BAB II
KAJIAN PUSTAKA

A. Kajian Teori

1. Sistem Informasi

Menurut Murdick (1991: 16), sistem informasi adalah suatu kelompok orang, seperangkat pedoman, dan petunjuk peralatan pengolahan data (seperangkat elemen), memilih, menyimpan, mengolah dan mengambil kembali data (mengoperasikan data dan barang) untuk mengurangi ketidakpastian pada pengambilan keputusan (mencari tujuan bersama) dengan menghasilkan informasi untuk manajer pada waktu mereka dapat menggunakannya dengan paling efisien. Sementara menurut Alter (1992), sistem informasi adalah kombinasi antar prosedur kerja, informasi, orang, dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi. Dari beberapa definisi tersebut dapat disimpulkan bahwa sistem informasi adalah suatu kombinasi yang terdiri dari informasi, manusia, seperangkat prosedur kerja, dan teknologi informasi yang diorganisasikan untuk mencapai tujuan tertentu. Abdillah (2003: 138), mengungkapkan bahwa komponen-komponen utama dalam suatu sistem informasi berbasiskan komputer terdiri dari: 1) *Database*, 2) *Database software*, 3) Aplikasi *software*, 4) *Hardware* komputer termasuk media penyimpanan, dan 5) Personal yang menggunakan dan mengembangkan *system.*

2. Pengembangan Perangkat Lunak

Pengembangan sistem menurut Jogiyanto (2005: 35), dapat berarti menyusun suatu system yang baru untuk menggantikan sistem yang lama secara keseluruhan atau memperbaiki sistem yang telah ada.

"Software engineering is the application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software". (IEEE *Standard Glossary of Software Engineering Terminology*, 1990:67)

"Software engineering is also concerned with theories, methods and tools that are needed to develop the software products in a cost effective way." (Somerville, 2011: 8).

Berdasarkan beberapa pendapat di atas, dapat disimpulkan bahwa pengembangan perangkat lunak adalah bagian dari rekayasa perangkat lunak. Di mana rekayasa perangkat lunak adalah pendekatan sistematik yang berhubungan dengan suatu teori, metode, dan alat untuk mengembangkan suatu perangkat lunak yang berupa pembuatan sistem baru, penggantian sistem yang sudah lama, perawatan, dan perbaikan sistem yang sudah ada.

Dalam pengembangan perangkat lunak, dibutuhkan suatu metode agar proses pengembangan berjalan sistematis, terkonsep, terstruktur, dan lebih efektif dalam segi waktu maupun biaya.

Menurut Jogiyanto (2005: 40), sebelum proses pengembangan sistem dilakukan, maka harus dibuat terlebih dahulu skedul kerja dan tugas-tugas pekerjaan yang akan dilakukan, sehingga proses pengembangan sistem dapat

dilakukan dan selesai dengan berhasil sesuai degan waktu dan anggaran yang direncanakan. Siklus atau daur hidup pengembangan sistem (*System Development Life Cycle* atau SDLC) umumnya menunjukkan tahapan-tahapan kerja dan tugas-tugas kerja yang harus dilakukan. Ada beberapa macam metode dalam SDLC yang dapat digunakan, namun secara garis besar proses pengembangan perangkat lunak setiap metode rata-rata sama. Menurut Pressman (2001: 41-51), ada beberapa macam model pengembangan perangkat lunak yaitu: Model *Waterfall* atau Sekuensial Linier, Model *Prototype*, Model RAD, Model Proses Perangkat Lunak Evolusioner, dan Model Formal.

Dalam penelitian ini, model pengembangan yang digunakan adalah *Waterfall Model*. Model *Waterfall* adalah model proses pengembangan perangkat lunak yang paling umum digunakan. (Agarwal, 2009: 27). Model *Waterfall* adalah sebuah pendekatan sistematis dan berurutan untuk pengembangan perangkat lunak yang dimulai dengan spesifikasi kebutuhan dan berlangsung melalui perencanaan, pemodelan, kontruksi, dan penyebaran, yang berujung pada dukungan yang berkelanjutan dari perangkat lunak yang telah selesai. (Pressman, 2010: 39). Pendapat lain menurut Limaye (2009:38), meskipun model *Waterfall* adalah metode yang paling banyak digunakan, tetapi bukan berarti model tersebut selalu layak untuk digunakan. Meskipun begitu model *Waterfall* tetap menjadi sebuah dasar dari pengembangan aktifitas dan model pengembangan perangkat lunak lain dimana model *Waterfall*-lah yang paling bisa merepresentasikan sebuah cara yang logis dalam melakukan suatu pekerjaan atau proses. Berikut adalah tahap-tahap

dalam proses pengembangan perangkat lunak dalam model *Waterfall* menurut Pressman (2001: 30) :

a. Anlaysis (Analisis Kebutuhan)

