COLLABORATIVE DOODLING

Software Engineering Project

Jasmine A. Sangma (B16CS002)

*Haphisabet Chyne (*B16CS007*)*

Laribok Syiemlieh (B16CS023)

Collaborative Doodling

TABLE OF CONTENTS

ABSTRACT

Collaborative Doodling Android Application

Haphisabet Chyne , *Jasmine A. Sangma* and *Laribok Syiemlieh*, **National Institute of Technology, Meghalaya**

Collaborative Doodling connects people by giving them the ability to share their thoughts and ideas with others and see their contributions in real time, regardless of where they are geographically. This application is perfect for collaborating on designs in any professional area or just making masterpieces with your friends.

The user of this software application can either create or enter an existing room and collaborate with every person in that same room. They can use different colours to illustrate their design and save their work to view it later.

TEAM MEMBERS

- ***Jasmine A. Sangma*** (B16CS002)
- ***Haphisabet Chyne*** (B16CS007)
- ***Laribok Syiemlieh*** (B16CS023)

INTRODUCTION

Collaborative art can be defined simply as artwork that involves working as a team to create art, and each person contributes in some significant way to the artwork.

The internet has provided a platform for people to express their views and ideas. Thus, we can use the internet to provide a way for artists or any person to share their ideas and art in real time.

The goal of Collaborative Doodling is to allow people to exchange ideas through their drawings without the need of being in the same location but give them the experience as though they are working together in a room, in one canvas.

MOTIVATION

Collaborating drawings is generally bounded by geography. People have to be in the same place in order to collaborate. It may also be the case that people from different locations would want to discuss on their ideas and would want to illustrate them as though they are in the same meeting room.

Collaborative Doodling allows people to overcome the limitation of being in the same room to be able to share, collaborate and show their work. Thus, people from any part of the globe can work together to draw and make designs for any task. It also provides a platform for artists from different locations to come together to create art.

MAIN CONTRIBUTIONS

- Git Version Control
- GitHub
- Firebase Real-time Database
- Firebase Authentication
- Firebase AuthUI
- Lottie Animations

FEASIBILITY ANALYSIS

Technical Feasibility

This software will not require high end infrastructure. Software is developed using Kotlin and Android Studio for IDE. Kotlin code is executed on the JVM. User interface has been developed using xml. We have used Real-time Database to get Real-time data and Authentication services of Firebase to sign-in into the software application.

Financial Feasibility

For any system if the expected benefits equal or exceed the expected costs, the system can be judged to be financially feasible. No financial aid will be required for developing the software apart from using Google Developer Account for publishing the software application to the Google Play Store Worldwide. Our software code will be written in Open source IDE.

Time Feasibility

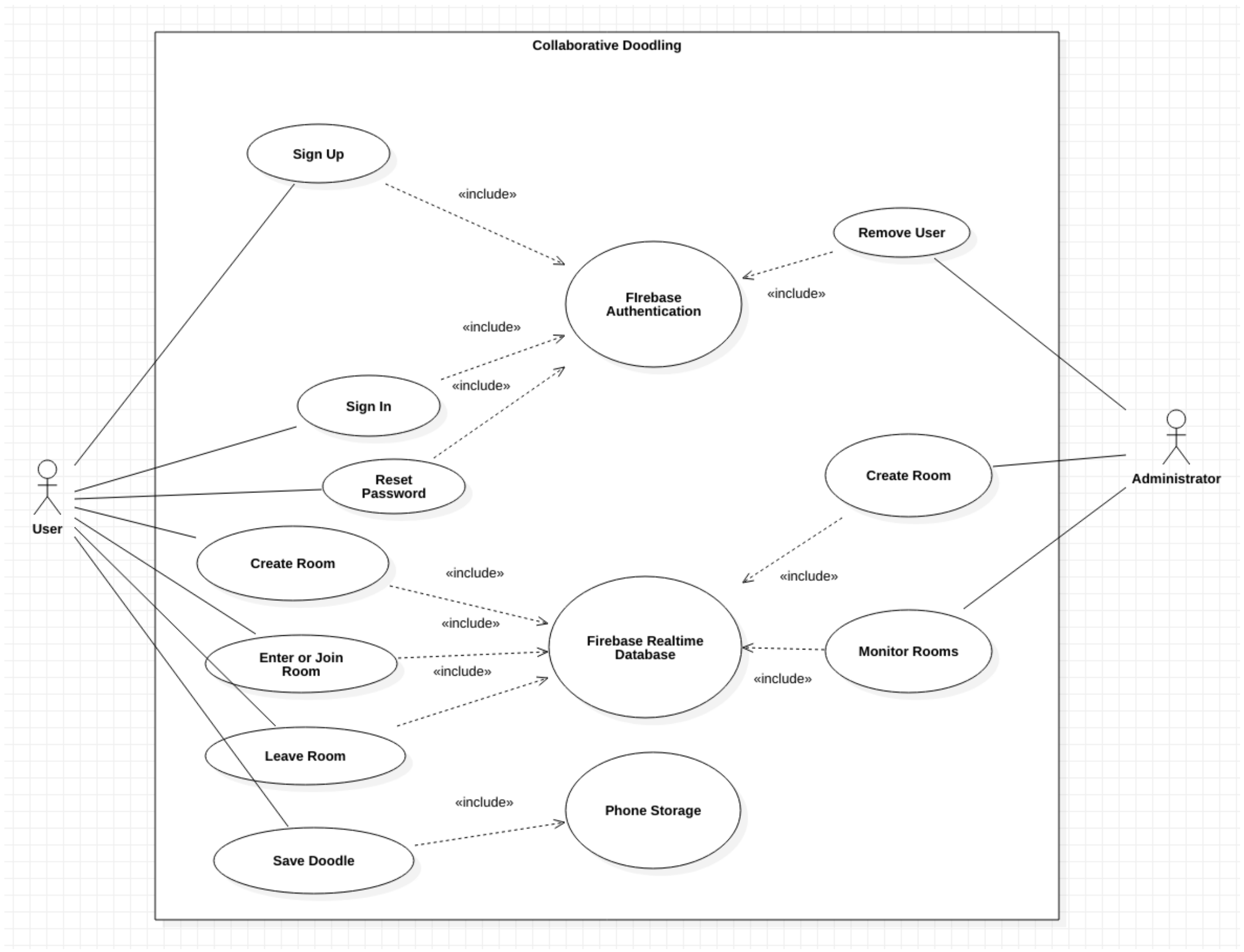
The project was divided into three phases/modules each requiring a number of days keeping in consideration proper development of the software with basic functionalities being incorporated. Hence the basic design and implementation of the project was covered in about two months. Further development will go along the process.

