BAB II
TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Penelitian Terkait dimaksudkan bahwa masalah yang hendak di teliti belum pernah di pecahkan oleh peneliti terlebih dahulu. Jika permasalahannya mirip, maka harus ditegaskan perbedaan penelitiannya dengan penelitian terlebih dahulu. Berikut adalah beberapa jurnal penelitian terdahulu terkait judul penelitian yang sedang dilaksanakan:

1. Kentol Mas Sudrajat, (2019) melakukan penelitian "Perancangan Aplikasi Pengamanan *File* Teks Menggunakan Algoritma *El Gamal* Dan Kompresi *File* Teks Menggunakan Algoritma *Huffmann*" Pada penelitian ini *file* hasil dekripsi sama persis, seperti *file* awal sebelum dilakukan proses enkripsi, misalnya jenis huruf, ukuran huruf dan sebagainya mengalami perubahan setelah dilakukan proses enkripsi dan dekripsi, sebagaimana telah dilakukan pengujian menggunakan program pengujian kemiripan *file* yaitu program *spy clone*, Dokumen atau *file* teks yang sudah dienkripsi menjadi *chiperteks* memiliki karakter yang lebih banyak atau panjang dibandingkan dengan *file* teks sebelum dilakukan proses enkripsi.
2. Ardelia Nidya Agustina, Aryanti, Nasron (2017) melakukan penelitian "Pengaman Dokumen Menggunakan Metode *RSA (Rivest Shamir Adleman)* Berbasis *Web"*. Pada penelitian ini Penerapan *program* pengamanan dokumen menghasilkan suatu aplikasi yang dapat mengubah *file* asli

(plaintext) menjadi *file* terenkripsi(*ciphertext*) yang tidak dapat dibaca informasi dari *filenya* kemudian mengembalikannya kembali menjadi *file* aslinya (*ciphertext*) tanpa merubah ataupun merusak isi *file* nya, Proses pengujian aplikasi menggunakan metode *RSA* dan telah di tes secara aplikasi, hasilnya adalah proses enkripsi dan dekripsi telah sesuai dengan kaidah algoritma kriptografi *RSA.*

3. Raja Nasrul Fuad, Haikal Nando Winata (2017) melakukan penelitian "Aplikasi Keamanan *File Audio WAV (Waveform)* Dengan Terapan Algoritma *RSA*". Perangkat lunak *(software)* dapat melakukan penyandian data *audio* dengan menerapkan algoritma *RSA* dan struktur data *audio WAV*, kemudian Ukuran berkas audio WAV menjadi bertambah besar setelah dilakukan enkripsi menggunakan algoritma *RSA* berdasarkan besar kunci yang digunakan.
4. Indra Gunawan (2018) melakukan penelitian "Kombinasi Algoritma *Caesar Cipher* Dan Algoritma *RSA* Untuk Pengamanan *File* Dan Pesan Teks". Pada penelitian ini kombinasi *Caesar Cipher* dan Algoritma *RSA* dapat membantu meningkatkan data, jika dibandingkan dengan hanya menggunakan satu metode saja. Menggunakan perhitungan dengan struktur *alphabetis* dan dipadukan dengan menggunakan memfaktoran bilangan prima dapat meningkatkan sistem keamanan data, sehingga data yang tersimpan akan semakin terjaga.

5. Penelitian terkait lain yang dilakukan oleh Nada Safarina (2017) dengan judul "Penerapan Algoritma *RSA* Dan *DES* Pada Pengamanan *File* Teks". Hasil implementasi kombinasi pengamanan *file* teks menggunakan *RSA* dan *DES* menunjukkan hasil yang cukup baik, dimana pesan teks yang terenkripsi memiliki keunggulan dalam proses autentikasi dimana menerapkan kunci publik dan kunci *private* dari algoritma *RSA* serta memiliki keunggulan dalam kompleksitas chiperteks dari algoritma *DES*.

