://mumaa012.in.oracle.com:18443
WEB_URL	Eg. https://mumaa012.in.oracle.com:18443
SERVER_CERTIFICATE_KEY	Refer point 6.7

   b. OBDXTOKEN (Token based mechanism)

SERVER_TYPE	OBDXTOKEN
KEY_SERVER_URL	Eg. https://mumaa012.in.oracle.com:18443
WEB_URL	Eg. https://mumaa012.in.oracle.com:18443
SERVER_CERTIFICATE_KEY	Refer point 6.7

   c. OAM Setup (Refer to installer pre requisite documents for OAuth configurations)

SERVER_TYPE	OAM
KEY_SERVER_URL	Eg. https://mumaa012.in.oracle.com:18443 (This URL must be of OHS without webgate)
WEB_URL	Eg. https://mumaa012.in.oracle.com:18443
KEY_OAUTH_PROVIDER_URL	http://mum00aon.in.oracle.com:14100/oauth2/rest/token
APP_CLIENT_ID	<Base64 of clientid:secret> of Mobile App client
APP_DOMAIN	OBDXMobileAppDomain
WATCH_CLIENT_ID	<Base64 of clientid:secret> of wearables
WATCH_DOMAIN	OBDXWearDomain
SNAPSHOT_CLIENT_ID	<Base64 of clientid:secret> of snapshot
SNAPSHOT_DOMAIN	OBDXSnapshotDomain
LOGIN_SCOPE	OBDXMobileAppResServer.OBDXLoginScope
SERVER_CERTIFICATE_KEY	Refer point 6.7

d. IDCS Setup

SERVER_TYPE	IDCS
KEY_SERVER_URL	Eg. https://mumaa012.in.oracle.com:18443 (This URL must be of OHS without webgate)
WEB_URL	Eg. https://mumaa012.in.oracle.com:18443
KEY_OAUTH_PROVIDER_URL	http://obdx-tenant01.identity.c9dev0.oc9qadev.com/oauth2/v1/token
APP_CLIENT_ID	<Base64 of clientid:secret> of Mobile App client
WATCH_CLIENT_ID	<Base64 of clientid:secret> of wearables
SNAPSHOT_CLIENT_ID	<Base64 of clientid:secret> of snapshot
LOGIN_SCOPE	obdxLoginScope
OFFLINE_SCOPE	urn:opc:idm:__myscopes__ offline_access
SERVER_CERTIFICATE_KEY	Refer point 6.7

6. Domain Based Setup (This is same for OBDX servicing App and Authenticator App)

   To use domain based setup please enable below flag in app.properties file -

   <string name="DOMAIN_BASED_CATEGORIZATION">true</string>

   If you are using local UI then enable below flag in config.js(platforms/android/app/src/main/assets/www/framework/js/configurations/config.js) file -

   domainDeployment: {

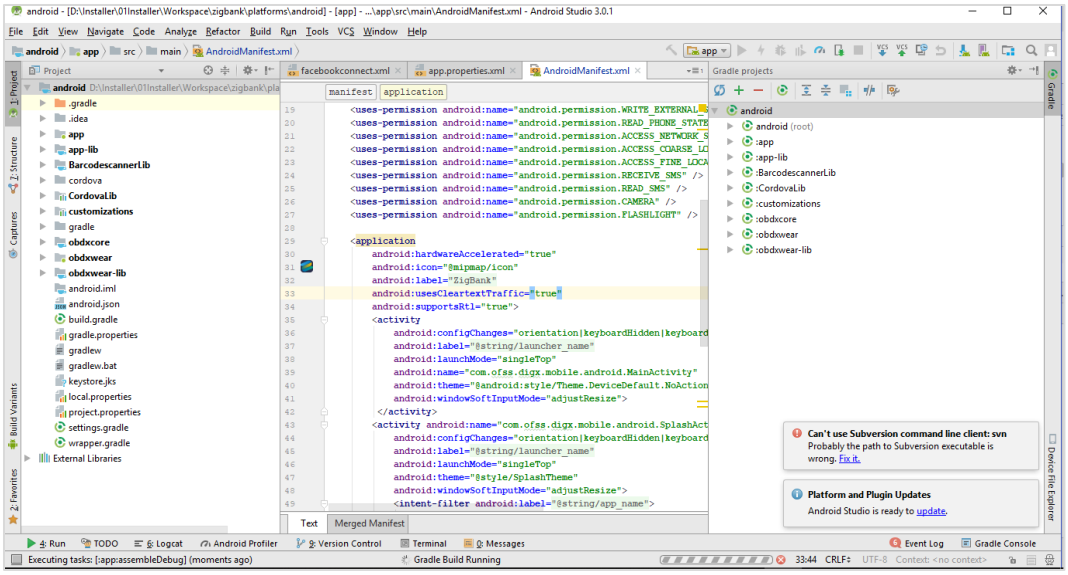
   enabled: true

   }

7. Adding chatbot support to mobile application (Optional)

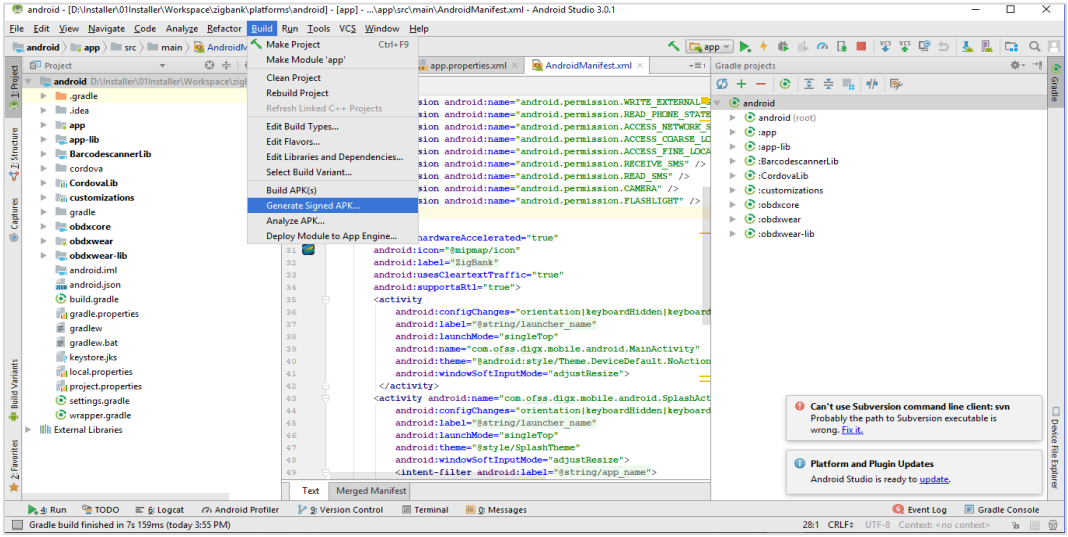
CHATBOT_ID	The tenant ID
CHATBOT_URL	The URL for the ChatApp application in ODA

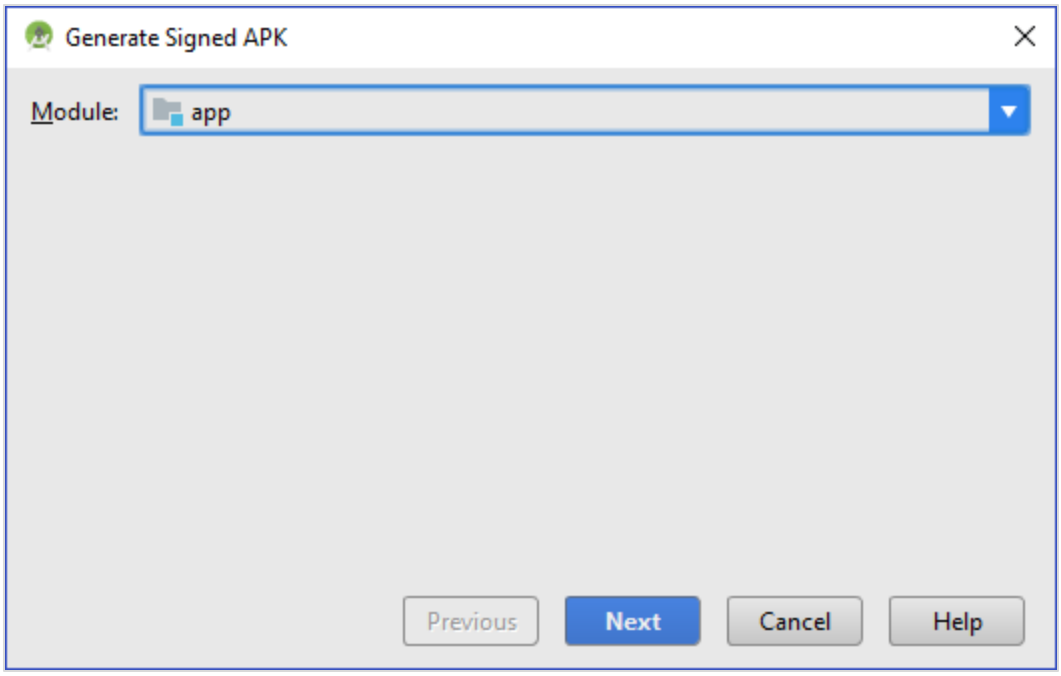
8. If using http protocol for development add (android:usesCleartextTraffic="true") to application tag of AndroidManifest.xml (on app & obdxwear target)

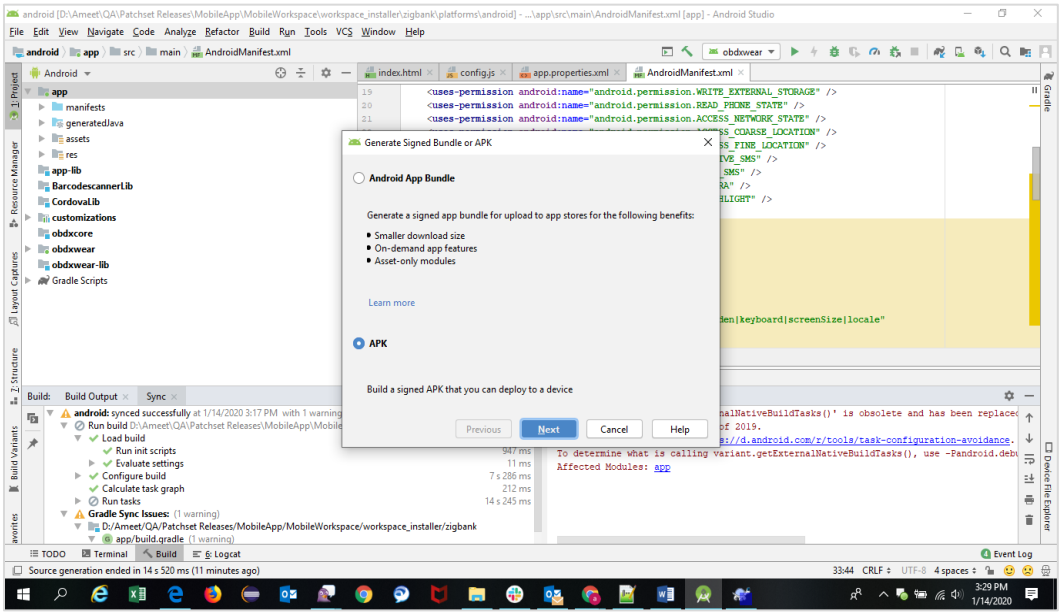


9. **For Generating Signed Apk:** To Generate release-signed apk as follows:

