BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian-penelitian terdahulu yang menjadi bahan acuan terhadap penelitian yang di lakukan saat ini yaitu.

1. (Ryandika, 2012) dengan judul "Sistem Aplikasi Penggajian Guru Dan Karyawan Yayasan Al Irsyad Al Islamiyah Surakarta". Melakukan penelitian dengan tujuan untuk mengembangkan sistem penggajian yang di gunakan untuk pengolahan data gaji karyawan dengan memanfaatkan *NetBeans* sebagai pembuatan sistemnya dan *MySQL* sebagai pengolah basis data.
2. (Dewi, Ningrum, & Sari, 2016) dengan judul "Perancangan Aplikasi Kenaikan Pangkat dan Gaji Berkala (Studi kasus: Dinsosnakertrans Kota Salatiga)". Melakukan penelitian dengan tujuan untuk mempermudah pencarian data pegawai, percetakan pengajuan gaji berkala, percetakan penjagaan kenaikan gaji berkala, dan percetakan penjagaan kenaikan pangkat, serta memberi peringatan *alert* kepada pengguna untuk mempersiapkan berkas-berkas kenaikan pangkat dan kenaikan gaji berkala dalam kurun waktu tiga bulan dengan membuat aplikasi menggunakan *Java* dan *MySQL*.
3. (Ryan, Ruliah, & Arnie, 2017) dengan judul "Aplikasi Pendataan Dan Pengingat Kenaikan Gaji Serta Kenaikan Golongan Berbasis *SMS Gateway*". Melakuan penelitian dengan tujuan untuk penginputan, percetakan laporan, *reminder* kenaikan gaji dan kenaikan golongan pegawai dan pengolahan data karyawan dengan membuat aplikasi menggunakan sistem *SMS Gateway* dan *MySQL*.

Setelah membandingkan penelitian-penelitian terdahulu dapat disimpulkan bahwa persamaanya adalah sama-sama membuat apalikasi pada subbagian penggajian sebagai solusi mempermudah dan mempercepat kinerja karyawan dalam bekerja.

Sementara itu, perbedaan penelitian ini dengan penelitian-penelitian yang di paparkan di atas terletak pada jenis aplikasi yang di buat. Ketiga penelitian di atas membangun aplikasi desktop. Sedangkan penelitian yang penulis kembangkan

adalah sebuah aplikasi berbasis web yang dapat di akses secara *online* sehingga Admin tidak perlu memasang aplikasi terlebih dahulu.

2.2 Landasan Teori

2.2.1. Aplikasi Web

Menurut (Remick, 2011) Aplikasi web merupakan sebuah aplikasi yang menggunakan teknologi *browser* untuk menjalankan aplikasi dan di akses melalui jaringan komputer. Sedangkan menurut (Rouse, 2011) aplikasi web adalah sebuah program yang di simpan di *server* dan dikirim melalui *internet* dan di akses melalui antarmuka *browser*. Dari pengertian di atas dapat di simpulkan aplikasi web merupakan aplikasi yang di akses menggunakan *web browser* juga merupakan suatu perangkat lunak komputer yang di kodekan dalam bahasa pemrograman yang mendukung perangkat lunak berbasis web seperti *HTML, JavaScript, CSS, Ruby, Python, Php, Java* dan bahasa pemrograman lainnya.

2.2.2 Pemrograman Java

Menurut (Rosa & Shalahuddin, 2014) *Java* menurut definisi dari sun adalah sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada *computer stand alone* ataupun pada lingkungan jaringan. *Java* 2 adalah generasi kedua dari *java platform.* Kata berdiri di atas sebuah mesin *interpreter* yang di beri nama *Java Virtual Machine (JVM). Java Virtual Machine (JVM)* inilah yang akan membaca *bytecode* dalam *file.class* dari suatu program sebagai representasi langsung dari program yang berisi bahasa mesin. Oleh karena itu, bahasa *Java* disebut sebagai bahasa pemrograman yang *portable* karena dapat di jalankan pada berbagai sistem operasi, asalkan sistem operasi tersebut terdapat *Java Virtual Machine (JVM).*

Agar sebuah program *Java* dapat dijalankan, maka file dengan *ekstensi, Java* harus dikompilasi menjadi *file bytecode*. Untuk menjalankan *bytecode* tersebut di butuhkan *Java Runtime Environment (JRE)* yang memungkinkan pemakai untuk menjalankan program *Java*. *Java Runtime Environment (JRE)* berisi *Java Virtual Machine (JVM)* dan *library Java* yang di gunakan. *Java* memiliki beberapa versi *library* atau teknologi yang di sebut juga sebagai edisi dari bahasa pemrograman *Java*. Tiga edisi utama dari *library* tersebut adalah *Micro, Standard,* dan *Enterprise*.

