



*Thakur Educational Trust's (Regd.)*  
**THAKUR COLLEGE OF SCIENCE & COMMERCE**  
AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI  
NAAC Accredited Grade 'A' (3<sup>rd</sup> Cycle) & ISO 9001: 2015 (Certified)  
**Best College Award by University of Mumbai for the Year 2018-2019**



THAKUR COLLEGE OF SCIENCE AND COMMERCE

(AUTONOMOUS)

Thakur village, Kandivali (E),

Mumbai 400101

A PROJECT REPORT ON

Video Conference App

BY

Mr. Vivek Tiwari

BACHELOR OF SCIENCE (COMPUTER SCIENCE)

UNDER THE GUIDANCE OF

ASST. PROF. SHIVKUMAR CHANDEY

DEPARTMENT OF COMPUTER SCIENCE

(2021-2022)



*Thakur Educational Trust's (Regd.)*  
**THAKUR COLLEGE OF SCIENCE & COMMERCE**  
AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI  
NAAC Accredited Grade 'A' (3<sup>rd</sup> Cycle) & ISO 9001: 2015 (Certified)  
**Best College Award by University of Mumbai for the Year 2018-2019**



## Certificate of Project Completion

### COMPUTER SCIENCE DEPARTMENT

(2021-2022)

This is to certify that the project work entitled “**Video Conference App**” is prepared by **Mr. Vivek NavalKishor Tiwari** a student of “**Third Year Bachelor of Science (Computer Science)**” course of University of Mumbai, which is conducted by our college.

This project has been completed within the given time period & project report is submitted in partial fulfillment of B.Sc. (Computer Science) course as per rules of University of Mumbai during the academic year 2021-2022.

Asst. Prof. Shivkumar Chandey

**Project Guide**

Mr. Ashish Trivedi

**Head of Department**

\_\_\_\_\_  
**External Examiner**

**College Stamp**

# Index

<b>Sr. No.</b>	<b>Index Topic</b>
1	<b>Acknowledgement</b>
2	<b>Organizational Overview</b>
3	<b>Description of System</b>
4	<b>Limitations of present system</b>
5	<b>Proposed system</b>
6	<b>Objectives</b>
7	<b>System Requirements</b>
8	<b>Overview of Technologies Used</b>
9	<b>Event Table</b>
10	<b>Activity Diagram</b>
11	<b>Database Flowchart</b>

12	<b>System Implementation</b>
13	<b>Future Enhancements</b>
14	<b>References</b>
15	<b>Plagiarism Report</b>

## Acknowledgement

Achievement is finding out what you would be doing rather than what you have to do. It is not until you undertake such a project that you realize how much effort and hard work it really is, what are your capabilities and how well you can present yourself or other things. It gives me immense pleasure to present this report towards the fulfillment of my project.

I take this opportunity to express my profound gratitude to management of Thakur Degree College of Science & Commerce for giving me this opportunity to accomplish this project work.

I am very much Thankful to **Mrs. C. T. Chakraborty - Principal of Thakur College** for their kind Co-Operation in the completion of my project.

A Special Vote of Thanks to our HOD (Head of Department) **Mr. Ashish Trivedi** and to our Project Guide **Asst. Prof. Shivkumar Chandey**.

Finally, I would like to Thank all my Friends & entire Computer Science Department who directly or indirectly helped me in Completion of this Project and to my Family, without whose Support, Motivation and Encouragement this would not have been Possible.

**Mr. Vivek Tiwari**

## Organizational Overview

The **Thakur College of Science and Commerce (TCSC)** is a College in Kandivali, Mumbai of Maharashtra, India running by Thakur Educational Trust.

Thakur College was started in 1992 to serve the needs of students passing SSC examination from the schools around Kandivali area and Thakur Vidhya Mandir which has already established itself as one of the schools in the area. It offers courses at primarily the higher secondary and under-graduate levels. The courses at the undergraduate and post-graduate level are offered in affiliation with Mumbai University, Mumbai. An ISO 9001:2008 College with A grade as assessed by the National Assessment and Accreditation Council NAAC.

Name	Thakur College of Science and Commerce
Founded	1997
Address	Thakur College of Science and Commerce, Thakur Village Kandivali(E), Mumbai – 400 001
Motto	Journey towards Excellence
Total Staff	200
Number of Students	12500
Email	helpdesk@tcsc.org.in

# Description of the System



## **Description of the System**

Teleconferencing systems and services, are the main set of technologies developed thus far to support group work. Within this set of technologies, videoconferencing is often thought of as a new, futuristic communication mode that lies between the telephone call and the face-to-face meeting. In fact, videoconferencing has been commercially available for over two decades, and, despite consistently brilliant market forecasts, to date it has failed to succeed except in limited niche markets.

- As the times we live are completely virtual, the world as of now depends on these apps.
- These apps are now a critical part of our lives.
- So necessarily, we should have a variety of options to choose from so as to increase the competition, which will lead to better products for end users.
- Quality of life changes to these apps will lead to better user experience overall.

## **Limitations of Present System**

- The current industry for such apps is dominated by the big tech giants.
- The biggest limitation faced by the users is their data and the feature set is controlled by these giants.
- So naturally, there's always this dependency on these giants in an integral necessary functionality of our lives.
- As users initially consent to EULA, they've little to no control of their data, except the assurances provided by these companies.
- Also, Closed source systems are prone to security attacks by hackers, in 2020 the security attack on zoom led to stealing of various user data.
- As the user doesn't have control of their data, they're at the mercy of these companies on how they handle these data breaches.

## **Proposed System**

- Open-Source options end up being more secure as the source code is available to everyone.
- So, any loop holes in the system are fixed by the users themselves as needed to be.
- Thus, creation and integration of this open-source app leads to better security.
- In addition to providing an open-source solution, the app collects minimal to no data, and just focuses on the task at hand i.e connecting people.
- As the proposed system is open source in nature, the users are also free to check where and how their data is stored without the
- A unique feature of the app being, the users don't need to enter their actual credentials to avail the functionality of the app.
- They're free to enter alias details, to sign in into the app and connect.

- As there's no EULA, the users aren't bound by any contracts of how their data is used.
- This in turn leads to promote the open-source nature of the app.
- A free option amongst paid options.

## Objectives

- Jitsi Meet SDK is a popular, free and open-source SDK that lets users incorporate Jitsi meet SDK in their own app.
- It runs on the famous WebRTC API, which is backed by various organization, which helps in developing cross-platform services.
- The app incorporates these SDKs, so as to enable cross platform services.
- This allows users to access the meet from a browser of choice.
- The main objective being allowing and giving the users the flexibility to use the app or to avoid it.
- This helps in building a healthy user base which trusts the development of the app as well as is in control of their data.
- Another major objective is to support a wide userbase, which will help the end users to connect seamlessly.

## System Requirements

### Minimum System requirements for development:

- x86\_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for virtualisation.
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution.

### Software Requirements:

- **Operating System:** Microsoft Windows 10/11 (64-bit)
- **IDE:** Google Android Studio
- **Front-End:** XML
- **Back-End:** Java
- **SDKs:** Jitsi meet SDK.
- **Database:** Google Firebase.

