BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Pada bab ini dibahas tinjauan pustaka dari pembuatan aplikasi *game Guess The Words* menggunakan Python.

2.1.1 Penelitian dari Angga Pranandaka yang Berjudul Rancang Bangun Aplikasi Infografis Kependudukan Kota Salatiga Berbasis Android dengan Web Service Python. (Pranandaka, 2020)

Kesimpulan dari penelitian yang dilakukan yakni:

a) Penggunaan aplikasi infografis kependudukan di Kota Salatiga yang berbasis Android dengan *web service* Phyton dapat memberikan peringkasan tahapan pada pengguna karena memiliki probabilitas yang tinggi dimana dapat mempermudah pengguna dalam mengakses data kependudukan dan memiliki tampilan yang menarik.
b) Hasil dari pengujian yang dilakukan melalui pengujian *blackbox* didapatkan bahwa semua aksi menu dan tampilan telah berfungsi dengan baik, sedangkan hasil yang didapatkan dari pengujian kuesioner memiliki persentase sebanyak 87,7% yang berarti tingkat kepuasan pengguna terhadap aplikasi terklasifikasi sangat baik.

2.1.2 Penelitian dari Fitria Aprilia Kartini yang Berjudul Aplikasi Game Gatotkaca Berbasis Android. (Kartini, 2017)

Kesimpulan dari penelitian yang dilakukan yakni:

a) Pengujian *blackbox* yang dilakukan pada aplikasi *game* Gatotkaca yang berbasis Android didapatkan hasil bahwa tidak terdapat bug atau kesalahan proses.
b) Berdasarkan pengujian kepuasan pengguna terhadap *game* Gatutkoco melalui kuesioner didapatkan hasil persentase 67,7% yang mengatakan bahwa permainan ini merupakan *game* yang menarik.

2.1.3 Penelitian dari Edwifa Hendri yang Berjudul Pengembangan Sistem Pengelolaan Keuangan Sekolah Berbasis Web dengan Metode Waterfall pada SMPIT Ummi Abiyyi. (Hendri, 2017)

Kesimpulan dari penelitian yang dilakukan yakni:

a) Penelitian yang dilakukan oleh penulis dari mulai melakukan identifikasi masalah hingga pengujian sistem yang baru dikembangkan dengan metode *waterfall* mendapatkan hasil bahwa waktu yang dibutuhkan dalam pembuatan laporan lebih singkat karena tidak ada pengulangan dalam penginputan.
b) Penggunaan metode *waterfall* dalam penelitian ini mendapatkan hasil bahwa metode tersebut dapat membantu jalannya penelitian sehingga lebih cepat dalam melakukan pengambilan keputusan.

2.1.4 Penelitian dari Lukas Tommy yang Berjudul Rancang Bangun Game Visual Novel "Cerita Si Budi" dengan Ren'Py. (Tommy, 2014)

Kesimpulan dari penelitian yang dilakukan yakni:

a) Penelitian yang dilakukan oleh penulis menghasilkan *game* “Cerita Si Budi” yang dapat membantu para orang tua maupun pendidik dalam memberikan contoh perilaku berbudi pekerti kepada anak-anak dengan efektif, interaktif, dan inovatif.
b) Penulis berhasil membuat *game* yang dapat dioperasikan dengan mudah oleh *user* dari segala usia.

