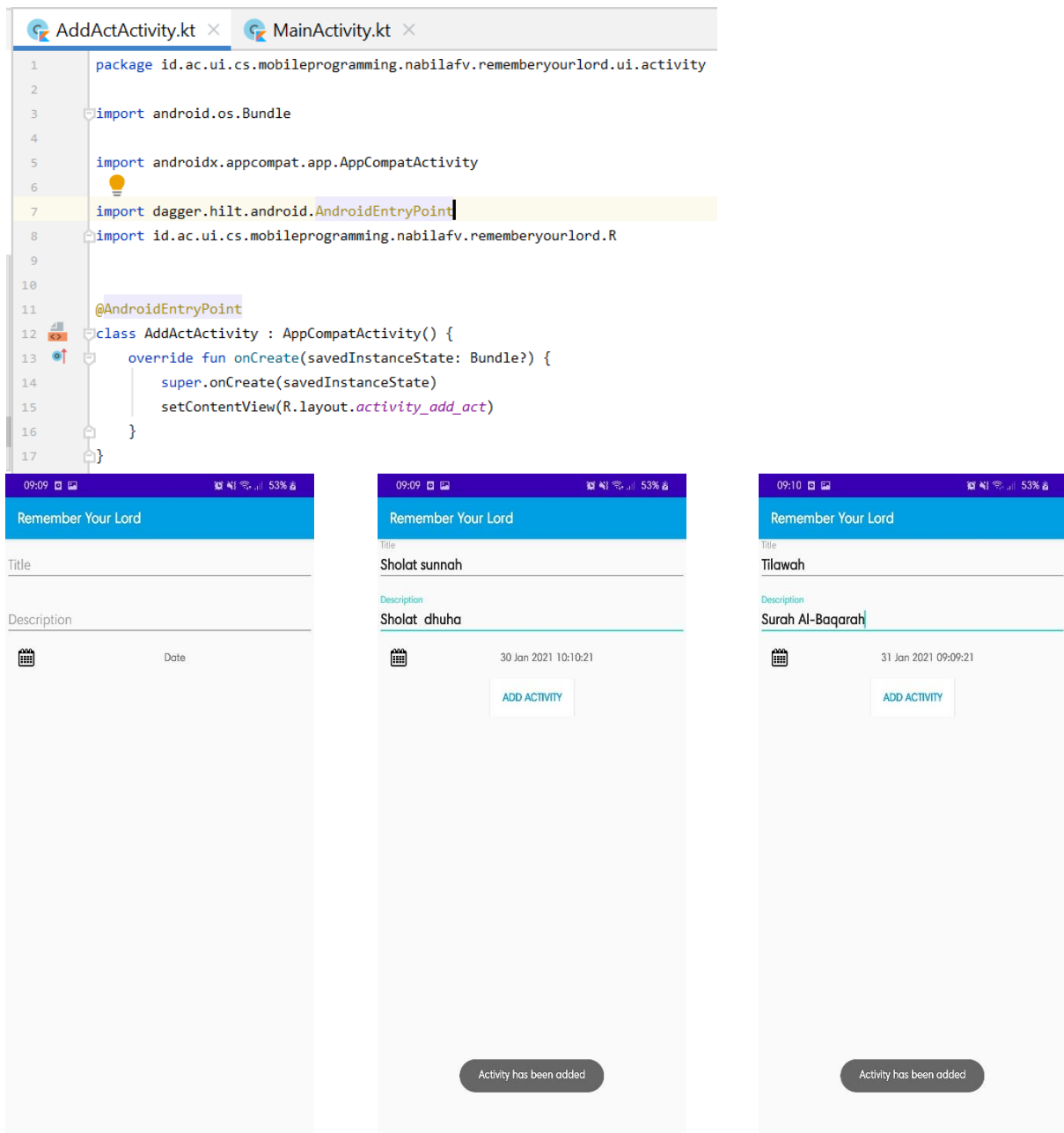


- Link Github: <https://github.com/nabilafebri/remember-your-lord>
- Link Mock-Up Awal:
<https://www.figma.com/file/VGTBLh4Wcu8bPCO5lJDaNd/TKTPL?node-id=0%3A1>
- Remember Your Lord: Aplikasi pencatat aktivitas ibadah harian. Aplikasi ini juga dapat menampilkan random surat beserta terjemahannya dan informasi cuaca.
- Stack Android Framework Standard
 - Activity untuk menambahkan “Activity” (aktivitas ibadah harian).



- Activity yang berfungsi untuk menampilkan informasi cuaca yang berasal dari WeatherFragment, random ayat Al-Qur'an beserta terjemahannya yang berasal dari QuranFragment, dan List Activity yang berasal dari ActListFragment. Activity ini juga berfungsi untuk menjadwalkan sebuah alarm yang berulang (Daily Notification).

```

18 @AndroidEntryPoint
19 class MainActivity : AppCompatActivity() {
20     private lateinit var actListViewModel: ActListViewModel
21     private lateinit var binding: ActivityMainBinding
22
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         binding = ActivityMainBinding.inflate(layoutInflater)
26         setContentView(binding.root)
27         binding.btnAddActivity.setOnClickListener { it: View!
28             startActivity(Intent( packageContext: this@MainActivity, AddActActivity::class.java))
29         }
30         actListViewModel = ViewModelProvider( owner: this).get(ActListViewModel::class.java)
31         actListViewModel.getSelectedActivity().observe( owner: this, { _ ->
32             if (findViewById<View>(R.id.fragment_activity_detail) == null) {
33                 val fragmentTransaction = supportFragmentManager.beginTransaction()
34                 fragmentTransaction.replace(R.id.layout_main_activity, ActDetailFragment())
35                 fragmentTransaction.addToBackStack( name: null)
36                 fragmentTransaction.commit()
37             }
38         })
39         setupDailyNotif()
40     }

