Implementation Document

Following document contains the source code of the Tractility application.

Click event on Start Button

btn\_start.setOnClickListener **{** isActivityStarted = true  
 NotificationManagerCompat.from(this).notify(NOTIFICATION\_ID,buildNotification)  
 startAndPauseActivity()  
**}**

start and pause Activity

private fun startAndPauseActivity(){  
 if (!running) {  
 startStopWatch()  
 } else {  
 pauseStopWatch()  
  
 }  
  
}

Start stop watch

// START THE STOPWATCH  
private fun startStopWatch(){  
 iArrow.startAnimation(AnimationUtils.loadAnimation(this, R.anim.*rotating\_arrow*))  
 c\_chronometer.*base* = SystemClock.elapsedRealtime() + pauseTime  
 c\_chronometer.start()  
 running = true  
 btn\_start.*text* = "Pause"  
}

Pause the stopwatch

// PAUSE THE STOPWATCH  
private fun pauseStopWatch(){  
 iArrow.clearAnimation()  
 pauseTime = c\_chronometer.*base* - SystemClock.elapsedRealtime()  
 c\_chronometer.stop()  
 running = false  
 btn\_start.*text*= "Resume"  
}

Click event on Reset button:

btn\_reset.setOnClickListener **{** if(isActivityStarted){  
 pauseStopWatch()  
 confirmationDialog()  
 }  
  
**}**

Reset stopwatch

private fun resetStopwatch(){  
  
 c\_chronometer.*base*= SystemClock.elapsedRealtime()  
 iArrow.clearAnimation()  
 pauseTime = 0  
 running=false  
 btn\_start.*text*="Start"  
}

Stop button of Stopwatch:

btn\_stop.setOnClickListener**{** if( isActivityStarted){  
 NotificationManagerCompat.from(this).cancel(NOTIFICATION\_ID)  
 progress = c\_chronometer.*text*.toString()  
 pauseStopWatch()  
 saveDialogFunction()  
  
 }else{  
 Toast.makeText(this, "Activity not started", Toast.*LENGTH\_LONG*).show()  
 }  
**}**

confirmation dialog

private fun confirmationDialog(){  
 val confBuilder = AlertDialog.Builder(this)  
 confBuilder.setTitle("Reset stopwatch?")  
 confBuilder.setMessage("resetting the stopwatch will cancel the progress")  
 confBuilder.setPositiveButton("RESET") **{** dialog: DialogInterface, i: Int **->** resetStopwatch()  
 isActivityStarted=false  
 dialog.cancel()  
 **}** confBuilder.setNegativeButton("Cancel") **{** dialog: DialogInterface, i: Int **->** dialog.cancel()  
 **}** confBuilder.show()  
}

Rotating clock arrow:

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shareInterpolator="@android:interpolator/linear"  
 >  
 <rotate  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:fromDegrees="0"  
 android:toDegrees="360"  
 android:pivotX="50%"  
 android:pivotY="50%"  
 android:repeatCount="infinite"  
 android:duration="1000"  
  
 />  
</set>

creating the notification channel

//creating a notification channel  
private fun createNotificationChannel() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
  
 val importance = NotificationManager.*IMPORTANCE\_DEFAULT* val channel1 = NotificationChannel(CHANNEL\_ID, CHANNEL\_NAME, importance).*apply* **{** //behaviour of the notification  
 setSound(null,null)  
 **}** // Register the channel with the system  
 val notificationManager: NotificationManager =  
 getSystemService(Context.*NOTIFICATION\_SERVICE*) as NotificationManager  
 notificationManager.createNotificationChannel(channel1)  
 }  
}

Build Notification panel

// to build the notification panel  
val buildNotification = NotificationCompat.Builder(this, CHANNEL\_ID)  
 .setSmallIcon(R.drawable.*ic\_stat\_name*)  
 .setContentTitle("Tractivity")  
 .setContentText("is running")  
 .setNotificationSilent()  
 .setContentIntent(pendingIntent)  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*).build()

Save Activity Dialog function

//SAVE ACTIVITY CUSTOM DIALOG  
@SuppressLint("SetTextI18n")  
private fun saveDialogFunction() {  
  
 val saveDialogView = LayoutInflater.from(this).inflate(R.layout.*dialog\_save*,null)  
 saveDialogView.tv\_description.*text*= "Following $progress is spent on:"  
 val saveDialogbuilder = AlertDialog.Builder(this)  
 .setView(saveDialogView)  
 val saveActivityDialog = saveDialogbuilder.show()  
 saveActivityDialog.setCancelable(false) // prevent user to close the dialog by clicking outside the dialog  
 saveDialogView.bt\_selectActivity.setOnClickListener **{** // ACTIVITY DIALOG populated with the activity lists from the DB  
 firestore.collection(Constants.USERS).document(FireStoreClass().getCurrentUserID())  
 .collection(Constants.ACTIVITIES)  
 .get()  
 .addOnSuccessListener **{** activities **->** Log.d("DB", "All documents received")  
 val activityNameList = ArrayList<String>()  
 for (activity in activities) {  
 val activityClass: ActivityClass? = activity.toObject(ActivityClass::class.*java*)  
 if (activityClass != null){  
 activityNameList.add(activityClass.name)  
  
