

BAB II

LANDASAN TEORI

2.1. Aplikasi

Menurut (Listianto et al., 2017) Aplikasi adalah suatu bagian dari perangkat lunak yang dapat digunakan untuk menyelesaikan masalah-masalah yang khusus yang dihadapi user dengan menggunakan kemampuan komputer.

Menurut (Susanti & Haevi, 2018) Aplikasi merupakan suatu perangkat lunak yang mempermudah pemakai dalam menyelesaikan tugasnya sehingga adanya proses *input* menjadi *output*.

2.2. Aplikasi Mobile

Menurut (Pribadi, 2017) Perangkat *Mobile* telah berubah menjadi salah satu perangkat multifungsi. Salah satunya perangkat multifungsi yang sering digunakan sekarang adalah aplikasi *mobile* sebagai media untuk mengakses informasi dengan mudah. Perkembangan aplikasi *mobile* didukung dengan semakin berkembangnya bahasa pemrograman.

Menurut (Razi et al., 2018) Secara umum aplikasi *mobile* (*mobile application*) adalah sebuah program aplikasi yang berjalan pada perangkat *mobile* misalnya *smartphone* maupun tablet PC yang dirancang untuk menunjang aktivitas pengguna sehingga dapat mempermudah dan meningkatkan fleksibilitas pengguna.

2.3. Antrian

Menurut (D. Irawan, 2017) Antrian tidak hanya terjadi pada suatu sistem transportasi, namun bisa pada banyak hal dalam kehidupan. Secara umum antrian timbul karena proses arus pergerakan orang/barang terpaksa terganggu akibat kegiatan pelayanan.

Sedangkan menurut (Yuliana & Santony, 2019) Antrian adalah kondisi dimana sekumpulan orang, komponen atau mesin yang membutuhkan layanan harus menunggu dalam suatu urutan tertentu sebelum akhirnya memperoleh layanan. Hal ini terjadi pada saat kemampuan penyelenggara layanan lebih kecil dibandingkan dengan kebutuhan layanan. Antrian terjadi ketika pelanggan yang datang ke suatu pelayanan melebihi kapasitas pelayanan yang tersedia.

2.4. Antrian *Online*

Menurut (Junirianto & Fadhliana, 2019) Antrian *online* merupakan suatu sistem antrian yang mengadopsi teknologi *online*. Sistem antrian konvensional yang sebelumnya memanfaatkan nomor ataupun kertas secara langsung di tempat antrian berubah menjadi sistem antrian dengan menggunakan aplikasi yang bisa mengambil nomor antrian dari mana saja dan tidak harus berada di lokasi antrian terlebih dahulu.

2.5. Android

Menurut (Sari & Ali, 2019) Android adalah sebuah sistem operasi untuk *smartphone* dan tablet. Sistem operasi dapat diilustrasikan sebagai jembatan antara piranti (*device*) dan penggunaannya, sehingga pengguna bisa berinteraksi dengan *device*-nya dan menjalankan aplikasi-aplikasi yang tersedia pada *device*. Dari uraian diatas dapat disimpulkan bahwa Android merupakan sistem operasi yang di kembangkan untuk perangkat *mobile* seperti telepon pintar dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang untuk membangun aplikasi mereka sendiri untuk digunakan oleh berbagai macam piranti bergerak.

Sedangkan menurut (Juhara, 2016) Android adalah sistem operasi berbasis Linux yang dimodifikasi untuk perangkat bergerak (*mobile devices*) yang terdiri dari sistem operasi , *middleware*, dan aplikasi-aplikasi utama.