Berdasarkan hasil penelitian yang terdahulu, maka dibuatlah kesimpulan untuk merancang sebuah aplikasi untuk menerapkan sebuah algoritma yang belum pernah digunakan sebelumnya untuk melakukan enkripsi dan dekripsi terhadap data berupa teks pada *file pdf* yaitu Algoritma *RSA*. Sehingga pada penulisan skripsi ini dibuatlah sebuah judul **"**Implementasi Keamanan Dan Enkripsi data menggunakan Algoritma *RSA* dalam Keamanan *File PDF"*. Berdasarkan judul tersebut nantinya akan dihasilkan sebuah aplikasi untuk melakukan enkripsi dan dekripsi terhadap data berupa teks pada *file pdf*.

II.2. Landasan Teori

II.2.1 Kriptografi

Kriptografi pada awalnya dijabarkan sebagai ilmu yang mempelajari ilmu bagaimana menyembunyikan pesan. Namun pada pengertian *modern* kriptografi adalah ilmu yang bersandarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi dan entitas.

Jadi pengertian kriptografi *modern* adalah tidak saja berurusan hanya dengan penyembunyian pesan namun lebih pada sekumpulan teknik yang menyediakan keamanan informasi (Kentol Mas Sudrajat, 2019;173).

II.2.2. Algoritma RSA

RSA merupakan salah satu dari *Public Key Cryptosystem* yang sangat sering digunakan untuk memberikan kerahasiaan terhadap keaslian suatu data *digital*. Keamanan enkripsi dan dekripsi data model ini terletak pada kesulitan untuk memfaktorkan *modulus n* yang sangat besar. Dalam kriptografi, *RSA* adalah algoritma untuk enkripsi kunci publik. Algoritma ini adalah algoritma pertama yang diketahui paling cocok untuk menandai (*signing*) dan untuk enkripsi dan salah satu penemuan besar pertama dalam kriptografi kunci publik. *RSA* masih digunakan secara luas dalam protokol-protokol perdagangan *elektronik* dan dipercayai sangat aman karena diberikan kunci-kunci yang cukup panjang dan penerapan-penerapannya yang sangat mutakhir.

Algoritma pembentukan kunci :

1. Tentukan *p* dan *q* bernilai dua bilangan prima besar, acak dan dirahasiakan, $p \neq q$, *p* dan *q* memiliki ukuran yang sama.
2. Hitung $n = p \times q$, dan hitung $\phi(n) = (p - 1) \times (q - 1)$, bilangan *integer n* disebut *(RSA) modulus*.
3. Tentukan *e* bilangan prima acak yang memiliki syarat : $1 < e < 1(n)$, GCD(*e*, i(*n*)) = 1, disebut *e* relatif prima terhadap i(*n*), bilangan *integer* n disebut

(RSA) *enciphering component*, sehingga menghasilkan $D_d\ (E_e(m)) = E_e(D_d(c)) \equiv m^d \bmod n$. (Indra Gunawan, 2018;125).

II.2.3 Enkripsi dan Dekripsi

1. **Enkripsi**

Enkripsi adalah proses dimana informasi atau data yang hendak dikirim diubah menjadi bentuk yang hampir tidak dikenali sebagai informasi awalnya dengan menggunakan algoritma tertentu.

2. **Dekripsi**

Dekripsi adalah kebalikan dari enkripsi yaitu mengubah kembali bentuk tersamar tersebut menjadi informasi awal. (Faturungi Muharram, Huzain Azis, Abdul Rachman Manga, 2018; 112)

II.2.4. *WEB*

Web merupakan sekumpulan halaman yang terdiri dari beberapa halaman yang berisi informasi dalam bentuk data digital baik berupa *text*, gambar, *video*, *audio*, dan animasi lainnya yang disediakan melalui jalur koneksi *internet* (Rohi Abdulloh, 2015;32)

II.2.5. *HTML*

HTML adalah kepanjangan atau singkatan dari *Hyper Text Markup Language*. Setiap penjelajhan di *internet*, membuka *website* apapun, pasti akan

bertemu dengan *HTML*. Hal ini dikarenakan semua *website* yang ada di *internet*, dibuat dengan menggunakan bahasa *markup HTML*. (Irsyad Djamaludin dan Agus Nursikuagus, 2017; 673)

