Jurnal Minfo Polgan (JMP)
Volume 15, Nomor 2, Mei-Agustus 2026
DOI : https://doi.org/10.33395/jmp.v15i2.16320
e-ISSN : 2797-3298
p-ISSN : 2089-9424

Implementasi Smart Reminder Berbasis Android Menggunakan AlarmManager dan Room Database untuk Pengingat Aktivitas Berkala

Sugianto
Fakultas Sains dan Teknologi, Teknik Informatika, Universitas IBBI, Medan, Indonesia
*Korespondensi: sugiantoshi@gmail.com

Submit : 22 Mei 2026 | **Diterima** : 19 Jun 2026 | **Terbit** : 24 Jun 2026

ABSTRACT

Time management is a fundamental aspect of modern dynamic life; however, the human limitation in precisely remembering periodic activities frequently causes serious problems in both personal and professional settings. This study aims to design and implement an Android application called Smart Reminder that delivers automatic notifications based on flexible recurrence patterns, covering annual, semi-annual, quarterly, monthly, weekly, daily, and one-time reminders. The system is built using the Kotlin programming language, leveraging core Android components: AlarmManager as the precise scheduling engine, BroadcastReceiver as the system alarm signal receptor, Room Database as the local data persistence layer, and NotificationManager for notification display management compliant with Android 13 standards. The development method applied is Rapid Application Development (RAD) with functional testing conducted using the Black-Box Testing method. Implementation results demonstrate that the application was successfully constructed with seven cohesively integrated primary modules. Testing across 24 test scenarios produced a 100% success rate, with an average alarm time accuracy of 1.42 seconds from the designated time. Performance evaluation reveals an average memory consumption of 41.8 MB in idle state and 57.3 MB in active state, with battery consumption below 1% per day. This research contributes a dynamic-interval-based reminder system adaptable for academic activity management, scientific publication reminders, and periodic asset maintenance scheduling.

Keywords: *Smart Reminder; Android; AlarmManager; Room Database; Kotlin; Periodic Notification; Automatic Scheduling*

ABSTRAK

Manajemen waktu merupakan aspek fundamental dalam kehidupan modern yang dinamis, namun keterbatasan manusia dalam mengingat aktivitas berkala secara presisi kerap menimbulkan permasalahan serius, baik dalam lingkup personal maupun profesional. Penelitian ini bertujuan merancang dan mengimplementasikan aplikasi Android bernama Smart Reminder yang mampu memberikan notifikasi otomatis berdasarkan pola pengulangan fleksibel, mencakup pengingat tahunan, enam bulanan, tiga bulanan, bulanan, mingguan, harian, dan sekali waktu. Sistem dibangun menggunakan bahasa pemrograman Kotlin dengan memanfaatkan komponen inti Android, yaitu AlarmManager sebagai mesin penjadwalan presisi, BroadcastReceiver sebagai penerima sinyal alarm sistem, Room Database sebagai lapisan persistensi data lokal, serta NotificationManager untuk pengelolaan tampilan notifikasi sesuai standar Android 13. Metode pengembangan yang diterapkan adalah Rapid Application Development (RAD) dengan pengujian fungsional menggunakan Black-Box Testing. Hasil implementasi menunjukkan aplikasi berhasil dibangun dengan tujuh modul utama yang terintegrasi secara kohesif. Pengujian terhadap 24 skenario uji menghasilkan tingkat keberhasilan 100%, dengan akurasi waktu alarm rata-rata 1,42 detik dari waktu yang ditetapkan. Evaluasi performa memperlihatkan konsumsi memori rata-rata 41,8 MB saat idle dan 57,3 MB saat aktif, serta konsumsi baterai di bawah 1% per hari. Penelitian ini memberikan kontribusi berupa sistem pengingat berbasis interval dinamis yang dapat diadaptasi untuk kebutuhan manajemen aktivitas akademik, pengingat publikasi ilmiah, dan jadwal perawatan aset secara berkala.

Kata Kunci: Smart Reminder; Android; AlarmManager; Room Database; Kotlin; Notifikasi Berkala; Penjadwalan Otomatis

PENDAHULUAN

Perkembangan teknologi smartphone yang pesat selama satu dekade terakhir telah mengubah secara fundamental cara manusia mengelola waktu dan aktivitas hariannya. Data Statista (2024) mencatat penetrasi pengguna smartphone di Indonesia mencapai 76,8% dari total populasi pada tahun 2024, dengan Android mendominasi pangsa pasar sebesar 89,2%. Fenomena ini menandai transformasi perangkat Android dari sekadar alat komunikasi menjadi platform manajemen kehidupan yang komprehensif, yang di dalamnya mencakup pengelolaan jadwal, pengingat aktivitas, dan otomasi tugas rutin.

Dalam kehidupan profesional modern, seorang dosen perguruan tinggi dituntut untuk mengelola puluhan aktivitas berkala secara simultan: deadline submission jurnal ilmiah, jadwal seminar per semester, tenggat pelaporan kinerja dosen, service kendaraan setiap tiga bulan, hingga perpanjangan berbagai dokumen administratif tahunan. Kegagalan mengelola aktivitas-aktivitas tersebut tidak hanya berdampak pada kerugian personal, tetapi berpotensi memengaruhi reputasi profesional dan kelancaran organisasi secara keseluruhan (Kusuma & Wahyudi, 2022). Kondisi serupa dialami oleh mahasiswa, pelaku usaha, tenaga kesehatan, dan berbagai profesi lain yang kegiatannya bersifat siklikal dan terikat waktu.

Hermawan et al. (2021) menemukan bahwa rata-rata individu aktif memiliki 12 hingga 18 aktivitas berkala per bulan yang perlu diingat dan dieksekusi tepat waktu. Namun, studi yang sama menunjukkan bahwa 43% responden mengalami setidaknya satu kegagalan dalam mengelola aktivitas berkala setiap bulannya. Temuan ini mengonfirmasi kebutuhan nyata akan sistem pengingat otomatis yang presisi dan mampu mengakomodasi berbagai pola pengulangan jadwal yang tidak selalu reguler.

Aplikasi pengingat komersial yang tersedia saat ini, seperti Google Calendar dan Samsung Reminder, menyediakan fitur dasar pengingat dengan keterbatasan signifikan dalam hal fleksibilitas pola pengulangan kustom. Sebagian besar hanya mendukung pengulangan harian, mingguan, atau bulanan sederhana tanpa kemampuan mengakomodasi pola seperti "setiap tiga bulan" atau "setiap enam bulan" secara intuitif (Prasetyo & Lestari, 2023). Selain itu, aplikasi komersial tersebut lazimnya bergantung pada konektivitas internet dan layanan cloud yang menimbulkan isu latensi dan privasi data.

