Nathasya Eliora

1706979404

TKTPL B

DOKUMEN LAPORAN UTS DAN UAS

Aplikasi: Today. a Pomodoro technique tools to get your tasks done today. Main features: timer, tasks

Github: https://github.com/nathasyae/today

Branch code: uas || branch apk & laporan: apk

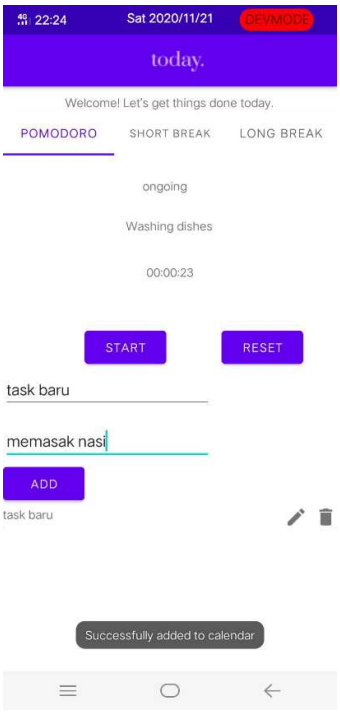
BAGIAN UTS

1. Menerapkan seluruh stack Android Framework standard
    1. Menerapkan activity
       Disini activitynya adalah main activity. Di main activity ini ada timer dan task. Berisi:
       - Initialisasi variable
       - Finder komponen ui
       - Setting listener komponen

    2. Menggunakan Service dan pemanggilan Remote Method (di luar aplikasi)
       Service yang digunakan adalah penghitungan untuk alarm. Jadi ketika dia memilih sesi pomodoro (missal), akan di set alarm yang akan berbunyi +25menit setelahnya.jangan lupa mendaftarkan dulu di manifest.

```
@RequiresApi(Build.VERSION_CODES.KITKAT)
fun startAlarm(duration: Long) {
    val millis :Long  = SystemClock.uptimeMillis() + duration
    val alarmManager :AlarmManager  = getSystemService(Context.ALARM_SERVICE) as AlarmManager
    val intent = Intent( packageContext: this, AlarmReceiver::class.java)
    val pendingIntent :PendingIntent!  = PendingIntent.getBroadcast( context: this,  requestCode: 0, intent,  flags: 0)
    alarmManager.setExact(AlarmManager.RTC_WAKEUP, millis, pendingIntent)
}
```

    3. Memanfaatkan Content Provider
       Menggunakan Calendar Provider ketika ada task baru akan langsung masuk ke calendar hp. Caranya adalah dengan mendaftarkan permission, lalu membuat fungsi di activity setiap tombol add button ditekan.

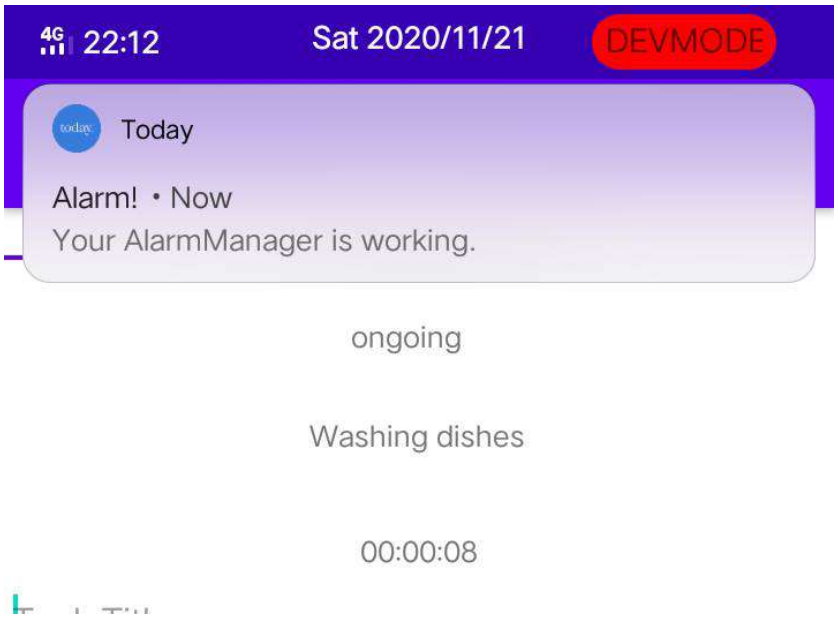



```
fun addToCalendar(tasktitle: String, taskdetail: String){
    var cr: ContentResolver? = this.getContentResolver()
    var cv: ContentValues = ContentValues()
    cv.put(CalendarContract.Events.TITLE, tasktitle)
    cv.put(CalendarContract.Events.DESCRIPTION, taskdetail)
    cv.put(CalendarContract.Events.DTSTART, Calendar.getInstance().getTimeInMillis());
    cv.put(
        CalendarContract.Events.DTEND,
        Calendar.getInstance().getTimeInMillis() + 60 * 60 * 1000
    );
    cv.put(CalendarContract.Events.CALENDAR_ID, 1)
    cv.put(CalendarContract.Events.EVENT_TIMEZONE, Calendar.getInstance().getTimeZone().getID())
    var uri: Uri = cr?.insert(CalendarContract.Events.CONTENT_URI, cv)!!

    Toast.makeText( context: this,  text: "Successfully added to calendar", Toast.LENGTH_SHORT).show()
}
```

