BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Dalam merancang sistem koper pintar, sejumlah studi terdahulu yang memiliki relevansi dijadikan sebagai referensi. Salah satu di antaranya adalah penelitian yang dilakukan oleh Maulana Setia Aji, Rahmi Hidayati, dan Kartika Sari dengan judul *"Integrasi Teknologi Biometri, RFID dan GPS pada Sistem Keamanan Koper Pintar"*. Penelitian tersebut bertujuan untuk meningkatkan keamanan koper melalui *integrasi* tiga *teknologi*, yaitu *biometrik* sidik jari, *RFID*, dan *GPS tracking*.

Sistem yang dibangun dalam penelitian tersebut menggunakan *sensor fingerprint* untuk *verifikasi biometrik* pengguna, serta *sensor RFID RC522* untuk membaca kartu identitas unik yang digunakan dalam membuka koper. Sebagai bentuk keamanan tambahan, digunakan pula *modul GPS Ublox Neo 6M* yang dapat mengirimkan lokasi koper secara *real-time* melalui *bot* Telegram. Notifikasi juga dikirim secara otomatis melalui *modul GSM SIM800L* apabila terjadi upaya akses dari kartu *RFID* yang tidak terdaftar.

Berdasarkan hasil pengujian, sistem berhasil mengidentifikasi pemilik koper dengan rata-rata delay 1,21 detik pada *sensor* sidik jari dan memberikan respons peringatan dengan *delay* 7,86 detik terhadap akses *RFID* yang tidak dikenal. *Akurasi* pelacakan lokasi koper menggunakan *GPS* juga tergolong tinggi, dengan rata-rata error hanya sekitar 1,9 meter dari lokasi sebenarnya.

Penelitian ini menunjukkan bahwa penggunaan gabungan *sensor RFID* dan *biometrik* dapat secara *signifikan* meningkatkan keamanan koper, terutama saat bepergian. Selain itu, *integrasi sistem* dengan Telegram *bot* dan *modul GSM* memungkinkan pemilik koper memantau kondisi dan lokasi koper secara jarak jauh secara *efisien*. Penelitian ini menjadi dasar penting dalam pengembangan koper pintar *modern* yang mendukung *sistem kontrol* berbasis *IoT* dan komunikasi *real-time*.

Dengan mengacu pada penelitian ini, pengembangan sistem koper pintar dalam proyek ini akan difokuskan pada penyempurnaan fitur pelacakan, penyederhanaan *integrasi perangkat keras*, dan kemungkinan pengembangan *antarmuka* aplikasi berbasis *mobile* untuk pemantauan koper secara langsung melalui smartphone[7].

Pada jurnal *internasional* dengan judul *"BLUETOOTH-BASED REAL-TIME LUGGAGE TRACKING AND STATUS UPDATES"* adalah oleh Anantha Raman Rathinam dan Srinivasan Chelliah, yang merancang sistem pelacakan koper secara *real-time* berbasis *Bluetooth* dan *GPS*. Sistem ini terdiri dari tas koper yang dilengkapi *GPS tracker* dan *modul Bluetooth*, sebuah *mikrokontroler* sebagai pusat pemrosesan data, aplikasi *mobile* sebagai *antarmuka* pengguna, serta sistem *backend* berbasis *cloud* untuk penyimpanan dan *analisis data*. Informasi lokasi koper dikirim secara *real-time* melalui *Bluetooth* ke aplikasi pengguna, sementara data status koper *dianalisis* dan disimpan di *cloud*. Sistem ini juga mengirimkan notifikasi otomatis jika terjadi perubahan status koper, seperti perubahan lokasi atau

kehilangan koneksi. Berdasarkan pengujian, sistem ini mampu memberikan *akurasi* pelacakan sebesar 95%, notifikasi rata-rata dikirim dalam waktu kurang dari 5 detik, dan berhasil mengurangi insiden koper hilang hingga 80%. Dengan fitur-fitur tersebut, sistem ini terbukti meningkatkan keamanan, kenyamanan, dan *efisiensi* dalam manajemen koper selama perjalanan[8].

2.2 Landasan Teori

2.2.1 *Android*

Android adalah sistem operasi untuk perangkat mobile yang berbasis Linux, mencakup sistem operasi inti middleware, serta berbagai aplikasi. Beberapa definisi lain mengenai Android antara lain:

1) Merupakan platform terbuka *(Open Source)* yang memungkinkan para pengembang *(Programer)* untuk menciptakan berbagai aplikasi.
2) Merupakan *sistem operasi* yang diakusisi oleh *Google Inc*. dari *Android Inc.*
3) Bukan merupakan *bahasa pemrograman*, melainkan menyediakan lingkungan hidup atau *run time enviroment* yang disebut *DVM (Dalvik Virtual Machine)* yang telah dioptimalkan untuk perangkat dengan kapasitas *memori* terbatas[9].