2.2.3 *HTML (Hyper Text Markup Language)*

Menurut (Prasetio, 2012), *HTML (Hyper Text Markup Language)* yang didefinisikan sebagai sebuah *file teks* yang berisi *tag-tag markup*. *Tag markup* berfungsi untuk memberitahukan *browser* bagaimana harus menampilkan sebuah halaman. Pada *file HTML* harus memiliki *ekstensi HTM* atau *HTML* dan dapat dibuat menggunakan *editor teks* yang biasa dipakai. Menurut James Sugrue, *HTML5* adalah standar untuk penataan dan penyajian konten di web. Ini menggabungkan fitur seperti *Geolocation*, pemutaran *video* dan *drag-and-drop*. *HTML5* memungkinkan pengembang untuk membuat aplikasi internet yang kaya tanpa membutuhkan *API* pihak ketiga dan *browser plug-in*.

2.2.4 *JSP (Java Server Page)*

Menurut (Foenadioen, 2007) *JSP (Java Server Pages)* adalah suatu bahasa pemrograman web *(scripting)* yang bersifat *server side* yang menggabungkan *HTML* dengan *scripting* tag dan program *java*. Suatu dokumen *JSP* ditandai dengan berkas ekstensi. *JSP Java Server Pages (JSP)* merupakan sebuah teknologi *servlet-based* yang digunakan pada web *tier* untuk menghadirkan dinamik dan statik *conten*t. *Java Server Pages (JSP)* merupakan *text-based* dan kebanyakan berisi *template text HTML* yang digabungkan dengan spesifik *tags dynamic content"*.

2.2.5 NetBeans IDE

Menurut (Nishom, 2012), *Netbeans* merupakan sebuah aplikasi *Integrated Development Environment (IDE)* yang berbasiskan *Java* dari *Sun Microsystems* yang berjalan di atas *swing*. *Swing* merupakan sebuah teknologi *Java* untuk pengembangan aplikasi *desktop* yang dapat berjalan pada berbagai macam *platform seperti windows, linux, Mac OS X* dan *Solaris.*

Sebuah *IDE* merupakan lingkup pemrograman yang di integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan *Graphic User Interface (GUI),* suatu kode editor atau *text,* suatu *compiler* dan suatu *debugger*. *Netbeans* juga digunakan oleh sang programmer untuk menulis, meng-*compile*, mencari kesalahan dan menyebarkan program *NetBeans* yang ditulis dalam bahasa pemrograman *java,* namun selain itu dapat juga mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat

Professional Desktop, Enterprise, web*, and Mobile Applications* dengan *Java Language, C/C++,* dan bahkan *Dynamic Languages* seperti *PHP*, *JavaScript, Groovy*, dan *Ruby*.

2.2.6 SQL Server

Menurut (Djunadi, 2002) dalam bukunya yang berjudul *SQL Server* untuk Profesional, mendefinisikan: *SQL Server* adalah sebuah sistem arsitektur terbuka yang memungkinkan para pengembang program memperluas dan menambahkan fungsi-fungsi ke dalam *data base* tersebut. Menurut Andri Kuniyo dan kusrini (2007) dalam bukunya yang berjudul Membangun Sistem Informasi Akuntasnsi dengan *Visual Basic* dan *SQL Server* mendefinisikan: *SQL Server* adalah perangkat lunak *relation data base management system (RDBMS)* yang di desain untuk melakuakan proses manipulasi *data base* berukuran besar dengan berbagai fasilitas.

2.2.7 Microsoft Visio

Menurut (Wijaya, 2010) Microsoft Office Visio adalah program aplikasi dari Microsoft corp. Untuk membuat berbagai diagram seperti alur, diagram proses bisnis, blok diagram, dan diagram aliran data dengan cepat dan mudah. Bentuk-bentuk Diagram pada Visio:

1. Diagram jaringan *(Network Diagram)*

   Membuat susunan bentuk jaringan computer atau lainua sesuai kebutuhan hanya dengan menggabungakan bentuk kombinasi bentuk dari Visio yang sudah tersedia.

