BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Tugas akhir atau sering disingkat TA merupakan Langkah awal untuk dapat belajar dalam menghadapi dunia kerja yang akan dihadapi, dengan adanya Tugas Akhir mahasiswa dapat mempersiapkan diri untuk menyelesaikan proyek-proyek dimasa kerja nanti. Sebelum Menyusun tugas akhir ada beberapa alur atau Langkah yang harus ditempuh, Mahasiswa perlu melakukan tahapan-tahapan untuk menyelesaikan tugas akhir dimulai dengan pengajuan judul, abstrak dan pengajuan dosen pembimbing (Siswanto, L.2020)

Adapun perbedaan penelitian dengan penelitian terdahulu dapat dilihat pada tabel dibawah ini:

Nama Penulis	Judul	Metode	Hasil
)Irvan Cianstury ,2023)	Sistem Informasi Pengajuan Judul Skripsi Berbasis Online Di Fakultas Ilmu Sosial	*Waterfall*	Hasil dari penelitian ini Adalah Aplikasi pengajua udul skripsi yang dibua liharapkan dapa nemberikan nilai positif bag elayanan Fakultas Ilmu lmu Sosial Universita Dehasen Bengkulu kepad nahasiswa sesuai denga rogram studi yang diambil.
(Muhammad Andhika Dharmawan et al., 2019)	Implementasi Sistem Informasi Tugas Akhir Menggunakan Metode Classic Life Cycle	Classic Life Cycle	Berdasarkan sistem yang sudah di jalankan maka penelitian sistem informasi pengajuan judul skripsi online ini sudah di uji cobakan kepada mahasiswa. Hasil dari pengujian tersebut adalah mahasiswa menjalankan sistem informasi pengajuan judul skripsi online

(Noberto Djami Nuli., 2023)	Sistem Pengajuan Tugas Akhir Berbasis Website Pada Program Studi Teknik Informatika	*Extreme Programming (XP)*	Dalam kesimpulan, penelitian ini berhasil mencapai tujuan dalam merancang dan mengimplementasikan aplikasi berbasis web yang efektif dan efisien. Aplikasi ini dapat mempercepat proses pengajuan judul skripsi, memberikan pengumuman hasil secara lebih efisien, dan meningkatkan pelacakan bimbingan skripsi.
(Nazla Fitria, 2024)	Perancangan Sistem Informasi Pengelolaan Judul Proposal Tugas Akhir Mahasiswa Prodi Sistem Informasi Berbasis Web	*Prototype*	Perancangan aplikasi sistem informasi Pengelolaan Judul Proposal Tugas Akhir Mahasiswa Program Studi Sistem Informasi memiliki dua kebutuhan, yaitu kebutuhan fungsional dan non fungsional. Pada kebutuhan fungsional yang dirancang yaitu memiliki dua user, yaitu admin dan mahasiswa.

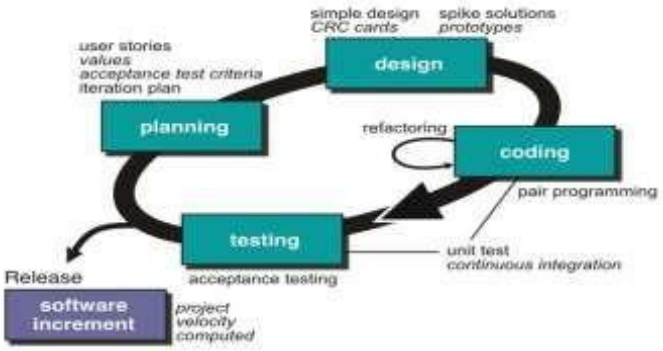
(Rifqi Ainun Niam., 2020)	Peerancangan Sistem Pengajuan Judul Skripsi Universitas PGRI Semarang Berbasis Web di UPT TIK Universitas PGRI Semarang	*Waterfall*	Sistem Pengajuan Judul Skripsi telah berhasil dirancang dengan menggunakan metode SDLC Waterfall dan dapat berfungsi dengan baik sesuai dengan alur kerja yang terdapat dalam proses pengajuan judul skripsi di Universitas PGRI Semarang. 2. Implementasi dari rancangan Sistem Pengajuan Judul Skripsi ini dapat menyimpan data dengan baik dan dapat dipanggil sewaktu-waktu jika diperlukan.