2.6. *Waiting Line*

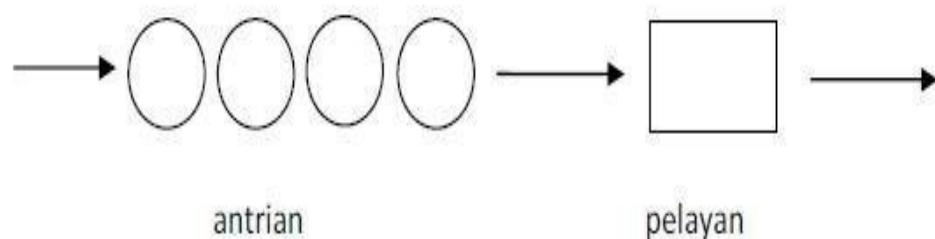
Menurut (Septiani, 2017) *Waiting line method* adalah salah satu metode yang digunakan dalam memecahkan masalah yang ditemukan dalam pengambilan keputusan seperti pemilihan staf, program antrian dari sistem komputer, pembuatan jadwal staf, dan lain-lain. *Waiting line method* sangat berguna untuk menganalisa bentuk panjang antrian, rata-rata waktu pelayanan, rata-rata waktu menunggu. Dengan bantuan perhitungan metode *waiting line* akan menghasilkan informasi tentang tingkat intensitas pelayanan dalam antrian yang dapat dijadikan pertimbangan untuk pengembangan dan perencanaan mutu serta pelayanan suatu perusahaan jasa.

Konsep Model Antrian (Ramanda, 2016), sebagai berikut:

1. Garis Tunggu atau Antrian Ada orang atau barang yang menunggu untuk mendapatkan jasa pelayanan.
2. Fasilitas Pelayanan atau *Server* Ada komputer atau staf yang melayani tetapi biasanya relatif mahal sehingga tersedia dalam jumlah terbatas, karena berusaha menekan *cost*.

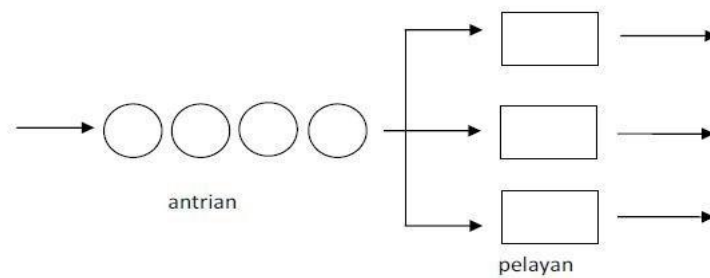
Terdapat 2 (dua) buah model dalam antrian dalam Metode *Waiting Line* (Wati, 2017), diantaranya:

1. Antrian Tunggal atau *Single Channel Model* (M/M/1).



Gambar 2. 1 *Single Channel Model* (M/M/1)

2. Antrian Banyak atau *Multiple Channel Model* (M/M/s).



Gambar 2. 2 Multi Channel Model (M/M/s).

Tabel 2. 1 Tabel Atribut Antrian

<i>Input</i>	Penjelasan
<i>Arrival Rate (lambda, λ)</i>	Tingkat kedatangan orang atau barang ke fasilitas pelayanan, dinyatakan dalam unit (barang atau orang) tiap satuan waktu (detik, menit, jam, hari, dst).
<i>Service Rate (mu, μ)</i>	Tingkat pelayanan atau jumlah orang/barang yang dapat dilayani selama periode waktu tertentu, dinyatakan dalam unit per satuan waktu.
Jumlah Fasilitas Pelayanan	Jumlah fasilitas pelayanan yang tersedia, yaitu 1 atau lebih.
<i>Server cost \$/time</i>	Biaya untuk mengoperasikan satu fasilitas pelayanan setiap satuan waktu, misalnya gaji teller per jam, dinyatakan dalam rupiah per satuan waktu.
<i>Waiting cost \$/time</i>	Kerugian karena pelanggan meninggalkan antrian atau kerugian karena hilangnya produktivitas selama pelanggan menunggu, dinyatakan dalam rupiah per satuan waktu.

