BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar

2.1.1 Rancang Bangun

Rancang bangun merupakan sebuah istilah yang biasa dipakai pada saat membangun atau merancang sebuah objek dari tahapan perancangan awal hingga tahapan penyelesaian. Tahapan yang ada dalam rancang bangun biasanya terdiri dari perencanaan, penggambaran, desain dan penggabungan dari elemen-elemen terpisah yang dijadikan menjadi satu kesatuan agar dapat berfungsi dengan baik (Wulandari, S., et al. 2021). Dengan demikian rancang bangun berarti kegiatan mengimplementasikan rancangan yang sudah dilakukan menjadi bentuk *software* atau aplikasi guna memperbaiki sistem yang sudah ada atau membuat sebuah sistem baru.

2.1.2 Sistem Informasi

Sistem didefinisikan sebagai sekumpulan bagian yang saling berkaitan satu sama lain guna menyelesaikan sebuah tujuan tertentu. Elemen-elemen dalam sistem dapat berinteraksi, saling ketergantungan, dan saling bekerja sama menjadi sebuah kelompok. (Issa Dyah Utami, 2021). Selain itu, sistem dapat didefinisikan sebagai sekumpulan bagian yang saling berhubungan secara fisik dan non-fisik untuk mencapai sebuah tujuan (Dedy Rahman, 2020).

Informasi adalah beberapa data yang sudah diolah dan menjadikannya berguna bagi orang yang menerima serta bisa membantu mereka pada saat ingin membuat keputusan. Informasi dapat dikatakan sebagai sebuah bentuk asli dari pengetahuan. Informasi dianggap sebagai sebuah bahan mentah yang masih memerlukan pemrosesan lebih lanjut dan informasi dapat diidentifikasi sebagai benda yang nyata melalui rangkaian simbol yang dapat ditangkap dan diproses oleh pancaindra yang dimiliki manusia. (Wahyudi Agustiono, et al., 2022)

Sistem informasi adalah kombinasi aktivitas manusia dan teknologi informasi yang mencakup pengumpulan, pemrosesan, penyimpanan, dan distribusi informasi yang berguna untuk membantu pengambilan keputusan, koordinasi, pengendalian, dan analisis bisnis. (Turban et al., 2018). Sistem informasi bisa dikatakan sebagai sebuah perantara yang dapat menggabungkan teknologi, manusia, proses bisnis, dan struktur organisasi untuk memudahkan control organisasi, koordinasi dan pengambilan keputusan. Sistem informasi bahkan dapat berguna bagi organisasi dalam menganalisis dan mengevaluasi kinerja bisnis serta mengidentifikasi peluang dan tantangan baru (Laudon, 2019)

Berikut merupakan beberapa klasifikasi sistem informasi berdasarkan fungsinya:

- TPS (*Transaction processing system*)
  Sistem informasi yang dapat memproses data transaksi bisnis mulai dari memproses serta mencatat data hingga dapat memperoleh informasi yang bermanfaat.
- MIS (*Management Information System*)
  Sistem yang terintegrasi dimana terdapat manusia dan mesin yang dapat memberikan informasi dengan cara yang membantu proses operasi, manajemen, dan pengambilan keputusan di dalam sebuah perusahaan.
- DSS (*Decision Support System*)
  Sistem informasi yang memberikan informasi untuk membantu dalam membuat keputusan atau mengidentifikasi peluang untuk membuat keputusan.
- EIS (*Executive Information System*)
  Sistem informasi yang dapat memberi tahu para eksekutif tentang kinerja perusahaan secara keseluruhan
- Sistem pakar (*Expert System*)
  Sistem informasi yang menggunakan keahlian seorang pakar dan mengimplementasikan keahlian tersebut agar orang yang tidak ahli dalam bidang tersebut dapat menggunakannya.

2.1.3 Aplikasi

Aplikasi didefinisikan sebagai pemecah masalah yang menggunakan sebuah teknik pemrosesan data pada smartphone atau komputerisasi yang diinginkan. Kata "Aplikasi" merupakan kata yang awalnya berasal dari bahasa inggris "*Application*" artinya kata kerja untuk menggunakan atau pengolah dalam bahasa Indonesia. Sedangkan aplikasi komputer dapat dikatakan sebagai sebuah subkelas dari *software* komputer yang menggunakan kemampuan komputer agar dapat membantu pekerjaan manusia. Secara sederhana, aplikasi komputer dapat didefinisikan sebagai software yang dapat melakukan berbagai jenis tugas dan pekerjaan tertentu. (Roni Habibi, 2020).