2.2 Metode Pengembangan *Extreme Programing*

XP adalah metodologi pengembangan perangkat lunak yang ditujukan untuk meningkatkan kualitas perangkat lunak dan tanggap terhadap perubahan kebutuhan pelanggan. Jenis pengembangan perangkat lunak semacam ini dimaksudkan untuk meningkatkan produktivitas dan memperkenalkan pos pemeriksaan di mana persyaratan pelanggan baru dapat diadopsi (Pressman, 2012).

Metode ini membawa unsur-unsur yang menguntungkan dari praktek rekayasa perangkat lunak tradisional ke tingkat “ekstrem”, sehingga metode ini dinamai *Extreme Programming*. Unsur-unsur yang menjadi karakteristik

metodologi adalah kesederhanaan, komunikasi, umpan balik, dan keberanian. Gambar tahapan XP dapat dilihat pada gambar 2.1 :



Gambar 2 1 Tahapan *Extreme Programm*

Tahapan-tahapan dari *Extreme Programming* terdiri dari :

1. *Planning*

   Pada *Planning* berfokus untuk mendapatkan gambaran fitur dan fungsi dari perangkat lunak yang akan dibangun. Aktivitas *planning* dimulai dengan membuat kumpulan gambaran atau cerita yang telah diberikan oleh klien yang akan menjadi gambaran dasar dari perangkat lunak tersebut. Kumpulan gambaran atau cerita tersebut akan dikumpulkan dalam sebuah indeks dimana setiap poin memiliki prioritasnya masing-masing. Tim pengembang aplikasi juga akan menentukan perkiraaan waktu serta biaya yang dibutuhkan untuk masing-masing indeks.

   Setelah semua kebutuhan terpenuhi, tim XP akan menentukan alur dari pengembangan aplikasi sebelum memulai pengembangan tugas.

2. *Design*

   Aktivitas *design* dalam pengembangan aplikasi ini, bertujuan untuk mengatur pola logika dalam sistem. Sebuah desain aplikasi yang baik adalah desain yang dapat mengurangi ketergantungan antar setiap proses pada sebuah sistem. Jika salah satu fitur pada sistem mengalami kerusakan, maka hal tersebut tidak akan mempengaruhi sistem secara keseluruhan.

   Tahap *Design* pada model proses *Extreme Programming* merupakan panduan dalam membangun perangkat lunak yang didasari dari cerita klien sebelumnya yang telah dikumpulkan pada tahap *planning*. Dalam XP, proses *design* terjadi sebelum dan sesudah aktivitas *coding* berlangsung. Artinya,

aktivitas *design* terjadi secara terus-menerus selama proses pengembangan aplikasi berlangsung.

3. *Coding*

Setelah menyelesaikan gambaran dasar perangkat lunak dan menyelesaikan *design* untuk aplikasi secara keseluruhan, XP lebih merekomendasikan tim untuk membuat modul unit tes terlebih dahulu yang bertujuan untuk melakukan uji coba setiap cerita dan gambaran yang diberikan oleh klien.

Setelah berbagai unit tes selesai dibangun, tim barulah melanjutkan aktivitasnya kepenulisan *coding* aplikasi. XP menerapkan konsep *Pair Programming* dimana setiap tugas sebuah modul dikembangkan oleh dua orang programmer. XP beranggapan, 2 orang akan lebih cepat dan baik dalam menyelesaikan sebuah masalah. Selanjutnya, modul aplikasi yang sudah selesai dibangun akan digabungkan dengan aplikasi utama.

4. *Testing*

Walaupun tahapan uji coba sudah dilakukan pada tahapan *coding*, XP juga akan melakukan pengujian sistem yang sudah sempurna. Pada tahap *coding*, XP akan terus mengecek dan memperbaiki semua masalah-masalah yang terjadi walaupun hanya masalah kecil. Setiap modul yang sedang dikembangkan, akan diuji terlebih dahulu dengan modul unit tes yang telah dibuat sebelumnya.

