

A PROJECT REPORT (CA316)
on
BUILDING A WEBRTC VIDEO CHAT APPLICATION

A report submitted in partial fulfilment of the requirement for the award of

The degree of

BACHELOR OF COMPUTER APPLICATION

by

Aman Bhardwaj (1000010458)

Ranjan Kumar(1000011369)

Under the Guidance of

Dr. Narendra Kumar

Assistant Professor

SOC-IT



SCHOOL OF COMPUTING
DIT UNIVERSITY, DEHRADUN

(State Private University through State Legislature Act No. 10 of 2013 of Uttarakhand and approved by UGC)

Mussoorie Diversion Road, Dehradun, Uttarakhand - 248009, India.

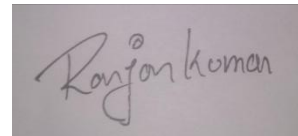
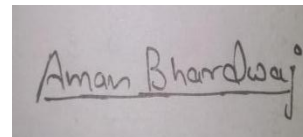
April,2021

CANDIDATES DECLARATION

I hereby certify that the work, which is being presented in the Report, entitled Building a WebRTC Video Chat Application, in partial fulfilment of the requirement for the award of the Degree of Bachelor of Computer Application and submitted to the DIT University is an authentic record of my work carried out during the period January 2021 to April 2021 under the guidance of Dr. Narendra kumar.

Date: 14th March 2021

Signature of Candidates

A photograph of a handwritten signature in black ink on a light-colored background. The signature appears to be 'Rajan Kumar' written in a cursive style.A photograph of a handwritten signature in black ink on a light-colored background. The signature appears to be 'Aman Bhardwaj' written in a cursive style, with a horizontal line drawn underneath the text.

ACKNOWLEDGEMENT

The joy of completing any task no matter how small or big it may be has its sense of satisfaction but our hard work will lose its value if every pupil's contribution is not recognized. We want to express our profound appreciation and admiration to each one of those behind the screen who guided and helped in the completion of our project work. We consider ourselves sufficiently fortunate to get such an opportunity to make a decent project which would surely add an advantage to our academic profile. We would like to express our gratitude to our project guide, **Dr.Narendra Kumar** for his constant Guidance and valuable help throughout the project. We also extend our thanks to our friends for their co-operation to complete this project.

Name of Candidates

Ranjan Kumar

Aman Bhardwaj

ABSTRACT

In the modern world and the rapid development of the Internet, communication between people is more important than ever, people are looking for new ways to make past communication between them without any problem, real-time communication is one of these ways. WebRTC (Real-time Web Communication) is a technology that enables real-time communication of audio, video, and data communications in real-time communication using web browsers using JavaScript APIs (Application Programming Interface) without a plugin. In this paper, we have developed a real-time web peer-to-peer communication system that allows users to connect to high-speed data transfers on a communication channel using WebRTC technology, HTML5, and use a Node server address. The result shows that the system is stable, fully functional, secure and can be used on an active network to transfer and receive multimedia data in real-time between users.

TABLE OF CONTENTS

<u>CHAPTER</u>	<u>PAGE NO.</u>
Candidate's Declaration	2
Acknowledgement	3
Abstract	4
Chapter 1-- Introduction	6
1.1 Node.js	7 to 8
1.2 How webrtc works?	9 to 14
Chapter 2-- Establish Video and Audio Communication Using webrtc	15
2.1 Front End (call.html, style.css)	15 to 16
2.2 Back End (call.js)	17 to 18
2.3 Make a Video and Audio communication	19 to 21
Chapter 3-- Make a Video Call App Using Android	22
3.1 Android Studio	22
3.2 Firebase	22
3.3 Create Firebase and Android Project	23 to 24
3.4 Register Android app to Firebase Database	25 to 30
3.5 Create Login Page (MainActivity)	31 to 35
3.6 Create Intent (CallActivity)	36 to 43
Chapter 4-- Conclusion	44
Bibliography	45
Plagiarism Report 30% only	46 to 48

1. INTRODUCTION

WebRTC means web real-time communications. It is a very exciting, powerful, and highly disruptive cutting-edge technology and standard. WebRTC leverages a set of plugin-free APIs that can be used in both mobile browsers and desktop and is progressively becoming supported by all major modern browser vendors. Previously, external plugins were required in order to achieve similar functionality as is offered by WebRTC.

WebRTC is a collection of standards, protocols, and JavaScript APIs, the combination of which enables peer-to-peer audio, video, and data sharing between browsers (peers).

WebRTC JavaScript API

WebRTC is a complicated topic where many technologies are involved. In WebRTC with the help of JS API, we can establish the connection, communicate, and transmit the data through JS APIs. The major APIs are:

RTCPeerConnection – creates and navigates peer-to-peer connections,

RTCSessionDescription – describes one end of a connection (or a potential connection) and how it's configured.

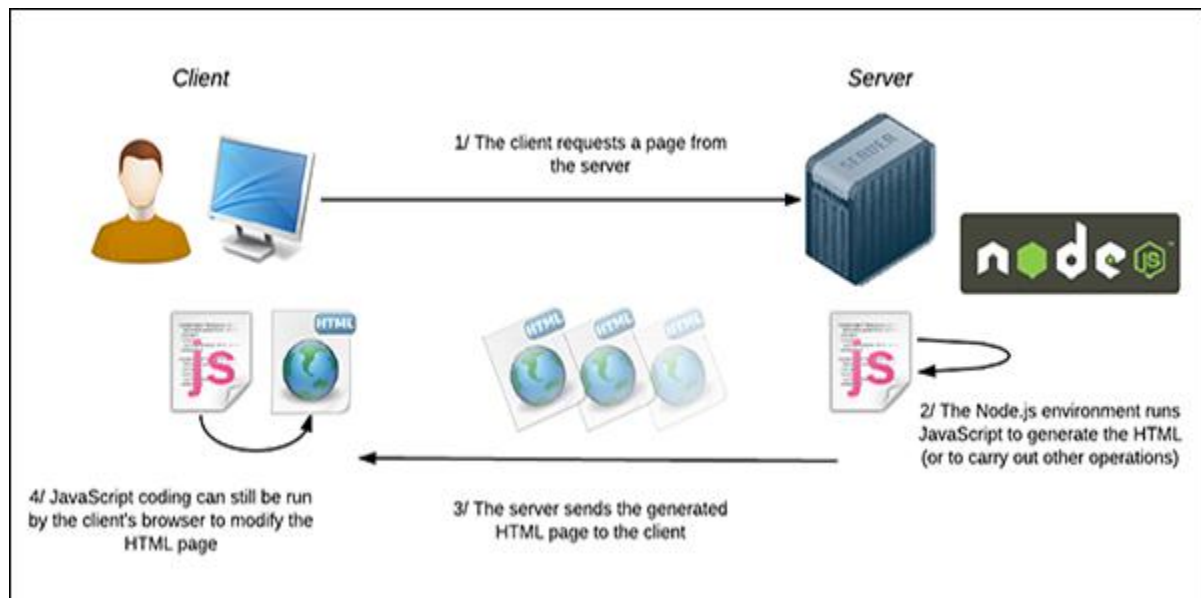
navigator.getUserMedia – captures audio and video.

1.1 Node.js

Node.js is an open source, cross-platform run-time environment for developing server-side and networking applications. Node.js applications are written in JavaScript and can be run within the Node.js run-time on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.



Features of Node.js

- 1.Asynchronous and Event-Driven
- 2.Very Fast
- 3.Single Threaded but Highly Scalable
- 4.No Buffering
- 5.License

Reasons for using Node JS

When you use Node JS, you will see that it comes with several features which have helped it to become a top choice for developers. The reasons are mentioned below.

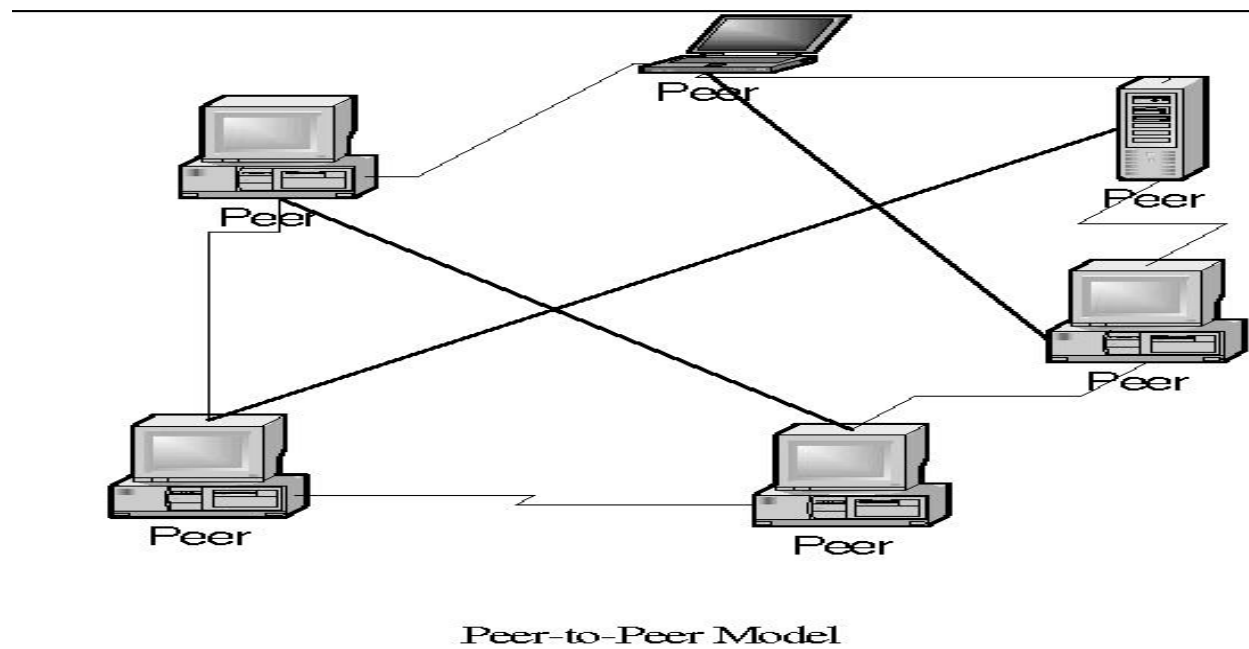
- 1.It helps in sharing with the Node Package Manager or NPM, and it is also fast as well.
- 2.This also helps in a good fit for real-time applications.
- 3.You can get the best data streaming in Node JS.
- 4.Node JS provides you with the chance to write the JavaScript server-side and also client-side coding.

1.2.How Dose WebRTC Works ?

Peer to Peer Communication

Stands for "Peer to Peer." In a P2P network, the "peers" are computer systems that are connected to each other via the Internet. Files can be shared directly between systems on the network without the need for a central server. In other words, each computer on a P2P network becomes a file server as well as a client.

The only requirements for a computer to join a peer-to-peer network are an Internet connection and P2P software



Signaling

Signaling allows two endpoints (sender, receiver, or both) to exchange metadata in order to limit the exchange of data to establish a call. This reply and reply message (also called reply message) contain important information about the number of messages that can be received, and the impact is a problem. The session abuse protocol is usually formulated with the Session Description Protocol (SDP), which is a standard format used by many real-world systems (including VoIP and WebRTC).

Network Address Translation Traversal - ICE, TURN and STUN

After the original streaming signaling connection is established for the used space, the two endpoints must initiate a NAT (Network Address Translation) connection. - Temporary video communication. NAT traversal is a method to solve the problems related to IP address translation.

In a video call with WebRTC support, if both ends are not in the same LAN, there will be one or more intermediate network devices (routers/gateways) between them. obstacle:

ICE

ICE stands for "interactive connection establishment". This is a general NAT traversal method used in WebRTC and is described in IETF RFC 5245. ICE provides a way to connect media through NAT through a connectivity test environment. There are candidates (neighbor IP address, reflection address-STUN, and forwarding address-TURN). All accumulated addresses are sent to the remote host via SDP. When the WebRTC client has all the ICE addresses collected by it and its partners, it will start to develop by starting the connection test. In most cases, these tests will attempt to successfully send media to other addresses. The disadvantage of using ICE is that it takes time, which may take 10 seconds. This new mechanism has been replaced by WebRTC (called Trickle ICE).

STUN

STUN (session traversal utility for NAT), using UDP to improve ICE on NAT. Using STUN, programs can discover the existence and form of NAT and firewalls between each other and usually in the public Internet. Any tool can use it to determine the IP and port assigned to it through NAT. Usually, the STUN client can send a message to the STUN server to obtain the public IP address and port data. Then, the STUN server calls this data. Participate in peer-to-peer communication on the Internet.

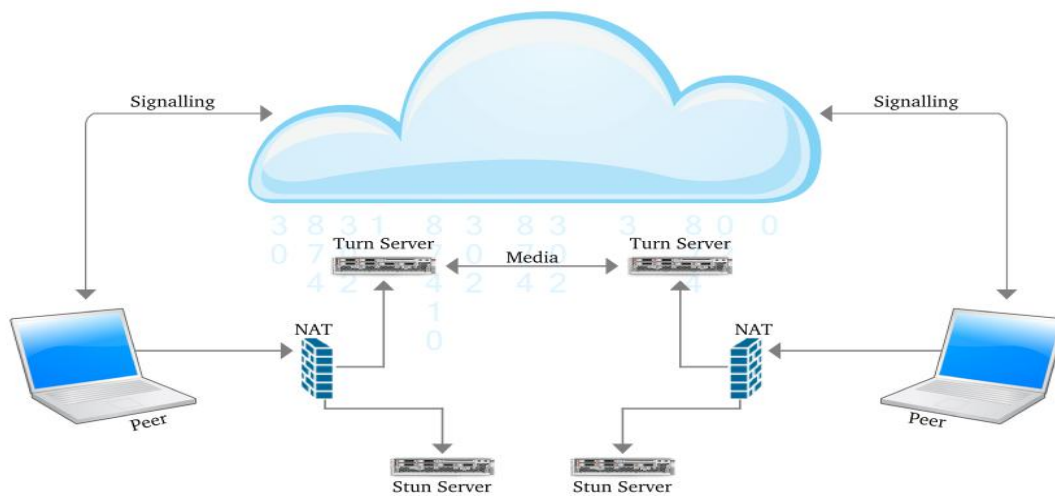


Fig:Communication using STUN server

TURN

TURN (traversal using relay and NAT) is a protocol that supports the use of translators (NAT) or firewalls for WebRTC applications to interact with the community. Using the TURN server, the client can send and receive data through the intermediate server. Is the STUN extension. In some cases, the client voice endpoint is locked in an unusual form of NAT, or when symmetric NAT is used, the difficulty of sending media through a relay server called a TURN server may be reduced.

