BAB II

LANDASAN TEORI

2.1 Mobile

Dalam (Prakarsya, A., 2019) Menjelaskan bahwa, *mobile* adalah suatu istilah yang digunakan untuk menggambarkan aplikasi pada piranti berukuran kecil, portable, dan wireless serta mendukung komunikasi. Konsumen menginginkan perangkat yang kecil untuk kenyamanan dan mobilitas mereka dan Perangkat mobile juga hanya menghabiskan sedikit daya dibandingkan dengan mesin desktop.

Kata *mobile* mempunyai arti bergerak atau berpindah, sehingga aplikasi *mobile* adalah sebutan untuk aplikasi yang berjalan di *mobile device*. Dengan menggunakan aplikasi *mobile*, dapat dengan mudah melakukan berbagai macam aktifitas mulai dari hiburan, berjualan, belajar, mengerjakan pekerjaan kantor, *browsing* dan lain sebagainya.

2.2 Android

Dalam (Sari, Y. P., & Ali, R, 2019) menjelaskan bahwa *Android* adalah sebuah sistem operasi untuk smartphone dan tablet. Sistem operasi dapat diilustrasikan sebagai jembatan antara piranti (*device*) dan penggunanya, sehingga pengguna bisa berinteraksi dengan *device*-nya dan menjalankan aplikasi – aplikasi yang tersedia pada *device*.

***Android* pertama kali dirilis pada bulan oktober 2003 oleh Andy Rubin, Rich Miner, Nich Sears dan Chris White di bawah sebuah perusahaan bernama *android Inc* di Palo Antom, California. Dan akhirnya *android* diakuisisi oleh *google* pada tahun 2005. *Android* sendiri memiliki beberapa kelebihan yaitu adalah sebagai berikut :**

a. ***Open Source* alias Gratis**
b. **Cepat dan *Responsive***

c. ***User Friendly***
d. **Variasi harga produk yang beragam**
e. ***Google*** **sebagai pengembang**
f. ***Hardware*** **pendukung yang beragam**

2.3 Perangkat Lunak Pengembangan Sistem

Pengembangan sistem untuk membangun aplikasi pemesanan berbasis *mobile android* diperlukan beberapa perangkat lunak yang digunakan dalam membangun aplikasi tersebut. Beberapa perangkat lunak yang digunakan adalah sebagai berikut

2.3.1 Android studio

Firly (2018) menjelaskan bahwa *Android studio* merupakan *integrated development environment* (*IDE*) atau dalam artian lain adalah sebuah lingkungan pengenbangan terintegrasi resmi yang memang merancang khusus untuk pengembangan system operasi *Google Android.* Aplikasi ini dibangun di atas sebuah perangkat lunak yang dinamakan *IntelliJ IDEA* milik *JetBrains.* Bisa juga dibilang bahwa *android* studio merupakan pengganti dari *Eclipse android development tool* atau *ADT* sebagai *IDE* utama dalam pengembangan aplikasi *android* yang asli.

Android studio* diluncurkan pada tanggal 16 mei 2013 dalam *konferensi google I/O* yang pada saat itu masih dalam tahap pratinjau akses *versi* 0.1 sebagai perintis. Hingga pada akhirnya *versi* stabil 3.0 yang *liris* pada pertengahan bulan oktober 2017 dan menjadi *software* terlaris dikalangan *developer* muda. Aplikasi ini dapat digunakan diberbagai sistem operasi yaitu *windows*, *linux* dan *macOS.

Aplikasi ini menawarkan berbagai fitur canggih yang akan meningkatkan kemampuan *produktivitas* dalam proses pengembangan aplikasi. Berikut ini

adalah beberapa hal yang akhirnya banyak mengundang *developer* untuk melirik *android studio* sebagai *software* pengembang :

a. **Dukungan dari C*++, NDK* dan sekarang *kotlin***
b. **Perkembangan yang *up to date***
c. **Sistem berbasis *Gradle* yang dinilai fleksibel**
d. **Lingkungan yang mencakup seluruh perangkat *android***
e. ***Emulator* yang cepat dan kaya akan fitur**
f. **Alat pengujian dan kerangka yang juga *ekstensif***
g. ***Instant Run***
h. **Dukungan *google cloud platform***

2.3.2 Java

Firly (2018) menjelaskan bahwa *Java* adalah bahasa pemrograman *multi platform*. *Java* tidak menyediakan *IDE* khusus seperti halnya bahasa pemrograman yang lain. Pemrogram bisa menggunakan *IDE* yang support ke *java*, misalnya *Netbeans*, *Eclips*, *TexPad*, dan lain-lain. elemen-elemen dasar pemrograman Java terdiri dari Himpunan karakter, Pengenal (*identifier*), Kata Kunci, Tipe Data *Primitif*. Tipe data *primitif* yang didukung oleh bahasa pemrograman *Java* adalah *byte*, *short*, *int*, *long*, *float*, *double*, *Boolean*, *char*.