Proses pengumpulan kebutuhan ini diintensifkan dan difokuskan pada *software.* Untuk memahami sifat dari program yang akan dibuat, analis harus memahami domain informasi dari *software* tersebut, seperti fungsi yang dibutuhkan, kebiasaan, kinerja, dan tampilan. Kebutuhan dari sistem dan *software* perlu didokumentasikan dan ditunjukkan kepada *user* atau pelanggan.

b. Design (Desain)

Desain perangkat lunak sebenarnya adalah proses yang terdiri dari gabungan beberapa langkah yang berfokus pada empat atribut berbeda: data struktur, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pemrograman. Desain harus dapat mengimplementasikan kebutuhan yang telah ditentukan pada tahap sebelumnya. Desain juga harus didokumentasikan yang merupakan bagian dari konfigurasi perangkat lunak. Berikut adalah beberapa hal yang dilakukan dalam proses desain sistem informasi presensi SMK Negeri 2 Sewon berbasis *web* menggunakan *fingerprint*:

1) UML *(Unified Modelling Language)*

"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system." (Booch et al., 1998). Menurut Nugroho (2011: 119) *Unified Modeling Language* (UML) adalah Bahasa untuk menspesifikasi, memvisualisasikan, serta mengonstruksi bangunan dasar sistem perangkat lunak,

termasuk melibatkan pemodelan dan aturan-aturan bisnis. UML digunakan dalam proses desain arsitektur perangkat lunak adalah karena UML menjadi bahasa yang paling umum, paling banyak digunakan, dan dipahami oleh sebagian besar *developer* (Hansen & Thomsen, 2004: 735).

UML merupakan pemodelan visual sistem dalam suatu perangkat lunak. UML memvisualisasikan sistem dari sebuah perangkat lunak dalam bentuk diagram. Berikut beberapa diagram UML yang digunakan menurut Shalahudin & Rosa, (2011: 122-152):

a) *Use Case* Diagram

Use case merupakan teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* mendeskripsikan interaksi antara satu actor atau lebih dengan sistem itu sendiri dan memberikan deskripsi bagaimana sistem itu berjalan.

b) *Class* Diagram

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

- Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

c) *Sequence* Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antara objek. Oleh karena itu untuk menggambar diagram sekuen maka harus

diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

2) MySQL

Basis data adalah kumpulan data, yang dapat digambarkan sebagai aktivitas dari satu atau lebih organisasi yang berelasi. (Kristanto, 2003: 73). Dari pendapat lain menurut Nugroho (2011: 4), basis data adalah koleksi dari data-data yang terorganisasi sedemikian rupa sehingga data mudah disimpan dan dimanipulasi. Berdasarkan beberapa pendapat di atas, dapat disimpulkan bahwa basis data adalah sekumpulan data terorganisasi dan saling berrelasi sehingga data tersebut mudah untuk disimpan dan dimanipulasi.

Nugroho (2005:1) menerangkan bahwa *MySQL* adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat, *multi user*, serta menggunakan perintah standar SQL *(Structures Query Language).*

MySQL adalah salah satu aplikasi atau *software* DBMS (*Database Management System*) atau Sistem Manajemen Basis Data. Yaitu sebuah aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data.

Nugroho (2005: 2) juga mengungkapkan kelebihan *MySQL* dibandingkan dengan *software* DBMS lain, diantaranya adalah:

a. *MySQL* adalah sebuah *Software* database yang *OpenSource*, artinya program ini bebas digunakan oleh siapa saja tanpa harus membayar atau membeli licence atau sertifikat resmi dari pembuatnya.

b. *MySQL* merupakan sebuah *database server*, dengan begitu *database* dapat terhubung dengan media internet sehingga dapat diakses dari jarak jauh.

c. *MySQL* merupakan sebuah database *client*. Selain menjadi *server* yang melayani permintaan, *MySQL* juga dapat melakukan *query* yang mengakses *database* pada *server.*

d. *MySQL* dapat menerima *query* yang tertumpuk dalam suatu permintaan atau yang disebut *Multi-Threading*.

e. *MySQL* sebagai database dapat menyimpan data dengan jumlah dan ukuran yang sangat besar hingga ukuran gigabyte sekalipun.

c. Code (Implementasi)

Dalam tahap ini, desain harus diterjemahkan dalam bentuk yang dapat dibaca oleh mesin yaitu bahasa pemrograman melalui proses *coding*. Tahap ini adalah implementasi dari desain yang telah dibuat.