Setelah semua modul selesai dan dikumpulkan ke dalam sebuah sistem yang sempurna, maka tim XP akan melakukan pengujian penerimaan atau *acceptance test*. Pada tahap ini, aplikasi akan langsung diuji coba oleh *user* dan klien agar mendapat tanggapan langsung mengenai penerapan gambaran dan cerita yang telah dilakukan sebelumnya.

2.3 Pengertian Sistem Informasi

Menurut (Arifin, 2020) mengatakan bahwa Sistem dalam kamus Webster New Collegiate Dictionary menyatakan bahwa kata “syn” dan “Histanai” berasal dari bahasa Yunani, artinya menempatkan bersama. Sehingga menurut Arifin Rahman bahwa Pengertian Sistem adalah sekumpulan beberapa pendapat (Collection of opinions), prinsip-prinsip, dan lain-lain yang telah membentuk satu

kesatuan yang saling berhubungan antar satu sama lain. Informasi adalah data yang diolah menjadi bentuk lebih berguna dan lebih berarti bagi yang menerimanya. Informasi juga disebut data yang diproses atau data yang memiliki arti.

Adapun pengertian sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan data transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi serta menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. (Listiyono, H., Sani, D. L., Khristianto, T., & Soelistijadi, R.2022)

2.4 Pengertian Website dan Database

Dalam www.sekawanmedia.co.id, pengertian website adalah kumpulan halaman dalam suatu domain yang memuat tentang berbagai informasi agar dapat dibaca dan dilihat oleh penggunaatau pemakaiinternet melalui sebuah mesin pencariatau search engine. Informasi yang dapat dimuatpada website biasanyaberisi mengenai konten gambar, ilustrasi, video, dan teks untuk berbagai macam kepentingan (Fitriani, Y., Utami, S., & Junadi, B.2022).

Basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematis sehingga dapat diperiksa menggunakan program komputer untuk mendapatkan informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan melakukan kueri basis data disebut sistem manajemen basis data (DBMS) (Efendi, E., Amin, A., Salim, M., & Rosadi, I.2023).

2.5 Pengujian Sistem

Pengujian software merupakan suatu proses menjalankan program dengan maksud menemukan kesalahan. Definisi tersebut menyangkut kegiatan mulai dari cek kode yang dilakukan oleh seorang pemimpin tim untuk menjalankan percobaan dari perangkat lunak yang dilakukan oleh seorang rekan, serta tes yang dilakukan oleh suatu unit pengujian, semua bias dianggap kegiatan pengujian. (Dhaifullah, I. R., Salsabila, A. A., & Yaqin, M. A.2022).

2.6 *Unifield Modeling Language* (UML)

Menurut Rosa dan Shalahuddin mendefinisakan *Unified Modeling Language* (UML) adalah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analis dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek (Aisyah, N., Asep, A., & Yoraeni, A.2019).

2.6.1 *Diagram Use Case*

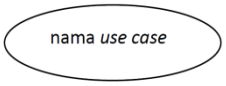
Menurut Rosa dan Shalahuddin mendefinisakan *diagram usecase* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. (Aisyah, N., Asep, A., & Yoraeni, A.2019).

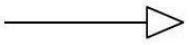
Syarat penamaan pada *usecase* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami (Rosa dan Shalahuddin dalam Ajusta, A. G., Addin, S., & Nurofiq, M. (2019). Ada dua hal utama pada *usecase* yaitu pendefinisian apa yang disebut aktor dan *usecase*:

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Usecase* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Rosa dan Shalahuddin (Riski, A. 2022), menjelaskan simbol-simbol *usecase* atau *diagram usecase* yang ditampilkan pada tabel 2.1 berikut.

Tabel 2.1 Simbol *Diagram Use Case*

No.	Simbol	Keterangan
1.	*Usecase* 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *usecase*.