```

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/layout_main_activity"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:background="?android:windowBackground"
9     tools:context=".MainActivity">
10
11     <androidx.fragment.app.FragmentContainerView
12         android:id="@+id/fragment_weather"
13         android:name="id.ac.ui.cs.mobileprogramming.nabilafv.rememberyourlord.ui.weather.WeatherFragment"
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19     <androidx.fragment.app.FragmentContainerView
20         android:id="@+id/fragment_quran"
21         android:name="id.ac.ui.cs.mobileprogramming.nabilafv.rememberyourlord.ui.quran.QuranFragment"
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         app:layout_constraintTop_toBottomOf="@id/fragment_weather"
25         app:layout_constraintLeft_toLeftOf="parent"
26         app:layout_constraintRight_toRightOf="parent"/>
27
28     <androidx.fragment.app.FragmentContainerView
29         android:id="@+id/fragment_activity_list"

```



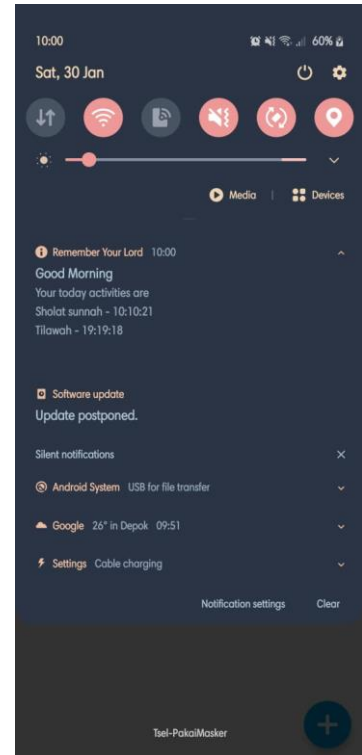
- Service & Remote Method
 - Alarm Manager untuk trigger notifikasi setiap hari, GoogleApi untuk mendapatkan lokasi pengguna untuk mengambil informasi cuaca, dan mengambil informasi cuaca dari API Weather.

```
private fun setupDailyNotif() {  
    val calendar = Calendar.getInstance()  
    calendar[Calendar.HOUR_OF_DAY] = 10  
    calendar[Calendar.MINUTE] = 0  
    calendar[Calendar.SECOND] = 0  
    val intent = Intent(applicationContext, NotificationReceiver::class.java)  
    val pendingIntent = PendingIntent.getBroadcast(  
        applicationContext,  
        requestCode = 0,  
        intent,  
        PendingIntent.FLAG_ONE_SHOT  
    )  
    val alarmManager = applicationContext.getSystemService(ALARM_SERVICE) as AlarmManager  
    alarmManager.setRepeating(  
        AlarmManager.RTC_WAKEUP,  
        calendar.timeInMillis,  
        AlarmManager.INTERVAL_DAY,  
        pendingIntent  
    )  
}
```

```
@AndroidEntryPoint  
class NotificationReceiver : BroadcastReceiver() {  
    @Inject  
    lateinit var activityRepository: ActivityRepositoryInterface  
  
    override fun onReceive(context: Context?, intent: Intent?) {  
        if (intent != null && context != null) {  
            var allActivity = ""  
            GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope  
                activityRepository.getTodayActivity().forEach { item ->  
                    if (item != null) {  
                        val date = Date(item.date)  
                        allActivity += "${item.title} - ${  
                            SimpleDateFormat.getTimeInstance().format(date)}\n"  
                    }  
                }  
            }  
            withContext(Dispatchers.Main) { this: CoroutineScope  
                NotificationCreator(context, allActivity).createNotification()  
            }  
        }  
    }  
}
```

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
    super.onViewCreated(view, savedInstanceState)  
    weatherViewModel = ViewModelProvider(requireActivity()).get(WeatherViewModel::class.java)  
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(requireView().context)
```

```
@SuppressWarnings("MissingPermission")  
fun getCurrentLocation() {  
    fusedLocationClient.lastLocation.addOnSuccessListener { location ->  
        if (location != null) {  
            weatherViewModel.fetchWeatherInfo(location.latitude, location.longitude)  
        }  
    }  
}
```



```

@Singleton
@Provides
fun provideOkHttpClient(): OkHttpClient {
    return if (BuildConfig.DEBUG) {
        val httpLoggingInterceptor = HttpLoggingInterceptor()
        httpLoggingInterceptor.setLevel(HttpLoggingInterceptor.Level.BODY)
        OkHttpClient.Builder().addInterceptor(httpLoggingInterceptor).build()
    } else {
        OkHttpClient.Builder().build()
    }
}

@Singleton
@Provides
fun provideRetrofit(okHttpClient: OkHttpClient, BASE_URL: String): Retrofit {
    return Retrofit.Builder()
        .addConverterFactory(
            GsonConverterFactory.create(
                GsonBuilder()
                    .setFieldNamingPolicy(FieldNamingPolicy.LOWER_CASE_WITH_UNDERSCORES)
                    .setPrettyPrinting()
                    .create()
            )
        )
        .baseUrl(BASE_URL)
        .client(okHttpClient)
        .build()
}

```



- Content Provider
 - Menyimpan Activity pada lokal kalender.

```

@SuppressLint(...)
fun addActToCal(
    context: Context,
    title: String,
    desc: String,
    date: Long
): Long? {
    var calId: Long? = getCalendarId(context)
    if (calId == -1L) {
        calId = makeCalendar(context)
    }

    val values = ContentValues()
    values.put(CalendarContract.Events.DTSTART, date)
    values.put(CalendarContract.Events.DTEND, date)
    values.put(CalendarContract.Events.TITLE, title)
    values.put(CalendarContract.Events.DESRIPTION, desc)
    values.put(CalendarContract.Events.CALENDAR_ID, calId)
    values.put(CalendarContract.Events.ORGANIZER, "nabilafv24@gmail.com")
    values.put(CalendarContract.Events.EVENT_TIMEZONE, "Asia/Jakarta")

    val uri: Uri? = context.contentResolver.insert(
        CalendarContract.Events.CONTENT_URI,
        values
    )
    addActViewModel.finishAddingToCalendar()
    return uri?.lastPathSegment?.toLong()
}

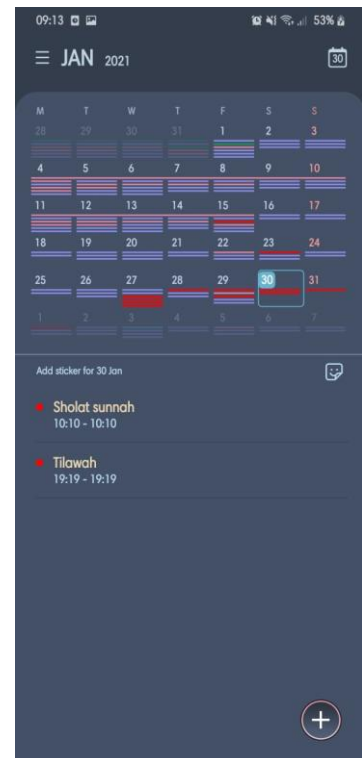
```



```
private fun getCalendarId(context: Context): Long {
    val projection = arrayOf(CalendarContract.Calendars._ID)
    val selection = CalendarContract.Calendars.ACCOUNT_NAME + " = ? AND " +
        CalendarContract.Calendars.ACCOUNT_TYPE + " = ? "
    val selArgs = arrayOf(calendarName, CalendarContract.ACCOUNT_TYPE_LOCAL)
    val cursor: Cursor? = context.contentResolver.query(
        CalendarContract.Calendars.CONTENT_URI,
        projection,
        selection,
        selArgs,
        sortOrder = null
    )
    if (cursor != null) {
        if (cursor.moveToFirst()) {
            return cursor.getLong( columnIndex = 0)
        }
    }
    return -1
}

private fun makeCalendar(context: Context): Long? {
    val values = ContentValues()
    values.put(
        CalendarContract.Calendars.ACCOUNT_NAME,
        calendarName
    )
    values.put(
        CalendarContract.Calendars.ACCOUNT_TYPE,
        CalendarContract.ACCOUNT_TYPE_LOCAL
    )
    values.put(
        CalendarContract.Calendars.NAME,
        "RYL - Activity Calendar"
    )
}

val builder: Uri.Builder = CalendarContract.Calendars.CONTENT_URI.buildUpon()
builder.appendQueryParameter(
    CalendarContract.Calendars.ACCOUNT_NAME,
    calendarName
)
builder.appendQueryParameter(
    CalendarContract.Calendars.ACCOUNT_TYPE,
    CalendarContract.ACCOUNT_TYPE_LOCAL
)
builder.appendQueryParameter(
    CalendarContract.Calendars.CALLER_IS_SYNCADAPTER,
    "true"
)
val uri: Uri? = context.contentResolver.insert(builder.build(), values)
return uri?.LastPathSegment?.toLong()
}
```