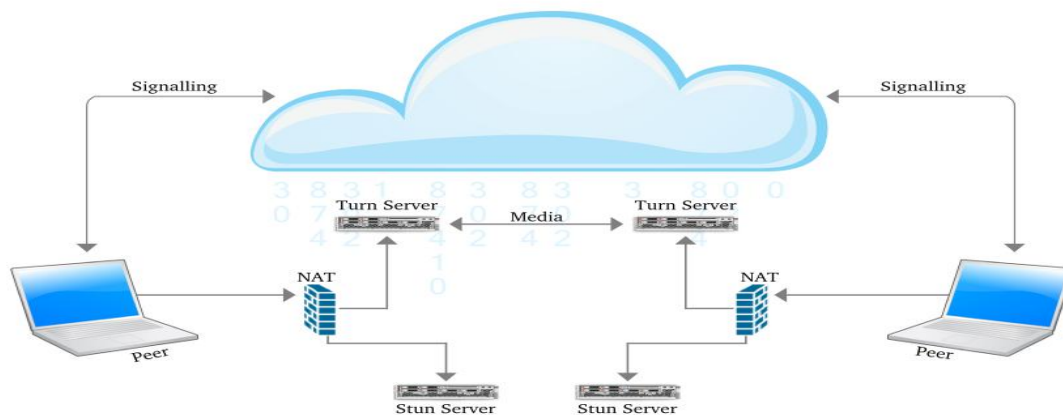


Fig:Communication using TURN server

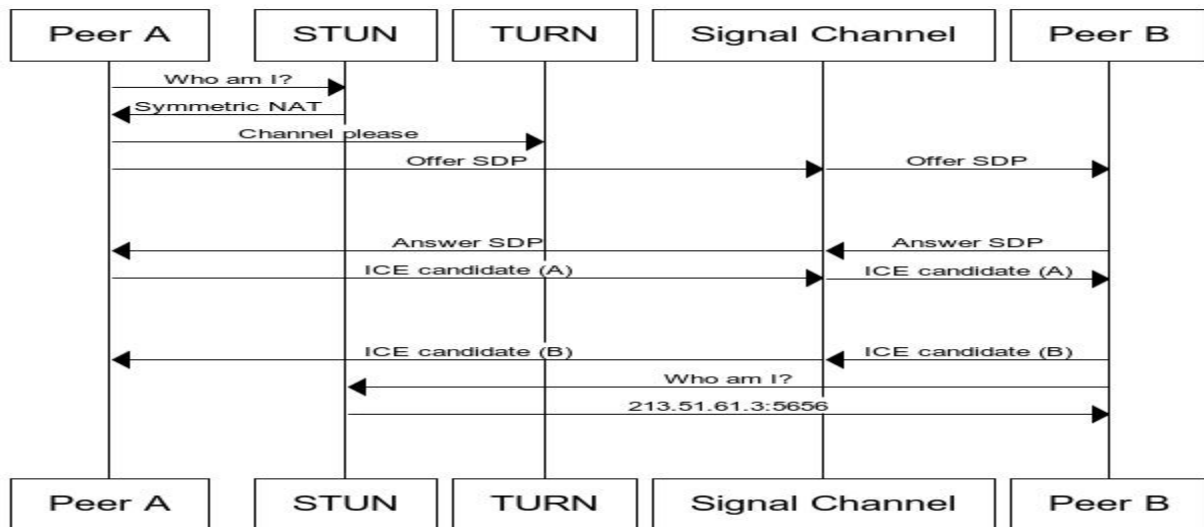


Fig: How webRTC work

WebRTC stands for Web real-time communication and in webrtc there are no servers involved to transfer real-time data from one client to another. The webrtc connections are peer to peer because of which sending data through the webrtc is very simple and all of this is done through webrtc APIs. Which are available for almost all the platforms.

When I said that there are no servers required for webrtc to work I was not completely right about it. There are servers required initially for the client computers to get connected but once the connection established then these clients can communicate with each other without the need to have a server in the middle the transferred data is a peer to peer once they connected.

On the surface what happens is, client1 computer will send some data about itself t the signaling server, and then the signaling server will send that data to the other client and then the client2 will store this data.

After that this client2 will send data about itself to the server and then the server will send that data to client1 now both of these client computers know about each other so the connection will get established between them, now they can send their data without the need of having the server.

What is the data these clients are sending because of which the RTC connection is happening?

And signaling server is not anything fancy it is just a normal server and the only purpose of this server is to get the data of the clients exchanged.

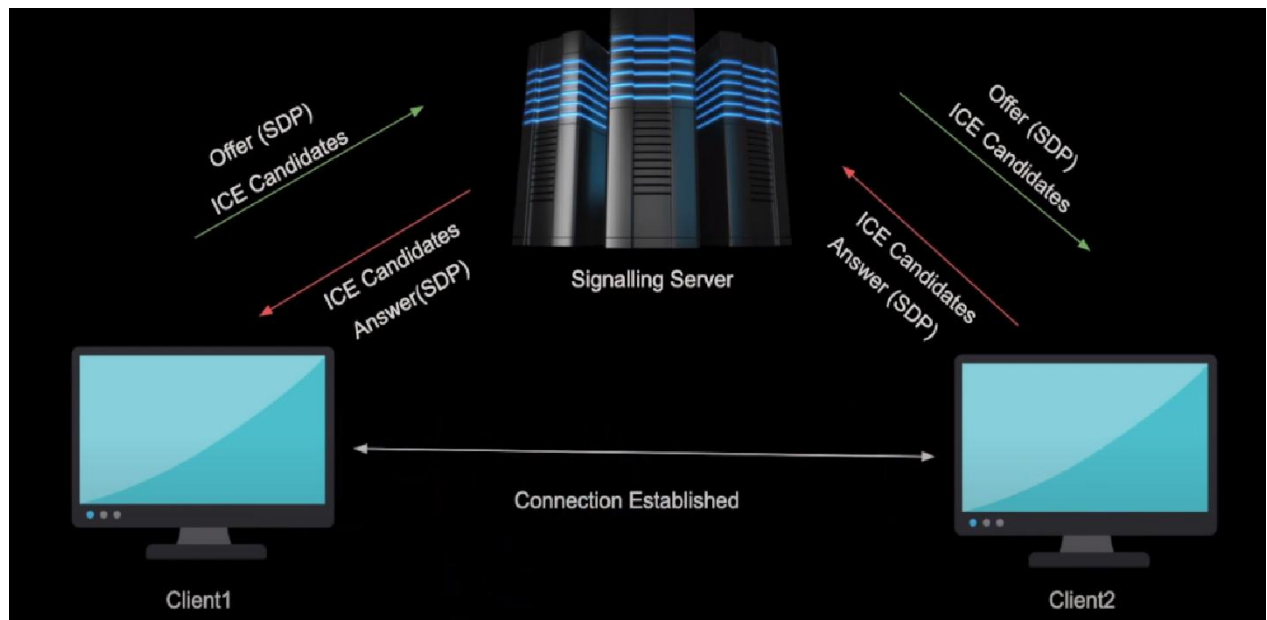
We are calling the signaling server because the process of sending the data from one client to another in a webrtc is called signaling.

The first step of a successful webrtc connection is the creation of offers. So if client1 wants to connect with client2 then it will create (SDP) we can imagine this offer to be an object that may be a javascript object. This offer called SDP, SDP stands for session description protocol and the SDP is basically the media configuration of the client. Media configuration like client1 wants to send audio, video, or both. So SDP is nothing but the media configuration of these clients.

Now, client1 creates the offer and it will send the SDP to the signaling server, and then the signaling server sends that SDP to client2 and client2 will store the SDP of client1 on its local memory.

Once the client2 has stored the SDP of client1 it will have to create an answer in response to that offer which is the media configuration of this client (client2) and then sends the SDP to the signaling server and then the signaling server will send that answer which contains media configuration (SDP) to client1.

Now both clients know about the media configuration of each other but they still do not know how to connect each other and for that, they will have to transfer their network information and that both clients will transfer the data called ICE (Interactive connectivity establishment) candidates start coming from webrtc API and ICE candidates are generated from STUN or TURN servers.



Advantages of WebRTC

- 1.It's free
- 2.Platform and device independence
- 3.Secure voice and video
- 4.Advanced voice and video quality
- 5.Reliable session establishment
- 6.Multiple media streams
- 7.Adaptive to network conditions
- 8.Rapid application development

2.Establish browser to browser video and audio communication using WebRTC.

2.1.Front end

Call.html

```
<!DOCTYPE html>
<html>
  <head>
    <link href="./style.css" rel="stylesheet" />
  </head>
  <body>

    <script src="./peerjs.js"></script>

    <video class="secondary-video" autoplay id="remote-video"></video>
    <video class="primary-video" autoplay muted id="local-video"></video>

    <script src="./call.js"></script>

  </body>
</html>
```

Style.css

```
html, body {  
    padding: 0;  
    margin: 0;  
}  
  
.primary-video {  
    position: absolute;  
    width: 100%;  
    height: 100%;  
    object-fit: cover;  
}  
  
.secondary-video {  
    position: absolute;  
    max-width: 30%;  
    width: 30%;  
    margin: 16px;  
    border-radius: 16px;  
}
```

Primary-video: The video which will take the entire screen is primary-video.

Secondary-video: The video which will be a picture in the picture is secondary-video.

2.2.Back End

Call.js

```
let localVideo = document.getElementById("local-video")
let remoteVideo = document.getElementById("remote-video")

localVideo.style.opacity = 0
remoteVideo.style.opacity = 0

localVideo.onplaying = () => { localVideo.style.opacity = 1 }
remoteVideo.onplaying = () => { remoteVideo.style.opacity = 1 }

let peer
function init(userId) {
  peer = new Peer(userId, {
    host: '192.168.43.237',
    port: 9000,
    path: '/videocall'
  })

  listen()
}

let localStream
function listen() {
  peer.on('call', (call) => {

    navigator.getUserMedia({
      audio: true,
      video: true
    }, (stream) => {
      localVideo.srcObject = stream
      localStream = stream

      call.answer(stream)
      call.on('stream', (remoteStream) => {
        remoteVideo.srcObject = remoteStream

        remoteVideo.className = "primary-video"
        localVideo.className = "secondary-video"
      })
    })
  })
}
```

```

function startCall(otherUserId) {
  navigator.getUserMedia({
    audio: true,
    video: true
  }, (stream) => {

    localVideo.srcObject = stream
    localStream = stream

    const call = peer.call(otherUserId, stream)
    call.on('stream', (remoteStream) => {
      remoteVideo.srcObject = remoteStream

      remoteVideo.className = "primary-video"
      localVideo.className = "secondary-video"
    })
  })
}

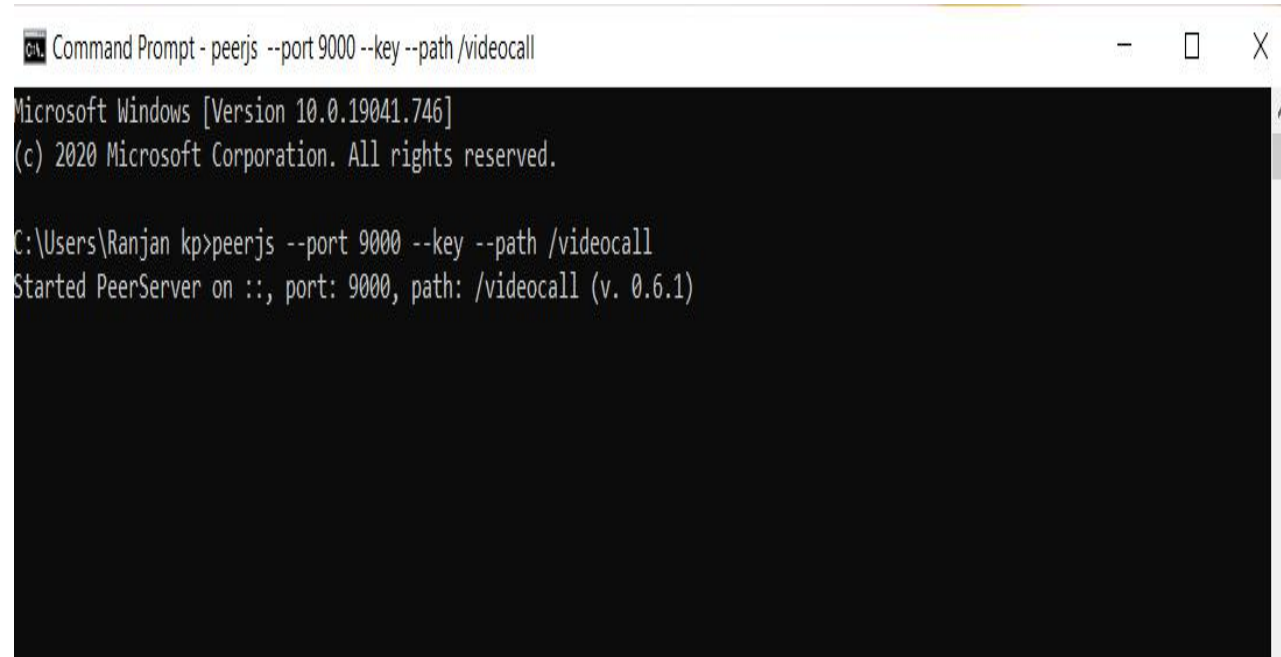
function toggleVideo(b) {
  if (b == "true") {
    localStream.getVideoTracks()[0].enabled = true
  } else {
    localStream.getVideoTracks()[0].enabled = false
  }
}

function toggleAudio(b) {
  if (b == "true") {
    localStream.getAudioTracks()[0].enabled = true
  } else {
    localStream.getAudioTracks()[0].enabled = false
  }
}

