BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Penelitian yang dilakukan oleh Qurotul Aini, et al. pada tahun 2017 dengan judul "Penerapan Absensi QR *Code* Mahasiswa Bimbingan Belajar pada *Website* berbasis YII *Framework*". Penelitian tersebut menghasilkan sebuah aplikasi berbasis web yang dapat digunakan sebagai sistem absensi bimbingan belajar. Yang membedakan dari penelitian yang akan dibuat adalah aplikasi yang akan dibuat saat ini menggunakan *smartphone* android untuk sistem absensi menggunakan QR *Code.*

Penelitian lain yang dilakukan oleh Nurmaliana Pohan, 2016 yang berjudul "Implementasi *Barcode* untuk Sistem Informasi Absensi pada PT. Coca Cola Distribution Indonesia Pekanbaru". Penelitian tersebut menghasilkan sebuah aplikasi untuk sistem absensi menggunakan *barcode* untuk karyawan pada sebuah perusahaan. Aplikasi yang dihasilkan pada penelitian tersebut berbasis *web* dan menggunakan *barcode*, sehingga terdapat perbedaan dengan penelitian yang akan dilaksanakan, yaitu pada penelitian yang akan dilaksanakan sistem absensi dilakukan menggunakan *smartphone* android dan juga QR *Code*.

II.2. Uraian Teoritis

II.2.1. Aplikasi

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanankan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus komputer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan. Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia, "Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu". (Andi Juansyah ; 2015 : 2)

II.2.2. Absensi

Absensi dapat dikatakan suatu pendataan kehadiran yang merupakan bagian dari aktifitas pelaporan yang ada dalam sebuah institusi. Absensi disusun dan diatur sehingga mudah untuk dicari dan dipergunakan ketika diperlukan oleh pihak yang berkepentingan. Secara umum, jenis-jenis absensi menurut cara penggunaannya dapat dikelompokkan menjadi dua, yaitu :

1. Absensi manual, yang merupakan cara penulisan kehadiran dengan cara menggunakan pena berupa tanda tangan
2. Absensi non manual, yang merupakan cara penulisan kehadiran dengan menggunakan alat yang terkomputerisasi. (Setiawan dan Kurniawan ; 2015 :

44-45)

II.2.3. QR Code

Kode QR adalah suatu jenis kode matriks atau kode batang dua dimensi yang dikembangkan oleh Denso Wave, sebuah divisi Denso *Corporation* yang merupakan sebuah perusahaan Jepang dan dipublikasikan pada tahun 1994. Agar dapat membaca QR *Code* diperlukan sebuah pembaca atau pemindai berupa *software* yaitu QR *Code Reader* atau QR *code Scanner* yang harus diinstal pada perangkat telepon *mobile*. QR merupakan singkatan dari *quick response* atau respons cepat, yang sesuai dengan tujuannya adalah untuk menyampaikan informasi dengan cepat dan mendapatkan respons yang cepat pula. Berbeda dengan kode batang, yang hanya menyimpan informasi secara horizontal, kode QR mampu menyimpan informasi secara horizontal dan vertikal. (Sugiantoro dan Hasan ; 2015 : 134-135)

II.2.4.Pemrograman Java

Java adalah sebuah bahasa pemrograman yang populer dikalangan para akademisi dan praktisi komputer. Java pertama kali dikembangkan untuk memenuhi kebutuhan akan sebuah bahasa komputer yang ditulis satu kali dan dapat dijalankan dibanyak sistem komputer berbeda tanpa perubahan kode berarti. Pada umumnya, para pakar pemrograman berpendapat bahwa bahasa Java memiliki konsep yang konsisten dengan teori pemrograman objek dan aman untuk digunakan.

Java sampai saat ini masih merupakan bahasa pemrograman yang masih sangat diminati dan banyak digunakan oleh para *programer* dan *software developer* untuk mengembangkan berbagai tipe aplikasi, mulai dari aplikasi *console*, aplikasi *desktop*, *game*, dan *applet* (aplikasi yang berjalan di lingkungan *web browser*), sampai ke aplikasi-aplikasi yang berskala *enterprise*. Untuk memenuhi kebutuhan tipe aplikasi yang beragam tersebut, Java dikategorikan menjadi tiga edisi, yaitu : J2SE (Java 2 *Platform Standart Edition*) untuk membuat aplikasi-aplikasi *desktop* dan *apllet*, J2EE (Java 2 *Platform Enterprise Edition*) untuk membuat aplikasi-aplikasi multitier berskala *enterprise*, dan J2ME (Java 2 *Platform Micro Edition*) untuk membuat aplikasi-aplikasi yang dapat dijalankan dilingkungan perangkat-perangkat mikro seperti *handphone*, PDA dan *Smartphone*. (Wardani dan Yaqin ; 2013 : 473-477)

II.2.5. Aplikasi *Mobile*

Aplikasi *mobile* dapat diartikan sebagai sebuah produk dari sistem komputasi *mobile*, yaitu sistem komputasi yang dapat dengan mudah dipindahkan secara fisik dan yang komputasi kemampuan dapat digunakan saat mereka sedang dipindahkan. Contohnya adalah *Personal Digital Assistant* (PDA), *smartphone* dan ponsel. (Ramadhan dan Utomo ; 2014 : 47-55)

II.2.6. Android

Menurut Arifianto Teguh, android adalah sebuah *platform* pertama yang betul-betul terbuka dalam pengembangannya dan komperehensif untuk perangkat

mobile, semua perangkat lunak yang ada difungsikan menjalankan sebuah *device mobile* tanpa memikirkan kendala kepemilikan yang menghambat inovasi pada teknologi *mobile*. Dalam definisi lain, Android merupakan subset perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware,* dan aplikasi inti yang dirilis oleh Google. Sedangkan Android *SDK (Software Development Kit)* menyediakan *tools* dan API yang diperlukan untuk mengembangkan aplikasi pada *platform* Android dengan menggunakan bahasa pemrograman Java.