1) PHP

Ullman (2003) dalam bukunya menjelaskan PHP pada awalnya diciptakan oleh Rasmus Lerdof pada tahun 1994. Pada saat itu PHP kepanjangan dari *"Personal Home Page"* yang sekarang dikenal dengan *"PHP: Hypertext Preprocessor"*. PHP merupakan bahasa pemrograman yang dapat dituliskan di dalam *script* HTML, karena PHP juga bagian dari HTML yang berfungsi untuk membuat *Website* lebih dinamis dan lebih mudah diakses. Menurut Vaswani (2002:1-2), PHP bersifat *open source* dan dapat digunakan oleh banyak system operasi seperti Windows, Macintosh, UNIX, termasuk Linux. Selain itu PHP dapat terhubung dengan

beberapa jenis database salah satunya yang paling sering digunakan dan bersifat *open source* yaitu MySQL.

2) *Framework* CodeIgniter

a) *Framework*

Framework secara sederhana dapat diartikan kumpulan dari fungsi fungsi/prosedur-prosedur dan *class* untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang pemrograman, tanpa harus membuat fungsi *class* dari awal. (Supono & Putratama, 2016: 108). Sedangkan Simarmata (2010: 82) mengungkapkan, *framework* adalah sistem perangkat lunak yang dapat digunakan kembali dengan fungsi umum yang telah diterapkan. *Framework* juga berfungsi seperti cetak biru untuk arsitektur dasar dan arsitektur fungsional untuk *field* aplikasi yang spesifik.

Penggunaan *Framework* dalam proses membangun sebuah aplikasi *website* tentunya mempunyai kelebihan. Daqiqil (2011: 1) menjelaskan beberapa keuntungan dalam penggunaan *framework*, diantaranya adalah:

1) Menghemat Waktu Pengembangan. Dengan adanya *library* dan *helper* yang telah disediakan, pengembang tidak perlu membuatnya lagi dari awal sehingga proses pengerjaan bisa lebih cepat.
2) *Reuse of code.* Dengan menggunakan *framework,* pekerjaan yang sebelumnya mempunyai struktur yang sudah tetap sehingga dapat digunakan kembali pada pekerjaan selanjutnya.
3) Bantuan komunitas. Terdapat forum yang berisi komunitas-komunitas tempat berbagi ilmu dan informasi mengenai *framework.* selain itu juga dapat menjadi

tempat untuk bertanya atau meminta bantuan untuk memacahkan masalah dalam *framework*. sehingga dapat lebih mengasah kemampuan dalam pemrograman *web*.

4) Kumpulan *best practice*. Sebuah *framework* merupakan kumpulan *best practice* yang sudah teruji. Jadi dapat meningkatkan kualitas kode dalam pemograman *web*.

b) *Framework* CodeIgniter

CodeIgniter adalah aplikasi *open-source* berupa *framework* dengan model MVC (*Model, View, Controller*) untuk membangun *website* dinamis dengan menggunakan PHP (Supono & Putratama, 2016: 109). Menurut Septian (2009: 1), CodeIgniter merupakan sekumpulan *tools* yang telah disediakan bagi orang yang ingin membangun aplikasi *web* menggunakan PHP.

Berdasarkan beberapa pendapat di atas, dapat disimpulkan bahwa *Framework* CodeIgniter adalah sekumpulan fungsi atau *toolkit* yang digunakan dalam pengembangan perangkat lunak berbentuk *website* dinamis menggunakan bahasa PHP.

Penggunaan CodeIgniter ini membuat pengembangan proyek menjadi lebih cepat dibandingkan memulai menulis *code* dari awal. *Framework* CodeIgniter berisi sekumpulan *library* dan *helper* yang digunakan untuk membantu memecahkan masalah dalam proses membangun *web*. Menurut Daqiqil (2011: 3), CodeIgniter mempunyai kelebihan dari jenis *framework* lain diantaranya adalah kecepatan, mudah dimodifikasi dan beradaptasi, dokumentasi yang lengkap dan jelas, serta learning curve rendah atau mudah untuk dipelajari. CodeIgniter

dibangun menggunakan sebuah *pattern* MVC (*Model View Controller*) yaitu teknik pemrograman untuk memisahkan antara *User Interface* atau antarmuka aplikasi, penyimpanan data, dan proses kerja aplikasi. Berikut adalah penjelasannya:

1) *Model*

Model berisi fungsi-fungsi yang berhubungan dengan, perhitungan, algoritma program atau struktur data, *webservice*, dan apapun yang berhubungan dengan pengolahan data pada sebuah aplikasi *website*. Di dalam *model*, terdapat *class* dan fungsi untuk mengambil, melakukan *update* dan menghapus data *website*.

2) *View*

View berisi fungsi-fungsi yag berhubungan dengan antarmuka atau sesuatu yang ditampilkan ke *end-user*. Tidak ada *logic* dan algoritma program sama sekali pada bagian ini karena hanya berisi *code-code* untuk *layout* atau halaman *web* berupa HTML dan CSS, rss, javascript dan sebagainya.