2.1.4 Pelaporan

Pelaporan adalah sebuah proses penyampaian hasil pekerjaan atau penjelasan terkait seluruh hal yang bersangkutan dengan pekerjaan dan tanggung jawab terhadap seseorang yang memiliki kuasa lebih tinggi. Laporan dapat disampaikan baik secara tertulis maupun lisan sehingga orang yang menerima laporan dapat memahami bagaimana seluruh pekerjaan tersebut dilakukan. Dengan kata lain pelaporan dapat berarti proses penyampaian informasi yang berisi fakta mengenai suatu kegiatan baik secara lisan maupun tulisan.

2.1.5 Beasiswa

Beasiswa biasa disebut sebagai sebuah bantuan finansial yang diberikan kepada individu dengan maksud untuk membantu mereka dalam melanjutkan pendidikan (Saputra, R. A., & Ayuningtias, S., 2016). Beasiswa dapat diberikan oleh suatu lembaga kepada seseorang atau kelompok yang memenuhi persyaratan tertentu untuk membiayai penelitian akademik. Beasiswa didefinisikan juga sebagai sebuah bentuk dukungan finansial untuk individu atau kelompok yang memenuhi kriteria tertentu sebagai upaya untuk memperluas kesempatan pendidikan, memperkaya pengalaman akademik, dan membantu mewujudkan cita-cita atau tujuan karir (Magdalena H. Lim dan Windya Sari, 2018).

Didasarkan pada penjelasan sebelumnya beasiswa adalah bentuk dukungan keuangan yang diberikan oleh suatu lembaga, organisasi, atau pihak lain kepada individu yang memenuhi syarat tertentu, seperti prestasi akademik, bakat, atau kebutuhan finansial, untuk membiayai pendidikan atau riset. Beasiswa bisa diberikan untuk bermacam tingkatan pendidikan mulai dari tingkatan terendah seperti SD hingga tingkat universitas. Tujuannya adalah guna membantu individu menggapai tujuan akademik dan karir mereka, serta memajukan pengembangan serta penelitian di berbagai bidang ilmiah yang akan berguna bagi kehidupan manusia.

2.1.6 Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) didefinisikan sebagai sebuah tahapan untuk memahami bagaimana sistem informasi (SI) dapat membantu suatu bisnis atau organisasi dengan melakukan analisis, pengembangan, dan mengimplementasikannya agar dapat digunakan oleh pemangku kepentingan atau pengguna. SDLC mempunyai empat tahapan yaitu perencanaan, analisis, desain, dan implementasi. Beberapa proyek terkadang berfokus pada bagian SDLC yang berbeda atau menggunakan pendekatan fase SDLC yang tidak serupa, namun setiap proyek pasti memiliki komponen berupa keempat tahapan tersebut (Alan Dennis, et al., 2019).

Berikut merupakan penjelasan dari setiap fase yang ada dalam tahapan SDLC:

1. **Perencanaan (*Planning*)**

   Fase perencanaan merupakan sebuah tahapan penting untuk mengetahui kenapa sistem informasi dibuat dan cara apa yang harus dilakukan oleh tim proyek untuk dapat membuatnya. Tahapan perencanaan menentukan cara sistem dapat memenuhi kebutuhan dan menghasilkan nilai bagi perusahaan.

2. ***Analysis***

   Dalam fase analisis, jawaban akan diberikan beberapa pertanyaan seperti apa saja yang akan bisa sistem lakukan, kapan

sistem akan dipakai dan siapa yang memakai sistem. Selama tahap ini, tim proyek memeriksa sistem saat ini untuk menemukan area yang dapat diperbaiki dan membuat ide untuk sistem baru.

3. ***Design***

Fase *design* merupakan tahapan dimana pengembang memutuskan bagaimana caranya sistem dapat beroperasi. Pada fase ini akan ditentukan komponen *user interface, forms*, rancangan database dan laporan yang diperlukan. Walaupun hampir seluruh keputusan strategis terkait sistem sudah dibuat selama tahap analisis, tetapi bagaimana sistem akan berfungsi dengan benar akan ditentukan oleh tindakan yang diambil selama tahap desain.

Seleuruh rancangan termasuk spesifikasi sistem, desain arsitektur dan desain antarmuka. Spesifikasi sistem yang diberikan kepada tim pengembang untuk diimplementasikan adalah spesifikasi *database* dan desain program. Setelah mencapai tahap akhir desain akan dilakukan pengecekan Kembali mengenai rencana proyek. Setelah itu, keputusan akan dibuat tentang apakah proyek akan dilanjutkan atau tidak.

4. ***Implementation***

Proses implementasi SDLC adalah fase terakhir, dimana sistem sebenarnya akan dibuat. Karena bagi kebanyakan sistem yang ada tahap implementasi merupakan tahapan SDLC yang paling membutuhkan banyak waktu dan paling mahal. Dalam kebanyakan kasus proses implementasi harus lebih diperhatikan oleh tim pengembang dibanding tahapan lainnya.