4. Menerapkan BroadcastReceiver

   Untuk menampilkan notifikasi ketika mencapai waktu yang telah ditentukan. Caranya adalah membuat dahulu kelas broadcastreceivernya (untuk mengatur onreceive), membuat notificationutil (untuk bicara pada notification managernya), dan akhirnya dipanggil dengan method di activity.

```
class AlarmReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        // This method is called when the BroadcastReceiver is receiving an Intent broadcast.
        val notificationUtils = NotificationUtils(context)
        val notification : Notification! = notificationUtils.getNotificationBuilder().build()
        notificationUtils.getManager().notify( id: 150, notification)
    }
}
```



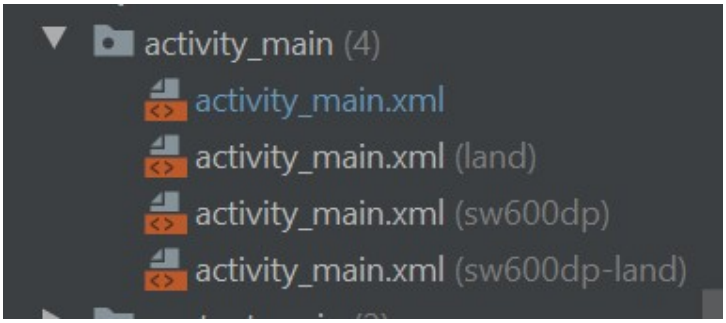
5. Menerapkan Background Process (cth: AsyncTask) yang tidak “mati” ketika activity tidak aktif

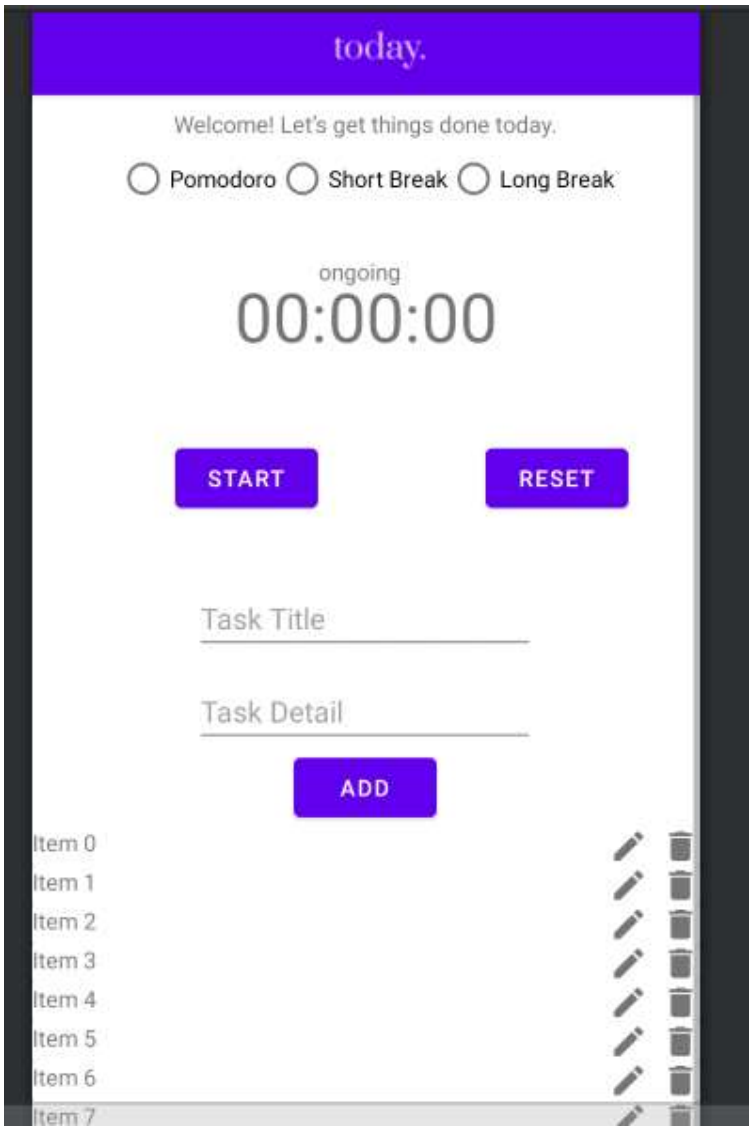
   Background process dilakukan dengan menggunakan runnable. Stopwatch tetap berjalan walau sedang tidak dibuka appnya.