2. *Flowchart* dasar *(Basic Flowchart)*

   Membuat dokumen prosedur, menganalisa proses, menunjukan alur kerja atau informasi, lagu dan efisiensi biaya, dan banyak lagi

3. Rencana denah (*Floor Plan*)

   Membuat denah rencana peletakan pintu, jendela, alat-alat listrik secara visual untuk bangunan.

4. Bagan struktur organisasi (*Organization Chart*)

   Membentuk bagan kelompok kerja dalam organisasi dan hubungan komunikasi antar departemen.

5. Diagram basis data (*Data Base Model Diagram*)

Membentuk model dari data base secara visual dengan penggambaran bentuk skema sesuai aslinya.

6. Diagram situs jaringan (*Web Site Diagram*)
   Menggambar bentuk susunan website seccara hierarki dan alur penggunaanya yang dapat dengan mudah di ubah sesuai dengan kebutuhan web saat ini yang dinamis.
7. Diagram blok
   Melakukan Brainstorming, rencana, dan berkomunikasi.
8. Peta petunjuk
   Peta petunjuk yang dapat menunjukan arah dengan disertai petunjuk alam seperti pohon, bangunan, sungai dan jalan raya sebagai petunjuknya.
9. Diagram proses mesin
   Menunjukan bagaimana proses dari suatu alat-alat dalam perindustrian serta alat-alat yang di gunakan seperti mesin, pipa, dan penampungnya.
10. Diagram *software*
    Membantu mengembangkan tim desain perangkat lunak untuk mengatur tampilan *user interface.*

2.2.8 *Balsamiq Mockup*

Menurut (Hanifah, 2015) *Balsamiq Mockup* adalah salah satu *software* yang di gunakan dalam pembuatan desain atau *prototype* dalam pembuatan tampilan user *interface* sebuah aplikasi.

2.2.9 *Unified Modeling Languange* (UML)

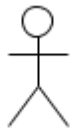
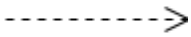
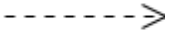
Menurut (Whitten & Bentley, 2007), UML (*Unified Modeling Language*) adalah seperangkat konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak dalam hal objek. Model UML yang dipakai dalam pengembangan aplikasi antara lain adalah Use Case Diagram, Activity Diagram dan Class Diagram.

2.2.10 *Use Case Diagram*

Menurut (Whitten & Bentley, 2007), *use case* diagram adalah sebuah diagram yang menggambarkan interaksi antara sistem dan sistem eksternal dan pengguna. dengan kata lain, grafis yang menggambarkan siapa yang akan menggunakan

sistem dan dalam cara apa pengguna mengharapkan untuk berinteraksi dengan sistem. Simbol-simbol yang digunakan *use case* diagram dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Simbol-simbol *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 *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.

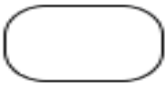
No	Gambar	Nama	Keterangan
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

2.2.11 *Activity Diagram*

Menurut (Whitten & Bentley, 2007), *activity* diagram adalah diagram yang dapat digunakan untuk grafis menggambarkan aliran proses bisnis, langkah-langkah dari kasus penggunaan, atau logika perilaku objek. Simbol-simbol langkah

dari kasus penggunaan, atau logika perilaku objek. Simbol-siml yang digunakan *activity* diagram dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Simbol-simbol *Activity Diagram*

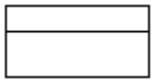
No	Gambar	Nama	Keterangan
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		*Initial Node*	Node Awal untuk memulai sebuah proses activity.
4		*Actifity Final Node*	Node akhir untuk mengakhiri sebuah proses activiti.
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.

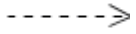
2.2.12 *Class Diagram*

Menurut (Whitten & Bentley, 2007), class diagram adalah suatu gambaran grafis dari struktur objek statis suatu sistem yang menunjukkan kelas objek bahwa

sistem terdiri dari serta hubungan antara kelas objek. Simbol-simbol yang digunakan *class* diagram dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Simbol-simbol *Class Diagram*

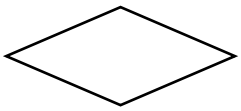
No	Gambar	Nama	Keterangan
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.

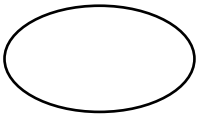
No	Gambar	Nama	Keterangan
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.