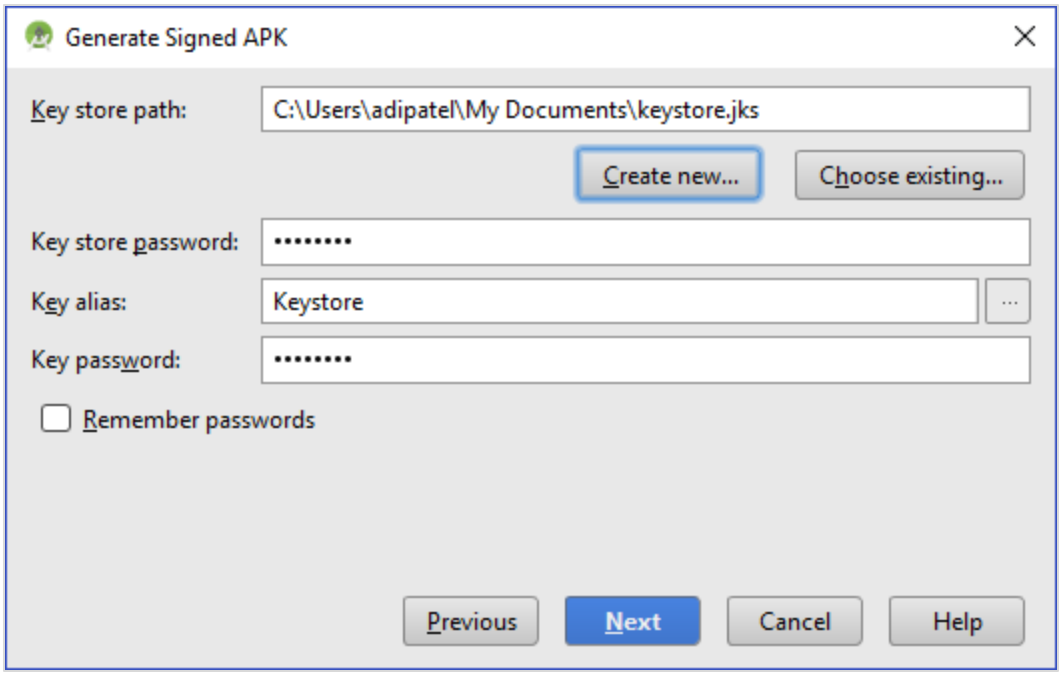
On menu bar click on Build -> Generate Signed Apk

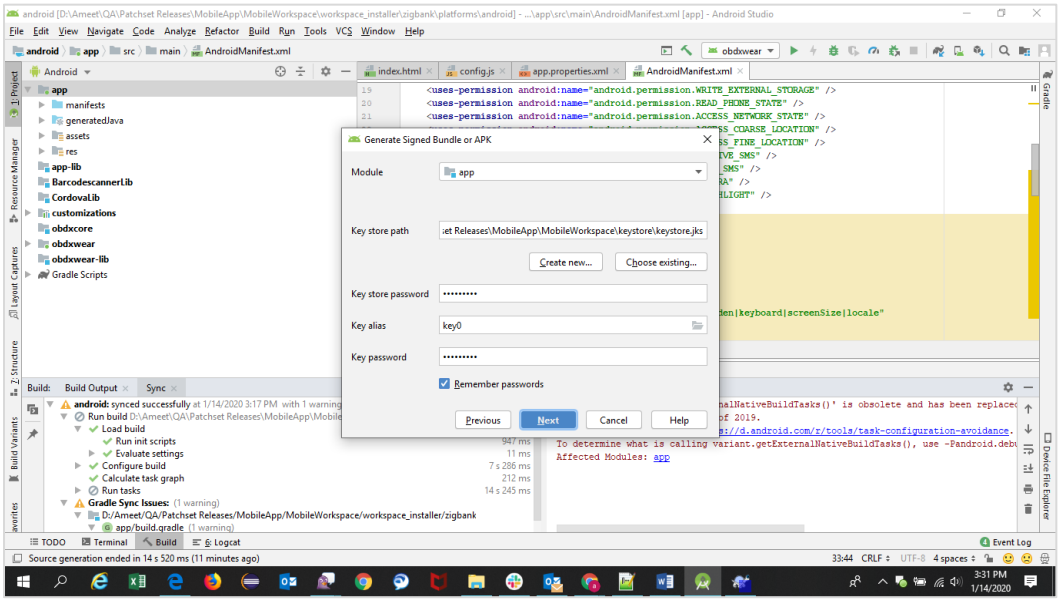




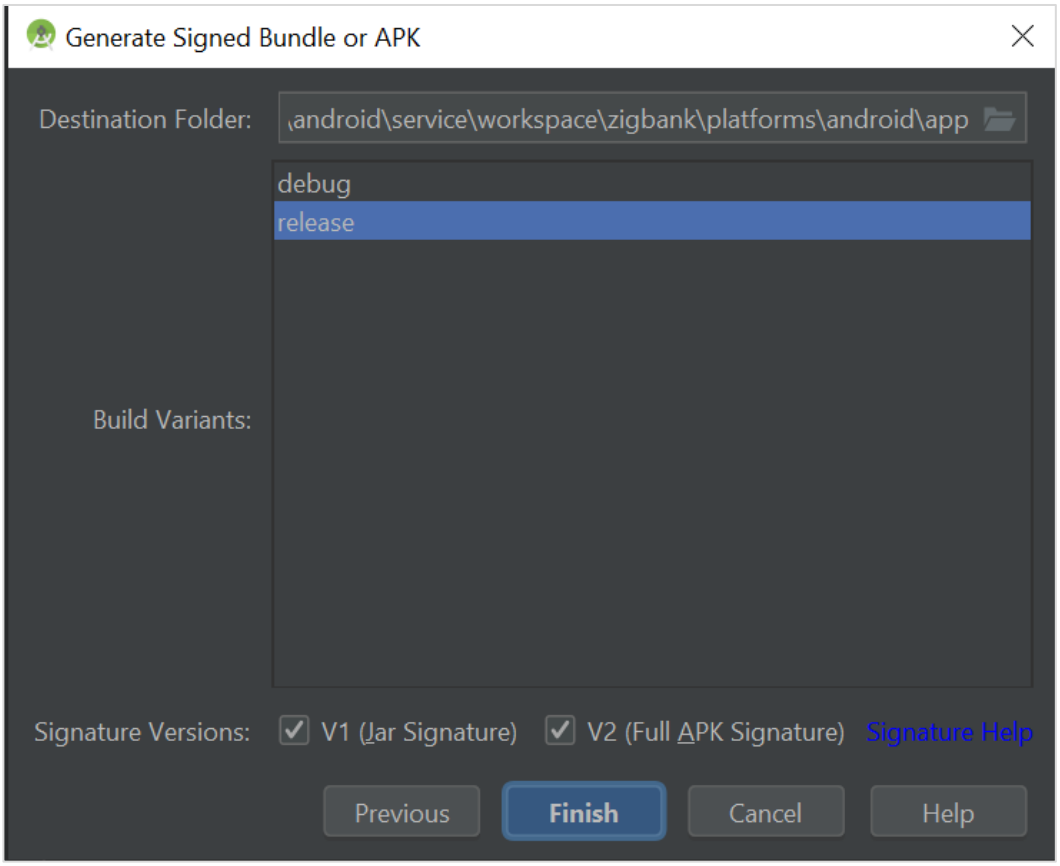


10. If you have an existing keystore.jks file then select choose Existing else click on Create New



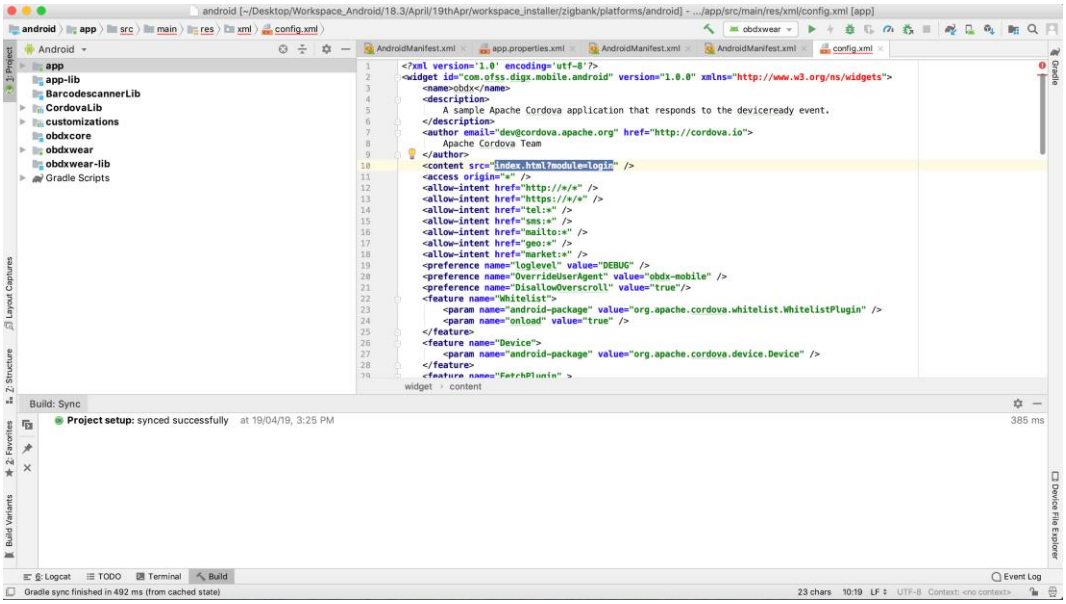


11. Select **Build Type** as **Release**, **Signature Version as V1(JAR Signature) and V2(Full APK Signature)** and Change APK Destination folder if you want and click on Finish



12. This will generate APK by the given name and destination folder. Default APK Destination folder is **zigbank\platforms\android\app\release**
13. Run the App and select Device or Simulator.
14. **Repeat same steps (From step 8 and obdxwear as module) for OBDX Wear App for Release Signing.** Use proguard-rules.pro from **workspace_installer\zigbank\platforms\android\obdxwear** using explorer. The select obdxwear as the module and follow same signing steps with same keystore.
15. The application has a config page at launch to enter the URL of the server (for development only). To remove this page, update the config.xml as shown below

The application has config page to add URL. This is for development purpose only and can be removed using below step. (Update content src tag)



16. Application will work on https only. If you want to run application on http then set targetSdkVersion, compileSdkVersion to 30 and buildToolsVersion to 30.0.3 in app's build.gradle(zigbank\platforms\android\app\) and remove remove below code from obdx.conf(config/obdx.conf).

```
<IfModule mod_headers.c>
    <If "%{HTTP_USER_AGENT} =~ /obdx-mobile-android/">
        Header edit Set-Cookie ^(.*)$ $1;SameSite=None;Secure
    </If>
    <If "%{HTTP_USER_AGENT} =~ /obdx-softtoken/">
        Header edit Set-Cookie ^(.*)$ $1;SameSite=None;Secure
    </If>
</IfModule>
```

17. To enable App Widget, please enable below flag in app.properties file:

```
<bool name="ENABLE_WIDGET">true</bool>
```

18. Disable below flag to reset the Biometric Alternate login on Add/Remove Fingerprint from mobile.

```
<bool name="ALLOW_FACE_BIOMETRIC">false</bool>
```

Note – This reset feature will support only if above flag is false.