Operational Feasibility

Operational feasibility is mainly concerned with issues like whether the software application will be used if it is developed and implemented. The working of the application is very simple. Special emphasis has been laid on the user-friendliness of the application.

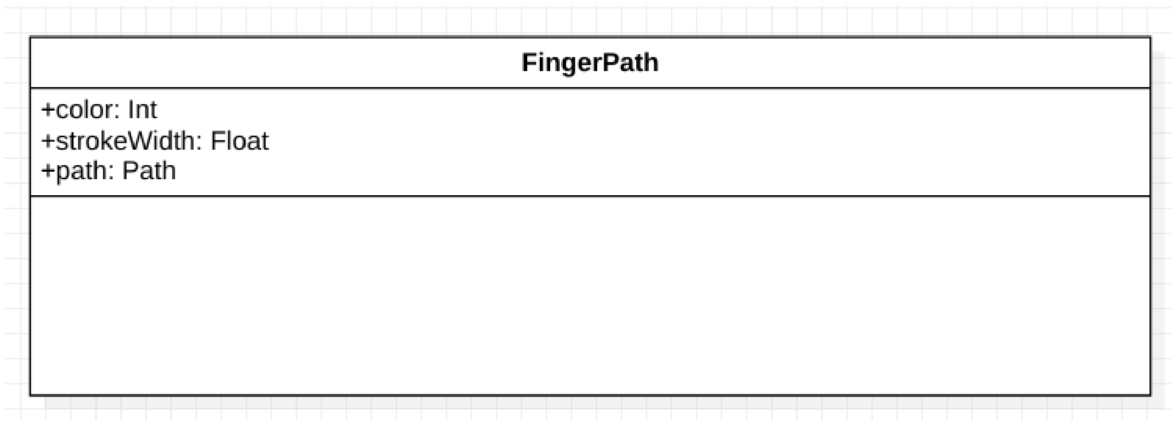
UML DIAGRAMS

Use-Case Diagram

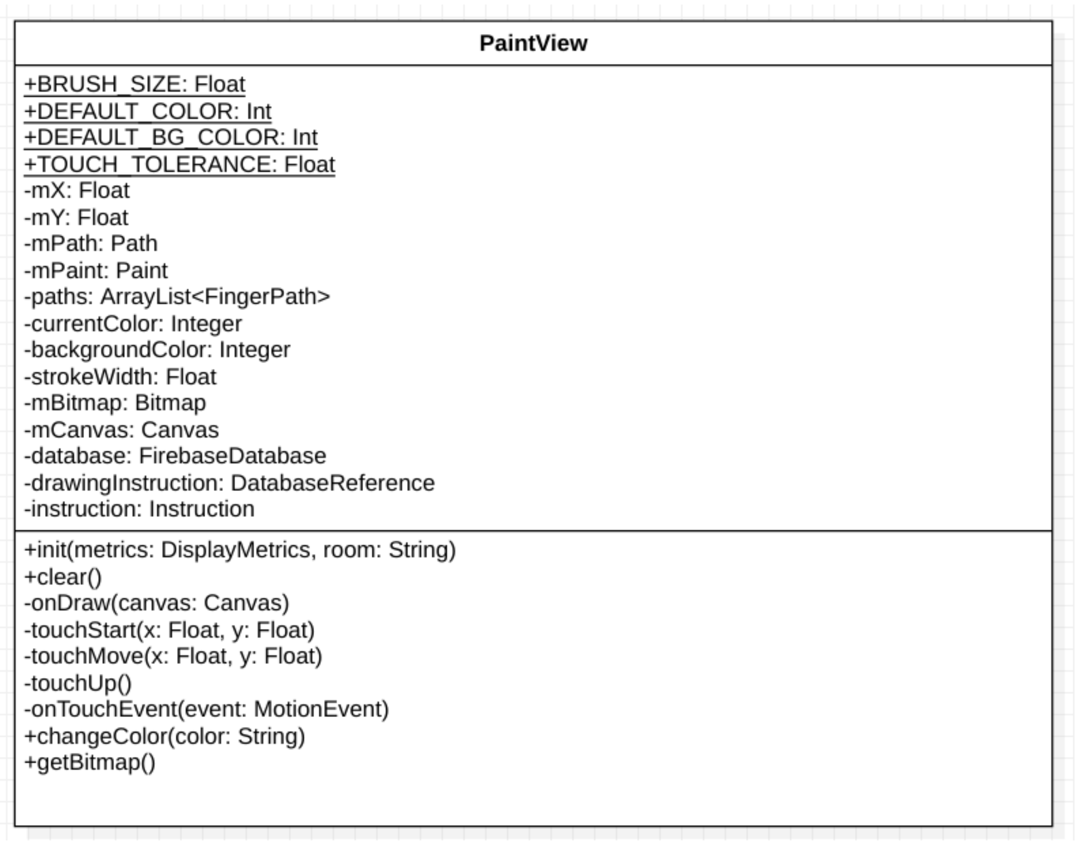


Class Diagrams

FingerPath class



PaintView class

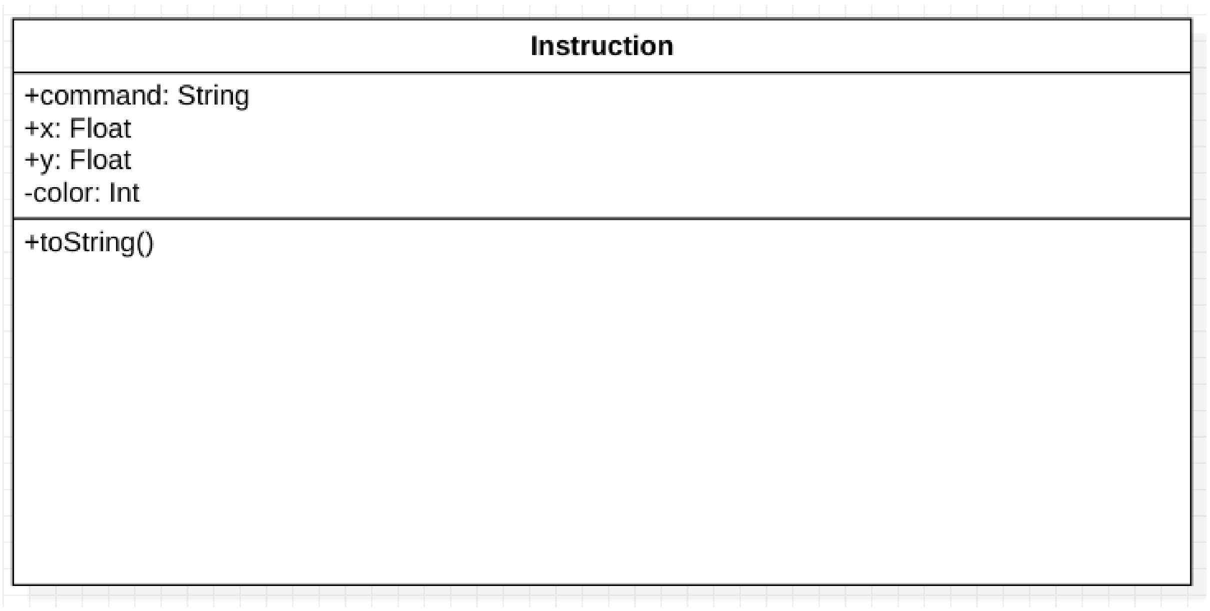


PaintView

```
+BRUSH_SIZE: Float
+DEFAULT_COLOR: Int
+DEFAULT_BG_COLOR: Int
+TOUCH_TOLERANCE: Float
-mX: Float
-mY: Float
-mPath: Path
-mPaint: Paint
-paths: ArrayList<FingerPath>
-currentColor: Integer
-backgroundColor: Integer
-strokeWidth: Float
-mBitmap: Bitmap
-mCanvas: Canvas
-database: FirebaseDatabase
-drawingInstruction: DatabaseReference
-instruction: Instruction
```

```
+init(metrics: DisplayMetrics, room: String)
+clear()
-onDraw(canvas: Canvas)
-touchStart(x: Float, y: Float)
-touchMove(x: Float, y: Float)
-touchUp()
-onTouchEvent(event: MotionEvent)
+changeColor(color: String)
+getBitmap()
```