Pengembangan sistem adalah langkah pertama dalam tahapan *implementation*, pengembangan dilakukan sesuai dengan rancangan yang dibuat pada tahapan sebelumnya. Setelah itu, dilakukan pengujian guna memastikan sistem yang diimplementasikan sudah sesuai dengan desain yang dibuat.

Proses ini merupakan langkah penting pada saat implementasi, karena kerugian yang disebabkan oleh kesalahan dapat sangat besar.

Setelah melakukan uji coba maka tahapan selanjutnya adalah instalasi dimana sistem lama diganti menjadi sistem baru. Proses terakhir dari implementasi adalah membuat program pelatihan untuk mendidik pengguna tentang cara mengoperasikan sistem baru dan membantu mereka menangani perubahan yang diakibatkan adanya sistem baru.

2.1.7 Waterfall

Metodelogi *Waterfall* didefinisikan sebagai rentetan aktivitas proyek dalam SDLC (*Software Development Life Cycle*) yang dikerjakan dalam bentuk linear dan berurutan. Fase-fase di dalamnya tergantung dari hasil dan dokumentasi pada fase sebelumnya. Selain itu, model ini membagi tugas-tugas yang jelas dalam seluruh fasenya. Kelebihan dari Model Waterfall termasuk kemudahan dalam pemahaman karena strukturnya yang linear, dokumentasi yang lengkap setiap fase, dan perencanaan yang dapat diandalkan karena semua langkah telah ditentukan sejak awal. Jika proyek memiliki kebutuhan yang stabil serta ketentuan yang tidak berubah sepanjang pengembangan, maka metodelogi ini sangat tepat untuk digunakan.

Metodelogi *waterfall* memang memiliki banyak kelebihan namun tetap terdapat beberapa kelemahan pada saat menggunakan metodelogi ini diantaranya keterbatasan untuk melakukan perubahan setelah memasuki fase tertentu dapat menjadi tantangan, dan kurangnya fleksibilitas dalam menanggapi perubahan kebutuhan pengguna atau perubahan dalam proyek dapat menyulitkan. Oleh karena itu, dalam proyek-proyek yang lebih kompleks atau yang memerlukan adaptasi cepat terhadap perubahan, model pengembangan lain seperti model Agile mungkin lebih cocok (Rahmat Kurniawan, 2023).

2.1.8 Analisis Perancangan

Istilah analisis perancangan biasanya digunakan untuk menggambarkan tahap awal dalam pembuatan sistem. Ananlisis perancangan berguna untuk memahami sistem yang nantinya akan diterapkan untuk merancang sistem informasi berbasis komputer. Tujuan dari proses ini adalah guna membangun sistem baru yang bisa menyelesaikan masalah perusahaan dengan membuat sistem alternatif terbaik. (Indyah Hartami, 2020).

Terdapat dua pendekatan logika yang umumnya digunakan dalam melakukan analisis perancangan yaitu:

- **Structured Systems Analysis and Design (SSAD)**

  SSAD merupakan sebuah method resmi pertama yang diterapkan dalam menganalisis sistem informasi. Method ini berfokus terhadap jalannya data yang berasal dari proses bisnis. Dalam method ini diharuskan untuk membuat paling tidaknya sebuah *data flow diagram* agar dapat dipakai untuk memberikan ilustrasi dari proses bisnis yang diterapkan saat ini dan juga yang akan diterapkan pada saat sistem baru selesai dibuat (Muhamad Muslihudin, 2016).

- **Object Oriented Analysis and Design (OOAD)**

  OOAD muncul guna mengelompokkan serta menghilangkan pemisahan data dan proses yang ada pada pendekatan lainnya. Pendekatan ini lebih spesifik dalam melakukan *create, read, update, delete* (CRUD) karena data diintegrasikan menjadi bentuk objek. Model OOA merupakan sebuah ilustrasi yang menggambarkan komponen sistem berdasarkan sudut pandang contohnya perilaku, hubungan tiap objek dan struktur. Model OOA yang cukup banyak dipakai diantaranya UML (*Unified Modelling Language*) (Muhamad Muslihudin, 2016).

2.1.9 Unified Modelling Language (UML)

Unified Modeling Language atau biasa disebut UML dikembangkan oleh *Object Management Group* (OMG). UML digunakan untuk menggambarkan dan merepresentasikan berbagai aspek dalam sebuah sistem, seperti struktur kelas, diagram *use case*, *activity* diagram, dan lain-lain. Pengembang perangkat lunak menggunakan UML untuk membantu proses analisis, desain, dan dokumentasi sistem. UML didefinisikan sebagai bahasa pemodelan yang terstandarisasi dan dapat mengilustrasikan berbagai jenis sistem, seperti sistem perangkat lunak dan sistem bisnis.