2.2.2 *Android Studio*

Android Studio adalah sebuah *Integrated Development Environment (IDE)* yang dirancang khusus untuk pengembangan aplikasi berbasis Android dan dibangun di atas platform *Intellij IDEA*. Selain berfungsi sebagai editor kode yang kuat dan efisien, *Android Studio* juga dilengkapi dengan berbagai fitur canggih yang dirancang untuk menunjang produktivitas pengembang selama proses pembuatan aplikasi[10].

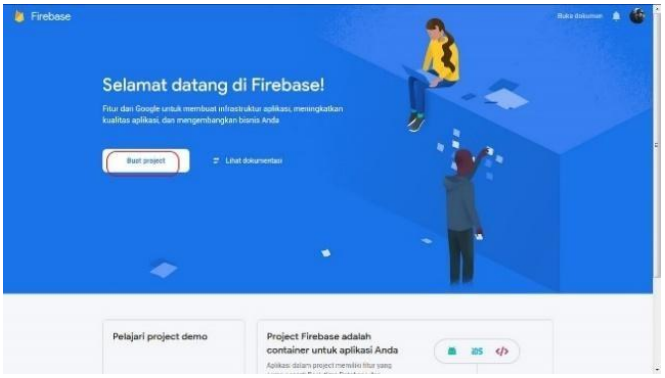
Gambar 2. 1 Logo Android Studio[10]

2.2.3 *Android SDK*

Android SDK merupakan alat atau *tools* yang digunakan untuk membuat aplikasi *platform Android* menggunakan bahasa *pemrograman Java. Android Software DevelopmentKit (SDK)* merupakan *kit* yang bisa digunakan oleh para *developer* untuk mengembangkan *aplikasi berbasis Android.* Di dalamnya, terdapat beberapa *tools* seperti *debugger, softwarelibraries, emulator, dokumentasi, sample code dan tutorial*[11].

2.2.4 *Firebase*

Firebase adalah *teknologi* yang *relatif* baru untuk menangani sejumlah besar data yang tidak *terstruktur* dalam mengembangkan aplikasi *Firebase* saat ini menjadi sebuah *alternatif database* yang *handal* yang mampu memberikan informasi secara cepat[12].



Gambar 2. 2 Tampilan Firebase[12]

2.2.5 *Bahasa Pemrograman Kotlin*

Bahasa *Kotlin* adalah bahasa pemrograman yang dikembangkan oleh JetBrains, perusahaan yang juga mengembangkan *IDE Android Studio*. Bahasa *Kotlin* adalah pengembangan dari bahasa *Java* yang sudah popular sebelumnya. Bahasa *Kotlin* memiliki fitur-fitur bahasa modern yang lebih dibandingkan bahasa[13].

2.2.6 *Google Maps API*

Google Maps API adalah sebuah jasa *peta globe* virtual gratis dan online yang disediakan oleh Google dan dapat ditemukan oleh di http://maps.google.com/, Google map API merupakan aplikasi interface yang dapat diakses menggunakan JavaScript agar Google

Map dapat ditampilkan pada halaman aplikasi yang sedang kita bangun[14].

.

Gambar 2. 3 Logo *Google Maps*[14]

2.2.7 *Unified Modeling Language* (UML)

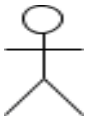
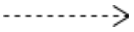
Unified Modeling Language (UML) Unified Modeling Language (UML) adalah sebuah bahasa pemodelan *visual* yang *digunakan untuk mendokumentasikan,merancang, dan mengkomunikasikan desain* suatu *sistem* secara *terstruktur dan sistematis. UML* merupakan standar yang dikembangkan oleh *Object Management Group (OMG*) dan digunakan secara luas dalam industri *perangkat lunak*[15]. Pengembangan perangkat lunak berbasis Objek. UML tersusun atas sejumlah *elemen grafis* membentuk 9 diagram-diagram. Dalam penelitian ini melakukan desain hanya 4 diagram yaitu *Use Case Diagram, Activity Diagram, Sequence Diagram dan Class Diagram.* Berikut pengertian dan bentuk simbol-simbol dari diagram.

