BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

A. SDLC (*Software Development Life Cycle*)

> Sukamto dan Shalahuddin (2014a:26) mendeskripsikan:
> SDLC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah teruji baik)".

Seperti halnya metamorfosis kupu-kupu, untuk menjadi kupu-kupu yang indah maka dibutuhkan beberapa tahap untuk dilalui, sama halnya dengan membuat perangkat lunak, memiliki daur tahapan yang dilalui agar menghasilkan perangkat lunak yang berkualitas.

Sukamto dan Shalahuddin (2014b:26) menjabarkan tahapan-tahapan yang ada pada SDLC secara global adalah sebagai berikut:

1. Inisiasi (*initiation*)

   Tahap ini biasanya ditandai dengan pembuatan proposal proyek perangkat lunak.

2. Pengembangan konsep sistem (*system concept development*)

   Mendefinisikan lingkup konsep termasuk dokumen lingkup sistem, analisis manfaat biaya, manajemen rencana, dan pembelajaran kemudahan sistem.

3. Perencanaan (*planning*)

   Mengembangkan rencana manajemen proyek dan dokumen perencanaan lainnya. Menyediakan dasar untuk mendapatkan sumber daya (*resources*) yang dibutuhkan untuk memperoleh solusi.

4. Analisis kebutuhan (*requirements analysis*)

   Menganalisis kebutuhan pemakai sistem perangkat lunak (*user*) dan mengembangkan kebutuhan *user*. Membuat dokumen kebutuhan fungsional.

5. Desain (*design*)

   Mentransformasikan kebutuhan detail menjadi kebutuhan yang sudah lengkap, dokumen desain sistem fokus pada bagaimana dapat memenuhi fungsi-fungsi yang dibutuhkan.

6. Pengembangan (*development*)

   Mengonversi desain ke sistem informasi yang lengkap termasuk bagaimana memperoleh dan melakukan instalasi lingkungan sistem yang dibutuhkan, membuat basis data dan mempersiapkan prosedur kasus pengujian, mempersiapkan berkas atau *file* pengujian, pengodean, pengompilasian, memperbaiki dan membersihkan program, peninjauan, pengujian.

7. Integrasi dan pengujian (*integration and test*)

   Mendemonstrasikan sistem perangkat lunak bahwa telah memenuhi kebutuhan yang dispesifikasikan pada dokumen fungsional. Dengan diarahkan oleh staf penjamin kualitas (*quality assurance*) dan *user*. Menghasilkan laporan analisis pengujian.

8. Implementasi (*implementation*)

Termasuk pada persiapan implementasi, implementasi perangkat lunak pada lingkungan produksi (lingkungan pada *user*) dan menjalankan resolusi dari permasalahan yang teridentifikasi dari fase integrasi dan pengujian.

9. Operasi dan pemeliharaan (*operations and maintenance*)

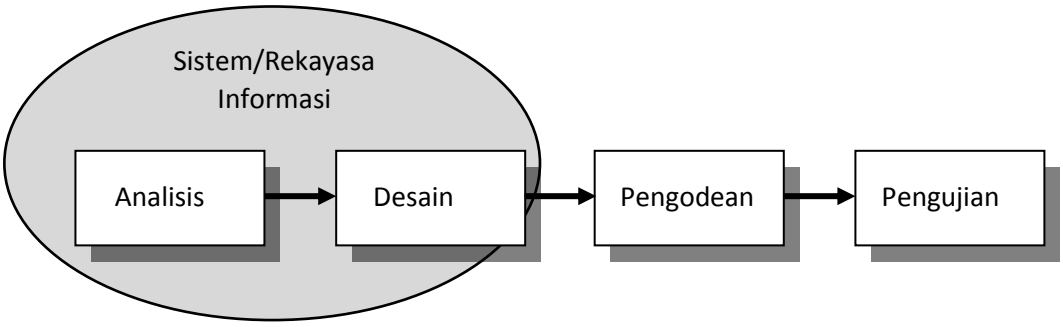
   Mendeskripsikan pekerjaan untuk mengoperasikan dan memelihara sistem informasi pada lingkungan produksi (lingkungan pada *user*), termasuk implementasi akhir dan masuk pada proses peninjauan.

10. Disposisi (*disposition*)

    Mendeskripsikan aktifitas akhir dari pengembangan sistem dan membangun data yang sebenarnya sesuai dengan aktifitas *user*.