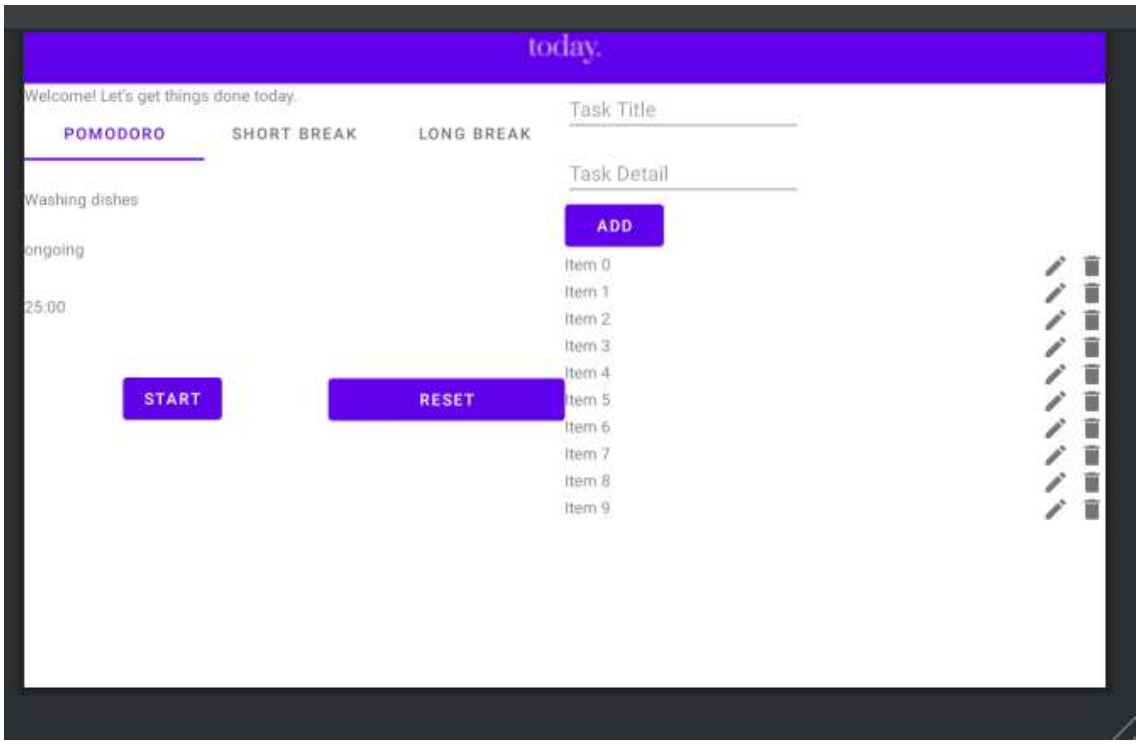
```
val runnable = object : Runnable {

    override fun run() {
        if (boolstart == false) {
            boolstart = true
            display.text = "Pause"
        } else {
            boolstart = false
            display.text = "Resume"
        }
        seconds = SystemClock.uptimeMillis() / 1000 + pauseTime - startTime
        pause = seconds
        minutes = seconds / 60
        seconds = seconds % 60
        hours = hours / 60
        minutes = minutes % 60
```

2. Menerapkan multi environment
    a. Multi Layout

    Ada 4, yaitu handphone screen portrait dan landscape, lalu ada tablet portrait dan landscape. Portrait di hp dan tablet sama, yaitu vertical linear. Sedangkan yang landscape dibagi dua kanan kiri.

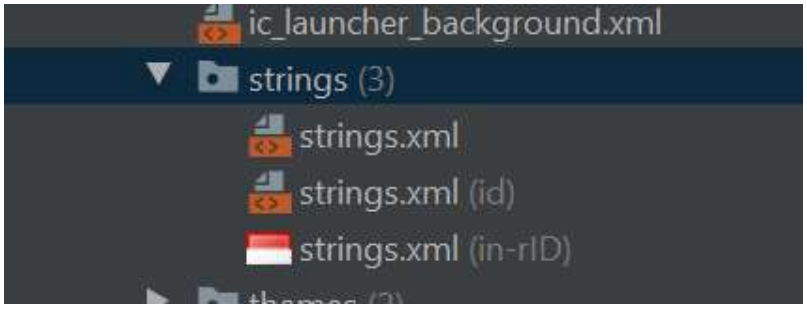



today.
Welcome! Let's get things done today.
Pomodoro
Short Break
Long Break
ongoing
00:00:00
START
RESET
Task Title
Task Detail
ADD
Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7



b. Multi Language (i18n)

Dibuat string English dan indo. Saya melakukannya dengan memencet Bahasa di section kanan desain xml lalu create new language. Lalu dari situ diisi terjemahannya satu-satu. Sebelumnya semua string telah diextract dan diberi id supaya mudah untuk direplace.



3. Menerapkan Design Pattern MVVM & Background Task

Saya membuat file view model ketika dibutuhkan. Contohnya saat mengakses task dan log.

```
class MainViewModel(application: Application) : AndroidViewModel(application) {

    private var taskRepository = TaskRepository(application)
    private var tasks: LiveData<List<Task>>? = taskRepository.getTasks()

    fun insertTask(task: Task) {
        taskRepository.insert(task)
    }

    fun getTasks(): LiveData<List<Task>>? {
        return tasks
    }
```

Binding dilakukan dengan membuat adapter, lalu ditampilkan di recyclerview. Diobserve juga di viewmodel.