```

2.3 Make a Video and Audio Communication

First connected the peerjs server

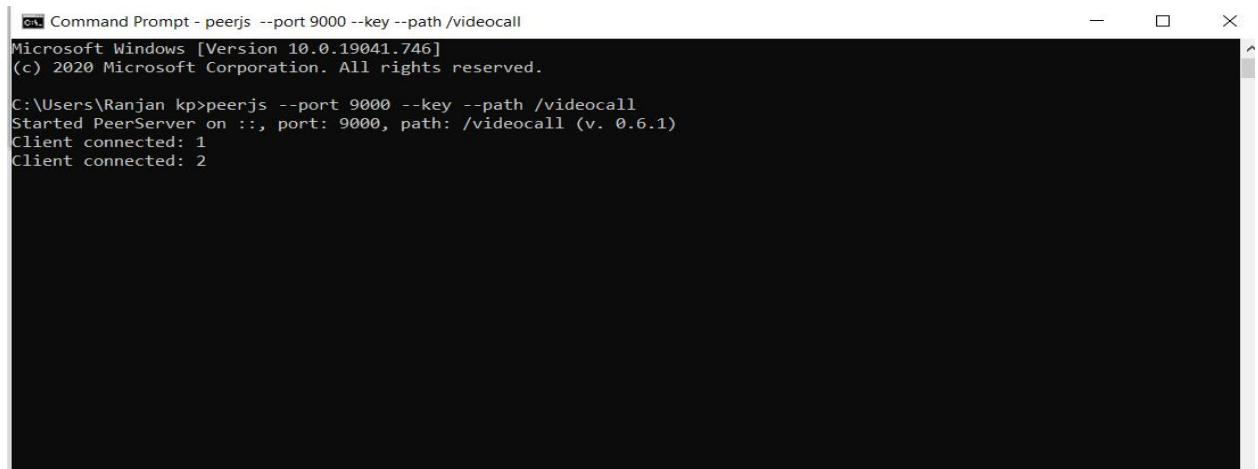
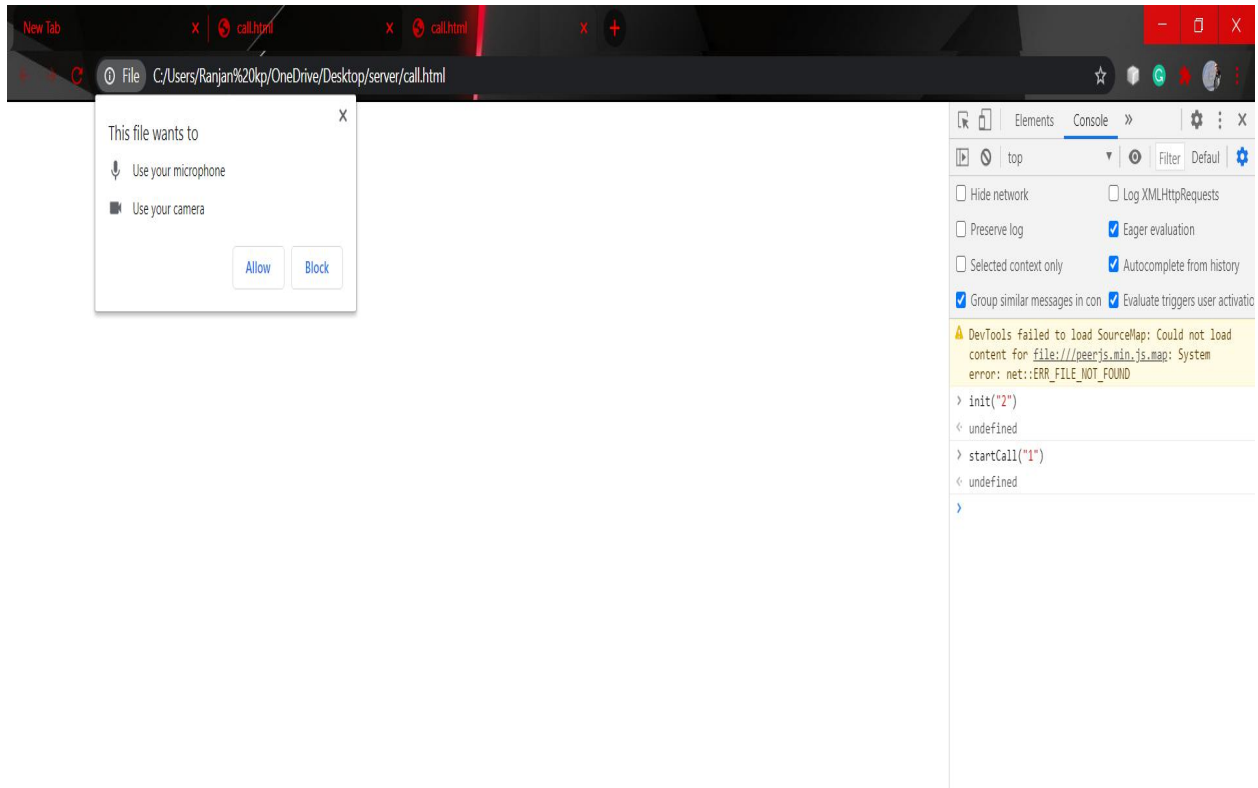


```
Command Prompt - peerjs --port 9000 --key --path /videocall
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

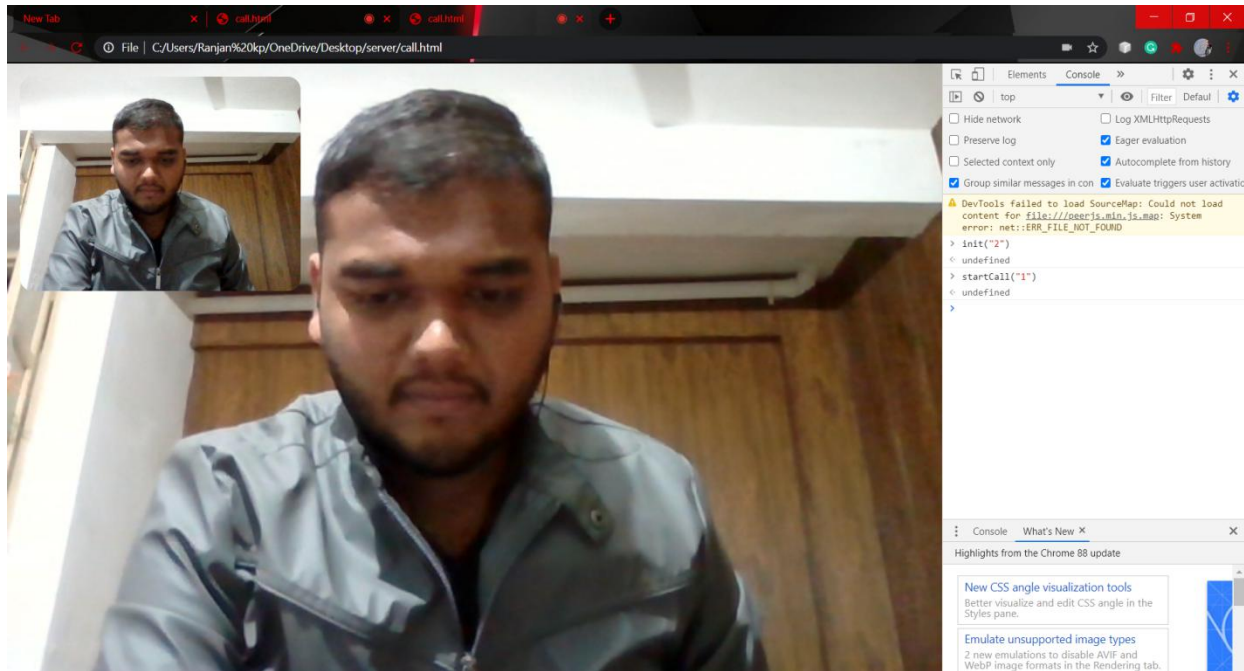
C:\Users\Ranjan kp>peerjs --port 9000 --key --path /videocall
Started PeerServer on ::, port: 9000, path: /videocall (v. 0.6.1)
```

After peerjs was connected then run call.html file two separate tabs and go to the console and initialize our peer with both tabs.

Once initialize peer then going to make a call and once make a call the tab will ask the permissions of the camera or microphone.



Result video streams



3. Make a Video Call App Using Android

3.1. Android Studio

Android STUDIO Authority has coordinated Advancement climate (IDE) for Mail Android Arrisation Development Android. Android Studio provides a wide range of functions to use factory devices for Android phones, tablets, Android Wear, Android TV, and Android. In addition, pre-packaged programs for testing pre-packaged programs and pre-packaged programs may be included. The studio will return to Kotlin and even has a branch to convert Javabashed to Kotlin.



3.2. Firebase

Firebase is a Backend-as-a-Service (Baas).

It gives engineers an assortment of instruments and administrations to assist them with creating quality applications, develop their client base, and acquire benefits.

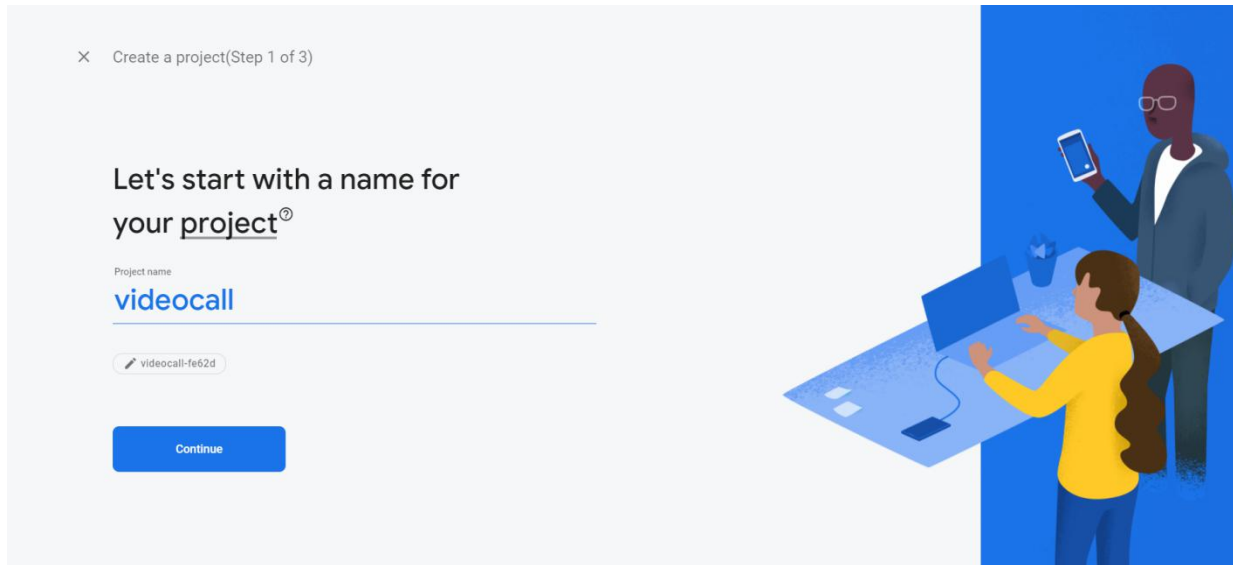
It is based on Google's framework. Firebase is sorted as a NoSQL data set program, which stores information in JSON-like records.

Firebase helps you assemble and run effective applications. Upheld by Google, adored by engineers. Speed up application advancement with a completely oversaw backend framework.

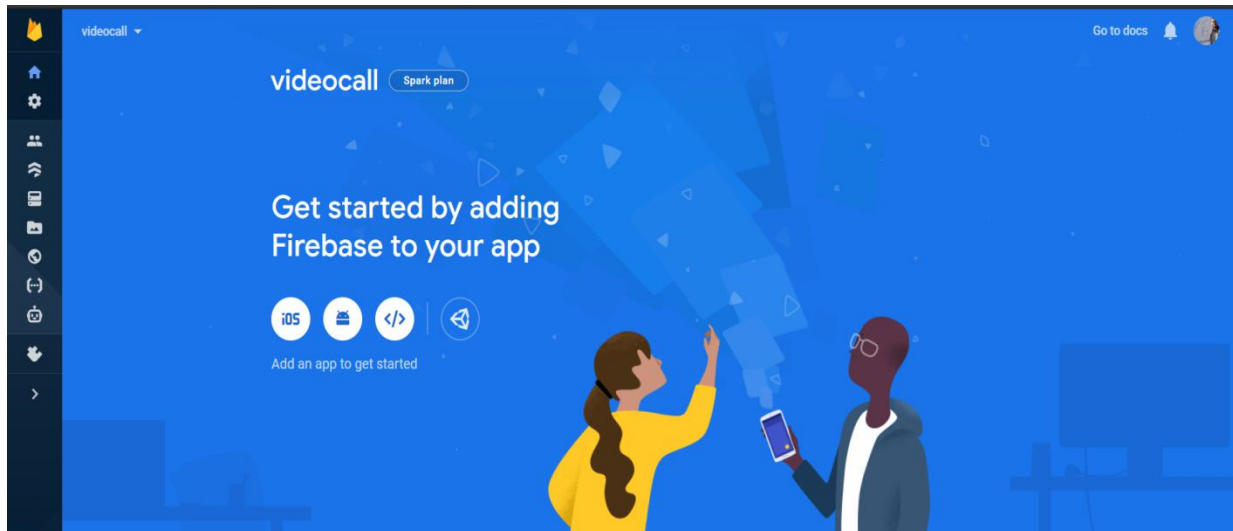
The Firebase Realtime Database is a cloud-facilitated NoSQL database that allows you to store and adjust information between your clients in realtime.

3.3.Creating a FireBase Project

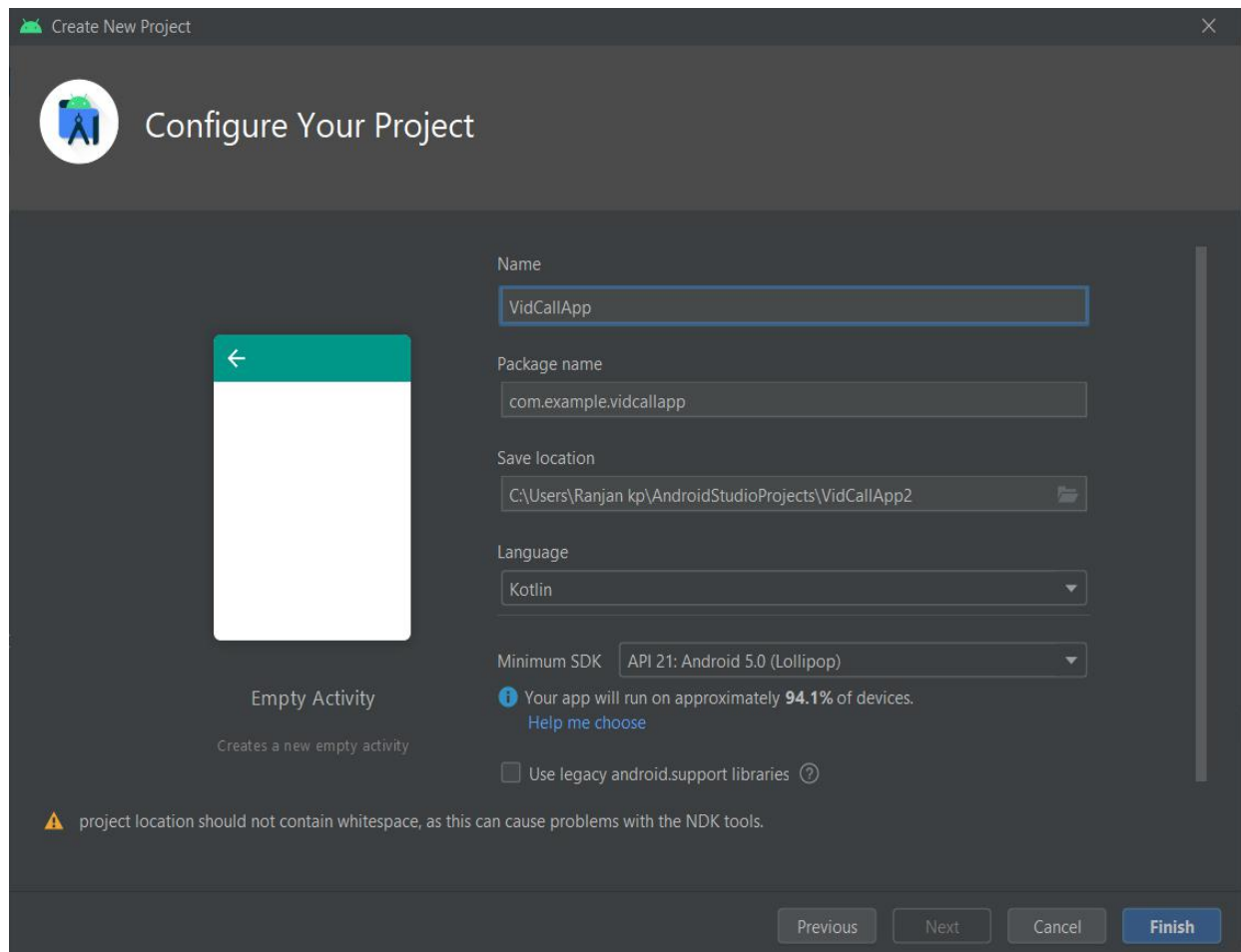
Inside the application, we are going to use a real-time firebase database to transfer some of the data from one client to another.