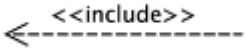
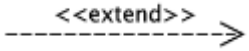
UML termasuk sebagai bahasa pemodelan yang sederhana dan fleksibel, sehingga dapat digunakan dalam berbagai konteks dan dapat disesuaikan dengan kebutuhan pengguna. UML juga dapat digunakan oleh berbagai stakeholder dalam pengembangan sistem, termasuk pengembang, analis, manajer proyek, dan pengguna akhir. Ada banyak diagram didalam UML diantaranya.

1. **Use Case Diagram**

   Jenis diagram dalam UML (*Unified Modeling Language*) yang disebut *use case* diagram digunakan untuk menunjukkan interaksi antara sistem dan pengguna atau antara sistem dengan sistem lain. *Use case* menggambarkan dengan jelas terkait fungsi-fungsi sistem dan bagaimana sistem berinteraksi dengan pengguna atau aktor lain.

   Tahap analisis dan desain biasanya menggunakan diagram ini untuk membantu dalam memahami kebutuhan fungsional sistem dan untuk memfasilitasi komunikasi antara para pemangku kepentingan. Diagram ini juga dapat membantu dalam mengidentifikasi kebutuhan bisnis dan dalam memastikan bahwa sistem memenuhi kebutuhan tersebut. berikut adalah simbol yang terdapat pada diagram tersebut.

Tabel 2.1 Simbol Activity Diagram

Simbol	Nama	Keterangan
	Actor	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan use case.
	Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) Dependency akan mempengaruhi elemen. yang bergantung padanya elemen yang tidak mandiri.
	Generalzation	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur daya dari objek yang ada diatasnya objek induk (ancestor).
	Include	Menspesifikasikan bahwa use case sumber secara eksplisit.
	Extend	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
	Association	yang menghubungkan antara objek satu dengan objek lainnya.
	System	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
	Use Case	Deskripsi dari uruan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	Collaboration	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah

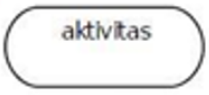
Simbol	Nama	Keterangan
		dan elemen-elemennya (sinergi).
	Note	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

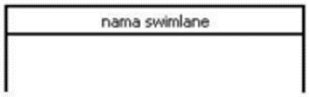
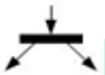
2. Activity Diagram

Salah satu jenis diagram yang tersedia dalam UML (Unified Modeling Language) adalah activity diagram yang umumnya dipakai untuk memberikan gambaran urutan tindakan atau kegiatan pada sebuah proses atau aliran kerja (*workflow*). Diagram ini memberikan gambaran yang jelas mengenai alur kerja sistem dan interaksi antara objek atau entitas dalam sistem.

Diagram ini berguna dalam menggambarkan proses dan sistem kompleks. Diagram ini bisa membantu untuk mengidentifikasi masalah dalam alur kerja yang ada dan berguna dalam merancang sebuah solusi yang lebih efektif. *Activity* diagram juga sangat berguna sebagai perantara komunikasi antara para pemangku kepentingan atau *client* dengan pengembang dan memastikan bahwa sistem telah memenuhi kebutuhan mereka. berikut merupakan simbol pada diagram tersebut.

Tabel 2.2 Simbol Activity Diagram

Simbol	Nama	Keterangan
	Start	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
	Activity	Aktivitas yang dilakukan sistem dan biasanya diawali dengan kata kerja.
	Decision	Asosiasi atau percabangan merupakan kondisi dimana ada pilihanaktivitas lebih dari satu

Simbol	Nama	Keterangan
	Join	Join merupakan penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
	End	Status akhir yang dilakukan sistem, sebuah sistem diharuskan untuk memiliki sebuah status akhir
	Swimlane	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi
	Fork	Digunakan untuk menunjukkan kegiatan yang dilakukan secara pararel
	Join	Digunakan untuk menunjukkan kegiatan yang digabungkan

3. Sequence Diagram

Diagram pada UML yang digunakan untuk memodelkan interaksi antar *object* pada sebuah sistem adalah sequence diagram. *Sequence* terdiri dari objek-objek yang direpresentasikan oleh kotak vertikal dan pesan-pesan atau panggilan metode yang direpresentasikan oleh panah horizontal antara objek-objek tersebut. Urutan pesan atau panggilan metode ditunjukkan oleh urutan panah, dan waktu dieksekusinya ditunjukkan oleh sumbu waktu vertikal.

