16/LP/FST/02/2017

PENELITIAN

ARSITEKTUR MODEL VIEW CONTROL (MVC) DAN PHP DATA OBJECT DALAM RANCANG BANGUN SISTEM INFORMASI FINANSIAL

Oleh :

SUENDRI, M.Kom
NIP. 19871208 201503 1 003

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN) SUMATERA UTARA
MEDAN
2017

PENELITIAN

ARSITEKTUR MODEL VIEW CONTROL (MVC) DAN PHP DATA OBJECT DALAM RANCANG BANGUN SISTEM INFORMASI FINANSIAL

Oleh :

SUENDRI, M.Kom
NIP. 19871208 201503 1 003

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN) SUMATERA UTARA
MEDAN
2017

KATA PENGANTAR

Puji dan syukur kepada Allah Ta'ala, Tuhan Yang Maha Esa yang telah memberikan pengetahuan, pengalaman, kekuatan, dan kesempatan kepada penulis, sehingga mampu menyelesaikan penyusunan penelitian ini dengan baik.

Penelitian yang berjudul "Arsitektur Model View Control (MVC) Dan PHP Data Object Dalam Rancang Bangun Sistem Informasi Finansial" ini penulis susun sebagai salah satu syarat yang diwajibkan untuk pengajuan syarat memperoleh jabatan fungsional Asisten Ahli pada Universitas Islam Negeri (UIN) Sumatera Utara, Medan.

Penelitan ini adalah hasil karya penulis sendiri, bukan merupakan plagiat, seluruh kutipan yang terdapat pada penelitian ini telah dicantunkan penulis aslinya. Walaupun sudah berupaya maksimal, namun penulis juga menyadari kemungkinan terdapat kekurangan dan kesilapan. Oleh sebab itu, penulis mengharapkan saran-saran dan kritikan yang dapat memperbaiki penelitian ini.

Akhir kata, semoga penelitian ini dapat bermanfaat bagi siapa pun yang membacanya.

Medan, 22 Pebruari 2017

Hormat penulis,

Suendri, M.Kom

NIP. 198712082015031003

REKOMENDASI

Setelah membaca dan menelaah hasil penelitian yang berjudul **"ARSITEKTUR MODEL VIEW CONTROL (MVC) DAN PHP DATA OBJECT DALAM RANCANG BANGUN SISTEM INFORMASI FINANSIAL"**. Yang dilakukan oleh Suendri, M.Kom maka saya berkesimpulan bahwa hasil penelitian ini dapat diterima sebagai karya tulis berupa hasil penelitian. Demikianlah rekomendasi diberikan kepada yang bersangkutan untuk dapat dipergunakan sebagaimana mestinya.

Medan, 22 Pebruari 2017
Konsultan

M. Irwan Padli Nst, MM, M.Kom
NIP. 197502132006041003

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Perkembangan usaha dibidang jasa pemberian kredit di Indonesia semakin tumbuh pesat, terlihat dari bermunculannya perusahaan-perusahaan baru, baik perusahaan besar maupun perusahaan kecil. Untuk mendukung proses transaksi yang dilakukan pada perusahaan tersebut, maka dibuatlah sistem yang dapat menyimpan seluruh data baik debitur, fasilitas yang digunakan, pinjaman hingga pembayaran. Berbagai software sudah dibuat oleh penyedia layanan perangkat lunak atau programmer. Namun seiring semakin kompleknya permasalahan dan semakin banyak data yang disimpan, dibutuhkan sistem yang handal, mampu memberikan respon cepat dan yang mudah dalam penggunaan untuk berbagai DBMS. Sistem Informasi finansial yang diusulkan dirancang menggunakan bahasa pemrogram PHP dengan arsitektur *Model View Controller* (MVC) dan *database MySQL*. Sistem diharapkan dapat menggantikan sistem manual, lebih mudah dalam proses transaksi, lebih cepat dan kuat serta mudah dalam proses migrasi antar DBMS. Dalam perancangan sistem pada penelitian ini, penulis menggunakan pemodelan UML (*Unified Modelling Language*) agar perancangan lebih terarah dan mudah dipahami.

Kata kunci: Sistem, Finansial, PHP, MVC, MySQL

ABSTRACT

Business development services in the field of lending in Indonesia is growing rapidly, as seen from the emergence of new companies, both large companies and small companies. To support the transaction process at the company, then made a system that can store all data both debtors, the facilities, the loan until repayment. Various software has been created by a software services provider or programmer. But as more and more complex the problem, and the more data stored, needed a reliable system, capable of providing instant response and easy to use for different DBMS. Proposed financial information systems designed using the PHP programming language with the Model View Controller (MVC) and MySQL database. The system is expected to replace the manual system, it is easier in the transaction process, faster and stronger and easier process migration between DBMS. In designing the system in this reseach, the authors used modeling UML (Unified Modeling Language) in order to design more effective and easier to understand.