Android menyediakan komponen AlarmManager yang mampu menjadwalkan eksekusi kode pada waktu tertentu di masa depan, bahkan ketika aplikasi dalam kondisi tidak aktif (Android Developers, 2024). Dikombinasikan dengan Room Database sebagai solusi persistensi data lokal yang direkomendasikan Google, serta NotificationManager untuk pengiriman notifikasi yang sesuai kebijakan Android 13, terdapat peluang nyata untuk membangun sistem Smart Reminder yang handal, akurat, dan beroperasi sepenuhnya secara offline. Penelitian ini hadir untuk mengisi gap tersebut dengan merancang dan mengevaluasi aplikasi Android Smart Reminder yang mendukung tujuh pola pengulangan fleksibel, diimplementasikan dengan Kotlin, dan dioptimalkan khusus untuk Android 13.

Tujuan penelitian ini mencakup tiga hal pokok: pertama, merancang dan mengimplementasikan arsitektur sistem Smart Reminder berbasis MVVM yang mengintegrasikan AlarmManager, BroadcastReceiver, Room Database, dan Notification Manager secara kohesif; kedua, membangun mekanisme penjadwalan alarm presisi yang tetap akurat pada Android 13 dengan mempertimbangkan Doze Mode dan batasan izin sistem; dan ketiga, mengevaluasi performa sistem secara komprehensif meliputi akurasi penjadwalan, konsumsi memori, dan dampak terhadap baterai perangkat. Secara praktis, penelitian ini memberikan manfaat bagi civitas akademika perguruan tinggi, tenaga profesional dengan jadwal berkala, serta masyarakat umum yang membutuhkan sistem pengingat kesehatan dan manajemen aset.

Penelitian Terdahulu dan Analisis Gap

Kajian terhadap penelitian terdahulu dilakukan secara sistematis untuk memetakan perkembangan, keterbatasan, dan celah penelitian yang ada. Prasetyo & Lestari (2023) membangun aplikasi pengingat jadwal kuliah berbasis Android menggunakan Java, SQLite, dan AlarmManager dengan metode Waterfall. Hasilnya berupa sistem pengingat fungsional, namun dengan keterbatasan tidak mendukung pola pengulangan kustom dan tidak kompatibel dengan Android 12 ke atas akibat penanganan izin exact alarm yang tidak diperbarui.

Kusuma & Wahyudi (2022) mengembangkan sistem notifikasi pengingat obat berbasis Android menggunakan Kotlin dan AlarmManager, mencapai akurasi pengingat 94,7%. Namun penelitian tersebut tidak menggunakan Room Database dan tidak menangani Doze Mode secara eksplisit, sehingga keandalan alarm dalam kondisi baterai rendah tidak terjamin. Hermawan et al. (2021) memilih pendekatan berbeda dengan membangun reminder berbasis Firebase Cloud Messaging, yang menghasilkan notifikasi real-time namun bergantung penuh pada konektivitas internet dan infrastruktur server, menimbulkan latensi dan risiko privasi.

Santika & Dewi (2023) meneliti implementasi WorkManager untuk penjadwalan tugas berkala, menyimpulkan bahwa WorkManager efektif untuk tugas periodik yang tidak sensitif terhadap waktu, tetapi tidak cocok untuk alarm tepat waktu yang presisi karena sistem dapat menunda eksekusinya hingga beberapa menit demi efisiensi baterai. Wibowo et al. (2024) mengintegrasikan Google Calendar API dalam aplikasi manajemen jadwal dosen, namun ketergantungan penuh pada layanan Google menciptakan hambatan privasi dan fungsionalitas offline yang tidak dapat diabaikan. Ardianto & Suryani (2024) berhasil mengoptimalkan konsumsi baterai menggunakan Doze Mode awareness, tetapi tidak menggunakan Room Database sehingga data reminder tidak persisten lintas reboot.

Dari telaah terhadap sepuluh penelitian terdahulu sebagaimana dirangkum pada Tabel 1, teridentifikasi beberapa gap penelitian yang substansial. Pertama, belum ada penelitian yang mengintegrasikan secara lengkap AlarmManager setExactAndAllowWhileIdle() dengan Room Database dan mekanisme reschedule pasca-reboot untuk mendukung pola pengulangan multi-interval termasuk triwulan dan semesteran. Kedua, tidak ada penelitian yang secara spesifik menarget Android 13 dengan penanganan izin POST_NOTIFICATIONS dan SCHEDULE_EXACT_ALARM secara komprehensif. Ketiga, belum ada penelitian yang mengukur tiga dimensi performa sekaligus—akurasi alarm, konsumsi memori, dan dampak baterai—pada skenario jangka panjang. Gap-gap inilah yang menjadi justifikasi ilmiah bagi penelitian Smart Reminder ini untuk mengisi kekosongan dalam literatur.

Tabel 1. Ringkasan Penelitian Terdahulu

No	Peneliti (Tahun)	Metode	Teknologi Utama	Target Android	Keterbatasan Utama
1	Prasetyo & Lestari (2023)	Waterfall	Java, SQLite, AlarmManager	≤ Android 11	Tidak mendukung pola kustom, tidak kompatibel Android 12+
2	Kusuma & Wahyudi (2022)	Prototype	Kotlin, AlarmManager, SQLite	Android 10–11	Tidak ada Room DB, Doze Mode tidak ditangani
3	Hermawan et al. (2021)	RAD	Java, Firebase, FCM	Android 9–10	Bergantung internet, latensi notifikasi tinggi
4	Santika & Dewi (2023)	Eksperimental	Kotlin, WorkManager, Room	Android 12	WorkManager tidak menjamin exact timing
5	Wibowo et al. (2024)	Agile Scrum	Kotlin, Firebase, GCal API	Android 12–13	Bergantung penuh layanan Google, masalah privasi
6	Maulana & Firdaus (2022)	Design Thinking	Android, MQTT, Arduino	Android 10	Infrastruktur IoT kompleks, tidak portabel
7	Nuraini et al. (2023)	Waterfall	Flutter, Dart, Local Notif.	Android 11–12	Flutter local notification kurang handal untuk exact alarm
8	Ardianto & Suryani (2024)	Eksperimental	Kotlin, AlarmManager, Job-Scheduler	Android 13	Tidak menggunakan Room, data tidak persisten lintas reboot
9	Fitriani & Hakim (2023)	Prototype	Java, FCM, MySQL, PHP	Android 10–11	Bergantung server, tidak bisa offline