Sequence diagram sangat berguna dalam memodelkan interaksi antara objek dalam sistem, terutama dalam kasus yang melibatkan banyak objek yang berinteraksi secara kompleks. Dengan menggunakan *sequence* diagram, pengembang dapat memvisualisasikan urutan pesan atau panggilan metode antar objek dan menemukan bug atau kelemahan dalam desain sistem sebelum diimplementasikan. Diagram ini juga dapat digunakan sebagai dokumen referensi atau dokumentasi untuk membantu

pemeliharaan dan pengembangan sistem di masa depan. Berikut ini merupakan beberapa simbol yang terdapat di *sequence* diagram:

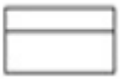
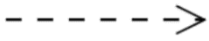
Tabel 2.3 Simbol Sequence Diagram

Simbol	Nama	Keterangan
	Objek	Menyatakan objek yang berinteraksi atau mengirimkan pesan dapat juga menjadi sebuah instance dari class dengan nama objek di dalamnya yang diawali titik koma.
	Aktor	Orang, proses atau sistem lain yang berinterkaksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang dibuat sendiri jadi walaupun aktor adalah gambar atau orang tetapi belum tentu merupakan orang.
	Lifeline	Menyatakan kehidupan suatu objek atau keberadaan sebuah object. Lifeline biasanya garis putus vertikal yang ditarik dari sebuah objek.
	Activation	Menyatakan objek dalam keadaan aktif dan berinteraksi dengan pesan, digambarkan dengan segi empat yang disimpan di lifeline yang menggambarkan suatu objek yang akan melakukan aksi.
	Message	Digambarkan berbentuk anak panah, massage menyatakan komunikasi antar objek.

4. Class Diagram

Class diagram digunakan dalam pemodelan yang ada pada sistem berdasarkan definisi class untuk membuat sistem. *Class* diagram biasanya menjadi titik awal untuk pemodelan berorientasi objek serta merepresentasikan struktur statis yang ada didalam sistem. Diagram ini digunakan untuk pemodelan konseptual umum dari sistem aplikasi dan untuk pemodelan detail yang akan diterjemahkan menjadi kode pemrograman. *Class* diagram juga dapat digunakan sebagai pemodelan data. Dibawah ini merupakan simbol yang terdapat didalam diagram tersebut:

Tabel 2.4 Simbol Class Diagram

Simbol	Nama	Keterangan
	Generalization	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
	Nary Association	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
	Class	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	Collaboration	Deskripsi dari urutan aksi aksi yang ditampilkan sistem yang menghasilkan sebuah hal terukur bagi suatu aktor.
	Realization	Operasi yang benar-benar dilakukan oleh suatu objek.
	Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen

Simbol	Nama	Keterangan
		yang bergantung padnya elemen yang tidak mandiri.
	Association	Yang menghubungkan antara objek satu dengan objek lainnya.

2.1.10 Website

Website yang juga dikenal sebagai "situs web" merupakan halaman web dengan konten text, video, gambar serta audio yang dapat diakses apabila sudah terhubung dengan koneksi internet. Dengan kata lain *website* adalah halaman *web* yang berisi konten yang bisa diakses dengan menggunakan *browser* dan memiliki kemampuan untuk memberikan informasi bermanfaat kepada penggunanya. (Rohi Abdulloh, 2016). *Website* berdasarkan sifatnya dapat dibedakan menjadi dua jenis (Muhammad Ibnu Sa`ad, 2020):

- **Web Statis**

  Konsep "web statis" mengacu pada web yang memiliki konten yang tidak dapat diubah. Dengan kata lain, konten yang terdapat pada halaman web tidak bisa diubah secara langsung dikarenakan bahasa pemrograman yang diimplementasikan tidak menyediakan fitur untuk melakukan perubahan data atau isi. Untuk web statis biasanya digunakan teknik skrip klien seperti HTML ataupun CSS. Untuk melakukan perubahan data atau halaman yang ada pada web statis maka harus melakukan perubahan langsung pada *script* atau *code* yang diimplementasikan dalam membangun website.

- **Web Dinamis**

  Web dinamis didefinisikan sebagai *web* dengan konten dan isi yang dapat diperbaharui setiap saat tanpa harus mengubah *script* atau *code* yang diimplementasikan guna membangun *website*. Setiap *web* dengan kemampuan menyuguhkan tampilan dinamis melalui halaman admin tanpa *flash* tidak seluruhnnya

bisa dikatakan *web* dinamis. Seringkali, web dinamis umumnya akan menyimpan data yang digunakan kedalam basis data MongoDB atau MySQL.

Jenis *website* yang memakai konsep *client-server* biasa disebut sebagai aplikasi *web*. Dalam konsep ini, aplikasi *client* tersambung ke *server* guna mendapatkan seluruh data yang diperlukan agar dapat menyelesaikan pekerjaan yang ditetapkan pengguna. Dalam melakukan pengembangan *web* terdapat istilah *Server-side* dan *Client-side Scripting* (Fitri Marisa, 2017).