Instruction class

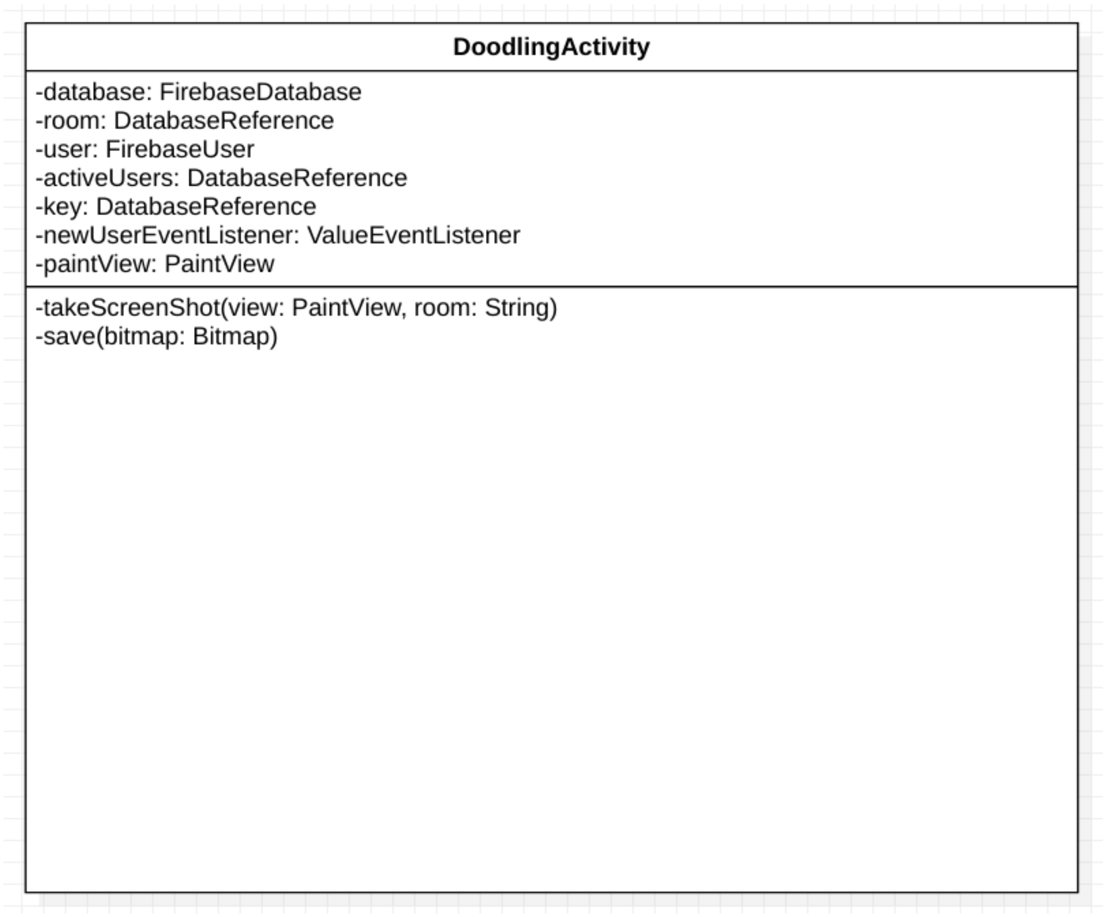


Instruction

```
+command: String
+x: Float
+y: Float
-color: Int
```

```
+toString()
```

DoodlingActivity class



DoodlingActivity
-database: FirebaseDatabase
-room: DatabaseReference
-user: FirebaseUser
-activeUsers: DatabaseReference
-key: DatabaseReference
-newUserEventListener: ValueEventListener
-paintView: PaintView
-takeScreenShot(view: PaintView, room: String)
-save(bitmap: Bitmap)

IMPLEMENTATION DETAILS

The implementation of the application can be explained in two parts

- Front End
- Back end

The front end is the visible area of the application where the user interacts and back end containing all of the code that drives the application.

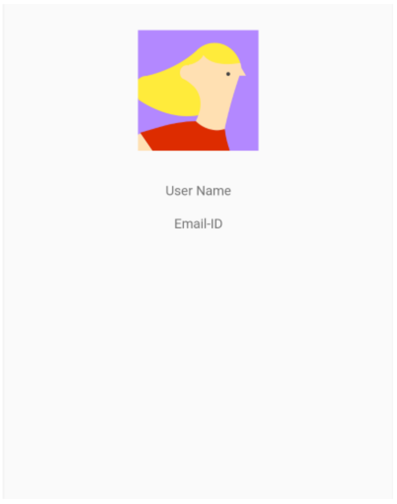
Front End

XML

The front end gives an easy and user friendly interface to users which is done with the help of **XML (eXtensible Markup Language)** in Android Studio 3.4. All the UI and layout of our app is designed using XML. Unlike Kotlin (which is Back Bone of your app), XML helps you to design our app , how it will look , how components like buttons , textview , etc will be placed and their styling.

Example XML code of Profile Layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activity.ProfileActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:srcCompat="@tools:sample/avatars"
        android:id="@+id/profilePic"
        android:layout_marginTop="32dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@drawable/fui_ic_googleg_color_24dp"/>

    <TextView
        tools:text="User Name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/nameTV" android:layout_marginTop="32dp"

        app:layout_constraintTop_toBottomOf="@+id/profilePic"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"/>
    <TextView
        tools:text="Email-ID"
        android:layout_width="wrap_content"
        android:layout_height="19dp"
        android:id="@+id/emailTV"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        android:layout_marginTop="16dp"
        app:layout_constraintTop_toBottomOf="@+id/nameTV"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```
and the corresponding Profile Layout when viewed by a user



Back End

Kotlin

Kotlin is a programming language introduced by JetBrains, the official designer of the most intelligent Java IDE, named Intellij IDEA. Kotlin is a strongly statically typed language that runs on JVM. In 2017, Google announced Kotlin is an official language for android development. Kotlin is an open source programming language that combines object-oriented programming and functional features into a unique platform.

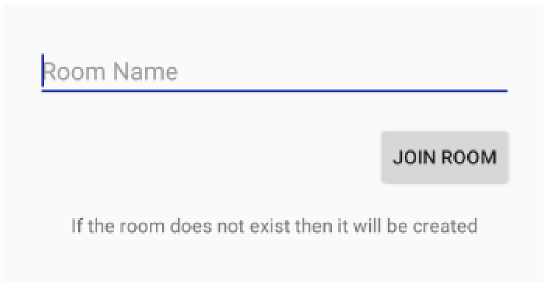
In every screen or activity, users interact in the activity area. Activities extended from the base Activity class. Each activity associated with an XML layout file which supplies the leading back end visuals.

```
class ProfileActivity : AppCompatActivity()
{
    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_profile)
    }
}
```

Events are produced, started again, and closed through the Android Operating System through many callback methods, which are called instantly through the OS on the appropriate occasions. A few examples of callback methods: onCreate, onDestroy, onResume, onStop. These techniques need to be overwritten for the activity to operate properly.

```
saveBtn.setOnClickListener
{
    val newbitmap = takeScreenShot(paintView)
    save(newbitmap!!)
}
override fun onCreate(savedInstanceState: Bundle?)
{
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_doodling)
}

override fun onDestroy() {
    super.onDestroy()
    activeUsers.removeEventListener(newUserEventListener)
    key.removeValue()
}
```

Information sent between activities is known as intent. The intent's carry standard data strings modified to hold extra data. For instance, clicking Join Room Button adds the text in Room Name EditText in the intent as an Extra string and transmits it to the Doodling activity where it creates a room under that room name entered in the Database for that user.



If the room does not exist then it will be created

```
joinRoomBtn.setOnClickListener
{
    val intent = Intent(this, DoodlingActivity::class.java)
    intent.putExtra(DoodlingActivity.ROOM_NAME, roomNameET.text.toString())
    startActivity(intent)
}
```

Firebase

In our Application we have used **Realtime Database** and **Authentication** services of Firebase.

Firebase is a mobile and web app development platform that provides developers with a plethora of tools and services to help them develop high-quality apps, grow their user base, and earn more profit.

