BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Dari penelitian Dwi Priyanti dan Siska Iriani (2013), yang berjudul Sistem Rekapitulasi Data Penduduk Pindah Pada Kecamatan Ngadirojo Kabupaten Pacitan bahwa sistem rekapitulasi penduduk yang pindah pada Kecamatan Ngadirojo belum bersifat komputerisasi. Hal ini menyebabkan data penduduk yang pindah tidak pernah tersimpan dengan baik. Dari masalah tersebut tujuan peneliti memindahkan data penduduk yang awalnya di catat dalam buku kemudian data di salin dalam komputer. Dwi dan Siska mengatasi hal tersebut dengan *Microsoft Access*.

Novi Haryo Kusumo (2011), telah melakukan penelitian yang berjudul "Sistem Informasi Penilaian di SMAN 1" bahwa sistem informasi pengolahan nilai di SMAN 1 Karanganyar masih dengan *Microsoft Excel*. Menginputkan semua nilai setiap akhir semester menjadi tanggungjawab besar bagi tim kurikulum. Novi Haryo Kusumo mengatasi hal tersebut dengan membuat sistem informasi penilaian berbasis *web*.

Pembuatan aplikasi pengolahan nilai atau rekapitulasi nilai juga sudah pernah dilakukan oleh peneliti sebelumnya dalam penelitian yang berjudul "Sistem Inormasi Pengolahan Data Nilai Siswa Berbasis Web Pada Sekolah Menengah Atas (SMA) Muhammadiyah Pacitan". Penelitian tersebut bertujuan untuk memperbaiki sistem sebelumnya yang masih manual dengan menulis di *raport* sehingga dikawatirkan bisa menyebabkan *raport* hilang, basah terkena air, dan kelalaian siswa, serta Guru Wali terkadang lupa akan tempat penyimpanan *raport* yang tidak tertata rapi. Peneliti membuat aplikasi berbasis *web* dengan *database* MySQL (Susy, 2013).

Yoni dan Berliana (2013), pada jurnalnya yang berjudul "Rancang Bangun Sistem Informasi Nilai Akademik Dan Presensi Siswa Berbasis SMS Gateway Pada

SDN Tulakan III", merancang sistem informasi nilai akademik berbasis SMS (*Short Message Service*). Peneliti merancang sistem penilaian dengan SMS karena SDN Tulakan III belum menerapkan sistem informasi. Dengan sistem tersebut diharapkan dapat mempermudah penyampaian informasi akademik khususnya nilai dan presensi siswa untuk membantu pemantauan para orang tua siswa dan menghindarkan penyampaian nilai akademik putera/puterinya di akhir semester.

Menurut analisis yang telah dilakukan oleh peneliti sebelumnya, aplikasi yang dibuat hanya berupa peng*input*an data nilai saja. Berdasarkan kekurangan dari penelitian sebelumnya maka penulis dalam skripsi ini menambahkan fitur-fitur sebagai berikut:

1. Aplikasi dibuat menggunakan Java *Desktop* karena aplikasi *offline* dan digunakan hanya untuk satu *user* yaitu *Admin*.
2. SQL Server sebagai penyimpanan data (*database)*.
3. Mengelola data mahasiswa, dosen, mata kuliah, dan nilai.
4. Fitur ekspor data untuk mengubah data menjadi tipe .csv dari *Microsoft Excel*.
5. Fitur impor data untuk mengambil data dari *Excel* dan disimpan ke *database*.
6. Fitur statistik nilai mahasiswa berdasarkan mata kuliah, semester, dan tahun ajaran.
7. Fitur *upload* gambar yang telah di *scan* sebagai bukti laporan.

2.2 Landasan Teori

2.2.1 Sistem Informasi Rekapitulasi Nilai

Menurut Sutarman (2009), Sistem informasi ini mengumpulkan, memproses, menyimpan, menganalisis, menyebarkan informasi untuk tujuan tertentu. Sehingga, sistem informasi dapat dikatakan sebagai sebuah kegiatan pengolahan data yang dimulai dari mengumpulkan, memproses, menganalisis, menyimpan, dan menyebarkan suatu informasi demi untuk kemajuan atau kepentingan suatu organisasi.