### Minimum requirements to run the application:

- API Level 27.
- 2 GB Ram or Higher.

## Overview of Technologies Used

### Android Studio:

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support

- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

## **XML:**

XML stands for Extensible Markup Language, which gives us a clue to what it does.

XML describes the views in your activities, and Java tells them how to behave. To make changes to the layout of your app then, you have two main options.

The first is to use the Design view. Open up the `activity_main.xml` file in Android Studio and get your first introduction to XML. You'll notice there are two tabs at the bottom of that window: Design and Text. The Text view will show you the actual XML code, but the Design view will let you manually edit the layout by dragging and dropping elements into the render of your activity.

## **Java:**

Java is fast, reliable and secure. From desktop to web applications, scientific supercomputers to gaming consoles, cell phones to the Internet, Java is used in every nook and corner.

Java is easy to learn and its syntax is simple and easy to understand. It is based on C++ (so easier for programmers who know C++). Java has removed many



confusing and rarely-used features e.g., explicit pointers, operator overloading etc.

Java does not provide low-level programming functionalities like pointers.

Also, Java code is always written in the form of classes and objects. Android heavily relies on the Java programming language all the SDKs required to build for android applications use the standard libraries of Java.

### **Jitsi Meet SDK:**

The Jitsi Meet Android SDK provides the same user experience as the Jitsi Meet app, in a customizable way which you can embed in your apps.

Jitsi Meet SDK is an Android library which embodies the whole Jitsi Meet experience and makes it reusable by third-party apps.

Jitsi conveniently provides a pre-build SDK artifacts/binary in its Maven repository. When you do not require any modification to the SDK itself or any of its dependencies, it's suggested to use the pre-build SDK. This avoids the complexity of building and installing your own SDK artifacts/binaries

**Firestore:**

Firestore is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure.

Firestore is categorized as a NOSQL database program, which stores data in JSON-like documents.

Firestore Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

Some features of firestore are:

1. Authentication
2. Realtime database
3. Hosting
4. Test lab
5. Notifications

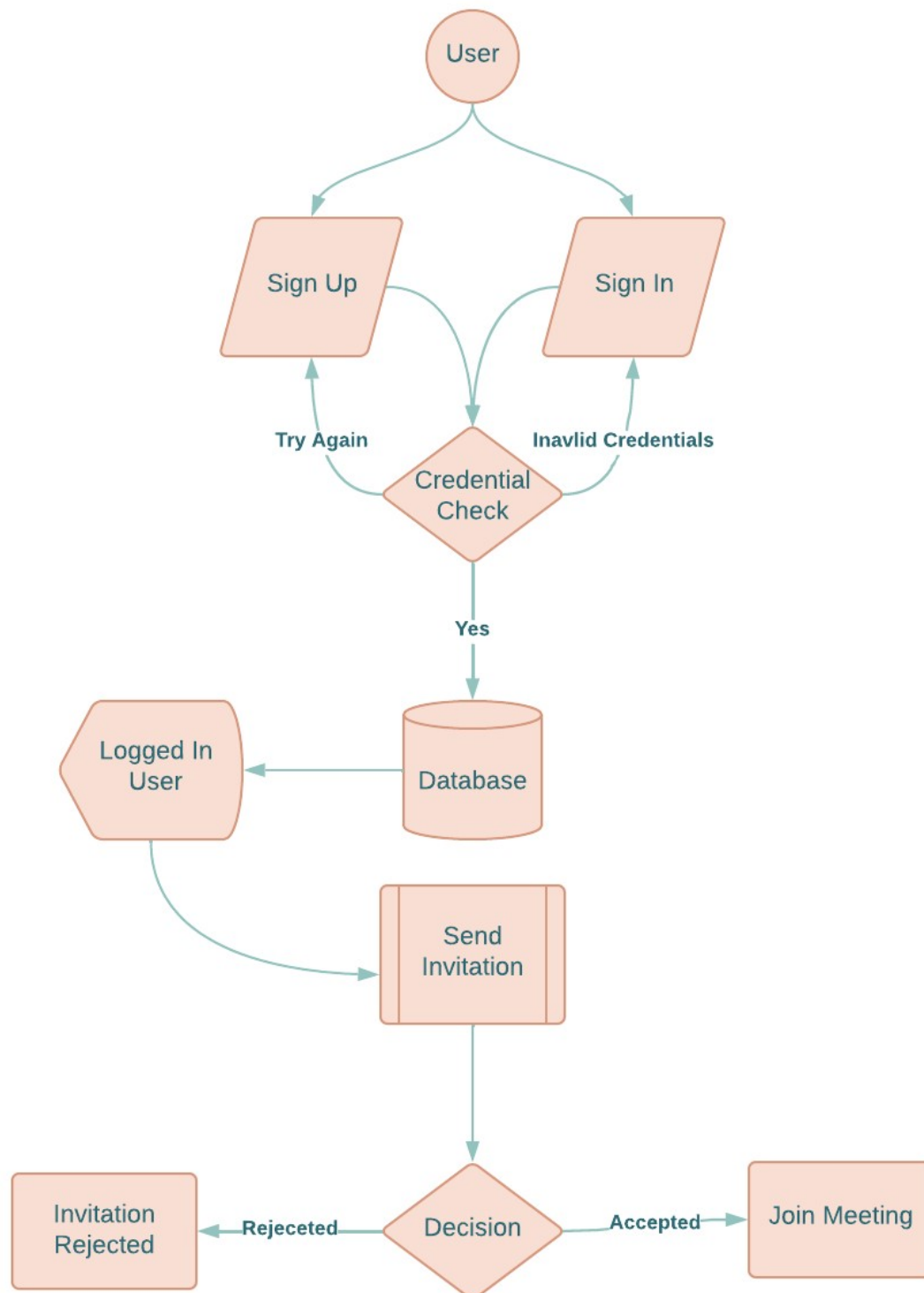
# Event Table

### Event Table

EVENT	TRIGGER	ACTIVITY	RESPONSE	DESTINATION
User Sign up	Initiates user creation	User credentials inserted into database.	User Created/User Creation Failed	Logged In
Clicks Sign In	Initiates user login	Validates user credentials	Signing In/Unable to sign in.	Signed In
Send Meeting Invitation	Prompts Outgoing Invitation	User Tokens exchanged from database to validate invitation sender and receiver.	Invitation Sent/User is not available.	Enter meeting room.
Incoming Meeting Invitation	Prompts Receiving Invitation	User Tokens exchanged from database	Accept/Deny invitation prompt	Join meeting.