Firebase Services

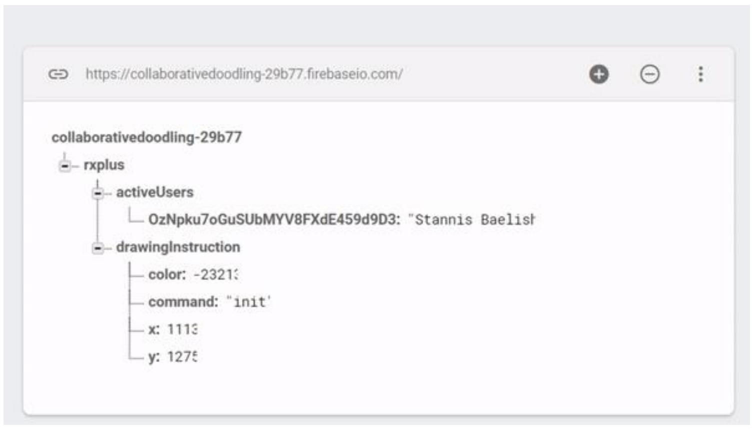
Firebase Services can be divided into two groups:

- Develop & test our app
    - Realtime Database
    - Authentication
    - Test Lab
    - Crashlytics
    - Cloud Functions
    - Firestore
    - Cloud Storage
    - Performance Monitoring
    - Crash Reporting
    - Hosting
- Grow & Engage your audience
    - Firebase Analytics
    - Invites
    - Cloud Messaging
    - Predictions
    - AdMob
    - Dynamic Links
    - Adwords
    - Remote Config
    - App Indexing

Realtime Database

The Firebase Real-time Database is a cloud-hosted NoSQL database that lets us store and sync between our users in real-time. The Real-time Database is really just one big JSON object that the developers can manage in real-time.

Our Realtime Database in action



```
https://collaborativedoodling-29b77.firebaseio.com/

collaborativedoodling-29b77
 - rxplus
    - activeUsers
       └ OzNpku7oGuSUbMYV8FXdE459d9D3: "Stannis Baelist
    - drawingInstruction
       ├ color: -2321
       ├ command: "init'
       ├ x: 1113
       └ y: 1275
```

With just a single API, the Firebase database provides your app with both the current value of the data and any updates to that data.

```
database = FirebaseDatabase.getInstance()
drawingInstruction = database!!.getReference(room).child("drawingInstruction")

drawingInstruction!!.addValueEventListener(object : ValueEventListener {
    override fun onCancelled(error: DatabaseError) {
        Log.w("error", "Failed to read value.", error.toException())
    }

    override fun onDataChange(dataSnapshot: DataSnapshot) {
        val value = dataSnapshot.getValue(Instruction::class.java)
        Log.d("value", value.toString())

        when (value!!.command) {
            "init" -> {
            }
            "touchStart" -> {
                touchStart(value.x!!, value.y!!)
                invalidate()

            }
            "touchMove" -> {
                touchMove(value.x!!, value.y!!)
                invalidate()

            }
            "touchUp" -> {
                touchUp()
                invalidate()
                instruction.command = "init"
                drawingInstruction!!.setValue(instruction)
            }
            "changeColor" -> {
                currentColor = value!!.color
            }
        }
    }
})
```

Realtime syncing makes it easy for our users to access their data from any device, be it web or mobile. Realtime Database also helps our users collaborate with one another.

Authentication

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to our app.

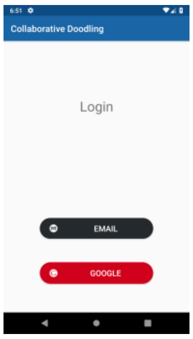
Normally, it would take you months to set up our own authentication system. And even after that, we would need to keep a dedicated team to maintain that system. But if we use Firebase, we can set up the entire system in under 10 lines of code that will handle everything for us, including complex operations like account merging.

```
private fun createSignInIntent() {
    val providers = arrayListOf(
        AuthUI.IdpConfig.EmailBuilder().build(),
        AuthUI.IdpConfig.GoogleBuilder().build()
    )

    val customLayout = AuthMethodPickerLayout.Builder(R.layout.custom_login)
        .setGoogleButtonId(R.id.googleBtn)
        .setEmailButtonId(R.id.emailBtn)
        .build()

    startActivityForResult(
        AuthUI.getInstance()
            .createSignInIntentBuilder()
            .setAvailableProviders(providers)
            .setAuthMethodPickerLayout(customLayout)
            .build(),
        RC_SIGN_IN
    )
}
```