Keywords: System, Financial, PHP, MVC, MySQL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pada saat ini perkembangan usaha di Indonesia semakin tumbuh pesat, terlihat dari bermunculannya perusahaan-perusahaan baru, baik perusahaan besar maupun perusahaan kecil. Diantara perusahaan tersebut sebagian bergerak dibidang jasa pemberian kredit, baik untuk Usaha Kecil Menengah (UKM) maupun untuk kebutuhan hidup lainnya. Hal ini menyebabkan persaingan diantara perusahaan semakin ketat, masing-masing menonjolkan produk yang dihasilkan, mulai dari proses yang cepat, bungan ringan hingga tanpa agunan.

Untuk mendukung proses transaksi yang dilakukan pada perusahaan tersebut, maka dibuatlah sistem yang dapat menyimpan seluruh data baik debitur, fasilitas yang digunakan, pinjaman hingga pembayaran. Berbagai *software* sudah dibuat oleh penyedia layanan perangkat lunak atau *programmer*. Namun seiring semakin kompleknya permasalahan dan semakin banyak data yang disimpan, dibutuhkan sistem yang handal, mampu memberikan respon cepat dan yang mudah dalam penggunaan untuk berbagai DBMS.

Dari latar belakang diatas, penulis berkeinginan merancang sistem informasi sistem informasi finansial yang tertuang dalam penelitian yang berjudul "Arsitektur Model View Control (MVC) Dan PHP Data Object Dalam Rancang Bangun Sistem Informasi Finansial". Penulis berharap dengan adanya sistem ini mampu

memberikan kontribusi yang signifikan dalam proses transaksi finansial dengan menggunakan metode mutakhir saat ini.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, maka ditemukan rumusan masalah sebagai berikut:

a. Bagaimanakah merancang UML Arsitektur *Model View Controller* dan PHP Data *Object*?
b. Bagaimanakah perancangan sistem informasi finansial menggunakan Arsitektur *Model View Controller* dan PHP Data Object?

1.3 Batasan Masalah

Agar pembahasan lebih fokus pada penulisan laporan penelitian ini, maka penulis hanya membahas perancangan sistem informasi finansial menggunakan Arsitektur *Model View Controller* dan PHP Data *Object.*

a. Sistem dirancang menggunakan bahasa pemrograman PHP dan *database* MySQL.
b. Sistem mencakup data transaksi finansial berupa pemberian kredit dan pembayaran cicilan.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah perancangan sistem informasi finansial menggunakan Arsitektur Model View Controller dan PHP Data Object. Secara jelas tujuan penelitian ini adalah sebagai berikut:

a. Untuk memahami arsitektur *Model View Controller* dan PHP Data *Object*.
b. Untuk merancang sistem informasi finansial menggunakan bahasa pemrograman PHP dan MySQL.
c. Menguji sistem yang dibuat untuk data transaksi finansial berupa pemberian kredit dan pembayaran cicilan.

1.5 Manfaat Penelitian

Manfaat penelitian ini mencakup:

a. Sistem diharapkan lebih stabil dan cepat karena menggunakan Arsitektur *Model View Controller*.
b. Pengguna sistem diharapkan lebih terbantu jika terjadi perobahan *Database Management Sistem* (DBMS), karena menggunakan PDO akan memudah proses migrasi antar *database*.

1.6 Sistematika Penulisan

Sistematika penulisan penelitian ini adalah sebagai berikut :

BAB I. PENDAHULUAN

Bab ini berisi tentang latar belakang masalah, batasan masalah, perumusan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

BAB II. LANDASAN TEORI

Bab ini berisi tentang teori yang berhubungan dengan sistem informasi, database, UML, Arsitektur *Model View Controller* (MVC) dan PHP Data *Object* serta Perangkat lunak yang digunakan dalam merancang program.

BAB III. METODOLOGI PENELITIAN

Bab ini berisi tentang metode yang digunakan dalam penelitian serta kerangka kerja penelitian

BAB IV. PERANCANGAN SISTEM

Bab ini berisi tentang langkah-langkah perancangan sistem, pemodelan dan penjelasannya

BAB V. IMPLEMENTASI SISTEM

Bab ini berisi tentang penerapan sistem dan penjelasannya serta tabel jalan sistem yang diharapkan.