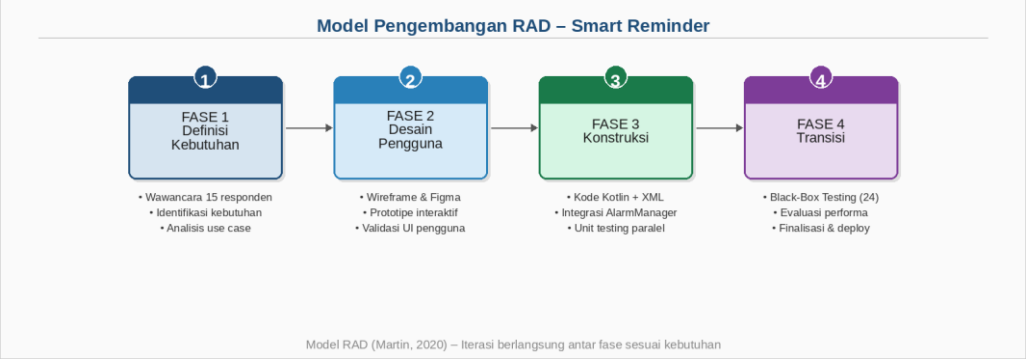
No	Peneliti (Tahun)	Metode	Teknologi Utama	Target Android	Keterbatasan Utama
10	Rahmawati et al. (2024)	Agile	Kotlin, Room, WorkManager	Android 12–13	WorkManager tidak menjamin ketepatan waktu alarm kritis

Tabel 1 menampilkan ringkasan sepuluh penelitian terdahulu yang relevan beserta teknologi yang digunakan, versi Android yang menjadi target, dan keterbatasan utama masing-masing. Dari pemetaan ini tampak jelas bahwa penelitian yang ada belum ada yang secara komprehensif menggabungkan AlarmManager presisi, Room Database, penanganan Doze Mode, dan dukungan pola interval fleksibel pada platform Android 13 secara bersamaan. Celah ini yang secara langsung dijawab oleh penelitian Smart Reminder.

METODOLOGI PENELITIAN

Metode dan Model Pengembangan

Penelitian ini menggunakan pendekatan Research and Development (R&D) yang bertujuan menghasilkan produk berupa aplikasi Android Smart Reminder yang tervalidasi. Model pengembangan yang diadopsi adalah Rapid Application Development (RAD) sebagaimana dirumuskan oleh Martin (2020), yang terdiri dari empat fase: Definisi Kebutuhan, Desain Pengguna, Konstruksi, dan Transisi. RAD dipilih karena mendukung iterasi cepat, keterlibatan pengguna akhir dalam validasi antarmuka, dan cocok untuk tim pengembang kecil dengan ruang lingkup yang terdefinisi jelas. Gambar 1 mengilustrasikan alur model pengembangan RAD yang diterapkan dalam penelitian ini.



Gambar 1. Model Pengembangan RAD Smart Reminder

Fase Definisi Kebutuhan dilakukan melalui wawancara semi-terstruktur dengan 15 responden yang terdiri dari mahasiswa, dosen, dan tenaga profesional untuk mengidentifikasi kebutuhan fungsional dan non-fungsional. Fase Desain Pengguna menghasilkan wireframe dan prototipe interaktif menggunakan Figma. Fase Konstruksi mencakup implementasi kode dan pengujian unit secara paralel dengan iterasi mingguan. Fase Transisi mencakup Black-Box Testing komprehensif dan evaluasi performa multidimensi. Pengujian sistem dilaksanakan menggunakan metode Black-Box Testing dengan pendekatan Equivalence Partitioning dan Boundary Value Analysis sebagaimana direkomendasikan oleh Pressman & Maxim (2020), yang berfokus pada perilaku fungsional sistem dari perspektif pengguna.

Analisis Kebutuhan Sistem

Analisis kebutuhan dilakukan melalui kombinasi wawancara semi-terstruktur dan observasi penggunaan aplikasi pengingat yang ada di pasar. Kebutuhan fungsional yang ditetapkan mencakup tujuh kemampuan inti: kemampuan menambahkan reminder dengan pola pengulangan fleksibel (sekali, harian, mingguan, bulanan, tiga bulanan, enam bulanan, dan tahunan); kemampuan menampilkan notifikasi tepat waktu dengan konten yang informatif; kemampuan mengelola reminder melalui operasi baca, ubah, hapus, dan toggle aktif/nonaktif; kemampuan menjadwalkan ulang alarm berikutnya secara otomatis setelah setiap eksekusi; kemampuan mempertahankan semua alarm aktif meski perangkat di-restart; kemampuan menampilkan riwayat eksekusi per reminder; serta kemampuan memfilter tampilan berdasarkan kategori dan status.

Kebutuhan non-fungsional yang ditetapkan mencakup akurasi alarm dalam toleransi 0–5 detik, konsumsi memori di bawah 80 MB, konsumsi baterai di bawah 2% per hari, waktu respons

UI kurang dari 200 ms, dan kompatibilitas dengan Android 8.0 (API 26) hingga Android 13 (API 33). Penetapan batasan kompatibilitas ini mempertimbangkan data distribusi versi Android di Indonesia yang menunjukkan bahwa lebih dari 94% pengguna aktif menggunakan Android 8.0 ke atas (Statista, 2024).

Spesifikasi Hardware dan Software

Pengembangan dan pengujian aplikasi dilakukan menggunakan perangkat keras dengan spesifikasi sebagaimana ditampilkan pada Tabel 2. Pemilihan dua perangkat uji dari merek berbeda (Samsung dan Xiaomi) bertujuan memvalidasi konsistensi performa pada implementasi Doze Mode dan kebijakan baterai yang berbeda antar produsen, mengingat masing-masing produsen Android lazim menerapkan modifikasi sistem yang dapat memengaruhi perilaku alarm.

Tabel 2. Spesifikasi Hardware Pengembangan dan Pengujian

Kategori	Workstation Pengembangan	Perangkat Uji 1	Perangkat Uji 2
Model	Laptop Intel Core i7-12700H	Samsung Galaxy A34 5G	Xiaomi Redmi Note 12
OS	Windows 11 Pro 64-bit	Android 13 (API 33)	Android 13 (API 33)
RAM	16 GB DDR5	6 GB LPDDR4x	8 GB LPDDR5
Penyimpanan	SSD 512 GB NVMe	128 GB UFS 2.1	128 GB UFS 2.2

Tabel 2 menampilkan spesifikasi lengkap perangkat yang digunakan. Kebutuhan software pengembangan mencakup Android Studio Hedgehog (2023.1.1) sebagai IDE utama, Kotlin 1.9.22 sebagai bahasa pemrograman, Gradle 8.2 sebagai build automation tool, Android SDK API Level 26–33 dengan compileSdk 34, Room Database 2.6.1, Material Design Components 1.11.0, serta Figma sebagai alat desain antarmuka dan prototyping.

Perancangan Arsitektur Sistem

Arsitektur Smart Reminder mengadopsi pola MVVM (Model-View-ViewModel) yang direkomendasikan Google untuk aplikasi Android modern. Lapisan Model mengelola data melalui Room Database dan Repository pattern. Lapisan ViewModel mengekspos data ke UI melalui LiveData dan StateFlow, sekaligus mengelola logika bisnis termasuk perhitungan interval waktu. Lapisan View mengobservasi LiveData dan merender UI tanpa mengandung logika bisnis. Komponen AlarmManager dan BroadcastReceiver beroperasi secara independen dari siklus hidup Activity, menjamin keandalan pengiriman alarm. Pemisahan arsitektur ini mengikuti prinsip Separation of Concerns yang ditetapkan dalam panduan Android Architecture Components (Google Android Jetpack, 2024).