- **Server-Side Scripting**

  Server-side Scripting didefinisikan sebagai bahasa pemrograman di mana data diproses oleh komputer server atau penyedia. Oleh karena itu, setiap kali seseorang mengakses *website server* akan mengambil data yang berasal dari *database server* kemudian ditampilkan pada halaman website. Karena *code* yang dipakai dalam server side bersifat rahasia untuk klien maka *code* yang ada di sisi server tidak dapat dilihat. Beberapa bahasa pemrograman dari *server-side Script* diantaranya ASP, Java, dan PHP.

- **Client-Side Scripting**

  Client-Side Scripting didefinisikan sebagai bahasa pemrograman yang mana data diproses oleh komputer milik *client* atau *user*. Oleh karena itu, komputer seseorang akan mengunduh *script client-side* ketika mereka mengunjungi *web*. Bahasa pemrograman *Client-side Script* adalah JavaScript, HTML dan CSS.

2.1.11 Database

Database juga dikenal sebagai basis data merupakan sekumpulan informasi atau data yang disimpan secara terstruktur serta terorganisir pada perangkat lunak tertentu atau sistem komputer. Tujuan utama dari penggunaan *database* adalah untuk menyimpan, mengelola, dan mengakses data secara lebih efisien. *Database* biasanya memiliki beberapa data yang

berkaitan dengan operasi bisnis atau organisasi tertentu. Setiap data dalam *database* biasanya memiliki atribut atau kolom yang merepresentasikan karakteristik atau properti tertentu dari data tersebut.

Database berisi data-data yang saling memiliki keterkaitan serta terorganisir secara logis. Data yang ada dalam database dapat berbentuk *table*, arsip dan file dengan keterkaitan satu sama lain dan seluruh data tersebut berada pada penyimpanan berbentuk digital. Basis data umumnya menjadi komponen penting dalam mendukung sistem yang terkomputerisasi. Semua program terkomputerisasi yang berkaitan dengan pengelolaan data harus menggunakan basis data untuk menampung seluruh data yang digunakan (Kadarsih, 2022).

Database Management System atau biasa disingkat dengan DBMS diperlukan pada saat aplikasi ingin memasukkan atau menarik data yang terdapat pada basis data. DBMS memungkinkan pengguna mengontrol, menjaga, dan menggunakan data secara efektif. DBMS berfungsi sebagai perantara antara user dan database sehingga sintaks atau perintah tertentu diperlukan untuk melakukan komunikasi antara mereka dan database (Tri Rachmadi, 2020). Berikut merupakan jenis dari perintah yang ada dalam database:

- *Data Definition Language* atau DDL yaitu bahasa pendefinisian data untuk membuat *table* baru, memasukkan *index* dan mengelola *table* yang sudah ada.
- *Data Manipulation Language* atau biasa disingkat DML berguna sebagai bahasa untuk memanipulasi data dalam *database* diantaranya menambah, menghapus, atau mengubah data.

Relational database adalah salah satu dari banyak jenis *database*, jenis ini merupakan database yang menyimpan data dalam bentuk table yang di mana kunci digunakan untuk menghubungkan beberapa tabel satu sama lain. Kunci-kunci ini memiliki nilai yang sebanding dengan nilai kolom yang ada di masing-masing tabel. Contoh dari database jenis ini adalah MySQL, Oracle dan SQL Server (Tri Rachmadi, 2020).

Dalam membuat sebuah relational database harus dilakukan normalisasi yang digunakan untuk menghilangkan kemungkinan adanya anomali yang terjadi pada saat melakukan pengelolaan data yang ada di dalam database. Metode normalisasi melibatkan penerapan sejumlah aturan untuk mengembangkan design logika *database* relasional, meskipun tidak langsung terkait dengan model data dan sebagai standar pada saat membuat struktur *table* yang konsisten (Abdul Kadir, 2020). Berikut merupakan bentuk dari normalisasi.

- **Bentuk Tidak Normal**

  Kumpulan data dalam bentuk ini dapat direkam tanpa harus mengikuti format tertentu; data dapat menjadi duplikat atau tidak lengkap. Data disini dapat langsung diambil tanpa melakukan perubahan.

- **1 Normal Form (1NF)**

  Tabel dianggap 1NF jika tidak ada data atau record yang memiliki nilai yang sama pada tabel sehingga setiap record hanya bernilai tunggal.

- **2 Normal Form (2NF)**

  Dalam bentuk 2NF terdiri dari semua record tanpa atribut primary key tetapi mempunyai ketergantungan terhadap primary key sepenuhnya.

- **3 Normal Form (3NF)**

  Tabel dianggap memenuhi bentuk (3NF) apabila seluruh ketergantungan fungsional yang ditunjukkan dengan notasi X -> A maka A harus menjadi bagian dari kunci utama *table* atau X harus menjadi *superkey* jika A mewakili seluruh *attribute* tunggal yang tidak terdapat pada X.