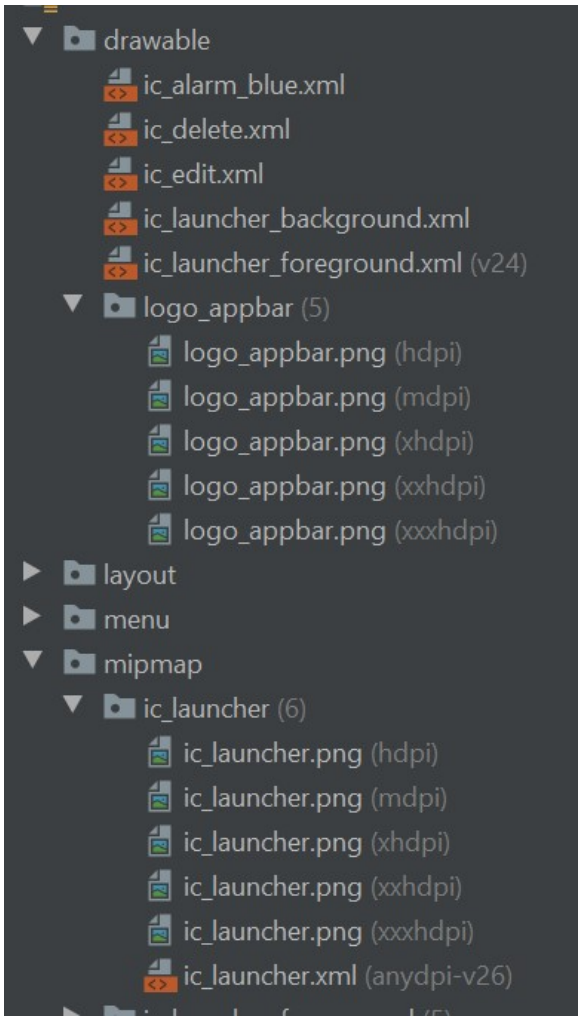
4. Menerapkan Assets dengan benar

   a. String resource

      Seperti di poin 2b

   b. Mempersiapkan drawable resource untuk 2 (dua) screen size (& beda resolusi).

      Saya mempersiapkan foto yang akan digunakan seperti logo dan clipart. Lalu file > new > asset. Lalu isi settingsnya dan akan otomatis dibuatkan untuk berbegai density.

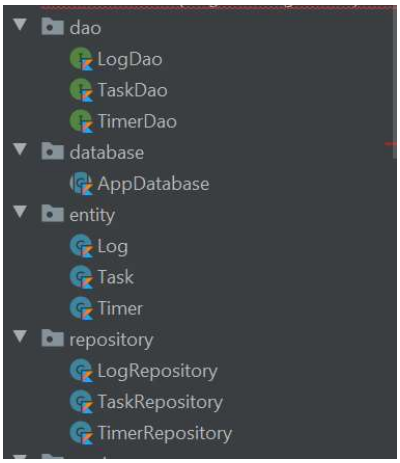


c. Memanfaatkan ContentProvider

Sudah di poin 1

5. Menerapkan Data Persistence

Data persistence diterapkan dengan menggunakan room. Pertama semua kebutuhan dimasukan ke gradle. Lalu mulai membuat entity, ada log, task, dan timer. Log menyimpan session yang dilakukan, jadi tiap pencet start, akan menyimpan log. Selanjutnya task untuk menyimpan task hari itu dan timer untuk menyimpan jenis2 sesi.



6. Inovasi

Saya menambahkan fitur split screen. Caranya adalah dengan memasukan beberapa settings ke manifest.

```
<activity
    android:name=".MainActivity"
    android:resizeableActivity="true"
    android:label="Today"
    android:theme="@style/Theme.Today.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>

    <layout android:defaultHeight="500dp"
        android:defaultWidth="600dp"
        android:gravity="top|end"
        android:minHeight="450dp"
        android:minWidth="300dp" />
</activity>
```

BAGIAN UAS

6. **Runtime Permission**

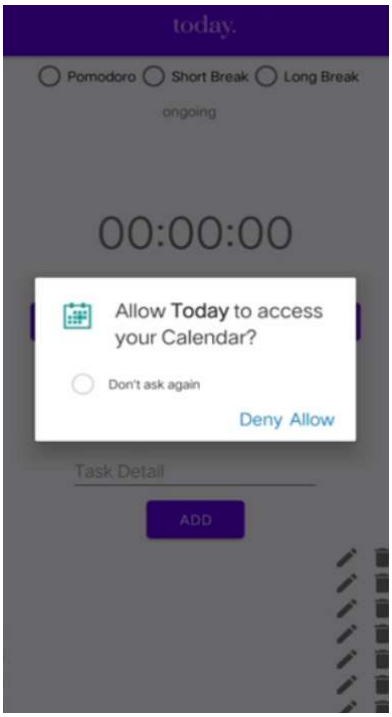
   Sebelum add task, harus memberikan izin akses kalender. Jika di-approve, dapat tertambah tasknya ke app dan kalender. Jika tidak ke halaman permission detail.

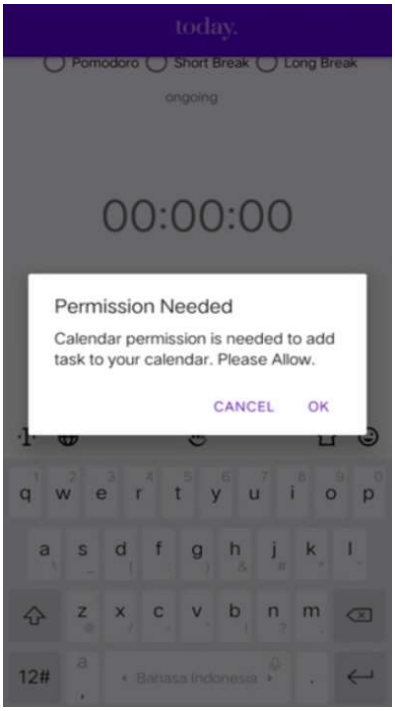
   - Membuat fungsi permission_fn untuk mengecek apakah sudah diberi permissions
   - Membuat fungsi requestCalendarPermission untuk merequest permission
   - Membuat fungsi onRequestPermissionsResult untuk memproses action pengguna jika deny tampilkan alasan, jika allow langsung lanjut ke addTask
   - Memanggil permission_fn di button listener add task