Perhitungan Metode *Waiting Line* (Wati, 2017), peneliti jabarkan sebagai berikut:

1. *Single Channel Model (M/M/1)*.

$$P = \lambda/\mu$$

$$Lq = \lambda/(\mu - \lambda)$$

$$L = \frac{\lambda^2}{\mu(\mu-\lambda)}$$

$$Wq = \frac{1}{\mu - \lambda}$$

$$W = \frac{\lambda}{\mu(\mu-\lambda)}$$

Keterangan :

P = Tingkat intensitas fasilitas pelayanan.

Lq = Jumlah kedatangan yang diharapkan menunggu dalam *Waiting Line*.

L = Jumlah rata-rata kedatangan yang diharapkan dalam sistem.

Wq = Waktu yang diharapkan oleh setiap kedatangan untuk menunggu dalam *Waiting Line*.

W = Waktu yang diharapkan oleh setiap kedatangan selama dalam sistem atau menunggu dalam pelayanan.

2. *Multiple Channel Model (M/M/S)*.

$$P = \frac{\lambda}{s\mu}$$

$$P_0 = 1 / \sum_{n=0}^{s-1} \left(1 \frac{(\frac{\lambda}{\mu})^n}{n!} + \frac{1(\frac{\lambda}{\mu})^s}{s!(1-\frac{\lambda}{s\mu})} \right)$$

$$Lq = \frac{\lambda\mu(\frac{\lambda}{\mu})^2}{(s-1)!(s\mu-\lambda)^2} P_0$$

$$Wq = Lq + \frac{\lambda}{\mu}$$

$$Wq = \frac{P_0}{\mu s (s!) \left[1 - \left(\frac{\lambda}{s\mu} \right)^2 \right]} \left(\frac{\lambda}{\mu} \right)$$

$$W = Wq + \frac{\lambda}{\mu}$$

Keterangan :

P = Tingkat intensitas fasilitas pelayanan.

S = Jumlah fasilitas layanan.

λ = jumlah rata-rata tingkat kedatangan persatuan waktu.

μ = jumlah rata-rata yang dilayani persatuan waktu.

P_0 = Probabilitas tidak ada kedatangan dalam sistem.

L_q = Jumlah kedatangan yang diharapkan menunggu dalam antrian untuk dilayani.

L = Jumlah rata-rata kedatangan yang diharapkan dalam sistem.

W_q = Waktu yang diharapkan oleh setiap kedatangan untuk menunggu dalam sistem.

W = Waktu yang diharapkan oleh setiap kedatangan selama dalam sistem atau menunggu dalam pelayanan.

2.7. Bahasa pemograman yang dipakai

2.7.1. Android Studio

Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat *open source* atau gratis. Peluncuran Android Studio ini diumumkan oleh Google pada 16 mei 2013 pada *event Google I/O Conference* untuk tahun 2013. Sejak saat itu, Android Studio menggantikan *Eclipse* sebagai IDE (*Integrated Development Environment*) resmi untuk mengembangkan aplikasi Android, Android studio sendiri dikembangkan berdasarkan *IntelliJ IDEA* yang mirip dengan *Eclipse* disertai dengan *ADT plugin (Android Development Tools)*. Android studio memiliki fitur:

- a. Projek berbasis pada *Gradle Build*.
- b. *Refactory* dan pembenahan *bug* yang cepat.
- c. *Tools* baru yang bernama “*Lint*” dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.
- d. Mendukung *Proguard And App-signing* untuk keamanan.

- e. Memiliki GUI (*Graphic User Interface*) aplikasi android lebih mudah.
- f. Didukung oleh *Google Cloud Platfrom* untuk setiap aplikasi yang dikembangkan.

2.7.2. Android SDK (*Software Development Kit*)

Menurut (Maiyana, 2018) Android SDK (*Software Development Kit*) adalah *tools API (Application Programming Interface)* yang diperlukan untuk memulai pengembangan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Pada Android SDK (*Software Development Kit*) ini terdiri dari *debugger, libraries, handset emulator, dokumentasi, kode contoh dan tutorial.*

Sedangkan Menurut (Yudhanto & Wijayanto, 2018) Android SDK (*Software Development Kit*) adalah *tools API (Application Programming Interface)* yang diperluakan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemograman java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi. Android memberi anda kesempatan membuat aplikasi yang kita butuhkan yang bukan aplikasi bawaan *handphone / smartphone* Beberapa fitur-fitur Android yang paling penting adalah:

1. *Framework* Aplikasi yang mendukung penggantian komponen *reusable.*
2. *Integrated browser* berdasarkan *engine open source Webkit.*
3. SQLite untuk menyimpan data.

