BAB II
TINJAUAN PUSTAKA

2.1 Aplikasi

Aplikasi adalah program siap pakai yang dirancang untuk melakukan tugas bagi pengguna layanan perangkat lunak yang dapat berjalan pada komputer yang berdiri sendiri, dari aplikasi sederhana hingga aplikasi besar atau kompleks [4].

2.2 *Reminder*/Notifikasi

Reminder/pengingat adalah jadwal yang bisa digunakan siapa saja untuk mengingat hal-hal penting yang harus dilakukan, notifikasi adalah pesan yang dikirim secara otomatis pada perangkat digital untuk pemegang akun (media sosial, aplikasi online, rekening bank, dan lain-lain) pengingat atau notifikasi berupa pesan untuk mengingatkan seseorang akan sesuatu. Pengingat paling berguna ketika anda ingin memberikan informasi pada waktu dan tempat yang tepat dengan mengatur pemberitahuan dalam bentuk catatan yang berisi informasi, waktu, dan lokasi [5].

2.3 *Google Calendar*

Google calender adalah layanan yang disediakan oleh *google* yang memungkinkan pengembang menemukan dan mengedit informasi *online* dengan mudah. Fitur ini tersedia untuk pengembang dan peneliti yang ingin menggunakan *google* sebagai sumber daya dalam aplikasi mereka, layanan ini memungkinkan pengembang untuk mengakses lebih dari (3) tiga miliar dokumen diinternet lansung dari aplikasi desktop pengembang [6].

2.4 *Google API*

Google Application API adalah antaramuka aplikasi yang dapat mengakses aplikasi *google* seperti *gmail*. *Google Calendar API* memungkinkan pengembang untuk membuat, melihat, dan memodifikasi acara di *google* kalender. Untuk memanggil fungsi *Google Calendar API*, digunakan beberapa *library* bahasa

pemrograman klien, seperti Java, .NET, PHP, Javascript, Ruby, Note.js, Python, dan iOS [7].

2.5 Basis Data

Basis Data adalah tempat penyimpanan data dengan kapasitas penyimpanan yang besar yang dapat digunakan oleh banyak pengguna dalam waktu yang bersamaan. Basis data merupakan kumpulan data yang terhubung ke *file* terkait. Basis data menggabungkan catatan yang disimpan sebelumnya yang disimpan dalam *file* terpisah ke dalam media umum yang menyediakan informasi untuk banyak aplikasi. Informasi yang disimpan dalam *database* tidak tergantung pada program aplikasi yang menggunakannya dan jenis penyimpanannya. Dengan demikian, *database* berisi elemen data yang menggambarkan objek dan hubungan antar objek [8].

SQL Server merupakan *Relational Database Management System* (RDBMS) yang dikembangkan sebagai perangkat lunak oleh *Microsoft*, fungsi utama *SQL Server* adalah untuk menyimpan dan menggunakan data yang disertakan dalam aplikasi pada komputer yang sama atau dari komputer lain dalam jaringan [9].

2.6 *Unified Modeling Language (UML)*

Unified Modelling Language (UML) adalah seperangkat istilah pemodelan yang digunakan untuk mendefiniskan atau menggambarkan sistem perangkat lunak dalam hal objek [10].

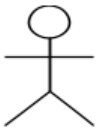
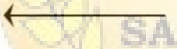
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML (*Unified Modelling Language*) adalah sebagai berikut :

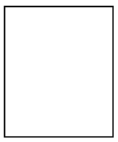
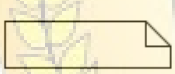
a. *Use Case Diagram*

Use Case diagram merupakan penggunaan pola perilaku (*behavior*) *Use Case* digunakan untuk mendefiniskan fungsi dan beberapa aktivitas atau kemampuan lanjutan dari sistem [11].

apa yang tersedia dalam sistem informasi dan siapa yang diizinkan untuk menggunakan fungsi tersebut. Simbol *Use Case Diagram* dapat dilihat pada tabel 2.1

Tabel 2. 1 ***Use Case Diagram***

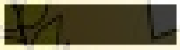
NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*Independen*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*Independen*)
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*)
4		*Inchude*	Menspesifikasikan bahwa use case sumber secara ***eksplisit***
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan

6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dan jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputer

b. Diagram Aktivitas *(Activity Diagram)*

Activity Diagram merupakan gambaran dari berbagai aliran aktivitas dalam suatu sistem yang masih dalam proses perancangan, seperti mulai dari setiap aliran hingga tahap keputusan yang terjadi dan bagaimana aliran sistem tersebut berakhir [12]. Adapun simbol-simbol *Activity Diagram* dapat dilihat pada tabel 2.2