2.3.3 MySQL

Dalam (Solichin, A. 2016) menjelaskan bahwa MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang *multithread, multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. Selain itu MySQL *software* merupakan suatu aplikasi yang sifatnya *open source* serta *server* basis data MySQL memiliki kinerja sangat cepat, reliable dan mudah untuk digunakan serta bekerja dengan arsitektur *client server* atau *embedded systems*.

2.3.4 Adobe XD

***Adobe XD* adalah perangkat lunak yang bisa digunakan oleh para *desainer* aplikasi *mobile*. *Adobe xd* bisa memudahkan *desainer* aplikasi *mobile* dalam pengembangan *UX/UI*, *adobe xd* ini sudah menyediakan fitur *UI* Desain dan juga *UX* Desain sebagai *prototype* tanpa membutuhkan *third-party* atau aplikasi lain untuk membantu membuat sebuah *prototype*. (*garudapixel.com, dewaweb.com*)**

2.4 Multilevel Feedback Queue

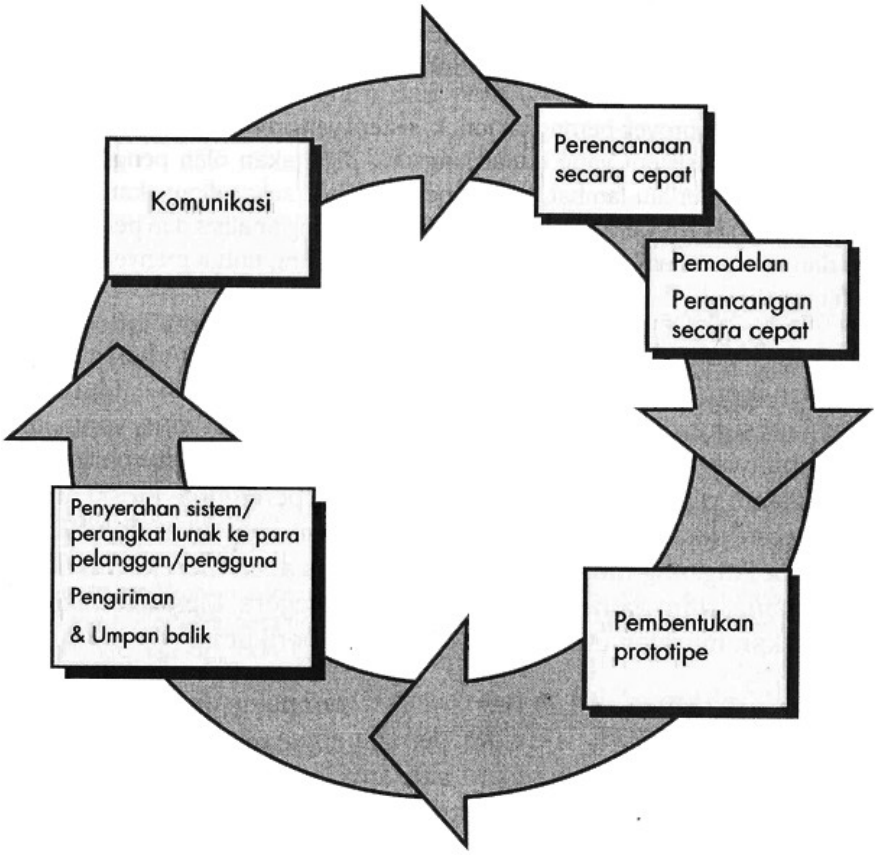
Dalam (Verawati, I., & Sulistiyono, M. 2018) menjelaskan bahwa *Multilevel feedback queue* adalah salah satu algoritma yang berdasar pada model antrian *Multi Channel Single Server*. Kelebihan mendasar yang dimiliki oleh *multilevel feedback queue* adalah kemungkinan adanya suatu proses berpindah dari satu antrian ke antrian lainnya, misalnya dengan prioritas yang lebih rendah ataupun lebih tinggi. Algoritma ini didefinisikan melalui beberapa parameter, antara lain :

1. **Jumlah antrian.**
2. **Algoritma penjadwalan tiap antrian.**
3. **Kapan menaikkan proses ke antrian yang lebih tinggi.**
4. **Kapan menurunkan proses ke antrian yang lebih rendah.**
5. **Antrian mana yang akan dimasuki proses yang membutuhkan.**

Algoritma ini biasa digunakan dalam pengembangan *Operating System.* Terutama dalam penjadwalan CPU. Jika suatu proses menyita CPU terlalu lama, maka proses itu akan dipindahkan ke antrian yang lebih rendah. Hal ini menguntungkan proses interaksi karena proses ini hanya memakai waktu CPU yang sedikit. Demikian pula dengan proses yang menunggu terlalu lama. Proses ini akan dinaikkan tingkatannya. Ada tiga jenis kategori yang

digunakan dalam *multilevel feedback queue* ini, yaitu *High Priority, Normal Priority* dan *Low Priority*.