- Broadcast Receiver
 - Menampilkan notifikasi setelah Alarm Manager selesai. BroadcastReceiver akan menangani PendingIntent.

```

@AndroidEntryPoint
class NotificationReceiver : BroadcastReceiver() {
    @Inject
    lateinit var activityRepository: ActivityRepositoryInterface

    override fun onReceive(context: Context?, intent: Intent?) {
        if (intent != null && context != null) {
            var allActivity = ""
            GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
                activityRepository.getTodayActivity().forEach { item ->
                    if (item != null) {
                        val date = Date(item.date)
                        allActivity += "${item.title} - ${
                            SimpleDateFormat.getTimeInstance().format(date)}\n"
                    }
                }
                withContext(Dispatchers.Main) { this: CoroutineScope
                    NotificationCreator(context, allActivity).createNotification()
                }
            }
        }
    }
}

```

```

private fun setupDailyNotif() {
    val calendar = Calendar.getInstance()
    calendar[Calendar.HOUR_OF_DAY] = 10
    calendar[Calendar.MINUTE] = 0
    calendar[Calendar.SECOND] = 0
    val intent = Intent(applicationContext, NotificationReceiver::class.java)
    val pendingIntent = PendingIntent.getBroadcast(
        applicationContext,
        requestCode = 0,
        intent,
        PendingIntent.FLAG_ONE_SHOT
    )
    val alarmManager = applicationContext.getSystemService(ALARM_SERVICE) as AlarmManager
    alarmManager.setRepeating(
        AlarmManager.RTC_WAKEUP,
        calendar.timeInMillis,
        AlarmManager.INTERVAL_DAY,
        pendingIntent
    )
}