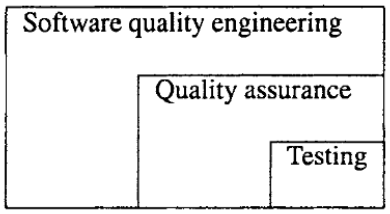
3) *Controller*

Seperti namanya, *controller* bertugas sebagai ruang kontrol, sebagai penghubung, dan menjalankan perintah. *Controller* berfungsi sebagai penghubung data dan *view*. Di dalam *controller* hanya terdapat fungsi yang bertugas untuk memproses permintaan *View* ke dalam struktur data di dalam *Model*. *Controller* tidak boleh berisi *code-code* untuk mengakses basis data ataupun data lainnya tetapi boleh berisi logika atau algoritma program.

d. Testing (Pengujian)

Tahap ini dilakukan ketika tahap *coding* telah selesai. Sebelum *software* digunakan, dilakukan pengujian untuk memastikan *software* tersebut nantinya akan

berjalan dengan baik tanpa ada kesalahan. Menurut Myers, et al. (2012: 2), *software testing* adalah sebuah proses, atau rangkaian proses, yang dirancang untuk memastikan bahwa perangkat lunak dapat menjalankan apa yang sebelumnya didesain untuk dijalankan, dan sebaliknya, yaitu tidak menjalankan sesuatu yang tidak diinginkan. Menurut Naik & Tripathy (2001: 7), *software testing* adalah sebuah proses verifikasi untuk melakukan penilaian kualitas suatu perangkat lunak dan perbaikan. "*Testing is one of the most important parts of quality assurance (QA) and the most commonly performed QA activity*." (Tian, 2005: 65). Beradasarkan beberapa pendapat ahli tersebut *software testing* merupakan salah satu proses dalam menentukan *quality assurance (QA)*. Sedangkan *quality assurance* ditentukan dengan standar internasional yang sudah ada, salah satunya adalah ISO/IEC 9126. Gambar 1 berikut mendeskripsikan cakupan *Software quality engineering*:



Gambar 1. Cakupan dan susunan antara *Testing, quality assurance (QA),* dan *software engineering.*

Software testing secara umum terbagi dalam dua konsep yaitu *verification* dan *validation*. *Verification* adalah sebuah kegiatan mengevaluasi apakah perangkat lunak yang dikembangkan sudah dapat berjalan dengan benar. Sedangkan *validation* adalah sebuah kegiatan untuk memastikan apakah perangkat lunak yang sudah dapat berjalan dengan benar, sesuai dengan keinginan dan kebutuhan *user*

atau *customer*. (Naik & Tripathy, 2011: 7-8). *Software* testing dilakukan menggunakan metode *black box.*

Model *Waterfall* merupakan siklus dari suatu proses pengembangan. Dalam penerapannya, masing-masing tahapan berkesinambungan satu sama lain. Untuk dapat melanjutkan ke tahap selanjutnya, maka tahap yang sedang dikerjakan harus melalui gerbang kualitas untuk memastikan tahap tersebut dapat dinyatakan selesai. Menurut Petersen, et al. (2009: 4) disesuaikan dengan Model *Waterfall* menurut Pressman (2001), setiap gerbang kualitas mempunyai indikasi tersendiri sebagai berikut:

1) Tahap *Analysis* (Analisis)

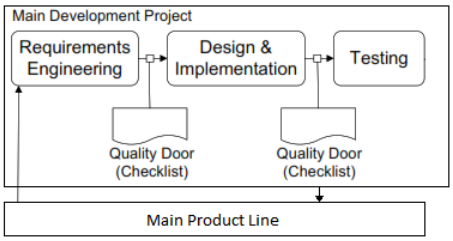
Pada tahap ini, sebelum seluruh persyaratan yang dipilih digunakan sebagai input untuk tahap desain dan implementasi (*code*), maka harus diperiksa terlebih dahulu apakah semua persyaratan telah dipahami, disepakati, dan didokumentasikan. Lebih lanjut, persyaratan juga diperiksa apakah sudah mengidentifikasi pemangku kepentingan yang terkait dan apakah solusinya dapat mendukung strategi bisnis.

2) Tahap *Design* (Desain) dan *Code* (Implementasi)

Pada *quality gate* tahap ini, dilakukan verifikasi apakah arsitektur sudah dievaluasi, apakah ada penyimpangan dari persyaratan (dibandingkan dengan hasil atau keputusan *quality gate* sebelumnya). Selain itu diperiksa apakah ada penyimpangan dari jangka waktu yang telah ditentukan, usaha, dan cakupan produk.

