

## **BAB III**

### **LANDASAN TEORI**

#### **3.1 Pengertian Sistem**

Pengertian sistem menurut Romney dan Steinbart (2015), Sistem adalah rangkaian dari dua atau lebih komponen-komponen yang saling berhubungan, yang berinteraksi untuk mencapai suatu tujuan. Sebagian besar sistem terdiri dari subsistem yang lebih kecil yang mendukung sistem yang lebih besar.

#### **3.2 Karakteristik Sistem**

Suatu sistem mempunyai ciri-ciri karakteristik yang terdapat pada sekumpulan elemen yang harus dipahami dalam mengidentifikasi pembuatan sistem. Adapun karakteristik sistem (Hutahaean, 2015) yang dimaksud adalah sebagai berikut:

1. **Komponen Sistem (*Component*)**

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang saling bekerja sama membentuk suatu komponen sistem atau bagian-bagian dari sistem.

2. **Batasan Sistem (*Boundary*)**

Merupakan daerah yang membatasi suatu sistem dengan sistem yang lain atau dengan lingkungan kerjanya.

3. **Subsistem**

Bagian-bagian dari sistem yang beraktivitas dan berinteraksi satu sama lain untuk mencapai tujuan dengan sasarannya masing-masing.

4. **Lingkungan Luar Sistem(*Environment*)**

Suatu sistem yang ada di luar dari batas sistem yang dipengaruhi oleh operasi sistem.

#### 5. Penghubung Sistem (*Interface*)

Media penghubung antara suatu subsistem dengan subsistem lainnya. Adanya penghubung ini memungkinkan berbagai sumber daya mengalir dari suatu subsistem ke subsistem lainnya.

#### 6. Masukan Sistem (*Input*)

Energi yang masuk ke dalam sistem, berupa perawatan dan sinyal. Masukan perawatan adalah energi yang dimasukkan supaya sistem dapat berinteraksi.

#### 7. Keluaran Sistem (*Output*)

Hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna, dan sisa pembuangan.

#### 8. Pengolahan Sistem (*Process*)

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan mengubah masukan menjadi keluaran.

#### 9. Sasaran (*Objective*)

Tujuan yang ingin dicapai oleh sistem, akan dikatakan berhasil apabila mengenai sasaran atau tujuan.

### 3.3 Joki Game Online

Joki Game Online adalah sebuah usaha yang menyangkut antara player biasa dan juga pro player dimana kedua nya saling berinteraksi untuk menjalankan bisnis joki yang dimana player biasa menyewa jasa dari pro player untuk meningkatkan level atau pangkat dalam game tersebut dengan cepat dan tidak mengulur waktu.

### 3.4 Analisa Sistem

Analisa sistem didefinisikan sebagaimana memahami dan menspesifikasikan dengan detail apa yang harus dilakukan oleh sistem. Analisa sistem sangat bergantung pada teori sistem umum sebagai sebuah landasan konseptual, yang tujuannya adalah untuk memperbaiki berbagai fungsi didalam sistem yang sedang berjalan, merancang *output* yang sedang digunakan, untuk mencapai tujuan yang sama dengan seperangkat *input* yang lain.

Tahapan dalam menganalisa sistem adalah sebagai berikut :

- a. Definisikan masalah yang mencakup *input*, proses dan *output* dari sistem yang sedang berjalan dan sistem yang akan dibangun.
- b. Pahami sistem yang sedang berjalan tersebut dan pahami definisinya.
- c. Mencari alternatif yang ditawarkan haruslah terdiri dari beberapa bentuk yang menunjukkan kelebihan dan kekurangan.
- d. Pilih salah satu alternatif yang telah dirumuskan pada tahap sebelumnya.
- e. Implementasikan alternatif terpilih dari sekian alternatif yang telah ditawarkan.
- f. Mengevaluasi dampak yang ditimbulkan akibat perubahan yang telah dilakukan terhadap sistem.

### 3.5 Perancangan Sistem

#### 3.5.1 *Unified Modeling Language (UML)*




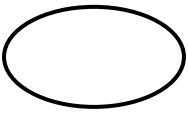
Menurut Mulyani (2016), “*UML* adalah sebuah teknik pengembangan sistem yang menggunakan bahasa grafis sebagai alat untuk pendokumentasian dan melakukan spesifikasi pada sistem”.

#### 3.5.2 *Use Case Diagram*

Menurut Muslihudin (2016), *Diagram Use Case* bersifat statis. *Diagram* ini memperlihatkan himpunan *use case* dan aktor-aktor (suatu jenis khusus dari kelas). *Diagram* ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.

Menurut Mulyani (2016), “*Use case diagram*, yaitu *diagram* yang digunakan untuk menggambarkan hubungan antara sistem dengan aktor”.