```

- Async Task
 - Mengambil data activity setiap hari, mengambil activity yang belum diselesaikan, dan menulis di database.

```
class ActivityRepository @Inject constructor(private val activityDao: ActivityDao) :
    ActivityRepositoryInterface {
    override suspend fun insertActivity(title: String, description: String, date: Long) {
        activityDao.insertActivity(
            Activity(UUID.randomUUID().toString(), title, description, isDone: false, date)
        )
    }

    override suspend fun updateActivity(idAct: String) {
        activityDao.updateActivity(idAct)
    }

    override fun getTodayActivity(): List<Activity?> {
        val calendar = Calendar.getInstance()
        val year = calendar.get(Calendar.YEAR)
        val month = calendar.get(Calendar.MONTH)
        val day = calendar.get(Calendar.DAY_OF_MONTH)
        calendar.set(year, month, day, hourOfDay: 0, minute: 0, second: 0)
        val todayEpoch = calendar.timeInMillis
        calendar.add(Calendar.MILLISECOND, amount: 1000 * 60 * 60 * 24)
        val tomorrowEpoch = calendar.timeInMillis
        return activityDao.getTodayActivity(todayEpoch, tomorrowEpoch)
    }

    override fun getAllUndoneActivity(): List<Activity?> {
        return activityDao.getAllUndoneActivity()
    }
}

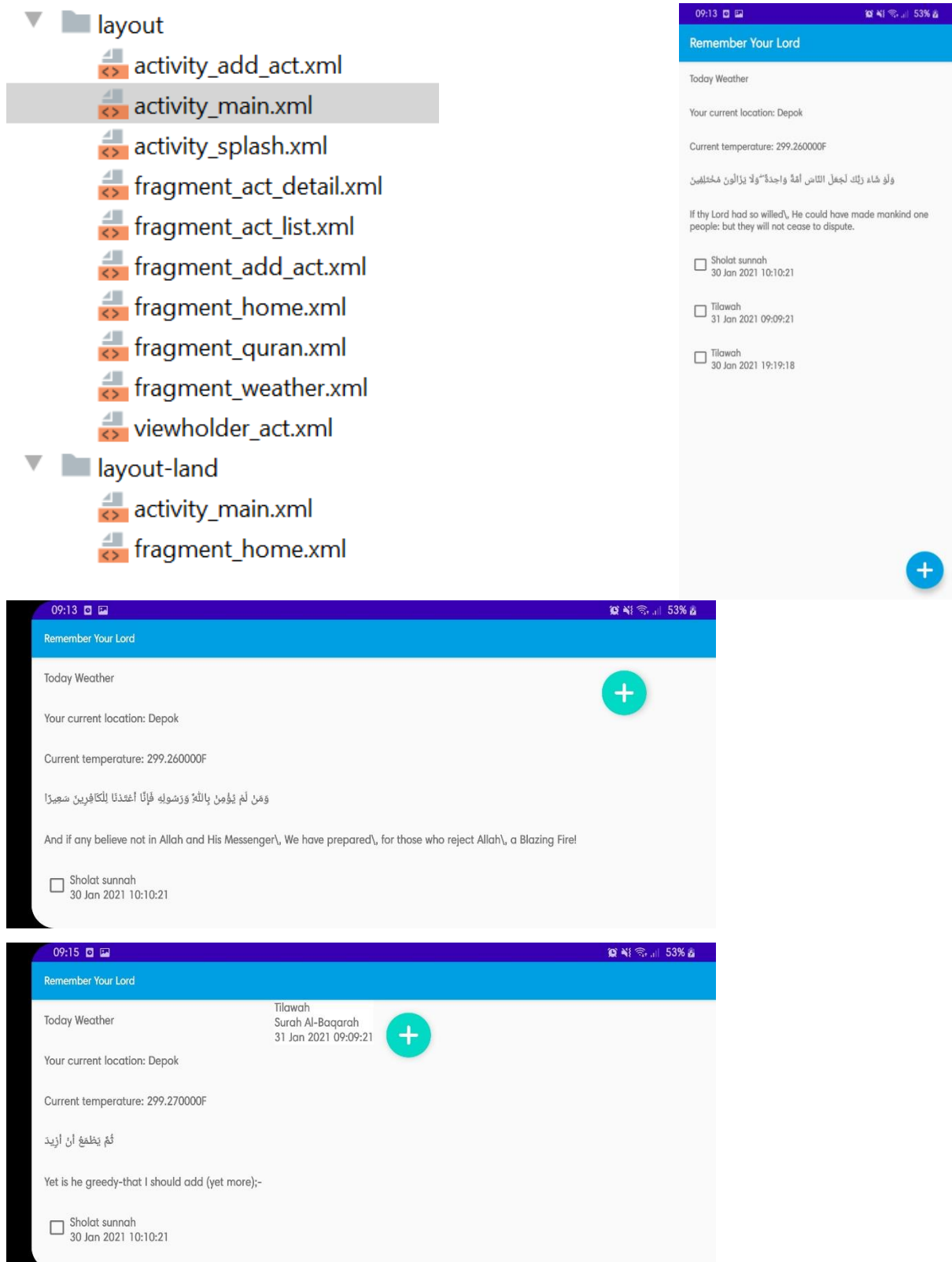
GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
    activityRepository.getTodayActivity().forEach { item ->
        if (item != null) {
            val date = Date(item.date)
            allActivity += "${item.title} - ${
                SimpleDateFormat.getTimeInstance().format(date)}\n"
        }
    }
    withContext(Dispatchers.Main) { this: CoroutineScope
        NotificationCreator(context, allActivity).createNotification()
    }
}

private fun getAllUndoneActivityAsync() {
    viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
        allUndoneActivity.postValue(State<List<Activity?>>().loading())
        val undoneActivity = activityRepository.getAllUndoneActivity()
        allUndoneActivity.postValue(State<List<Activity?>>().success(undoneActivity))
    }
}

fun addActivity(title: String, description: String, date: Long) {
    isInsert.value = true
    activity.value = State<Activity?>().loading()
    viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
        activityRepository.insertActivity(title, description, date)
    }
}
```

- Multi Layout

- Terdapat dua layout berbeda untuk menampilkan MainActivity (orientation horizontal dan vertical). Detail Aktivitas dapat dilihat pada layout main activity ketika orientationnya landscape.



- Multi Language

- Menggunakan string resource Bahasa Indonesia dan Bahasa Jerman.

Defaultnya Bahasa Inggris. Selain itu, terjemahan Quran juga disesuaikan dengan default language yang digunakan (terdapat Bahasa Indonesia, defaultnya Bahasa Inggris).

```

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    quranTextViewModel = ViewModelProvider(requireActivity()).get(QuranTextViewModel::class.java)
    enTransViewModel = ViewModelProvider(requireActivity()).get(EnTransViewModel::class.java)
    idTransViewModel = ViewModelProvider(requireActivity()).get(IdTransViewModel::class.java)

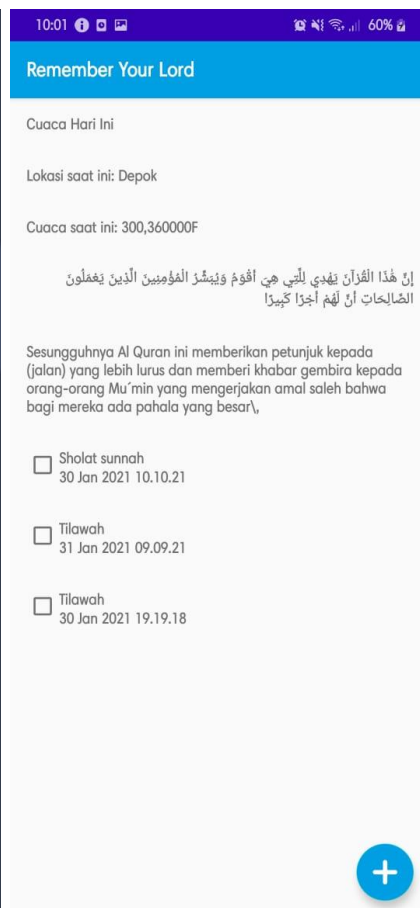
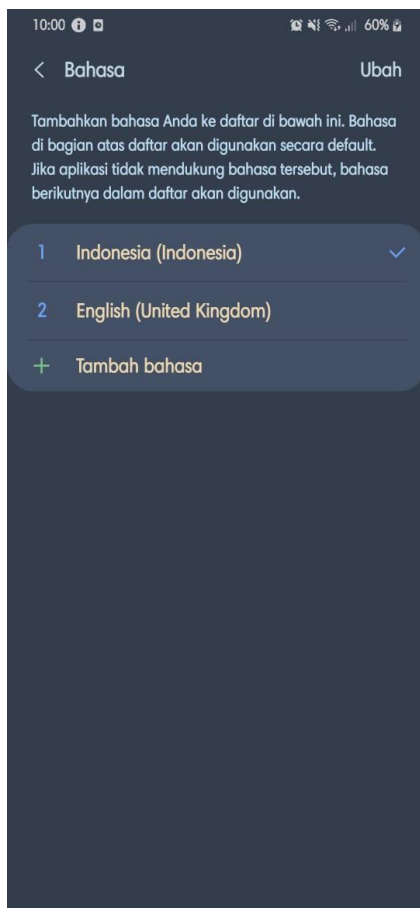
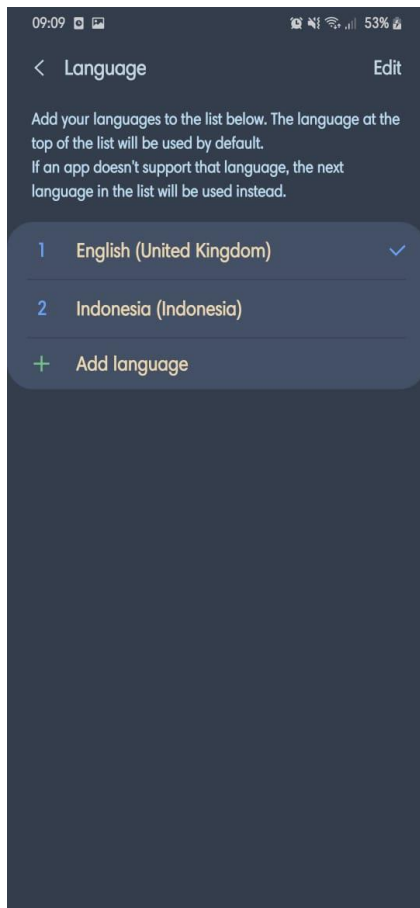
    val ayahNum = (0..6235).random()

    quranTextViewModel.getQuranText(ayahNum).observe(viewLifecycleOwner) { quranText ->
        binding.quranText.text = quranText.text
    }

    val lang = Locale.getDefault().getLanguage()
    if (lang == "in") {
        idTransViewModel.getIdTrans(ayahNum).observe(viewLifecycleOwner) { idTrans ->
            binding.quranTrans.text = idTrans.text
        }
    } else {
        enTransViewModel.getEnTrans(ayahNum).observe(viewLifecycleOwner) { enTrans ->
            binding.quranTrans.text = enTrans.text
        }
    }
}