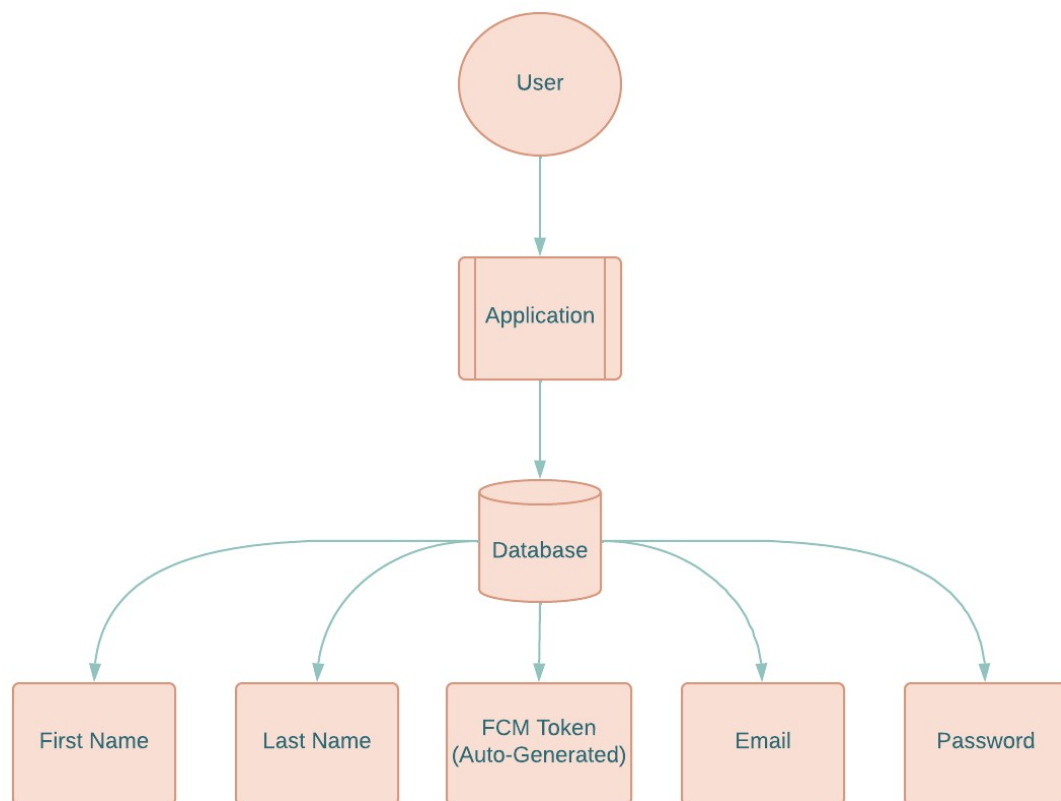
# Activity Diagram

## Activity Diagram



# Flowchart

## Database Flowchart





# **System Implementation**

## System Implementation

### Firestore Connectivity:

```
package com.viv.videomeeting.firestore;

import android.content.Intent;
import android.util.Log;

import androidx.annotation.NonNull;
import androidx.localbroadcastmanager.content.LocalBroadcastManager;

import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;
import com.viv.videomeeting.activities.IncomingInvitationActivity;
import com.viv.videomeeting.utilities.Constants;

public class MessagingService extends FirebaseMessagingService {
    @Override
    public void onNewToken(@NonNull String token) {
        super.onNewToken(token);
    }

    @Override
    public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
        super.onMessageReceived(remoteMessage);

        final String type = remoteMessage.getData().get(Constants.REMOTE_MSG_TYPE);
        if (type != null) {
            if (type.equals(Constants.REMOTE_MSG_INVITATION)) {
                Intent intent = new Intent(getApplicationContext(), IncomingInvitationActivity.class);

                intent.putExtra(Constants.REMOTE_MSG_MEETING_TYPE,
                    remoteMessage.getData().get(Constants.REMOTE_MSG_MEETING_TYPE));
                intent.putExtra(Constants.KEY_FIRST_NAME,
                    remoteMessage.getData().get(Constants.KEY_FIRST_NAME));
                intent.putExtra(Constants.KEY_LAST_NAME,
                    remoteMessage.getData().get(Constants.KEY_LAST_NAME));
                intent.putExtra(Constants.KEY_EMAIL, remoteMessage.getData().get(Constants.KEY_EMAIL));
                intent.putExtra(Constants.REMOTE_MSG_INVITER_TOKEN,
                    remoteMessage.getData().get(Constants.REMOTE_MSG_INVITER_TOKEN));
                intent.putExtra(Constants.REMOTE_MSG_MEETING_ROOM,
                    remoteMessage.getData().get(Constants.REMOTE_MSG_MEETING_ROOM));

                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            } else if (type.equals(Constants.REMOTE_MSG_INVITATION_RESPONSE)) {
                Intent intent = new Intent(Constants.REMOTE_MSG_INVITATION_RESPONSE);
                intent.putExtra(Constants.REMOTE_MSG_INVITATION_RESPONSE,
```

```
remoteMessage.getData().get(Constants.REMOTE_MSG_INVITATION_RESPONSE));
    LocalBroadcastManager.getInstance(getApplicationContext()).sendBroadcast(intent);
    }
    }
    }
}
```

## Firestore Preferences:

```
package com.viv.videomeeting.utilities;
```

```
import java.util.HashMap;
```

```
public class Constants {
    // Firestore
    public static final String KEY_COLLECTION_USERS = "users";
    public static final String KEY_FIRST_NAME = "first_name";
    public static final String KEY_LAST_NAME = "last_name";
    public static final String KEY_EMAIL = "email";
    public static final String KEY_PASSWORD = "password";
    public static final String KEY_USER_ID = "user_id";
    public static final String KEY_FCM_TOKEN = "fcm_token";

    // Preference
    public static final String KEY_PREFERENCE_NAME = "videoMeetingPreference";
    public static final String KEY_IS_SIGNED_IN = "isSignedIn";

    public static final String REMOTE_MSG_AUTHORIZATION = "Authorization";
    public static final String REMOTE_MSG_CONTENT_TYPE = "Content-Type";

    public static final String REMOTE_MSG_TYPE = "type";
    public static final String REMOTE_MSG_INVITATION = "invitation";
    public static final String REMOTE_MSG_MEETING_TYPE = "meetingType";
    public static final String REMOTE_MSG_INVITER_TOKEN = "inviterToken";
    public static final String REMOTE_MSG_DATA = "data";
    public static final String REMOTE_MSG_REGISTRATION_IDS = "registration_ids";

    public static final String REMOTE_MSG_INVITATION_RESPONSE =
    "invitationResponse";

    public static final String REMOTE_MSG_INVITATION_ACCEPTED = "accepted";
    public static final String REMOTE_MSG_INVITATION_REJECTED = "rejected";
    public static final String REMOTE_MSG_INVITATION_CANCELLED = "cancelled";

    public static final String REMOTE_MSG_MEETING_ROOM = "meetingRoom";
```

```

public static HashMap<String, String> getRemoteMessageHeaders() {
    HashMap<String, String> headers = new HashMap<>();
    headers.put(Constants.REMOTE_MSG_AUTHORIZATION,
        "key=AAAAsb0O3ig:APA91bEJIB5Wary91hnpAe-lQfQ_sfdKQOOZ6507-
91lLtOQTc-
GlyElzn7pUcFWaqPHY4n7utBxMvKAZsM_fnE_wRczFklFv6hVS_07j_LjdyLoZHqGDQg
73Bk7zFhDoBj27yHJ0Py3");
    headers.put(Constants.REMOTE_MSG_CONTENT_TYPE, "application/json");
    return headers;
}
}