Rekapitulasi nilai merupakan suatu kegiatan meringkaskan data sehingga menjadi lebih berguna bentuk, susunan, sifat atau isinya dengan bantuan tenaga tangan atau suatu peralatan dan mengikuti rangkaian langkah, rumus, atau pola tertentu (Mintorogo dan Sedarmayanti, 1992).

2.2.2 Pengertian Aplikasi *Desktop*

Aplikasi *desktop* merupakan suatu aplikasi yang independen atau berjalan sendiri dalam suatu komputer tanpa menggunakan *browser* atau tanpa terkoneksi Internet (Konixbam, 2009).

Aplikasi *desktop* memiliki beberapa kelebihan, namun juga memiliki beberapa kelemahan. Berikut merupakan beberapa kelebihan dari aplikasi *desktop*:

1. *Desktop* merupakan aplikasi yang independen, dalam menjalankan aplikasi tidak perlu terkoneksi ke jaringan atau internet.
2. Tanpa adanya koneksi internet, penggunaan aplikasi tersebut digunakan setelah berhasil dalam menginstal. Sehingga file-file yang diperlukan untuk menjalankan aplikasi sudah terinstal sebelumnya.
3. Prosesnya cepat. Karena aplikasi *desktop* tidak menggunakan koneksi internet, sehingga dalam menggunakan aplikasi tersebut terhindar dari masalah *erorr connection*.

Dari beberapa kelebihan diatas, aplikasi *desktop* juga memiliki kelemahan. Berikut merupakan beberapa kelemahannya :

1. Telah di sebutkan pada kelebihan diatas, bahwa aplikasi *desktop* harus melewati proses penginstalan terlebih dahulu untuk dijalankan.
2. Jika komputer yang akan dipasang belum terinstal aplikasi, maka tidak dapat menggunakan atau tidak dapat menjalankan aplikasi tersebut.Perlunya *hardware* dengan spesifikasi tinggi,
3. Dalam menjalankan aplikasi yang independen perlu *hardware* dengan spesifikasi tinggi. Hal ini dapat mengatasi masalah seperti *loading* lama yang terjadi pada komputer yang digunakan. (Adi Wicaksana, 2014).

2.2.3 Bahasa Pemrograman Java

Bahasa Java merupakan salah satu bahasa pemrograman yang digunakan untuk mengembangkan aplikasi *desktop*. Bahasa Java dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa pemrograman Java secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi *web* (Yuhendra, 2013).

Menurut Ibnu Rachman Chalid (2009), Java memiliki beberapa keunggulan bila dibandingkan dengan bahasa pemrograman lainnya. Diantaranya:

1. Java bersifat lebih sederhana dan relatif mudah. Java dimodelkan sebagian dari bahasa C++, namun dengan memperbaiki beberapa karakteristik C++, seperti mengurangi kompleksitas beberapa fitur, penambahan fungsionalitas, dan penghilangan beberapa aspek pemicu ketidakstabilan sistem pada C++.
2. Java termasuk bahasa pemrograman berorientasi objek (OOP). Artinya, suatu konsep pemrograman yang memecahkan masalah dengan cara memilah program menjadi objek yang saling berinteraksi satu sama lain.
3. Java bersifat *multiplatform*. Artinya, dapat diterjemahkan oleh Java *interpreter* pada berbagai sistem operasi.
4. Java bersifat *multithread*. Artinya, Java dapat mengerjakan beberapa proses dalam waktu yang hampir bersamaan.

2.2.4 NetBeans IDE

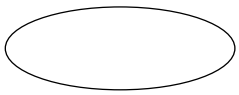
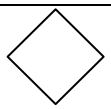
Netbeans merupakan proyek *open source* yang disponsori oleh Sun Microsystem. Ada 2 produk dari Netbeans, yaitu Netbeans IDE dan Netbeans Platform. Netbeans IDE merupakan produk yang digunakan untuk melakukan pemrograman baik menulis kode, meng-*compile*, mencari kesalahan dan mendistribusikan program. Sedangkan Netbeans Platform merupakan sebuah modul yang merupakan kerangka awal dalam membangun aplikasi *desktop* yang besar (Nita Yuli, 2017).