II.2.6 *PHP*

PHP adalah kependekan dari *PHP Hypertext Preprocessor*, bahasa *interpreter* yang mirip dengan Bahasa *C* dan *Perl* yang memiliki kesederhanaan perintah. *PHP* dapat digunakan bersama dengan *HTML* sehingga memudahkan dalam pembangunan aplikasi *web* dengan cepat. (Irsyad Djamaludin dan Agus Nursikuagus, 2017; 673)

II.2.7. *Database*

Basisdata atau *database* adalah kumpulan data yang disimpan secara sistematik didalam komputer. Basisdata juga dapat diolah atau dimanipulasi menggunakan perangkat lunak atau program aplikasi untuk menghasilkan suatu informasi. (Windah Supartini dan Hindarto, 2016; 149).

II.2.8 *SQL*

SQL adalah sebuah inti konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data data, yang memungkinkan pada pengoperasian data dikerjakan dengan mudah secara otomatis. (Hendra Agusvianto, 2017; 41).

II.2.9. *MySQL*

MySQL adalah sistem manajemen database *SQL* yang bersifat *Open Source* dan paling populer saat ini. Sistem *Database MySQL* mendukung beberapa fitur seperti *multithreaded, multi-user*, dan *SQL database managemen sistem (DBMS).* (Irsyad Djamaludin dan Agus Nursikuagus, 2017; 673).

II.2.10. *Unified Modeling Language* (UML)

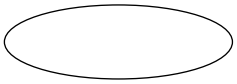
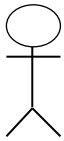
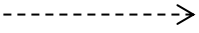
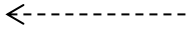
Hasil pemodelan pada *OOAD* terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah satu alat bantu yang sangat handal didunia pengembnagan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam membentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain (Munawar ; 2018 : 49).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

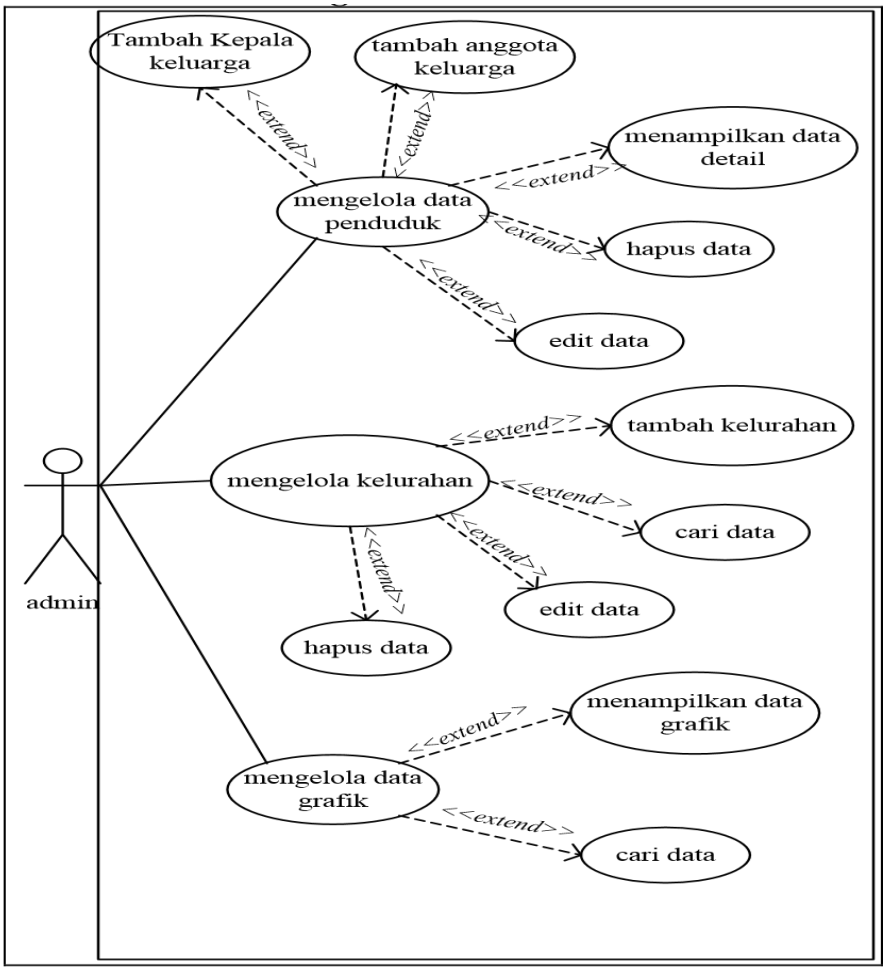
1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan tipikal interaksi antara (pengguna) sebuah *system*dengan sistemnya sendiri melalui sebuah cerita bagaima sebuah sistem yang dipakai (Munawar ; 2018 : 89).