2.7.3. JDK (*Java Development Kit*)

Menurut (Haqi, 2019) JDK (*Java Development Kit*) merupakan perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode Java menjadi *bytecode* yang dapat dimengerti dan dapat dijalankan oleh JRE (*Java Runtime Environment*).

JDK (*Java Development Kit*) wajib terinstal pada komputer yang akan melakukan proses pembuatan aplikasi berbasis Java. Namun, JDK (*Java Development Kit*) tidak wajib terinstal di komputer yang akan menjalankan aplikasi yang dibangun menggunakan Java.

Sedangkan menurut (Maiyana, 2018) JDK (*Java Development Kit*) adalah Paket fungsi API (*Application Programming Interface*) untuk bahasa pemrograman Java, meliputi JRE (*Java Runtime Environment*) dan JVM (*Java Virtual Machine*).

2.7.4. *SQLite Database*

Menurut (Kusniyati et al., 2016) SQLite merupakan sebuah system management basis data relasional yang bersifat ACID (*Atomicity, Consistency, Isolation, Durability*) - *compliant* dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C. SQLite merupakan proyek yang bersifat publik *domain* yang dikerjakan oleh D. Richard Hipp.

SQLite mengimplementasikan hampir seluruh elemen-elemen standar yang berlaku pada SQL-92, termasuk transaksi yang bersifat atomic, konsistensi basis data, isolasi, dan durabilitas (dalam bahasa inggris lebih sering disebut ACID), trigger, dan kueri-kueri yang kompleks. Untuk menggunakan SQLite dalam pembuatan *database* pada Android tidak tersedia secara otomatis, melainkan harus *created database* sendiri, mendefinisikan *table, index*, serta datanya. Untuk membuat dan membuka *database* menggunakan *libraries*, yaitu :

Import android.database.sqlite.SQLiteOpenHelper.

Libraries tersebut menyediakan tiga metode yaitu:

1. *Constructor*, menyediakan representasi versi dari *database* skema yang kita gunakan.
2. *onCreate()*, menyediakan SQLite *Database object* yang kita butuhkan dalam definisi *table* dan inisialisasi data.

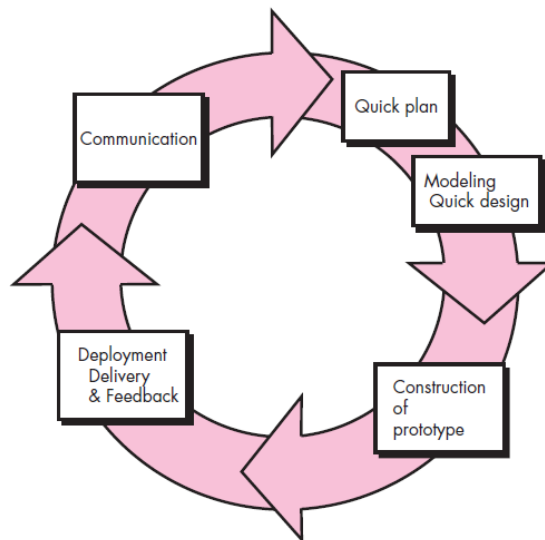
3. *onUpgrade()*. Menyediakan fasilitas *convert database* dari skema *database* versi lama ke skema *database* versi yang baru.

2.8. Metode Perangkat Lunak

2.8.1. *Prototype*

Menurut (Andikos & Gusteri, 2016) *Prototype* merupakan suatu metode dalam pengembangan sistem yang menggunakan pendekatan untuk membuat sesuatu program dengan cepat dan bertahap sehingga segera dapat dievaluasi oleh pemakai. *Prototype* mewakili model produk yang akan dibangun atau mensimulasikan struktur, fungsional, dan operasi sistem.