Firestore Database Was Created

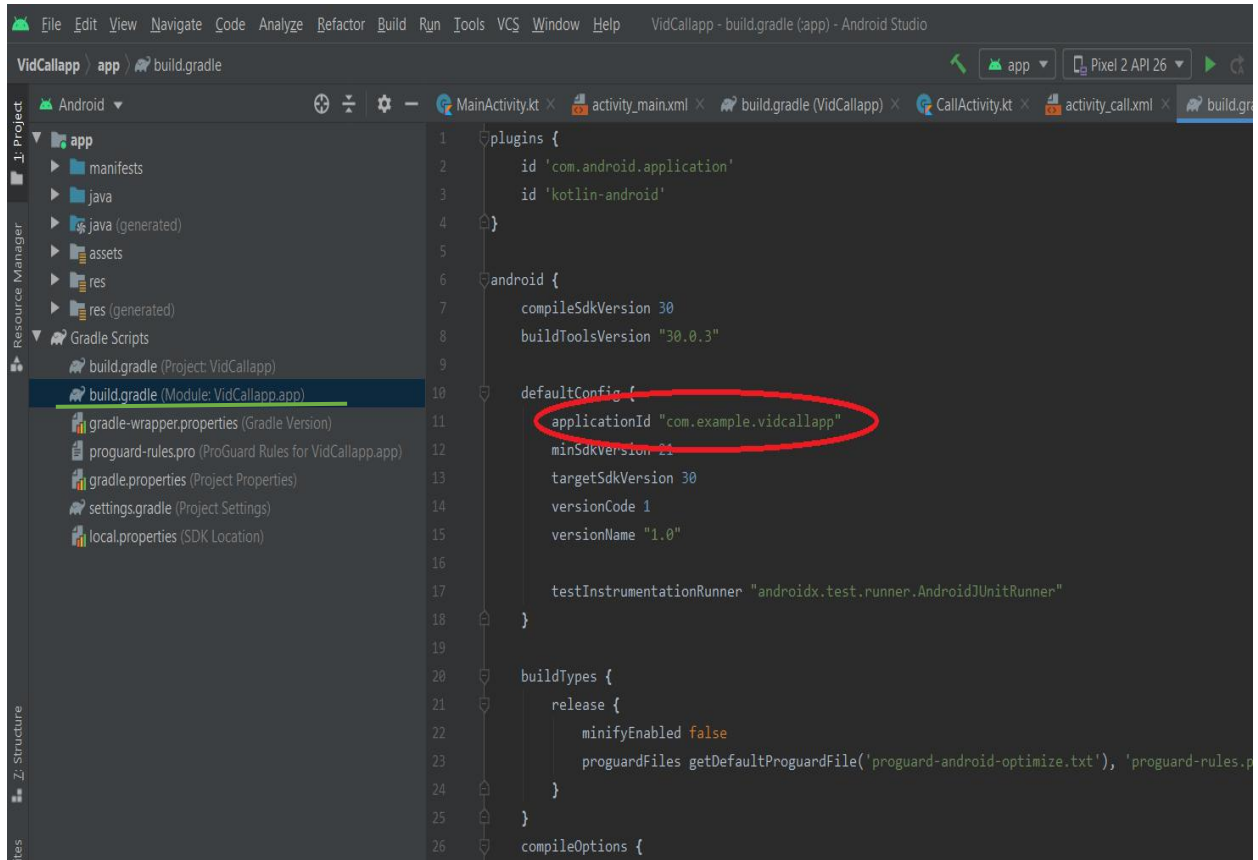


3.3 Create android studio project



3.4.Register android application to firebase database

Once the android project loads up go to the build.gradle (module: VidCallapp.app) and copy the application Id.



Write the application ID in the Android package name and click the register app.

[illegible]

After that download the google-services.json file and click next.

1

Register app

Android package name: com.example.abs

2

Download config file

Download google-services.json

Switch to the **Project view** in Android Studio to see your project root directory.

Move the google-services.json file that you just downloaded into your Android app module root directory.

google-services.json

Next

3

Add Firebase SDK

4

Next steps

Instructions for Android Studio below | [Unity C++](#)

Project Packages Scratch

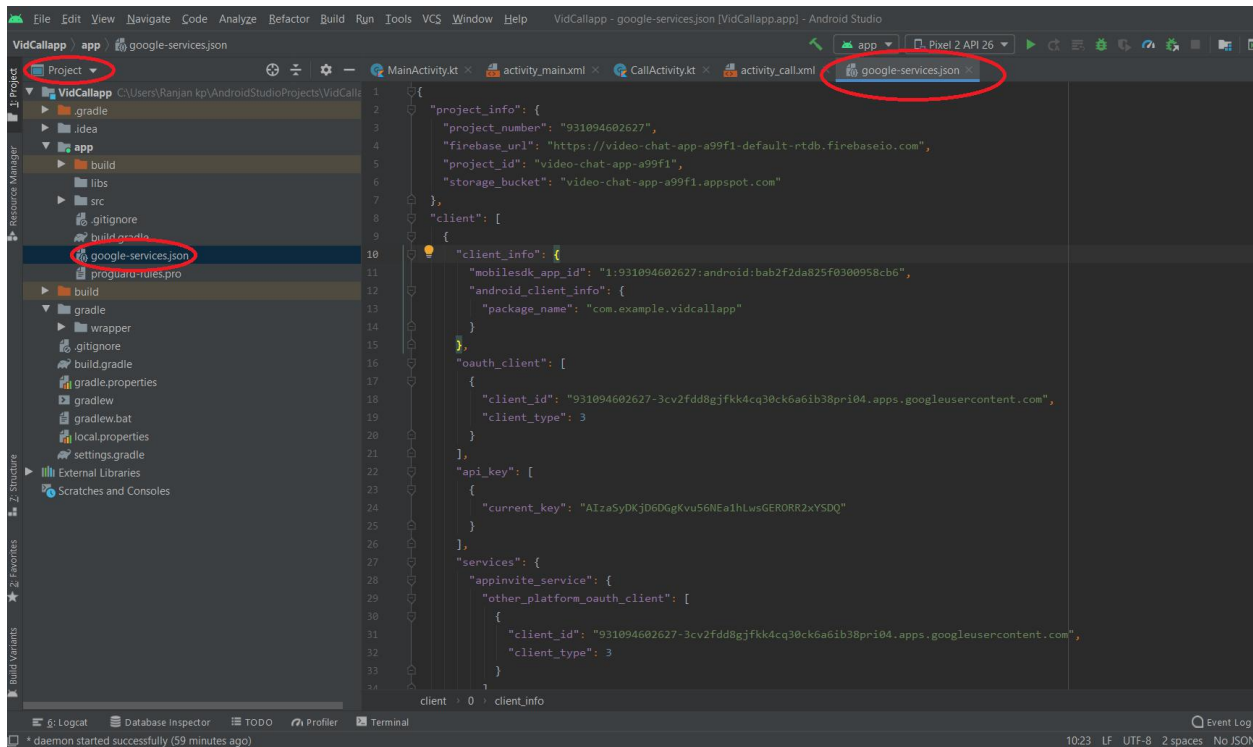
MyApplication (~\Desktop\MyApplication)

- .gradle
- idea
- app
 - build
 - libs
 - src
 - .gitignore
 - app.iml
 - build.gradle
 - google-services.json
 - proguard-rules.pro
- gradle

Illustration of a smartphone and a laptop with a download icon on the screen.

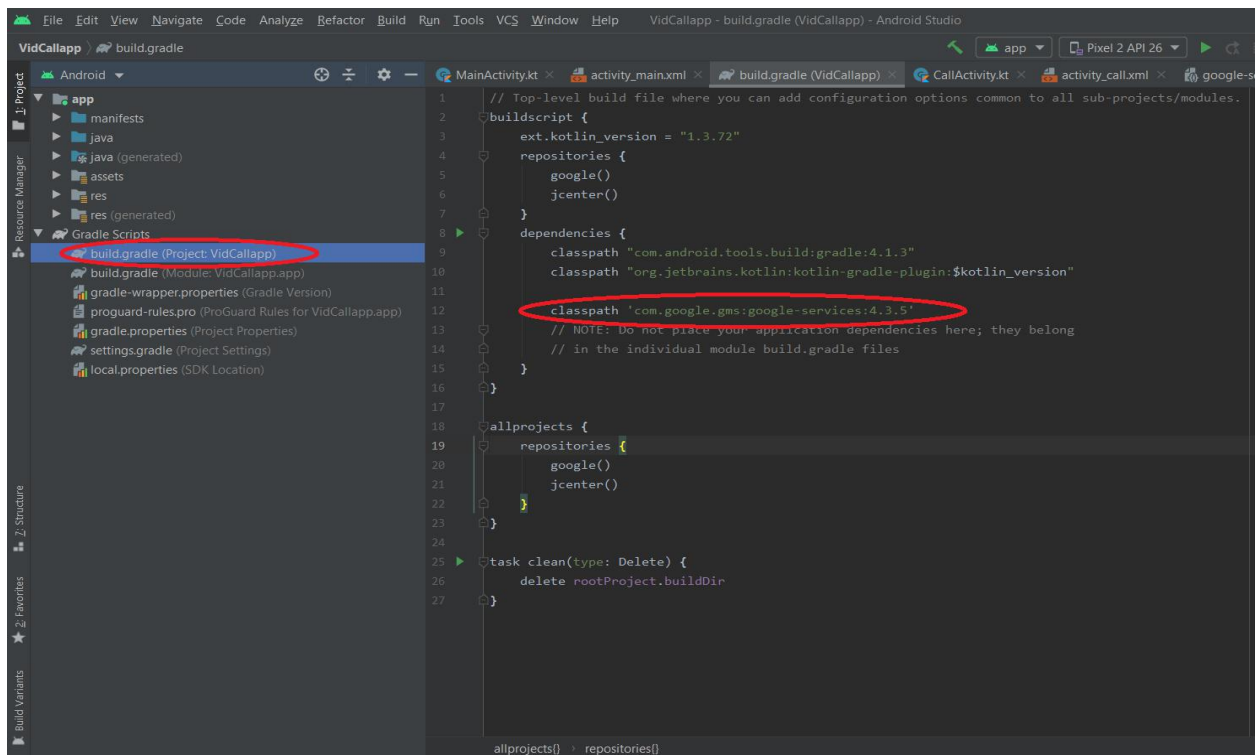
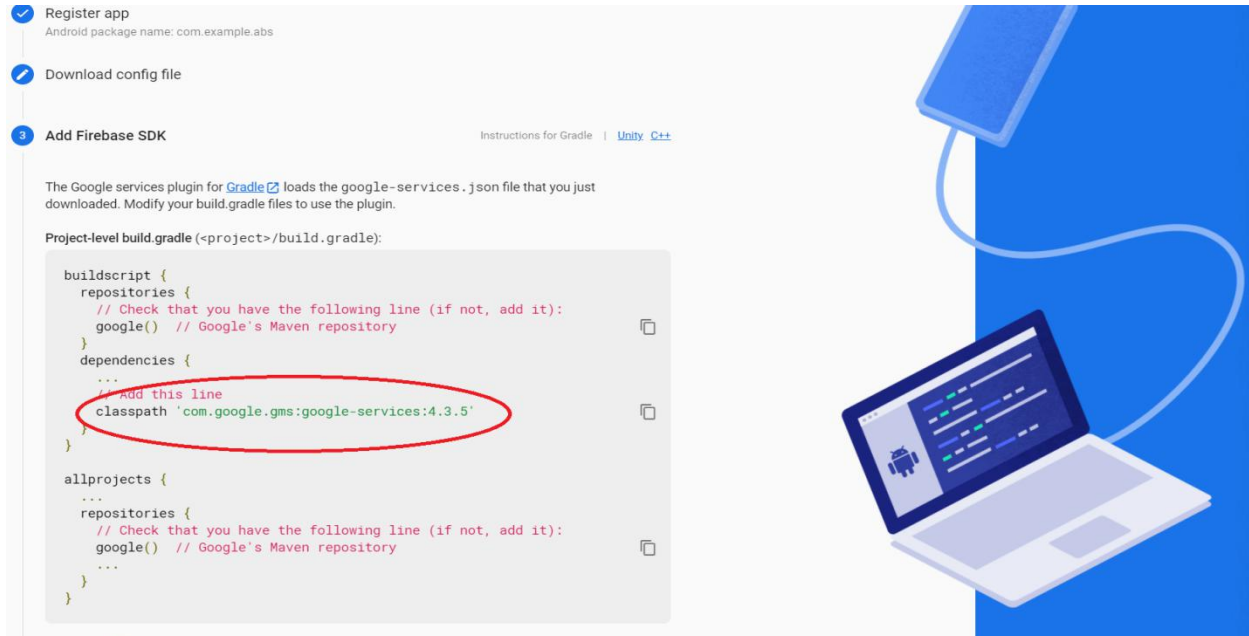
Copy the google-services.json file in android studio.

In android studio change the android to project level and in the project directory paste the google-services.json file.



Add Firebase SDK in the android project(android studio)

1.Copy the classpath and go to the android studio and paste the classpath in build.gradle(project:VidCallapp)



2. Copy the plugin and go to the android paste the plugin in build.gradle(module: VidCallapp.app) and add the dependency of firebase real-time database once you have added click sync no

App-level build.gradle (<project>/<app-module>/build.gradle):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:26.7.0')

    // Add the dependency for the Firebase SDK for Google Analytics
    // When using the BoM, don't specify versions in Firebase dependencies
    implementation 'com.google.firebase:firebase-analytics-ktx'

    // Add the dependencies for any other desired Firebase products
    // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. [Learn more](#)

Finally, press 'Sync now' in the bar that appears in the IDE:

Gradle files have changed since last sync. [Sync now](#)

Previous [Next](#)

Next steps

build.gradle (Module: VidCallapp.app)

```
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
}

compileOptions {
    sourceCompatibility kotlin_version
    targetCompatibility kotlin_version
}

kotlinOptions {
    jvmTarget = '1.8'
}

dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version'
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.firebase:firebase-database-ktx:19.7.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}

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

Permissions Required

Go to the manifest and declare all the permissions that we going to require.

Run time permissions : This means the permissions are requested during the run time when the app was running.

Syntax : `<uses-permission android:name="android.PERMISSION NAME" />`

Camera permission:

```
<uses-permission android:name="android.permission.CAMERA" />
```

Record audio permission:

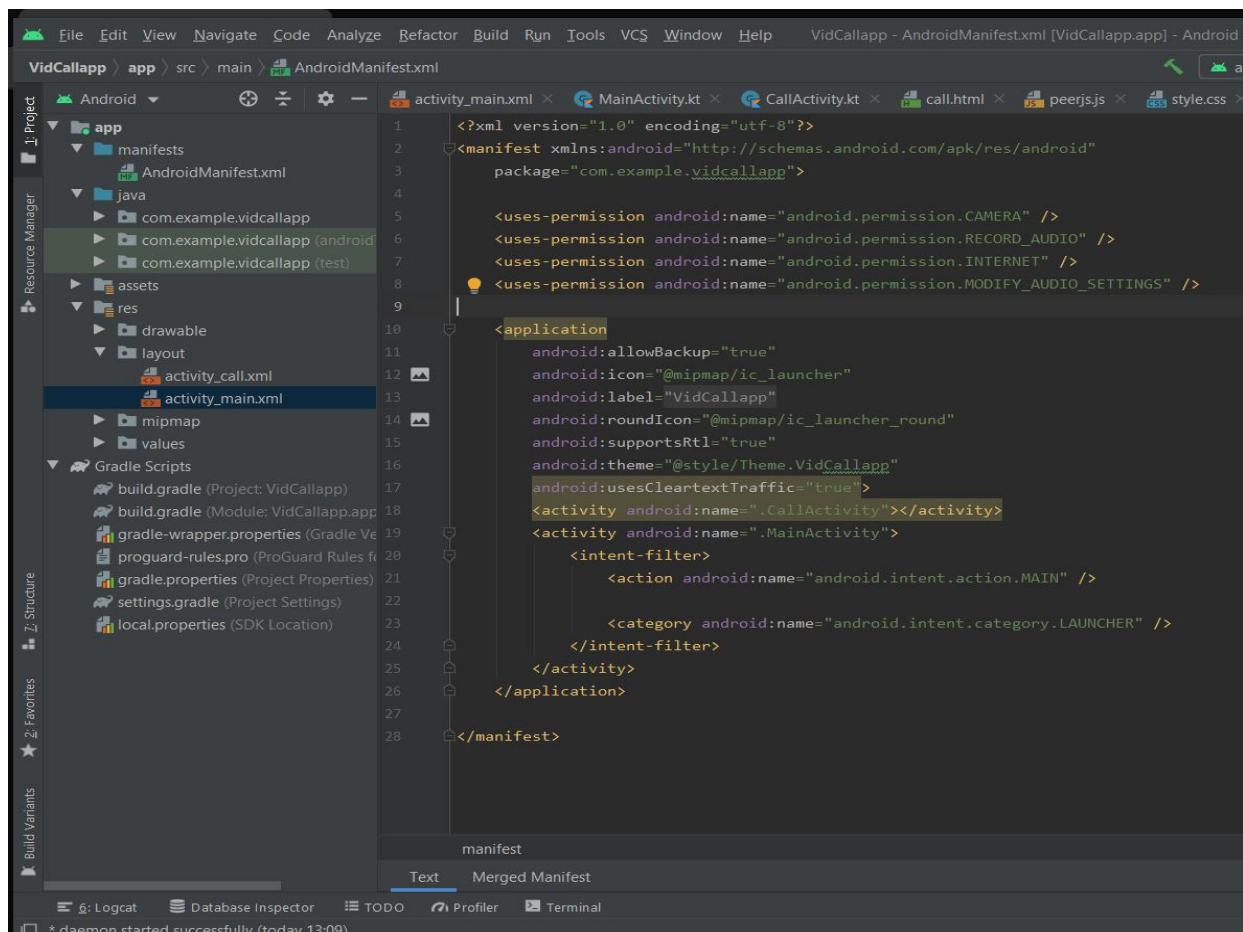
```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

Internet Permission :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Modify audio settings :

```
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
```



3.5.Create a Login Page

Front-End

LinearLayout: LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.