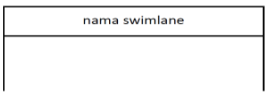
2.	Aktor /*aktor*  nama aktor	Orang, proses, atau sistem lain yang Berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal *frase* nama aktor.
3.	Asosiasi / *association* 	Komunikasi antar aktor dan *usecase* yang berpartisipasi pada *usecase* atau *usecase* memiliki interaksi dengan aktor
4.	Ekstensi / << *extend* >> 	Relasi *usecase* tambahan ke sebuah *usecase* dimana *usecase* yang ditambahkan dapat berdiri sendiri walau tanpa *usecase* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *usecase* tambahan memiliki nama depan yang sama dengan *usecase* yang ditambahkan, misal arah panah mengarah pada *usecase* yang ditambahkan; biasanya *usecase* yang menjadi extend-nya merupakan jenis yang sama dengan *usecase* yang menjadi induknya.
5.	Generalisasi/ generalization 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *usecase* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya: Arah panah mengarah pada *usecase* yang menjadi generalisasinya (umum)
6.	Ekstensi / << include >> 	Relasi *usecase* tambahan ke sebuah *usecase* di mana *usecase* yang ditambahkan memerlukan *usecase* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *usecase*.

(Rosa dan Shalahuddin, 2019)

2.6.2 *Diagram* Aktivitas

Rosa dan Shalahuddin menyatakan bahwa *diagram* aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak (Pramono, S., Ahmad, I., & Borman, R. I. 2020). Menuru Rosa dan Shalahuddin (dalam Ramdhani, M. R. (2022) menjelaskan simbol-simbol *diagram* aktivitas yang ditampilkan pada tabel 2.2 berikut.

Tabel 2.2 Simbol *Diagram* Aktivitas

No.	Simbol	Keterangan
1.	Status awal 	Status awal aktivitas sistem, sebuah *diagram* aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	Status akhir 	Status akhir yang dilakukan sistem, sebuah *diagram* aktivitas memiliki sebuah status akhir.
6.	Swimlane 	Swimlane memisahkan organisasi bisnis yang bertanggungjawab terhadap aktivitas yang terjadi.

(Ramdhani, M. R. 2022)

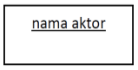
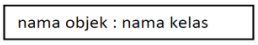
2.6.3 *Diagram* Sekuensial

Rosa dan Shalahuddin mendefinisakan *diagram* sekuensial menggambarkan kelakuan objek pada *usecase* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirim dan diterima antar objek. (Aisyah, N., Asep, A., & Yoraeni, A.2019).

Banyaknya *diagram* sekuensial yang harus digambar adalah minimal sebanyak pendefinisian *usecase* yang memiliki proses sendiri atau yang penting semua *usecase* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *diagram* sekuensial sehingga semakin banyak *usecase* yang didefinisikan maka *diagram* sekuensial yang harus dibuat juga semakin banyak (Rosa dan Shalahuddin dalam Kristina, K., Willay, T., & Lahirsa, A. A. 2022).

Rosa dan Shalahuddin (dalam Aisyah, N., Asep, A., & Yoraeni, A.2019) menjelaskan simbol-simbol *diagram* sekuensial yang ditampilkan pada tabel 2.3 berikut.

Tabel 2.3 Simbol *Diagram* Sekuensial

No.	Simbol	Keterangan
1.	Aktor  nama aktor  tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
2.	Garis hidup / lifeline 	Menyatakan kehidupan suatu objek.
3.	Objek  	Menyatakan objek yang berinteraksi pesan.
4.	Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.

(Aisyah, N., Asep, A., & Yoraeni, A.2019)

2.6.4 *Diagram* Kelas

Menurut Rosa dan Shalahuddin mendefinisakan *diagram* kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun system (Sumadi, G., & Ruslan, R. 2023).

Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

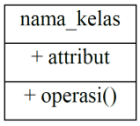
Susunan struktur kelas yang baik pada *diagram* kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main, Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2. Kelas yang menangani tampilan sistem *(view)*, Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *usecase (controller)*, Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *usecase*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
4. Kelas yang diambil dari pendefinisian data *(model)*, Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Rosa dan Shalahuddin (2019) menjelaskan simbol-simbol *diagram* kelas yang ditampilkan pada tabel 2.4 berikut.

Tabel 2.4 Simbol *Diagram* Kelas

No.	Simbol	Keterangan
1.	Kelas 	Kelas pada struktur sistem
2.	Antarmuka / *interface*  nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
3.	Asosiasi / *association* 	Asosiasi / *association* Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
4.	Asosiasi berarah / *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity.
5.	Generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus).
6.	Kebergantungan / *dependency* 	Relasi antarkelas dengan makna kebergantungan antarkelas.