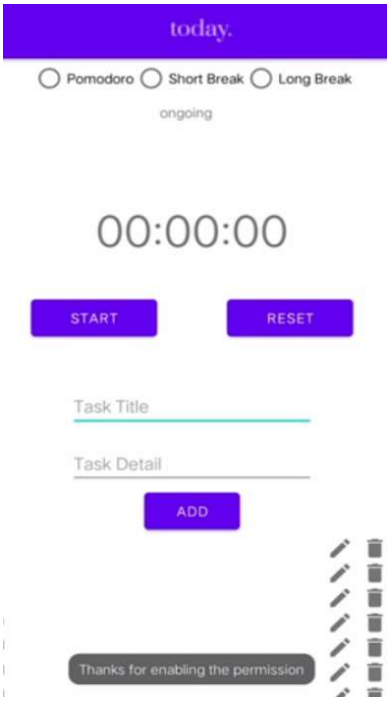
```
private fun permission_fn() {
    if (ContextCompat.checkSelfPermission(
                    context: this@MainActivity,
                Manifest.permission.WRITE_CALENDAR
            ) == PackageManager.PERMISSION_GRANTED
    ) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(
                        activity: this@MainActivity,
                    Manifest.permission.WRITE_CALENDAR
                )
        ) {
        } else {
            ActivityCompat.requestPermissions(
                    activity: this,
                arrayOf(Manifest.permission.WRITE_CALENDAR),
                REQUEST_PERMISSION
            )
        }
    } else {
        requestCalendarPermission()
    }
}
```

```
private fun requestCalendarPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(
                    activity: this,
                Manifest.permission.WRITE_CALENDAR
            )
    ) {
        ActivityCompat.requestPermissions(
                activity: this@MainActivity,
            arrayOf(Manifest.permission.WRITE_CALENDAR),
            REQUEST_PERMISSION
        )

    } else {
        ActivityCompat.requestPermissions(
                activity: this,
            arrayOf(Manifest.permission.WRITE_CALENDAR),
            REQUEST_PERMISSION
        )
    }
}
```

```
override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<String>,
        grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (requestCode == REQUEST_PERMISSION) {
        if (grantResults.size > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText( context: this,  text: "Thanks for enabling the permission", Toast.LENGTH_SHORT)
                    .show()

            addTask()

        } else {
            android.app.AlertDialog.Builder( context: this)
                    .setTitle("Permission Needed")
                    .setMessage("Calendar permission is needed to add task to your calendar. Please Allow.")
                    .setPositiveButton( text: "OK",
                            DialogInterface.OnClickListener { dialog, which ->
                                ActivityCompat.requestPermissions(
                                        activity: this@MainActivity,
                                    arrayOf(Manifest.permission.WRITE_CALENDAR),
                                    REQUEST_PERMISSION
                                )
                            })
```







7. **JNI (Java Native Interface)**

   Membuat greeting function di C sesuai waktu buka. Jam 1am-12pm morning, afternoon 12pm-4pm, evening 4pm-9pm, night (9pm to 12pm).

   - Install NDK dan CMake di SDKManager
   - Membuat cmakelist.txt di app berisi minimum cmake required, add_library, find_library, target_link_library yang akan mengarah ke native-lib.cpp

```
cmake_minimum_required(VERSION 3.4.1)
```

```
add_library( # Sets the name of the library.
             native-lib

             # Sets the library as a shared library.
             SHARED

             # Provides a relative path to your source file(s).
             src/main/cpp/native-lib.cpp )
```

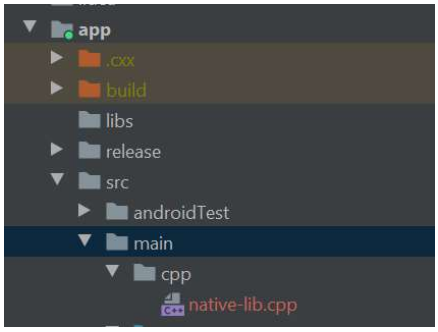
   - Memasukan externalNativeBuild di android & default config dan ndkversion di gradle app

```
externalNativeBuild {
    cmake {
        path "CMakeLists.txt"
    }
}
```

```
externalNativeBuild {
    cmake {
        cppFlags ""
    }
}
```

```
ndkVersion '22.0.6917172'
```

- Membuat folder cpp dan membuat native-lib.cpp yang berisi fungsi mengecek morning/afternoon/evening/night dan mengembalikan greeting yang sesuai.

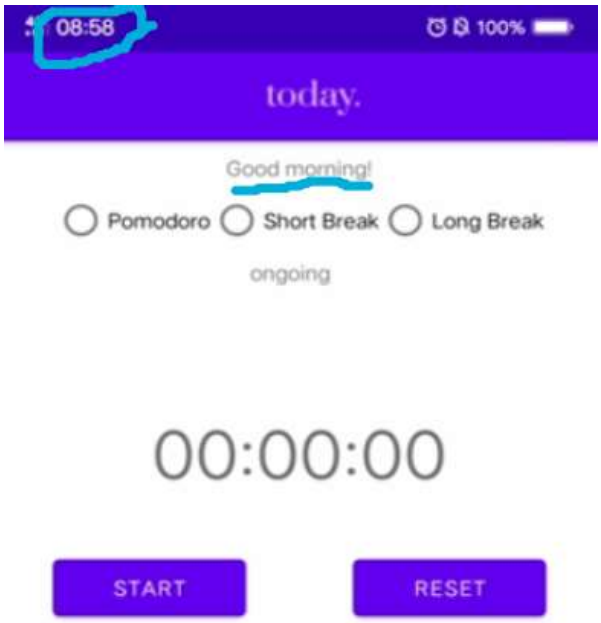



```
#include <jni.h>

extern "C"
JNIEXPORT jstring JNICALL
Java_id_ac_ui_cs_mobileprogramming_nathasyaeliora_Today_MainActivity_getGreeting(
        JNIEnv *env, jobject thiz, jint hour) {

    if (hour >= 0 && hour < 12){
        return env->NewStringUTF("Good morning!");
    } else if(hour >= 12 && hour < 16){
        return env->NewStringUTF("Good afternoon!");
    } else if(hour >= 16 && hour < 21){
        return env->NewStringUTF("Good evening!");
    } else{
        return env->NewStringUTF("Good night!");
    }
```