```

## Users Adapter:

```

package com.viv.videomeeting.adapters;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.recyclerview.widget.RecyclerView;

import com.viv.videomeeting.R;
import com.viv.videomeeting.listeners.UsersListener;
import com.viv.videomeeting.models.User;

import java.util.ArrayList;
import java.util.List;

public class UsersAdapter extends
RecyclerView.Adapter<UsersAdapter.UserViewHolder> {

    private final List<User> users, selectedUsers;
    private final UsersListener usersListener;

    public UsersAdapter(List<User> users, UsersListener
usersListener) {
        this.users = users;
        this.usersListener = usersListener;
        this.selectedUsers = new ArrayList<>();
    }
}

```

```
public List<User> getSelectedUsers() {
    return selectedUsers;
}

@NonNull
@Override
public UserViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
    return new
UserViewHolder(LayoutInflater.from(parent.getContext())
        .inflate(R.layout.item_container_user, parent,
false));
}

@Override
public void onBindViewHolder(@NonNull UserViewHolder
holder, int position) {
    holder.setUserData(users.get(position));
}

@Override
public int getItemCount() {
    return users.size();
}

class UserViewHolder extends RecyclerView.ViewHolder {

    TextView textFirstChar, textUsername, textEmail;
    ImageView imageAudioMeeting, imageVideoMeeting,
imageSelected;
    ConstraintLayout userContainer;

    UserViewHolder(@NonNull View itemView) {
        super(itemView);
        textFirstChar =
itemView.findViewById(R.id.textFirstChar);
        textUsername =
itemView.findViewById(R.id.textUsername);
        textEmail = itemView.findViewById(R.id.textEmail);

        imageAudioMeeting =
itemView.findViewById(R.id.imageAudioMeeting);
        imageVideoMeeting =
itemView.findViewById(R.id.imageVideoMeeting);
        imageSelected =
itemView.findViewById(R.id.imageSelected);
    }
}
```

```
        userContainer =
itemView.findViewById(R.id.userContainer);
    }

    void setData(User user) {
        textFirstChar.setText(user.firstName.substring(0,
1));
        textUsername.setText(String.format("%s %s",
user.firstName, user.lastName));
        textEmail.setText(user.email);
        imageAudioMeeting.setOnClickListener(v ->
usersListener.initiateAudioMeeting(user));
        imageVideoMeeting.setOnClickListener(v ->
usersListener.initiateVideoMeeting(user));

        userContainer.setOnLongClickListener(v -> {
            if (imageSelected.getVisibility() ==
View.GONE) {
                selectedUsers.add(user);
                imageSelected.setVisibility(View.VISIBLE);

imageAudioMeeting.setVisibility(View.GONE);

imageVideoMeeting.setVisibility(View.GONE);
                usersListener.onMultipleUsersAction(true);
            }
            return true;
        });

        userContainer.setOnClickListener(v -> {
            if (imageSelected.getVisibility() ==
View.VISIBLE) {
                selectedUsers.remove(user);
                imageSelected.setVisibility(View.GONE);

imageAudioMeeting.setVisibility(View.VISIBLE);

imageVideoMeeting.setVisibility(View.VISIBLE);
                if (selectedUsers.size() == 0) {

usersListener.onMultipleUsersAction(false);
                }
            } else {
                if (selectedUsers.size() > 0) {

imageSelected.setVisibility(View.VISIBLE);
```

```
imageAudioMeeting.setVisibility(View.GONE);
package com.viv.videomeeting.adapters;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.recyclerview.widget.RecyclerView;

import com.viv.videomeeting.R;
import com.viv.videomeeting.listeners.UsersListener;
import com.viv.videomeeting.models.User;

import java.util.ArrayList;
import java.util.List;

public class UsersAdapter extends RecyclerView.Adapter<UsersAdapter.UserViewHolder>
{

    private final List<User> users, selectedUsers;
    private final UsersListener usersListener;

    public UsersAdapter(List<User> users, UsersListener usersListener) {
        this.users = users;
        this.usersListener = usersListener;
        this.selectedUsers = new ArrayList<>();
    }

    public List<User> getSelectedUsers() {
        return selectedUsers;
    }

    @NonNull
    @Override
    public UserViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        return new UserViewHolder(LayoutInflater.from(parent.getContext())
            .inflate(R.layout.item_container_user, parent, false));
    }

    @Override
    public void onBindViewHolder(@NonNull UserViewHolder holder, int position) {
```

```
        holder.setUserData(users.get(position));
    }

    @Override
    public int getItemCount() {
        return users.size();
    }

    class ViewHolder extends RecyclerView.ViewHolder {

        TextView textFirstChar, textUsername, textEmail;
        ImageView imageAudioMeeting, imageVideoMeeting, imageSelected;
        ConstraintLayout userContainer;

        ViewHolder(@NonNull View itemView) {
            super(itemView);
            textFirstChar = itemView.findViewById(R.id.textFirstChar);
            textUsername = itemView.findViewById(R.id.textUsername);
            textEmail = itemView.findViewById(R.id.textEmail);

            imageAudioMeeting = itemView.findViewById(R.id.imageAudioMeeting);
            imageVideoMeeting = itemView.findViewById(R.id.imageVideoMeeting);
            imageSelected = itemView.findViewById(R.id.imageSelected);

            userContainer = itemView.findViewById(R.id.userContainer);
        }

        void setUserData(User user) {
            textFirstChar.setText(user.firstName.substring(0, 1));
            textUsername.setText(String.format("%s %s", user.firstName, user.lastName));
            textEmail.setText(user.email);
            imageAudioMeeting.setOnClickListener(v ->
usersListener.initiateAudioMeeting(user));
            imageVideoMeeting.setOnClickListener(v ->
usersListener.initiateVideoMeeting(user));

            userContainer.setOnLongClickListener(v -> {
                if (imageSelected.getVisibility() == View.GONE) {
                    selectedUsers.add(user);
                    imageSelected.setVisibility(View.VISIBLE);
                    imageAudioMeeting.setVisibility(View.GONE);
                    imageVideoMeeting.setVisibility(View.GONE);
                    usersListener.onMultipleUsersAction(true);
                }
                return true;
            });
        }
    }
}
```



```

        userContainer.setOnClickListener(v -> {
            if (imageSelected.getVisibility() == View.VISIBLE) {
                selectedUsers.remove(user);
                imageSelected.setVisibility(View.GONE);
                imageAudioMeeting.setVisibility(View.VISIBLE);
                imageVideoMeeting.setVisibility(View.VISIBLE);
                if (selectedUsers.size() == 0) {
                    usersListener.onMultipleUsersAction(false);
                }
            } else {
                if (selectedUsers.size() > 0) {
                    imageSelected.setVisibility(View.VISIBLE);
                    imageAudioMeeting.setVisibility(View.GONE);
                    imageVideoMeeting.setVisibility(View.GONE);
                }
            }
        });
    }
}

package com.viv.videomeeting.adapters;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.recyclerview.widget.RecyclerView;

import com.viv.videomeeting.R;
import com.viv.videomeeting.listeners.UsersListener;
import com.viv.videomeeting.models.User;

import java.util.ArrayList;
import java.util.List;

public class UsersAdapter extends
    RecyclerView.Adapter<UsersAdapter.UserViewHolder> {

    private final List<User> users, selectedUsers;
    private final UsersListener usersListener;