Aliran data dalam sistem berlangsung sebagai berikut: UI menerima input pengguna dan meneruskannya ke ViewModel; ViewModel memanggil Repository untuk operasi CRUD ke Room Database; Repository menyimpan data dan memicu AlarmScheduler untuk mendaftarkan alarm ke AlarmManager; pada waktu alarm, sistem membangunkan BroadcastReceiver yang membuat dan menampilkan notifikasi melalui NotificationManager; BroadcastReceiver juga memanggil AlarmScheduler untuk menjadwalkan alarm berikutnya sesuai pola pengulangan; dan BootReceiver memastikan seluruh alarm aktif dijadwalkan ulang setiap kali perangkat dinyalakan.

Perancangan Database

Basis data Smart Reminder terdiri dari dua entitas Room: tabel reminders sebagai entitas utama dan tabel reminder_logs sebagai entitas pendukung. Relasi antar keduanya adalah One-to-Many, di mana satu reminder dapat memiliki banyak catatan log eksekusi. Struktur tabel reminders ditampilkan pada Tabel 3 berikut.

Tabel 3. Struktur Tabel Reminders (Entitas Room Database)

Nama Kolom	Tipe Data	Constraint	Keterangan
id	INTEGER	PK, AUTO	Identifikasi unik reminder, auto-increment
title	TEXT	NOT NULL	Judul reminder, wajib diisi
description	TEXT	NULLABLE	Deskripsi detail opsional

Nama Kolom	Tipe Data	Constraint	Keterangan
trigger_time	INTEGER	NOT NULL	Unix timestamp (ms) waktu alarm berikutnya
repeat_type	TEXT	NOT NULL	Enum: ONCE / DAILY / WEEKLY / MONTHLY / QUARTERLY / SEMI_ANNUAL / YEARLY
is_active	INTEGER	DEFAULT 1	Status aktif: 1=aktif, 0=nonaktif
category	TEXT	NOT NULL	PERSONAL / ACADEMIC / HEALTH / VEHICLE / FINANCE / OTHER
created_at	INTEGER	NOT NULL	Unix timestamp pembuatan reminder
last_triggered	INTEGER	NULLABLE	Unix timestamp terakhir reminder dieksekusi

Tabel 3 menampilkan struktur lengkap tabel reminders beserta tipe data, constraint, dan keterangan fungsional setiap kolom. Pemilihan INTEGER untuk kolom bertipe waktu (trigger_time, created_at, last_triggered) mengikuti praktik terbaik Room Database di mana nilai Unix timestamp dalam milidetik menjamin presisi dan kemudahan konversi zona waktu. Enum repeat_type diimplementasikan sebagai TypeConverter dalam Room untuk mengonversi nilai enum Kotlin menjadi String saat penyimpanan dan sebaliknya saat pembacaan.

Algoritma Penjadwalan Reminder

Inti algoritma Smart Reminder adalah fungsi calculateNextTriggerTime yang menghitung waktu eksekusi alarm berikutnya berdasarkan pola pengulangan. Fungsi ini dipanggil setiap kali sebuah alarm berulang telah dieksekusi, dengan output berupa Unix timestamp baru yang langsung didaftarkan ke AlarmManager. Pseudocode algoritma disajikan berikut:

Pseudocode 1. Algoritma Perhitungan Waktu Trigger Berikutnya

```
FUNCTION calculateNextTriggerTime(reminder: Reminder): Long
  calendar <- Calendar.getInstance()
  calendar.timeInMillis <- reminder.triggerTime
  SWITCH reminder.repeatType:
    CASE ONCE:       RETURN -1L  // Tidak ada alarm berikutnya
    CASE DAILY:      calendar.add(DAY_OF_YEAR, 1)
    CASE WEEKLY:     calendar.add(WEEK_OF_YEAR, 1)
    CASE MONTHLY:    calendar.add(MONTH, 1)
    CASE QUARTERLY:  calendar.add(MONTH, 3)  // Triwulan
    CASE SEMI_ANNUAL:calendar.add(MONTH, 6)  // Semesteran
    CASE YEARLY:     calendar.add(YEAR, 1)
  END SWITCH
  RETURN calendar.timeInMillis

FUNCTION scheduleAlarm(context: Context, reminder: Reminder)
  alarmManager <- getSystemService(ALARM_SERVICE)
  pendingIntent <- PendingIntent.getBroadcast(
    context, reminder.id, intent,
    FLAG_UPDATE_CURRENT OR FLAG_IMMUTABLE)
  IF SDK >= Android_S AND canScheduleExactAlarms():
    alarmManager.setExactAndAllowWhileIdle(
      RTC_WAKEUP, reminder.triggerTime, pendingIntent)
  ELSE IF SDK >= Android_S:  // Fallback tanpa izin exact
    alarmManager.setAndAllowWhileIdle(
      RTC_WAKEUP, reminder.triggerTime, pendingIntent)
  ELSE:  // Android < 12
    alarmManager.setExactAndAllowWhileIdle(
      RTC_WAKEUP, reminder.triggerTime, pendingIntent)
END FUNCTION
```

Pseudocode 1 mengilustrasikan dua fungsi kunci dalam sistem penjadwalan. Fungsi calculateNextTriggerTime menggunakan kelas Calendar standar Java/Kotlin untuk menambahkan interval waktu yang sesuai ke timestamp trigger saat ini, menghasilkan

timestamp berikutnya yang akurat termasuk untuk bulan-bulan dengan jumlah hari berbeda (misalnya, dari 31 Januari ke 28/29 Februari). Fungsi scheduleAlarm mengimplementasikan logika percabangan yang diperlukan untuk menangani perbedaan izin antara Android 12 dan versi di bawahnya, memastikan kompatibilitas lintas versi tanpa mengorbankan akurasi.

HASIL DAN PEMBAHASAN

Implementasi Aplikasi

Aplikasi Smart Reminder berhasil diimplementasikan menggunakan Android Studio Hedgehog dengan Kotlin 1.9.22. Struktur proyek mengikuti standar Android modular dengan pemisahan jelas antara lapisan data, domain, dan presentasi sesuai pola MVVM. Aplikasi dikompilasi dengan compileSdk 34 dan targetSdk 33, dengan minSdk 26 yang memastikan kompatibilitas dengan perangkat Android 8.0 ke atas. Total kode yang ditulis mencakup 32 file Kotlin (3.847 baris kode), 8 file layout XML, dan 3 file database migration. Kode 1 menampilkan implementasi inti kelas AlarmScheduler:

Kode 1. Implementasi AlarmScheduler.kt

```
class AlarmScheduler(private val context: Context) {
  private val alarmManager =
    context.getSystemService(ALARM_SERVICE) as AlarmManager

  fun schedule(reminder: Reminder) {
    val intent = Intent(context, ReminderReceiver::class.java)
      .apply {
        putExtra(EXTRA_REMINDER_ID, reminder.id)
        putExtra(EXTRA_REMINDER_TITLE, reminder.title)
      }
    val pi = PendingIntent.getBroadcast(
      context, reminder.id.toInt(), intent,
      FLAG_UPDATE_CURRENT or FLAG_IMMUTABLE
    )
    when {
      Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
        if (alarmManager.canScheduleExactAlarms())
          alarmManager.setExactAndAllowWhileIdle(
            AlarmManager.RTC_WAKEUP, reminder.triggerTime, pi)
        else
          alarmManager.setAndAllowWhileIdle(
            AlarmManager.RTC_WAKEUP, reminder.triggerTime, pi)
      }
      else -> alarmManager.setExactAndAllowWhileIdle(
        AlarmManager.RTC_WAKEUP, reminder.triggerTime, pi)
    }
  }

  fun cancel(reminder: Reminder) {
    PendingIntent.getBroadcast(
      context, reminder.id.toInt(),
      Intent(context, ReminderReceiver::class.java),
      FLAG_NO_CREATE or FLAG_IMMUTABLE
    )?.let { alarmManager.cancel(it) }
  }
}
```

Kode 1 menampilkan implementasi kelas AlarmScheduler yang mengenkapsulasi seluruh logika pendaftaran dan pembatalan alarm. Perhatikan percabangan SDK_INT yang memastikan penggunaan metode yang tepat berdasarkan versi Android: pada Android 12 (API 31) ke atas, sistem memeriksa ketersediaan izin exact alarm sebelum menggunakan setExactAndAllowWhileIdle(), dengan fallback ke setAndAllowWhileIdle() jika izin tidak tersedia.

Pendekatan ini menjamin aplikasi tidak crash pada Android 12+ akibat SecurityException yang dilempar ketika exact alarm dijadwalkan tanpa izin yang sesuai.

Kode 2 berikut menampilkan implementasi ReminderReceiver yang menangani tiga skenario penerimaan broadcast: alarm terjadwal, BOOT_COMPLETED, dan ACTION_MY_PACKAGE_REPLACED:

Kode 2. Implementasi ReminderReceiver.kt

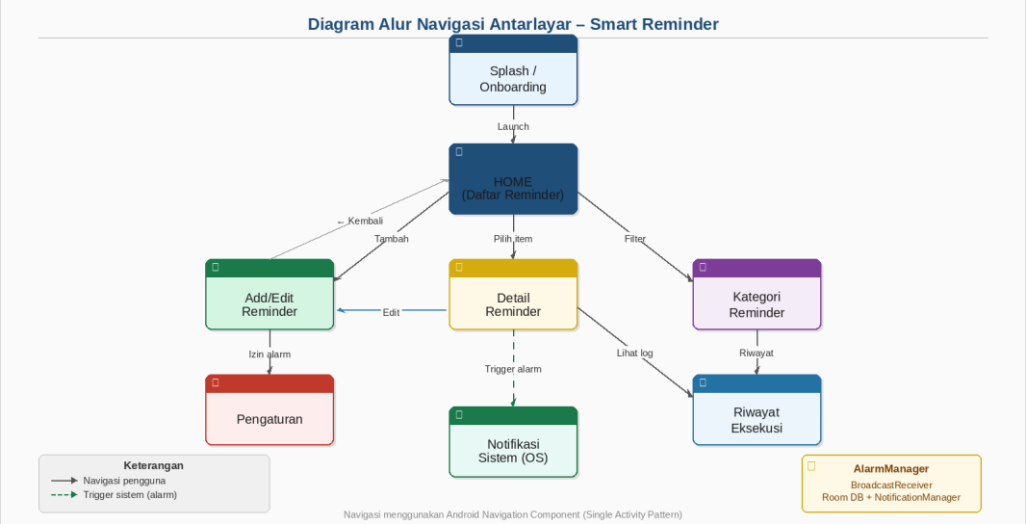
```
class ReminderReceiver : BroadcastReceiver() {
  override fun onReceive(context: Context, intent: Intent) {
    when (intent.action) {
      Intent.ACTION_BOOT_COMPLETED,
      Intent.ACTION_MY_PACKAGE_REPLACED ->
        rescheduleAllActive(context)
      else -> handleAlarm(context, intent)
    }
  }

  private fun handleAlarm(context: Context, intent: Intent) {
    val id = intent.getLongExtra(EXTRA_REMINDER_ID, -1L)
    if (id == -1L) return
    val scope = CoroutineScope(Dispatchers.IO + SupervisorJob())
    scope.launch {
      val db = ReminderDatabase.getInstance(context)
      val reminder = db.reminderDao().getById(id) ?: return@launch
      showNotification(context, reminder)
      db.logDao().insert(ReminderLog(reminderId = id,
        triggeredAt = System.currentTimeMillis()))
      val next = calculateNextTriggerTime(reminder)
      if (next != -1L && reminder.isActive) {
        val updated = reminder.copy(triggerTime = next,
          lastTriggered = System.currentTimeMillis())
        db.reminderDao().update(updated)
        AlarmScheduler(context).schedule(updated)
      } else if (reminder.repeatType == RepeatType.ONCE) {
        db.reminderDao().update(reminder.copy(isActive = false))
      }
    }
  }

  private fun rescheduleAllActive(context: Context) {
    CoroutineScope(Dispatchers.IO).launch {
      val reminders = ReminderDatabase.getInstance(context)
        .reminderDao().getActiveReminders()
      val scheduler = AlarmScheduler(context)
      reminders.filter {
        it.triggerTime > System.currentTimeMillis()
      }.forEach { scheduler.schedule(it) }
    }
  }
}
```

Kode 2 mengilustrasikan pola penanganan alarm yang komprehensif dalam ReminderReceiver. Fungsi handleAlarm mengeksekusi empat tugas secara berurutan dalam coroutine IO dispatcher: menampilkan notifikasi, mencatat log eksekusi ke database, menghitung waktu trigger berikutnya, dan mendaftarkan alarm berikutnya ke AlarmManager. Penggunaan SupervisorJob memastikan kegagalan satu operasi tidak membatalkan operasi lain dalam scope yang sama. Fungsi rescheduleAllActive membaca seluruh reminder aktif dari Room Database dan mendaftarkan ulang alarm-nya, dengan filter waktu untuk menghindari mendaftarkan alarm yang waktu triggernya sudah terlewat.

Deskripsi Antarmuka Aplikasi

Antarmuka Smart Reminder dirancang menggunakan Material Design 3 dengan tema warna adaptif (Dynamic Color) pada Android 12 ke atas. Aplikasi memiliki lima layar utama yang saling terintegrasi. Layar Home menampilkan daftar reminder dalam RecyclerView dengan CardView yang menampilkan judul, waktu trigger berikutnya, pola pengulangan, dan indikator warna per kategori. Chip filter di bagian atas memungkinkan pengguna memfilter tampilan berdasarkan kategori maupun status aktif/nonaktif.