19. Maintenance page configs-

Enable below flag to show maintenance page when server is under maintenance

<string name="SHOW_MAINTENANCE_PAGE">true</string>

Also add the status code returned when server is under main in below property-

<string-array name="MAINTENANCE_PAGE_STATUS_CODE">

<item>Your Status Code</item>

</string-array>

Note- You can add multiple status code

20. To disable caching in app, make below flag to flase

<bool name="ENABLE_CACHING">true</bool>

6. OBDX Authenticator Application

1. This is an Authenticator Application which is used when bank has enabled Soft Token Authentication as Authentication mechanism for any transaction. This application basically supports one of below authentication:
   - HOTP: Random based Soft Token
   - TOTP: Time based Soft Token

2. Users should have this application installed and logged in and PIN is set before initiating any transaction which needs this token.
3. Based on the configuration set, user can any time log in with PIN and check the token and use that token for completing any transaction based on “Soft Token Authentication”

6.1 Authenticator UI (Follow any one step below)

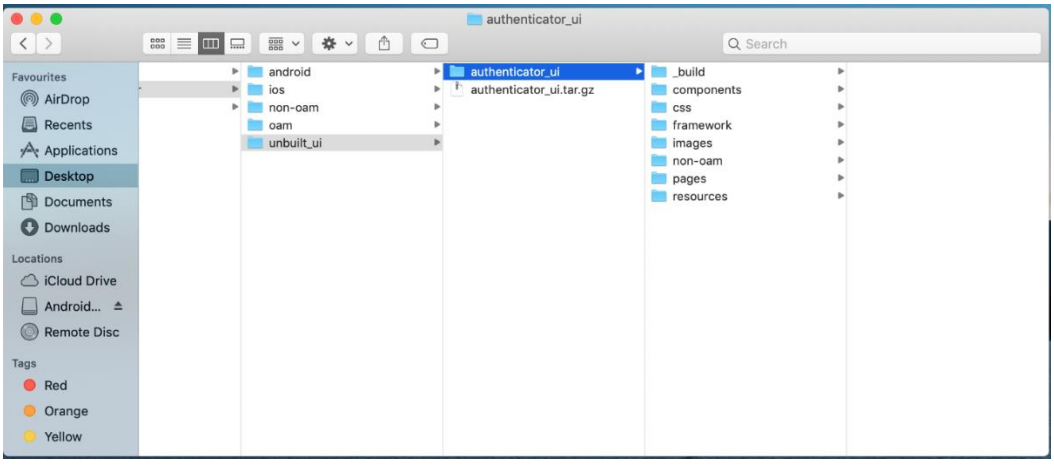
6.1.1 Using built UI

For TOKEN-BASED - Unzip dist.tar.gz directory fromOBDX_Patch_Mobile\authenticator\TOKEN-BASED

6.1.2 Building UI manually

Extract authenticator_ui.tar.gz from OBDX_Patch_Mobile\authenticator\unbuilt_ui.

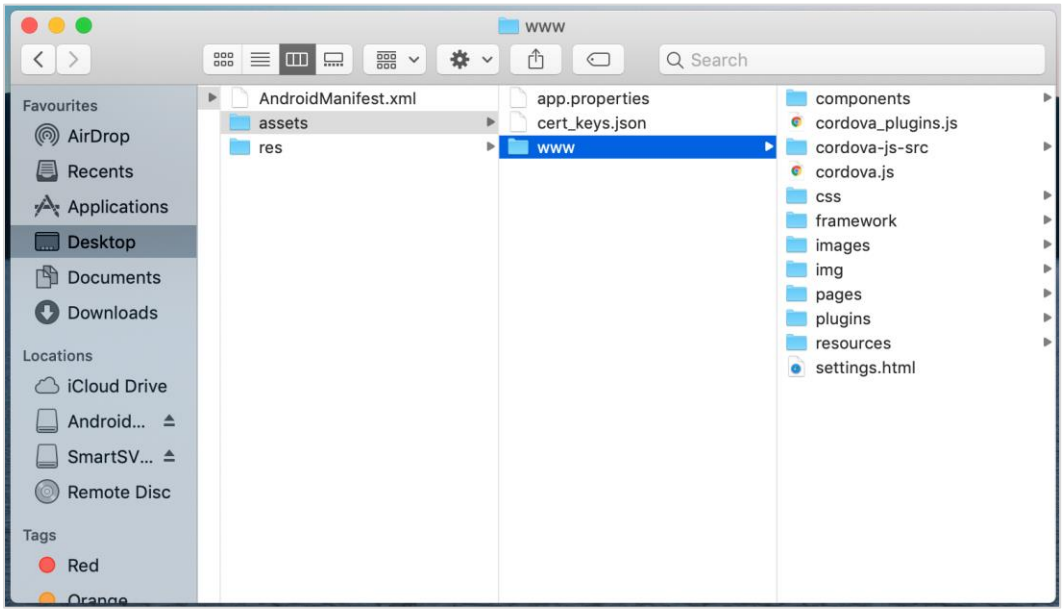
The folder structure is as shown:



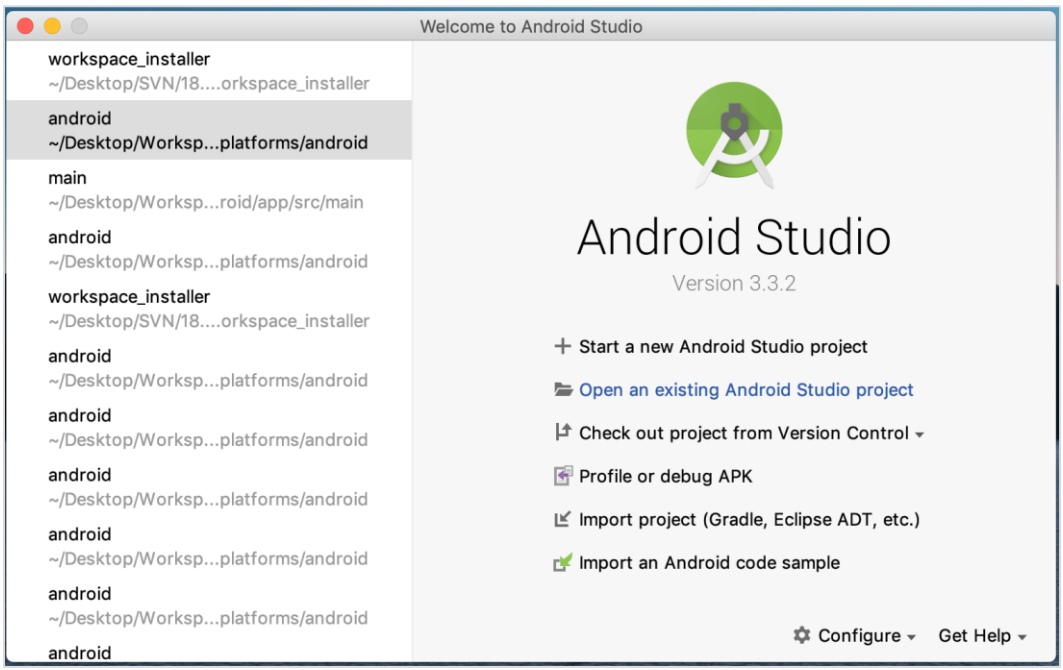
6.2 Authenticator Application Workspace Setup

1. Copy UI (Directories – components, css, framework, images, pages, resources)from /dist directory to workspace/installer/app/src/main/assets/www/

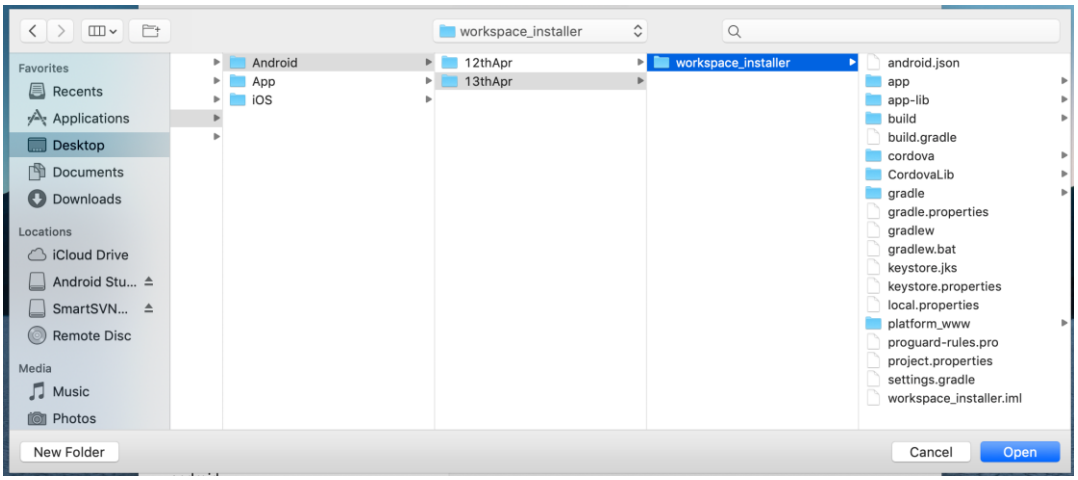
   In case any popup appears, click replace



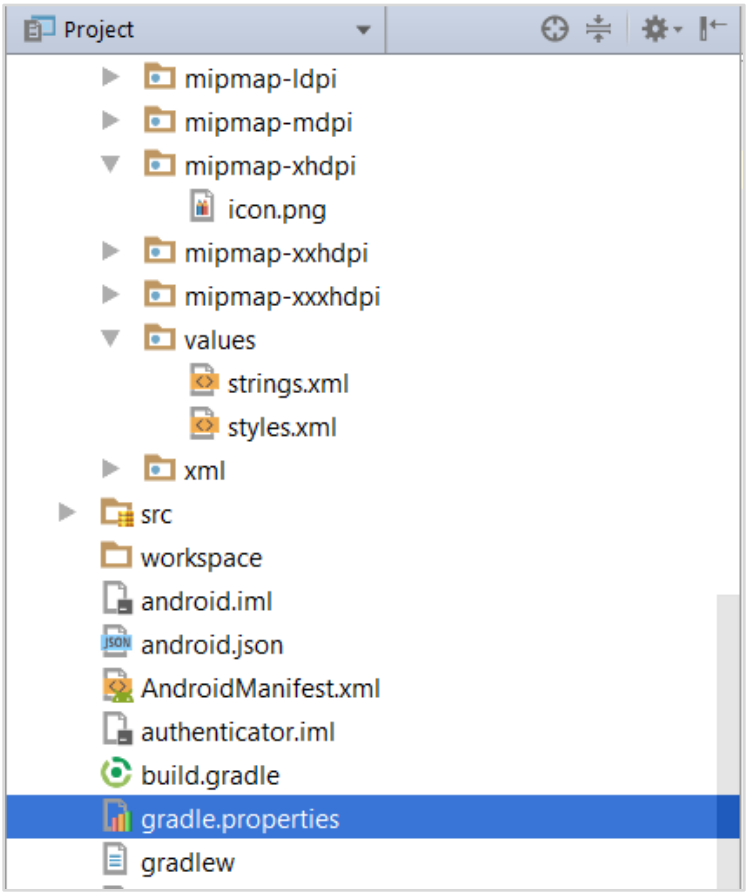
2. Launch Android Studio and open existing project