2.5



Metode Pengembangan Perangkat Lunak

Pressman (2012) menjelaskan meskipun pembuatan *prototype* dapat di gunakan sebagai model proses yang berdiri sendiri, pembuatan *prototype* lebih umum digunakan sebagai teknik yang dapat diimplementasikan di dalam kontek setiap model proses perangkat lunak, dan paradigma pembuatan *prototype* seringkali membantu tim pengembang perangkat lunak dan para *stakeholder* untuk memahami lebih baik apa yang akan dikembangkan saat spesifikasi kebutuhan belum jelas.

Berikut ini adalah penjelasan dari gambar 2.2

Gambar 2.1 Tahapan Pengembangan Aplikasi

1. **Komunikasi**

Tahap komunikasi ini adalah tahapan komunikasi antara *developer* dan pelanggan mengenai tujuan pembuatan *software*, mengidentifikasi apakan kebutuhan diketahui.

2. **Perancangan secara cepat**

Tahap perancangan secara cepat ini adalah tahapan perancangan cepat setelah terjalin komunikasi.

3. **Pemodelan perancangan secara cepat**

Tahap pemodelan perancangan secara cepat ini adalah tahapan segera membuat model, dan pemodelan cepat fokus pada gambaran dari segi *software* apakah *visible* menurut pelanggan.

4. **Pembentukan prototipe**

Tahap pembentukan prototipe ini adalah tahapan pemodelan cepat menuntun pada pembuatan dari prototipe.

5. **Penyerahan sistem / perangkat lunak ke para pelanggan / pengguna pengiriman dan umpan balik**

Tahap penyerahan sistem / perangkat lunak ke para pelanggan / pengguna pengiriman dan umpan balik ahapan ini adalah prototipe yang dikirimkan kemudian dievaluasi oleh pelangan.

2.6 Pengujian Kotak Hitam

Pressman (2012) menjelaskan bahwa pengujian kotak hitam, juga disebut pengujian perilaku, yang berfokus pada persyaratan fungsional perangkat lunak. Teknik Pengujian kotak hitam memungkinkan untuk membuat beberapa kumpulan kondisi masukan yang sepenuhnya akan melakukan semua kebutuhan fungsional untuk program. Pengujian kotak hitam berupaya untuk menemukan kesalahan dalam kategori berikut :

1. **fungsi yang salah atau hilang,**
2. **kesalahan antarmuka,**
3. **kesalahan dalam struktur data atau akses basis data eksternal,**
4. **kesalahan perilaku atau kinerja, dan**
5. **kesalahan inisialisasi dan penghentian.**

2.7 Unifed Modelling language (UML)

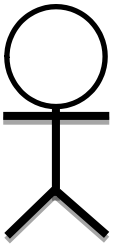
Dalam (Marini, M. 2019) menjelaskan bahwa *Unified Modeling Language (UML*) merupakan sebuah bahasa pemodelan objek standar sebagai ganti dari pendekatan atau metode berorientasi objek standar. UML adalah satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek. UML dapat digunakan untuk mempermudah pengembangan Aplikasi yang berkelanjutan.

2.7.1 Use Case Diagram

Dalam (Marini, M. 2019) menjelaskan bahwa *Use Case Diagram* menggambarkan kebutuhan sistem yang di butuhkan dan dengan nyata siapa saja yang akan menggunakan sistem dan dengan cara apa pemakai

dapat saling berhubungan dengan sistem. Dengan kata lain, *Use Case Diagram* menggambarkan hubungan antara Aktor dan U*se Case.* Simbol – simbol *Use Case* dapat dilihat pada tabel 2.1

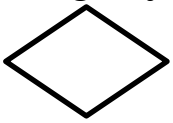
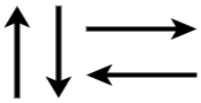
Tabel 2.1 Simbol *Use Case Diagram*

Simbol	Keterangan
Use Case 	**Fungsionalitas yang disediakan sistem sebagai unit – unit yang saling bertukar pesan antar unit dan aktor.**
Aktor / *Actor* 	**Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi.**
Asosiasi / *Association* 	**Komunikasi antar aktor dan *Use Case* yang berpartisipasi.**
Ekstensi / *extend* **<<extend>>** 	**Relasi Use Case tambahan ke sebuah Use Case dimana Use Case yang ditambah dapat berdiri sendiri walau tanpa *Use Case* tambahan.**
Generalisasi / generalization 	**Hubungan generalisasi dan spesialisasi antara dua buah *Use Case* yang mana fungsi yang satu lebih umum dari yang lainnya.**
Include / Use Case **<<include>>**  **uses**	**Relasi *Use Case* tambahan ke sebuah *Use Case* dimana *Use Case* yang ditambahkan memerlukan *Use Case* ini untuk menjalankan fungsinya.**