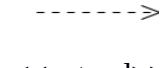
1. *Usecase* diagram

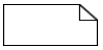
Menggambarkan *fungsionalitas* yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case

merepresentasikan sebuah *interaksi* antara aktor dengan sistem. Misalnya login ke sistem, membuat sebuah daftar belanja, dan sebagainya. Seorang aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Berikut simbol-simbol *use case*:

Tabel 2. 1 Simbol *Use case*[15]

No.	**Gambar**	**Nama**	**Keterangan**
1		*Actor*	Orang,proses atau sistem yang berinteraksi dengan sistem lain.
2		*Dependency*	Hubungan perubahan yang terjadi pada suatu elemen mandiri
3		*Generalization*	Hubungan generalisasi (umum-khusus) antar dua buah use case
4	<<include>> 	*Include*	Relasi use case tambahan ke sebuah use case, dimana use case yang ditambahkan dapat berdiri sendiri.

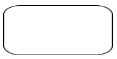
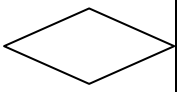
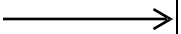
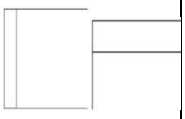
5	 <<extend>>	*Extend*	Relasi tambahan yang Menunjukan use case lain memerlukan use case. syarat untuk dijalankan atau agar fungsinya dapat berjalan dengan baik.
6		*Association*	Komunikasi antar aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan

			elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

2. *Activity Diagram*

Activity Diagram adalah *Activity Diagram* menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem[16]. Activity diagram digunakan untuk menampilkan tindakan dan sebagian dasar transisi yang dipicu oleh penyelesaian tindakan yang berasal dari sumber. Activity diagram sama seperti halnya flowchart yang menggambarkan proses yang terjadi antara actor dan sistem.

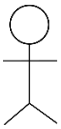
Tabel 2. 2 Simbol Activity Diagram[16]

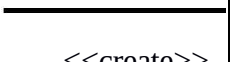
No.	Gambar	Nama	Keterangan
1		*Activity*	Memperlihatkan bagaimana kelas saling berinteraksi
2		*Decision*	Asosiasi percabanagan jika ada pilihan yang lebih dari satu
3		*Actifity Final Node*	Bagaimana objek diakhiri. Dalam satu diagram minimal ada satu fina state
4		*Fork Node*	Satu aliran pada tahap yang berubah menjadi beberapa aliran
5		*Initial Node*	Bagaimana objek diawali. Hanya ada satu initial dalam satu diagram.
6		*Control Flow*	Menunjukan urutan aliran aktivitas.
7		*Partition*	Simbol yang membatasi aktivotas antar orang, organisasi, sistem atau kelompok

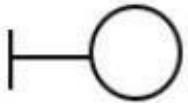
3. *Sequence Diagram*

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar obyek dan mengindikasikan komunikasi di antara obyek-obyek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu[17]. Diagram ini menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* biasa digunakan untuk menggambarkan scenario atau langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu.

Tabel 2. 3 Simbol sequence diagram[17]

1		*Actor*	Orang , proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat
2		*Lifeline*	Menyatakan kehiduan suatu objek
3		*Object*	Menyatakan objek yang berinteraksi pesan

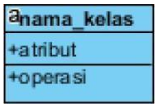
4		*Timelife*	Menyatakan objek dalam berinteraksi pesan
5	 <<create>>	Pesan tipe *Create*	Menyatakan objek dalam keadaan aktif dan berinteraksi
6	1: nama_metode()	Pesan tipe *Call*	Menyatakan suatu objek memanggil operasi/ metode yang ada pada objek lain atau dirinya sendiri
7	1: masukan	Pesan tipe *Send*	Menyatakan bahwa suatu objek mengirimkan data/masukan/infor masi ke objek lain
8	1: keluaran 	Pesan tipe *Return*	Menyatakan bahwa suatu objek telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu

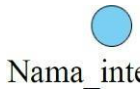
9	<<destroy>>	Pesan tipe *Destroy*	Menyatakan suatu objek telah menjalankan suatu operasi
10		Boundary Class	Berupan tepi dari sistem, seperti user interface atau alat yang berinteraksi dengan sistem lain
11		Control Class	Mengatur aliran dari informasi untuk sebuah skenario

4. Class Diagram

Class adalah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. Class memiliki tiga areapokok, yaitu Nama (dan stereotype), Attribute dan Metode[18].

Tabel 2. 4 Simbol class diagram[18]

No.	Gambar	Nama	Keterangan
1		Kelas	Kelas pada struktur sistem

2		Antarmuka/ *Interface*	Sama dengan konsepn interface dalam pemrograman berorientasi objek
3		Asosiasi/ *Association*	Relasi antar kelas dengan makna umum
4		Asosiasi berarah/ *Directed Association*	Relasi antar kelas dengan makna kelas
5		Generalisasi	Relasi antar kelas dengan makna generalisasi
6		Kebergantungan /*Dependency*	Relasi antar kelas dengan maksa kebergantungan antar kelas
		Agregasi /*Aggregation*	Relasi antar kelas dengan makna whole-part