```

	Edit translations for ... Open editor Hide notification	Edit translations for ... Open editor Hide notification	Edit translations for all locale... Open editor Hide notification
fragment_home.xml			
mipmap-anydpi-v26			
mipmap-hdpi			
mipmap-mdpi			
mipmap-xhdpi			
mipmap-xxhdpi			
values			
colors.xml			
dimens.xml			
strings.xml			
styles.xml			
values-de			
strings.xml			
values-in			
strings.xml			
ic_launcher-web.png			
playstore-icon.png			



- MVVM
 - Menggunakan ViewModel untuk Activity, Teks Quran, Terjemahan Quran, dan informasi cuaca. MVVM dilakukan dengan DI menggunakan Hilt.

```
@HiltViewModel
class QuranTextViewModel @Inject constructor(private val quranTextRepository: QuranTextRepository) :
    ViewModel() {
    private val _selectedQuranText = MutableLiveData<QuranText>()
    private val _allQuranTexts = MutableLiveData<List<QuranText>>()

    val allQuranTexts: LiveData<List<QuranText>> get() = _allQuranTexts;

    fun getQuranText(id: Int): LiveData<QuranText> {
        viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
            _selectedQuranText.postValue(quranTextRepository.getQuranText(id))
        }
        return _selectedQuranText
    }

    private fun getAllQuranTexts() {
        viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
            _allQuranTexts.postValue(quranTextRepository.getAllQuranTexts())
        }
    }

    init {
        getAllQuranTexts()
    }
}
```

```
@HiltViewModel
class WeatherViewModel @Inject constructor(private val weatherRepository: WeatherRepository) :
    ViewModel() {
    private val weatherResponse = MutableLiveData<State<WeatherResponse>>()
    private val isConnectedToNetwork = MutableLiveData<Boolean>()

    fun getWeatherResponse(): LiveData<State<WeatherResponse>> {
        return weatherResponse
    }

    fun getIsConnectedToNetwork(): LiveData<Boolean> {
        return isConnectedToNetwork
    }

    fun updateNetwork(isConnected: Boolean) {
        isConnectedToNetwork.postValue(isConnected)
    }

    fun fetchWeatherInfo(lat: Double, lon: Double) {
        viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
            weatherResponse.postValue(State<WeatherResponse>().loading())
            weatherRepository.getWeather(lat, lon).let { response ->
                if (response.isSuccessful) {
                    weatherResponse.postValue(State<WeatherResponse>().success(response.body()))
                }
            }
        }
    }

    init {
        weatherResponse.postValue(State<WeatherResponse>().init())
    }
}
```

```

class ActListViewModel @Inject constructor(private val activityRepository: ActivityRepository) :
    ViewModel() {
        private val selectedActivity = MutableLiveData<Activity>()
        private val allUndoneActivity = MutableLiveData<State<List<Activity?>>>()

        fun getSelectedActivity(): LiveData<Activity> {
            return selectedActivity
        }

        fun getAllUndoneActivity(): LiveData<State<List<Activity?>>> {
            return allUndoneActivity
        }

        private fun getAllUndoneActivityAsync() {
            viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
                allUndoneActivity.postValue(State<List<Activity?>>>().loading())
                val undoneActivity = activityRepository.getAllUndoneActivity()
                allUndoneActivity.postValue(State<List<Activity?>>>().success(undoneActivity))
            }
        }

        fun selectActivity(selected: Activity) {
            selectedActivity.postValue(selected)
        }

        fun checkActivity(selected: Activity) {
            viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
                activityRepository.updateActivity(selected.id)
            }
        }

        init {
            allUndoneActivity.postValue(State<List<Activity?>>>().init())
            getAllUndoneActivityAsync()
        }
    }

```

```

class AddActViewModel @Inject constructor(private val activityRepository: ActivityRepository) :
    ViewModel() {
        private val activity = MutableLiveData<State<Activity?>>()
        private val isInsert = MutableLiveData<Boolean>()
        private val isDatePicked = MutableLiveData<Boolean>()

        fun getActivity(): LiveData<State<Activity?>> {
            return activity
        }

        fun getIsDatePicked(): LiveData<Boolean> {
            return isDatePicked
        }

        fun setIsDatePicked() {
            isDatePicked.value = true
        }

        fun addActivity(title: String, description: String, date: Long) {
            isInsert.value = true
            activity.value = State<Activity?>().loading()
            viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
                activityRepository.insertActivity(title, description, date)
            }
        }

        fun finishAddingToCalendar() {
            activity.value = State<Activity?>().success(data: null)
        }

        init {
            activity.value = State<Activity?>().init()
            isInsert.value = false
            isDatePicked.value = false
        }
    }

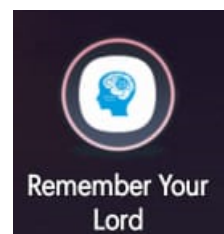
```

- Assets
 - Menggunakan string resource pada strings.xml dan custom icon launcher.

