CHAT CONNECT - A REALTIME CHAT AND COMMUNICATION APP

Akshara K M -715522243005
Bhavadharani S -715522243011
Supriya R -715522243052
Yuvashree S -715522243061

OBJECTIVE

The objective of the ChatConnect project is to develop a real-time chat application using the modern Android Jetpack Compose UI toolkit, showcasing an efficient, user-friendly, and responsive messaging interface. This project aims to leverage Compose's declarative UI approach to simplify and enhance the development of interactive components, allowing users to easily send and receive text messages in real-time. Additionally, Chat Connect demonstrates state management within Compose and explores the seamless integration of Firebase for real-time data synchronization, user authentication, and data storage. Through this project, developers will gain insight into Compose’s input handling, navigation, and Firebase-backed data population, creating a foundation for scalable, interactive Android communication applications.

DESCRIPTION

- **Real-Time Messaging:** Facilitates instant text communication between individuals or groups. Supports multimedia sharing, including images, videos, and documents.
- **Voice and Video Calls**: High-quality voice and video calling options. Group call support for team meetings or social gatherings.
- **Multi-Device Compatibility**: Synchronizes messages and data across multiple devices (smartphones, tablets, desktops). Ensures accessibility no matter where you are.
- **Enhanced Privacy and Security:** End-to-end encryption to protect user data and conversations. Customizable privacy settings to control visibility and accessibility.
- **Customizable User Profiles:** Allows users to personalize profiles with photos, status updates, and contact details. Includes business-specific fields for professional use.
- **Group and Community Features** Tools for creating and managing groups with various purposes (friends, family, work). Community forums and channels for large-scale discussions or announcements.
- **Notifications and Updates:** Real-time notifications for messages, calls, and events. Option to customize notification preferences for uninterrupted focus.

ComponentLaunchActivity.kt

```
package com.cometchat.kotlinsampleapp.activity

import android.os.Bundle
import android.widget.LinearLayout
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import com.cometchat.chatuikit.shared.resources.utils.Utils
import com.cometchat.kotlinsampleapp.AppUtils
import com.cometchat.kotlinsampleapp.R
import com.cometchat.kotlinsampleapp.databinding.ActivityComponentLaunchBinding
import com.cometchat.kotlinsampleapp.fragments.calls.CallButtonFragment
import com.cometchat.kotlinsampleapp.fragments.calls.CallLogDetailsFragment
import com.cometchat.kotlinsampleapp.fragments.calls.CallLogHistoryFragment
import com.cometchat.kotlinsampleapp.fragments.calls.CallLogParticipantsFragment
import com.cometchat.kotlinsampleapp.fragments.calls.CallLogRecordingFragment
import com.cometchat.kotlinsampleapp.fragments.calls.CallLogWithDetailsFragment
import com.cometchat.kotlinsampleapp.fragments.calls.CallLogsFragment
```

Application.kt

```
package com.cometchat.kotlinsampleapp
import android.app.Application
import android.content.res.Configuration
import com.cometchat.chat.core.Call
import com.cometchat.chat.core.CometChat
import com.cometchat.chatuikit.calls.CometChatCallActivity
import com.cometchat.chatuikit.shared.resources.theme.CometChatTheme
import com.cometchat.chatuikit.shared.resources.theme.Palette
import com.cometchat.kotlinsampleapp.AppUtils.Companion.isNightMode
class Application : Application() {
    override fun onCreate() {
        super.onCreate()
        addCallListener()
        if (isNightMode(this)) {
            Palette.getInstance().mode(CometChatTheme.MODE.DARK)
        }
    }
    override fun onConfigurationChanged(newConfig: Configuration) {
        super.onConfigurationChanged(newConfig)
        if (isNightMode(this)) {
            Palette.getInstance().mode(CometChatTheme.MODE.DARK)
        } else {
            Palette.getInstance().mode(CometChatTheme.MODE.LIGHT)
        }
    }
    private fun addCallListener() {
        val LISTENER_ID = System.currentTimeMillis().toString() + ""
        CometChat.addCallListener(LISTENER_ID, object : CometChat.CallListener() {
            override fun onIncomingCallReceived(call: Call) {
                CometChatCallActivity.launchIncomingCallScreen(applicationContext, call, null)
            }

            override fun onOutgoingCallAccepted(call: Call?) {}
            override fun onOutgoingCallRejected(call: Call?) {}
            override fun onIncomingCallCancelled(call: Call?) {}
        })
    }

}
```