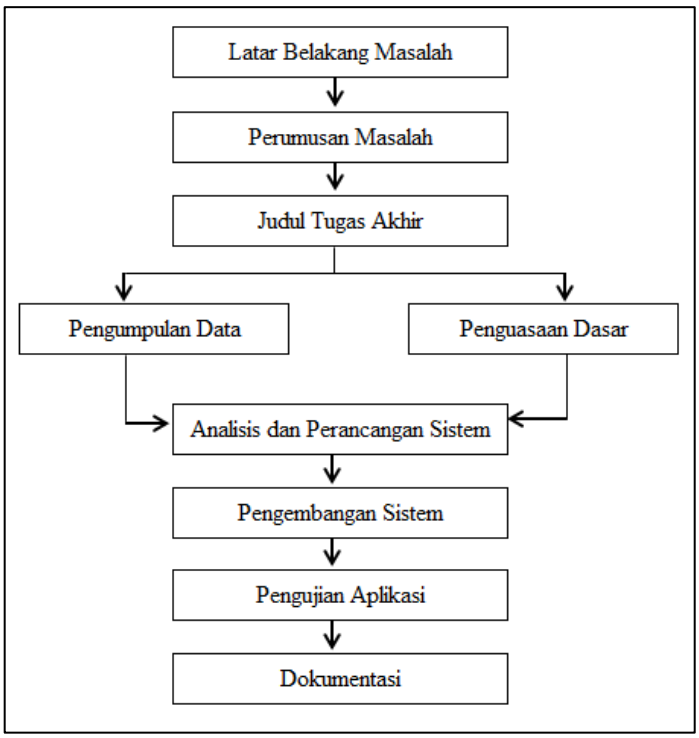
2.1.5 Penelitian dari Rizky Nur Falah dan Suhana Minah Jaya yang Berjudul Penerapan Cerita Rakyat Tangkuban Perahu Menggunakan Visual Novel (VN) dengan Menggunakan Ren'Py. (Falah & Jaya, 2020)

Kesimpulan dari penelitian yang dilakukan yakni:

a) Penelitian yang dilakukan oleh penulis menghasilkan sebuah aplikasi vsiual novel menggunakan bahasa pemrograman Python yang membuat cerita menjadi lebih interaktif.
b) Penelitian ini berhasil menerapkan cerita rakyat Legenda Tangkuban Perahu menjadi sebuah aplikasi Visual Novel dengan bahasa pemrograman Python.

2.2 Kerangka Pemikiran

Gambar 2.1 di bawah ini merupakan kerangka pemikiran untuk menjelaskan urutan proses dari penelitian tugas akhir ini.



Gambar 2.1 Kerangka Pemikiran

A. Latar Belakang Masalah

Pokok permasalahan yang mendasari pembuatan rancang bangun aplikasi *game* Guess The Words menggunakan Python.

B. Perumusan Masalah

Perumusan masalah merupakan inti dari permasalahan yang muncul dari latar belakang yang digunakan dalam penelitian.

C. Judul Tugas Akhir

Judul yang sesuai untuk menangani masalah yang sesuai dengan latar belakang dan perumusan masalah yang ada.

D. Pengumpulan Data

Mengumpulkan data yang akan digunakan untuk keperluan penelitian dan proses berjalannya pembuatan aplikasi dari awal hingga akhir.

E. Penguasaan Dasar (Python, Pygame, UML)

Pada penelitian ini dilakukan pembelajaran agar lebih menguasai *platform* yang digunakan untuk membangun sistem atau kinerja sistem yaitu Python, Pygame, dan UML (*Unified Modelling Language*).

F. Analisis dan Perancangan Sistem

Pada penelitian ini dilakukan perancangan bagaimana sistem nantinya akan dibuat untuk membantu memecahkan permasalahan yang ada.

G. Pengembangan Sistem

Pada tahap pengembangan sistem dilakukan *coding source code* dan pembuatan GUI (*Graphical User Interface*) sehingga menghasilkan aplikasi yang bisa digunakan.

H. Pengujian Aplikasi

Pada penelitian dilakukan uji coba aplikasi yang telah dibuat apakah masih terjadi *error* ataupun kekurangan pada sistem yang mengakibatkan aplikasi belum berjalan sesuai rencana.

I. Dokumentasi

Aplikasi yang telah lolos uji coba kemudian dibuat dokumentasi dari keseluruhan kegiatan penyusunan Tugas Akhir.