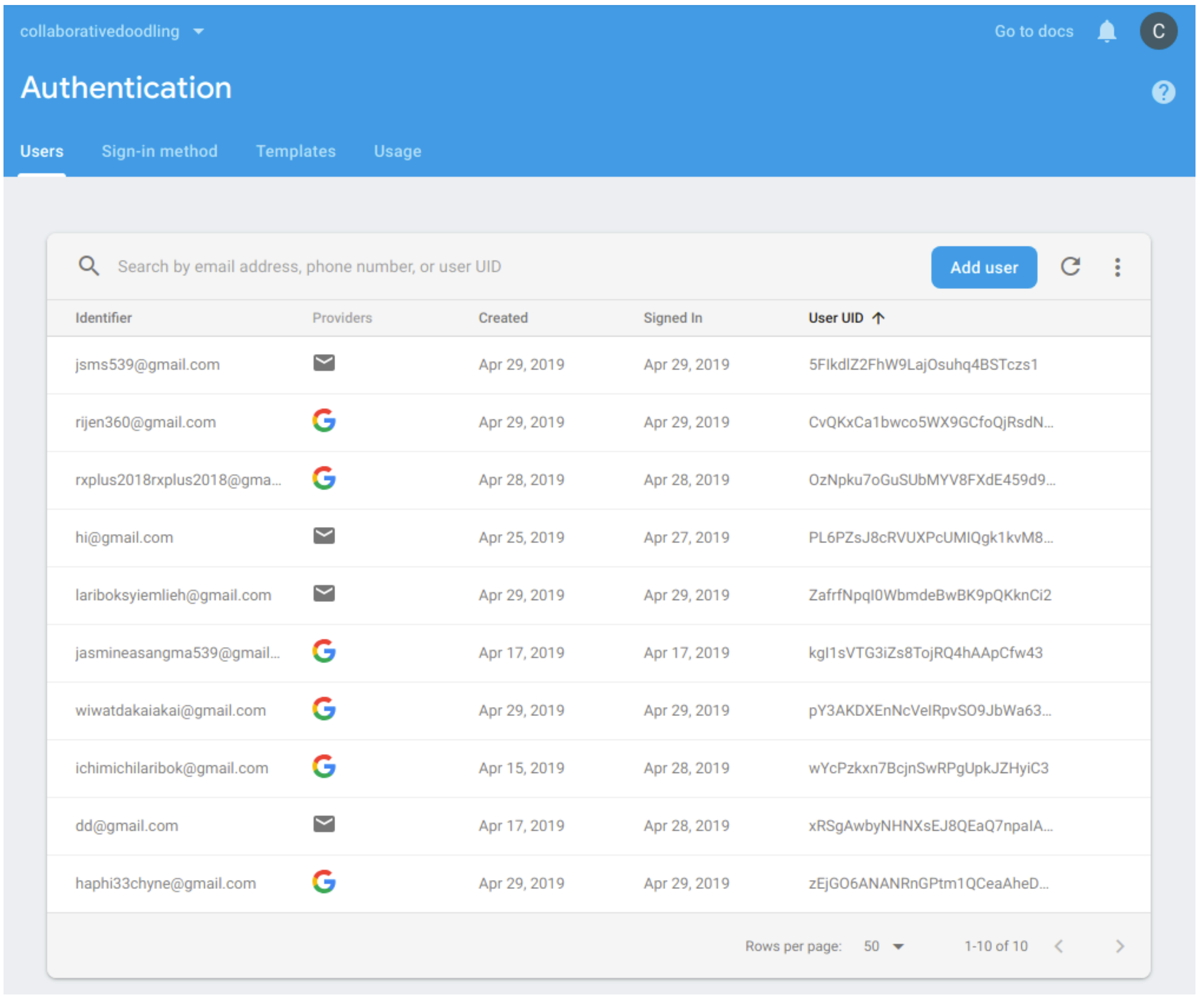
We have authenticate our application's users through the following methods:

- Email & Password
- Google

Using Firebase Authentication makes building secure authentication systems easier, while also improving the sign-in and onboarding experience for end users.



List of users signed up in our application



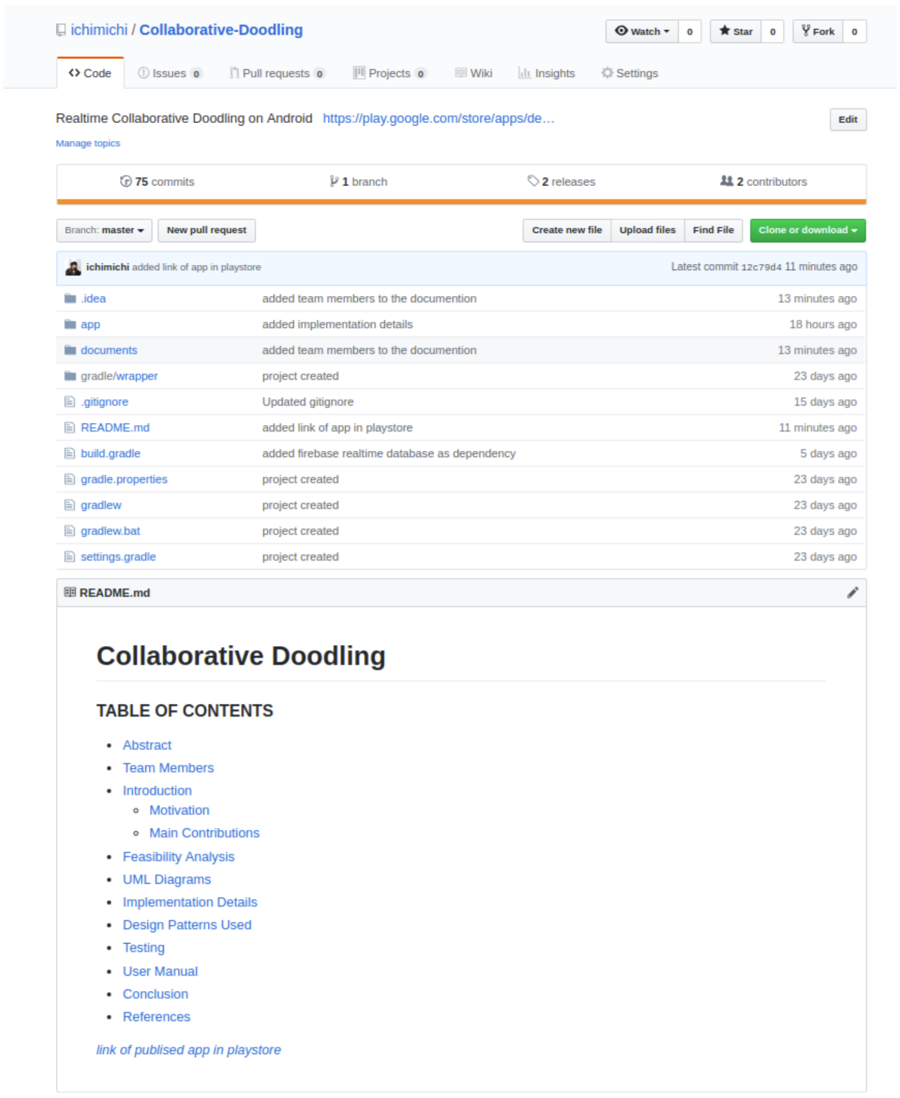
Identifier	Providers	Created	Signed In	User UID ↑
jsms539@gmail.com	✉	Apr 29, 2019	Apr 29, 2019	5FIkdlZ2FhW9LajOsuhq4BSTczs1
rijen360@gmail.com	G	Apr 29, 2019	Apr 29, 2019	CvQKxCa1bwco5WX9GCfoQjRsdN...
rxplus2018rxplus2018@gma...	G	Apr 28, 2019	Apr 28, 2019	OzNpku7oGuSUbMYV8FXdE459d9...
hi@gmail.com	✉	Apr 25, 2019	Apr 27, 2019	PL6PZsJ8cRVUXPcUMIQgk1kvM8...
lariboksyiemlieh@gmail.com	✉	Apr 29, 2019	Apr 29, 2019	ZafrfNpql0WbmdeBwBK9pQKknCi2
jasmineasangma539@gmail...	G	Apr 17, 2019	Apr 17, 2019	kgI1sVTG3iZs8TojRQ4hAApCfw43
wiwatdakaiakai@gmail.com	G	Apr 29, 2019	Apr 29, 2019	pY3AKDXEnNcVelRpvSO9JbWa63...
ichimichilaribok@gmail.com	G	Apr 15, 2019	Apr 28, 2019	wYcPzkxn7BcjnSwRPgUpkJZHyiC3
dd@gmail.com	✉	Apr 17, 2019	Apr 28, 2019	xRSgAwbyNHNXsEJ8QEaQ7npalA...
haphi33chyne@gmail.com	G	Apr 29, 2019	Apr 29, 2019	zEjGO6ANANRnGPtm1QCeaAheD...