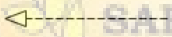
Tabel 2. 2 ***Activity Diagram***

NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antaramuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali
4		*Actifity Final Node*	Bagaimana objek dibentuk dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

c. Diagram Kelas *(Class Diagram)*

Merupakan hubungan antar kelas dan deskripsi rinci dari setiap kelas dalam model, itu menentukan perilaku sistem. Diagram kelas juga menujukan atribut fungsional kelas dan batasan yang terkait dengan objek yang terkait dengannya [13]. Adapun simbol-simbol *Activity Diagram* dapat dilihat pada tabel 2.3

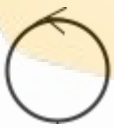
Tabel 2. 3 ***Class Diagram***

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku ada di atasnya objek induk (*anscestor*)
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek
6		*Dependeng*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.

d. *Sequence Diagram* (Diagram urutan)

Sequence Diagram adalah diagram interaktif yang menekankan pengiriman dan penerimaan pesan antar objek [14]. Adapun simbol-simbol *Sequence Diagram* dapat dilihat pada tabel 2.4

Tabel 2.4 Sequence Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menggambar orang yang sedang berinteraksi dengan sistem
2		*Entiti Class*	Menggambarkan hubungan yang akan dilakukan
3		*Boundary Class*	Menggambarkan sebuah gambaran dari foem
4		*Control Class*	Menggambarkan penghubungan antara boundary dengan tabel
5		*A Focus Of Control & A Life Line*	Menggambarkan tempat mulai dan berakhirnya massage
6		*A Massage*	Menggambarkan pengiriman pesan

2.7 .NET Framework

Microsoft .NET Framework (diucapkan *Microsoft Dot-Net Framework*) adalah komponen yang dapat ditambahkan atau dibangun ke dalam sistem operasi *Microsoft Windows* (dimulai dengan *Windows Server* 2003 dan versi *Windows* yang lebih baru). Kerangka kerja ini menyediakan sejumlah besar solusi pemrograman untuk memenuhi kebutuhan umum aplikasi yang ditulis khusus untuk kerangka kerja .NET. Kerangka kerja adalah penawaran utama *Microsoft* dan digunakan untuk sebagian besar aplikasi baru yang dikembangkan untuk platform *Windows*. Pada dasarnya, *.NET Framework* terdiri dari 2 komponen utama: CLR dan *.NET Framework Class Library*.

Program-program yang ditulis untuk *.NET Framework* berjalan pada perangkat lunak yang mengelola persyaratan waktu proses aplikasi. Lingkungan runtime ini, yang juga merupakan bagian dari kerangka .NET, disebut CLR *(Common Language Runtime)* [15].

Pada tahun 2000, Microsoft mengumumkan bahasa pemrograman C# yang dirancang khusus untuk platform .NET dan berakar pada C, C++, dan Java. Seperti Visual Basic, C# berorientasi objek dan memiliki akses ke *.NET Framework Class Library*, kumpulan kaya komponen pra-bangun yang dapat Anda gunakan untuk mengembangkan aplikasi dengan cepat. C# adalah bahasa pemrograman visual. Bahasa visual ini menyediakan beberapa instruksi pemrograman untuk membangun berbagai bagian aplikasi. Fitur GUI (*graphical user interface*) dari visual studio memungkinkan Anda menarik dan melepas objek yang telah ditentukan sebelumnya, seperti tombol dan kotak teks, ke lokasi di layar [16].

2.8 Pengujian *White-Box*

White-Box Testing adalah pengujian yang menggunakan *cyclomatic complexity*. salah satu cara untuk menguji dengan melihat kode program yang ada dengan melihat modul dan menganalisis kesalahannya. Jika *output* yang dihasilkan oleh modul tidak sesuai dengan proses yang benar. Maka barisan program, variabel, dan parameter yang termasuk dalam unit tersebut akan diperiksa atau dikoreksi secara individual dan kemudian dikompilasi ulang [17].

2.9 Pengujian *Black-Box*

Black-Box adalah metode pengujian perangkat lunak tanpa memperhatikan detail perangkat lunak. Pengujian ini hanya memeriksa nilai keluaran berdasarkan setiap nilai masukan. Tidak ada upaya yang dilakukan untuk mencari tahu kode program mana yang digunakan *output*. Proses pengujian *Black-Box* dilakukan dengan menguji aplikasi yang dibuat dengan mencoba memasukkan data pada setiap halaman. Pengujian ini digunakan untuk menentukan apakah aplikasi berfungsi sesuai kebutuhan [18].