7.	Agregasi / *Aggregation* 	Relasi antarkelas dengan makna semua bagian *(whole-part).*

(Rosa dan Shalahuddin, 2019)

2.7 Alat Pengembangan *Web*

2.7.1 XAMPP

Menurut Nugroho (dalam Chang, F. D.2022) XAMPP adalah paket program *web* lengkap yang dapat dipakai untuk belajar pemograman *web*, khususnya PHP dan MySQL. XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache* HTTP *Server*, MySQL *database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl.

Nugroho Bagian penting dari XAMPP yang diasa digunakan (Chang, F. D.2022):

1. Htdoc adalah *folder* tempat meletakkan berkas-berkas yang akan dijalankan, seperti berkas PHP, HTML dan *skript* lain.
2. Phpmyadmin merupakan bagian untuk mengelola basis data mysql yang ada dikomputer. Untuk membukanya, buka *browser* lalu ketikkan alamat http://*localhost*/phpmyadmin, maka akan muncul halaman phpmyadmin.
3. Kontrol Panel yang berfungsi untuk mengelola layanan (*service*) XAMPP. Seperti menghentikan (*stop*) layanan, ataupun memulai (*start*).

2.7.2 MySQL

Abdul kadir (dalam Rachmawati, R. N. 2021) menyatakan MySQL adalah nama *database server*. *Database server* adalah *server* yang berfungsi untuk menangani *database*. *Database* adalah suatu pengorganisasian data dengan tujuan memudahkan penyimpanan dan pengaksesan data. Dengan menggunakan MySQL, kita bisa menyimpan data dan kemudian data bisa diakses dengan dengan cara mudah dan cepat. MySQL tergolong sebagai *database* relasional.

pada model ini, data dinyatakan dalam bentuk dua dimensi yaitu secara khusus dinamakan tabel, tabel tersusun atas baris dan kolom.

2.7.3 *Web Server Apache*

Menurut Nugroho (dalam Alakel, W.,Ahmad,I.,& Santoso, E.B.2019) *Apache* merupakan aplikasi *web server*. Tugas utama apache adalah menghasilkan halaman *web* yang benar kepada *user* berdasarkan kode PHP yang dituliskan oleh pembuat halaman *web*.

2.7.4 HTML

Menurut Nugroho (dalam Alakel, W.,Ahmad,I.,& Santoso, E.B.2019) HTML adalah bahasa dasar yang digunakan untuk menyusun halaman *web*. Keberadaannya tetap diperlukan walaupun muncul bahasa seperti PHP ataupun JSP. PHP dan HTML dipakai secara bersama-sama. Dalam hal ini, posisi skrip PHP adalah melekat pada dokumen HTML. Dengan demikian, didokumen HTML bisa disisipkan skrip PHP. Namun, konsekuensinya, dokumen HTML harus disimpan dengan ektensi berupa (dot/.)php.

Berikut Struktur dasar dokumen HTML adalah sebagai berikut:

```
<!doctype html>
<!DOCTYPE html>
<html>
       <head>
              <title> ..... </title>
       </head>
       <body>.....</body>
</html>
```

Di dalam dokumen HTML, tanda < > menyatakan *tag*. *Tag* menyatakan elemen dalam dokumen HTML. Umumnya *tag* berpasangan. Contoh, <head> berpasangan dengan </head>, dan <body> dengan </body>. Namun, ada pula *tag* yang tidak berpasangan. Sebagai contoh,
 dn <hr> tidak memiliki pasangan. Penjelasan untuk *tag-tag* HTML pada contoh sebagai berikut:

1. Pasangan <html>...</html> menyatakan awal dokumen HTML.
2. Di dalam <html>...</html> terdapat pasangan <head>...</head> dan <body>...</body>.
3. Pasangan <head>...</head> menyatakan bagian judul dokumen HTML. Isinya paling tidak berupa pasangan <title>...</title>.