3. Open OBDX_Installer/workspace_installer folder in Android Studio.

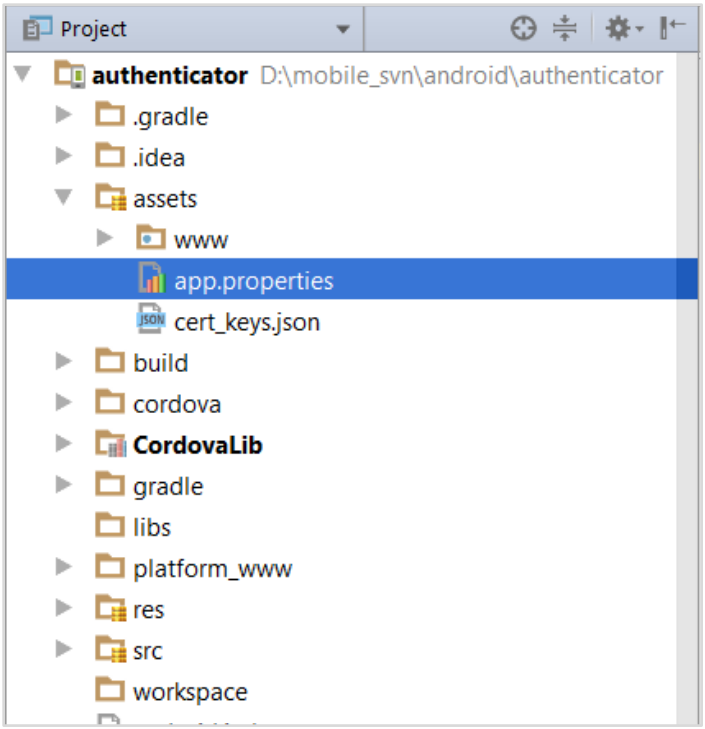


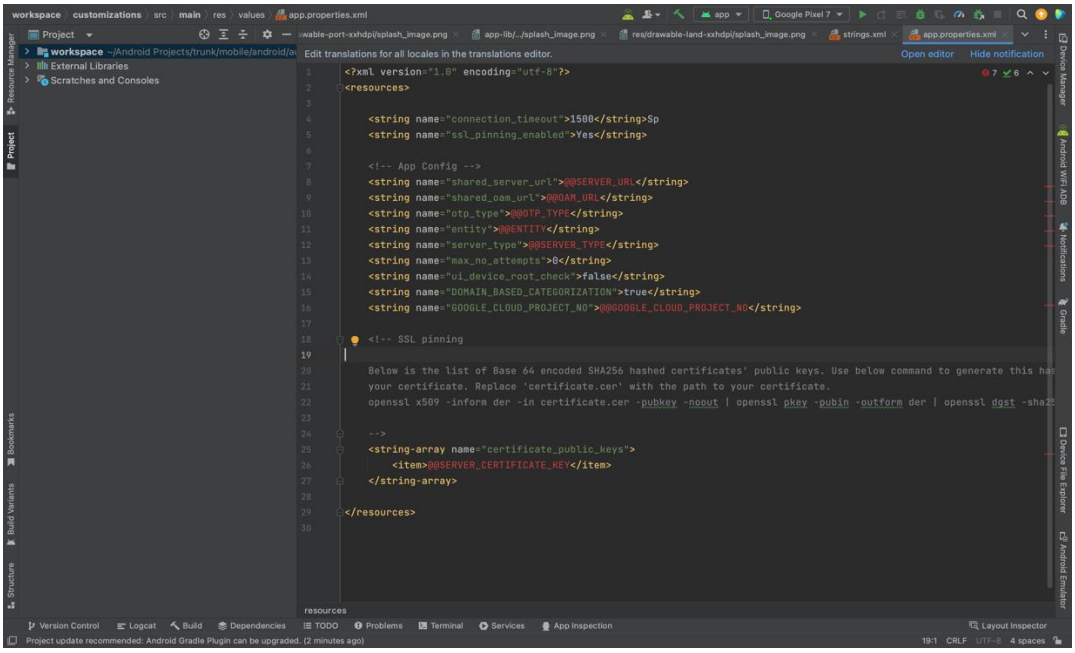
4. Open gradle.properties file and update following properties with relevant proxy address if required



```
systemProp.http.proxyHost = <proxy_address>
systemProp.https.proxyPort = <port_number>
systemProp.https.proxyHost = <proxy_address>
systemProp.http.proxyPort = <port_number>
```

5. Open "*assets\app.properties*" file and update following properties as per requirement





Set OTP type to HOTP/TOTP as per requirement.

Set Server Type to OBDXTOKEN

Set MAX No Attempts greater than 0

Set UI Device root check to true if you want to add check on login button.

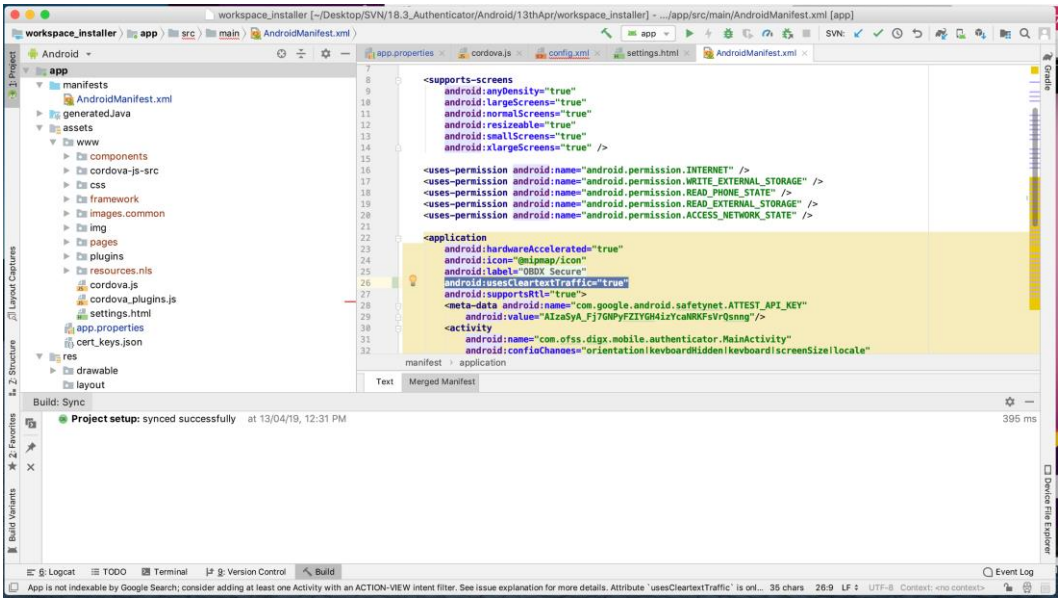
Note: If selected authentication mechanism is not OAM based then remove "*shared_oam_url*" property.

6. Click Build → Clean & Build → Rebuild project in Android Studio.
7. Click on Build → Edit Build Type → app → release

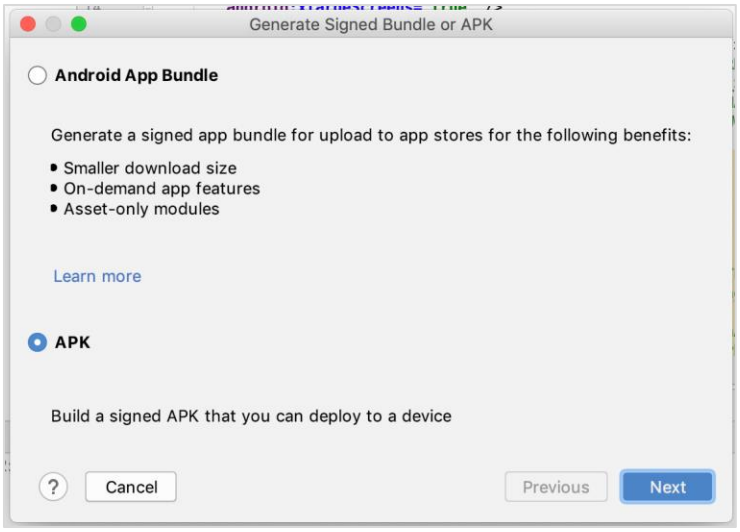
   Enable minify → true

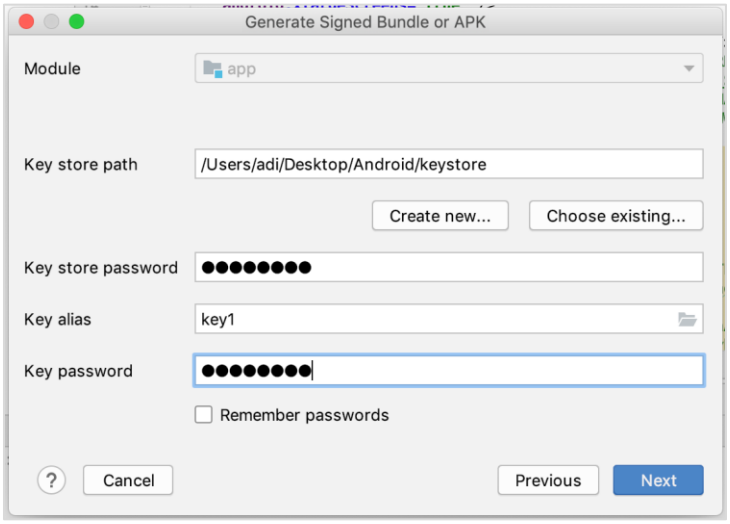
   Add progurard file from workspace_installer/proguard-rules.pro

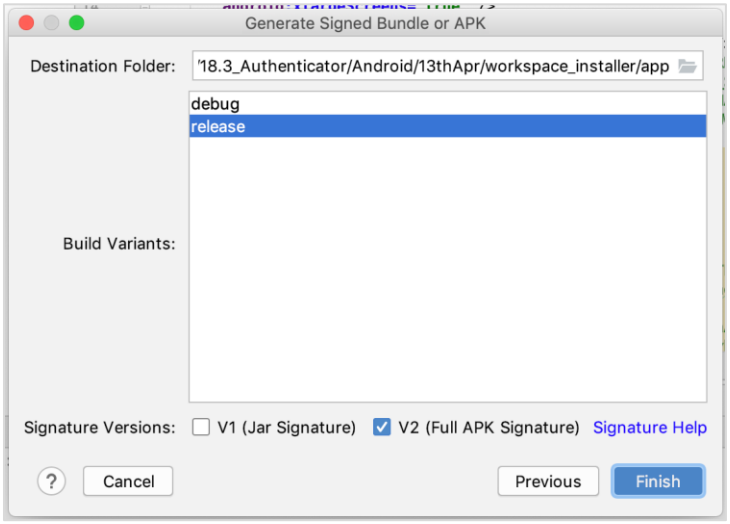
   Click OK
8. If using http protocol for development add (android:usesCleartextTraffic="true") to application tag of AndroidManifest.xml



9. **For Generating Signed Apk:** To Generate release-signed apk as follows:
10. On menu bar click on Build -> Generate Signed Apk

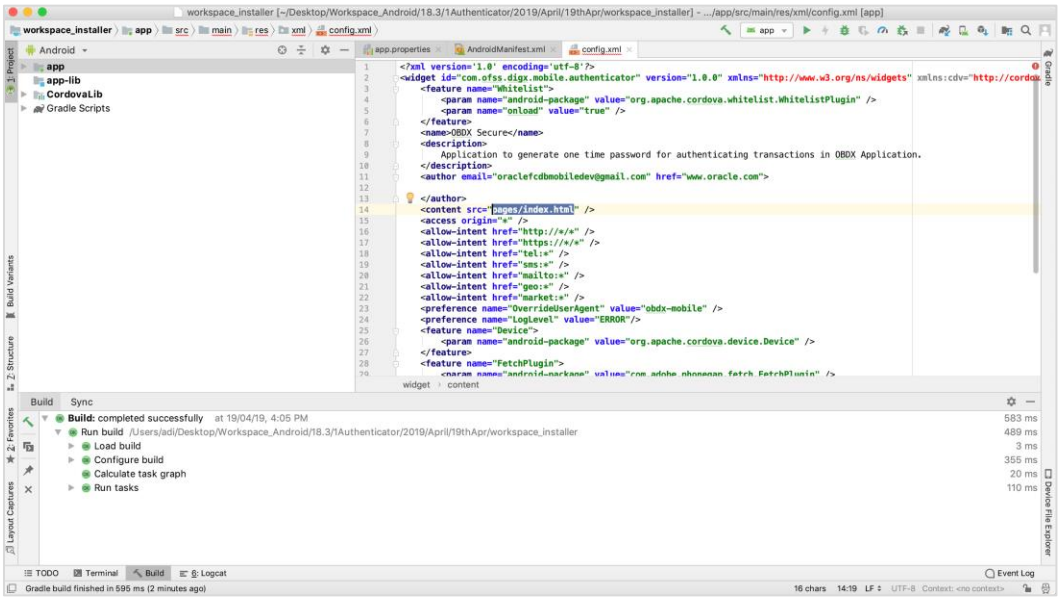

Generate Signed Bundle or APK
Android App Bundle
Generate a signed app bundle for upload to app stores for the following benefits:
Smaller download size
On-demand app features
Asset-only modules
Learn more
APK
Build a signed APK that you can deploy to a device
Cancel
Previous
Next


Generate Signed Bundle or APK
Module
app
Key store path
/Users/adi/Desktop/Android/keystore
Create new...
Choose existing...
Key store password
Key alias
key1
Key password
Remember passwords
Cancel
Previous
Next


Generate Signed Bundle or APK
Destination Folder:
'18.3_Authenticator/Android/13thApr/workspace_installer/app
debug
release
Build Variants:
Signature Versions:
V1 (Jar Signature)
V2 (Full APK Signature)
Signature Help
Cancel
Previous
Finish

Click Finish to generate .apk

The application has config page to add URL. This is for development purpose only and can be removed using below step. (Update content src tag)



Home

7. Application Security Configuration

Root Check → Ensure Step 3.1 is completed

1. Open google developer console. Select your app then navigate to

   Setup-> App Integrity-> change option of Response Encryption

   In the window that appears, click Manage and download my response encryption keys and follow below steps to generate response encryption keys-

   a. Create a new private-public key pair. RSA key size must be 2048 bits using below command-

      openssl genrsa -aes128 -out your_path/private.pem 2048

      Then use your password phrase for creating private.pem and also use the same password for verifying the private.pem. Then hit the below command.

      openssl rsa -in your_path/private.pem -pubout -out your_path/public.pem

      Enter the same password which you have used while creating private.pem. These two files will now appear on your mentioned path. Then upload the public.pem file on the window which was appeared after clicking on Manage and download my response encryption keys option.Once you upload the public.pem file it will automatically download your_app_pkg_name.enc file. Then hit below command as,

      openssl rsautl -decrypt -oaep -inkey your_path/private.pem -in your_app_pkg_name.enc -out your_path/api_keys.txt

      Enter the password for private.pem. It will create api_keys.tx file on your path. It must be consist of VERIFICATION_KEY and DECRYPTION_KEY.