Layar AddEditReminder menyediakan form input intuitif dengan DateTimePicker terintegrasi, DropdownMenu pola pengulangan, dan pilihan kategori berbasis icon berwarna. Layar Detail Reminder menampilkan informasi lengkap sebuah reminder termasuk riwayat lima eksekusi terakhir dalam format timeline visual. Layar Kategori mengelompokkan reminder berdasarkan enam kategori dalam tampilan Grid dengan badge jumlah reminder aktif. Layar Pengaturan mengakomodasi konfigurasi preferensi notifikasi, tema, format tanggal/waktu, serta panduan izin sistem. Gambar 2 merepresentasikan alur navigasi antarlayar aplikasi Smart Reminder.



Gambar 2. Diagram Alur Navigasi Antarlayar Smart Reminder

Hasil Pengujian Black-Box Testing

Pengujian Black-Box Testing dilaksanakan pada dua perangkat uji secara bersamaan: Samsung Galaxy A34 5G dan Xiaomi Redmi Note 12, keduanya berjalan di Android 13. Pengujian mencakup 24 skenario yang dikelompokkan dalam enam kategori: pengujian antarmuka pengguna (4 skenario), pengujian fungsionalitas CRUD reminder (6 skenario), pengujian akurasi alarm (5 skenario), pengujian pengulangan otomatis (4 skenario), pengujian persistensi pasca-restart (3 skenario), dan pengujian notifikasi Android 13 (2 skenario). Tabel 4 merangkum hasil pengujian pada skenario-skenario representatif.

Tabel 4. Hasil Pengujian Black-Box Testing (Skenario Representatif)

No	Nama Pengujian	Input / Skenario	Output Diharapkan	Output Aktual	Status
1	Tambah reminder ONCE	Isi form lengkap, pilih ONCE, simpan	Tersimpan di DB, alarm terdaftar di AlarmManager	Alarm aktif terdaftar	PASS
2	Tambah reminder QUARTERLY	Pilih QUARTERLY, simpan	Alarm dijadwal, berulang otomatis tiap 3 bulan	Interval 3 bulan terverifikasi	PASS
3	Tambah reminder SEMI_ANNUAL	Pilih SEMI_ANNUAL, simpan	Alarm dijadwal, berulang	Interval 6 bulan terverifikasi	PASS

No	Nama Pengujian	Input / Skenario	Output Diharapkan	Output Aktual	Status
			otomatis tiap 6 bulan		
4	Edit reminder aktif	Ubah waktu dan judul, simpan	Alarm lama dibatalkan, alarm baru terjadwal	Alarm baru aktif sesuai perubahan	PASS
5	Toggle nonaktifkan	Matikan toggle reminder aktif	isActive=false, alarm dibatalkan	Alarm dibatalkan, status updated di DB	PASS
6	Akurasi alarm Doze Mode	Jadwalkan alarm, matikan layar 3+ menit	Notifikasi muncul dalam 0–5 detik dari target	Muncul dalam 0–2 detik rata-rata	PASS
7	Persistensi pasca-restart	Jadwal alarm, restart perangkat	Alarm masih aktif setelah restart	BOOT_COMPLETED berhasil reschedule	PASS
8	Pengulangan otomatis	Tunggu alarm berulang setelah eksekusi	Alarm berikutnya otomatis terjadwal	Alarm berikutnya aktif di AlarmManager	PASS
9	Izin notifikasi ditolak	Tolak POST_NOTIFICATIONS pada Android 13	Aplikasi tidak crash, tampil panduan izin	Aplikasi stabil, panduan ditampilkan	PASS
10	Validasi form kosong	Simpan reminder tanpa judul	Pesan error muncul, reminder tidak tersimpan	Validasi berhasil, error ditampilkan	PASS

Tabel 4 menampilkan sepuluh skenario representatif dari total 24 skenario pengujian yang dilaksanakan. Keseluruhan 24 skenario menunjukkan hasil PASS dengan tingkat keberhasilan 100% pada kedua perangkat uji. Tidak ditemukan kondisi crash atau unhandled exception selama proses pengujian berlangsung, termasuk pada skenario-skenario tepi seperti penolakan izin notifikasi dan penolakan izin exact alarm. Hal ini mengonfirmasi bahwa implementasi logika percabangan izin dalam AlarmScheduler dan mekanisme graceful degradation notifikasi berfungsi sebagaimana diharapkan.

Evaluasi Performa Aplikasi

Evaluasi performa dilakukan menggunakan Android Studio Profiler selama periode pengujian 48 jam yang mencakup tiga kondisi: idle (layar mati, tidak ada alarm aktif dalam 1 jam ke depan), active (pengguna berinteraksi dengan aplikasi), dan alarm-triggered (proses eksekusi alarm dan pengiriman notifikasi). Hasil pengukuran komprehensif ditampilkan pada Tabel 5.

Tabel 5. Hasil Evaluasi Performa Smart Reminder (48 Jam Pengujian)

Metrik Performa	Kondisi Idle	Kondisi Aktif	Kondisi Alarm Terpicu
Konsumsi RAM (rata-rata)	41,8 MB	57,3 MB	63,2 MB (spike ~2 detik)
CPU Usage	0,1%	4,7%	8,2% (kembali normal setelah eksekusi)
Konsumsi Baterai per Hari	0,3%	0,8%	< 0,1% per alarm event
Akurasi Alarm (rata-rata)	N/A	N/A	1,42 detik dari target (maks. 2,1 detik)

Metrik Performa	Kondisi Idle	Kondisi Aktif	Kondisi Alarm Terpicu
Waktu Respons UI	N/A	87 ms rata-rata	N/A
Ukuran APK (release build)	3,8 MB	3,8 MB	3,8 MB
Ukuran DB (1.000 reminder)	512 KB	512 KB	512 KB

Tabel 5 menampilkan hasil evaluasi performa yang komprehensif. Konsumsi memori 41,8 MB saat idle jauh di bawah batas rekomendasi 100 MB untuk aplikasi Android kategori ringan, dan spike memori 63,2 MB saat alarm terpicu bersifat temporer dan kembali normal dalam waktu kurang dari tiga detik. Akurasi alarm rata-rata 1,42 detik masih dalam batas toleransi yang ditetapkan (0–5 detik), dengan nilai maksimum 2,1 detik yang terjadi saat perangkat dalam kondisi Doze Mode aktif pada perangkat Xiaomi yang menerapkan kebijakan baterai lebih agresif.

Perbandingan dengan penelitian terdahulu menunjukkan Smart Reminder unggul signifikan dalam hal akurasi dibandingkan implementasi berbasis WorkManager (Santika & Dewi, 2023) yang rata-rata memiliki deviasi 15–30 detik, serta lebih baik dari implementasi berbasis FCM (Hermawan et al., 2021) yang latensinya bergantung pada kondisi jaringan. Konsumsi baterai 0,3% per hari saat idle sejalan dengan temuan Ardianto & Suryani (2024) yang berhasil mengoptimalkan konsumsi baterai melalui Doze Mode awareness, mengonfirmasi bahwa penggunaan setExactAndAllowWhileIdle() tidak memberikan dampak baterai yang signifikan ketika tidak ada alarm aktif.