Tabel II.1. Simbol *Use Case Diagram*

Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *Use Case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan 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 *Use Case*, tetapi tidak memiliki *control* terhadap *Use Case*.
	Asosiasi antara aktor dan *Use Case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
	Asosiasi antara aktor dan *Use Case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*, merupakan di dalam *Use Case* lain (*required*) atau pemanggilan *Use Case* oleh *Use Case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*,merupakan perluasan dari *Use Case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Munawar ; 2018 : 93)



Gambar II.1 Contoh kasus *Use Case Diagram;* (*Sumber* :*Winda Aprianti & Umi Maliha 2016:24)*

Use Case diagram yang disajikan pada Gambar mendeskripsiskan interaksi aktor, yaitu admin sistem informasi data penduduk yang dapat mengelola data penduduk, mengelola kelurahan, dan mengelola data grafik. Pengelolaan data penduduk meliputi tambah kepala keluarga, tambah anggota keluarga, menampil

mengedit, dan menghapus data. Sedangkan pengelolaan kelurahan meliputi tambah, cari, *edit*, dan hapus data kelurahan. (Winda Aprianti & Umi Maliha 2016:24)

2. *Class Diagram* (Diagram Kelas)

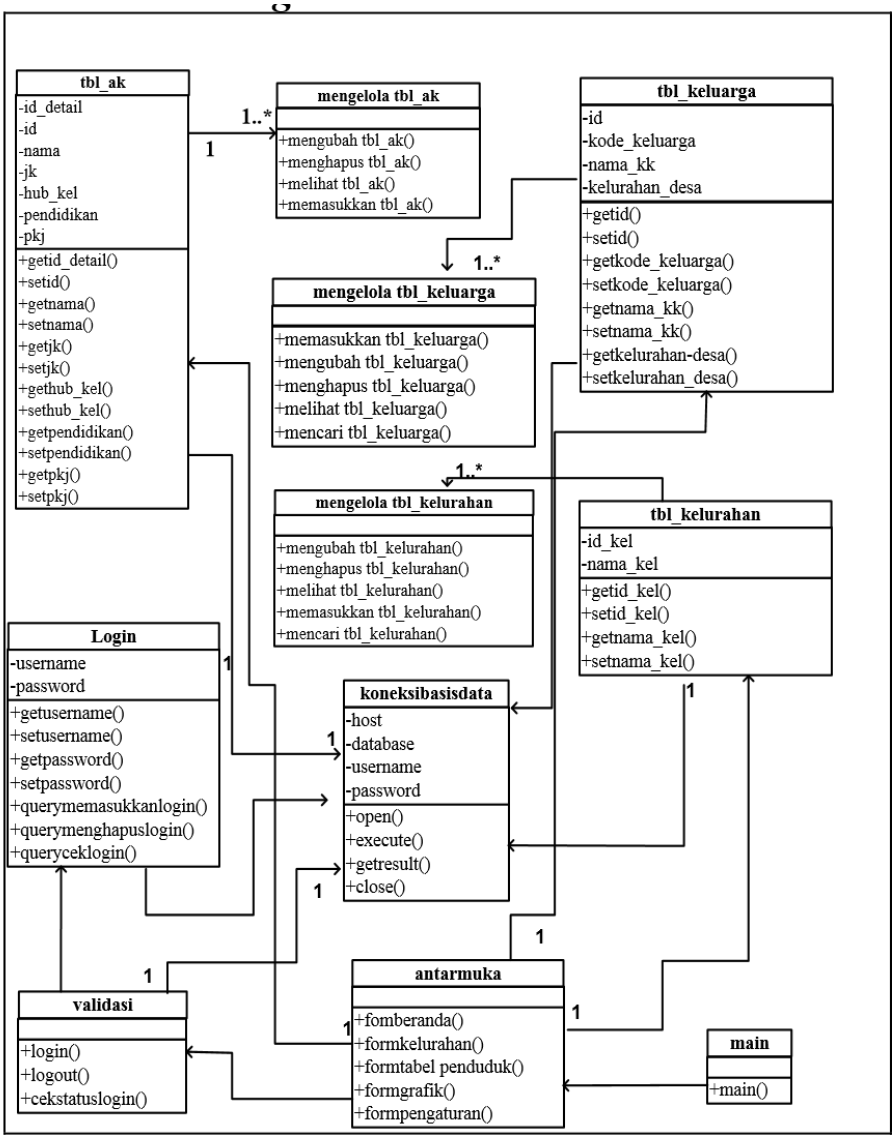
Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram merupakan diagram statis dari suatu aplikasi. Class Diagram tidak hanya digunakan untuk memvisualisasikan, menggambarkan, dan mendokumentasikan berbagai aspek sistem tetapi juga untuk membangun kode eksekusi *(executable code)* dari aplikasi perangkat lunak (Munawar ; 2018 : 101).

Tabel II.2. Simbol *Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber :* Munawar ; 2018 : 101*)



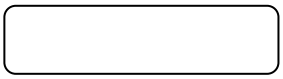
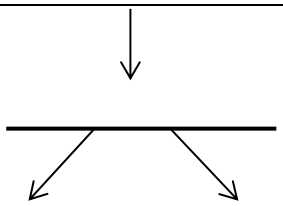
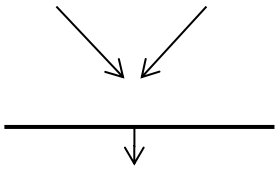
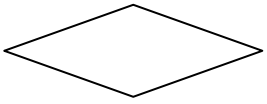
Gambar II.2 Contoh Kasus *Class Diagram* (*Sumber* :*Winda Aprianti & Umi Maliha 2016:24)*

Class diagram disajikan pada Gambar di atas terdiri dari 11 kelas yang meliputi kelas *Main*, Antarmuka, *login*, Koneksi BasisData, *Validasi*, tbl_keluarga, mengelola tbl_keluarga, tbl_kelurahan, mengelola tbl_kelurahan, tbl_ak, dan mengelola tbl_ak. (Winda Aprianti & Umi Maliha 2016:24)