2. Maintain this VERIFICATION_KEY and DECRYPTION_KEY in **DIGX_FW_CONFIG_ALL_B** table corresponding to the following keys respectivel:

   PLAY_INTEGRITY_ENCRYPTION_KEY and **PLAY_INTEGRITY_DECRYPTION_KEY**

   An example query will be:

   update DIGX_FW_CONFIG_ALL_B set prop_value = 'YOUR_DECRYPTION_KEY' where prop_id = 'PLAY_INTEGRITY_DECRYPTION_KEY';

   update DIGX_FW_CONFIG_ALL_B set prop_value = 'YOUR_ENCRYPTION_KEY' where prop_id = 'PLAY_INTEGRITY_ENCRYPTION_KEY';

3. Similarly, Obtain the same keys for authenticator app by using above step 1 and then maintain those in **DIGX_FW_CONFIG_ALL_B** table corresponding to the following keys respectivel:

   PLAY_INTEGRITY_ENCRYPTION_KEY_AUTHENTICATOR and **PLAY_INTEGRITY_DECRYPTION_KEY_AUTHENTICATOR**

An example query will be:

update DIGX_FW_CONFIG_ALL_B set prop_value = 'YOUR_DECRYPTION_KEY' where prop_id = 'PLAY_INTEGRITY_DECRYPTION_KEY_AUTHENTICATOR';

update DIGX_FW_CONFIG_ALL_B set prop_value = 'YOUR_ENCRYPTION_KEY' where prop_id = 'PLAY_INTEGRITY_ENCRYPTION_KEY_AUTHENTICATOR';

4. Similarly, we also have to maintain package names of Servicing and Authenticator app in the same table, i.e. **DIGX_FW_CONFIG_ALL_B** corresponding to the following keys respectively:

   ANDROID_SERVICING_PACKAGE and ANDROID_AUTHENTICATOR_PACKAGE

   An example query will be:

   insert into digx_fw_config_all_b (PROP_ID, CATEGORY_ID, PROP_VALUE, FACTORY_SHIPPED_FLAG, PROP_COMMENTS, SUMMARY_TEXT, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE, OBJECT_STATUS, OBJECT_VERSION_NUMBER) values ('ANDROID_SERVICING_PACKAGE', 'mobileconfig', 'com.ofss.zigbank', 'N', '', 'Stores device id in OUD', 'ofssuser', sysdate, 'ofssuser', sysdate, 'Y', 1,);

SSL Pinning

5. Get the list of Base 64 encoded SHA256 hashed certificates' public keys of server's valid certificates. Use below command to generate this hash for your certificate. Replace '<certificate.der>' with the path to your certificate.

   openssl x509 -inform der -in <certificate.der> -pubkey -noout | openssl pkey -pubin -outform der | openssl dgst -sha256 -binary | openssl enc -base64

6. Add the hashed keys generated in point 6 to **zigbank\platforms\android\customizations\src\main\res\values\app.properties.xml file** in 'certificate_public_keys' array. Append this key to 'sha256/' in an <item> tag as shown below. Multiple certificate keys can be added to 'certificate_public_keys' array by adding them in <item> tags.

   Eg.:

```
<string-array name="certificate_public_keys">

    <item>sha256/5kJvNEMw0KjrCAu7eXY5HZdvyCS13BbA0VJG1RSP91w=</item>

</string-array>
```

   Eg. for multiple certificates (In case OAM/IDCS is used):

```
<string-array name="certificate_public_keys">

    <item>sha256/5kJvNEMw0KjrCAu7eXY5HZdvyCS13BbA0VJG1RSP91w=</item>
     <item>sha256/3rgsgghoqrDegekpkkgk92Fgw1w7exyYCS1okef9Oo1w=</item>

</string-array>
```

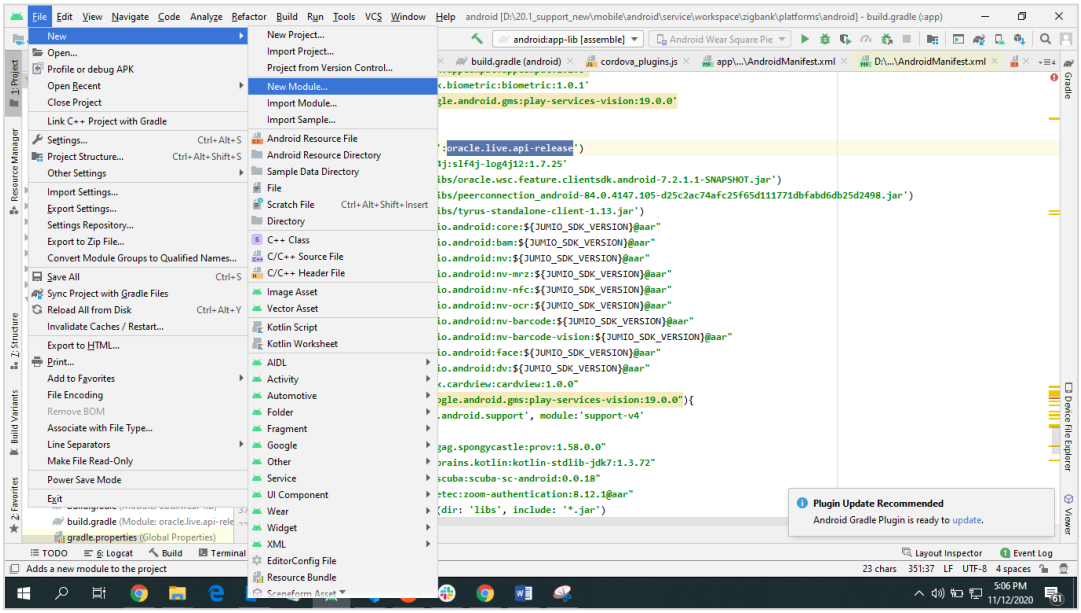
Home

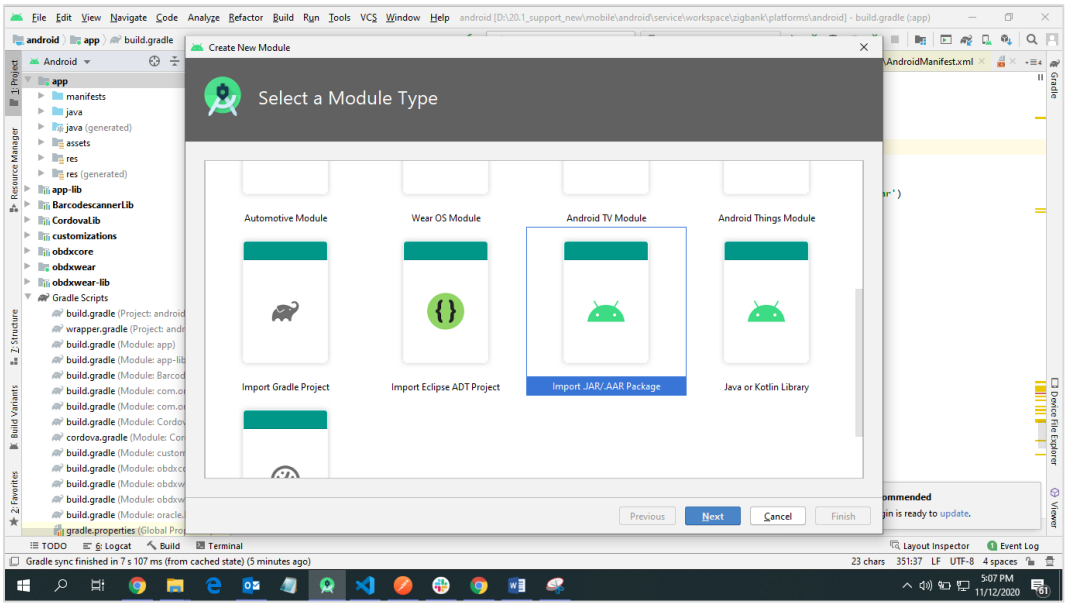
8. Live Experience With Jumio Integration

1. Download live experience android sdk from below download link.

   https://www.oracle.com/downloads/cloud/oracle-live-experience-downloads.html

2. Import ‘oracle.live.api-release’ file as a New Module.





3. Add Live Experience Client ID and Cloud Address in below two properties under app.properties.xml(zigbank\platforms\android\customizations\src\main\res\values)

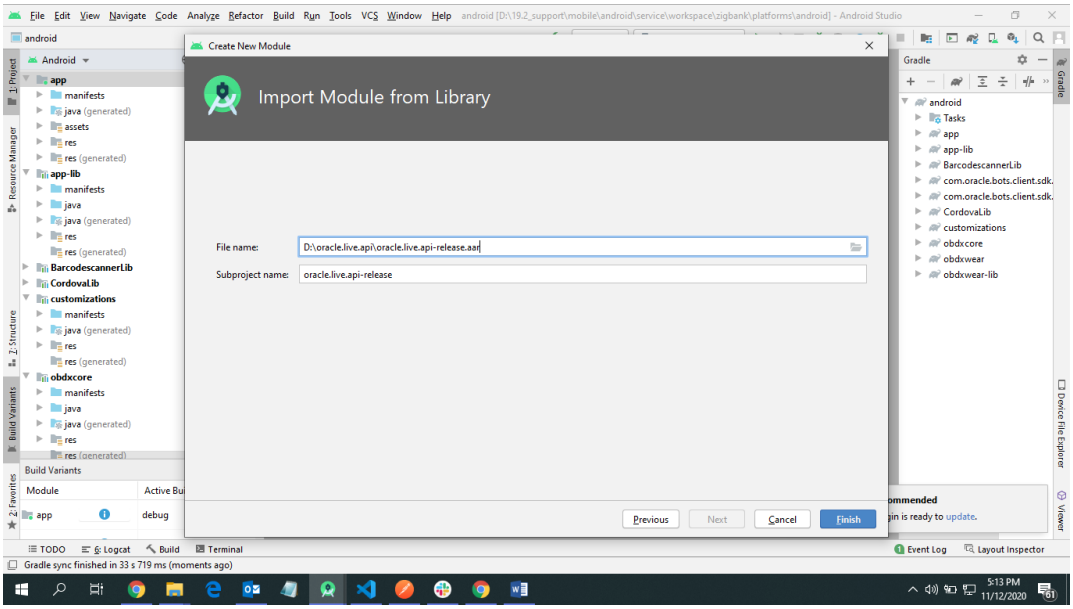
   <string name="LX_CLIENT_ID">@@CLIENT_ID</string>