3) Tahap *Testing* (Pengujian)

Pada tahap ini, diperiksa apakah ada penyimpangan dari hasil *quality gate* sebelumnya dalam hal kualitas dan waktu, apakah hasil dari proyek telah memenuhi persyaratan pelanggan. Gambar 2 berikut adalah alur proses dan *quality gate* pada tiap tahapan Model *Waterfall*:



Gambar 2. *Quality Gate* pada Model *Waterfall*

Pengujian yang dilakukan pada aplikasi ini adalah dengan metode *black box.* Pengujian *black box* adalah pengujian yang berfokus pada persyaratan fungsional perangkat lunak. Selain itu, *black box* juga bertujuan untuk menemukan kesalahan dalam fungsi, antarmuka, struktur data, kinerja, inisialisasi, dan penghentian (Pressman, 2010: 587). Pengujian menggunakan metode *black box* karena metode ini merupakan salah satu metode yang paling muda karena hanya memerlukan batas bawah dan batas atas dari data yang diharapkan (Mustaqbal, dkk, 2015: 36).

3. Analisis Kualitas Perangkat Lunak

Kualitas perangkat lunak adalah sebuah proses subyektif dimana sebuah manajemen kualitas berhak untuk memutuskan apakah perangkat lunak telah mencapai level kelayakan yang seharusnya. (Sommerville, 2011:655)

Untuk menentukan apakah sebuah perangkat lunak telah mencapai standar kualitas, perlu dilakukan pengujian atau *testing*. Menurut Pressman (2010: 527), bila pengujian dilakukan secara sukses (sesuai dengan sasaran tersebut), maka akan

ditemukan kesalahan di dalam perangkat lunak. Sebagai keuntungan sekunder, pengujian menunjukkan bahwa fungsi perangkat lunak bekerja sesuai dengan spesifikasi dan bahwa persyaratan kerja telah dipenuhi.

Di dalam jurnal Olsina (1999: 1) mengusulkan *Web Quality Evaluation Method* (*Web*QM) sebagai pendekatan yang digunakan untuk menguji kualitas sistem informasi berbasis *web*. Dari jurnal tersebut dapat diketahui bahwa di dalam *Web*QEM terdapat beberapa karakteristik yang dipilih berdasarkan standar kualitas ISO/IEC 9126. ISO/IEC 9126 adalah salah satu standar kualitas yang dibuat oleh *International Standard Organization* dan *International Electrotechnic Comission* yang digunakan untuk pengujian kualitas perangkat lunak. ISO/IEC 9126 mempunyai 6 karakteristik yaitu: *usability, functionality, reliability, efficiency, maintainability, dan portability*. Olsina (1999: 2), mengemukakan bahwa dalam suatu *website* akademik, sudut pandang yang paling diperhatikan hanya sudut pandang pengguna atau *user*. Maka dari itu Olsina hanya memilih aspek *usability, functionality, reliability,* dan *efficiency* karena aspek *maintainability* dan *portability* tidak merujuk pada *visitor* atau pengunjung. Di dalam jurnal lain yang ditulis oleh Padayachee, et al. (2010:4) juga menggunakan ISO/IEC 9126 dalam pengujian *website e-Learning* tetapi hanya menggunakan keempat aspek yang dipilih Olsina. Hal ini juga dikarenakan keempat aspek tersebut dikembangkan untuk membantu menguji kualitas sistem *e-Learning* berdasarkan sudut pandang *user*.

a. *Reliability*

"*The capability of the software product to maintain a specified level of performance when used under specified conditions*" (ISO/IEC 9126, 1995: 8). Dari

kutipan tersebut dapat diartikan bahwa *reliability* adalah kemampuan suatu produk perangkat lunak untuk mempertahankan suatu level tertentu dalam kinerjanya ketika digunakan dalam suatu kondisi tertentu. *Software reliability* adalah kemungkinan dari tidak adanya kegagalan dalam eksekusi *software* dalam waktu tertentu dan kondisi tertentu. (Pham, 2000: 44).

Pengujian *software* berdasarkan aspek *reliability*, Taylor (2016: 5) mengungkapkan ada beberapa tipe pengujian yang tersedia untuk aspek *reliability* diantaranya adalah*: Functional Testing, Load Testing, Performance Testing, Regression Testing, Scenarion Testing,* dan *Stress Testing*. Pada pengujian aspek *reliability* pada sebuah *web* dapat dilakukan dengan *stress testing*. (MSDN, 2003). *Stress testing is testing conducted to evaluate a system or component at or beyond the limits of its specified requirements*. (IEEE Standard 610.12-1990). *Stress testing* merupakan pengujian yang bertujuan untuk mengevaluasi suatu sistem atau komponen pada batas kemampuannya. Dengan *stress testing* dapat diketahui seberapa kuat kemampuan suatu sistem dalam mempertahankan kinerjanya dalam keadaan yang tidak normal atau melebihi batas kemampuan sistem. Pengujian stress testing dilakukan menggunakan *software* yang dapat melakukan stress testing yaitu WAPT Pro 5.0. Cara kerja *stress testing* pada *software* ini adalah melakukan sebuah skenario menjalankan aplikasi yang akan diuji dengan sejumlah *virtual user* dalam waktu tertentu. Kemudian *software* menganalisis apakah dengan beban tersebut aplikasi tetap dapat berjalan dengan baik. Hasil pengujian dengan *software* ini berupa jumlah dari *successful sessions, pages, hits* dan *failed sessions, pages,* dan *hits.*

b. *Functionality*

Functionality adalah kemampuan produk perangkat lunak untuk menyediakan fungsi yang sesuai dengan spesifikasi kebutuhan yang telah ditetapkan ketika digunakan pada kondisi tertentu. (ISO/IEC 9126-1, 2000:7).