EditText: EditText is a user interface control that is used to allow the user to enter or modify the text. While using EditText control in our android applications, we need to specify the type of data the text field can accept using the inputType attribute.

Button: A Button is a Push-button that can be pressed, or clicked, by the user to perform an action.

Activity_main.xml

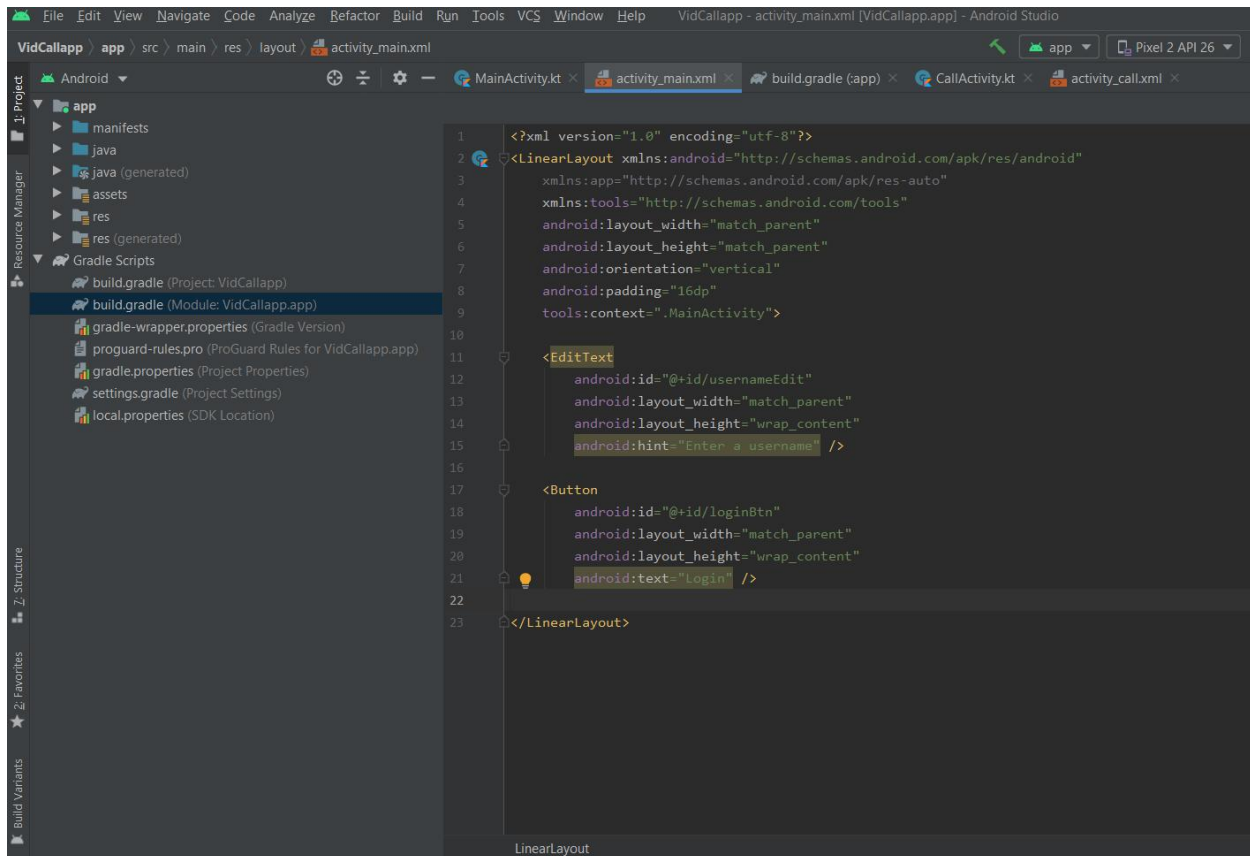
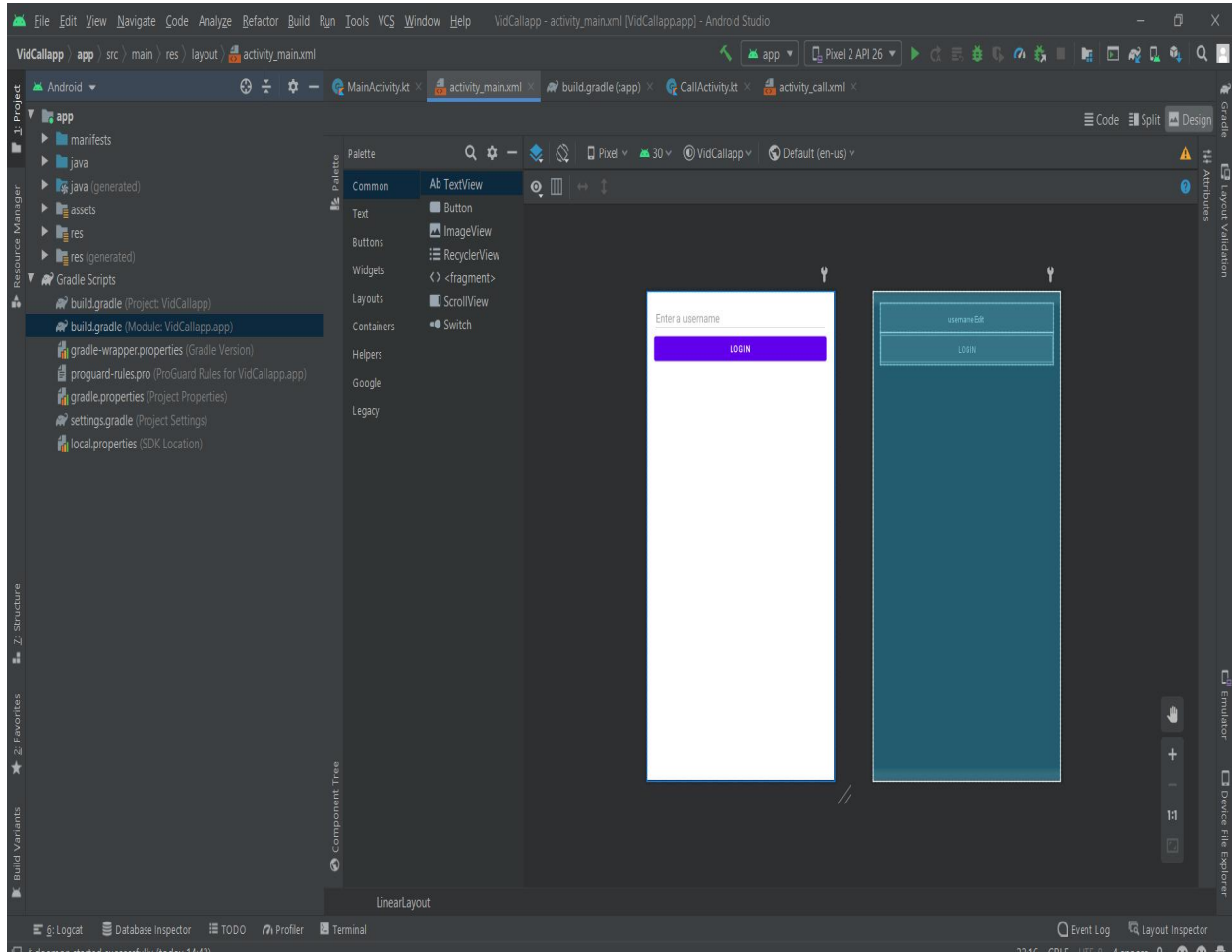


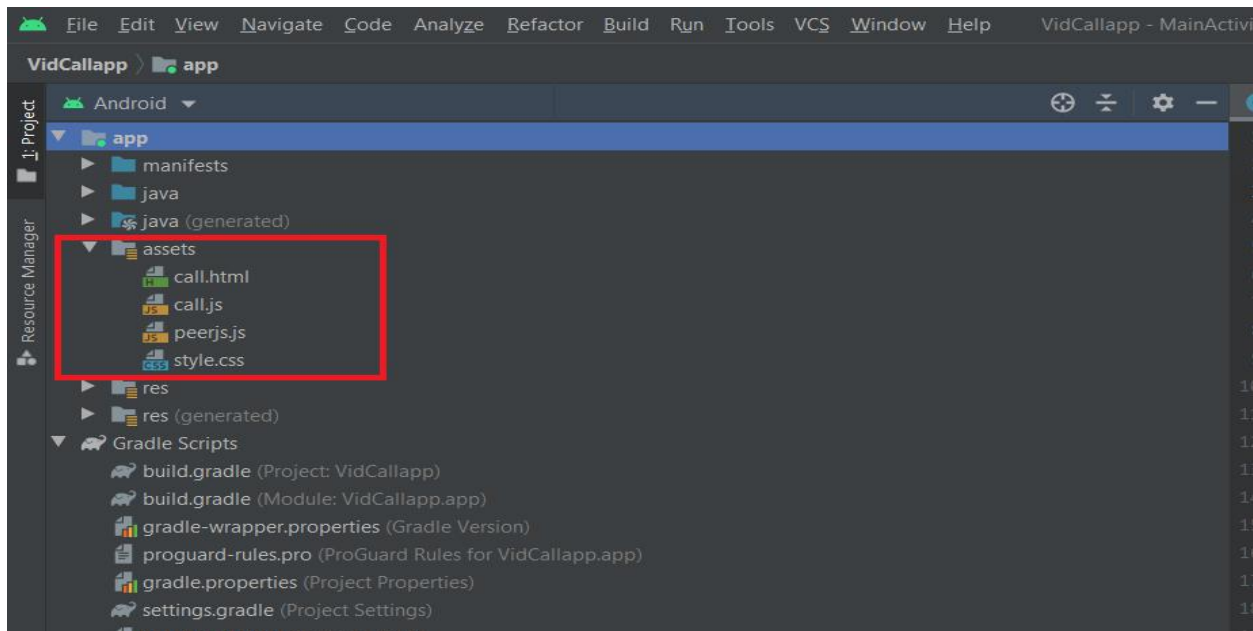
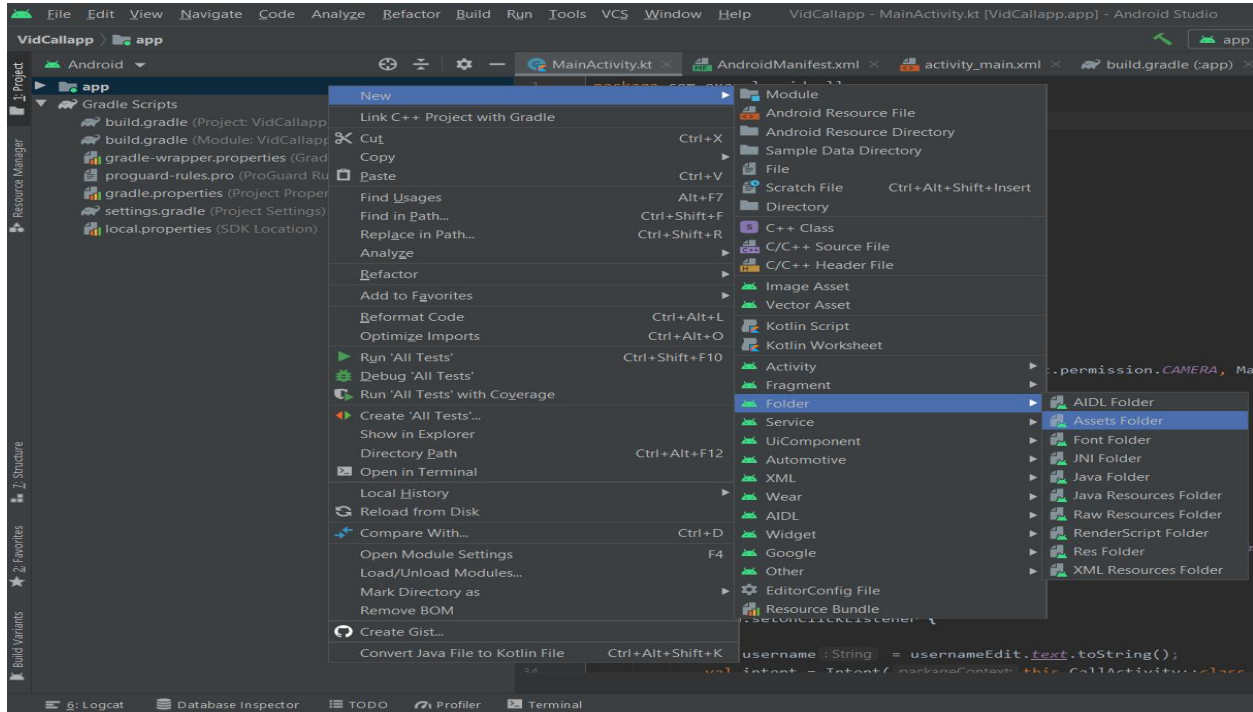
Fig 1. screenshot of activity_main.xml

Design of Login page



Back-End of Login Page

First we create the assets folder and add the all assets there like call.html, call.js, style.css, peerjs.js.