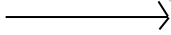
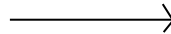
Tabel 3.1 Daftar Simbol *Use Case Diagram* (Gellysa Urva, Helmi Fauzi Siregar; 2015)

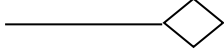
Simbol	Nama	Keterangan
	<p><i>Actor</i></p>	<p>Aktor adalah <i>abstraction</i> dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasi aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan <i>use case</i>, tetapi tidak memiliki control terhadap <i>use case</i>.</p>
	<p><i>Extend</i></p>	<p><i>Extend</i> merupakan perluasan dari <i>use case</i> lain jika kondisi atau syarat terpenuhi.</p>
	<p><i>Association</i></p>	<p>Asosiasi antara aktor dan <i>usecase</i> yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.</p>
	<p><i>Use case</i></p>	<p><i>Use Case</i> menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama <i>Use Case</i>.</p>

### 3.5.3 Class Diagram

Menurut Muslihudin (2016), *Diagram Kelas* bersifat statis. *Diagram* ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. *Diagram* ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas aktif.

Tabel 3.2 Daftar Simbol *Class Diagram* (Gellysa Urva, Helmi Fauzi Siregar; 2015)

Simbol	Nama	Keterangan
	<i>Class</i>	Kelas pada struktur sistem
	<i>Interface</i>	Sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek
	<i>Association</i>	Relasi antar kelas dengan makna umum, asosiasi biasanya di sertai dengan <i>multiplicity</i> .
	<i>Directed Association</i>	Relasi ant al kelas dengan kelas yang satu digunakan oleh kelas yang lain,
	<i>Generaliation</i>	Relasi antar kelas dengan makna generalisasi-spesialisasi(umum khusus).
	<i>Dependency</i>	Relasi antar kelas dengan makna kebergantungan antar kelas.




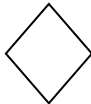
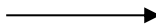
	<i>Aggregation</i>	Relasi antar kelas dengan makna semua bagian( <i>whole-part</i> ).
---	--------------------	--

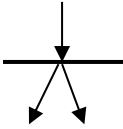
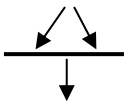

### 3.5.4 Activity Diagram

Menurut Muslihudin (2016), *Diagram* Aktivitas (*Activity Diagram*) bersifat dinamis. *Diagram* aktivitas adalah tipe khusus dari *diagram* status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. *Diagram* ini terutama penting dalam suatu sistem serta pemodelan fungsi-fungsi suatu sistem dan memberi tekanan pada aliran kendali antar objek.

Didalam *activity diagram* terdapat juga beberapa simbol. Berikut ini adalah simbol-simbol yang ada pada diagram aktivitas yaitu :

Tabel 3.3 Daftar simbol *Activity Diagram* (Gellysa Urva, Helmi Fauzi Siregar; 2015)



Simbol	Nama	Keterangan
	<i>Start state</i>	Diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	<i>End state</i>	Titik akhir atau akhir dari aktivitas.
	<i>Activity</i>	<i>Activity</i> atau aktivitas yang menggambarkan suatu proses / kegiatan bisnis.
	<i>Decision Points</i>	Menggambarkan pilihan untuk pengambilan keputusan <i>true/false</i> .
	<i>Message</i>	Simbol mengirim pesan antar <i>class</i> .


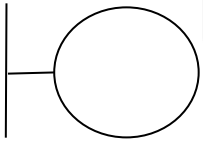
	<i>Fork</i>	Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau menggabungkan dua kegiatan paralel menjadi satu.
	<i>Join</i>	<i>Join</i> (penggabungan)/ <i>rake</i> digunakan untuk menunjukkan adanya dekomposisi.
	<i>Swimlane</i>	Pembagian <i>activity diagram</i> untuk menunjukkan siapa melakukan apa.

### 3.5.5 Sequence Diagram

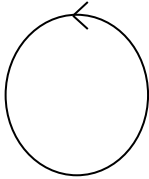
Menurut Muslihudin (2016), *Diagram* Interaksi dan Sequence (Urutan) bersifat dinamis. Dinamis urutan adalah *diagram* interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.