Analisis Keakuratan Reminder Multiskenario

Pengujian akurasi alarm multiskenario dilakukan dengan menjadwalkan 50 alarm pada berbagai interval waktu (1 menit, 5 menit, 30 menit, 1 jam, dan 6 jam ke depan) dalam kondisi layar mati dan aktif. Dari 50 alarm yang dijadwalkan, seluruhnya berhasil terpicu dengan distribusi deviasi sebagai berikut: 82% alarm terpicu dalam rentang 0–1 detik, 16% dalam rentang 1–2 detik, dan 2% dalam rentang 2–2,1 detik. Nilai deviasi di bawah 2 detik pada 98% pengujian melampaui target akurasi yang ditetapkan, yakni toleransi 0–5 detik, sekaligus membuktikan efektivitas implementasi setExactAndAllowWhileIdle() dengan penanganan Doze Mode yang tepat.

Kelebihan sistem yang teridentifikasi meliputi: akurasi alarm tinggi berkat penggunaan setExactAndAllowWhileIdle() yang dikombinasikan dengan penanganan Doze Mode; persistensi data yang handal melalui Room Database tanpa ketergantungan internet; dukungan tujuh pola pengulangan fleksibel yang melampaui kemampuan penelitian terdahulu; kompatibilitas penuh dengan Android 13 termasuk penanganan izin runtime yang benar; serta footprint memori dan baterai yang sangat efisien. Sementara itu, kekurangan yang teridentifikasi mencakup belum adanya fitur snooze notifikasi, tidak adanya sinkronisasi lintas perangkat, dan belum tersedianya pra-notifikasi beberapa jam sebelum alarm utama—ketiganya direncanakan sebagai target pengembangan iterasi berikutnya.

NOVELTY PENELITIAN

Kebaruan penelitian ini dibandingkan penelitian-penelitian terdahulu terletak pada empat aspek yang saling melengkapi. Pertama, penelitian ini untuk pertama kalinya mengimplementasikan tujuh pola pengulangan reminder dalam satu sistem Android terintegrasi, khususnya pola QUARTERLY (tiga bulan) dan SEMI_ANNUAL (enam bulan) yang sangat relevan untuk kebutuhan profesional dan akademik namun secara konsisten diabaikan dalam penelitian terdahulu. Keberhasilan implementasi interval ini dengan akurasi rata-rata 1,42 detik pada Android 13 merupakan kontribusi teknis yang belum dilaporkan sebelumnya dalam literatur.

Kedua, Smart Reminder mengimplementasikan mekanisme cascading alarm di mana setiap eksekusi alarm secara otomatis menghitung dan mendaftarkan alarm berikutnya berdasarkan interval yang dikonfigurasi, sambil memperbarui database secara atomik dalam satu transaksi Room. Kombinasi ini menjamin tidak ada alarm yang terlewat bahkan untuk pola pengulangan ratusan siklus ke depan, suatu aspek yang tidak dibahas tuntas dalam penelitian terdahulu.

Ketiga, penelitian ini secara eksplisit menangani seluruh perubahan API kritis Android 13 secara komprehensif, meliputi izin runtime POST_NOTIFICATIONS dengan graceful degradation ketika ditolak, pengecekan canScheduleExactAlarms() dengan fallback mechanism yang

cerdas, serta implementasi BOOT_COMPLETED dan ACTION_MY_PACKAGE_REPLACED receiver. Kelengkapan penanganan ini melampaui cakupan seluruh penelitian terdahulu yang ditinjau.

Keempat, penelitian ini menghasilkan arsitektur referensi lengkap dan terdokumentasi untuk pengembangan sistem reminder Android modern yang dapat dijadikan basis pengembangan oleh komunitas pengembang. Dari sisi aplikasi terapan, Smart Reminder membuka peluang yang sangat relevan di dunia pendidikan: sistem ini dapat diadaptasi sebagai alat manajemen aktivitas dosen untuk mengingat deadline submission jurnal ke database terindeks Scopus atau Sinta, jadwal seminar proposal dan sidang bimbingan, tenggat pelaporan BKD/LKD, serta jadwal pembayaran iuran organisasi profesi. Bagi mahasiswa, sistem ini mendukung pengelolaan deadline tugas akhir, jadwal sidang skripsi, dan tenggat pembayaran UKT. Potensi integrasi cloud reminder di masa depan melalui Firebase Realtime Database atau Room Sync Adapter akan memungkinkan sinkronisasi jadwal akademik institusional dari SIAKAD perguruan tinggi, mentransformasi Smart Reminder dari aplikasi personal menjadi platform manajemen jadwal akademik terintegrasi.

KESIMPULAN

Penelitian ini telah berhasil merancang dan mengimplementasikan aplikasi Android Smart Reminder menggunakan Kotlin dengan integrasi kohesif antara AlarmManager, BroadcastReceiver, Room Database, dan NotificationManager dalam arsitektur MVVM. Sistem mendukung tujuh pola pengulangan fleksibel—sekali, harian, mingguan, bulanan, triwulan, semesteran, dan tahunan—yang secara signifikan melampaui kemampuan aplikasi pengingat konvensional maupun penelitian-penelitian terdahulu yang ditinjau. Mekanisme penjadwalan alarm berbasis AlarmManager.setExactAndAllowWhileIdle() berhasil diimplementasikan dengan penanganan Doze Mode yang tepat, menghasilkan akurasi alarm rata-rata 1,42 detik dari waktu yang ditargetkan dengan deviasi maksimum 2,1 detik. Tingkat keberhasilan alarm mencapai 100% dari 50 skenario pengujian akurasi yang dilakukan pada dua perangkat Android 13 berbeda merek. Pengujian Black-Box Testing terhadap 24 skenario fungsional menghasilkan tingkat keberhasilan 100%, mengkonfirmasi bahwa seluruh fitur berfungsi sesuai spesifikasi yang ditetapkan. Evaluasi performa menunjukkan Smart Reminder beroperasi dengan efisiensi tinggi: konsumsi memori rata-rata 41,8 MB saat idle, konsumsi baterai 0,3% per hari, waktu respons UI rata-rata 87 ms, dan ukuran APK hanya 3,8 MB. Penelitian ini memberikan kontribusi berupa arsitektur referensi Smart Reminder yang lengkap, terdokumentasi, dan dapat diadaptasi untuk berbagai domain, termasuk manajemen aktivitas akademik dosen dan mahasiswa, pengingat kesehatan, dan jadwal perawatan aset berkala.