```

```
public UsersAdapter(List<User> users, UsersListener
usersListener) {
    this.users = users;
    this.usersListener = usersListener;
    this.selectedUsers = new ArrayList<>();
}

public List<User> getSelectedUsers() {
    return selectedUsers;
}

@NonNull
@Override
public UserViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
    return new
UserViewHolder(LayoutInflater.from(parent.getContext())
.inflate(R.layout.item_container_user, parent,
false));
}

@Override
public void onBindViewHolder(@NonNull UserViewHolder
holder, int position) {
    holder.setUserData(users.get(position));
}

@Override
public int getItemCount() {
    return users.size();
}

class UserViewHolder extends RecyclerView.ViewHolder {

    TextView textFirstChar, textUsername, textEmail;
    ImageView imageAudioMeeting, imageVideoMeeting,
imageSelected;
    ConstraintLayout userContainer;

    UserViewHolder(@NonNull View itemView) {
        super(itemView);
        textFirstChar =
itemView.findViewById(R.id.textFirstChar);
        textUsername =
itemView.findViewById(R.id.textUsername);
        textEmail = itemView.findViewById(R.id.textEmail);
    }
}
```

```
        imageAudioMeeting =
itemView.findViewById(R.id.imageAudioMeeting);
        imageVideoMeeting =
itemView.findViewById(R.id.imageVideoMeeting);
        imageSelected =
itemView.findViewById(R.id.imageSelected);

        userContainer =
itemView.findViewById(R.id.userContainer);
    }

    void setUserData(User user) {
        textFirstChar.setText(user.firstName.substring(0,
1));
        textUsername.setText(String.format("%s %s",
user.firstName, user.lastName));
        textEmail.setText(user.email);
        imageAudioMeeting.setOnClickListener(v ->
usersListener.initiateAudioMeeting(user));
        imageVideoMeeting.setOnClickListener(v ->
usersListener.initiateVideoMeeting(user));

        userContainer.setOnLongClickListener(v -> {
            if (imageSelected.getVisibility() ==
View.GONE) {
                selectedUsers.add(user);
                imageSelected.setVisibility(View.VISIBLE);

imageAudioMeeting.setVisibility(View.GONE);

imageVideoMeeting.setVisibility(View.GONE);
                usersListener.onMultipleUsersAction(true);
            }
            return true;
        });

        userContainer.setOnClickListener(v -> {
            if (imageSelected.getVisibility() ==
View.VISIBLE) {
                selectedUsers.remove(user);
                imageSelected.setVisibility(View.GONE);

imageAudioMeeting.setVisibility(View.VISIBLE);

imageVideoMeeting.setVisibility(View.VISIBLE);
                if (selectedUsers.size() == 0) {
```

```
usersListener.onMultipleUsersAction(false);
    }
    } else {
        if (selectedUsers.size() > 0) {

imageSelected.setVisibility(View.VISIBLE);

imageAudioMeeting.setVisibility(View.GONE);

imageVideoMeeting.setVisibility(View.GONE);
        }
    }
    });
}
}
```

## Main Activity:

```
package com.viv.videomeeting.activities;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.PowerManager;
import android.provider.Settings;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;

import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.iid.FirebaseInstanceId;
import com.google.firebase.messaging.FirebaseMessaging;
import com.google.gson.Gson;
import com.viv.videomeeting.R;
```

```
import com.viv.videomeeting.adapters.UsersAdapter;
import com.viv.videomeeting.listeners.UsersListener;
import com.viv.videomeeting.models.User;
import com.viv.videomeeting.utilities.Constants;
import com.viv.videomeeting.utilities.PreferenceManager;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/*import com.google.firebase.iid.FirebaseInstanceId;*/

public class MainActivity extends AppCompatActivity implements UsersListener {

    private PreferenceManager preferenceManager;
    private List<User> users;
    private UsersAdapter usersAdapter;
    private TextView textErrorMessage;
    private SwipeRefreshLayout swipeRefreshLayout;
    private ImageView imageConference;

    private final int REQUEST_CODE_BATTERY_OPTIMIZATIONS = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        preferenceManager = new PreferenceManager(getApplicationContext());

        imageConference = findViewById(R.id.imageConference);

        TextView textTitle = findViewById(R.id.textTitle);
        textTitle.setText(String.format("%s %s",
            preferenceManager.getString(Constants.KEY_FIRST_NAME),
            preferenceManager.getString(Constants.KEY_LAST_NAME)));

        findViewById(R.id.textSignOut).setOnClickListener(v -> signOut());

        FirebaseInstanceId.getInstance().getInstanceId().addOnCompleteListener(task -> {
            if (task.isSuccessful() && task.getResult() != null) {
                sendFCMTokenToDatabase(task.getResult().getToken());
            }
        });

        RecyclerView usersRecyclerView = findViewById(R.id.usersRecyclerView);
```

```
textErrorMessage = findViewById(R.id.textErrorMessage);

users = new ArrayList<>();
usersAdapter = new UsersAdapter(users, this);
usersRecyclerView.setAdapter(usersAdapter);

swipeRefreshLayout = findViewById(R.id.swipeRefreshLayout);
swipeRefreshLayout.setOnRefreshListener(this::getUsers);

getUsers();
checkForBatteryOptimizations();
}

private void getUsers() {
    swipeRefreshLayout.setRefreshing(true);
    FirebaseFirestore database = FirebaseFirestore.getInstance();
    database.collection(Constants.KEY_COLLECTION_USERS)
        .get()
        .addOnCompleteListener(task -> {
            swipeRefreshLayout.setRefreshing(false);
            String myUserId = preferenceManager.getString(Constants.KEY_USER_ID);
            if (task.isSuccessful() && task.getResult() != null) {
                users.clear();
                for (QueryDocumentSnapshot documentSnapshot : task.getResult()) {
                    if (myUserId.equals(documentSnapshot.getId())) {
                        continue;
                    }
                    User user = new User();
                    user.firstName =
documentSnapshot.getString(Constants.KEY_FIRST_NAME);
                    user.lastName =
documentSnapshot.getString(Constants.KEY_LAST_NAME);
                    user.email = documentSnapshot.getString(Constants.KEY_EMAIL);
                    user.token = documentSnapshot.getString(Constants.KEY_FCM_TOKEN);
                    users.add(user);
                }
                if (users.size() > 0) {
                    usersAdapter.notifyDataSetChanged();
                } else {
                    textErrorMessage.setText(String.format("%s", "No users available"));
                    textErrorMessage.setVisibility(View.VISIBLE);
                }
            } else {
                textErrorMessage.setText(String.format("%s", "No users available"));
                textErrorMessage.setVisibility(View.VISIBLE);
            }
        })
    }
```

```

    });
}

private void sendFCMTokenToDatabase(String token) {
    FirebaseFirestore database = FirebaseFirestore.getInstance();

    DocumentReference documentReference =
        database.collection(Constants.KEY_COLLECTION_USERS)
            .document(preferenceManager.getString(Constants.KEY_USER_ID));

    documentReference.update(Constants.KEY_FCM_TOKEN, token)
        .addOnFailureListener(e ->
            Toast.makeText(MainActivity.this, "Unable to send token: " + e.getMessage(),
                Toast.LENGTH_SHORT).show());
}

private void signOut() {
    Toast.makeText(this, "Signing Out...", Toast.LENGTH_SHORT).show();

    FirebaseFirestore database = FirebaseFirestore.getInstance();
    DocumentReference documentReference =
        database.collection(Constants.KEY_COLLECTION_USERS)
            .document(preferenceManager.getString(Constants.KEY_USER_ID));

    HashMap<String, Object> updates = new HashMap<>();
    updates.put(Constants.KEY_FCM_TOKEN, FieldValue.delete());
    documentReference.update(updates)
        .addOnSuccessListener(aVoid -> {
            preferenceManager.clearPreferences();
            startActivity(new Intent(getApplicationContext(), SignInActivity.class));
            finish();
        })
        .addOnFailureListener(e ->
            Toast.makeText(MainActivity.this, "Unable to sign out: " + e.getMessage(),
                Toast.LENGTH_SHORT).show());
}

@Override
public void initiateVideoMeeting(User user) {
    final boolean isValidToken = user.token != null && !user.token.trim().isEmpty();

    if (!isValidToken) {
        Toast.makeText(this, user.firstName + " " + user.lastName + " is not available for
            meeting", Toast.LENGTH_SHORT).show();
    } else {
        Intent intent = new Intent(getApplicationContext(), OutgoingInvitationActivity.class);
    }
}