2.2.5 *Entity Relationship* Diagram (ERD)

Entity Relationship Diagram merupakan diagram yang menggambarkan hubungan antar entitas dimana setiap entitas terdiri dari beberapa atribut yang mempresentasikan seluruh kondisi atau fakta dari dunia nyata yang ditinjau. ER-Diagram mentransformasikan keadaan dari dunia nyata ke dalam bentuk basis data (Edi Winarko, 2006).

Dalam ER-D terdapat beberapa komponen terkait. Komponen-komponen pada ER-D dapat dilihat pada Tabel 2.1 dibawah ini.

Tabel 2. 1. Simbol-Simbol ER-D

No	Gambar	Nama Gambar	Keterangan
1.		*Entity*	Individu yang mewakili suatu objek dan dapat dibedakan dengan objek yang lain.
2.		*Attribute*	Atribut yang dimiliki oleh entitas, dimana dapat mendeskripsikan karakteristik dari entitas tersebut.
3.		*Relationship*	Menunjukkan hubungan antara sejumlah entitas yang berbeda.

2.2.6 SQL Server

SQL Server merupakan *Relational Database Management System* (RDBMS) yang dikembangkan oleh Microsoft. Sebagai *software*, fungsi utama SQL Server adalah menampung dan menggunakan data yang terintegrasi dengan aplikasi baik pada komputer yang sama atau dari komputer lain melalui jaringan (Jubilee Enterprise, 2015).

Microsoft SQL Server Management Studio adalah sebuah aplikasi sistem manajemen basis data relasional (RDBMS) produk Microsoft.

Berikut beberapa fitur dari SQL Server :

1. Microsoft SQL Server mendukung ODBC (*Open Database Connectivity*).
2. Mempunyai *driver* JDBC untuk bahasa pemrograman Java.
3. Dapat membuat basis data *mirroring* dan *clustering*.
4. Bahasa *query* utamanya adalah Transact-SQL yang merupakan implementasi dari SQL standar ANSI/ISO dan digunakan oleh Microsoft dan Sybase.

Microsoft SQL Server termasuk DBMS profesional. Beberapa pesaing seperti MySQL, Oracle, telah mengembangkan software serupa dalam beberapa tahun terakhir, tetapi Microsoft SQL Server lebih mudah digunakan dan memiliki lebih banyak fitur. Pemicunya adalah dukungan penuh dari Microsoft. Software yang ditawarkan oleh Microsoft menawarkan integrasi yang erat dengan .NET framework, dan ini tidak dimiliki oleh produk lain (Aiska, 2011).

2.2.7 *System Development Life Cycle* (SDLC)

Systems Development Life Cycle (SDLC) adalah proses dalam membangun sebuah Sistem Informasi dengan melalui beberapa fase. SDLC terdiri dari 6 fase, yaitu *Planning*, *Analysis*, *Design*, *Implementation*, *Testing* dan *Maintenance*.

1. *Planning*

Fase *planning* merupakan fase awal dalam pembuatan sistem informasi yang mendefinisikan kebutuhan-kebutuhan sumber daya. Dalam fase ini dilakukan langkah-langkah sebagai berikut:

a. Mendefinisikan masalah dan menentukan tujuan sistem
b. Mengidentifikasi berbagai kendala sistem dan membuat studi kelayakan

2. *Analysis*

Fase *analysis* merupakan fase penelitian pada sistem yang berjalan dengan tujuan untuk merencanakan sistem yang baru menggunakan tools atau UML (*Unified modeling Language*) dengan *software* Visio 2013.

3. *Design*

Fase *design* merupakan proses perancangan sistem yang akan dibangun. Menentukan bagaimana sistem akan beroperasi berdasarkan hasil analisa sebelumnya, dengan menggunakan *hardware*, *software*, infrastruktur jaringan, program, *database*, form dan laporan, serta file yang dibutuhkan.