4. Isian yang berada pada <title>...</title> menentukan judul dalam *browser.*
5. Pasangan <body>...</body> menyatakan bagian tubuh dokumen. Bagian ini bisa berbagai tag misalnya ... atau <h1>...</h1>.

2.7.5 PHP

Menurut Nugroho (dalam Alakel, W.,Ahmad,I.,& Santoso, E.B.2019) mendefinisikan PHP (*Hypertext Preprocessor*) itu bahasa pemograman berbasis *web*. Jadi, PHP itu adalah bahasa program yang digunakan untuk membuat aplikasi berbasis *web*. PHP termasuk bahasa program yang bisa bisa berjalan di sisi *server*, atau sering disebut *Side Server Language*. Jadi, program yang dibuat dengan kode PHP tidak bisa berjalan kecuali dia dijalankan pada *server web*, tanpa adanya *server web* yang terus berjalan dia tidak akan bisa dijalankan.

Sedangkan, Menurut Kadir (dalam Cavanaugh, A. B. 2021) mendefinisikan PHP merupakan bahasa pemrograman yang ditunjuk untuk membuat aplikasi *web.* Ditinjau dari pemrosesannya, PHP tergolong berbasis *server side*. Artinya, pemrosesan dilakukan di *server.* Hal ini berkebalikan dengan bahasa seperti *JavaScript*, yang pemrosesannya dilakukan di sisi klient (*client side*). PHP sering dikatakan bahasa untuk membangun aplikasi *web* dinamis. Pengertian dinamis di sini adalah memungkinkan untuk menampilkan data yang tersimpan dalam *database*. Dengan demikian, halaman *web* akan menyesuaikan dengan isi *database.*

2.7.5.1 *JavaScript*

Menurut Kadir (dalam Vincent, T., & Sukoco, S. 2020) mendefinisikan *JavaScript* merupakan bahasa pemrogaman yang penggunaannya dilekatkan di dokumen HTML. Kode ditulis di dalam pasangan *tag* <*script*> <*/script* >. Berbeda dengan kode PHP yang diproses di *server*, kode *JavaScript* diproses di sisi klient. Karena sifat tersebut, *JavaScript* dapat digunakan misalnya untuk melakukan validasi data di formulir di sisi klien sehingga tidak terjadi komunikasi bolak-balik dengan *server*. Di samping itu, *JavaScript* melalui pendekatan AJAX dapat digunakan untuk mengubah isi sebagian area di halaman *web* berdasarkan data yang diperoleh dari *server*.

Apabila sudah terbiasa dengan PHP, mempelajari *JavaScript* tidaklah sulit karena ada kemiripan-kemiripan. Akan tetapi, hal yang terpenting untuk disadari, jangan sampai mencampuradukkan kode PHP dan *JavaScript* karena keduanya merupakan bahasa yang berbeda. *JavaScript* tidak memerlukan peranti khusus. *Browser* telah menyediakan pemroses *JavaScript*. Oleh karena itu, pengguna dapat langsung membuat kode *JavaScript* tanpa memerlukan *software* tambahan (Kadir dalam Hermawanto, A. B. 2020)

2.7.6 CSS

Menurut Kadir (dalam Gunawan, R., Yudiana, Y., & Apriansyah, W. Y. 2021) *Cascading Style Sheets* (CSS) digunakan untuk mengatur tampilah halaman *web*. Banyak hal yang bisa ditangani oleh CSS, dari mengatur bingaki elemen HTML, penawaran latar belakang yang bergradasi, pembuatan bayangan pada elemen HTML, pengaturan teks, hingga pembuatan menu. Akan tetapi, halaman *web* yang menarik tentu saja tidak sekedar dibentuk dengan CSS, namun juga dipadu dengan kode *JavaScript* atau *jQuery* untuk mendapatkan efek-efek tertentu. Dapat dikatakan bahwa hampir semua halaman *web* turut melibatkan CSS. Oleh karena itu, memahami CSS perlu dilakukan bagi pengembang *website*.