   <string name="LX_ADDRESS">@@ADDRESS</string>

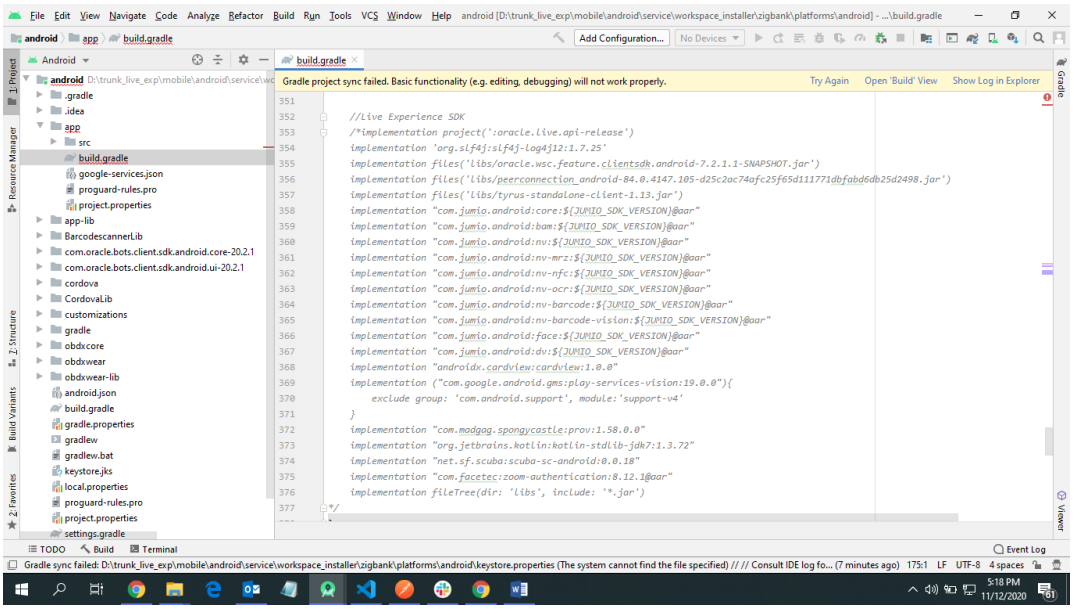
Note: Add LX_ADDRESS without https://

For example. If the LX_ADDRESS is https://live.oraclecloud.com then add only live.oraclecloud.com.

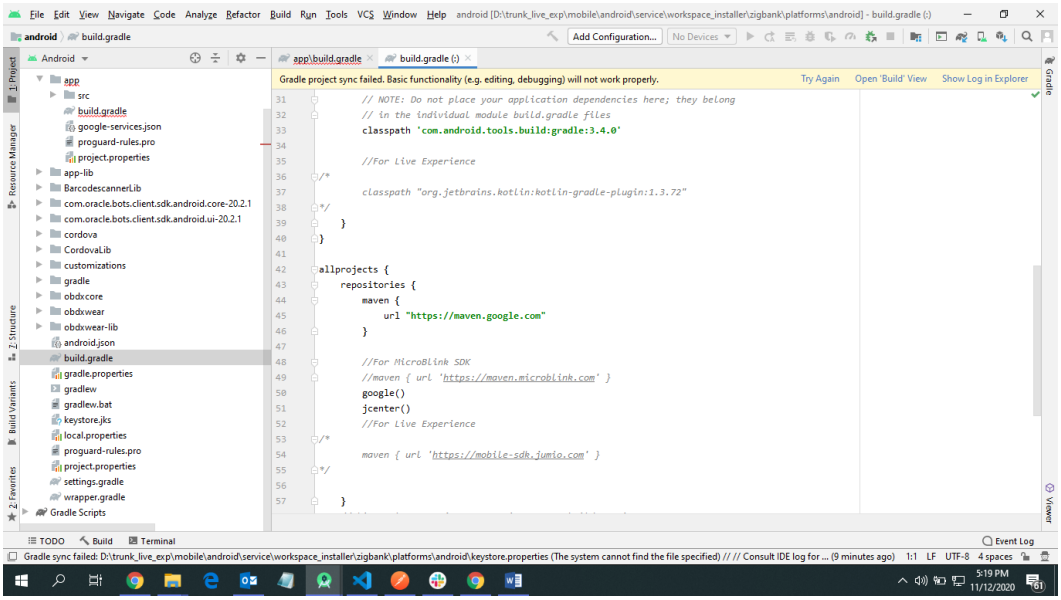
4. Click Next and navigate to oracle.live.api-release aar file location and click Finish.



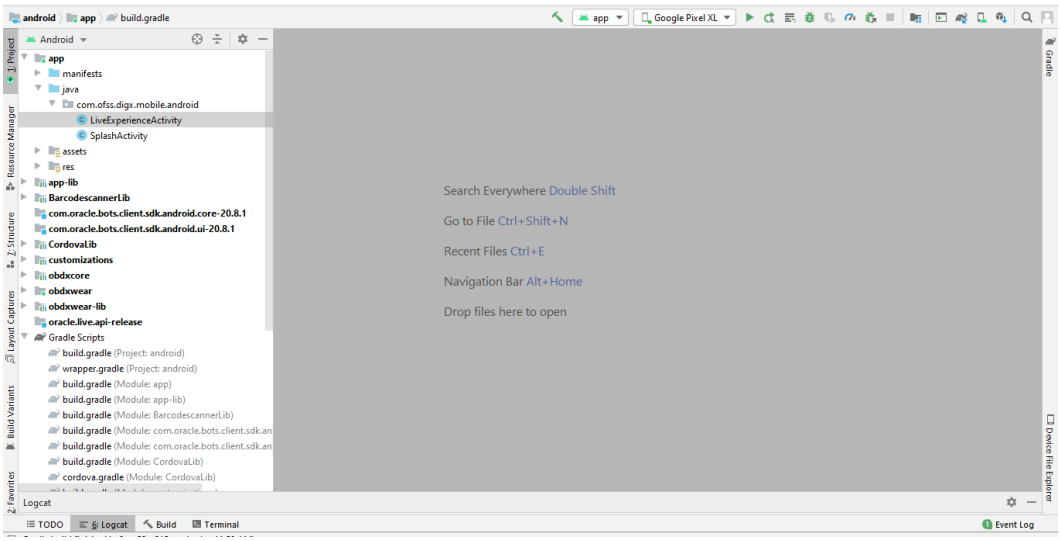
5. Un-comment the Live Experience SDK's from zigbank\platforms\android\app\build.gradle.



6. Un-comment the gradle maven files for Live Experience from zigbank\platforms\android\ build.gradle



7. Add LiveExperienceActivtiy.java folder from AppExtensions\live experience\ at zigbank\platforms\android\app\src\main\java\com\ofss\digx\mobile\android

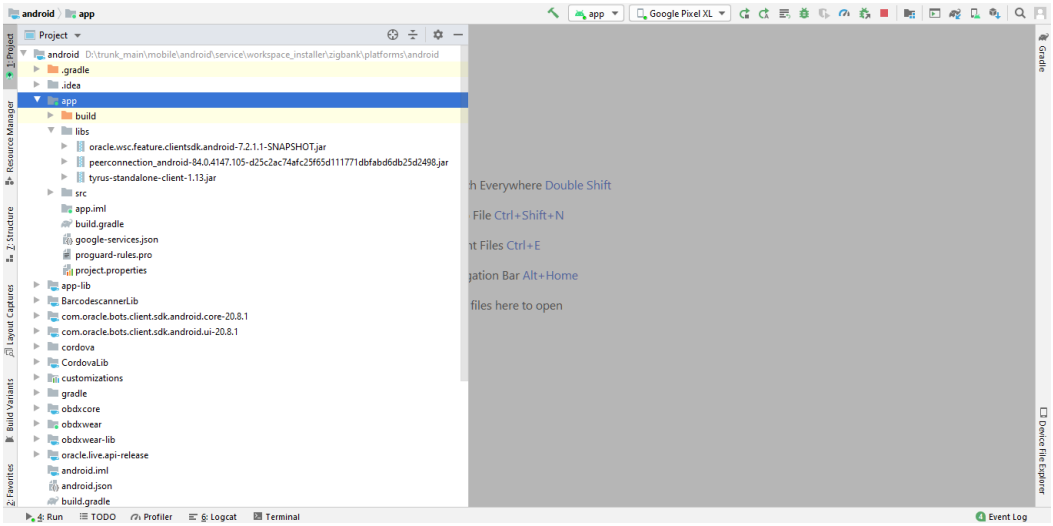


8. Add libs folder at zigbank\platforms\android\app and copy below jars from downloaded sdk folder in it.

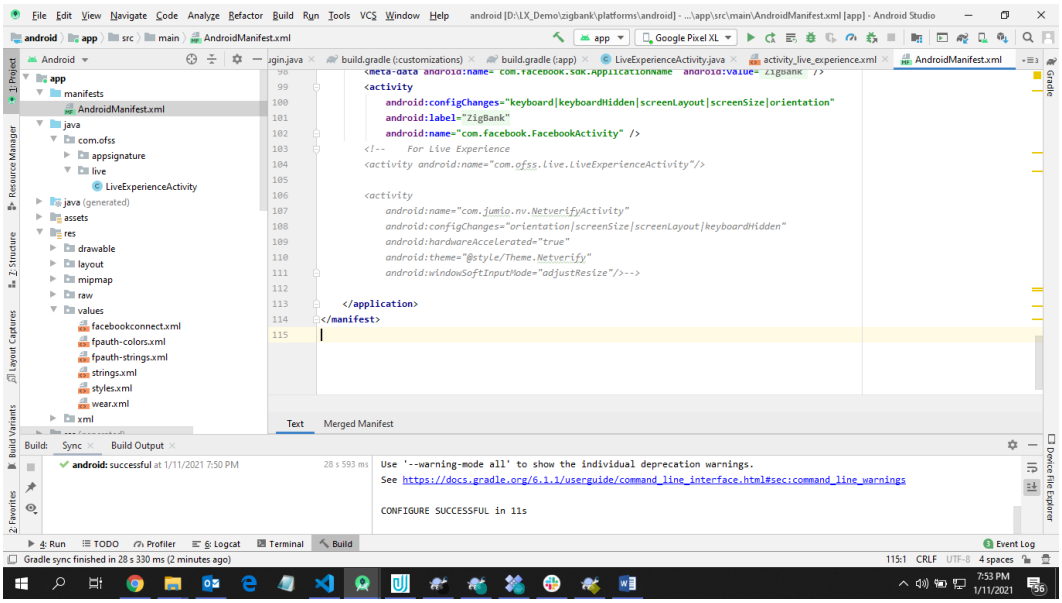
   i) oracle.wsc.feature.clientsdk.android-7.2.1.1-SNAPSHOT.jar

   ii) peerconnection_android-84.0.4147.105-25c2ac74afc25f65d111771dbfabd6db25d2498.jar

   iii) tyrus-standalone-client-1.13.jar



9. Un-comment LiveExperienceActivity and NetverifyActivity from zigbank\platforms\android\app\src\main\AndroidManifest.xml


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
android [D:\LX_Demo\zigbank\platforms\android] - ...\app\src\main\AndroidManifest.xml [app] - Android Studio
android app src main AndroidManifest.xml
app
Google Pixel XL
Android
build.gradle (:customizations)
build.gradle (:app)
LiveExperienceActivity.java
activity_live_experience.xml
AndroidManifest.xml
app
manifests
AndroidManifest.xml
java
com.ofss
appsignature
live
LiveExperienceActivity
java (generated)
assets
res
drawable
layout
mipmap
raw
values
facebookconnect.xml
fpauth-colors.xml
fpauth-strings.xml
strings.xml
styles.xml
wear.xml
xml
<activity
android:configChanges="keyboard|keyboardHidden|screenLayout|screenSize|orientation"
android:label="ZigBank"
android:name="com.facebook.FacebookActivity" />
<!-- For Live Experience
<activity android:name="com.ofss.Live.LiveExperienceActivity"/>
<activity
android:name="com.jumio.nv.NetverifyActivity"
android:configChanges="orientation|screenSize|screenLayout|keyboardHidden"
android:hardwareAccelerated="true"
android:theme="@style/Theme.Netverify"
android:windowSoftInputMode="adjustResize"/>-->
</application>
</manifest>
Text
Merged Manifest
Build:
Sync
Build Output
android: successful at 1/11/2021 7:50 PM
28 s 593 ms
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.1.1/userguide/command_line_interface.html#sec:command_line_warnings
CONFIGURE SUCCESSFUL in 11s
4: Run
TODO
Profiler
6: Logcat
Terminal
Build
Event Log
Gradle sync finished in 28 s 330 ms (2 minutes ago)
115:1 CRLF UTF-8 4 spaces
7:53 PM
1/11/2021