2.3 Landasan Teori

2.3.1 Rancang Bangun

Rancang bangun atau desain adalah tahapan dari setelah analisis dari siklus pengembangan sistem yang didefinisikan dari kebutuhan-kebutuhan fungsional, yang juga menggambarkan bagaimana suatu sistem dibentuk dengan hasil berupa penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah menjadi satu kesatuan yang utuh dan memiliki fungsi, termasuk mengkonstruksi dari gabungan *software* dan *hardware* dari suatu sistem. (Jogiyanto, 2012)

2.3.2 Aplikasi

Aplikasi adalah *software* yang dibuat dan digunakan untuk membantu manusia dalam mempermudah pekerjaannya, seperti Microsoft Office, Adobe, dan Email. Aplikasi berasal dari kata *application* yang artinya penerapan atau penggunaan. (Sanjaya, 2015)

Jadi aplikasi bisa dikatakan sebagai suatu bentuk perbaikan dari suatu masalah atau pekerjaan dari yang tadinya merupakan hal yang sulit dipahami menjadi lebih sederhana atau lebih mudah dipahami oleh pengguna. Dengan adanya aplikasi, maka permasalahan akan terselesaikan lebih cepat dan tepat.

Aplikasi dikategorikan menjadi beberapa jenis, seperti aplikasi *desktop* yang beroperasi secara *offline* dan *website* yang beroperasi secara *online*. Aplikasi juga dapat diartikan sebagai perangkat lunak yang menggabungkan beberapa fitur tertentu dengan berbagai macam cara yang dapat diakses oleh pengguna.

Ada beberapa klasifikasi dan jenis aplikasi untuk mempermudah *user* dalam mengetahui jenis-jenis aplikasi yang ingin digunakan. Beberapa aplikasi dikelompokkan berdasarkan klasifikasi aplikasinya yaitu *real time software, system software, business software, personal computer software, web based software,* dan *engineering and scientific software.* Pada aplikasi yang akan digunakan dalam penelitian ini dikategorikan dalam kategori *personal computer software* karena dapat digunakan oleh pengguna resmi maupun pribadi.

2.3.3 Permainan (Game)

Ludwig Wittgenstein mungkin adalah filsuf akademis pertama yang membahas definisi kata *game*. Dalam investigasi filosofinya, Wittgenstein berpendapat bahwa unsur-unsur permainan seperti mulainya permainan, aturan, dan kompetisi, semuanya tidak berhasil untuk mendefinisikan secara tepat apa itu permainan (*game*). Dari sini Wittgenstein menyimpulkan bahwa orang menerapkan istilah permainan ke berbagai aktivitas manusia yang berbeda yang hanya berhubungan satu sama lain yang sejenis dan mirip. (Wittgenstein, 1953)

Permainan adalah suatu bentuk hiburan yang biasanya dilakukan untuk kesenangan dan digunakan sebagai alat pendidikan. Permainan berbeda dengan pekerjaan yang biasanya dilakukan agar mendapatkan penghasilan, namun kini masyarakat bisa mendapatkan penghasilan dari bermain *game*, seperti pemain *game* profesional yang ikut dalam tim *e-sport*. Komponen kunci dari permainan adalah tujuan, aturan, tantangan, dan interaksi. Permainan pada umumnya melibatkan stimulasi mental atau fisik, dan seringkali keduanya. Banyak permainan yang membantu mengembangkan keterampilan praktis, berfungsi sebagai bentuk latihan, melakukan peran pendidikan, simulasi, dan psikologis.

2.3.4 Python

Python adalah bahasa pemrograman modern yang mendukung gaya pemrograman berorientasi objek, fungsional, dan imperatif. Python merupakan bahasa pemrograman yang sangat ideal untuk pemula karena mudah dibaca dan mudah digunakan. Keuntungan dari Python adalah program yang ditulis memiliki lebih sedikit baris kode daripada program C/C++ atau Java yang setara. (Kelly, 2019)

Python dikembangkan oleh seseorang dari Belanda yang bernama Guido van Rossum pada tahun 1990 yang merupakan kelanjutan dari bahasa pemrograman ABC. Saat ini pengembangan Python terus dilakukan oleh sekumpulan *programmer* yang dikoordinasikan oleh Guido dan Python *Software* Foundation. Python *Software* Foundation adalah sebuah organisasi yang dibentuk sebagai pemegang hak cipta Python sejak versi 2.1 yang bertujuan untuk mencegah Python

dimiliki oleh perusahaan komersial. Saat ini distribusi Python sudah mencapai versi 3.9.5 yang mana digunakan untuk pembuatan penelitian ini.