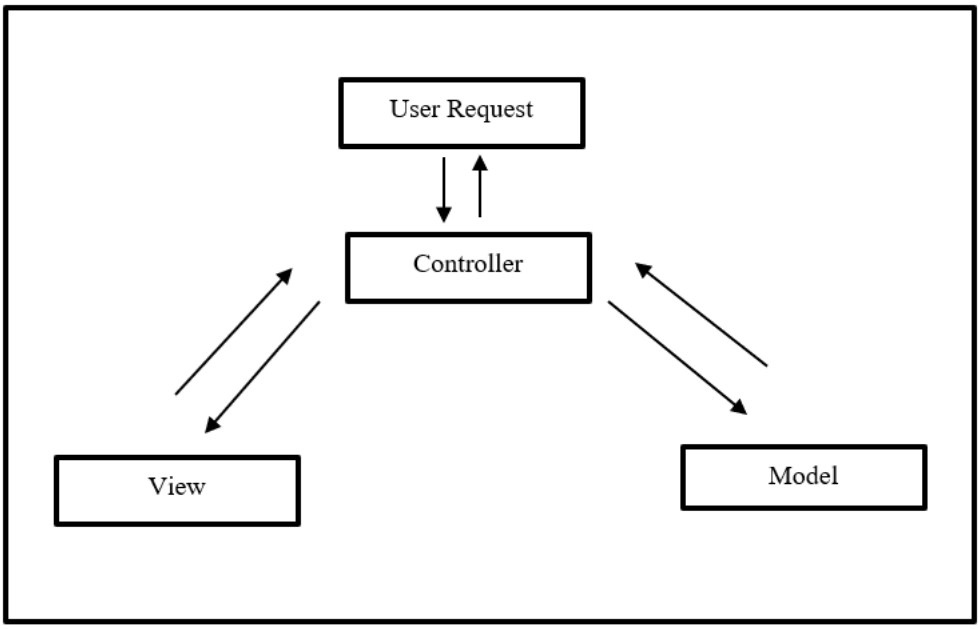
2.7.7 *Framework* CodeIgniter

Menurut Basuki mendefinisikan *framework* codeigniter adalah sebuah *framework* PHP yang dapat membantu mempercepat developer dalam pengembangan aplikasi *web* berbasis PHP dibanding jika menulis semua kode program dari awal (Vincent, T., & Sukoco, S.2020).

2.7.7.1 *Model-View-Controller* (MVC)

Menurut Basuki (dalam Vincent, T., & Sukoco, S. 2020) mendefinisikan *model-view-controller* adalah *framework* PHP yang dibuat berdasarkan kaidah *model-view-controller*. Dengan MVC, maka memungkinkan pemisahan antara *layer application-logic* dan *presentation*. Sehingga, dalam sebuah pengembangan *web*, seorang *programmer* bisa berkonsentrasi pada *core-system*, sedangkan *web designer* bisa berkonsentrasi pada tampilan *web*. Menariknya, skrip PHP, *query* MySQL, Javascript dan CSS bisa saling terpisah, tidak dibuat dalam satu skrip berukuran besar yang membutuhkan *resource* besar pula untuk mengesekusinya.

Adapun alur program aplikasi berbasis *framework* codeigniter dapat dilihat pada gambar 2.1 berikut.



GAMBAR 2 2 *MODEL-VIEV-CONTROLLER* (VINCENT, T., & SUKOCO, S. 2020)

Gambar 2.1 *Model-Viev-Controller* menerangkan bahwa ketika datang sebuah *user request,* maka akan ditangani oleh *controller,* kemudian *controller* akan memanggil *model* jika memang diperlukan operasi *database*. Hasil dari *query* oleh *model* kemudian akan dikembalikan ke *controller.* Selanjutnya *controller* akan memanggil *view* yang tepat dan mengkombinasikannya dengan hasil *query model.* Hasil akhir dari operasi ini akan ditampilkan di *browser* (Basuki, 2010).

Dalam konteks codeigniter dan aplikasi berbasis *web,* maka penerapan konsep MVC mengakibatkan kode program dapat dibagi menjadi tiga kategori, yaitu:

1. *Model,* kode program (berupa OOP *class*) yang digunakan untuk memanipulasi *database.*
2. *View,* berupa *template* html/xml atau php untuk menampilkan data pada browser.
3. *Controller,* kode program (berupa OOP *class*) yang digunakan untuk mengontrol aliran aplikasi (sebagai pengontol *model* dan *view*).