Home

9. Adding Custom Cordova Plugin

Step 1 -

Create java folder and add yout package under app(zigbank\platforms\android\app)

Create java file under your package which will extends CordovaPlugin

Override execute method with JsonArray as a parameter

Retrive jsonobject from JsonArray and get the data which passed from js file

Example:

```
public class GetDirectionMapPlugin extends CordovaPlugin {
    @Override
    public boolean execute(String action, JSONArray args, CallbackContext callbackContext)
            throws JSONException {
        try{
            JSONObject object = args.getJSONObject(0);
            String yourKey = object.getString("your_key");
        }catch (Exception e){
            Log.e(TAG,e.getMessage());
        }
        return true;
    }
}
```

Step 2 –

Create plugin file under plugins folder of

www(zigbank\platforms\android\service\workspace\app\src\main\assets\www\plugins)

Example:

```
        cordova.define("cordova-plugin-getdirection", function(require, exports, module) {
```

```
    var exec = cordova.require('cordova/exec');
    exports.navigate = function(args, successCallback, errorCallback) {
        cordova.exec(successCallback, errorCallback, "GetDirectionMapPlugin", "direction",
            [args]);
    };
});
```

cordova-plugin-getdirection.getDirectionPlugin -> user defined id from cordova_plugin.js(zigbank\platforms\android\service\workspace\app\src\main\assets\www\cordova_plugin.js)

GetDirectionMapPlugin-> name of java plugin class

direction -> action

navigate -> this can be use in js file to this function

Step 3 –

Make entry of plugin in cordova_plugin.js(zigbank\platforms\android\service\workspace\zigbank\platforms\android\app\src\main\assets\www) as below ->

Example:

```
{
    "id": "cordova-plugin-getdirection.getDirectionPlugin", -> user defined id
    "file": "plugins/cordova-plugin-getdirection/www/mapgetdirection.js", -> path of plugin js
    file
    "pluginId": "cordova-plugin-getdirection",
        "clobbers": [
    "window.getDirection" -> this can be used in js file to call plugin
]
}
```

Step 4 -

Make entry of java plugin class in config.xml(zigbank\platforms\android\service\workspace\zigbank\platforms\android\app\src\main\res\xml) file of app as below -

Example:

```
<feature name="GetDirectionMapPlugin">
<param name="android-package" value="Your_Plugin_Java_Class_Path" />
</feature>
```

GetDirectionMapPlugin -> Name of java plugin class

Step 5 -

Plugin calling in js file ->

Example:

```
window.getDirection.navigate({
originLatLng: origin,
destinationLatLng: location
})
```

window.getDirection -> clobber define in the cordova_plugin.js file

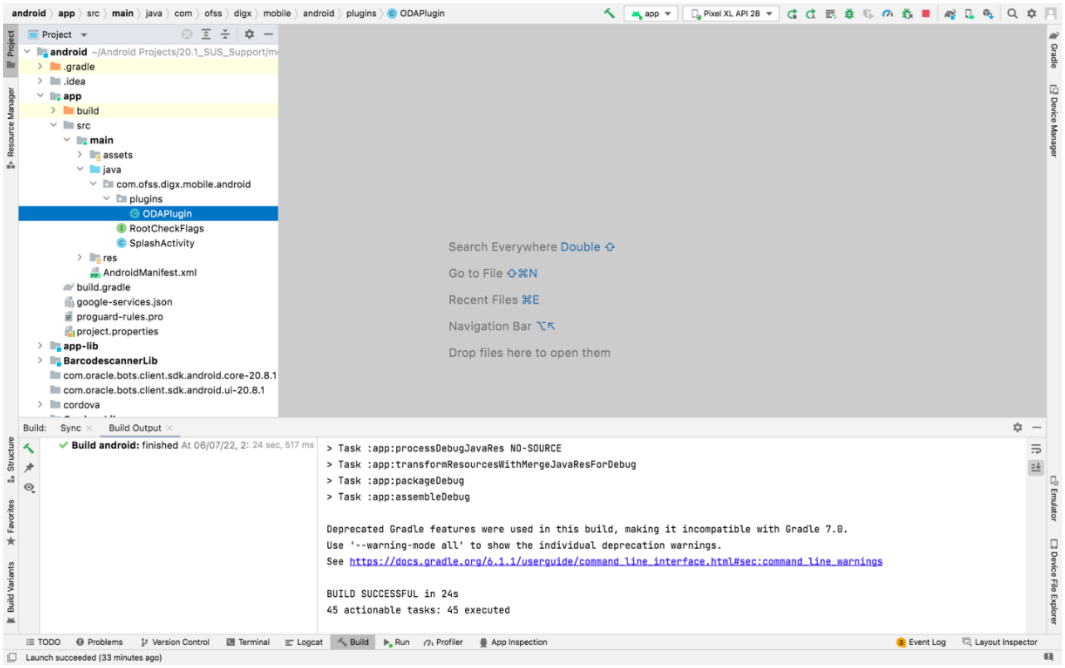
navigate -> name of the function defined in plugin js file

Home

10. ODA Chatbot Inclusion

To enable ODA Chatbot services in the mobile app, the following changes needs to be made:

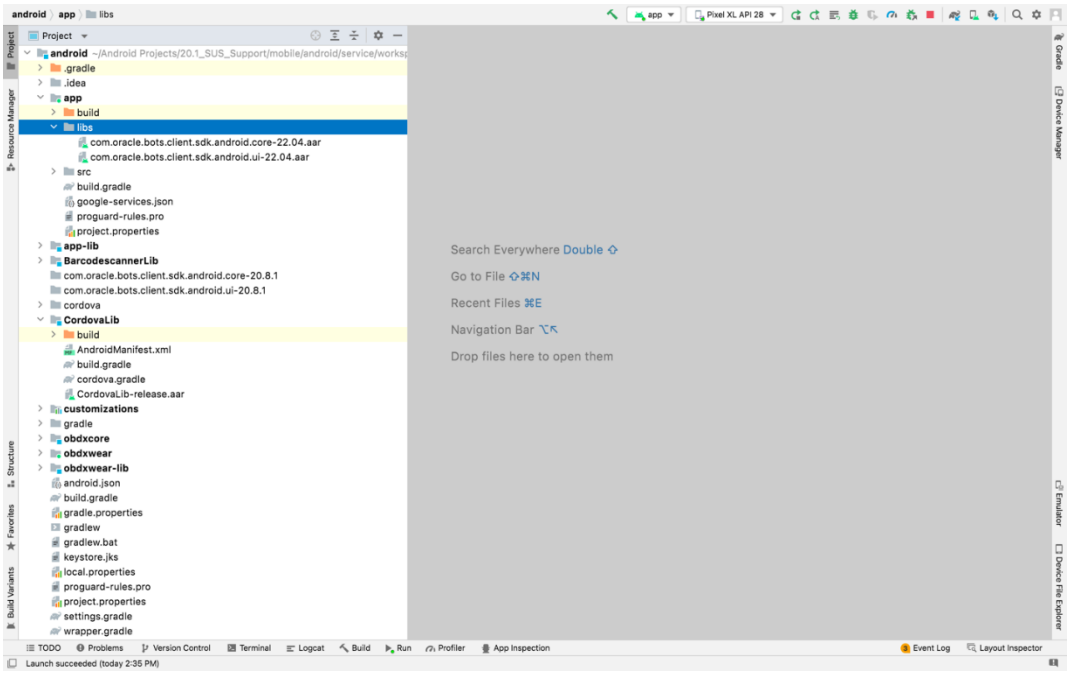
1. Copy ODAPlugin.java from workspace_installer/AppExtension/oda to workspace_installer/zigbank/platforms/android/app/src/main/java/com/ofss/digx/mobile/android/plugins/



2. Download ODA Android sdk from below link-

https://www.oracle.com/downloads/cloud/amce-downloads.html

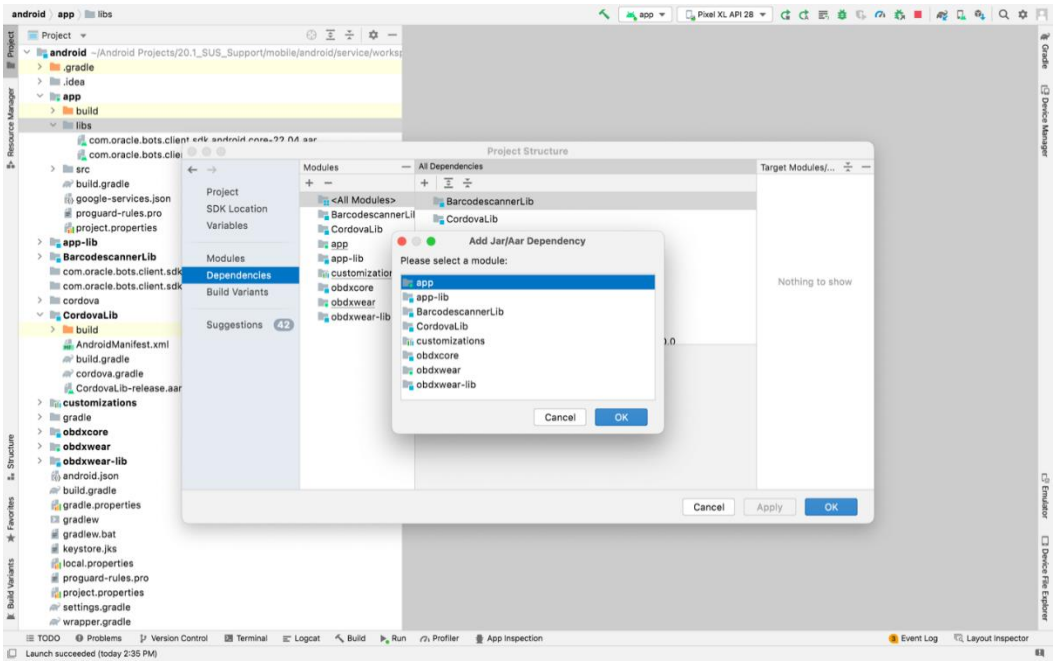
3. Add libs folder at zigbank\platforms\android\app and copy below files from downloaded sdk folder in it.
   a. com.oracle.bots.client.sdk.android.core-xx.aar
   b. com.oracle.bots.client.sdk.android.ui-22.04.aar



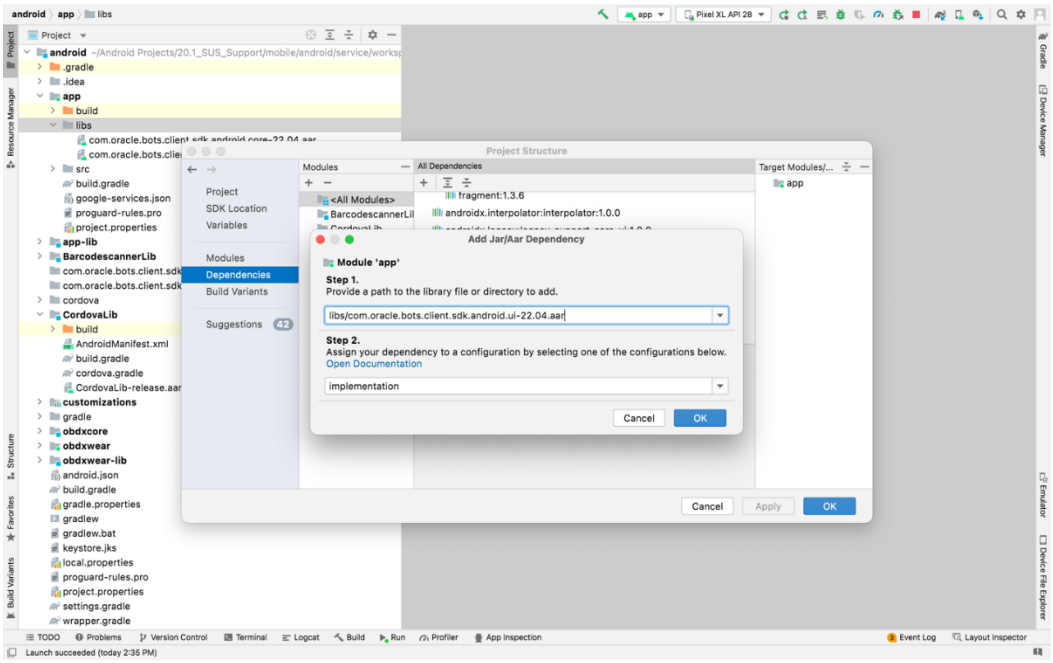
4. In Android Studio follow below steps-