Rows per page: 50 1-10 of 10

Version Control

We have extensively used Git version control in our project right from the start. We have also created a remote repository on GitHub, where we have all the commits and the documentation of this report.

link to GitHub repository : https://github.com/ichimichi/Collaborative-Doodling



ichimichi / Collaborative-Doodling

Realtime Collaborative Doodling on Android https://play.google.com/store/apps/de...

75 commits · 1 branch · 2 releases · 2 contributors

ichimichi added link of app in playstore — Latest commit 12c79d4 11 minutes ago

.idea	added team members to the documention	13 minutes ago
app	added implementation details	18 hours ago
documents	added team members to the documention	13 minutes ago
gradle/wrapper	project created	23 days ago
.gitignore	Updated gitignore	15 days ago
README.md	added link of app in playstore	11 minutes ago
build.gradle	added firebase realtime database as dependency	5 days ago
gradle.properties	project created	23 days ago
gradlew	project created	23 days ago
gradlew.bat	project created	23 days ago
settings.gradle	project created	23 days ago

README.md

Collaborative Doodling

TABLE OF CONTENTS

- Abstract
- Team Members
- Introduction
  - Motivation
  - Main Contributions
- Feasibility Analysis
- UML Diagrams
- Implementation Details
- Design Patterns Used
- Testing
- User Manual
- Conclusion
- References

link of publised app in playstore

Version control systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members

DESIGN PATTERNS USED

- **Singleton Design Pattern**

```
val user = FirebaseAuth.getInstance()

database = FirebaseDatabase.getInstance()
```

- **Adapter Design Pattern**

```
class SavedDoodleAdapter : RecyclerView.Adapter<CustomViewHolder>()
{

    val doodleTitles = listOf("doodle 1", "doodle 2", "doodle 3")

    override fun getItemCount(): Int
    {
        return doodleTitles.size
    }
}

saved_doodleRV.adapter = SavedDoodleAdapter()
```

- **Observer Design Pattern**

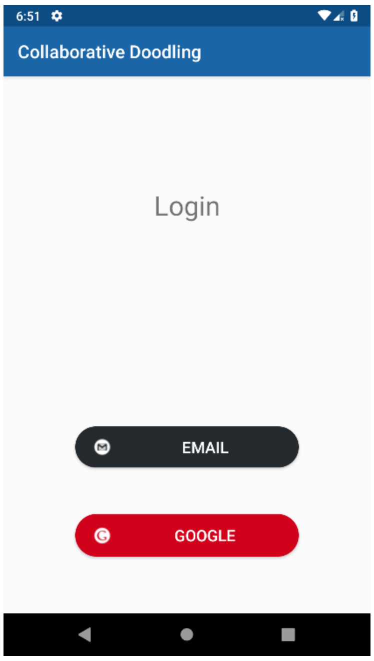
```
newDoodleBtn.setOnClickListener {
    val intent = Intent(this,sessionOptionActivity::class.java)
    startActivity(intent)
}

pencilAV.setOnClickListener {
    index = ((index + 1) % strokeList.size )
    pencilAV.setAnimation(strokeList[index])
    pencilAV.resumeAnimation()
}

drawingInstruction!!.addValueEventListener( object: ValueEventListener {

    override fun onCancelled(error: DatabaseError) {
        Log.w("error", "Failed to read value.", error.toException())
    }

    override fun onDataChange(dataSnapshot: DataSnapshot) {
        val value = dataSnapshot.getValue(Instruction::class.java)
        Log.d("command", value.toString())
        // execute commands
    }
})
```

TESTING

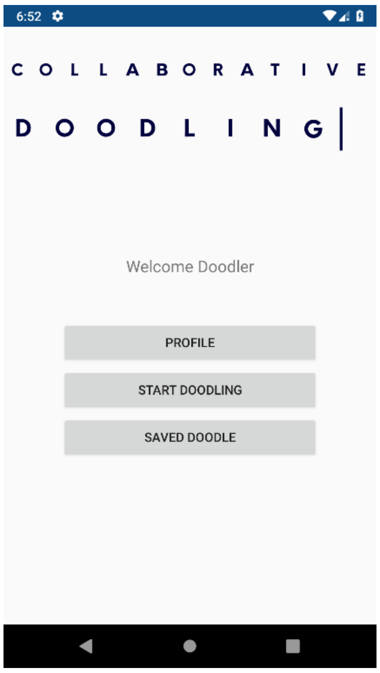
We have not used any testing tool for testing our project. However, we have tested it manually. We tested the ability of the software application to update the drawings in real time. We have performed testing for two or more users to be able to enter the same room and give their input. We have tested the speed of drawing, with which a user can draw without affecting the quality of the image.