Tabel 3.4 Daftar Simbol *Sequence Diagram* (Gellysa Urva, Helmi Fauzi Siregar; 2015)

Simbol	Nama	Keterangan
	<i>Lifeline</i>	<i>Lifeline</i> mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk <i>Lifeline</i> adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.
	<i>Actor</i>	<i>Actor</i> juga dapat berkomunikasi dengan object, maka <i>actor</i> juga dapat diurutkan sebagai kolom. Simbol <i>Actor</i> sama dengan simbol pada <i>Actor</i>

		<i>Use Case Diagram.</i>
	<i>Activation</i>	<p><i>Activation</i> dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah <i>lifeline</i>. <i>Activation</i> mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.</p>
	<i>Object</i>	<p><i>Object</i> merupakan instance dari sebuah <i>class</i> dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah <i>class</i> (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma.</p>
	<i>Boundary Class</i>	<p><i>Boundary Class</i>, berisi kumpulan kelas yang menjadi interface atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan <i>form entry</i> dan <i>form cetak</i>.</p>



	<p><i>Entity Class</i></p>	<p><i>Entity Class</i>, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal system dan menjadi landasan untuk menyusun basis data.</p>
---	----------------------------	--

### 3.6 Alat Bantu Implementasi Sistem

#### 3.6.1 *Framework*

Salah satu alasan mengapa orang menggunakan *framework* terutama dalam membangun sebuah aplikasi adalah kemudahan yang ditawarkan. Didalam sebuah *framework* biasanya sudah tersedia struktur aplikasi yang baik, *standard coding*, *best practice*, *design pattern*, dan *common function*. Dengan menggunakan *framework* kita dapat langsung fokus kepada *business process* yang dihadapi tanpa harus berfikir banyak masalah struktur aplikasi, *standar coding* dan lain-lain.

Sedangkan menurut Raharjo (2015), *Framework* adalah suatu kumpulan kode berupa pustaka (*library*) dan alat (*tool*) yang dipadukan sedemikian rupa menjadi satu kerangka kerja (*framework*) guna memudahkan dan mempercepat proses pengembangan aplikasi *web*.

Jadi, *Framework* adalah kumpulan-kumpulan potongan program yang dipadukan menjadi satu kerja yang digunakan untuk membantu dalam pembuatan sebuah aplikasi.

#### 3.6.2 *Codeigneter*

Proses pengembangan *web* dapat dilakukan dengan beragam bahasa pemrograman seperti *PHP*, *Python*, *Ruby*, *Perl*, *C++*, *JAVA* dan sebagainya. Saat ini, banyak bermuculan *framework web* yang

dirancang untuk bahasa-bahasa pemrograman tersebut. Salah satunya adalah *Code Igniter*.

Sedangkan, menurut Raharjo (2015) *CodeIgniter* adalah *framework web* untuk bahasa pemrograman *PHP*, yang dibuat oleh Rick Ellis pada tahun 2006, penemu dan pendiri Ellis Lab.

Jadi *CodeIgniter* adalah sebuah *framework* buatan Rick Ellis yang digunakan untuk mempermudah pada *developer* dalam mengembangkan suatu aplikasi *web*.

Keuntungan menggunakan *CodeIgniter* menurut Raharjo (2015) *CodeIgniter* merupakan sebuah *toolkit* yang ditujukan untuk orang yang ingin membangun aplikasi *web* dalam bahasa pemrograman *PHP*. Beberapa keunggulan yang ditawarkan oleh *CodeIgniter* adalah sebagai berikut :

1. *CodeIgniter* adalah *framework* yang bersifat *free* dan *opensource*.
2. *CodeIgniter* memiliki ukuran yang kecil dibandingkan dengan *framework* lain. Setelah proses instalasi, *framework CodeIgniter* hanya berukuran kurang lebih 2 MB. Dokumentasi *CodeIgniter* memiliki ukuran sekitar 6 MB.
3. Aplikasi yang dibuat menggunakan *CodeIgniter* bisa berjalan cepat.
4. *CodeIgniter* menggunakan pola desain *Model-View-Controller(MVC)* sehingga satu *file* tidak terlalu berisi banyak kode. Hal ini menjadikan kode lebih mudah dibaca, dipahami, dan dipelihara dikemudian hari.
5. *CodeIgniter* dapat diperluas sesuai dengan kebutuhan.
6. *CodeIgniter* terdokumentasi dengan baik. Informasi tentang pustaka kelas dan fungsi yang disediakan oleh *CodeIgniter* dapat diperoleh melalui dokumentasi yang disertakan didalam paket distribusinya.

### 3.6.3 Bahasa Pemrograman *Hypertext Preprocessor (PHP)*

*PHP* sering dipakai para programmer untuk membuat situs *web* yang bersifat dinamis karena gratis dan berguna dalam merancang aplikasi *web*.

Menurut Supono dan Putratama (2016) mengemukakan bahwa ”*PHP (Hypertext Preprocessor)* adalah suatu bahasa pemrograman yang digunakan untuk menerjemahkan baris kode program menjadi kode mesin yang dapat dimengerti oleh komputer yang berbasis *server-side* yang dapat ditambahkan ke dalam *HTML*”.

Sedangkan, menurut Solichin (2016) mengemukakan bahwa “*PHP* merupakan salah satu bahasa pemrograman berbasis *web* yang ditulis oleh dan untuk pengembang *web*”.