2.3.5 Pygame

Pygame merupakan salah satu modul dari Python. Pygame berfungsi untuk membangun sebuah *game* dari Python. Pygame dibuat untuk memungkinkan pengembangan *game* tanpa menggunakan bahasa pemrograman seperti C atau C++. Pygame dapat digunakan untuk menulis *game* 2D serba cepat dalam gaya retro, atau *game* kasual dan hiper-kasual modern, yang dapat mengatasi kesulitan dalam memuat gambar, menampilkan grafik komputer yang dapat dipindahkan di layar dan dimanipulasi sebagai satu kesatuan, memutar suara, dan menjalankan *command*.

Pygame ditulis oleh Pete Shinners untuk menggantikan PySDL setelah pengembangannya terhenti. Kemudian dikelola oleh komunitas sejak tahun 2000 dan dirilis di bawah GNU Lesser General Public Licence. (Kelly, 2019)

2.3.6 PyCharm

PyCharm adalah *Integrated Development Environment* (IDE) yang digunakan dalam pemrograman komputer, khusus untuk bahasa Python. PyCharm dikembangkan oleh perusahaan Ceko JetBrains atau yang sebelumnya dikenal dengan IntelliJ. PyCharm menyediakan analisis kode, debugger grafis, unit tester terintegrasi, dan juga mendukung pengembangan web dengan Django serta *data science* dengan Anaconda. (Kroger, 2015)

PyCharm merupakan lintas *platform* yang menyediakan versi untuk Windows, macOS, dan Linux. Edisi komunitas PyCharm dirilis di bawah Lisensi Apache, dan ada juga Edisi Profesional dengan fitur tambahan yang dirilis di bawah lisensi kepemilikan.

2.3.7 Pengujian Black Box

Pengujian Black Box merupakan langkah atau tahapan yang digunakan untuk menguji kelancaran program atau aplikasi yang telah dibuat. Pengujian ini penting dilakukan agar tidak terjadi *miss* terhadap alur program yang telah dibuat.

Black Box *testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. (Rosa & Shalahuddin, 2015)

Black Box *testing* berfokus pada spesifikasi fungsional dari perangkat lunak, kumpulan kondisi input dan melakukan pengetesan pada fungsional program. (Mustaqbal, Firdaus, & Rahmadi, 2015)

Uji Black Box hanya terdiri dari meninjau fungsi-fungsi dari aplikasi, struktur dan fungsi internalnya tidak dipelajari. Dengan demikian penguji perlu mengetahui peran dari sistem yang akan diuji dan fungsinya. Jadi pengujian ini bertujuan untuk memeriksa program setelah program selesai dibuat atau telah berada di tahap akhir pembuatan.

Black Box *testing* memiliki beberapa teknik yaitu Equivalance Partitioning, Boundary Value Analysis, Fuzzing, Cause-Effect Graph, Orthogonal Array Testing, All Pair Testing, State Transition.

2.3.8 Pengujian Kuesioner

Kuesioner adalah suatu teknik pengumpulan informasi yang memungkinkan analis mempelajari sikap-sikap, keyakinan, perilaku, dan karakteristik beberapa orang utama di dalam organisasi yang bisa terpengaruh oleh sistem yang diajukan atau oleh sistem yang sudah ada. dengan menggunakan kuesioner, penulis berupaya mengukur pengalaman pengguna terhadap apa yang ditemukan dalam sistem. (Sugiyono, 2017)

Cara pengumpulan informasi dalam jumlah besar yang relatif murah, cepat, dan efisien. Dengan kuesioner data yang didapatkan juga bersifat masif dan menghasilkan *sample* yang banyak. Pengumpulan datanya juga relatif cepat karena peneliti tidak perlu hadir pada saat pengisian kuesioner. Hal ini berguna untuk meneliti populasi besar.

Namun metode ini tetap memiliki kekurangan. Masalah pada kuesioner adalah bahwa responden bisa saja memberikan jawaban yang tidak sesuai dengan kenyataan karena keinginan sosial.

Terlepas dari kekurangan tersebut kuesioner adalah alat yang efektif untuk mengukur perilaku, sikap, preferensi, pendapat, dan niat dari subjek dalam jumlah yang relatif besar dengan biaya yang lebih murah dan cepat jika dibandingkan dengan metode lain.