USER MANUAL

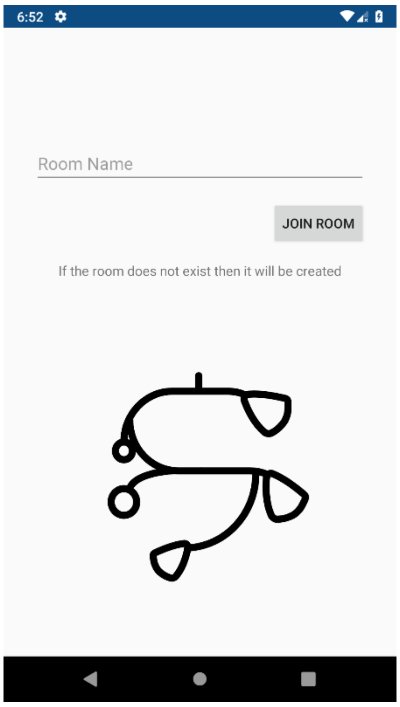
- Step 1: For starting the application click on the application icon. This will open the app and bring you to a page to sign in with an existing account or create a new account.



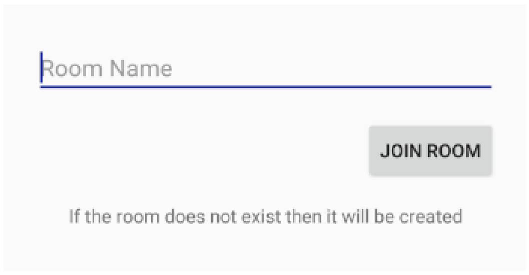
- Step 2: In the main page you will get options to choose to view your **Profile**, view **Saved Doodles** or **Start Doodling**.



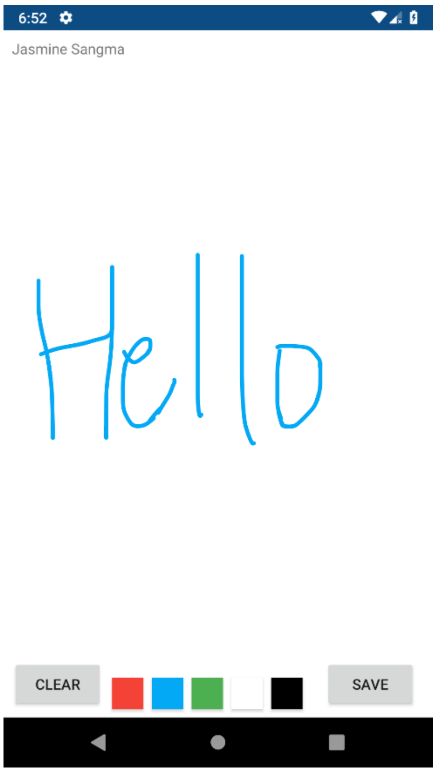
- Step 3: On selecting **START DOODLING** you will be asked to enter the name of an existing room or a new room and select **JOIN ROOM**.



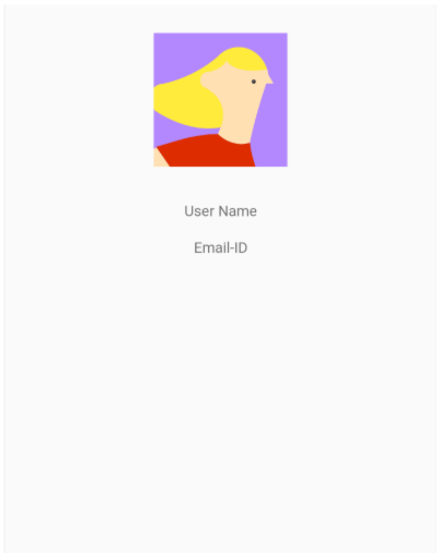
- Step 4: To allow other people to join your room, your friends must enter the name of your Room while joining a room from their phone.



- Step 5: You can now collaborate with anyone who joins the room. You can save your collaborated art by selecting **SAVE**, or clear your canvas by selecting **CLEAR**



- You can view your profile details on selecting **PROFILE**



PROJECT CONCLUSION

The primary goal of this report is to identify action initiatives that make up the **COLLABORATIVE DOODLING APPLICATION** for a *New Change* in communicating our ideas with one another. To that end, The Project Summary Report calls for an increase in sharing ideas including discussion of various Project Models among students or business related work, it is time saving, and easy to access and it enables distance learning.

The goals of the Collaborative Doodling Project effectiveness can be realized through continuing use among all members of the campus community. To conclude with the words “Let’s not limit our Ideas of Creation and set Boundaries to it; Let's Launch and grow”.

REFERENCES

- **Head First Android Development: A Brain-Friendly Guide, 2nd Edition** - *Dawn Griffiths, David Griffiths*
- **Kotlin for Android Developers** - *Antonio Leiva*
- **Firebase Documentation**
- **Android Documentation**
- **FirebaseAuthUI Documentation**
- **Lottie Android Documentation**