BAB VI. PENUTUP

Bab ini berisi tentang kesimpulan dan saran-saran.

BAB 2
LANDASAN TEORI

2.1 Perancangan Sistem Informasi

2.1.1 Perancangan

Perancangan adalah merancang atau mendesain suatu sistem yang baik, yang isinya adalah langkah-langkah dalam proses pengolahan data dan prosedur untuk mendukung operasi sistem. Menurut Abdul Kadir (2007), perancangan adalah proses penerapan berbagai teknik dan prinsip dengan tujuan untuk mentransformasikan hasil analisa kedalam bentuk yang mudah diimplementasikan.

2.1.2 Sistem

Istilah sistem berasal dari bahasa Yunani yaitu *Systema* yang mengandung arti kesatuan atau keseluruhan. Berarti sistem adalah sekumpulan objek-objek yang bekerja sama untuk menghasilkan satu kelompok, prosedur, teknik yang digunakan dan diatur sedemikian rupa satu kesatuan yang berfungsi untuk mencapai tujuan tertentu. Sistem merupakan salah satu yang terpenting dalam sebuah perusahaan yang dapat membentuk kegiatan usaha untuk mencapai kemajuan dan target yang dibutuhkan (Siti Fatima, 2013). Berikut ini adalah defenisi sistem menurut beberapa ahli:

1. Sistem merupakan suatu himpunan dari berbagai bagian atau elemen, yang saling berhubungan secara terorganisasi atau teratur berdasar fungsi-fungsinya menjadi satu kesatuan menuju tujuan tertentu. (Bambang Hartono, 2013 : 10).

2. Sistem adalah jaringan dari pada elemen-elemen yang saling berhubungan, membentuk satu kesatuan untuk melaksanakan suatu tujuan pokok dari sistem tersebut. (Jogiyanto, 2005 : 4).

Secara umum sistem adalah suatu kesatuan yang terdiri dari bagian-bagian yang saling berinteraksi dan bekerjasama untuk mencapai sasaran (*goal*). Dari defenisi di atas dapat diartikan sistem sebagai suatu proses berkelanjutan dari sekumpulan data yang saling berhubungan untuk tujuan bersama yang ingin dicapai serta sistem merupakan jaringan dari elemen-elemen yang saling berhubungan membentuk satu kesatuan untuk melaksanakan suatu tujuan pokok dari sistem tersebut.

2.1.3 Karakteristik Sistem

Model umum sebuah sistem terdiri dari *input*, proses dan *output*. Selain itu sebuah sistem juga memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai sistem. Adapun karakteristik sistem (Siti Fatima, 2015) adalah sebagai berikut:

1. Komponen Sistem (*Components*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-kompoonen sistem tersebut dapat berupa suatu bentuk subsitem.

2. Batasan Sistem (*Boundary*)

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem lainnya atau sistem dengan lingkungan luarnya.

3. Lingkungan Luar Sistem (*Environtment*)

   Lingkungan luar sistem adalah apapun yang ada diluar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut.

4. Penghubung Sistem (*Interface*)

   Penghubung sistem yaitu media yang menghubungkan antara satu subsistem dengan sub sistem yang lain.

5. Masukan Sistem (*Input*)

   Masukan sistem merupakan energi yang dimasukan ke dalam sistem.

6. Keluaran Sistem (*Output*)

   Keluaran sistem merupakan hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna.

7. Pengolah Sistem (*Proses*)

   Pengolah sistem merupakan suatu proses yang akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem (*Objective*)

   Suatu sistem memiliki tujuan dan sasaran yang pasti jika sistem tidak memiliki sasaran maka sistem tidak ada gunanya.

2.1.4 Informasi

Informasi berarti data yang telah dibentuk kedalam suatu format yang mempunyai arti dan guna bagi manusia. Sebaliknya, data merupakan sekumpulan baris fakta yang mewakili peristiwa yang terjadi pada organisasi atau pada lingkungan fisik sebelum diolah kedalam suatu format yang dapat dipahami dan digunakan orang. Jadi

informasi adalah hasil dari pengolahan data kedalam bentuk yang lebih berguna dan lebih berarti yang membutuhkannya.

Informasi adalah hasil dari kegiatan pengolahan data yang memberikan bentuk yang lebih berarti dari suatu kejadian (Jogiyanto, 2005 : 3). Informasi adalah data yang telah diolah menjadi suatu bentuk yang berguna bagi penerimanya dan memiliki nilai bagi pengambilan keputusan disaat ini atau dimasa yang akan dating. (Bambang Hartono, 2013 : 15).