2.3.9 Analisis SWOT

Analisis SWOT adalah metode perencanaan strategis yang digunakan untuk mengevaluasi mengevaluasi kekuatan (*strengths*), kelemahan (*weaknesses*), peluang (*opportunities*), dan ancaman (*threats*) dalam suatu proyek atau suatu spekulasi bisnis. Keempat faktor itulah yang membentuk akronim SWOT (*strengths, weaknesses, opportunities, dan threats*). (Rangkuti, 2016)

Proses ini melibatkan penentuan tujuan yang spesifik dari spekulasi bisnis atau proyek dan mengidentifikasi faktor internal dan eksternal yang mendukung dan yang tidak mendukung.

Analisis SWOT dapat diaplikasikan dengan cara menganalisis dan menyaring berbagai hal yang mempengaruhi keempat faktornya. Kemudian diterapkan dalam tabel yang akan lebih mempermudah untuk dipahami. Analisisnya adalah bagaimana kekuatan (*strengths*) mampu mengambil keuntungan dari peluang (*opportunities*) yang ada, bagaimana cara mengatasi kelemahan yang ada, selanjutnya bagaiaman kekuatan (*strengths*) mampu menghadapi ancaman (*threats*) yang ada, dan bagaimana cara mengatasi kelemahan (*weaknesses*) yang mampu membuat ancaman (*threats*) menjadi nyata atau malah menciptakan sebuah ancaman baru.

2.3.10 UML (Unified Modelling Language)

Menurut Rosa & Shalahuddin (2015), berpendapat bahwa UML (*Unified Modelling Language*) adalah "Salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisa & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek". Sedangkan (Mulyani, 2016) mengatakan UML (*Unified Modelling Language*) adalah "Sebuah teknik pengembangan sistem yang menggunakan bahasa grafis sebagai alat untuk pendokumentasian dan melakukan spesifikasi pada sistem".

Dari beberapa penjelasan teori tersebut dapat disimpulkan bahwa UML (*Unified Modelling Language*) adalah bahasa yang sering digunakan untuk membangun sebuah sistem perangkat lunak dengan melakukan penganalisaan desain dan spesifikasi dalam pemrograman berorientasi objek.

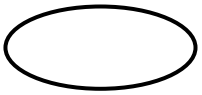
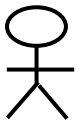
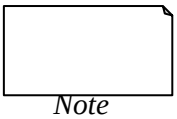
UML menyediakan 6 macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek yaitu Use Case Diagram, Class Diagram, Activity Diagram, Sequence Diagram, Component Diagram, Deployment Diagram.

2.3.10.1 Use Case Diagram

Use Case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case diagram* adalah sebuah diagram yang menunjukkan hubungan antara *actors* dan *use cases*. Digunakan untuk analisis dan desain sebuah sistem. (Rosa & Shalahuddin, 2015)

Keterangan simbol-simbol yang digunakan pada *use case diagram* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Simbol-simbol pada Use Case Diagram

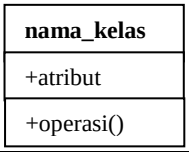
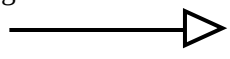
No.	Simbol	Deskripsi
1.	 *Use case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan anta unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *Use case.*
2.	 Aktor/*Actor*	Orang, proses atau sistem lain yang berinteraksi dengan sistem 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.	*Association*	Komunikasi anatara aktor dan *Use case* yang berpartisispasi pada *Use case* atau *Use case* memiliki interaksi dengan aktor.
4.	*<<extend>>* 	Relasi *Use case* tambahan ke sebuah *Use case* dimana *Use case* yang ditambahkan dapat berdiri sendiri walau tanpa *Use case* tambahan itu, mirip dengan prinsip inheritance pada pemrograman berorientasi objek.
5.	 *<<include>>*	Relasi *Use case* tambahan ke sebuah *Use case* dimana *Use case* yang ditambahkan memerlukan *Use case* ini untuk menjalankan fungsina atau sebagai syarat dijalankan *Use case* ini.
6.	 *Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *Use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
7	 *Realization*	Penjelasan use case dari sisi physical yaitu basis datanya.
8	 *Note*	Memberikan informasi method apa yang ada didalamnya dan nama database.