```

```

        intent.putExtra("user", user);
        intent.putExtra("type", "video");
        startActivity(intent);
    }
}

@Override
public void initiateAudioMeeting(User user) {
    final boolean isValidToken = user.token != null && !user.token.trim().isEmpty();

    if (!isValidToken) {
        Toast.makeText(this, user.firstName + " " + user.lastName + " is not available for meeting", Toast.LENGTH_SHORT).show();
    } else {
        Intent intent = new Intent(getApplicationContext(), OutgoingInvitationActivity.class);
        intent.putExtra("user", user);
        intent.putExtra("type", "audio");
        startActivity(intent);
    }
}

@Override
public void onMultipleUsersAction(Boolean isMultipleUsersSelected) {
    if (isMultipleUsersSelected) {
        imageConference.setVisibility(View.VISIBLE);
        imageConference.setOnClickListener(v -> {
            Intent intent = new Intent(getApplicationContext(), OutgoingInvitationActivity.class);
            intent.putExtra("selectedUsers", new Gson().toJson(usersAdapter.getSelectedUsers()));
            intent.putExtra("type", "video");
            intent.putExtra("isMultiple", true);
            startActivity(intent);

            System.out.println(usersAdapter.getSelectedUsers().toString());
        });
    } else {
        imageConference.setVisibility(View.GONE);
    }
}

private void checkForBatteryOptimizations() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        PowerManager powerManager = (PowerManager) getSystemService(POWER_SERVICE);
        if (!powerManager.isIgnoringBatteryOptimizations(getPackageName())) {

```



```

        new AlertDialog.Builder(MainActivity.this)
            .setTitle("Warning")
            .setMessage("Battery optimization is enabled. It can interrupt running
background services")
            .setPositiveButton("Disable", (dialog, which) -> {
                Intent intent = new
Intent(Settings.ACTION_IGNORE_BATTERY_OPTIMIZATIONS_SETTINGS);
                startActivityForResult(intent,
REQUEST_CODE_BATTERY_OPTIMIZATIONS);
            })
            .setNegativeButton("Cancel", (dialog, which) -> dialog.dismiss())
            .show();
    }
}
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE_BATTERY_OPTIMIZATIONS) {
        checkForBatteryOptimizations();
    }
}
}
}

```

## API Client config:

```

package com.viv.videomeeting.network;

import retrofit2.Retrofit;
import retrofit2.converter.scalars.ScalarsConverterFactory;

public class ApiClient {

    private static Retrofit retrofit = null;

    public static Retrofit getClient() {
        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                .baseUrl("https://fcm.googleapis.com/fcm/")
                .addConverterFactory(ScalarsConverterFactory.create())
                .build();
        }
        return retrofit;
    }
}

```

```
}  
}
```

## Dependencies:

```
plugins {  
    id 'com.android.application'  
}
```

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

```
android {  
    compileSdkVersion 30
```

```
    defaultConfig {  
        applicationId "com.viv.videomeeting"  
        minSdkVersion 21  
        targetSdkVersion 30  
        versionCode 1  
        versionName "1.0"  
        multiDexEnabled = true
```

```
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
    }
```

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

```
    compileOptions {  
        sourceCompatibility JavaVersion.VERSION_1_8  
        targetCompatibility JavaVersion.VERSION_1_8  
    }  
}
```

```
dependencies {
```

```
    implementation 'androidx.appcompat:appcompat:1.2.0'  
    implementation 'com.google.android.material:material:1.2.1'  
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
```

```
implementation 'com.google.firebase:firebase-iid:+'
testImplementation 'junit:junit:4.13.1'
implementation 'com.google.code.gson:gson:2.8.9'
androidTestImplementation 'androidx.test.ext:junit:1.1.2'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

// Scalable Size Units
implementation 'com.intuit.sdp:sdp-android:1.0.6'
implementation 'com.intuit.ssp:ssp-android:1.0.6'

// RecyclerView
implementation 'androidx.recyclerview:recyclerview:1.1.0'

// Material Design
implementation 'com.google.android.material:material:1.2.1'

// MultiDex
implementation 'com.android.support:multidex:1.0.3'

// Retrofit & Scalars
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-scalars:2.9.0'

// SwipeRefreshLayout
implementation 'androidx.swiperefreshlayout:swiperefreshlayout:1.1.0'

// Import the BoM for the Firebase platform
implementation platform('com.google.firebase:firebase-bom:26.4.0')

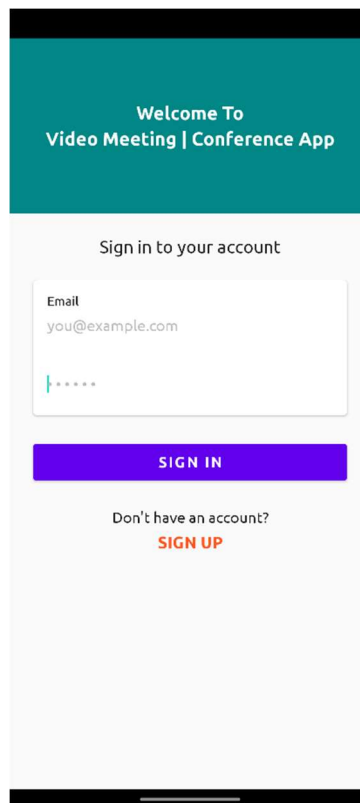
// Declare the dependencies for the Firebase Cloud Messaging and Analytics libraries
// When using the BoM, you don't specify versions in Firebase library dependencies
implementation 'com.google.firebase:firebase-messaging'
implementation 'com.google.firebase:firebase-analytics'

//Firebase
implementation 'com.google.firebase:firebase-messaging:23.0.2'
implementation 'com.google.firebase:firebase-firestore:24.0.2'

// (other dependencies)
implementation ('org.jitsi.react:jitsi-meet-sdk:2.8.2') { transitive = true }
```

# Screen Layouts

## Screen Layouts



Mobile app screen for signing in. The header is teal with the text "Welcome To Video Meeting | Conference App". Below the header, the text "Sign in to your account" is centered. There is a form with two input fields: "Email" with the placeholder "you@example.com" and a password field with masked characters "\*\*\*\*\*". Below the form is a purple "SIGN IN" button. At the bottom, there is a link "Don't have an account?" with a red "SIGN UP" button next to it.