Dalam pengujian fungsionalitas Sistem Informasi Presensi Siswa Berbasis *Web* untuk SMK Negeri 2 Sewon, digunakan metode *black-box testing*. Pressman (2012 : 597) mengungkapkan, *black-box testing* juga disebut pengujian perilaku, berfokus pada persyaratan fungsional perangkat lunak. Menurut Al-Fatta (2007: 172), *Black box testing* adalah pengujian perangkat lunak di mana pengujiannya hanya dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses bisnis yang diinginkan. Pengujian *black-box* terfokus pada fungsi-fungsi masukan dan keluaran, desain antar muka, maupun basis data perangkat lunak tersebut apakah telah memenuhi spesifikasi yang dibutuhkan. Kemudian hasil pengujian akan dianalisis menggunakan metode analisis deskriptif. Pengujian aspek *functionality* dilakukan oleh ahli media yang telah berpengalaman dalam bidang rekayasa *web*.

c. *Usability*

"The capability of the software product to be understood, learned, used and attractive to the user, when user under specified conditions." (ISO/IEC 9126-1, 2000: 9). Dalam kutipan tersebut dapat dijelaskan bahwa *usability* adalah kemampuan perangkat lunak untuk dipahami, dipelajari, digunakan dan menarik bagi pengguna atau *user* saat digunakan pada kondisi tertentu. Menurut Lautenbach et al., 1999: 2, sebuah sistem lebih dikenali oleh *user* karena desain antarmuka-nya,

sehingga ketika desainer memperbaiki *usability* sebuah sistem maka dia harus memperbaiki desain antarmuka-nya. Sebuah *website* atau sistem akan lebih mudah digunakan jika desain antarmukanya baik. Dengan menguji aspek *usability,* maka dapat diketahui seberapa efektif dan mudah seorang *user* menggunakan *website* tersebut.

Pengujian *usability* dilakukan menggunakan kuisioner dengan model USE *Questionnaire* yang dikembangkan oleh A.M. Lund (2001). Instrumen USE *questionnaire* terdiri dari 4 kriteria yaitu *usefulness, ease of use, ease of learning,* dan *satisfaction* yang tergabung dalam 30 butir pertanyaan. Kemudian untuk jumlah titik respon digunakan Skala Likert. Skala Likert digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang fenomena sosial (Sugiyono, 2011:93). Skala Likert mempunyai 5 titik respon yaitu tidak setuju, kurang setuju, netral, setuju, dan sangat setuju. Menurut Preston & Colman (2000: 12), Skala Likert dengan titik respon kurang dari 5 dinilai mudah digunakan tetapi juga dinilai kurang membantu responden dalam mengemukakan perasaan atau pendapatnya.

d. *Efficiency*

"The capability of the software product to adhere to standards, conventions, style guides or regulations." (ISO/IEC 9126-1, 2000:10). Dari kutipan tersebut dapat dijelaskan bahwa *efficiency* adalah kemampuan suatu produk perangkat lunak untuk memberikan kinerja yang tepat, sesuai dengan jumlah sumber daya yang digunakan pada kondisi tertentu. Menurut Berander, et al. (2005: 7), *efficiency* secara umum dapat diartikan sebagai penggunaan sumber daya, seperti kecepatan

proses dan penggunaan memory atau penyimpanan. *Efficiency* memiliki beberapa karakteristik yang terdiri dari, 1) *time behavior* berhubungan dengan respon dan waktu pengolahan atau proses dalam menjalankan fungsi suatu perangkat lunak, 2) *resource behavior* berhubungan dengan jumlah dan tipe sumber daya yang dimiliki dalam menjalankan fungsi yang dalam kondisi tertentu, 3) *compliance* adalah kemampuan suatu perangkat lunak dalam memenuhi standar dan kebutuhan yang berkaitan dengan efisiensi.