 }  
  
 }  
 val items:Array<String> = activityNameList.*toTypedArray*()  
 val selectActivityBuilder = AlertDialog.Builder(this)  
 selectActivityBuilder.setTitle("Choose activity")  
 selectActivityBuilder.setSingleChoiceItems(items,-1)**{** dialogInterface: DialogInterface, i :Int **->** saveDialogView.et\_activityName.setText(items[i])  
 isActivitySelected = true  
 dialogInterface.dismiss()  
  
 **}** selectActivityBuilder.setNeutralButton("cancel")**{** dialog:DialogInterface, which**->** dialog.cancel()  
 **}** val activitySelectDialog = selectActivityBuilder.create()  
 activitySelectDialog.show()  
 **}** .addOnFailureListener **{** exception **->** Log.d("DB", "Error getting documents: ", exception)  
 **}  
  
 }** saveDialogView.bt\_submit.setOnClickListener**{** activityName = saveDialogView.et\_activityName.*text*.toString()  
 if (TextUtils.isEmpty(activityName)){  
 Toast.makeText(*applicationContext*, "Enter or select Activity Name", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 val activity = ActivityClass(activityName)  
 val record = ActivityRecordClass(parseProgress(progress!!))  
 if(isActivitySelected){  
 FireStoreClass().saveRecordOnDB(record,activity.name)  
  
 } else{  
 FireStoreClass().saveActivityOnDB(activity)  
 FireStoreClass().saveRecordOnDB(record,activity.name)  
 }  
 isActivitySelected = false  
 isActivitySelected = false  
 resetStopwatch()  
 saveActivityDialog.dismiss()  
 }  
  
 **}** saveDialogView.bt\_cancel.setOnClickListener**{** Toast.makeText(*applicationContext*, "Activity not saved", Toast.*LENGTH\_SHORT*).show()  
 saveActivityDialog.dismiss()  
 **}**}

Ligin button on intro activity;

SignUp button on Intro activity:

bt\_signUp.setOnClickListener**{** val intent = Intent(this, SignUpActivity::class.*java*)  
 startActivity(intent)  
 finish()  
**}**

validate login form:

private fun validateLogin ( email: String ,  
 password: String) :Boolean{  
 when {  
 TextUtils.isEmpty(email) ->{  
 showError("Please enter your email address")  
 return false  
 }  
 TextUtils.isEmpty(password) ->{  
 showError("Please enter your password")  
 return false  
 }  
 else ->{  
 return true  
 }  
 }  
}

error snakbar

fun showError(errorMessage:String){  
 val snackBar= Snackbar.make(findViewById(android.R.id.*content*),errorMessage,Snackbar.*LENGTH\_LONG*)  
 snackBar.show()  
}

implementing user Login:

private fun loginUser (){  
 val email:String = et\_email\_login.*text*.toString().*trim***{ it** <= ' '**}** val password:String = et\_password\_login.*text*.toString()  
 if(validateLogin(email,password)){  
 auth.signInWithEmailAndPassword(email, password)  
 .addOnCompleteListener(this) **{** task **->** if (task.*isSuccessful*) {  
 Log.d("Login", "LoginInWithEmail:success")  
 val user = auth.*currentUser* Toast.makeText(this,  
 "Welcome back to Tractivity",Toast.*LENGTH\_SHORT*).show()  
 val intent = Intent(this, TractivityMain::class.*java*)  
 startActivity(intent)  
 finish()  
  
  
 } else {  
 // If sign in fails, display a error message to the user.  
 Log.w("Login", "LogInWithEmail:failure", task.*exception*)  
 Toast.makeText(*baseContext*, "Authentication failed, incorrect email or password was typed",Toast.*LENGTH\_SHORT*).show()  
 }  
 **}** }  
}

validate SignUp form:

private fun validateSignUp (name:String , email: String ,  
 password: String, reTypedPassword : String) :Boolean{  
 when {  
 TextUtils.isEmpty(name) ->{  
 showError("Please enter your name")  
 return false  
 }  
 TextUtils.isEmpty(email) ->{  
 showError("Please enter your email address")  
 return false  
 }  
 TextUtils.isEmpty(password) ->{  
 showError("Please enter a password")  
 return false  
 }  
 TextUtils.isEmpty(reTypedPassword) ->{  
 showError("Please re-Type your password")  
 return false  
 }  
 !isPasswordMatched(password,reTypedPassword) ->{  
 return false  
 }  
 else ->{  
 return true  
  