3. Diagram Aktivitas (*Activity Diagram*)

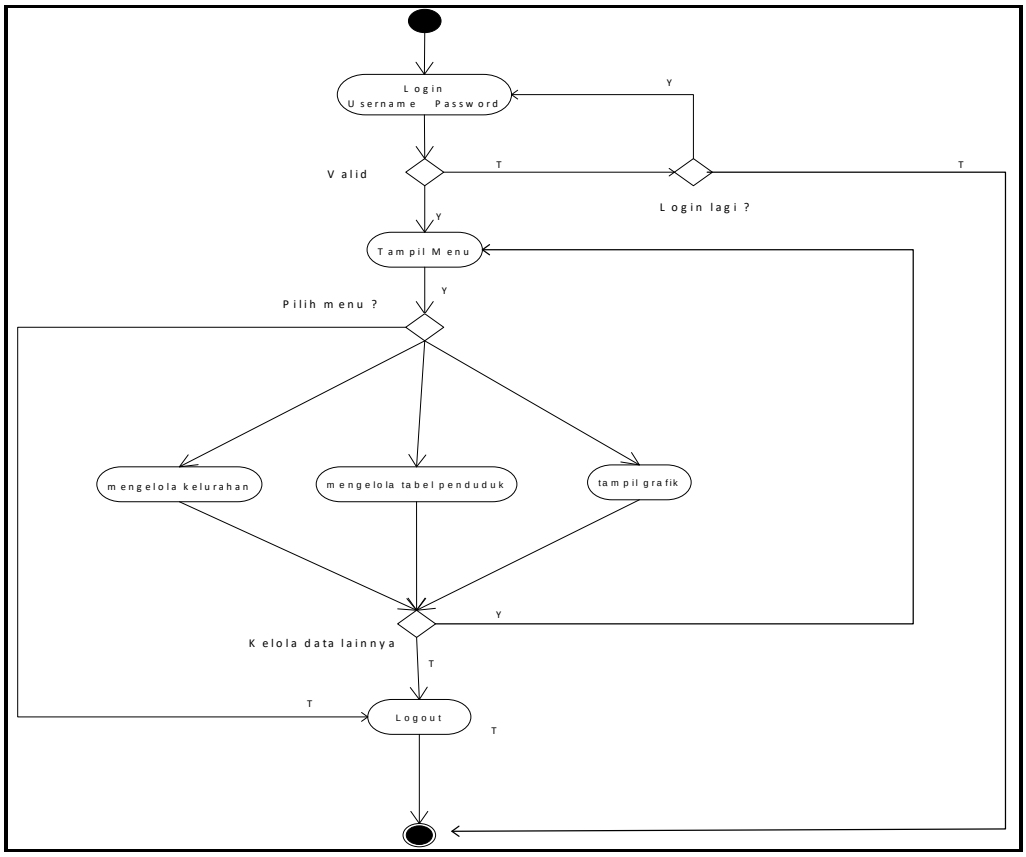
Activity Diagram bagian penting dari UML yang menggambarkan aspek dinamis dari sistem. Logika prosedural, proses bisnis dan aliran kerja suatu bisnis bisa dengan mudah dideskripsikan dalam *activity diagram* (Munawar ; 2018 : 137).

Tabel II.3. Simbol Diagram Aktivitas

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*

New Swimline	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(*Sumber : Munawar ; 2018 : 137*)



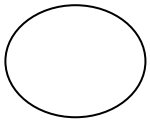
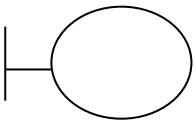
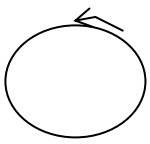
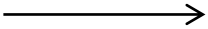
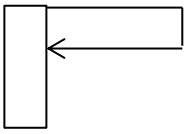
Gambar II.3 Contoh kasus Activity Diagram (*Sumber* :*Winda Aprianti & Umi Maliha 2016:25)*

Activity diagram yang diilustrasikan pada Gambar merupakan diagram yang menggambarkan *workflow* (aliran kerja) dari Sistem Informasi Kepadatan Penduduk di Setiap Kelurahan atau Desa pada Badan Pemberdayaan Masyarakat dan Pemerintah Desa (BPMPD) Studi Kasus pada Kecamatan Bati-Bati Kabupaten Tanah Laut. (Winda Aprianti & Umi Maliha 2016:25)