Menurut Croll & Power (2009: 249), untuk menguji *efficiency* desain halaman sebuah *website* dapat menggunakan tools seperti Firebug dan YSlow. Dalam penelitian ini, analisis aspek *efficiency* dilakukan menggunakan *website page speed analyzer bernama GTmetrix.com yang dikembangkan oleh GT.net* karena Firebug dan YSlow sudah tidak *compatible* dengan *browser* versi terbaru saat ini. Meskipun tidak digunakan secara langsung, tetapi nantinya hasil pengujian yang dikeluarkan GTmetrix akan mengacu pada dua aturan menurut YSlow dan PageSpeed. GTmetrix akan melakukan *load test* yaitu "Penentuan atau validasi kinerja karakteristik sistem atau aplikasi yang diuji ketika mengalami beban kerja dan volume beban yang diantisipasi selama operasi produksi." (Proko & Ninka, 2013). Menurut Yahoo *Developer Network*, YSlow berfungsi untuk menganalisa kinerja suatu halaman *web* berdasarkan seluruh komponen yang ada pada halaman *web* tersebut ketika sedang dieksekusi, kemudian YSlow akan memberikan laporan tentang hal apa saja yang membuat kinerja halaman *web* tersebut menjadi lambat dan memberikan saran untuk meningkatkan kinerjanya.

B. Kajian Penelitian yang Relevan

Beberapa penelitian yang relevan dengan penelitian ini yaitu sebagai berikut:

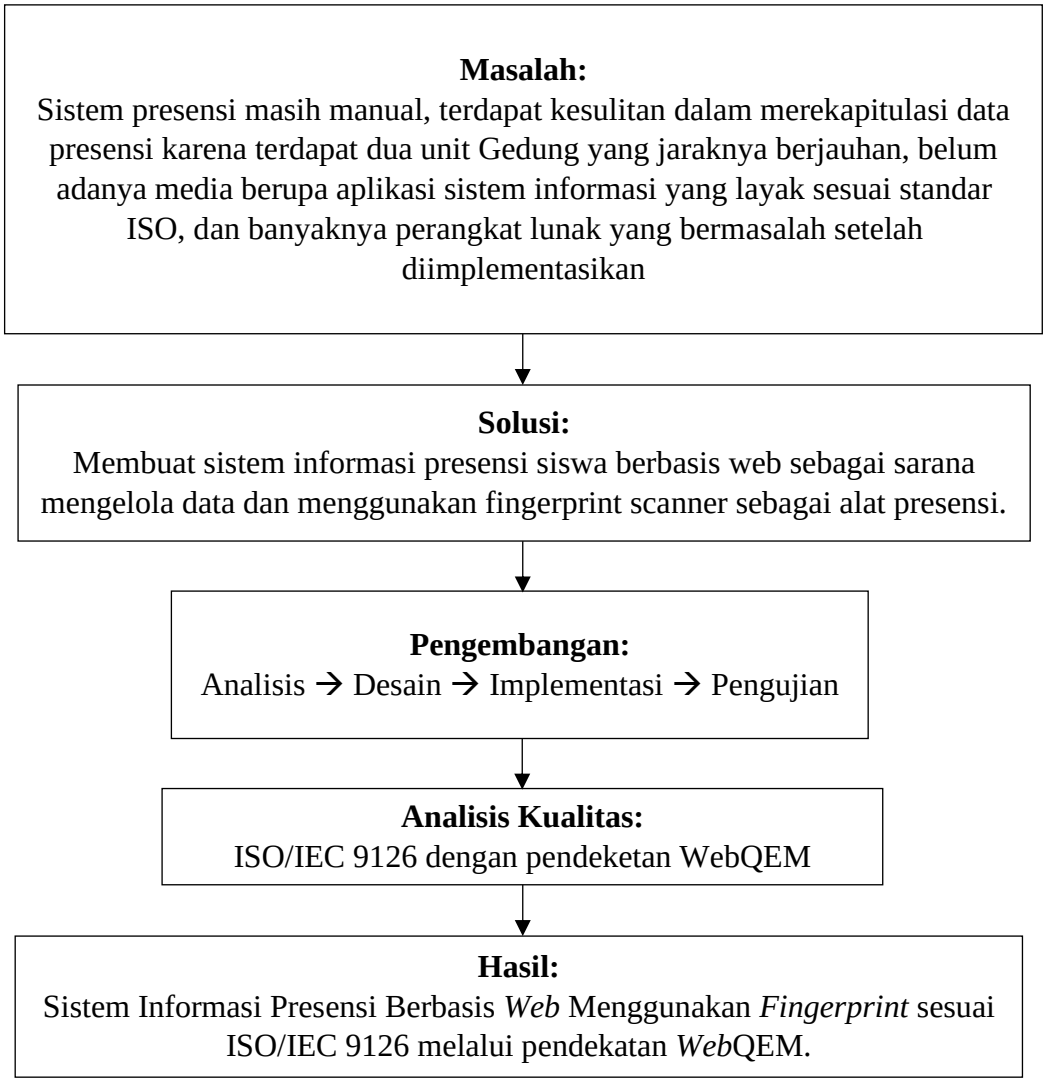
1. Penelitian Tika Novita Sari (2014) dengan judul "Pengembangan dan Analisis Kualitas Sistem Informasi Akademik Berbasis *Web* Menggunakan *Standard* ISO 9126". Peneliatian tersebut dilakukan menggunakan metode *Research and Development* (R&D), dan model pengembangan perangkat lunak menggunakan model *Waterfall*. Standar kualitas yang digunakan untuk analisis kualitas perangkat lunak menggunakan ISO/IEC 9126 melalui pendekatan *Web*QEM yaitu meliputi aspek *functionality, usability, reliability,* dan *efficiency*.
2. Penelitian W. Ahmad Nurrohman (2017) dengan judul Pengembangan Dan Analisis Kualitas Sistem Informasi Perijinan Siswa Berbasis *Web* Di SMK Negeri 1 Wonosari. Penelitian ini juga menggunakan model pengembangan R&D dan menggunakan model *Waterfall*. Peneliti mengembangkan perangkat lunak dalam bidang pendidikan yaitu perizinan siswa yang juga termasuk dalam bagian system presensi. Perangkat lunak dibuat berbasis *website* dengan *framework CodeIgniter.* Pengujian perangkat lunak menggunakan standar ISO 9126 dengan menggunakan 4 aspek yaitu *functionality, reliability, efficiency, dan usability*. Hasil penelitian ini hamper semua aspek mencapai nilai sempurna dan layak untuk digunakan atau diimplementasikan di sekolah.
3. Penelitian Elisabeth Febrina Tuto Burak Lamatokan (2016) dengan judul Sistem Presensi Mahasiswa Berbasis Web Menggunakan Finger Print Scanner. Penelitian ini menggunakan *finger print scanner* sebagai inputan data presensi