AppUtils.kt

```
class AppUtils {
    companion object {
        private var group: Group? = null
        private var user: User? = null
        private var userList: MutableList<User> = ArrayList()
        fun fetchDefaultObjects() {
            CometChat.getGroup("supergroup", object :
CometChat.CallbackListener<Group?>() {
                override fun onSuccess(group_: Group?) {
                    group = group_
                }

                override fun onError(e: CometChatException?) {}
            })
            CometChat.getUser("superhero5", object :
CometChat.CallbackListener<User?>() {
                override fun onSuccess(user_: User?) {
                    user = user_
                }

                override fun onError(e: CometChatException?) {}
            })
        }

        fun fetchSampleUsers(listener:
CometChat.CallbackListener<List<User>>) {
            if (userList.isEmpty()) {
                val request: Request =

Request.Builder().url(StringConstants.SAMPLE_APP_USERS_URL)
                        .method("GET", null).build()
                val client = OkHttpClient()
                client.newCall(request).enqueue(object : Callback {
                    override fun onFailure(call: Call, e: IOException) {
                        Utils.runOnMainThread {
                            listener.onError(
                                CometChatException("11", e.message)
                            )
                        }
                    }

                    override fun onResponse(call: Call, response: Response) {
                        if (response.isSuccessful && response.body != null) {
                            try {
                                userList =
processSampleUserList(response.body!!.string())
                            } catch (e: IOException) {
                                Utils.runOnMainThread {
                                    listener.onError(
                                        CometChatException("10", e.message)
                                    )
                                }

                            }
                        } else {
                            Utils.runOnMainThread {
                                listener.onError(
                                    CometChatException(
                                        "Unexpected code ",
                                        response.code.toString()
        }}}}
```

```
        fun loadJSONFromAsset(context: Context): String? {
            var json: String? = null
            try {
                val `is` = context.assets.open("SampleUsers.json")
                val size = `is`.available()
                val buffer = ByteArray(size)
                `is`.read(buffer)
                `is`.close()
                json = String(buffer, charset("UTF-8"))
            } catch (ex: IOException) {
                ex.printStackTrace()
                return null
            }
            return json
        }
        val defaultUserList: List<User?>
            get() = userList

        val defaultGroup: Group?
            get() = group

        val defaultUser: User?
            get() = user

        fun switchLightMode() {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO)
        }

        fun switchDarkMode() {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES)
        }

        fun isNightMode(context: Context): Boolean {
            val currentNightMode =
                context.resources.configuration.uiMode and Configuration.UI_MODE_NIGHT_MASK
            return currentNightMode == Configuration.UI_MODE_NIGHT_YES
        }

        fun changeIconTintToWhite(context: Context?, imageView: ImageView) {
            imageView.imageTintList = ColorStateList.valueOf(
                ContextCompat.getColor(
                    context!!,
                    R.color.white
                )
            )
        }

        fun changeIconTintToBlack(context: Context?, imageView: ImageView) {
            imageView.imageTintList = ColorStateList.valueOf(
                ContextCompat.getColor(
                    context!!,
                    R.color.black
                )
            )
        }

        fun changeTextColorToWhite(context: Context?, textView: TextView) {
            textView.setTextColor(ContextCompat.getColor(context!!, R.color.white))
        }

        fun changeTextColorToBlack(context: Context?, textView: TextView) {
            textView.setTextColor(ContextCompat.getColor(context!!, R.color.black))
        }
    }
}
```