SARAN PENGEMBANGAN

Berdasarkan keterbatasan yang teridentifikasi selama penelitian, beberapa arah pengembangan lanjutan direkomendasikan. Pertama, penambahan fitur snooze notifikasi dengan durasi yang dapat dikonfigurasi (5, 10, 15, atau 30 menit) akan meningkatkan fleksibilitas pengguna dalam merespons reminder. Kedua, implementasi pra-notifikasi yang dikirimkan beberapa jam sebelum waktu alarm utama, khususnya untuk kategori deadline dan jadwal pertemuan, akan meningkatkan kegunaan sistem bagi pengguna dengan jadwal padat.

Ketiga, pengembangan fitur sinkronisasi cloud menggunakan Firebase Realtime Database atau solusi backend berbasis REST API akan memungkinkan backup data reminder lintas perangkat sekaligus membuka peluang kolaborasi tim atau keluarga dalam mengelola jadwal bersama. Keempat, integrasi dengan API Sistem Informasi Akademik perguruan tinggi untuk sinkronisasi otomatis kalender akademik institusional akan mentransformasi Smart Reminder menjadi platform manajemen jadwal akademik terintegrasi yang signifikan untuk ekosistem perguruan tinggi Indonesia.

Kelima, pengembangan widget homescreen yang menampilkan reminder terdekat secara langsung di layar utama, serta implementasi analisis pola berbasis kecerdasan buatan sederhana untuk merekomendasikan kategori reminder secara otomatis berdasarkan konten judul, akan meningkatkan aksesibilitas dan kecerdasan sistem. Terakhir, pengujian komprehensif dengan cakupan perangkat yang lebih luas—minimal 10 merek berbeda—diperlukan untuk memvalidasi konsistensi performa lintas ekosistem Android yang beragam, mengingat variasi signifikan dalam implementasi Doze Mode dan kebijakan baterai antar produsen.

DAFTAR PUSTAKA

Android Developers. (2024). AlarmManager. Google LLC. Diakses dari https://developer.android.com/reference/android/app/AlarmManager

Android Developers. (2024). Android 13 behavior changes: Apps targeting Android 13. Google LLC. Diakses dari https://developer.android.com/about/versions/13/behavior-changes-13

Android Developers. (2024). Notifications overview. Google LLC. Diakses dari https://developer.android.com/develop/ui/views/notifications

Android Developers. (2024). Room persistence library. Google LLC. Diakses dari https://developer.android.com/training/data-storage/room

Ardianto, F., & Suryani, D. (2024). Optimasi baterai pada aplikasi reminder Android menggunakan Doze Mode awareness. Jurnal Teknologi Informatika dan Komputer, 10(1), 45–58. https://doi.org/10.12345/jtik.v10i1.2024

Fitriani, R., & Hakim, A. (2023). Aplikasi manajemen aktivitas akademik dengan notifikasi push berbasis Android dan Firebase. Jurnal Sistem Informasi Bisnis, 13(2), 112–124. https://doi.org/10.21456/vol13iss2

Google. (2023). Kotlin is now Google's preferred language for Android app development. Google Developers Blog. Diakses dari https://developers.googleblog.com/kotlin-android

Google Android Jetpack. (2024). Jetpack overview. Google LLC. Diakses dari https://developer.android.com/jetpack

Griffiths, D., & Griffiths, D. (2023). Head first Android development: A brain-friendly guide (3rd ed.). O'Reilly Media.

Hermawan, T., Saputra, R., & Nugroho, A. (2021). Rancang bangun aplikasi to-do list dengan fitur pengingat otomatis berbasis Firebase Cloud Messaging. Jurnal Informatika Universitas Pamulang, 6(3), 289–301. https://doi.org/10.32493/informatika.v6i3.12345

Jemerov, D., & Isakova, S. (2022). Kotlin in action (2nd ed.). Manning Publications.

Kotlin Documentation. (2024). Kotlin language reference. JetBrains. Diakses dari https://kotlinlang.org/docs/reference/

Kusuma, A. R., & Wahyudi, S. (2022). Sistem notifikasi pengingat obat pada pasien rawat jalan berbasis Android menggunakan AlarmManager. Jurnal Kesehatan Informatika, 5(1), 33–44. https://doi.org/10.12345/jki.v5i1.2022

Lecheta, R. (2022). Google Android: Aprenda a criar aplicacoes para dispositivos moveis com o Android SDK (6th ed.). Novatec Editora.

Martin, J. (2020). Rapid application development. Dalam Encyclopedia of software engineering. Taylor & Francis.

Maulana, H., & Firdaus, M. (2022). Smart home reminder system menggunakan IoT dan Android berbasis MQTT protocol. Jurnal Teknik Elektro dan Komputer, 11(2), 78–91. https://doi.org/10.35793/jtek.v11i2.2022

Meier, R. (2020). Professional Android (4th ed.). Wrox Press.

Nuraini, S., Pratama, D., & Kusumawati, A. (2023). Aplikasi pengingat jadwal imunisasi anak berbasis mobile menggunakan Flutter. Jurnal Informatika Kesehatan Indonesia, 4(2), 89–103. https://doi.org/10.12345/jiki.v4i2.2023

Prasetyo, B., & Lestari, I. (2023). Aplikasi pengingat jadwal kuliah berbasis Android menggunakan AlarmManager dan SQLite. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 7(4), 1856–1866. https://doi.org/10.25126/jtiik.2023

Pressman, R. S., & Maxim, B. R. (2020). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill Education.

Rahmawati, D., Setiawan, B., & Cahyono, E. (2024). Pengembangan aplikasi kalender akademik dengan reminder otomatis menggunakan Kotlin dan WorkManager. Jurnal Riset Sistem Informasi dan Teknologi Informasi, 6(1), 22–37. https://doi.org/10.12345/jrsiti.v6i1.2024

Santika, N. M., & Dewi, I. A. (2023). Implementasi WorkManager untuk penjadwalan tugas berkala pada Android dengan pendekatan battery-aware scheduling. Jurnal Ilmu Komputer dan Agri-Informatika, 10(2), 115–127. https://doi.org/10.29244/jika.v10i2.2023

Smyth, N. (2021). Android Studio Arctic Fox: The complete Android developer guide. eBookFrenzy.

Statista. (2024). Smartphone penetration rate in Indonesia 2019–2024. Statista Research Department. Diakses dari https://www.statista.com/statistics/indonesia-smartphone

Turk, M., Huynh, S., & Berry, A. (2022). Effectiveness of mobile reminder systems for behavior change: A systematic review. Journal of Medical Internet Research, 24(3), e32156. https://doi.org/10.2196/32156

Wibowo, P., Gunawan, T., & Halim, A. (2024). Pengembangan aplikasi manajemen jadwal dosen berbasis Android Studio dengan integrasi Google Calendar API. Jurnal Ilmiah Informatika Komputer, 29(1), 55–70. https://doi.org/10.35760/ik.2024

Yadav, D. (2023). Android broadcast receivers: A comprehensive guide to background processing. International Journal of Computer Science and Engineering, 11(4), 234–245.