4. Diagram Urutan (*Sequence Diagram*)

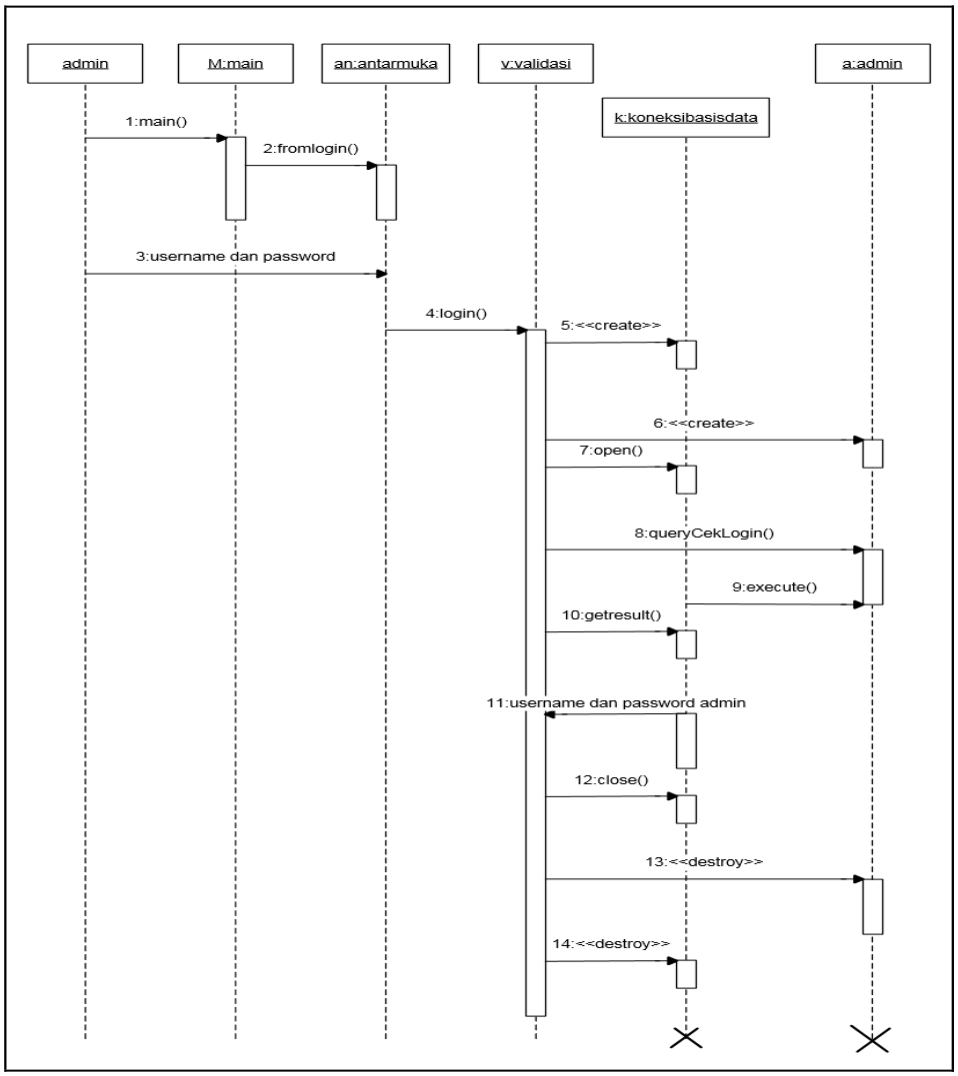
Sequence diagram adalah salah satu *interaction diagram*. Karena sequence diagram mengacu kepada obyek, maka sbelum membuat diagram ini class diagram sudah harus teridentifikasi (Munawar ; 2018 : 186).

Tabel II.4. Simbol Diagram Urutan

Gambar	Keterangan
	*EntityClass*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *formentry* dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.

	*Activation*, mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Munawar ; 2018 : 186)



Gambar II.4 Contoh kasus *Sequence Diagram*; (*Sumber :Winda Aprianti & Umi Maliha 2016:25)*

Urutan proses pada *Sequence login* pada Gambar dimulai dari admin sebagai pengguna yang masuk ke antarmuka untuk masuk ke *form login*. Setelah itu *admini* memasukkan *username* dan *password* menuju validasi untuk *login* setelah itu data dikirim ke *database*. (Winda Aprianti & Umi Maliha 2016:25)