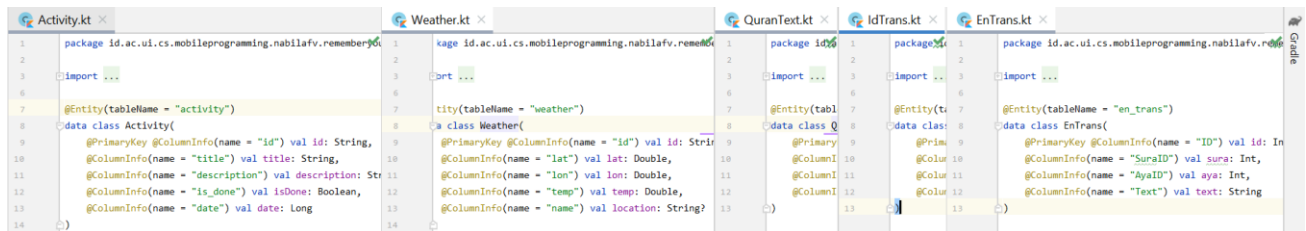
```
<resources>
    <string name="app_name">Remember Your Lord</string>
    <string name="title_home">Home</string>
    <string name="title_read">Read</string>
    <string name="title_faves">Faves</string>
    <string name="title_weather">Today Weather</string>
    <string name="act_title">Title</string>
    <string name="act_desc">Description</string>
    <string name="act_date">Date</string>
    <string name="btn_add_act">Add Activity</string>
    <string name="add_act_success">Activity has been added</string>
    <string name="weather_warning">No Connection, can\'t get weather and location data</string>
    <string name="weather_location">Your current location: %1$s</string>
    <string name="weather_temperature">Current temperature: %1$fF</string>
    <string name="weather_failed">Can\'t fetch weather data</string>
    <string name="card_sholat">Prayer Schedule</string>
    <string name="card_aktivitas">Daily Activity</string>
    <string name="card_kajian">Study Schedule</string>
    <string name="card_catatan">Study Notes</string>
</resources>
```

```
binding.weatherWarning.text = ""
binding.weatherLocation.text = getString(
    R.string.weather_location,
    response.data.name
)
binding.weatherTemperature.text = getString(
    R.string.weather_temperature,
    response.data.main.temp
)
```

```
AndroidManifest.xml
10 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
11
12 <application
13     android:name=".RememberYourLordApp"
14     android:allowBackup="true"
15     android:icon="@mipmap/ic_launcher"
16     android:label="@string/app_name"
17     android:roundIcon="@mipmap/ic_launcher_round"
18     android:supportRtl="true"
19     android:theme="@style/AppTheme">
```



- Data Persistence
 - Terdapat entity Activity, Weather, QuranText, EnTranslation, dan IdTranslation pada Room Database. Dilakukan pre-populate menggunakan database sqlite yang sebelumnya didapat dari database online (lupa dari mana sumbernya) untuk Room database QuranText, EnTranslation, dan IdTranslation.



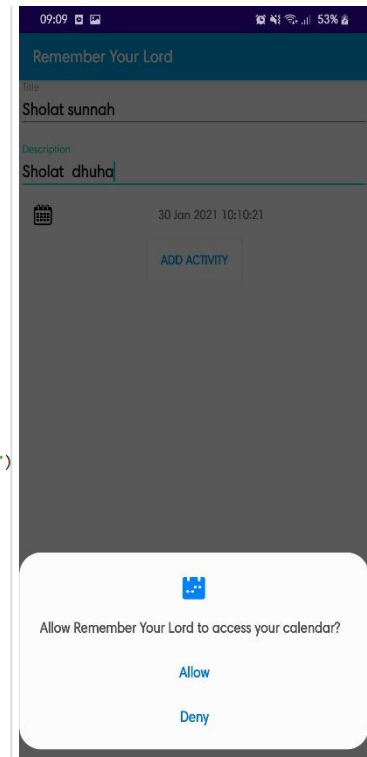
```
@Singleton
@Provides
fun provideDatabase(@ApplicationContext appContext: Context): RememberYourLordDatabase {
    return Room.databaseBuilder(
        appContext,
        RememberYourLordDatabase::class.java,
        name: "ryl-db"
    )
        .fallbackToDestructiveMigration()
        .allowMainThreadQueries()
        .build()
}
```

```
@Singleton
@Provides
fun provideEnTransDatabase(@ApplicationContext appContext: Context): EnTransDatabase {
    return Room.databaseBuilder(
        appContext,
        EnTransDatabase::class.java,
        name: "en-trans-db"
    )
        .createFromAsset(databaseFilePath: "database/en.db")
        .fallbackToDestructiveMigration()
        .build()
}
```

- Runtime Permission
 - Request permission: akses lokasi (memanggil API Weather) dan calendar (menambahkan Activity)