1. Create variable named permissions like camera and record audio.

```
val permission= arrayOf(Manifest.permission.CAMERA,  
Manifest.permission.RECORD_AUDIO)
```

2. Inside the onCreate function if permission granted if it is not granted then ask the users for permissions.

```
if(!isPermissionGranted()){  
    askPermissions()  
}
```

And create the function.

```
private fun askPermissions() {  
    ActivityCompat.requestPermissions(this,permission, requestcode)  
}  
  
private fun isPermissionGranted(): Boolean {  
  
    permission.forEach {  
        if (ActivityCompat.checkSelfPermission(this, it) !=  
PackageManager.PERMISSION_GRANTED)  
            return false  
    }  
    return true  
}
```

3. Initialize the Firebase app and then login button they have in XML and we will set this thing in an onclicklistener and when this is clicked then we will get the username from usernameEdit which is also available in the XML

```
Firebase.initialize(this)  
loginBtn.setOnClickListener {  
  
    val username = usernameEdit.text.toString();
```

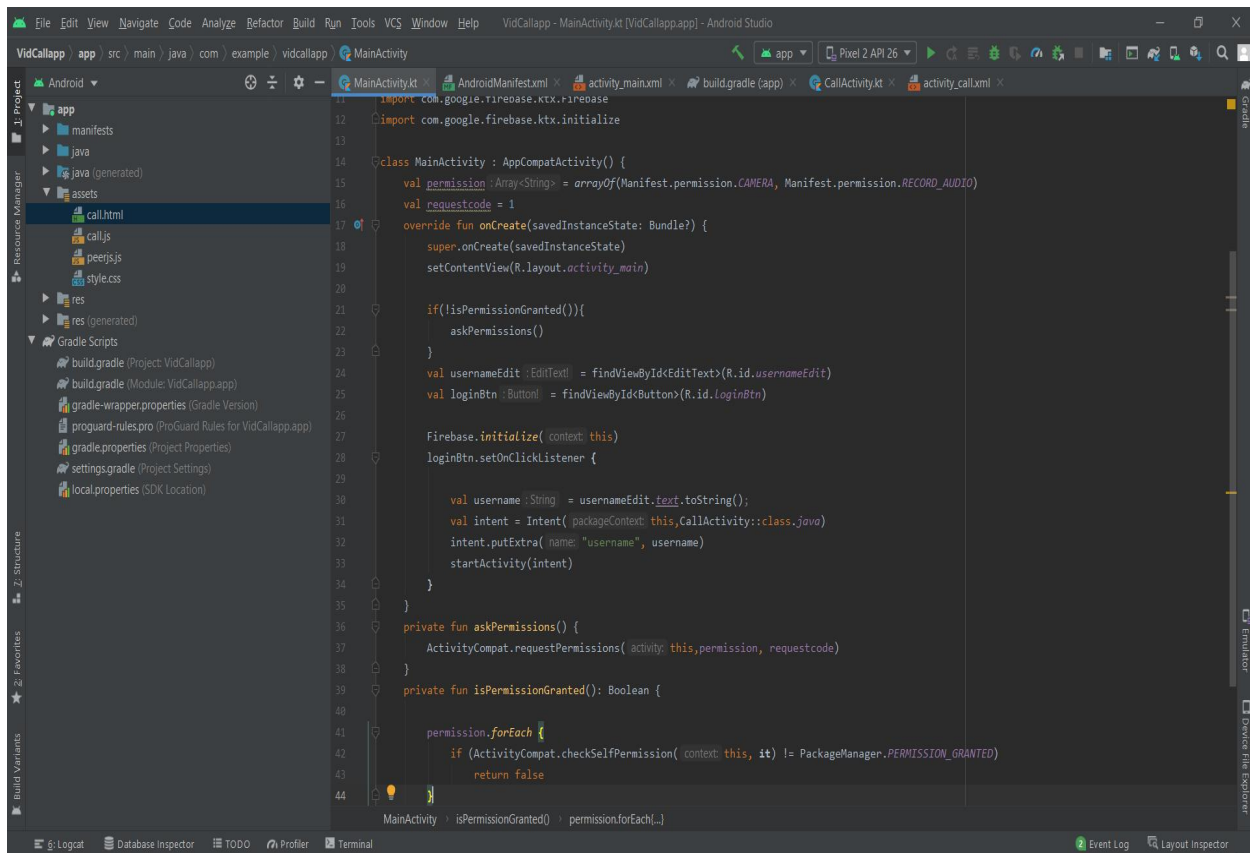
4. Create a intent activity (CallActivity)

```
val intent = Intent(this, CallActivity::class.java)
```

```
intent.putExtra("username", username)
```

```
startActivity(intent)
```

MainActivity.kt



3.6.Create Intent (CallActivity)

Activity_call.xml (Front End)

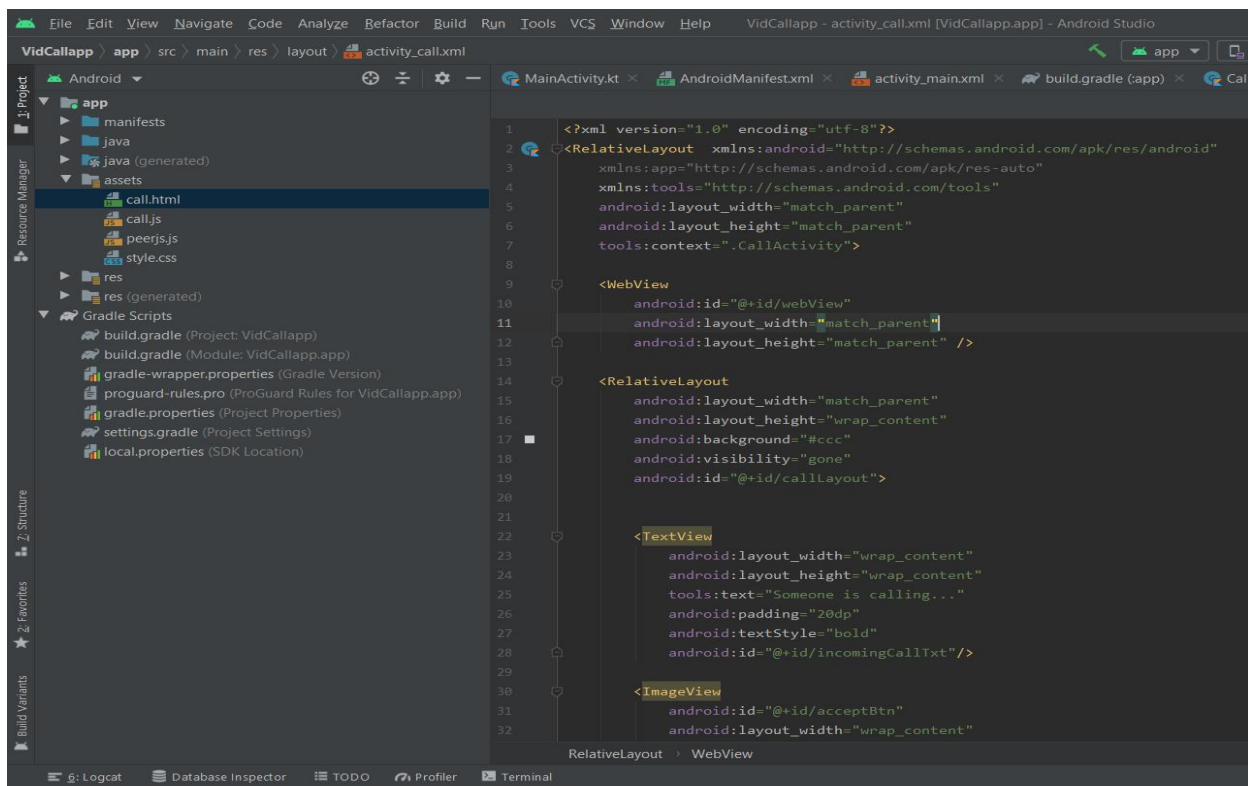
RelativeLayout: RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.

Webview: WebView is a view that displays web pages inside your application. You can also specify HTML string and can show it inside your application using WebView.

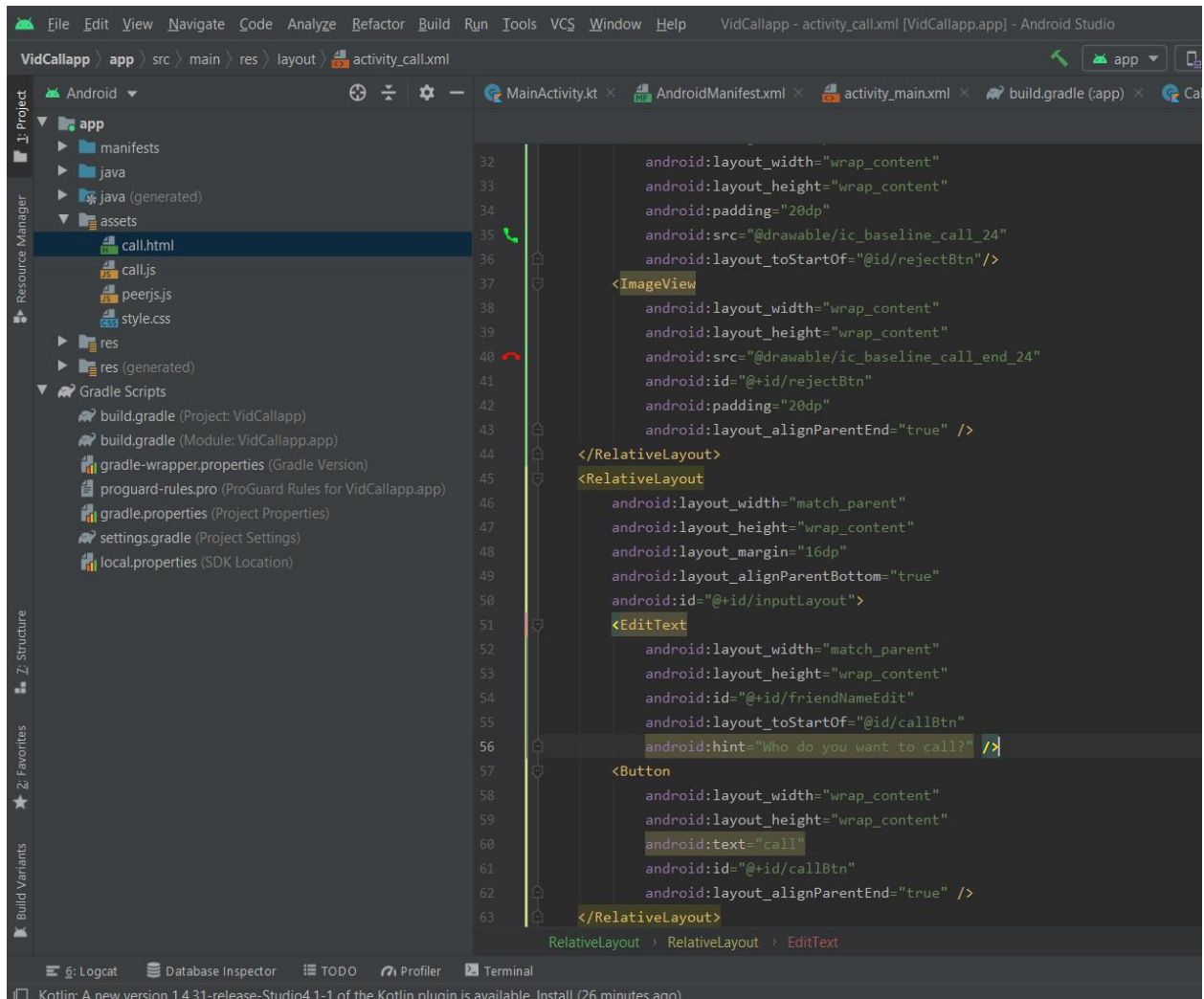
Textview: A TextView displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however, the basic class is configured to not allow editing.

ImageView: ImageView class is used to display an image file in the application. Image file is easy to use but hard to master in Android, because of the various screen sizes in Android devices