2.2.13 *Entyty Relationship Diagram* (ERD)

Menurut (Suntana, 2004) *Entity Relationship Diagram* adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan *relationship* data. Simbol-simbol yang di gunakan *Entity Relationship Diagram* dapat dilihat pada Tabel 2.4

Tabel 2. 4 Simbol-simbol *Entyty Relationship Diagram Diagram*

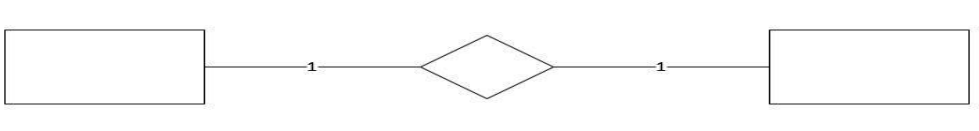
Simbol	Nama	Keterangan
	Entitas	Digambarkan dalam bentuk persegi panjang. Entitas adalah sesuatu apa saja yang ada dalam sistem, nyata maupun abstrak dimana data disimpan atau dimana terdapat data.
	Relasi	Relasi adalah hubungan ilmiah yang terjadi antara entitas.

Simbol	Nama	Keterangan
	Atribut	Adalah sifat atau karakteristik dari tiap-tiap entitas dan relasi atau elemen data dari entitas dan relasi. Atribut ini digunakan untuk penanaman dari bagian-bagian yang terdapat dalam entitas.
	Garis lurus	Menghubungkan antara entitas satu dengan entitas lainya.

Hubungan Antar Entitas Pada ERD

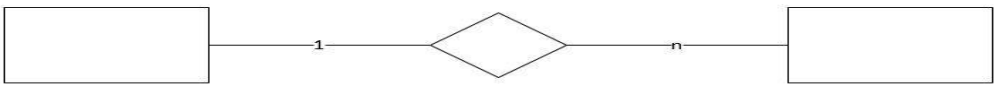
1. *One to one relationship* (relasi satu ke satu)

Dalam relasi dari satu ke satu, setiap record dalam tabel A hanya dapat memiliki satu record yang bersesuaian dalam tabel B, dan sebaliknya. Jenis relasi ini tidak umum karena sebenarnya tabel A dan tabel B dapat digabungkan menjadi satu tabel. Relasi ini dapat di gunakan untuk membagi tabel yang memiliki field yang banyak, untuk mengisolasi sebagian tabel dengan alasan keamanan data.



2. *One to many relationship* (relasi satu ke banyak)

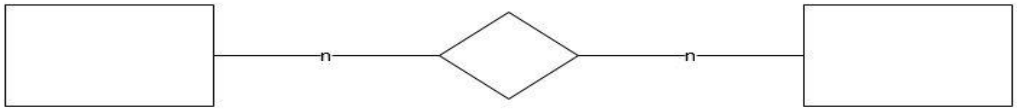
Relasi satu ke banyak adalah jenis relasi yang paling umum. Dalam relasi satu ke banyak, sebuah record dalam tabel A dapat memiliki banyak record bersesuaian dalam tabel B. Tetapi sebuah record dalam tabel B hanya memiliki sebuah record yang bersesuaian dalam tabel.



3. Many to many relationship (relasi banyak ke banyak)

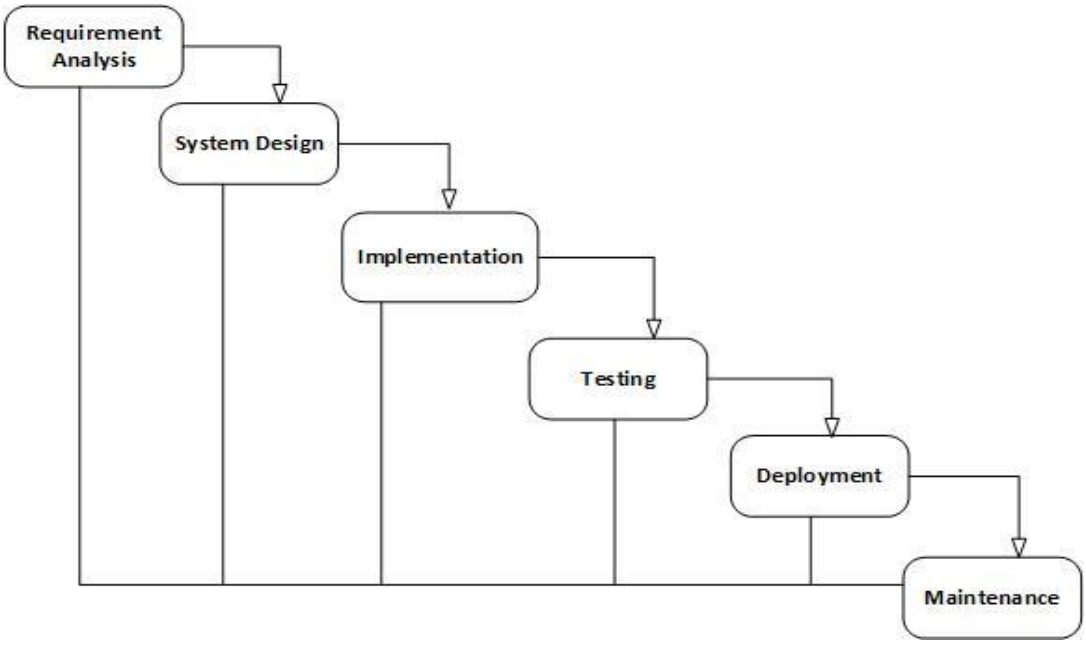
Dalam relasi banyak ke banyak, sebuah record dalam tabel A dapat memiliki banyak record yang bersesuaian dalam tabel B, dan sebuah record dalam