4. *Implementation*

Fase *implementation* merupakan fase mengimplementasikan *design* sistem pada fase-fase sebelumnya. Di dalam implementasi terdapat beberapa aktivitas yakni:

a. Pembuatan *database* sesuai *scema* rancangan
b. Proses pembuatan aplikasi berdasarkan *design* sistem
c. Proses *debugging* atau pengujian dan perbaikan suatu aplikasi

5. *Testing*

Tahap ini *software* hasil produksi harus diuji coba, termasuk semua fungsi-fungsinya, agar *software* bebas dari *error* dan hasilnya harus sesuai dengan kebutuhan.

6. *Maintenance*

Proses ini merupakan tahap menerapkan dan pemeliharaan *software*.

2.2.8 Metode *Black Box Testing*

Pengujian atau biasa disebut *testing* terdapat 2 metode, yaitu metode *white box testing* dan metode *black box testing*. Pada skripsi ini dipilih menggunakan *black box testing* karena metode tersebut menguji perangkat lunak dengan mencari kesalahan fungsi-fungsi dalam aplikasi. Sehingga, metode *black box testing* lebih sesuai dibanding dengan *white box testing*. Hal ini bertujuan untuk mengetahui apakah fungsi-fungsi dalam perangkat lunak sudah sesuai dengan yang diharapkan atau belum, bukan untuk memperlihatkan cara kerja dari produk.

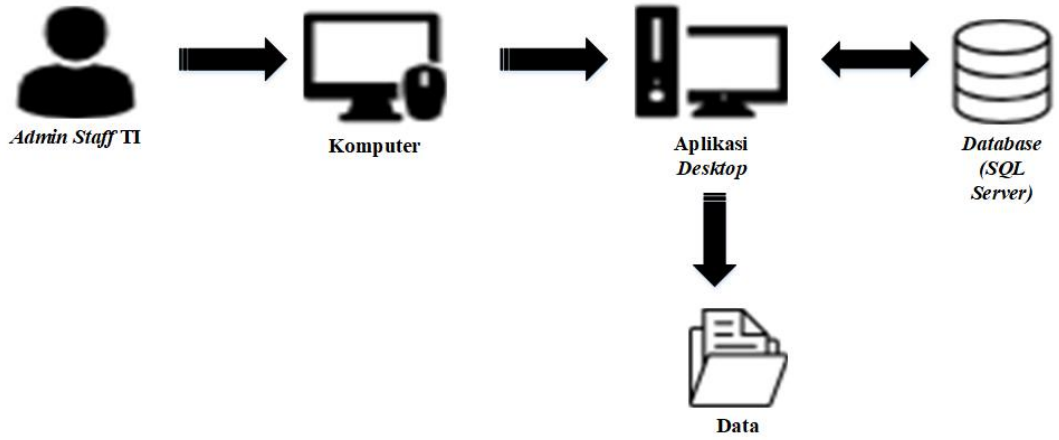
Black Box Testing berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineer* untuk memperoleh *input* yang sepenuhnya akan

melaksanakan persyaratan fungsional untuk sebuah program. *Black Box Testing* dapat menemukan kesalahan dalam kategori berikut (Roger S. Pressman, 2010):

1. Jika ada fungsi yang tidak benar atau hilang
2. Kesalahan pada tampilan
3. Sensitifnya sistem terhadap nilai *input* tertentu
4. Kesalahan dalam struktur data atau akses *database* eksternal
5. Validitas fungsional
6. Batasan dari suatu data

2.2.9 Arsitektur Perangkat Lunak

Dalam membuat aplikasi diperlukan perancangan arsitektur perangkat lunak. Fungsinya untuk menggambarkan bagaimana system dibuat dan dijalankan. Arsitektur perangkat lunak pada aplikasi "Rekapitulasi Nilai Mahasiswa" dapat dilihat pada Gambar 2.1.