- Mendefine external function dan companian object untuk meng-load file berisi fungsi native. Lalu memanggil fungsi tsb di oncreate dengan parameter hour of day now dari calendar. Lalu hasil return dari fungsi tsb ditampilkan di komponen greetings text.

```
external fun getGreeting(hour: Int): String

companion object {
    init {
        System.loadLibrary( libname: "native-lib")
    }
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val greetingresult = getGreeting(Calendar.getInstance()[Calendar.HOUR_OF_DAY])
    setContentView(R.layout.activity_main)
    greetings_text.text = greetingresult.toString()
```



8. **Animasi OpenGL**

   Membuat animasi dengan OpenGL di splash screen.

   - Menambahkan user-feature android:glEsVersion="0x00020000" ke android manifest
   - Membuat shape yang ingin dianimasikan di square.kt berisi koordinat shape, warna, fungsi draw, dan keterangan-keterangan lain yang dibutuhkan.

```
class Square() {
    private val vertexShaderCode =  // This matrix member variable provides a hook to manipulate
        // the coordinates of the objects that use this vertex shader
        "uniform mat4 uMVPMatrix;" +
                "attribute vec4 vPosition;" +
                "void main() {" +  // The matrix must be included as a modifier of gl_Position.
                // Note that the uMVPMatrix factor *must be first* in order
                // for the matrix multiplication product to be correct.
                "  gl_Position = uMVPMatrix * vPosition;" +
                "}"
```

   - Membuat renderer di MyGLRenderer.kt untuk merender frame opengl dan semua animasinya (translation x)

```
// based on Google docs and https://github.com/mauroghiglia/OGLDrawTest
class MyGLRenderer : GLSurfaceView.Renderer {
    private var mSquare: Square? = null
    var i = 0f
    var direction = 0
```

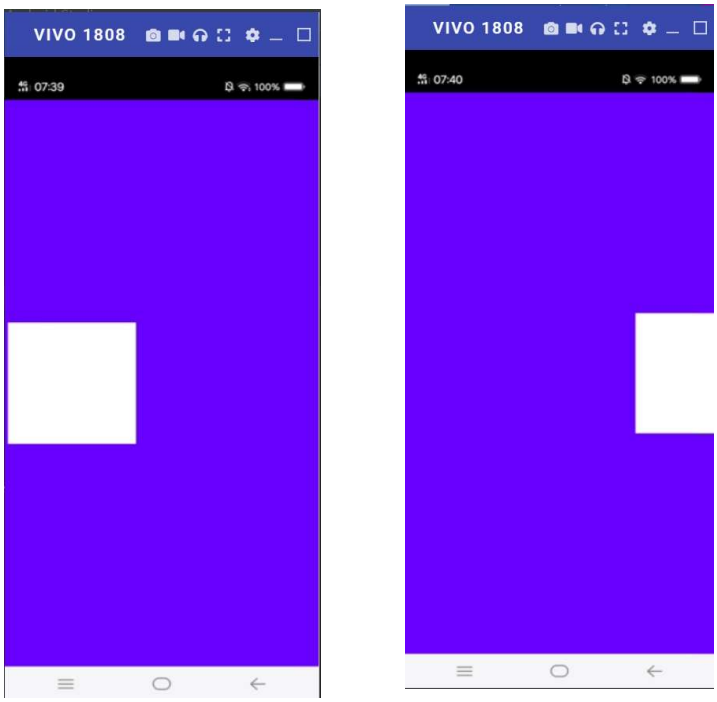
```
if (i > 1) {
    direction = -1
}
if (i < -1) {
    direction = 1
}
i += (0.1 * direction).toFloat()

//Introduce a translation
Matrix.translateM(squareScratch,  mOffset: 0, i,  y: 0.0f,  z: 0.0f)
```

- Membuat surface view di MyGLSurface.kt untuk initialize context dan renderer

```
class MyGLSurfaceView(context: Context) : GLSurfaceView(context) {

    private val renderer: MyGLRenderer

    init {

        // Create an OpenGL ES 2.0 context
        setEGLContextClientVersion(2)

        renderer = MyGLRenderer()

        // Set the Renderer for drawing on the GLSurfaceView
        setRenderer(renderer)
    }
```

- Membuat SplashActivity untuk mengeset konten opengl sebagai tampilan awal splash screen dilanjut ke main activity. Lalu menambahkannya di android manifest.

```
class SplashActivity : AppCompatActivity() {
    private lateinit var glView: GLSurfaceView

    // This is the loading time of the splash screen
    private val SPLASH_TIME_OUT:Long = 4000 // 1 sec
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        glView = MyGLSurfaceView( context: this)
        setContentView(glView)

        Handler().postDelayed({
            // This method will be executed once the timer is over
            // Start your app main activity

            startActivity(Intent( packageContext: this, MainActivity::class.java))

            // close this activity
            finish()
        }, SPLASH_TIME_OUT)
    }
```



9. Connectivity Manager

Cek konektifitas sebelum bisa add to calendar. Jika sudah ada network, bisa add to calendar. Jika tidak akan keluar Toast pemberitahuan tidak bisa menyimpan dan harus menyalakan network.