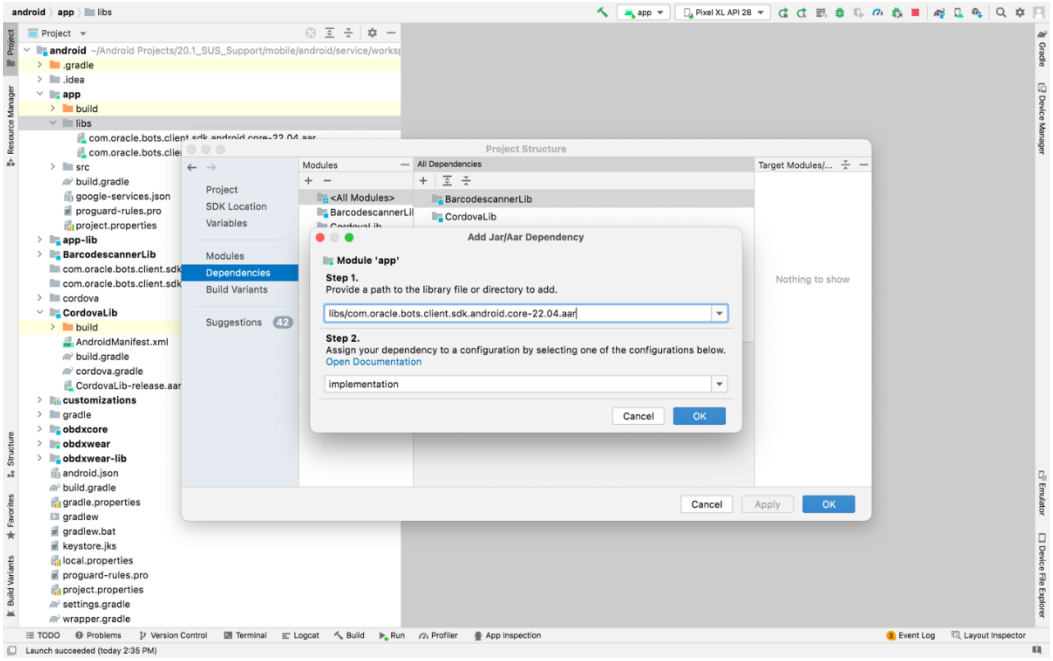
File -> Project Structure -> Dependencies

5. Click on "+" icon and select JR/AAR Dependency and select app module and click Ok.



6. Add both .aar file paths from step3. Then click Apply and Ok.





7. Add Chatbot ID and Chatbot URL in app.properties.xml(zigbank\platforms\android\customizations\src\main\res\values)

   <string name="CHATBOT_ID">@@CHATBOT_ID</string>

   <string name="CHATBOT_URL">@@CHATBOT_URL</string>

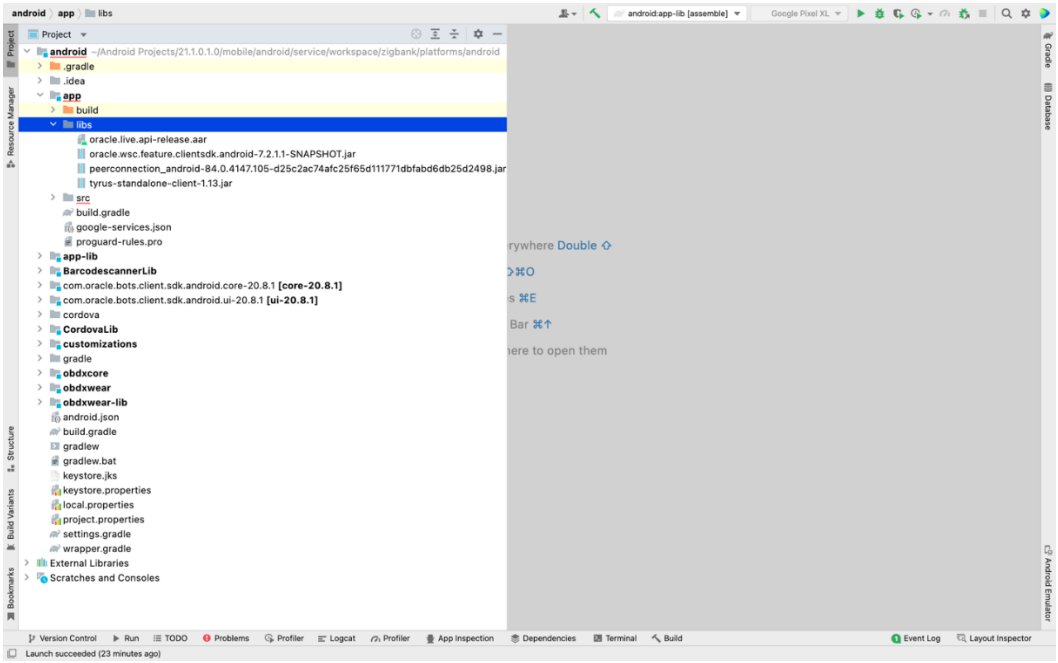
11. Live Experience Integration

1. Download live experience android sdk from below download link.

   https://www.oracle.com/downloads/cloud/oracle-live-experience-downloads.html

2. Add libs folder at zigbank\platforms\android\app and copy below jars from downloaded sdk folder in it.

   - oracle.wsc.feature.clientsdk.android-7.2.1.1-SNAPSHOT.jar
   - peerconnection_android-84.0.4147.105-25c2ac74afc25f65d111771dbfabd6db25d2498.jar
   - tyrus-standalone-client-1.13.jar
   - oracle.live.api-release.aar



3. Add Live Experience Client ID and Cloud Address in below two properties under

   app.properties.xml(zigbank\platforms\android\customizations\src\main\res\values)

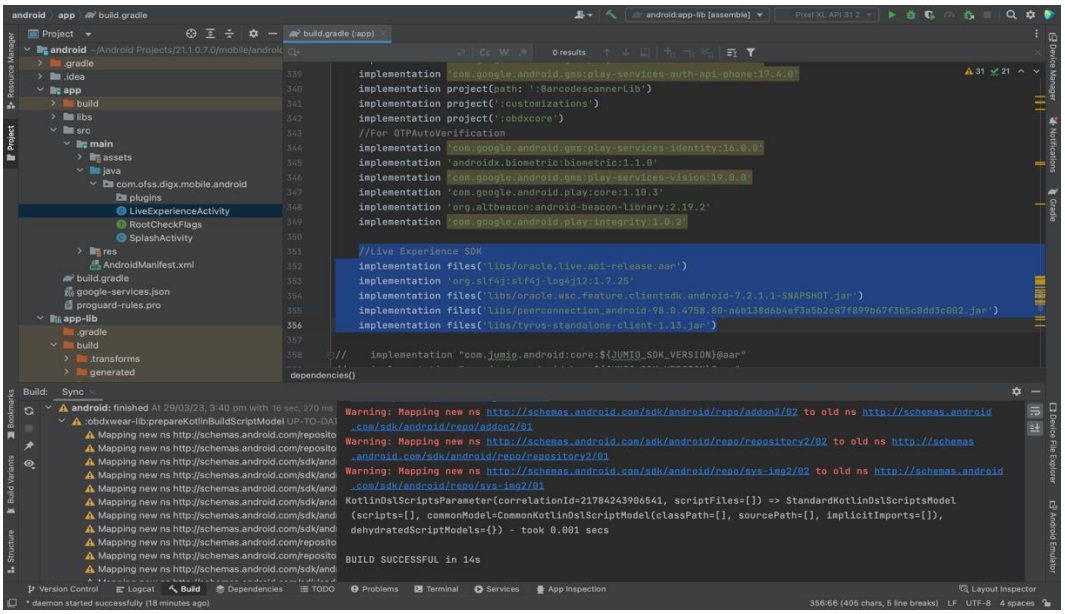
   <string name="LX_CLIENT_ID">@@CLIENT_ID</string>

   <string name="LX_ADDRESS">@@ADDRESS</string>

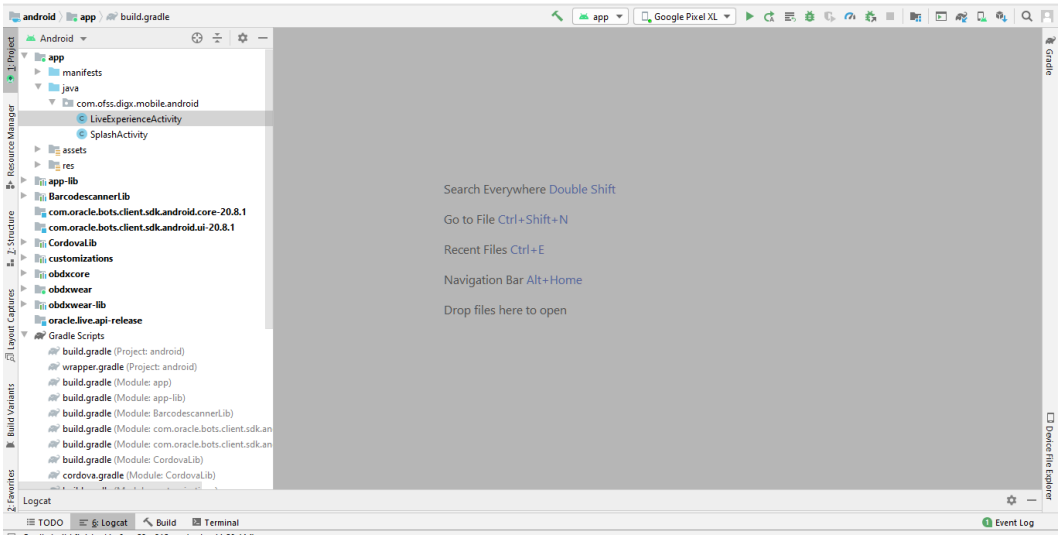
   Note: Add LX_ADDRESS without https://

   For example. If the LX_ADDRESS is https://live.oraclecloud.com then add only live.oraclecloud.com.

4. Un-comment the Live Experience SDK's from zigbank\platforms\android\app\build.gradle.



5. Add LiveExperienceActivtiy.java folder from AppExtensions\live experience\ at zigbank\platforms\android\app\src\main\java\com\ofss\digx\mobile\android



6. Un-comment LiveExperienceActivity from zigbank\platforms\android\app\src\main\AndroidManifest.xml

12. Push Notification 2FA configuration

If Push notification 2fa is enabled at bank side for any transaction then, the screen displays message to wait for the push notification to accept/reject the transaction authentication. The message as well contains a timer of 5 minutes displayed on the UI. This value is set in the UI code. If bank needs to change this value, bank needs to update the value in UI code:

File path: channel/metadata/user-components/push-out-of-band/push-out-of-band/hook.js

Code to be changed: const mins = <<value>>;

Update the value to what bank needs to set it. This value is in minutes.

So, ideally 5 minutes (existing value in base UI code) is an ideal time. Any changes made in this value should satisfy below pre-condition.

1. There is an OTP expiration time set in “digx_fw_config_ALL_b” table.
2. Also, there is business policy check set to 10 minutes for validation of the generated 2fa token. Bank can write their own business policy where they can modify the 10 minutes time.

So, the time in UI code should not exceed 10 minutes and OTP expiration time in “digx_fw_config_ALL_b” table.

Home