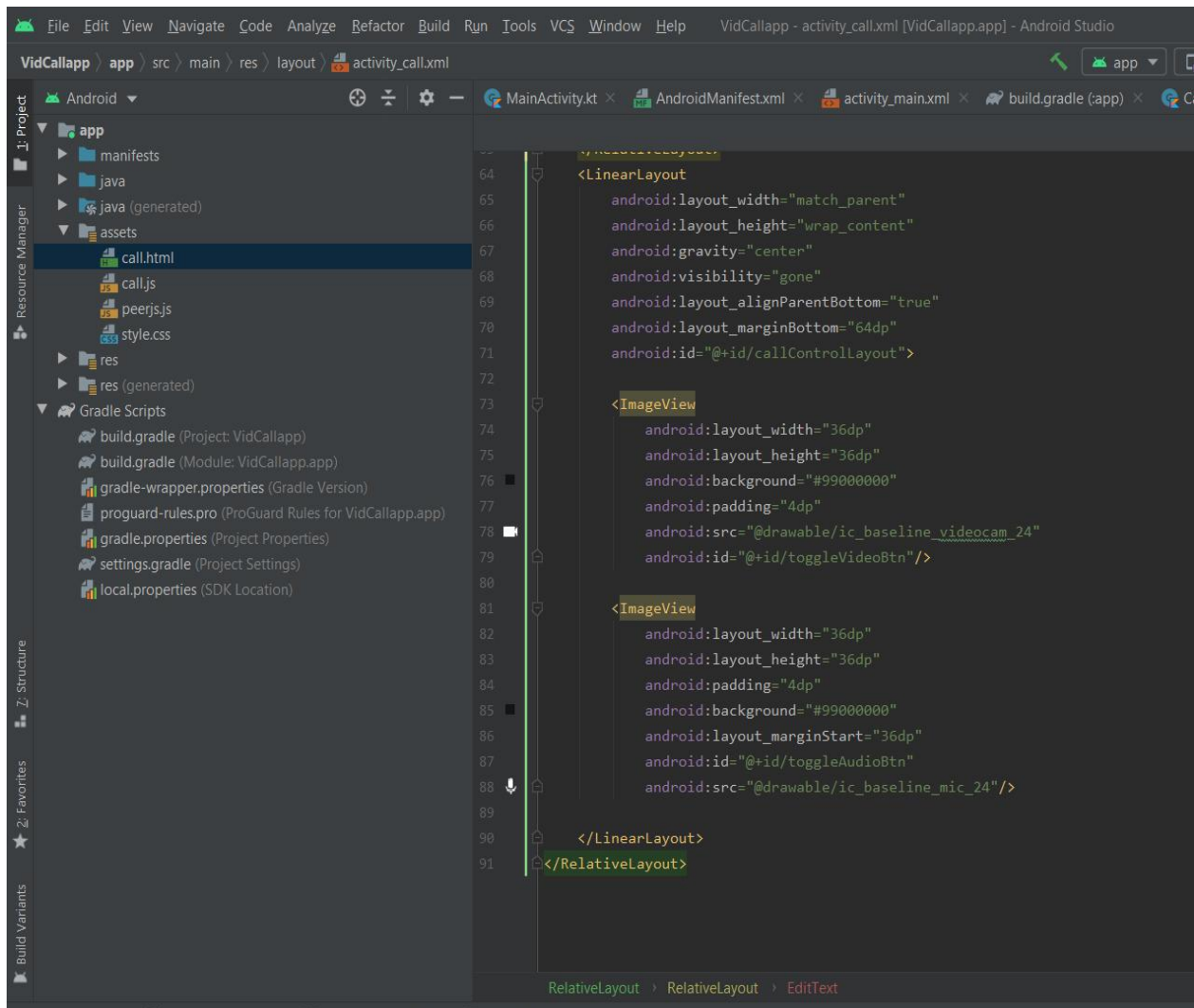
Code



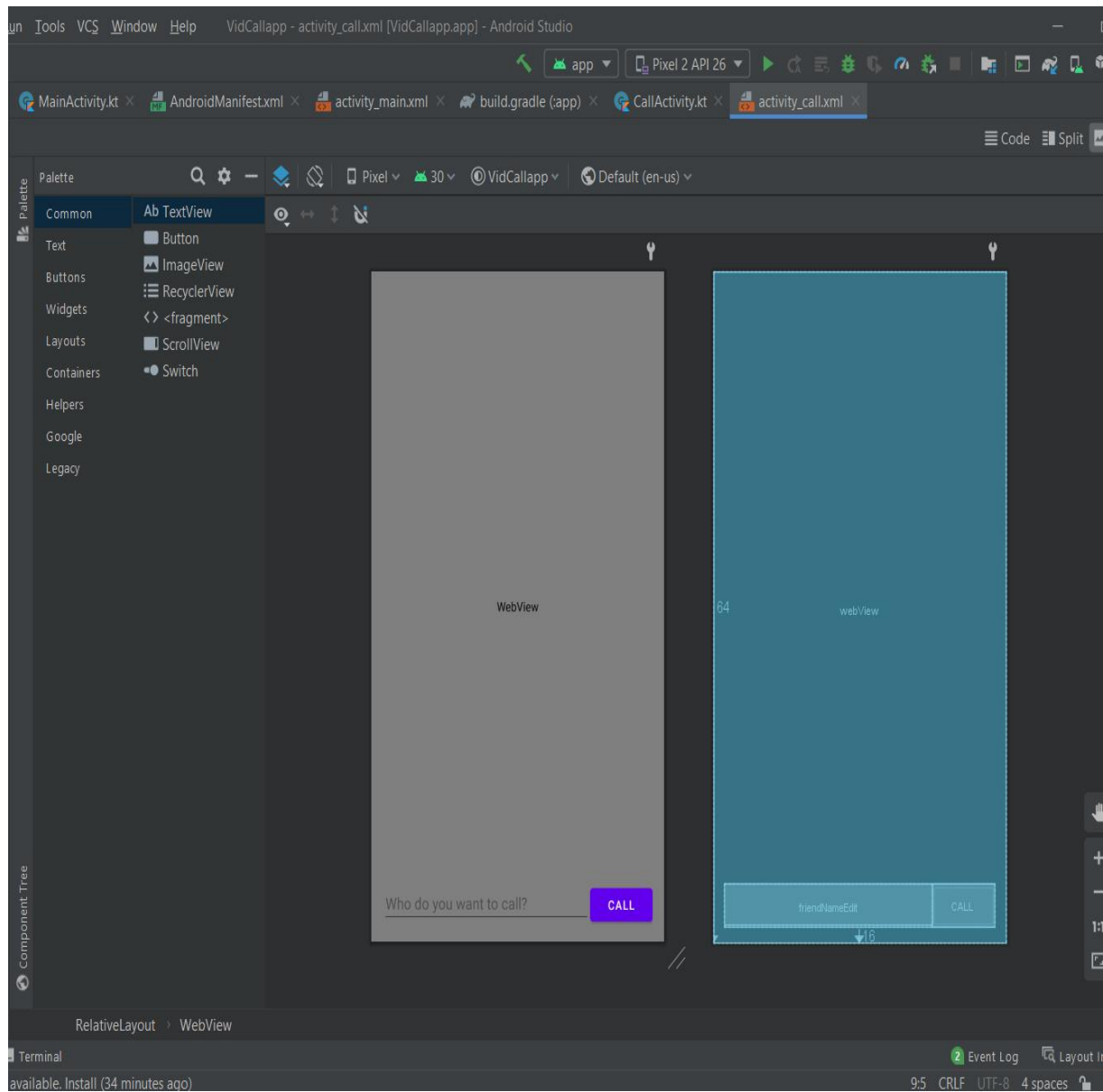
Code



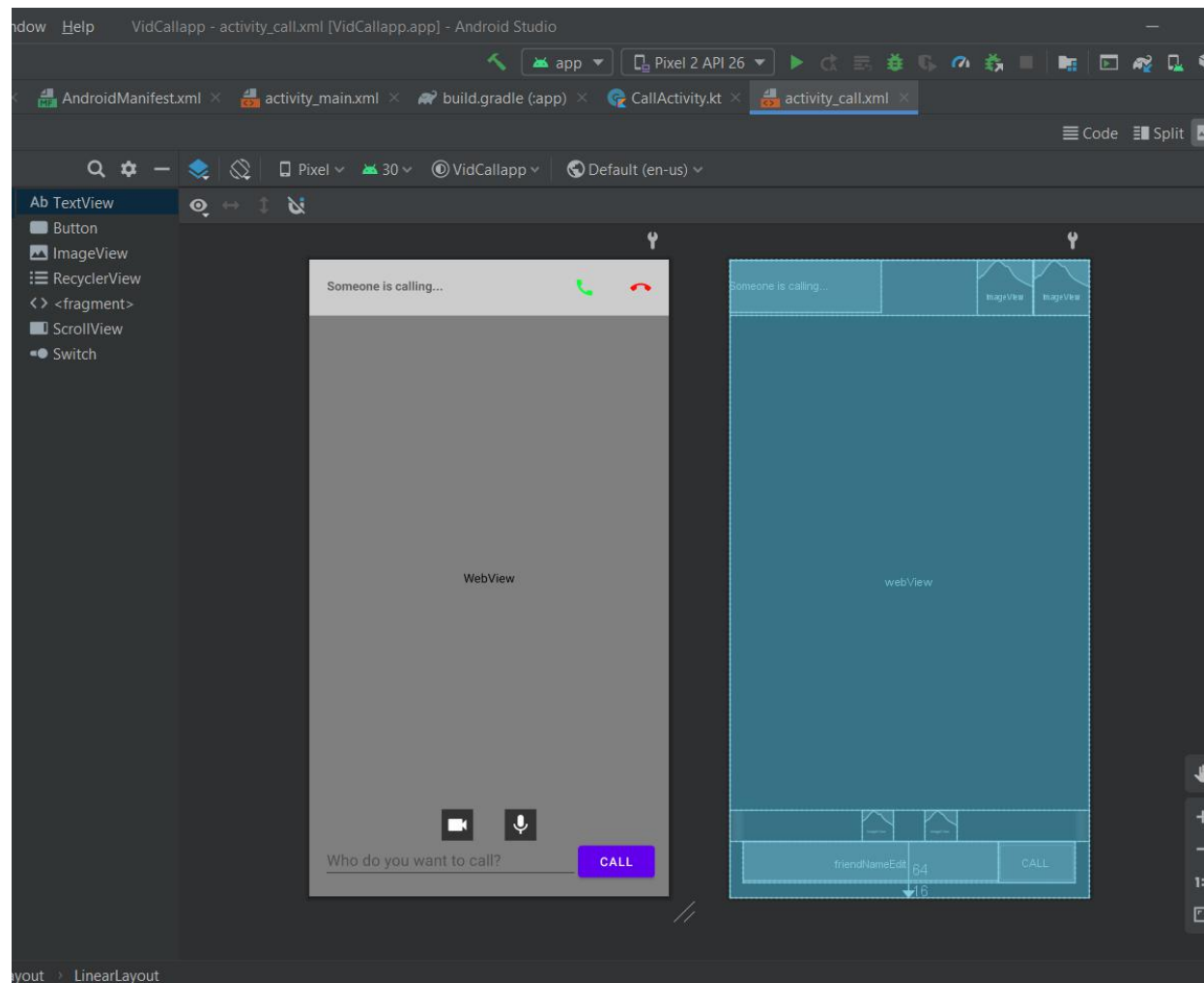
Code



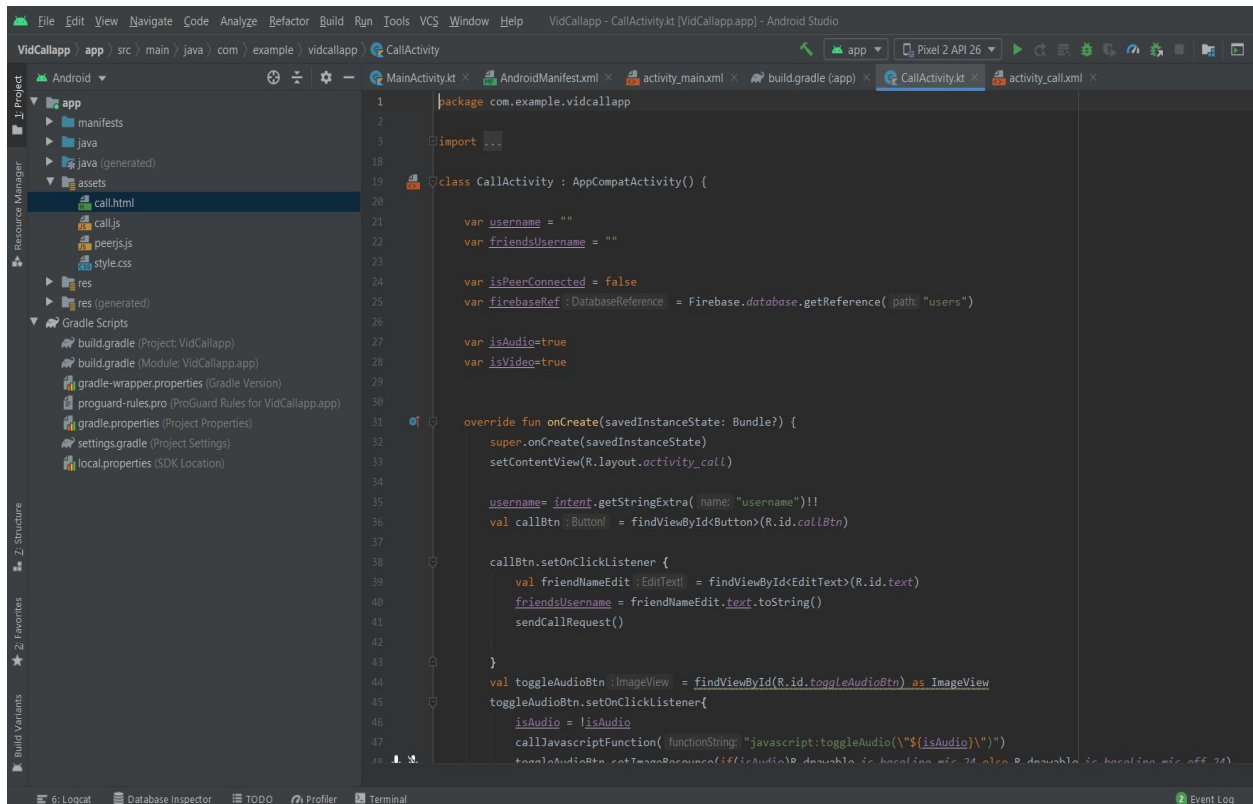
Design



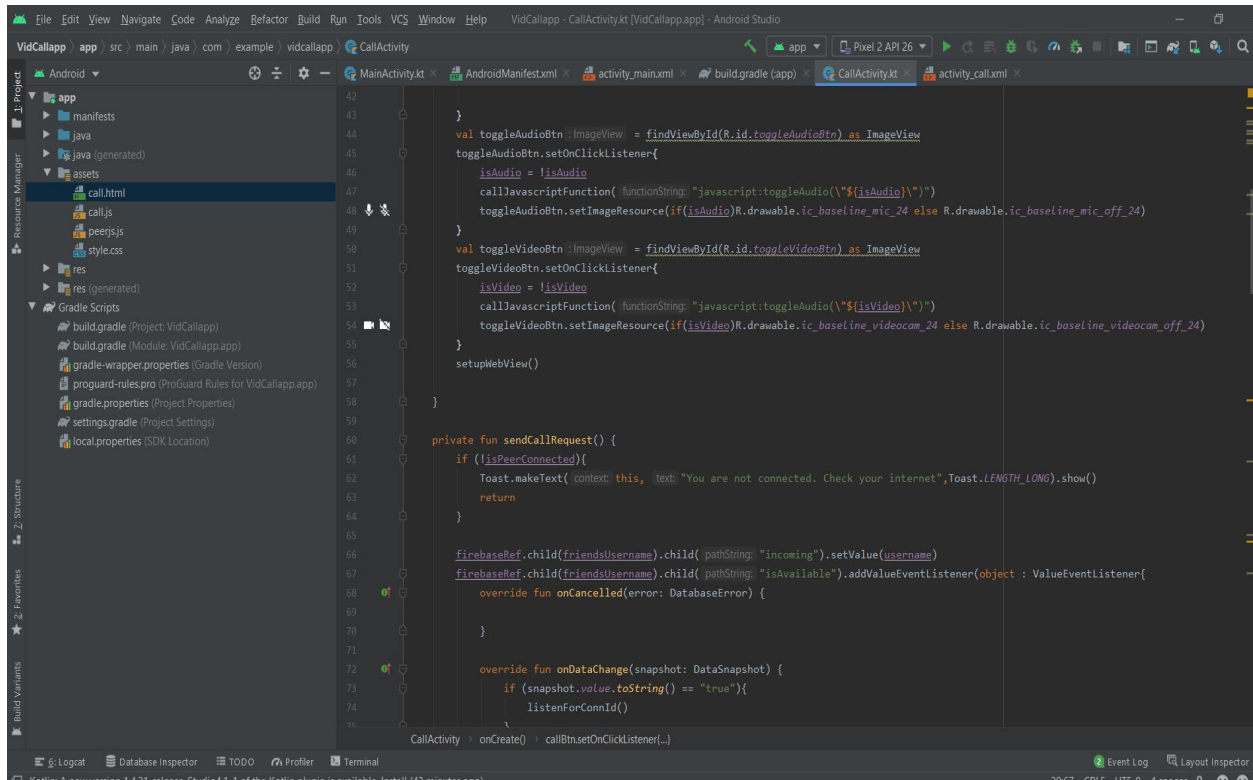
Design



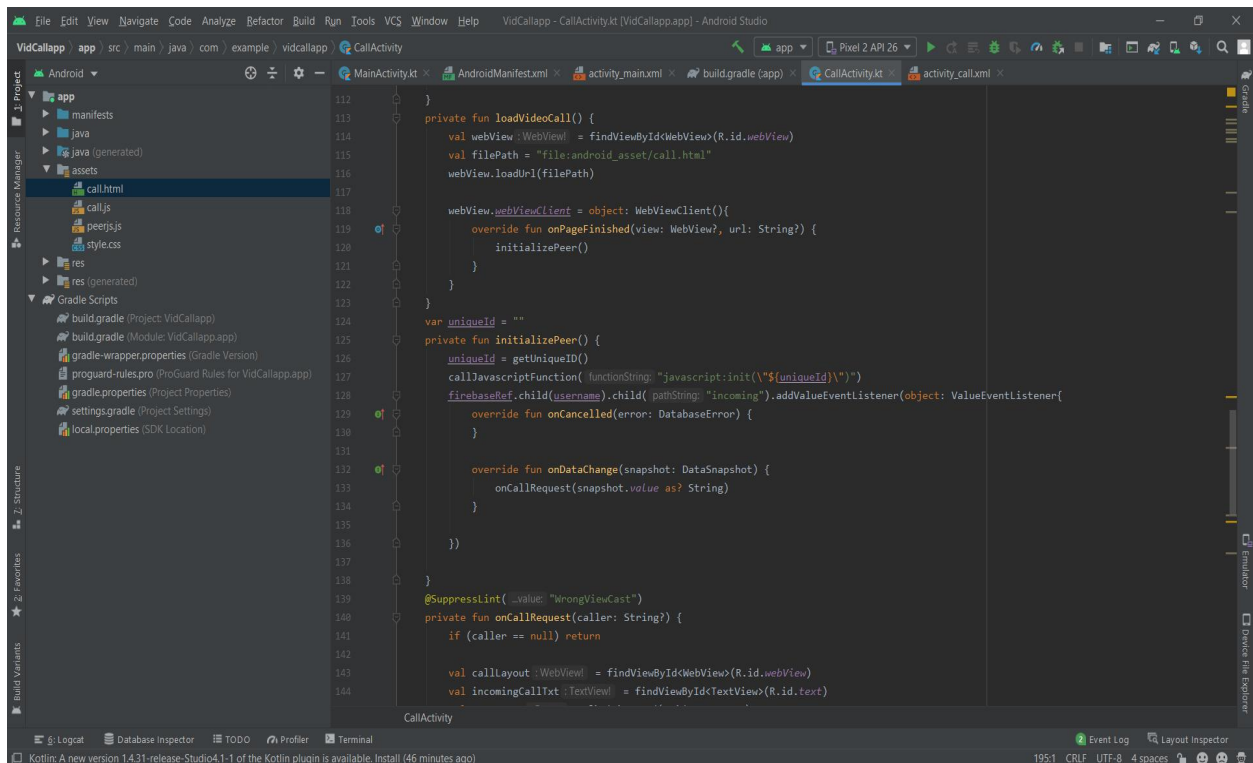
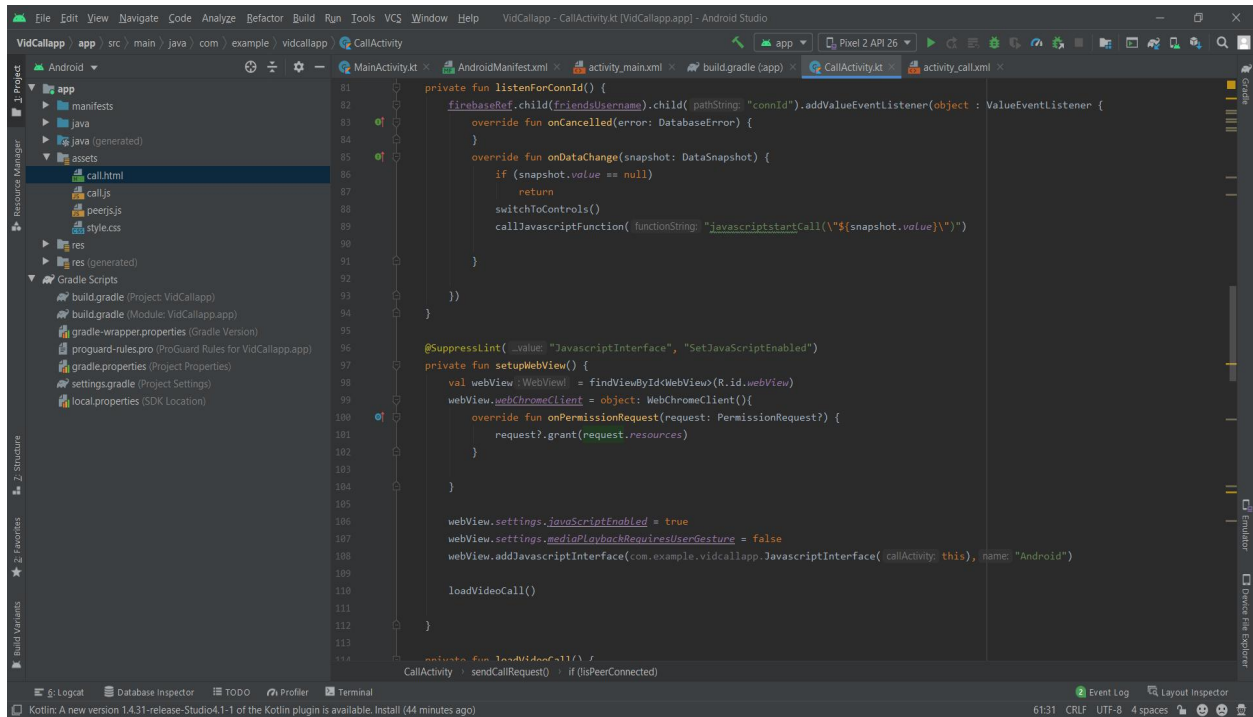
CallActivity.kt

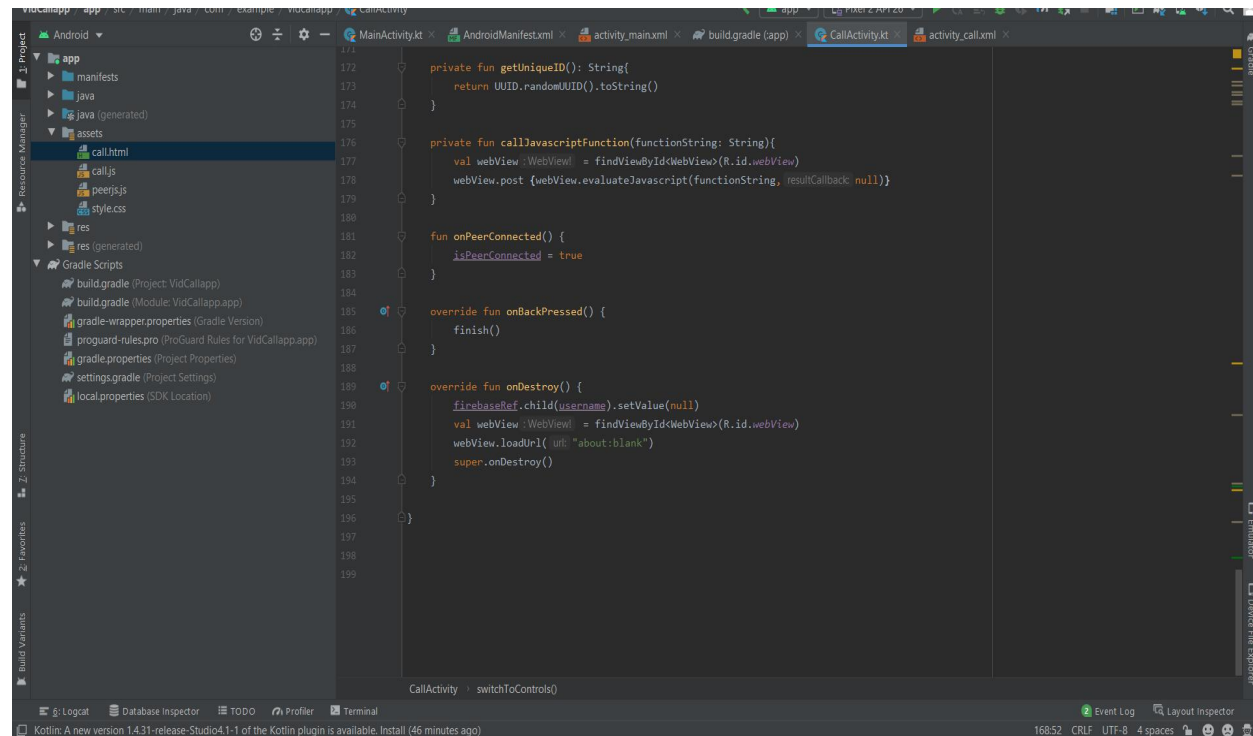
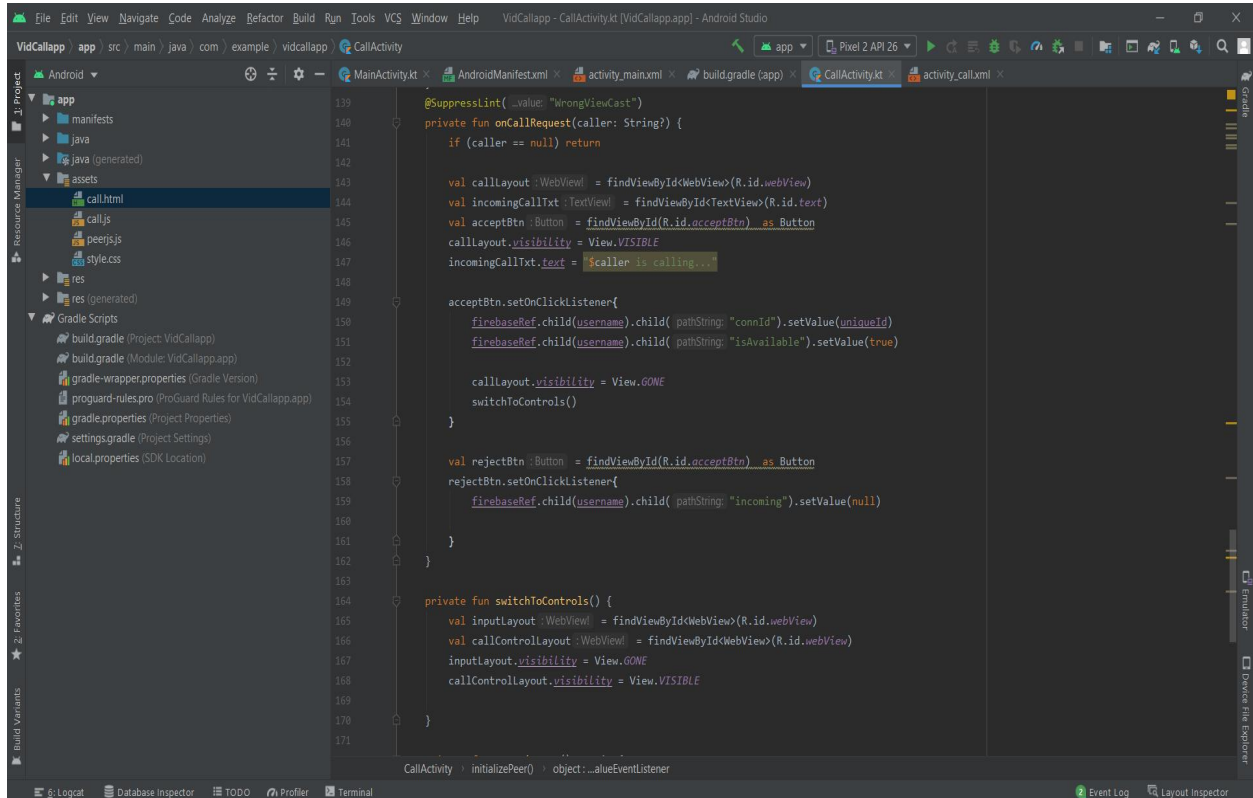


```
1 package com.example.vidcallapp
2
3 import ...
4
5 class CallActivity : AppCompatActivity() {
6
7     var username = ""
8     var friendsUsername = ""
9
10    var isPeerConnected = false
11    var firebaseRef : DatabaseReference = Firebase.database.getReference( path: "users")
12
13    var isAudio=true
14    var isVideo=true
15
16    override fun onCreate(savedInstanceState: Bundle?) {
17        super.onCreate(savedInstanceState)
18        setContentView(R.layout.activity_call)
19
20        username= intent.getStringExtra( name: "username")!!
21        val callBtn : Button! = findViewById<Button>(R.id.callBtn)
22
23        callBtn.setOnClickListener {
24            val friendNameEdit : EditText! = findViewById<EditText>(R.id.text)
25            friendsUsername = friendNameEdit.text.toString()
26            sendCallRequest()
27        }
28
29        val toggleAudioBtn : ImageView = findViewById(R.id.toggleAudioBtn) as ImageView
30        toggleAudioBtn.setOnClickListener{
31            isAudio = !isAudio
32            callJavaScriptFunction( functionString: "javascript:toggleAudio('${isAudio}')" )
33            toggleAudioBtn.setImageResource(if(isAudio)R.drawable.ic_baseline_mic_24 else R.drawable.ic_baseline_mic_off_24)
34        }
35    }
36}
```



```
42
43
44 val toggleAudioBtn : ImageView = findViewById(R.id.toggleAudioBtn) as ImageView
45 toggleAudioBtn.setOnClickListener{
46     isAudio = !isAudio
47     callJavaScriptFunction( functionString: "javascript:toggleAudio('${isAudio}')" )
48     toggleAudioBtn.setImageResource(if(isAudio)R.drawable.ic_baseline_mic_24 else R.drawable.ic_baseline_mic_off_24)
49 }
50
51 val toggleVideoBtn : ImageView = findViewById(R.id.toggleVideoBtn) as ImageView
52 toggleVideoBtn.setOnClickListener{
53     isVideo = !isVideo
54     callJavaScriptFunction( functionString: "javascript:toggleVideo('${isVideo}')" )
55     toggleVideoBtn.setImageResource(if(isVideo)R.drawable.ic_baseline_videocam_24 else R.drawable.ic_baseline_videocam_off_24)
56 }
57
58 setupWebView()
59
60 private fun sendCallRequest() {
61     if (!isPeerConnected){
62         Toast.makeText( context: this, text: "You are not connected. Check your internet", Toast.LENGTH_LONG).show()
63         return
64     }
65
66     firebaseRef.child(friendsUsername).child( pathString: "incoming").setValue(username)
67     firebaseRef.child(friendsUsername).child( pathString: "isAvailable").addValueEventListener(object : ValueEventListener{
68         override fun onCancelled(error: DatabaseError) {