tabel B dapat memiliki banyak record yang bersesuaian dalam tabel A. Jenis relasi ini hanya di mungkinkan jika kita mendefinisikan tabel baru sebagai perantara. Relasi banyak ke banyak merupakan dua buah relasi satu ke banyak terhadap tabel perantaranya.



2.2.13 Prosedur Penelitian

Menurut (Sommervile, 2011), tahapan utama dari *waterfall model* langsung mencerminkan aktifitas pengembangan dasar. Terdapat 5 tahapan pada *waterfall model,* yaitu *requirement analysis and definition, system and software design, implementation and unit testing, integration and system testing,* dan *operation and maintence*. Proses metode *waterfall* dapat dilihat pada gambar 2.1.



Gambar 2. 1 Metode *Waterfall*

Tahap-tahap model *waterfall* di gunakan penulis sebagai acuan prosedur penelitian dengan rincian sebagai berikut:

1) Tahap *Requirement Analysis*

   Tahap *Requirement Analysis* merupakan tahap yang di lakukan untuk menganalisa kebutuhan-kebutuhan yang di perlukan sistem. Analisis kebutuhan

yang di perlukan sistem meliputi kebutuhan seperti data-data Pegawai, fitur apa saja yang di perlukan, bagaimana proses sistem berjalan, dan lainya.

2) Tahap *System and Software design*

Tahap *System design* bertujuan untuk menggambarkan bagaimana suatu sistem. *System Design* menjelaskan bentuk atau tampilan dari sistem yang di rancang dan membantu dalam menjelaskan arsitektur dari sistem

3) Tahap *Implementation*

Tahap Implementasi *(Implementation)* merupakan proses pembuatan sistem dan nantinya saling berintegrasi dengan tahap selanjutnya. Proses *Implementation* di

buat berdasarkan hasil dari tahap *Requirement Analysis* dan *System Design.*

4) Tahap *Integration and system Testing*

Tahap *Testing* atau *Integration* merupakan tahap yang di lakukan untuk uji coba terhadap tahap *implementation* yang telah di lakukan. *Testing* atau *Integration* bertujun untuk mengetahui kualitas sistem dan mengevaluasi apakah sistem siap atau tidak untuk di gunakan divisi penggajian pegawai.

5) Tahap *Deployment*

Tahap *Deployment* merupakan tahap yang di lakukan stelah proses *testing* atau *integration.* Setelah *functional* dan *nonfunctional testing* telah selesai di lakukan, *deployment* atau persiapan sistem untuk di gunakan oleh devisi penggajian pegawai.

6) Tahap *Maintenance*

Tahap *Maintenance* merupakan proses perawatan sistem stelah di pakai atau di gunakan. Dalam penelitian ini, penulis membatasi pada tahap ini, karena tahap ini berada di luar kewenangan penulis.

2.2.14 *Black Box Testing*

Menurut (Pressman, 2010) *"black box testing* atau juga di sebut *behavior* Testing, berfokus pada persyaratan fungsional dari perangkat lunak". Artinya, teknik black box testing memungkinkan untuk mendapatkan set kondisi masukan yang sepenuhnya akan melaksanakan semua persyaratan fungsional untuk satu program.