MessageComposer.kt

```
class MessageComposerFragment : Fragment() {
    private var parentView: RelativeLayout? = null
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        // Inflate the layout for this fragment
        val view: View = inflater.inflate(R.layout.fragment_message_composer, container, false)
        parentView = view.findViewById(R.id.parent_view)
        requireActivity().window.setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_ADJUST_RESIZE)
        val cometChatMessageComposer = view.findViewById<CometChatMessageComposer>(R.id.composer)
        cometChatMessageComposer.user = defaultUser
        setUpUI(view)
        return view
    }

    private fun setUpUI(view: View) {
        if (isNightMode(requireContext())) {
            parentView!!.setBackgroundColor(
                    ContextCompat.getColor(
                        requireContext(), R.color.app_background_dark
                    )
            )
        } else {
            parentView!!.setBackgroundColor(ContextCompat.getColor(requireContext(), R.color.app_background))
        }
    }
}
```

GroupMembersFraagment.kt

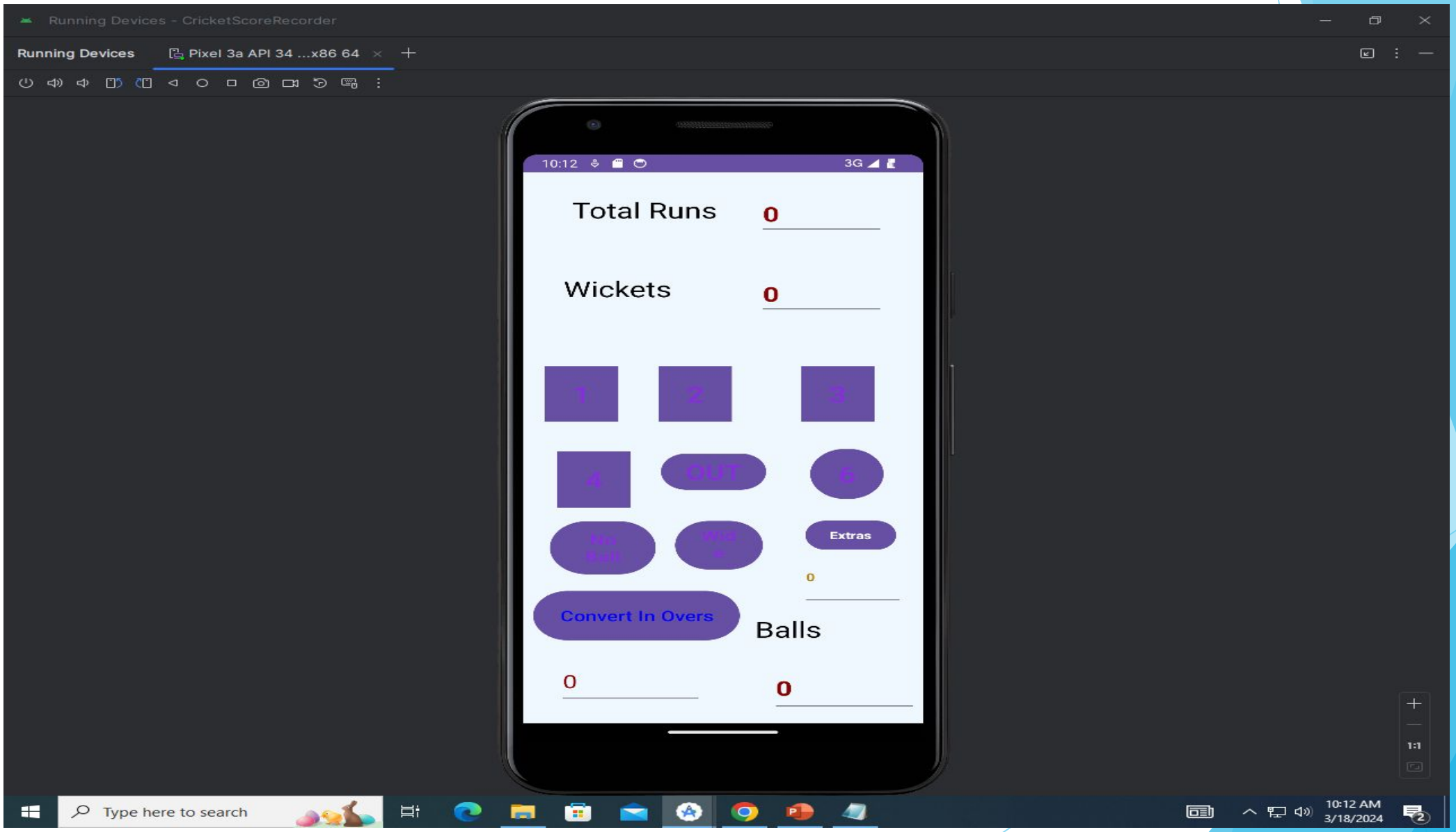
```
package com.cometchat.kotlinsampleapp.fragments.groups
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.cometchat.chatuikit.groupmembers.CometChatGroupMembers
import com.cometchat.kotlinsampleapp.AppUtils.Companion.defaultGroup
import com.cometchat.kotlinsampleapp.R

class GroupMembersFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        val view: View = inflater.inflate(R.layout.fragment_group_members, container, false)
        val groupMembers = view.findViewById<CometChatGroupMembers>(R.id.members)
        groupMembers.setGroup(defaultGroup)
        return view
    }
}
```