B. Teori Model *Waterfall*

Sukamto dan Shalahuddin (2014c:31) mendefinisikan "Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linier atau alur hidup klasik. Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut mulai dari analisis, desain, pengodean, pengujian dan tahap pendukung (*support*)".



Sumber: Sukamto dan Shalahuddin (2014d:29)

Gambar II.1. Ilustrasi Model *Waterfall*

Pembuatan model *waterfall* tentu tidak terlepas dari beberapa tahapan yang harus dikerjakan secara terstruktur. Berikut ini adalah tahapan-tahapan model pengembangan sistem model *waterfall*:

1. Analisa Kebutuhan Perangkat Lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk memspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan pengguna. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antar muka, dan prosedur pengkodean. Tahap ini mentranslasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Penulisan Kode Program

   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian fokus pada perangkat lunak dari segi logika dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung (*Support*) atau Pemeliharaan (*Maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke pengguna. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

C. Tools System

Penulis menggunakan tiga (3) *tools system* untuk membantu pembangunan sistem informasi akademik berbasis web ini, yaitu:

1. UML (*Unified Modeling Language*)

   Menurut Sukamto dan Shalahuddin (2014e:133) "UML *(Unifed Modeling Language)* adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement,* membuat analisis, & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek".

   UML menyediakan 9 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

   a. *Use Case Diagram*

      Use case diagram atau diagram *use case* 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. Secara kasar *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan

siapa saja yang berhak menggunakan fungsi-fungsi itu (Sukamto dan Shalahuddin, 2014f:155).

b. *Sequence Diagram*

Diagram sekuen atau *sequence* diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuaen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang dinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case* (Sukamto dan Shalahuddin, 2014g:165).

c. *Package Diagram*

Menurut Widodo dan Herlawati (2011a:10) "*Package diagram* bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen".

d. *State Machine Diagram*

State machine diagram atau *statechart diagram* atau dalam bahasa Indonesia disebut diagram mesin status atau sering juga disebut diagram status yang digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. Jika diagram sekuen digunakan untuk interaksi antar objek maka diagram status digunakan untuk interaksi didalam sebuah objek. Perubahan tersebut digambarkan dalam suatu grafik berarah (Sukamto dan Shalahuddin, 2014h:163).

e. *Activity Diagram*

Sukamto dan Shalahuddin (2014i:161) menjelaskan "Diagram aktivitas atau *activity diagram* adalah menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak". Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktifitas sistem bukan apa yang dilakuakn oleh aktor. Diagram aktifitas juga banyak digunakan untuk mendefinisikan hal-hal sebagai berikut:

1) Rancangan proses bisnis dimana setiap urutan aktifitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2) Urutan atau pengelompokan tampilan dari sitem atau *user interface* dimana setiap aktifitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3) Rancangan pengujian dimana setiap aktifitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4) Rancangan menu yang ditampilkan pada perangkat lunak.

f. *Class Diagram*

Diagram kelas atau *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 variabel-variabel yang dimiliki oleh suatu kelas. Metode atau operasi adalah fungsi-fungsi yang dimiliki oleh suatu kelas (Sukamto dan Shalahuddin, 2014j:141).

g. *Object Diagram*

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan. Diagram objek juga berfungsi untuk mendefinisikan contoh nilai atau isi dari atribut tiap kelas (Sukamto dan Shalahuddin, 2014k:147).

h. *Component Diagram*

Diagram komponen komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yan telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas, antarmuka, serta kolaborasi (Widodo dan Herlawati, 2011b:11).

i. *Deployment Diagram*

Menurut Sukamto dan Shalahuddin (2014l:154) "Diagram *deployment* atau *deployment diagram* menunjukan konfigurasi komponen dalam proses eksekusi aplikasi". Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal sebagai berikut:

1) Sistem tambahan *(embedded system)* yang menggambarkan rancangan *device, node,* dan *hardware*-nya.
2) Sistem *client/server*.
3) Sistem terdistribusi murni.
4) Rekayasa ulang aplikasi.

2. ERD (*Entity Relationship Diagram*)

Menurut Ladjamudin (2008:189), "*Entity Relationship Diagram* (ERD) adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem scara abstrak". Jadi jelaslah bahwa ERD ini berbeda dengan DFD yang merupakan suatu model jaringan fungsi yang akan dilaksanakan oleh sistem, sedangkan ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan *relationship* data.