2.7.2 Activity Diagram

Dalam (Marini, M. 2019) menjelaskan bahwa *Activity diagram* menggambarkan *workflow* (alir kerja) atau aktivitas dari sebuah system atau proses bisnis yang ada pada menu perangkat lunak. Tahap perancangan *activity* diagram menjabarkan masing – masing activity pada perancangan *use case.* Simbol – simbol *Activity Diagram* dapat dilihat pada tabel 2.2.

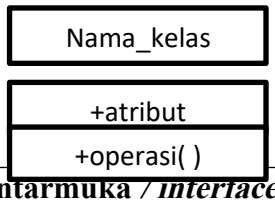
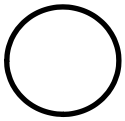
Tabel 2.2 Simbol *Activity Diagram*

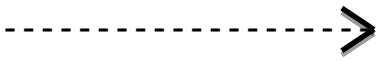
Simbol	Keterangan
Aktivitas 	**Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.**
Status awal 	**Bagaimana objek dibentuk atau diawali.**
Status akhir 	**Bagaimana objek dibentuk dan diakhiri.**
Percabangan / join 	**Asosiasi percabangan dimana jika ada pilihan aktifitas lebih dari satu.**
Penggabungan / join 	**Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu**
Garis Penghubung 	**Digukana untuk menghubungkan satu simbol dengan simbol lainnya**

2.7.3 Class Diagram

Dalam (Marini, M. 2019) menjelaskan bahwa *Class Diagram* mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat di antara mereka. *Class diagram* adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang sedang kita gunakan. *Class diagram* memberi kita gambaran tentang sistem atau perangkat lunak dan relasi-relasi yang ada didalamnya. *Class diagram* juga menunjukan *property* dan operasi sebuah *class* dan Batasan - batasan yang terdapat dalam hubungan - hubungan objek tersebut. Simbol – simbol yang ada pada *Class Diagram* ditunjukan oleh Tabel 2.3.

Tabel 2.3 Simbol *Class Diagram*

Simbol	Deskripsi
Kelas 	**Kelas pada struktur system.**
Antarmuka / *interface*  **Nama_*interface***	**Sama dengan konsep *interface* dalam pemograman berorientasi objek.**
Asosisasi / *association* 	**Relasi antarkelas dengan makna umum, asosia biasanya disertai dengan *multiplicity*.**
Asosisasi berarah / *directed association* 	**Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.**
generalisasi 	**Relasi antarkelas dengan makna Generalisasi – spesialisasi (umum khusus).**

Kebergantungan / *dependency* 	**Relasi antarkelas dengan makna kebergantungan antarkelas.**
Agregasi / *aggregation* 	**Relasi antar kelas dengan makna semua –bagian**

2.8 Penelitian Terdahulu

Berikut merupakan jurnal terkait dengan penelitian terdahulu :

No	Nama	Judul	Terbit / Tahun	Keterangan
1	Asilah salma, Irfan darmawan & Faisal mufied	Perancangan Aplikasi Callme Berbasis Android menggunakan metode Prototyping (Modul administrasi Customer dan Admin)	e-Proceeding of Engineering / 2017	Penelitian ini membahas mengenai perancangan aplikasi callme berbasis android untuk mempermudah dalam pemesanan makanan dan menghemat biaya costumer karena tidak melalui pihak ketiga. Metode yang digunakan dalam perancangan aplikasi ini adalah metode Prototyping.
2	Adrianus, Dian Andrian Ginting	Implementasi Algoritma Multilevel Feedback Queue Untuk Pembuatan Aplikasi Pemesanan Makanan Pada Restoran Dengan Platform Android Dan IOS	Universitas Lampung / 2015	Penelitian ini membahas tentang pembuatan aplikasi web pemesanan makanan pada Android dan iOS dengan menerapkan metode *Multilevel Feedback Queue* untuk membuat antrian pemesanan makanan pada koki
3	Rony Adi Nugroho	Sistem Informasi Pemesanan Jasa Fotografi Berbasis Web Pada Karma	Universitas Dian Nuswantoro / Semarang 2015	Penelitian ini membahas mengenai sebuah perusahaan yang bergerak dibidang jasa fotografi. Untuk memberikan pelayanan terbaik bagi

		Kreatif Semarang		**pelanggan dan mendapatkan kepercayaan dari pelanggan. Untuk mewujudkan visi tersebut maka diperlukan suatu sarana informasi yang berbasis web.**