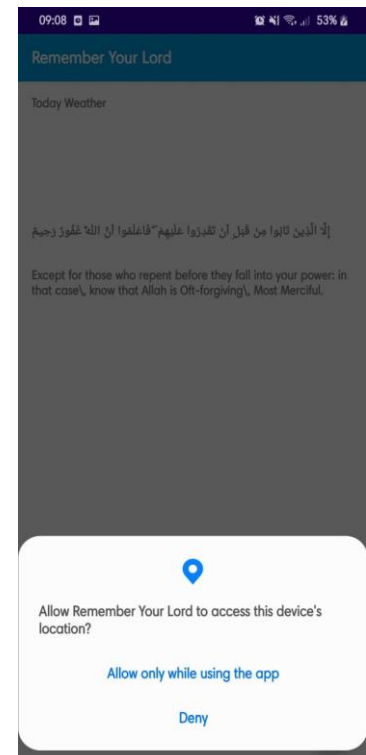
```
private val requestPermissions = registerForActivityResult(RequestMultiplePermissions())
{ isGranted ->
    if (!isGranted.containsValue(false)) {
        submitAct(requireView().context)
    }
}

private fun checkCalendarPermissonAndAddActToCal(context: Context) {
    when {
        ((PermissionChecker.checkSelfPermission(context, Manifest.permission.READ_CALENDAR)
            == PermissionChecker.PERMISSION_GRANTED) &&
            (PermissionChecker.checkSelfPermission(
                context, Manifest.permission.WRITE_CALENDAR
            )
                == PermissionChecker.PERMISSION_GRANTED)) -> {
            submitAct(context)
        }
        shouldShowRequestPermissionRationale( permission: "Want to add activity to your calender")
        -> {
        }
        else -> {
            val calendarPermissions = arrayOf(
                Manifest.permission.READ_CALENDAR,
                Manifest.permission.WRITE_CALENDAR
            )
            requestPermissions.launch(calendarPermissions)
        }
    }
}
```



```
private val requestPermissions = registerForActivityResult(RequestMultiplePermissions())
{ isGranted ->
    if (!isGranted.containsValue(false)) {
        getCurrentLocation()
    }
}

private fun checkLocationPermission(context: Context) {
    when {
        ((PermissionChecker.checkSelfPermission(
            context,
            Manifest.permission.ACCESS_COARSE_LOCATION
        ) == PermissionChecker.PERMISSION_GRANTED)
            || (PermissionChecker.checkSelfPermission(
                context,
                Manifest.permission.ACCESS_FINE_LOCATION
            ) == PermissionChecker.PERMISSION_GRANTED))
        -> {
            getCurrentLocation()
        }
        shouldShowRequestPermissionRationale( permission: "To get weather info") -> {
        }
        else -> {
            val locationPermissions = arrayOf(
                Manifest.permission.ACCESS_COARSE_LOCATION,
                Manifest.permission.ACCESS_FINE_LOCATION
            )
            requestPermissions.launch(locationPermissions)
        }
    }
}
```



- JNI
 - Menggunakan fungsi native C untuk animasi OpenGL pada splash screen.

```

237
238 static Renderer* g_renderer = NULL;
239
240 extern "C" {
241     JNIEXPORT void JNICALL Java_id_ac_ui_cs_mobileprogramming_nabilafv_rememberyourlord_ui_gle_GLES3JNIlib_init(JNIEnv* env, jobject obj);
242     JNIEXPORT void JNICALL Java_id_ac_ui_cs_mobileprogramming_nabilafv_rememberyourlord_ui_gle_GLES3JNIlib_resize(JNIEnv* env, jobject obj, jint width, jint height);
243     JNIEXPORT void JNICALL Java_id_ac_ui_cs_mobileprogramming_nabilafv_rememberyourlord_ui_gle_GLES3JNIlib_step(JNIEnv* env, jobject obj);
244 }
245
246 #if !defined(DYNAMIC_ES3)
247 static GLboolean gl3stubInit() {
248     return GL_TRUE;
249 }
250 #endif
251
252 JNIEXPORT void JNICALL
253 Java_id_ac_ui_cs_mobileprogramming_nabilafv_rememberyourlord_ui_gle_GLES3JNIlib_init(JNIEnv* env, jobject obj) {
254     if (g_renderer) {
255         delete g_renderer;
256         g_renderer = NULL;
257     }
258
259     printGLString( name: "Version", GL_VERSION);
260     printGLString( name: "Vendon", GL_VENDOR);
261     printGLString( name: "Renderer", GL_RENDERER);
262     printGLString( name: "Extensions", GL_EXTENSIONS);
263
264     const char* versionStr = (const char*)glGetString(GL_VERSION);
265     if (strstr(versionStr, "OpenGL ES 3.") && gl3stubInit()) {
266         g_renderer = createES3Renderer();
267     } else if (strstr(versionStr, "OpenGL ES 2.") {
268         g_renderer = createES2Renderer();
269     } else {
270         ALOGE("Unsupported OpenGL ES version");
271     }
272 }
273
274 JNIEXPORT void JNICALL
275 Java_id_ac_ui_cs_mobileprogramming_nabilafv_rememberyourlord_ui_gle_GLES3JNIlib_resize(JNIEnv* env, jobject obj, jint width, jint height) {
276     if (g_renderer) {
277         g_renderer->resize(width, height);
278     }
279 }
280
281 JNIEXPORT void JNICALL
282 Java_id_ac_ui_cs_mobileprogramming_nabilafv_rememberyourlord_ui_gle_GLES3JNIlib_step(JNIEnv* env, jobject obj) {
283     if (g_renderer) {
284         g_renderer->render();
285     }
286 }
    
```

```

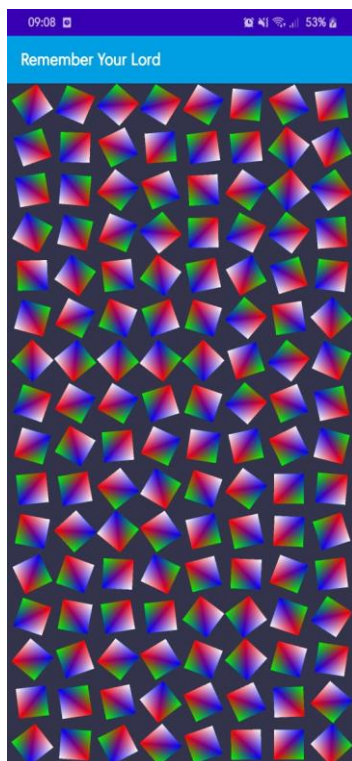
GLS3JNIView.kt  GLS3JNIlib.kt  GLS3JNIActivity.kt
package id.ac.ui.cs.mobileprogramming.nabilafv.rememberyourlord.ui.gle

object GLES3JNIlib {
    external fun init()
    external fun resize(width: Int, height: Int)
    external fun step()

    init {
        System.loadLibrary( libname: "gles3jni")
    }
}
    
```


- OpenGL
 - Menampilkan animasi OpenGL pada splash screen.

```
class GLES3JNIView(context: Context?) : GLSurfaceView(context) {  
    private class Renderer : GLSurfaceView.Renderer {  
        override fun onDrawFrame(gl: GL10) {  
            step()  
        }  
  
        override fun onSurfaceChanged(gl: GL10, width: Int, height: Int) {  
            resize(width, height)  
        }  
  
        override fun onSurfaceCreated(gl: GL10, config: EGLConfig) {  
            init()  
        }  
    }  
}  
  
companion object {  
    private const val TAG = "GLES3JNI"  
    private const val DEBUG = true  
}  
  
init {  
    // Pick an EGLConfig with RGB8 color, 16-bit depth, no stencil,  
    // supporting OpenGL ES 2.0 or later backwards-compatible versions.  
    setEGLConfigChooser( redSize: 8, greenSize: 8, blueSize: 8, alphaSize: 0, depthSize: 16, stencilSize: 0)  
    setEGLContextClientVersion(3)  
    setRenderer(Renderer())  
}
```



- Connectivity Manager
 - Mengambil data cuaca hanya jika menggunakan Wifi.