Salah satu metode siklus hidup sistem yang didasarkan pada konsep model bekerja (*working model*) dimana kebutuhan (*requirement*) diubah kedalam sistem yang bekerja (*Working system*). Metode pengembangan perangkat lunak yang menggunakan pendekatan untuk membuat sesuatu program dengan cepat dan bertahap memungkinkan adanya hubungan antara pengembang sistem dan pengguna sistem yang akibatnya bisa mengatasi ketidakserasian antara pengembang & pengguna. Walaupun begitu *Prototype* bukanlah sesuatu yang lengkap, tetapi sesuatu yang harus di evaluasi dan di *modifikasi* kembali secara terus menerus dan diperbaiki melalui kerjasama antara pengguna dan pengembang.



Gambar 2. 3 Diagram *Prototype*

Adapun dalam tahapan-tahapan dalam metode *Prototype* sesuai pada gambar adalah sebagai berikut:

1. Komunikasi (*Communication*) : pengumpulan data awal, yaitu komunikasi dengan klien dan *user* untuk menentukan kebutuhan.
2. Perencanaan Cepat (*Quick Plan*) : pembuaan perencanaan analisis terhadap kebutuhan pengguna.
3. Pemodelan Perancangan Cepat (*Modeling Quick Design*) : membuat rancangan desain program.
4. Pembentukan *Prototype* (*Construction of prototype*) : pembuatan aplikasi berdasarkan dari pemodelan desain yang telah dibuat.
5. Penyerahan Sistem dan Umpan Balik (*Development Delevery and Feedback*) : memproduksi perangkat secara benar sehingga dapat digunakan oleh pengguna.

2.8.2. UML (*Unfield Modeling Language*)

Menurut (Hendini, 2016) UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML (*Unified Modeling Language*) merupakan metodologi dalam mengembangkan

sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem.

Sedangkan menurut (Marini, 2019) Perencanaan berorientasi obyek biasanya menggunakan model yang dikenal dengan UML (*Unified Modeling Language*) yang merupakan sebuah bahasa pemodelan objek standar sebagai ganti dari pendekatan atau metode berorientasi objek standar. UML (*Unified Modeling Language*) adalah satu kumpulan konveksi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek.

UML (*Unified Modeling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan karena UML (*Unified Modeling Language*) menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancangan dengan yang lain.

2.8.3. XML (*eXtensible Markup Language*)

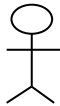
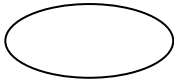
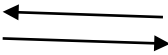
Menurut (Mulyadi et al., 2017) XML (*eXtensible Markup Language*) merupakan suatu bahasa yang digunakan untuk mendeskripsikan dan memanipulasi dokumen secara terstruktur. Secara teknis XML (*eXtensible Markup Language*) didefinisikan sebagai suatu bahasa *meta-markup* yang menyediakan format tertentu untuk dokumen-dokumen yang mempunyai data terstruktur. Bahasa *markup* adalah mekanisme untuk mengenal suatu struktur didokumen.

2.8.4. Use Case Diagram

Menurut (M. D. Irawan & Simargolang, 2018) *Use case diagram* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. *Use case diagram* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat.

Komponen *use case* terdapat pada tabel 2.2

Tabel 2. 2 Komponen-komponen *Use case diagram*.


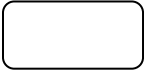
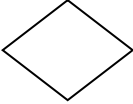
Simbol	Keterangan
Aktor 	Mendefinisikan entitas diluar sistem yang memakai sistem.
<i>Use Case</i> 	Gambaran fungsinalitas dari suatu sistem, sehingga user mengerti kegunaan dari sistem yang akan dibangun.
Relasi 	Menceritakan suatu hubungan antara aktor dan <i>use case</i> sehingga diagram dapat mudah dipahami.