- Menambahkan permission ke android manifest

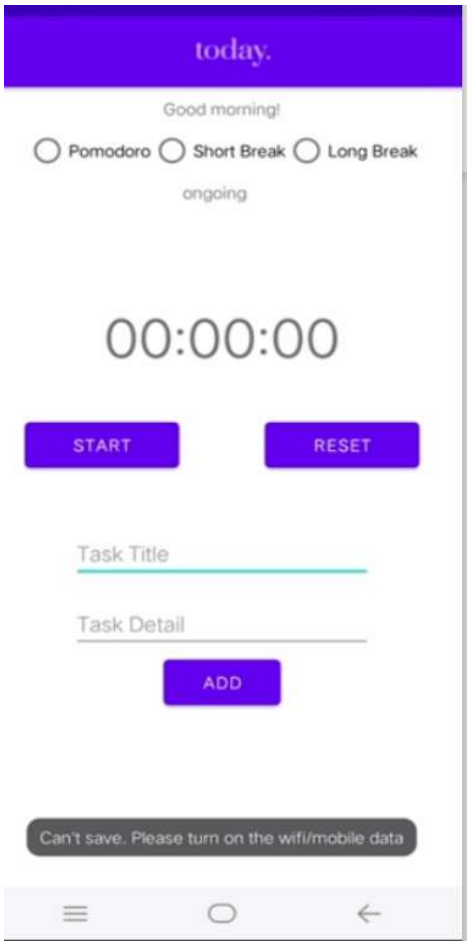
```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET"/>
```

- Membuat fungsi mengecek network sudah menyala atau belum di mainactivity

```
fun checkConnectivity(context: Context): Boolean {
    val cm = context.getSystemService(Context.CONNECTIVITY_SERVICE) as ConnectivityManager
    var activeNetworkInfo: NetworkInfo? = null
    activeNetworkInfo = cm.activeNetworkInfo
    return activeNetworkInfo != null && activeNetworkInfo.isConnectedOrConnecting
}
```

- Memanggil fungsi tsb ketika button add task ditekan

```
add_button.setOnClickListener {  it: View!

    if (checkConnectivity( context: this)){

        mainViewModel.insertTask(
                Task(taskTitleInput.text.toString(), taskDetailInput.text.toString())
        )
        addToCalendar(taskTitleInput.text.toString(), taskDetailInput.text.toString())

    } else{
        Toast.makeText( context: this,  text: "Can't save. Please turn on the wifi/mobile data", Toast.LENGTH_SHORT).show(
    }
```



10. Service Background

Untuk bagian ini saya mengganti timer yang tadinya memakai runnable di UTS, sekarang menjadi dengan service sendiri. Untuk waktunya berubah disesuaikan dengan setting durasi notifikasi (setelah timer habis, muncul notifikasi) agar tidak perlu menunggu puluhan menit.

- Membuat Time Service dengan fungsi-fungsi seperti override onCreate, onDestroy, onBind, onStartCommand, dan companion object. Lalu mendaftarkannya di Android Manifest.

```
class TimeService : Service() {
    var bi = Intent(COUNTDOWN_BR)
    var cdt: CountDownTimer? = null

    override fun onCreate() {
        super.onCreate()
        Log.i(TAG, msg: "Starting timer...")
        cdt = object : CountDownTimer( millisInFuture: 900000, countDownInterval: 1000) {
            override fun onTick(millisUntilFinished: Long) {
                Log.i(TAG, msg: "Countdown seconds remaining: " + millisUntilFinished / 1000)
                bi.putExtra( name: "countdown", millisUntilFinished)
                sendBroadcast(bi)
            }

            override fun onFinish() {
                Log.i(TAG, msg: "Timer finished")
            }
        }
        (cdt as CountDownTimer).start()
    }
```

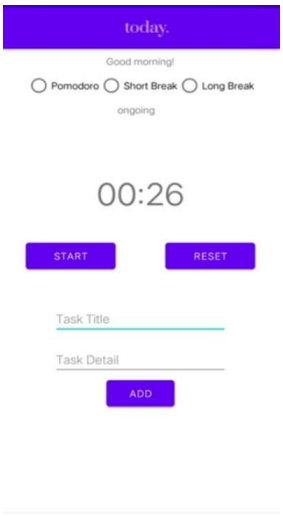
- Menginitialisasi broadcast receiver dan membuat fungsi yang dibutuhkan seperti updateGUI, onResume, onDestroy, onStop, dan onPause.

```
val br: BroadcastReceiver = object : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        if (intent != null) {
            updateGUI(intent)
        }
    }
}
```

```
private fun updateGUI(intent: Intent) {
    if (intent.extras != null) {
        val millisUntilFinished = intent.getLongExtra( name: "countdown", defaultValue: 0)
        display.setText(
            String.format("%02d", (millisUntilFinished /1000 / 60)% 60)
                    + ":" + String.format("%02d", millisUntilFinished / 1000 % 60))
        Log.i(TAG, msg: "Countdown seconds remaining: " + millisUntilFinished / 1000)
    }
}
```