2.1.12 Black Box Testing

Black box testing didefinisikan sebagai sebuah cara menguji *software* namun hanya menguji fungsi yang dimiliki sistem tanpa mempertimbangkan implementasi atau strukturnya. Dalam konteks ini, "*black box*" merujuk pada pandangan sistem sebagai suatu *black box* yang mana proses uji dilakukan tanpa pengetahuan tentang bagaimana sistem mencapai hasil tersebut. Dengan kata lain pengujian ini hanya melihat apa yang terjadi pada bagian luarnya saja tanpa melihat lebih dalam bagaimana sebuah proses dalam sistem dapat terjadi pada bagian dalamnya (Romindo, et al., 2023).

Fokus utama dilakukannya pengujian ini adalah agar bisa mengetahui apakah aplikasi telah memenuhi keinginan dari *end user*. Pengujian dengan menggunakan pendekatan *black box testing* memungkinkan penguji untuk memiliki *subset test* secara efektif dan efisien. Pendekatan ini memungkinkan para penguji untuk menemukan sebuah *bug* atau kecacatan pada sebuah perangkat lunak agar bisa diperbaiki secara langsung. Penggunaan pendekatan ini juga memungkinkan dilakukannya *testing investment* yang lebih maksimal.

2.2 Tinjauan Studi

Laporan tugas akhir ini dilandasi oleh beberapa acuan dari jurnal sebelumnya yang relevan dengan subjek penelitian. Berikut merupakan jurnal-jurnal tersebut.

1. Jurnal pertama merupakan sebuah pengembangan sistem yang dilakukan oleh Dian Agustini, Muthia Farida, Mayang Sari, Muhammad Edya Rosadi dengan judul "**Rancang Bangun Sistem Informasi Beasiswa (Studi Kasus: UNISKA MAB Banjarmasin)**" yang dimuat dalam jurnal Technologia: Jurnal Ilmiah pada tahun 2022. Jurnal tersebut meneliti mengenai metode desain dan pembuatan sistem. beasiswa yang dapat menyelesaikan masalah dimana terjadinya penumpukan kertas yang disebabkan karena mahasiswa melakukan pengumpulan berkas persyaratan yang diminta oleh pihak kampus. Pada kasus ini kampus yang dibahas ialah Universitas Islam Kalimantan Muhammad Arsyad Al Banjari. Menurut penjelasan dari jurnal dapat diketahui bahwa pihak kampus masih melakukan proses penerimaan beasiswa secara manual dimana para mahasiswa diharuskan untuk mengumpulkan setiap berkas persyaratan dan mengisi sebuah form yang terkait dengan pendaftaran beasiswa. Didasarkan pada penelitian yang dilakukan oleh Dian Agustini dkk dimana mereka mengembangkan suatu sistem yang bisa mempermudah proses pengelolaan data beasiswa serta proses penerimaan beasiswa. Penelitian ini menggunakan model *waterfall* dikarenakan menurut peneliti model ini dipilih agar memudahkan proses pengembangan aplikasi dan menghindari terjadinya pengulangan dalam proses pengulangan tahapan karena metode waterfall mengharuskan mengikuti urutan pengerjaan.

2. Penelitian kedua yaitu pengembangan aplikasi yang dilakukan oleh Diska Amalia Widyaningrum, Mohamad Ali Murtadho, Teguh Priyo Utomo dengan judul "**Rancang Bangun Sistem Informasi Manajemen Penerimaan Beasiswa Berbasis Web Menggunakan Framework Codeigniter Di UNIPDU Jombang**" yang dimuat dalam jurnal

SITECH; Sistem Informasi dan Teknologi pada tahun 2022. Jurnal ini membahas tentang pengembangan sebuah aplikasi berbasis web untuk mengelola penerimaan beasiswa. Menurut penjelasan dari jurnal dapat diketahui bahwa pelamar beasiswa diharuskan untuk mengumpulkan segala dokumen persyaratan untuk pendaftaran beasiswa. Petugas yang bertanggung jawab untuk melakukan seleksi beasiswa akan menggunakan Microsoft Excel untuk membandingkan data setiap pelamar beasiswa dari dokumen tersebut. Dalam implementasinya, penelitian ini menggunakan framework Codeigniter untuk mempercepat proses pengembangan sistem dan meminimalkan kesalahan yang terjadi pada saat proses pemrograman. Hasil dari studi ini menunjukkan bahwa aplikasi pengelola penerimaan beasiswa yang dibangun dengan framework Codeigniter berbasis web di UNIPDU Jombang dapat membantu memudahkan proses penerimaan beasiswa. Aplikasi web sudah selesai dibangun sehingga dapat membuat proses seleksi penerimaan beasiswa lebih transparan serta memberikan informasi dengan lebih tepat dan mudah dipahami untuk para calon penerima beasiswa.