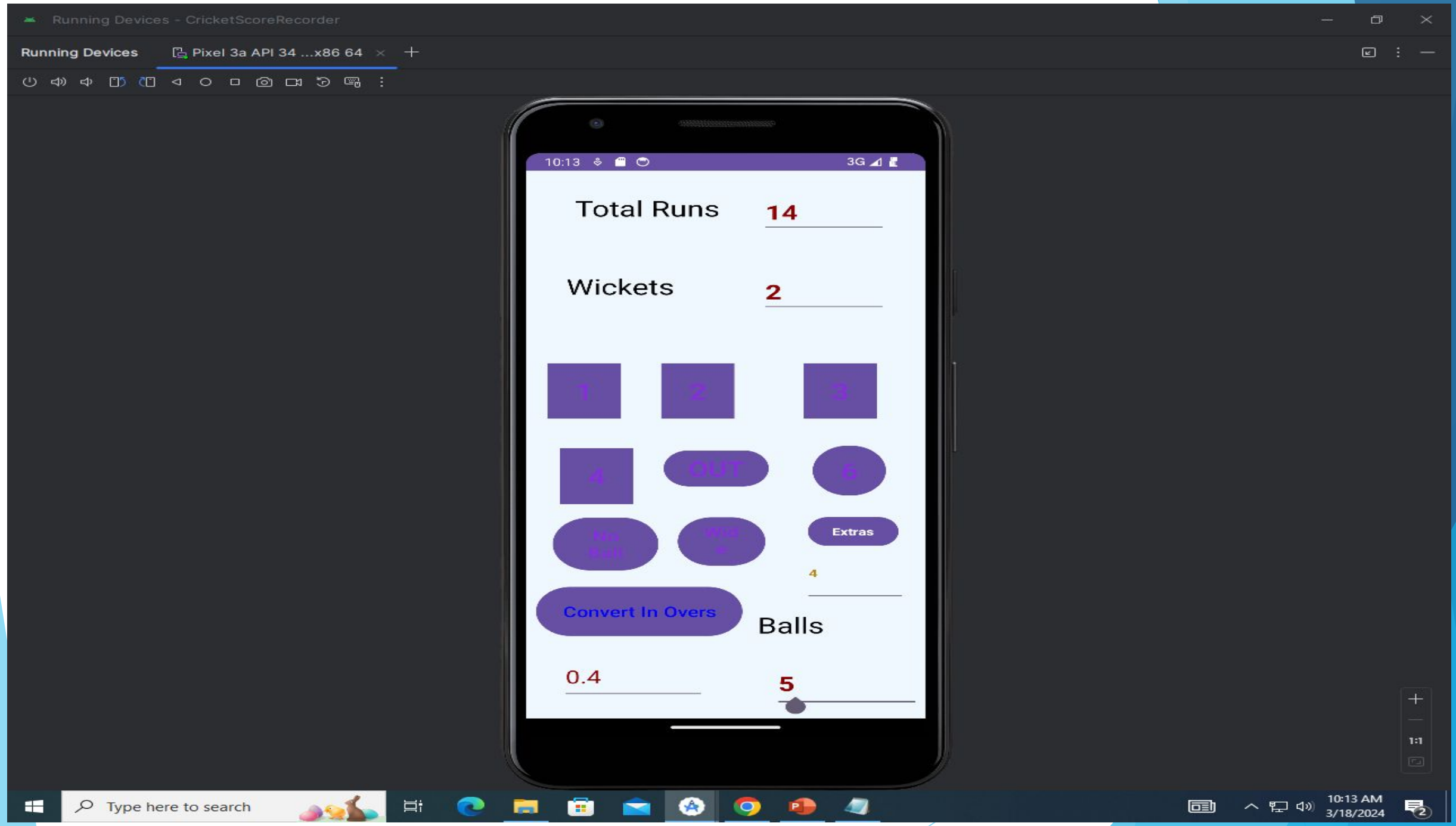
CallLogParticipantsFragment.kt

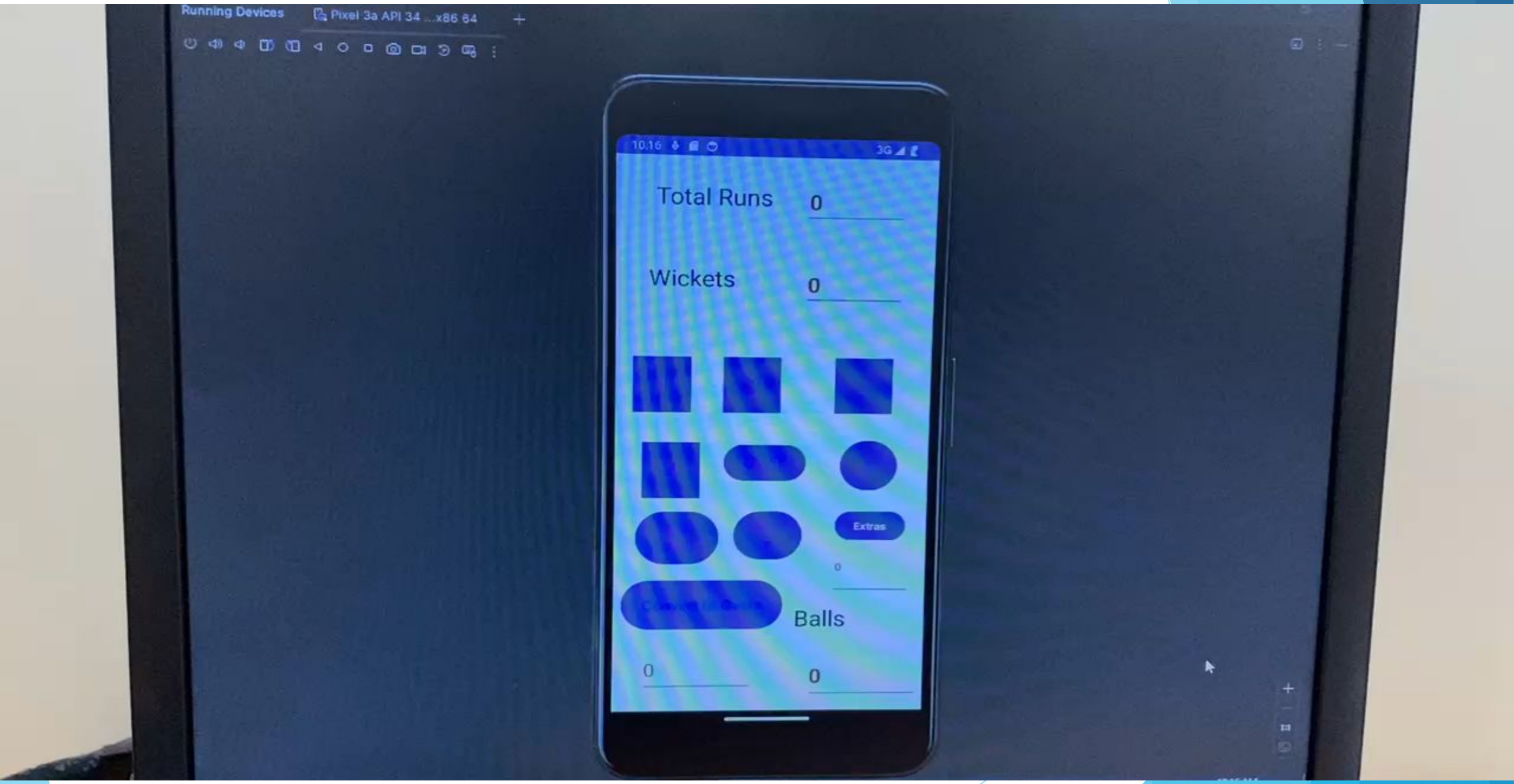
```
class CallLogParticipantsFragment : Fragment() {
    private var cometChatCallLogParticipants: CometChatCallLogParticipants? = null
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        // Inflate the layout for this fragment
        val view: View = inflater.inflate(R.layout.fragment_call_log_participants, container, false)
        cometChatCallLogParticipants =
            view.findViewById(R.id.call_logs_participants)
        createCallParticipants()
        return view}
    private fun createCallParticipants() {
        val participants: MutableList<Participant> = ArrayList()