Welcome To  
Video Meeting | Conference App

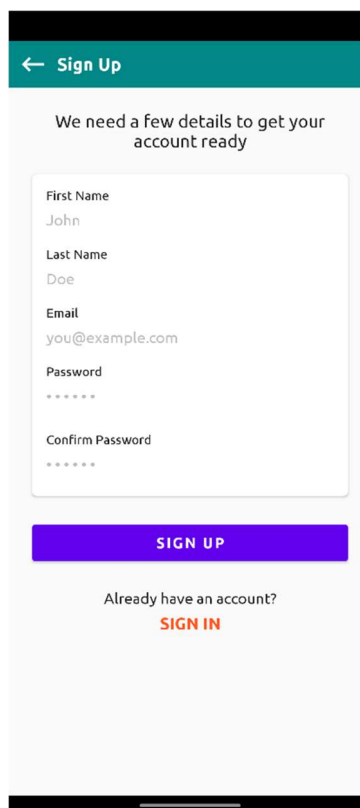
Sign in to your account

Email  
you@example.com

\*\*\*\*\*

SIGN IN

Don't have an account?  
SIGN UP



Mobile app screen for signing up. The header is teal with a back arrow and the text "Sign Up". Below the header, the text "We need a few details to get your account ready" is centered. There is a form with five input fields: "First Name" with the placeholder "John", "Last Name" with the placeholder "Doe", "Email" with the placeholder "you@example.com", "Password" with masked characters "\*\*\*\*\*", and "Confirm Password" with masked characters "\*\*\*\*\*". Below the form is a purple "SIGN UP" button. At the bottom, there is a link "Already have an account?" with a red "SIGN IN" button next to it.

← Sign Up

We need a few details to get your  
account ready

First Name  
John

Last Name  
Doe

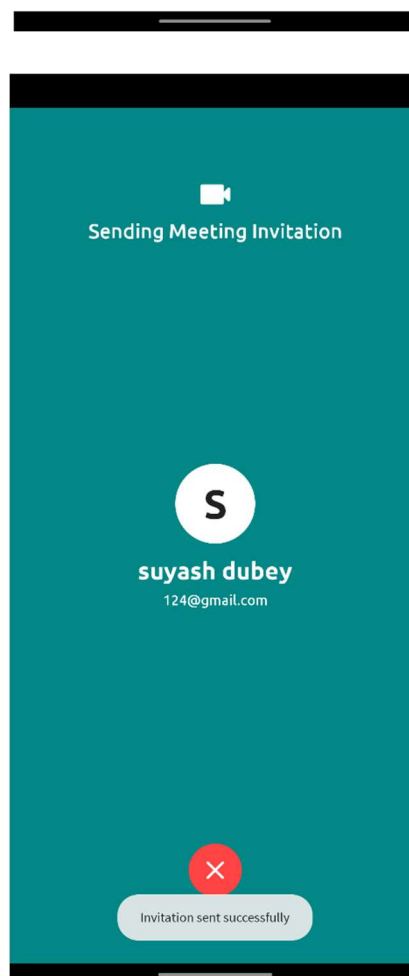
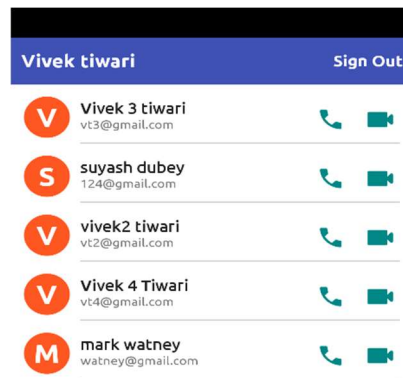
Email  
you@example.com

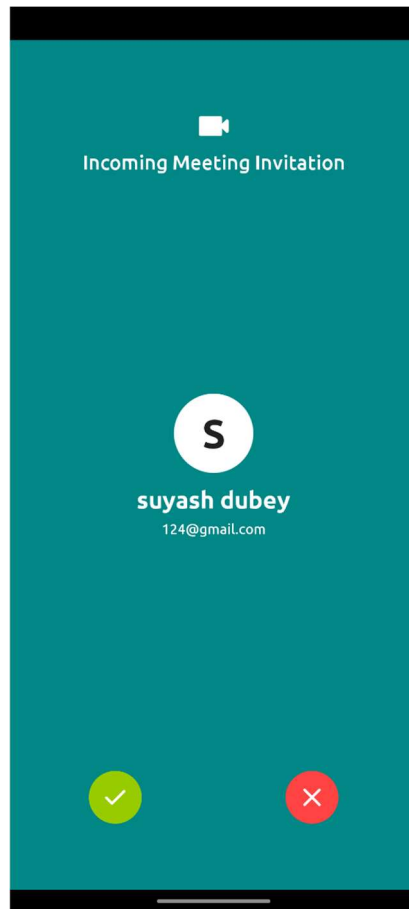
Password  
\*\*\*\*\*

Confirm Password  
\*\*\*\*\*

SIGN UP

Already have an account?  
SIGN IN





### **Future Enhancements:**

Following future enhancements are possible:

- Application could be more optimized.
- Application could've supported older android versions.
- UI/UX can be made even better.
- Multiple user calls can be done in a better way.
- Provisions for adding hosts to control the meet.
- Publishing the app on the Play Store to increase the reach and enabling monetization.



## References

- <https://developer.android.com/docs>
- <https://firebase.google.com/docs>
- <https://stackoverflow.com/>
- <https://github.com/>
- <https://dl.acm.org/>

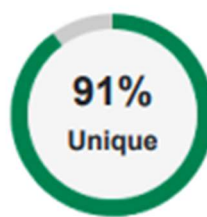
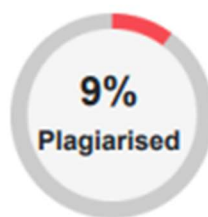
## Plagiarism Report