69
70         }
71
72         override fun onDataChange(snapshot: DataSnapshot) {
73             if (snapshot.value.toString() == "true"){
74                 listenForConnId()
75             }
76         }
77     })
78 }
```





4.Conclusion

WebRTC represents the most complete standardized collection of technologies needed to create a reliable, high-quality, open, and compatible real-time communication platform.

In this project, we learned how to use Firebase and FirebaseUI to create a very simple group chat application. You also saw how easy it is to work with the classes available in FirebaseUI to quickly create new screens and implement complex functionality.

Android as a full, open, and free mobile device platform, with its powerful function and good user experience rapidly developed into the most popular mobile operating system. This report provides an overview of the various challenges and problems faced in android app development. The experience of developing an android app is kind of challenging, motivating as well as satisfying.

BIBLIOGRAPHY

- WebRTC and How Does It Work?

<https://www.innoarchitech.com/blog/what-is-webrtc-and-how-does-it-work>

- Real time communication with WebRTC

<https://codelabs.developers.google.com/codelabs/webrtc-web#0>

- Android Studio Tutorial - Android developers

<https://developer.android.com/studio>

- Android - Login Screen

https://www.tutorialspoint.com/android/android_login_screen.html

Project Report

ORIGINALITY REPORT

30%	27%	8%	19%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Bedfordshire Student Paper	3%
2	sites.google.com Internet Source	2%
3	proceedings.elseconference.eu Internet Source	2%
4	www.innoarchitech.com Internet Source	2%
5	www.frozenmountain.com Internet Source	2%
6	www.scribd.com Internet Source	2%
7	Submitted to Softwarica College Of IT & E-Commerce Student Paper	2%
8	Submitted to DIT university Student Paper	1%
9	www.xspdf.com	

	Internet Source	1%
10	maxprog.net.pl Internet Source	1%
11	Submitted to Hoa Sen University Student Paper	1%
12	code.tutsplus.com Internet Source	1%
13	Submitted to Emirates Aviation College, Aerospace & Academic Studies Student Paper	1%
14	tsh.io Internet Source	1%
15	Submitted to Universiti Malaysia Sarawak Student Paper	1%
16	abhiandroid.com Internet Source	1%
17	Submitted to General Sir John Kotelawala Defence University Student Paper	1%
18	download.atlantis-press.com Internet Source	1%
19	sourceforge.net Internet Source	1%

20	Submitted to Kingston University Student Paper	1%
21	Submitted to CSU, San Jose State University Student Paper	1%
22	www.techlegends.in Internet Source	1%
23	Boni García, Micael Gallego, Francisco Gortázar, Antonia Bertolino. "Understanding and estimating quality of experience in WebRTC applications", Computing, 2018 Publication	1%
24	ocw.cs.pub.ro Internet Source	1%

Exclude quotes On
Exclude bibliography On

Exclude matches < 1%