 }  
 }  
}

create User account:

private fun createUser(){  
 val name: String = et\_name.*text*.toString().*trim***{ it** <= ' '**}** val email: String = et\_email.*text*.toString().*trim***{ it** <= ' '**}** val password : String = et\_password.*text*.toString()  
 val reTypedPassword : String = et\_reTypePassword.*text*.toString()  
  
 if(validateSignUp(name,email,password,reTypedPassword)){  
 auth.createUserWithEmailAndPassword(email,password)  
 .addOnCompleteListener(this)**{** task **->** if (task.*isSuccessful*) {  
 val user = UserClass(Firebase.*auth*.*currentUser*!!.*uid*,name,email)  
 FireStoreClass().registerUserOnDB(user)  
 Toast.makeText(this,  
 "Welcome $name to Tractivity",Toast.*LENGTH\_SHORT*).show()  
 val intent = Intent(this, TractivityMain::class.*java*)  
 startActivity(intent)  
  
 finish()  
 }else {  
 Toast.makeText(*baseContext*, "Authentication failed.",  
 Toast.*LENGTH\_SHORT*).show()  
 }  
 **}** }  
}

reset Password:

private lateinit var auth: FirebaseAuth  
  
override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_password\_reset*)  
 *window*.*decorView*.*systemUiVisibility* = View.*SYSTEM\_UI\_FLAG\_FULLSCREEN* btn\_resetPass.setOnClickListener**{** val email:String = et\_email\_login.*text*.toString().*trim***{ it** <= ' '**}** if (email.*isEmpty*()){  
 Toast.makeText(this, "Please enter your email",  
 Toast.*LENGTH\_SHORT*).show()  
 } else{  
 auth.sendPasswordResetEmail(email).addOnCompleteListener**{** task **->** if(task.*isSuccessful*){  
 Toast.makeText(this, "Email sent, check your inbox",  
 Toast.*LENGTH\_SHORT*).show()  
 finish()  
 }  
 **}** }  
 **}**}

parse string time to long second:

fun parseProgress (progress:String) : Long{  
 var progressInSeconds :Long = 0  
 val strArray : Array<String> = progress.*split*(":").*toTypedArray*()  
 if(strArray.size<3){  
 progressInSeconds += strArray[0].*toLong*()\*60  
 progressInSeconds += strArray[1].*toLong*()  
 return progressInSeconds  
 }  
 progressInSeconds += strArray[0].*toLong*()\*3600  
 progressInSeconds += strArray[1].*toLong*()\*60  
 progressInSeconds += strArray[2].*toLong*()  
 return progressInSeconds  
}

Navigation Drawer:

override fun onNavigationItemSelected(item: MenuItem): Boolean {  
 when (item.*itemId*){  
 R.id.*main\_pg* -> {  
 drawer\_layout.closeDrawer(GravityCompat.*START*)  
 return true  
 }  
 R.id.*activities* ->{  
 val intent = Intent(this, ActivitiesActivity::class.*java*)  
 startActivity(intent)  
 }  
 R.id.*charts* ->{  
 val intent = Intent(this, ChartsActivity::class.*java*)  
 startActivity(intent)  
 }  
 R.id.*profile* ->{  
 val intent = Intent(this, UserProfileActivity::class.*java*)  
 startActivity(intent)  
 }  
 R.id.*logout* ->{  
 FirebaseAuth.getInstance().signOut()  
 val intent = Intent(this, IntroActivity::class.*java*)  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP* or Intent.*FLAG\_ACTIVITY\_NEW\_TASK*)  
 startActivity(intent)  
 finish()  
 }  
 }  
 drawer\_layout.closeDrawer(GravityCompat.*START*)  
 return true  
}

populate the Profile screen:

fun populateProfileActivity () {  
 firestore.collection(Constants.USERS)  
 .document(getCurrentUserID()).get()  
 .addOnSuccessListener **{**documentSnapshot **->** Log.d(ContentValues.*TAG*, "Document received")  
 val loggedUser: UserClass? = documentSnapshot

.toObject(UserClass::class.*java*)  
 if (loggedUser != null) {  
 gbloggedUser = loggedUser  
 tv\_username\_profile.*text* = loggedUser.name  
 tv\_profile\_name.*text*=loggedUser.name  
 tv\_profile\_email.*text* = loggedUser.email  
 tv\_score.*text* = ""  
 }  
 **}** .addOnFailureListener**{** exception **->** Log.d(ContentValues.*TAG*, "get failed with ", exception)  
 **}** }