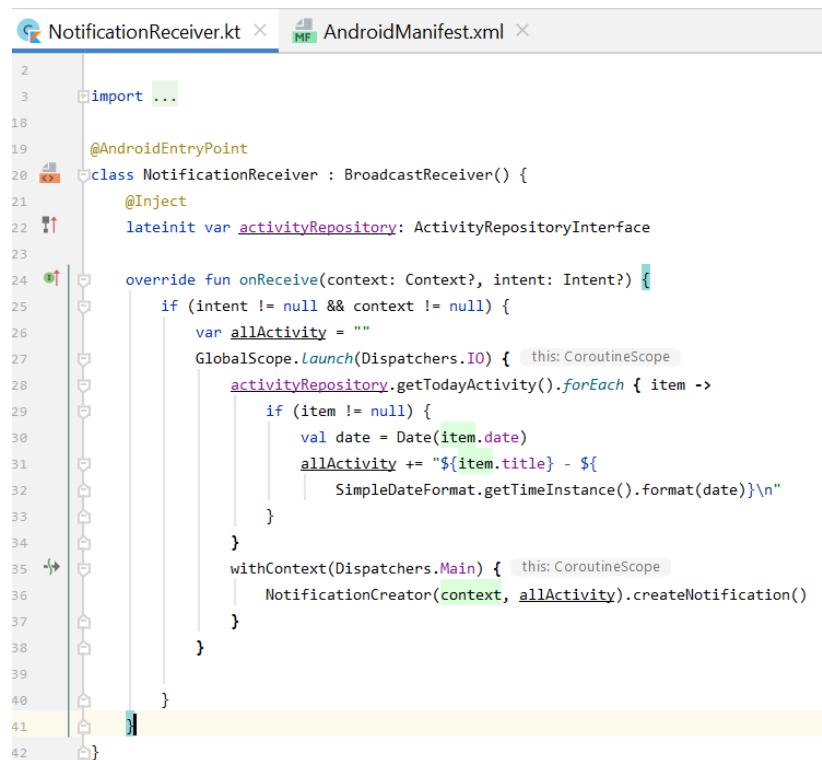
```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    weatherViewModel = ViewModelProvider(requireActivity()).get(WeatherViewModel::class.java)
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(requireView().context)

    weatherViewModel.getIsConnectedToNetwork().observe(viewLifecycleOwner) { item ->
        if (item) {
            checkLocationPermission(view.context)
        } else {
            binding.weatherWarning.text = "No Connection, can't get weather and location data"
        }
    }

    val connectivityManager =
        view.getContext().getSystemService(Context.CONNECTIVITY_SERVICE) as ConnectivityManager

    weatherViewModel.updateNetwork(!connectivityManager.isActiveNetworkMetered)
```

- Service Background
 - Menampilkan Activity setiap hari meskipun aplikasi sedang tidak dibuka.



```
NotificationReceiver.kt x AndroidManifest.xml x
2
3 import ...
18
19 @AndroidEntryPoint
20 class NotificationReceiver : BroadcastReceiver() {
21     @Inject
22     lateinit var activityRepository: ActivityRepositoryInterface
23
24     override fun onReceive(context: Context?, intent: Intent?) {
25         if (intent != null && context != null) {
26             var allActivity = ""
27             GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
28                 activityRepository.getTodayActivity().forEach { item ->
29                     if (item != null) {
30                         val date = Date(item.date)
31                         allActivity += "${item.title} - ${
32                             SimpleDateFormat.getTimeInstance().format(date)}\n"
33                     }
34                 }
35                 withContext(Dispatchers.Main) { this: CoroutineScope
36                     NotificationCreator(context, allActivity).createNotification()
37                 }
38             }
39         }
40     }
41 }
42 }
```

```
<application
    android:name=".RememberYourLordApp"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Remember Your Lord"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
    <activity
        android:name=".ui.gle.GLES3JNIActivity"
        android:configChanges="orientation|screenSize|keyboard|keyboardHidden"
        android:label="Remember Your Lord">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name="id.ac.ui.cs.mobileprogramming.nabilafv.rememberyourlord.MainActivity" />
    <activity android:name="id.ac.ui.cs.mobileprogramming.nabilafv.rememberyourlord.ui.activity.AddActActivity" />
    <receiver android:name="id.ac.ui.cs.mobileprogramming.nabilafv.rememberyourlord.receiver.NotificationReceiver" />
</application>
```

- Notifikasi
 - Memberikan notifikasi Activity setiap hari.

```
]internal class NotificationCreator(private val mContext: Context, private val activities: String) {
} fun createNotification() {
    val rUriIntent = Intent(mContext, MainActivity::class.java)
    rUriIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
    val pendingIntent = PendingIntent.getActivity(
        mContext, requestCode = 0, rUriIntent, PendingIntent.FLAG_UPDATE_CURRENT
    )
    val mBuilder = NotificationCompat.Builder(mContext, NOTIFICATION_CHANNEL_ID)
    mBuilder.setSmallIcon(R.drawable.ic_dialog_info).setContentTitle("Good Morning")
        .setStyle(
            NotificationCompat
                .BigTextStyle()
                .bigText( CS: "Your today activities are\n$activities")
        )
        .setAutoCancel(false)
        .setContentIntent(pendingIntent)
    val mNotificationManager =
        mContext.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val notificationChannel = NotificationChannel(
            NOTIFICATION_CHANNEL_ID,
            name: "My Notifications",
            NotificationManager.IMPORTANCE_HIGH
        )
        notificationChannel.enableLights( lights: true)
        notificationChannel.LightColor = Color.RED
        notificationChannel.enableVibration( vibration: true)
        notificationChannel.vibrationPattern = LongArrayOf(0, 1000, 500, 1000)
        mBuilder.setChannelId(NOTIFICATION_CHANNEL_ID)
        mNotificationManager.createNotificationChannel(notificationChannel)
    }
}
```