2.3.10.2 Class Diagram

Diagram Kelas atau *Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. (Rosa & Shalahuddin, 2015)

Keterangan simbol-simbol yang digunakan pada *class diagram* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Simbol-simbol pada Class Diagram

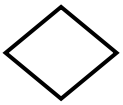
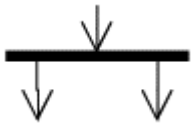
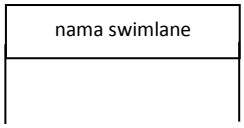
No.	Simbol	Deskripsi
1.	Kelas 	Kelas pada struktur sistem.
2.	antarmuka / *interface*  **nama_interface**	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
3.	asosiasi / a*ssociation*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplcity.*
4.	asosiasi 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 antar kelas dengan menggunakan makna generalisasi-spesialisasi (umum-khusus).
6.	kebergantungan / *dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
7.	agregasi / *aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)
8.	*multiplicity* 1...*	Satu atau lebih objek sebuah kelas yang berelasi dengan sebuah objek lain dari kelas lain yang berasosiasi dengan kelas tersebut.

2.3.10.3 Activity Diagram

Activity Diagram atau diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. (Rosa & Shalahuddin, 2015)

Keterangan simbol-simbol yang digunakan pada *activity diagram* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Simbol-simbol pada Activity Diagram

No.	Simbol	Deskripsi
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 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.3.10.4 Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. *Sequence diagram* adalah sebuah diagram yang menggambarkan kolaborasi dari objek-objek yang saling berinteraksi antar elemen dari suatu *class*. (Rosa & Shalahuddin, 2015)

Keterangan simbol-simbol yang digunakan pada *sequence diagram* dapat dilihat pada Tabel 2.4.

Tabel 2.4 Simbol-simbol pada Sequence Diagram

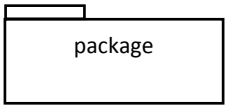
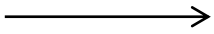
No.	Simbol	Deskripsi
1.	objek nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan.
2.	garis hidup / *lifeline*	Menyatakan kehidupan suatu objek.
3.	waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi.
4.	pesan tipe create <<create>>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
5.	pesan tipe call 1 : nama_metode()	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
6.	pesan tipe send 1 : masukan	Menyatakan suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
7.	pesan tipe return 1 : keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

2.3.10.5 Component Diagram

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan di antara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. (Rosa & Shalahuddin, 2015)

Keterangan simbol-simbol yang digunakan pada *component diagram* dapat dilihat pada Tabel 2.5.

Tabel 2.5 Simbol-simbol pada Component Diagram

No.	Simbol	Deskripsi
1.	*Package* 	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen.
2.	komponen 	Komponen sistem.
3.	kebergantungan / *dependency* 	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai.
4.	antar muka / *interface*  nama_interface	Sama dengan konsep interface pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen.
5.	*link*	Relasi antar komponen.

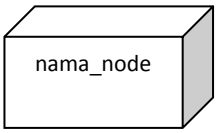
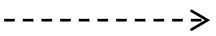
2.3.10.6 Deployment Diagram

Diagram konfigurasi atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. (Rosa & Shalahuddin, 2015)

Alur proses penerapan atau *deploy* aplikasi ini terdiri dari beberapa langkah meliputi *planning* (perencanaan), *development* (pengembangan), *testing* (pengujian), *deployment* (penerapan), dan *monitoring* (pemantauan).

Keterangan simbol-simbol yang digunakan pada *deployment diagram* dapat dilihat pada Tabel 2.6.

Tabel 2.6 Simbol-simbol pada Deployment Diagram

No.	Simbol	Deskripsi
1.	*Constraint* 	*Constraint* adalah mekanisme perpanjangan yang memungkinkan untuk menyempurnakan semantik elemen model UML.
2.	*Node* 	Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika di dalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen.
3.	kebergantungan / *dependency* 	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai.
4.	*link*	Relasi antar *node.*