Gambar 2. 1. Arsitektur Perangkat Lunak

2.2.10 *Unified Markup Language* (UML)

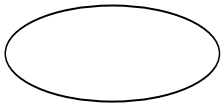
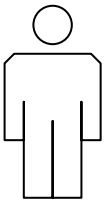
Unified Markup Language (UML) merupakan bahasa pemodelan yang digunakan untuk menspesifikasikan, memvisualisasikan, membangun, dan mendokumentasikan seluruh rancangan aplikasi perangkat lunak. Penggunaan dengan model UML berfungsi untuk mengidentifikasi bagian-bagian yang termasuk dalam lingkup sistem di dalam aplikasi. Pemodelan UML yang digunakan

dalam pembuatan aplikasi “Rekapitulasi Nilai Mahasiswa” antara lain adalah *Use Case* Diagram, *Class* Diagram, dan *Activity* Diagram.

a. *Use Case* Diagram

Diagram yang mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem yang dibuat. *Use Case* juga dapat mengetahui fungsi yang ada di dalam sistem dan siapa saja yang berhak menggunakan fungsi-fungsi yang dibutuhkan. Pada tabel 2.2 akan dijelaskan simbol-simbol yang ada pada *Use Case* Diagram.

Tabel 2. 2. Simbol-Simbol *Use Case* Diagram

No	Gambar	Nama Gambar	Keterangan
1.		*Use Case*	Merupakan fungsionalitas yang disediakan sistem sebagai unit yang bertukar pesan dengan aktor.
2.		Aktor	Merupakan penggambaran abstrak dari orang yang mengaktifkan fungsi dari target sistem dan merupakan orang yang berinteraksi dengan *use case*.
3.		*Association*	Digambarkan dengan garis tanpa panah mengindikasikan siapa yang beriinteraksi secara langsung dengan sistem.
4.		*Include*	Mengidentifikasi hubungan antar dua *use case* dimana satu *usecase* memanggil *usecase* yang lain.
5.		*Extend*	Merupakan perluasan dari *use case* jika kondisi atau syarat terpenuhi.

b. *Class* Diagram

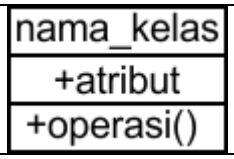
Diagram yang menggambarkan struktur dan penjelasan *class*, paket, dan objek serta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain. *Class* Diagram juga sebagai penjelasan hubungan antar *class* dalam sebuah sistem yang dibuat.

Pada *class* diagram terdapat 3 komponen penting yaitu:

a. Nama, merupakan nama dari sebuah *class*
b. Atribut, merupakan property dari sebuah *class*. Atribut dibagi menjadi 4 macam, yaitu *private*, *protected*, *public*, dan *package*
c. Operasi, merupakan sesuatu yang bisa dilakukan oleh sebuah *class* terhadap *class* lain

Pada Tabel 2.3 dapat dilihat simbol-simbol *class* diagram yang digunakan dalam aplikasi "Rekapitulasi Nilai Mahasiswa".

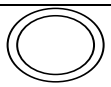
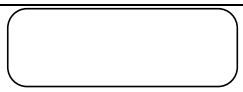
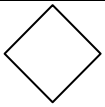
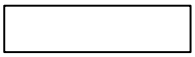
Tabel 2. 3. Simbol-Simbol *Class* Diagram

No	Gambar	Nama Gambar	Keterangan
1.		*Class*	Merupakan simbol dalam sebuah *class* dimana dibagi menjadi 3 yaitu nama kelas, atribut, dan operasi
2.		*Association*	Garis yang menghubungkan *class* satu dengan *class* lainnya dengan makna umum
3.		*Aggregation*	Menghubungkan antar *class* dengan makna untuk semua bagian
4.		*Dependency*	Menunjukkan operasi pada suatu *class* yang menggunakan *class* lain

c. *Activity* Diagram

Diagram yang digunakan untuk menggambarkan aliran kerja atau *workflow* atau aktivitas sebuah sistem aplikasi. Pada tabel 2.4 dapat dilihat simbol-simbol *activity* diagram yang digunakan dalam aplikasi "Rekapitulasi Nilai Mahasiswa".

Tabel 2. 4. Simbol-Simbol *Activity* Diagram

No	Gambar	Nama Gambar	Keterangan
1.		*End Point*	Merupakan akhir dalam aktifitas.
2.		*Activities*	Menggambarkan suatu proses atau kegiatan bisnis
3.		*Decision Point*	Menggambarkan pilihan untuk pengambilan keputusan dalam aktifitas.
4.		*Fork Node*	Menggambarkan awalan dari percabangan aktifitas.