### Content Report:



Date: March, 24 2022

### PLAGIARISM SCAN REPORT



Excluded Url : None

### Content Checked For Plagiarism

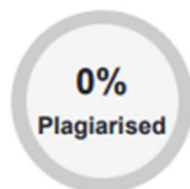
Description of the System • As the times we live are completely virtual, the world as of now depends on these apps. • These apps are now a critical part of our lives. • So necessarily, we should have a variety of options to choose from so as to increase the competition, which will lead to better products for end users. • Quality of life changes to these apps will lead to better user experience overall. Limitations of Present System • The current industry for such apps is dominated by the big tech giants. • The biggest limitation faced by the users is their data and the feature set is controlled by these giants. • So naturally, there's always this dependency on these giants in an integral necessary functionality of our lives. • As users initially consent to EULA, they've little to no control of their data, except the assurances provided by these companies. Proposed System • To provide an open-source option to all the users. • The app developed collects minimal to no data. • As the proposed system is open source in nature, the end users are in control of their data. • Users can enter alias details to use of the application, without worrying about their data. • No EULA, no worries. • A free option amongst paid options. Objectives • This application is developed using the Jitsi Meet SDK, which uses and incorporates the famous WebRTC API which doesn't restrict anyone to use the app incorporated with the SDK. • This allows users to access the meet from a browser of choice. • The main objective being allowing and giving the users the flexibility to use the app or to avoid it. • This helps in building a healthy user base which trusts the development of the app as well as is in control of their data. • Another major objective is to support a wide userbase, which will help the end users to connect seamlessly. • Open-source resources help to create a great dependency backed up by big organization. APIs like WebRTC and SDKs like Jitsi meet are great options for users to integrate in their very own apps if they want to. Minimum hardware requirements needed for development: • x86\_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for virtualisation. • 8 GB RAM or more • 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator) • 1280 x 800 minimum screen resolution. Software Requirements • Operating System: Microsoft Windows 10/11 (64-bit) • IDE: Google Android Studio • Front-End: XML • Back-End: Java • SDKs: Jitsi meet SDK. • Database: Google Firebase. Minimum requirements to run the application: • API Level 27. • 2 GB Ram or Higher Event Table EVENT TRIGGER ACTIVITY RESPONSE DESTINATION User Sign up Initiates user creation User credentials inserted into database. User Created/User Creation Failed Logged In Clicks Sign In Initiates user login Validates user credentials Signing In/Unable to sign in. Signed In Send Meeting Invitation Prompts Outgoing Invitation User Tokens exchanged from database to validate invitation sender and receiver. Invitation Sent/User is not available. Enter meeting room. Incoming Meeting Invitation Prompts Receiving Invitation User Tokens exchanged from database Accept/Deny invitation prompt Join meeting.

## Code Report:



Date: March, 24 2022

## PLAGIARISM SCAN REPORT



Excluded Url : None

## Content Checked For Plagiarism

```
package com.viv.videomeeting.firebase; import android.content.Intent; import android.util.Log; import androidx.annotation.NonNull; import androidx.localbroadcastmanager.content.LocalBroadcastManager; import com.google.firebase.messaging.FirebaseMessagingService; import com.google.firebase.messaging.RemoteMessage; import com.viv.videomeeting.activities.IncomingInvitationActivity; import com.viv.videomeeting.utilities.Constants; public class MessagingService extends FirebaseMessagingService { @Override public void onNewToken(@NonNull String token) { super.onNewToken(token); } @Override public void onMessageReceived(@NonNull RemoteMessage remoteMessage) { super.onMessageReceived(remoteMessage); final String type = remoteMessage.getData().get(Constants.REMOTE_MSG_TYPE); if (type != null) { if (type.equals(Constants.REMOTE_MSG_INVITATION)) { Intent intent = new Intent(getApplicationContext(), IncomingInvitationActivity.class); intent.putExtra(Constants.REMOTE_MSG_MEETING_TYPE, remoteMessage.getData().get(Constants.REMOTE_MSG_MEETING_TYPE)); intent.putExtra(Constants.KEY_FIRST_NAME, remoteMessage.getData().get(Constants.KEY_FIRST_NAME)); intent.putExtra(Constants.KEY_LAST_NAME, remoteMessage.getData().get(Constants.KEY_LAST_NAME)); intent.putExtra(Constants.KEY_EMAIL, remoteMessage.getData().get(Constants.KEY_EMAIL)); intent.putExtra(Constants.REMOTE_MSG_INVITER_TOKEN, remoteMessage.getData().get(Constants.REMOTE_MSG_INVITER_TOKEN)); intent.putExtra(Constants.REMOTE_MSG_MEETING_ROOM, remoteMessage.getData().get(Constants.REMOTE_MSG_MEETING_ROOM)); intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK); startActivity(intent); } else if (type.equals(Constants.REMOTE_MSG_INVITATION_RESPONSE)) { Intent intent = new Intent(Constants.REMOTE_MSG_INVITATION_RESPONSE); intent.putExtra(Constants.REMOTE_MSG_INVITATION_RESPONSE, remoteMessage.getData().get(Constants.REMOTE_MSG_INVITATION_RESPONSE)); LocalBroadcastManager.getInstance(getApplicationContext()).sendBroadcast(intent); } } } Firebase Preferences: package com.viv.videomeeting.utilities; import java.util.HashMap; public class Constants { // Firebase public static final String KEY_COLLECTION_USERS = "users"; public static final String KEY_FIRST_NAME = "first_name"; public static final String KEY_LAST_NAME = "last_name"; public static final String KEY_EMAIL = "email"; public static final String KEY_PASSWORD = "password"; public static final String KEY_USER_ID = "user_id"; public static final String KEY_FCM_TOKEN = "fcm_token"; // Preference public static final String KEY_PREFERENCE_NAME = "VideoMeetingPreference"; public static final String KEY_IS_SIGNED_IN = "isSignedIn"; public static final String REMOTE_MSG_AUTHORIZATION = "Authorization"; public static final String REMOTE_MSG_CONTENT_TYPE = "Content-Type"; public static final String REMOTE_MSG_TYPE = "type"; public static final String REMOTE_MSG_INVITATION = "invitation"; public static final String REMOTE_MSG_MEETING_TYPE = "meetingType"; public static final String REMOTE_MSG_INVITER_TOKEN = "inviterToken"; public static final String REMOTE_MSG_DATA = "data"; public static final String REMOTE_MSG_REGISTRATION_IDS = "registration_ids"; public static final String REMOTE_MSG_INVITATION_RESPONSE = "invitationResponse"; public static final String REMOTE_MSG_INVITATION_ACCEPTED = "accepted"; public static final String REMOTE_MSG_INVITATION_REJECTED = "rejected"; public static final String REMOTE_MSG_INVITATION_CANCELLED = "cancelled"; public static final String REMOTE_MSG_MEETING_ROOM = "meetingRoom"; public static HashMap getRemoteMessageHeaders() { HashMap headers = new HashMap(); headers.put(Constants.REMOTE_MSG_AUTHORIZATION, "key=AAAAAb003ig:APA91bEJIB5Wary91hnpAe-iQfQ_sfdKQOOZ6507-91lIQTe-GlyElzn7pUcFWaqPHY4n7utBxMvKAZsM_frE_wRczFkFv6hVS_07j_LjdyLoZhQGDQg73Bk7zFhDoBj27yHJ0Py3"); headers.put(Constants.REMOTE_MSG_CONTENT_TYPE, "application/json"); return headers; } } Main Activity: package com.viv.videomeeting.activities; import android.content.Intent; import android.os.Build; import android.os.Bundle; import android.os.PowerManager; import android.provider.Settings; import android.view.View; import android.widget.ImageView; import android.widget.TextView; import android.widget.Toast; import androidx.annotation.Nullable; import androidx.appcompat.app.AlertDialog;
```