Diagram hubungan entitas yang biasa lebih dikenal dengan sebutan ER *diagram*, adalah notasi grafik dari sebuah model yang data atau sebuah model jaringan yang menjelaskan tentang data yang tersimpan (*storage data*) dalam sistem secara abstrak. Diagram hubungan entitas tidak menyatakan bagaimana memanfaatkan data, membuat data, mengubah data dan menghapus data. Adapun elemen-elemen diagram hubungan entitas antara lain:

a. Entitas (*entity*)

Pada ERD, entity digambarkan dalam sebuah bentuk persegi panjang. Entity adalah sesuatu apa saja yang ada didalam sistem, nyata maupun abstrak dimana data tersimpan atau dimana terdapat data. Entitas diberi nama dengan kata benda dan dapat dikelompokan dalam empat jenis nama yaitu: orang, benda, lokasi, kejadian (terdapat unsur waktu di dalamnya).

b. Atribut *Value*

Atribut *Value* atau nilai attribute adalah suatu *occurrence* tertentu dari sebuah *attribute* didalam suatu *entity* atau *relationship.* Ada dua jenis attribute yaitu *identifier (key)* digunakan untuk menentukan suatu *entity*

secara unik *(primary key)* dan *descriptor (non-key attribute)* digunakan untuk menspesifikasikan karakteristik dari suatu *entity* yang tidak unik.

c. Kardinalitas (*Cardinality)*

Kardinalitas relasi menunjukan jumlah maksimum tupel yang dapat berelasi dengan entitas yang lain. Pada contoh sebelumnya dapat kita lihat bahwa tupel-tupel pada entitas mahasiswa dapat berelasi dengan satu tupel, banyak tupel atau bahakan tidak satupun tupel dari entitas kuliah. Begitu juga sebaliknya, entitas-entitas pada entitas kuliah ada yang berelasi dengan beberapa tupel pada entitas mahasiswa dan ada pula yang berelasi dengan satu tupel pada entitas mahasiswa.

d. Relasi

Relasi adalah hubungan antara beberapa entitas. sebagai contoh relasi antar mahaiswa dengan mata kuliah dimana setiap mahasiswa bisa mengambil beberapa mata kuliah dan setiap mata kuliah bisa diambil oleh lebih dari 1 mahasiswa. relasi tersebut memiliki hubungan banyak ke banyak. Macam-macam relasi antara lain:

1) One-to-one

Sebuah entitas pada A berhubungan dengan entitas B paling banyak 1 contoh diatas relasi pegawai dan departemen dimana setiap pegawai hanya bekerja pada 1 departemen.

2) One-to-many

Sebuah entitas pada A berhubungan dengan entitas B lebih dari satu contoh diatas adalah 1 depertemen memiliki banyak pegawai.

3) Many-to-many

Sebuah entitas pada A berhubungan dengan entitas B lebih dari satu dan B berhubungan dengan A lebih dari satu juga contoh diatas adalah relasi mahasiswa dengan mata kuliah.

3. LRS (*Logical Record Structure*)

Setelah proses desain E-R diagram selesai dilakukan, langkah berikutnya adalah mentransformasi E-R diagram tersebut ke dalam LRS (*Logical Record Structure*). Aturan-aturan dalam melakukan transformasi E-R diagram ke *Logical Record Structure* menurut Ladjamudin (2013:159) sebagai berikut :

a. Setiap *entity* akan diubah ke bentuk sebuah kotak dengan nama *entity* berada di luar kotak dan atribut berada di dalam kotak.
b. Sebuah relasi kadang disatukan dalam sebuah kotak bersama *entity*, kadang dipisah dalam sebuah kotak tersendiri.

D. Pemrograman Terstruktur

Sukamto dan Shalahuddin (2014m:67) mendeskripsikan "Pemrograman terstruktur adalah konsep atau paradigma atau sudut pandang pemrograman yang membagi-bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer". Modul-modul (pembagian program) biasanya dibuat dengan mengelompokkan fungsi-fungsi dan prosedur-prosedur yang diperlukan sebuah proses tertentu.

Fungsi-fungsi dan prosedur-prosedur ditulis secara sekuensial atau terurut dari atas ke bawah sesuai dengan kebergantungan antar fungsi atau prosedur (fungsi atau prosedur yang dapat dipakai oleh fungsi atau prosedur dibawahnya harus yang sudah ditulis atau dideklarasikan diatasnya).