3. Jurnal ketiga yaitu perancangan sebuah sistem informasi yang dilakukan oleh Ardiansyah Putra, Zuhar Musliyana, Mulkan Fadhi dengan judul "**Perancangan Sistem Informasi Beasiswa Tahunan Pemerintah Kota Sabang Menggunakan Framework CodeIgniter**" yang dimuat dalam *Journal of Informatics and Computer Science* pada tahun 2019. Jurnal ini membahas tentang rancangan aplikasi penerimaan beasiswa tahunan yang dibangun dengan framework codeigniter untuk pemerintah kota sabang. Beasiswa tersebut merupakan dukungan finansial yang berasal dari pemerintah kota Sabang untuk para siswa yang memiliki tempat tinggal di kota Sabang. Menurut penjelasan pada jurnal dapat diketahui bahwa proses penerimaan beasiswa masih tidak efisien karena pemohon diharsukan menyerahkan berkas mereka ke pemerintah kota Sabang, kemudian pada proses mengecek dokumen dilakukan secara

konvensional satu persatu. Pemohon harus mengirimkan berkas ke pemerintah kota Sabang lagi jika berkasnya tidak sesuai. Peneliti menyimpulkan bahwa sistem informasi beasiswa yang dirancang menggunakan Framework CodeIgniter dapat memberikan kemudahan dalam proses pengajuan, verifikasi, dan seleksi penerima beasiswa. Aplikasi tersebut juga dapat meningkatkan produktivitas serta efisiensi dalam penanganan beasiswa tahunan di Pemerintah Kota Sabang.

4. Penelitian keempat yaitu penelitian yang dilakukan oleh Titania Pricillia dam Zulfachmi dengan judul "**Survey Paper: Perbandingan Metode Pengembangan Perangkat Lunak (Waterfall, *Prototype*, RAD)**" yang dimuat dalam jurnal MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer pada tahun 2017. Dalam jurnal ini, peneliti mengevaluasi penggunaan model RAD dalam pengembangan sistem informasi dengan menganalisis beberapa faktor seperti efektivitas, efisiensi, kualitas, keamanan, dan kemudahan penggunaan. Hasil analisis menunjukkan bahwa model RAD cukup efektif dan efisien dalam pengembangan sistem informasi, dengan tingkat kepuasan pengguna yang tinggi dan kualitas sistem yang baik. Selain itu, peneliti juga membahas keuntungan dan tantangan dalam menggunakan model RAD. Beberapa keuntungan yang disebutkan antara lain waktu pengembangan yang lebih cepat, lebih mudah untuk memenuhi kebutuhan pengguna, dan kemampuan untuk melakukan perubahan secara cepat dan fleksibel. Namun, tantangan yang muncul dalam penggunaan model RAD adalah adanya risiko keamanan yang lebih tinggi dan kesulitan dalam membangun sistem yang kompleks. Secara keseluruhan, jurnal ini memberikan tinjauan yang cukup komprehensif tentang penggunaan model RAD dalam pengembangan sistem informasi.

5. Penelitian kelima yaitu penelitian yang dilakukan oleh Muhammad Rizki Hidayat, Irsan Jaelani, Mutiara Andayani Komara dengan judul "**Web-Based Scholarship Management Information System Using Rapid**

Application Development (RAD) Model" yang dimuat dalam jurnal RISTEC: Research in Information Systems and Technology pada tahun 2021. Dalam jurnal ini peneliti membangun sebuah sistem beasiswa menggunakan PHP yang dibantu dengan framework codeigniter sedangkan basis data yang diterapkan untuk sistem tersebut adalah MySQL. Pada saat aplikasi sudah selesai peneliti melakukan pengujian *black box testing*. Menurut penjelasan yang ada di dalam jurnal diketahui bahwa proses pengajuan beasiswa yang diterapkan pada saat ini masih dilakukan secara konvensional mulai dari tahapan pengumpulan dokumen persyaratan hingga proses menentukan penerima beasiswa yang dilakukan oleh ketua program studi. model *Rapid Application Development* (RAD) menjadi pilihan metode yang digunakan oleh peneliti dan terdiri dari tahapan Perencanaan Persyaratan, Desain Workshop Rapid Application Development, dan Implementasi. Penelitian pada jurnal ini menghasilkan sebuah aplikasi untuk pengelolaan beasiswa yang memudahkan pihak perguruan tinggi selama proses penerimaan beasiswa. Dengan adanya sistem tersebut segala proses bisa dilakukan secara terkomputerisasi mulai dari informasi tentang peluang beasiswa, permohonan beasiswa, seleksi dan menentukan penerima.