Kumpulan kutipan diatas menerangkan bahwa *Hypertext Preprocessor (PHP)* merupakan bahasa pemrograman untuk membuat/mengembangkan aplikasi berbasis *web* dan bersifat *open soure* dan ditanamkan ke dalam script *HTML*.

### 3.6.4 Bahasa Pemrograman *Hyper Text Markup Language (HTML)*

Proses tampilnya sebuah halaman *website* di *browser* melibatkan *HTML. Hyper Text Markup Language (HTML)* tergolong dalam salah satu format yang digunakan dalam pembuatan dokumen yang terbaca oleh *web*.

Menurut Solichin (2016) mengemukakan bahwa “*HTML* merupakan bahasa pemrograman *web* yang memberitahukan peramban *web (web browser)* bagaimana menyusun dan menyajikan konten di halaman *web*”.

Berdasarkan teori dari para ahli di atas, maka *Hypter Text Markup Language (HTML)* merupakan bahasa pemrograman yang dikenal oleh *browser* untuk menampilkan informasi lebih menarik di halaman *web* melalui *web browser*.

### 3.6.5 Database

Menurut Indrajani (2015), basis data adalah kumpulan data yang saling berhubungan secara logis dan didesain untuk mendapatkan data yang dibutuhkan oleh suatu organisasi.

Menurut Sukamto dan Shalahuddin (2015) *DBMS(Database management system)* atau dalam bahasa Indonesia sering disebut Sistem manajemen basis data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data.

Dari pengertian *Database* diatas dapat disimpulkan bahwa *database* adalah sekumpulan tabel-tabel yang berisi data dan merupakan kumpulan dari *field* atau kolom dan suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data.

### 3.6.6 MySQL

Menurut Hidayatullah dan Jauhari (2015) “*MySQL* adalah salah satu aplikasi *DBMS* yang sudah banyak oleh para pemogram aplikasi *web*. Contoh *DBMS* lainnya adalah *PostgreSQL (freeware)*, *SQLServer*, *MS Access* dari *Microsoft*, *DB2* dari *IBM*, *Oracle* dan *Oracle Corp*, *Dbase*, *FoxPro*, dsb”.

Berdasarkan penjelasan diatas dapat disimpulkan bahwa *MySQL* adalah aplikasi *DBMS* yang menjalankan fungsi pengolahan data untuk membangun sebuah aplikasi *web*.

### 3.6.7 XAMPP

Menurut Purbadian (2016), berpendapat bahwa “*XAMPP* merupakan suatu software yang bersifat *open source* yang merupakan pengembangan dari *LAMP (Linux, Apache, MySQL, PHP dan Perl)*”.

Berdasarkan pengertian diatas dapat disimpulkan bahwa *XAMPP* merupakan *tool* pembantu pengembangan paket perangkat lunak berbasis *open source* yang menggabungkan *Apache web server*, *MySQL*, *PHP* dan beberapa modul lainnya didalam satu paket aplikasi.

### 3.7 Pengujian Sistem

Menurut Pressman (2010), pengujian sistem atau *system testing* adalah serangkaian pengujian dengan tujuan utamanya untuk menjalankan seluruh elemen sistem yang dikembangkan. Pengujian dilakukan untuk menemukan dan memperbaiki sebanyak mungkin kesalahan dalam program sebelum menyerahkan program kepada *customer*. Salah satu pengujian yang baik adalah pengujian yang memiliki probabilitas tinggi dalam menemukan kesalahan.

#### 3.7.1 White - Box Testing

Pengujian *White-box* atau *Glass-box* adalah metode *test-case design* yang menggunakan struktur kontrol desain *procedural* untuk memperoleh *test-case*. Dengan menggunakan metode pengujian *white-box*, perancang sistem dapat memperoleh *test-case* yang:

- a) Memberikan jaminan bahwa semua jalur *independent* pada suatu modul telah digunakan paling tidak satu kali.
- b) Menggunakan semua keputusan logis dari sisi *true* dan *false*.
- c) Mengeksekusi semua batas fungsi *loops* dan batas operasionalnya.
- d) Menggunakan struktur internal untuk menjamin validitasnya.

#### 3.7.2 Black - Box Testing

*Black-box testing* merupakan pengujian yang berpusat pada kebutuhan fungsional perangkat lunak dimana memungkinkan untuk memperoleh sekumpulan kondisi input yang secara penuh memeriksa fungsional dari sebuah aplikasi. *Black-box testing* berusaha menemukan kesalahan - kesalahan seperti kesalahan fungsi dan kesalahan tampilan aplikasi. *Black-box testing* dapat digunakan untuk menguji aplikasi konvensional dan aplikasi yang berorientasi objek.

### 3.8 Tinjauan Pustaka



USM