Pemodulan pada pemrograman terstruktur dibagi berdasarkan fungsi-fungsi dan prosedur-prosedur. Oleh karena itu, pemodelan pada pemrograman terstruktur lebih fokus bagaimana memodelkan data dan fungsi-fungsi atau prosedur-prosedur yang harus dibuat. Jenis paradigma pemrograman yang digunakan dapat dideteksi dari bahasa pemrograman apa yang akan digunakan untuk membuat program, baru setelah itu ditentukan paradigma pemrograman apa yang akan digunakan.

E. Bahasa Pemrograman

Bahasa pemrograman yang penulis gunakan untuk membangun sistem informasi akademik berbasis *web* ini adalah sebagai berikut:

1. HTML (*HyperText Markup Languange*)

   Menurut Sibero (2013a:19) "*HyperText Markup Language* atau HTML adalah bahasa yang digunakan pada dokumen *web* sebagai bahasa untuk pertukaran dokumen *web*". Bahasa pemrograman HTML versi 1.0 dibangun oleh W3C, dan terus mengalami perkembangan. Sampai saat ini HTML terakhir adalah versi 5.0. Struktur dokumen HTML terdiri dari *tag* pembuka dan *tag* penutup.

2. PHP (*Personal Home Page*)

   Menurut Sibero (2013b:46), "PHP (*PHP Hypertext Preprocessor*) adalah sebuah bahasa pemrograman yang berbentuk *scripping*, yang digunakan untuk membuat halaman *web* yang dinamis". Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme menyebabkan informasi yang akan diterima *client* selalu yang terbaru. Sistem kerja dari program ini adalah sebagai *interpreter* dan bukan sebagai *compiler*.

Dikatakan sebagai bahasa *interpreter*, *script* mentahnya tidak harus diubah ke dalam bentuk *source code*. Sedangkan disebut sebagai bahasa *compiler* bahasa yang akan mengubah *script-script* program ke dalam *source code*, yang akan diubah menjadi bentuk objek kode dan akan menghasilkan *file* yang lebih kecil dari *file* mentah sebelumnya, sehingga hasil dari bahasa pemograman yang berbentuk *compiler* akan membentuk sebuah program yang berstatus sebagai program exe. PHP dalam bahasa pemograman *server side* sudah banyak digunakan pada saat ini, terutama untuk pembuatan *website* dinamis. Untuk hal-hal tertentu dalam pembuatan *web*, bahasa pemograman PHP memang diperlukan, misalnya saja untuk memproses data yang dikirimkan oleh pengunjung *web.*

PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP bernama FI (*form interpreted*). Pada saat tersebut PHP adalah sekumpulan *script* yang digunakan untuk mengelola data *form* dari *web*. *Web server* yang mendukung php dapat ditemukan dimana-mana dari mulai IIS sampai dengan *apache*, dengan konfigurasi yang relatif mudah. Hubungan PHP dengan HTML halaman *web* biasanya disusun dari kode-kode HTML yang disimpan dalam sebuah *file* berekstensi. Program PHP harus diterjemahkan oleh *web-server* sehingga menghasilkan kode html yang dikirim ke *browser* agar dapat ditampilkan. PHP diawali dengan <?php dan diakhiri dengan ?>. Pasangan kedua kode inilah yang berfungsi sebagai *tag* kode PHP.

3. CSS (*Cascading Style Sheets*)

Menurut Sugiri dan Kurniawan (2007:21) CSS (*Cascading Style Sheets*) adalah "sebuah cara untuk memisahkan isi dengan *layout* dalam halaman-halaman *web* yang dibuat". CSS memperkenalkan *template* yang berupa *style* untuk membuat dan mempermudah penulisan dari halaman-halaman yang dirancang. Hal ini sangat pernting karena halaman yang menggunakan CSS dapat dibaca secara bolak balik dan isinya dapat dilihat oleh pengunjung dari manapun.

CSS mampu menciptakan halaman yang tampak sama pada resolusi layar dari pengunjung yang berbeda tanpa memerlukan suatu tabel. Dengan CSS akan lebih mudah melakukan *setting* tampilan keseluruhan *web* hanya dengan menggantikan atribut-atribut atau perintah dalam *style* CSS dengan atribut yang diinginkan tanpa harus mengubah satu persatu atribut tiap elemen yang ada dalam situs yang dibuat.