dan *website* sebagai *user* interface dan media pengelolaan data. Dalam pengembangan *website*, peneliti menggunakan PHP dan *database* MySQL. Sistem Presensi ini layak digunakan yang dibuktikan dengan nilai akhir uji coba sistem yaitu 4.65 dari skala 1-5, serta dapat dibuktikan dengan 13 dari 20 responden (65%) menyatakan sangat setuju, 7 dari 20 responden (35%) menyatakan setuju.

C. Kerangka Berpikir

Presensi siswa merupakan salah satu aspek administrasi di sekolah yang cukup kompleks dan perlu dikelola dengan baik. Karena presensi siswa juga berpengaruh pada nilai akhir siswa tersebut. Di SMK N 2 Sewon, masih terdapat suatu permasalahan terkait dengan proses presensi, pengelolaan data presensi per-hari, dan rekap data presensi per-semester karena hal tersebut masih dilakukan dengan cara manual. Di samping itu SMK N 2 Sewon juga terbagi dalam 2-unit gedung sekolah yang jarak berjauhan. Berawal dari analisis masalah tersebut, maka dikembangkan suatu media yang berupa sistem informasi presensi berbasis *web* dan penambahan alat berupa *fingerprint* guna membantu tugas guru piket dan guru BK dalam mengelola data presensi tersebut. Proses pengembangan pada penelitian ini menggunakan pendekatan model *Waterfall* yang meliputi tahap *analysis, design, implementation,* dan *test*. Selanjutnya untuk mengurangi tingkat kesalahan atau *error* sistem informasi tersebut ketika digunakan, maka dilakukan pengujian kualitas sistem informasi menggunakan standar kualitas ISO/IEC 9126 dengan menggunakan pendekatan *Web*QEM yang meliputi empat aspek yaitu *functionality*,

efficiency, *reliability*, dan *usability*. Gambar 3 berikut adalah kerangka berpikir penelitian.



Gambar 3. Kerangka Berpikir Penelitian

D. Pertanyaan Penelitian

Berdasarkan kerangka berpikir di atas, terdapat beberapa pertanyaan penelitian terkait pengembangan Sistem Informasi Presensi Berbasis *Web* Menggunakan *Fingerprint* di SMK N 2 Sewon dan analisis kualitas perangkat lunak sesuai standar

kualitas ISO/IEC 9126 melalui pendekatan *Web*QEM dari aspek *functionality*, *efficiency*, *reliability*, dan *usability*.

1. Bagaimana proses pengembangan Sistem Informasi Presensi Berbasis *Web* menggunakan *fingerprint* di SMK N 2 Sewon?
2. Bagaimana analisis kualitas Sistem Informasi Presensi Berbasis *Web* menggunakan *fingerprint* di SMK N 2 Sewon pada aspek *functionality?*
3. Bagaimana analisis kualitas Sistem Informasi Presensi Berbasis *Web* menggunakan *fingerprint* di SMK N 2 Sewon pada aspek *reliability?*
4. Bagaimana analisis kualitas Sistem Informasi Presensi Berbasis *Web* menggunakan *fingerprint* di SMK N 2 Sewon pada aspek *efficiency?*
5. Bagaimana analisis kualitas Sistem Informasi Presensi Berbasis *Web* menggunakan *fingerprint* di SMK N 2 Sewon pada aspek *usability?*