        val participant = Participant()
        participant.uid = CometChatUIKit.getLoggedInUser().uid
        participant.name = CometChatUIKit.getLoggedInUser().name
        participant.avatar = CometChatUIKit.getLoggedInUser().avatar
        participant.setJoinedAt(1327349241)
        participant.isHasJoined = true
        participant.totalDurationInMinutes = 1327349241.0
        val participant1 = Participant()
        participant1.uid = "UID233"
        participant1.name = "Kevin"
        participant1.avatar = "https://data-us.cometchat.io/assets/images/avatars/spiderman.png"
        participant1.setJoinedAt(1327349241)
        participant1.isHasJoined = true
        participant1.totalDurationInMinutes = 1327349241.0
        participants.add(participant)
        participants.add(participant1)
        cometChatCallLogParticipants!!.setParticipantList(participants) }}
```

THANK YOU!

OUTPUT
:


Running Devices - CricketScoreRecorder
Running Devices
Pixel 3a API 34 ...x86 64
10:12
3G
Total Runs
0
Wickets
0
1
2
3
4
OUT
6
Extras
0
Convert In Overs
Balls
0
0
Type here to search
10:12 AM
3/18/2024


Running Devices - CricketScoreRecorder
Running Devices
Pixel 3a API 34 ...x86 64
10:13
3G
Total Runs
14
Wickets
2
1
2
3
4
OUT
6
No Ball
Wide
Extras
4
Convert In Overs
Balls
0.4
5
Type here to search
10:13 AM
3/18/2024


Running Devices
Pixel 3a API 34 ...x86 64
10.16
3G
Total Runs
0
Wickets
0
Extras
0
Balls
0
0