Aplikasi *Android* ditulis dalam bahasa pemrograman *java*, yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi Android, sebuah *file* yang ditandai dengan akhiran *.apk. file* inilah yang didistribusikan sebagai aplikasi dan diinstal pada *handset Android*. *File* ini diunduh oleh pengguna ke perangkat *mobile* mereka. Semua kode dijadikan satu *file .apk,* dan kemudian kita sebut sebagai sebuah aplikasi.(Ahmad : 2015 : 190-200)

II.2.7. Android *Studio*

Android *studio* adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat *opensource* 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 mengantikan *Eclipse* sebagai IDE 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 aplikasi android lebih mudah
f. Didukung oleh Google *Cloud Platfrom* untuk setiap aplikasi yang dikembangkan. (Andi Juansyah ; 2015)

II.2.8. Android SDK (*Software Development Kit*)

Android SDK mencakup perangkat *tools* pengembangan yang komprehensif. Android SDK terdiri dari *debugger*, *libraries*, *handsetemulator*, dokumentasi, contoh kode program dan tutorial. Saat ini Android sudah mendukung arsitektur x86 pada Linux (distribusi Linux apapun untuk *desktop modern*), Mac OS X 10.4.8 atau lebih, Windows XP atau Vista. Persyaratan mencakup JDK, *Apache Ant* dan Python 2.2 atau lebih. IDE yang didukung secara resmi adalah Eclipse 3.2 atau lebih dengan menggunakan *plugin Android Development Tools* (ADT), dengan ini pengembang dapat menggunakan IDE untuk mengedit dokumen Java dan XML serta menggunakan peralatan *commandline* untuk menciptakan, membangun, melakukan *debug* aplikasi

Android dan pengendalian perangkat Android (misalnya *reboot*, menginstal paket perangkat lunak). (Sulihati dan Andriyani ; 2016 : 20)

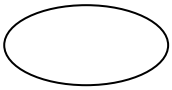
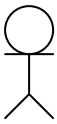
II.2.9. Pengertian UML

UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodalan umum dalam industri perangkat lunak dan pengembangan sistem (Windu dan Grace ; 2013 : 81). Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.2.9.1. *Use Case Diagram*

Use case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. (Windu dan Grace ; 2013). Adapun *UseCase* Diagram dapat dilihat pada Tabel II.1. sebagai berikut :

Tabel II.1. *UseCase* Diagram

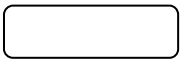
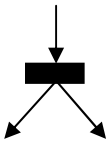
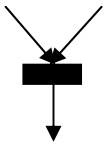
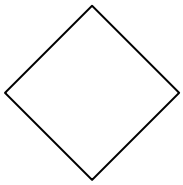
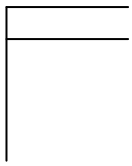
Gambar	Keterangan
	*UseCase* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktir, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktir, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *UseCase*, tetapi tidak memiliki kontrol terhadap *usecase*
	Asosiasi antara aktor dan *usecase*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
	Asosiasi antara aktor dan *usecase* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem
	Include, merupakan di dalam *usecase* lain (*required*) atau pemanggilan *usecase* oleh *usecase* lain, contohnya adalah pemanggilan sebuah fungsi program
	Extend, merupakan perluasan dari *usecase* lain jika kondisi atau syarat

(Sumber : Ade Hendini ; 2016)

II.2.9.2. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Windu dan Grace ; 2013 : BIT. 10. 80-87). Adapun *Activity* Diagram dapat dilihat pada Tabel II.2. sebagai berikut :

Tabel II.2. *Activity* Diagram

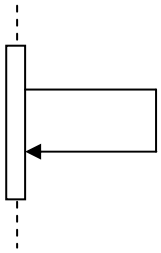
Gambar	Keterangan
	*StartPoint*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*EndPoint*, akhir aktivitas
	*Activities*, menggambarkan suatu proses/kegiatan bisnis
	*Fork*/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabung kan dua kegiatan paralel menjadi satu
	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi
	*DecisionPoints*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

(Sumber : Ade Hendini ; 2016)

II.2.9.3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan obyek pada *usecase* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Windu dan Grace ; 2013). Adapun *Sequence* Diagram dapat dilihat pada Tabel II.3. sebagai berikut :

Tabel II.3. *Sequence Diagram*

Gambar	Keterangan
	*EntityClass*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data
	*BoundaryClass*, berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *formentry* dan *form* cetak
	*Controlclass*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek
	Message, simbol mengirim pesan antar *class*
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
	*Activation*,mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

(Sumber : Ade Hendini ; 2016)

II.2.9.4. *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. *Class diagram* juga menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan obyek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas(*Class*), *Relasi, Associations, Generalization* dan *Aggregation,* Atribut *(Attributes),* Operasi *(Operations/Method),* dan *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar Kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau kardinaliti (Windu dan Grace ; 2013). Adapun *Multiplicity Class* Diagram dapat dilihat pada Tabel II.4. sebagai berikut :

Tabel II.4. *Multiplicity Class* Diagram

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimal 4

(Sumber : Ade Hendini ; 2016)