Penggunaan CSS dalam *web* akan lebih efisien karena CSS dapat digunakan untuk penggunaan secara berulang pada *tag-tag* tertentu sehingga tidak usah menggantikan ulang seluruh perintah pemformatan seperti halnya HTML klasik. CSS akan lebih mudah dipelajari jika telah mengetahui struktur pembuatan dokumen *web* dengan bahasa HTML.

4. Javascript

Menurut Sibero (2013c:150) "*Javascript* adalah bahasa skrip (*scripting language*), yaitu kumpulan instruksi atau perintah yang digunakan untuk mengendalikan beberapa bagian dari sistem operasi". Bentuk bahasa skrip

dari *javascript* mengambil model penulisan pada pemrograman bahasa C dan Java, yang terdiri dari variabel, fungsi dan lainnya.

Sebagai bahasa skrip yang berjalan pada web browser atau sisi klien (*client side*). *Javascript* tidak memiliki fungsi untuk menjalankan suatu perintah pada *server* atau sisi *server* (*server side*). Dengan keterbatasan itu para pengembang *javascript* kemudian menambahkan suatu mekanisme agar *javascript* dapat berinteraksi dengan *server*. Mekanisme tersebut adalah AJAX (*asynchronous javascript and XML*), yaitu mekanisme komunikasi antara *javascript* yang berada di sisi klien dengan bahasa di sisi *server* seperti PHP dan lainnya.

2.2. Penelitian Terkait

Setiyawan dkk (2012:1) menjelaskan:
Pendidikan adalah salah satu faktor yang sangat penting dalam perkembangan suatu negara. Semakin baik perkembangan, isi dan kualitas pendidikan suatu bangsa maka akan semakin baik pula perkembangan negara tersebut. Semakin mengerti suatu bangsa akan pentingnya suatu pendidikan maka akan semakin baik kualitas pendidikan negara tersebut. Indonesia adalah negara yang masih dalam tahap perkembangan, tentunya dalam segala aspek dan tidak terkecuali dunia pendidikan. Saat ini kenyataan yang terjadi tentang dunia pendidikan di Indonesia masih dapat dibilang tidak terlalu baik, terutama dalam bidang pemerataan pendidikan di negara ini. Sistem pendidikan yang diterapkan kebanyakan masih memakai sistem pendidikan konvensional atau dengan sistem pendidikan cara lama yang menggunakan cara-cara manual dalam aktivitas pendidikannya baik mengenai cara pengolahan data maupun sistem akademik pendidikannya. Padahal di era serba teknologi ini kecepatan mengolah data dan memperoleh informasi sangat diperlukan dalam dunia pendidikan sekarang.

Susilo dan Kesuma (2014:91) menjelaskan:
Seperti halnya yang ada di SMA Negeri 1 Kutasari dimana sistem informasi akademik masih menggunakan proses yang mengacu pada dokumen yang berupa berkas atau arsip dan belum memiliki penyimpanan basis data yang terbagi ke dalam beberapa tahapan yaitu pertama, melakukan pencatatan data siswa di buku induk. Kedua, membuat laporan data siswa. Ketiga, pencatatan nilai akademik siswa. Keempat, pencatatan data staf dan guru di buku staf dan guru. Kelima, membuat jadwal

pelajaran siswa dan nama guru yang mengajar mata pelajaran tersebut ke dalam buku mata pelajaran siswa. Dengan demikian, sistem informasi yang ada tersebut masih berjalan manual, sehingga sering terjadi kesalahan-kesalahan dalam proses pengolahan data akademik siswa.

Ridho dkk (2012:41) menjelaskan:
Sistem informasi akademik SMK Triatma Jaya yang dibuat dengan memanfaatkan teknologi informasi untuk mempermudah siswa dan guru untuk mengakses informasi akademik sekolah. SMK Triatma Jaya Semarang merupakan sekolah yang sedang berkembang yang salah satu program keahliannya yaitu rekayasa perangkat lunak yang menginginkan sebuah sistem informasi yang memanfaatkan teknologi informasi berbasis web. Sistem informasi yang berbasis *web* dapat membantu pekerjaan kurikulum dan guru dalam mengolah data siswa berkaitan dengan akademik siswa. Sistem informasi akademik yang berbasis *web* menggunakan pemrograman PHP dengan *database* MySQL.