2.8.5. Activity Diagram

Menurut (M. D. Irawan & Simargolang, 2018) *Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis.

Berikut adalah simbol-simbol pada diagram aktivitas pada table 2.3

Tabel 2. 3 Komponen-komponen *Activity Diagram*.

Simbol	Keterangan
Status Awal 	Status Awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan oleh sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan 	Asosiasi percabangan dimana ada pilihanaktivitas lebih dari satu.

Penggabungan 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.



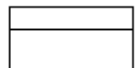
2.8.6. Class Diagram

Menurut (Kusniyati et al., 2016) *Class diagram* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.

Sedangkan menurut (Hendini, 2016) *Class diagram* merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Penjelasan simbol *class diagram* terdapat pada tabel 2.4

Tabel 2. 4 Simbol *Class Diagram*.

SIMBOL	KETERANGAN
<i>Generalization</i> 	Menggambarkan relasi generalisasi.
<i>Nary Association</i> 	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
<i>Class</i> 	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.

2.8.7. Black Box Testing

Pengujian *Black Box* adalah pengujian yang memverifikasi hasil eksekusi aplikasi berdasarkan masukan yang diberikan (data uji) untuk memastikan fungsional dari aplikasi sudah sesuai dengan persyaratan (*requirement*) (Febrian et al., 2020)

Sedangkan menurut (Debiyanti et al., 2020). Pengujian terhadap perangkat lunak sangat penting dilakukan dengan tujuan untuk memberikan jaminan kualitas perangkat lunak yang dihasilkan agar bebas dari terjadinya kesalahan .

Metode pengujian *Equivalence Partitions* membagi domain masukan dari program ke dalam masing-masing kelas data. *Equivalence Partitioning* merupakan sebuah pengujian berdasarkan masukkan data pada setiap form yang memecah domain masukan ke dalam kelompok valid dan tidak valid (Aziz et al., 2020).

2.9. Penelitian Terkait

Dalam penyusunan skripsi ini, penulis terinspirasi dan mereferensi dari penelitian-penelitian sebelumnya yang berkaitan dengan skripsi ini. Daftar penelitian ditunjukkan pada table 2.5.

Tabel 2. 5 Penelitian Terkait

No	Nama	Judul	Tahun	Uraian
1	Dinar Ajeng Kristiyanti.	Penerapan Metode <i>Waiting Line</i> Untuk Evaluasi Pelayanan Penjualan <i>Merchandise</i> Kampus Pada Pt. Come Indonusa Jakarta	2018	Penelitian ini untuk mengelola antrian pelayanan penjualan <i>merchandise</i> kampus yang masih belum optimal di PT. COME Indonusa Jakarta.
2	Devi Yuliana, Julius Santony, Sumijan.	Model Antrian <i>Multi Channel Single Phase</i> Berdasarkan Pola Kedatangan Pasien	2019	Untuk mengetahui jumlah pasien dalam sistem dan waktu tunggu yang dihabiskan oleh pasien dalam antrian berdasarkan pola kedatangan pasien agar terbagi menjadi

		Untuk Pengambilan Obat Di Apotik		beberapa waktu dan jalur yang disesuaikan dengan kebutuhan waktu yang diinginkan pasien.
3	Harni Kusniyati , Nicky Saputra Pangondian Sitanggung	Aplikasi Edukasi Budaya Toba Samosir Berbasis Android	2016	Upaya dalam mempermudah mengakses informasi budaya toba samosir dalam mengenalkan informasi mengenai budaya, wisata dan juga kesenian khas toba samosir kepada user.
4	Ehtur Enjelita Gultom , Dwi Oktarina	Rancang Bangun Sistem Informasi Pemesanan Antrian <i>Service</i> Mobil Berbasis Android	2019	Berfungsi untuk mempermudah perusahaan dan costumer dalam memperoleh informasi perawatan dan perbaikan terakhir kali <i>service</i> mobil agar menjadi lebih efektif dan efesien.