- Memanggil start service jika startbutton diklik, atau stopservice ketika stop button (reset) diklik di mainactivity.

```
startbutton.setOnClickListener {  it: View!
    startService(Intent( packageContext: this, BroadcastService::class.java))
    Log.i(TAG, msg: "Started service")
    startAlarm( duration: 30000)
}


stopbutton.setOnClickListener {  it: View!
    stopService(Intent( packageContext: this, BroadcastService::class.java))
    Log.i(TAG, msg: "Stopped service")
    display.setText("00:00")
}
```



11. Notifikasi

Notifikasi ketika timer sudah mencapai waktu sesuai Pomodoro. Tapi karena lama, saya set dulu lebih cepat supaya bisa dicek di 30 detik saja. Untuk tahapannya secara garis besar seperti di poin 4 Broadcast Receiver karena sudah saya kerjakan di UTS dengan sedikit perubahan.

- Membuat Alarm Receiver dari Broadcast Receiver

```
class AlarmReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        // This method is called when the BroadcastReceiver is receiving an Intent broadcast.
        val notificationUtils = NotificationUtils(context)
        val notification = notificationUtils.getNotificationBuilder().build()
        notificationUtils.getManager().notify( id: 150, notification)
    }
}
```

- Mendaftarkan Alarm Receiver di Android Manifest

- Membuat Notification Utils untuk mengatur komunikasi dengan Notification Manager, membuat channel notifikasi, dan set up notifikasi yang akan muncul

```
// Get Manager
fun getManager() : NotificationManager {
    if (manager == null) manager = getSystemService(Context.NOTIFICATION_SERVICE)
            as NotificationManager
    return manager as NotificationManager
}

fun getNotificationBuilder(): NotificationCompat.Builder {
    val intent = Intent( packageContext: this, MainActivity::class.java).apply {  this: Intent
        flags = Intent.FLAG_ACTIVITY_CLEAR_TASK
    }
    val pendingIntent = PendingIntent.getActivity(
         context: this,  requestCode: 0, intent,  flags: 0)
    return NotificationCompat.Builder(applicationContext, MYCHANNEL_ID)
        .setContentTitle("Alarm!")
        .setContentText("Your AlarmManager is working.")
        .setSmallIcon(R.drawable.ic_alarm_blue)
        .setColor(Color.YELLOW)
        .setContentIntent(pendingIntent)
        .setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION))
        .setAutoCancel(true)
}
```

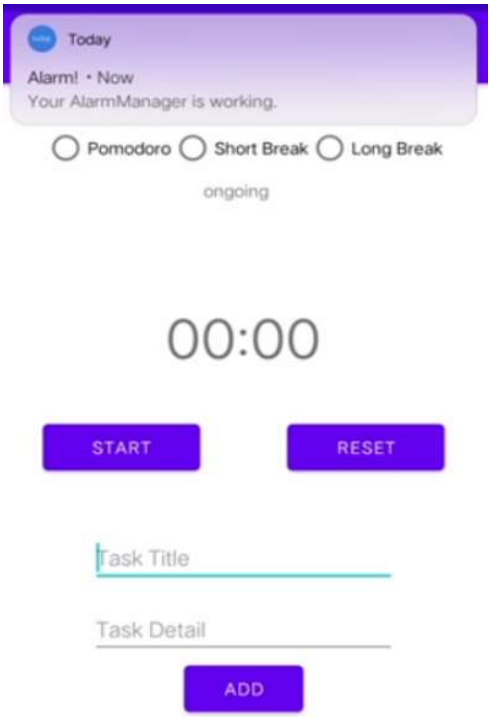
```
// Create channel for Android version 26+
@TargetApi(Build.VERSION_CODES.O)
private fun createChannels() {
    val channel = NotificationChannel(
        MYCHANNEL_ID, MYCHANNEL_NAME,
        NotificationManager.IMPORTANCE_HIGH)
    channel.enableVibration( vibration: true)

    getManager().createNotificationChannel(channel)
}
```

- Membuat fungsi startAlarm untuk memanggil alarmmanagernya. Perubahan dengan yang kemarin sekarang menggunakan alarmManager.set()

```
@RequiresApi(Build.VERSION_CODES.KITKAT)
    fun startAlarm(duration: Long) {
        val millis = SystemClock.uptimeMillis() + duration
        val alarmManager = getSystemService(Context.ALARM_SERVICE) as AlarmManager
        val intent = Intent( packageContext: this, AlarmReceiver::class.java)
        val pendingIntent = PendingIntent.getBroadcast( context: this,  requestCode: 0, intent,  flags: 0)
        Log.i( tag: "millis",millis.toString())
        alarmManager.set(
            AlarmManager.ELAPSED_REALTIME_WAKEUP,
             triggerAtMillis: SystemClock.elapsedRealtime() + duration,
            pendingIntent
        )
}
```

- Memanggil start di listener start button main activity. Perubahan dengan kemarin sekarang durasinya 30000 atau 30 detik.

```
startbutton.setOnClickListener {  it: View!
    startService(Intent( packageContext: this, BroadcastService::class.java))
    Log.i(TAG,  msg: "Started service")
    startAlarm( duration: 30000)
}
```


Today
Alarm! • Now
Your AlarmManager is working.
Pomodoro
Short Break
Long Break
ongoing
00:00
START
RESET
Task Title
Task Detail
ADD