Sumber informasi adalah data. Data merupakan bentuk jamak dan bentuk tunggal atau data-data item. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian (*event*) suatu yang terjadi pada saat tertentu.

2.1.5 Kualitas Informasi

Kualitas dari suatu informasi tergantung dari tiga hal (Siti Fatima, 2015) yaitu :

1. Akurat

   berarti informasi harus bebas dari kesalahan-kesalahan dan tidak biasa atau menyesatkan. Informasi harus mencerminkan maksudnya, informasi harus akurat karena dari sumber informasi sampai ke penerima informasi kemungkinan banyak terjadi ganguan (*noise*) yang dapat merubah atau merusak informasi tersebut.

2. Tepat pada waktunya

   Berarti informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah using tidak ada nilainya lagi, karena informasi merupakan landasan dalam pengambilan keputusan.

3. Relevan

Informasi yang didapat mempunyai manfaat untuk pemakainya. Relevansi informasi untuk setiap orang berbeda.

2.1.6 Sistem Informasi

Sistem informasi adalah sekumpulan *hardware, software, brainware*, prosedur dan atau aturan yang diorganisasikan secara *integral* untuk mengolah data menjadi informasi yang bermanfaat guna memecahkan masalah dan pengambilan keputusan. Sistem informasi merupakan suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transakasi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan. (Tata Sutabri, 2005:42)

Sistem informasi adalah sekumpulan komponen yang bekerja sama secara terorganisasi dan terpadu dari sejumlah bagian atau komponen yang secara bersama-sama berfungsi dalam pengolahan data untuk memperoleh informasi, dengan maksud dan tujuan tertentu. (Bambang Hartono, 2013 : 20)

Sistem informasi mempunyai komponen-komponen yang disebut istilah blok yang saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasaran (Siti Fatima, 2015) Komponen tersebut terdiri dari:

1. Blok Masukan (*Input Blok*)

Input blok ini mewakili data yang masuk kedalam sistem informasi. Input disini termasuk metode dan media mengambil data yang akan dimasukan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model (*Model Blok*)

Blok ini terdiri dari kombinasi prosedur, logika, dan model matematik yang

akan memanipulasi data dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok Keluaran (*Output Blok*)

Hasil dari sistem informasi adalah keluaran yang merupakan informasikan yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta pemakai sistem.

4. Blok Teknologi (*Technology Blok*)

Teknologi merupakan Kotak Alat (*Tool Box*) dalam sistem informasi karena digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirim keluaran dan membentuk pengendalian dari sistem secara keseluruhan.

5. Blok Busis Data (*Database Blok*)

Busis data merupakan kumpulan daridata yang saling

berhubungan satu dengan yang lainnya, yang tersimpan

diperangkat keras dan dimanipulasi oleh perangkat lunak

komputer.

2.1.7 Perancangan Sistem

Desain sistem menentukan bagaimana suatu sistem akan menyelesaikan apa yang mesti diselesaikan tahap ini menyangkut konfigurasi dari komponen-komponen perangkat lunak dan perangkat

kerja dari suatu sistem, sehingga setelah instalasi dari sistem akan benar-benar memuaskan rancang bangun yang telah ditetapkan pada akhir tahap analisis sistem. Desain sistem dapat didefinisikan sebagai penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi. Perancangan sistem membutuhkan suatu sistem yang akan dibentuk. Perancangan sistem mempunyai dua tujuan yaitu:

a. Untuk memenuhi kebutuhan pemakai sistem.
b. Untuk memberikan gambaran yang jelas rancang bangun yang lengkap kepada pemrograman komputer dan ahli-ahli teknik lainnya.

Untuk mencapai tujuan ini haruslah dapat mencapai sasaran-sasaran yaitu:

a. Perancangan sistem harus berguna, mudah dipahami dan nantinya mudah digunakan dan Perancangan harus dapat mendukung tujuan utama suatu perusahaan.
b. Perancangan sistem harus cepat dan tepat untuk dapat mendukung pengolahan transaksi.

2.2 *Database*

Database atau basis data adalah koleksi data yang bisa mencari secara menyeluruh dan secara sistematis memelihara informasi (Janner, 2007:2). Sedangkan menurut Abdul Kadir (2014:218), basis data (*database*) adalah pengorganisasi sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data dimaksudkan untuk mengatasi problem pada sistem yang memakai

pendekatan berbasis berkas. Untuk mengelola database diperlukan perangkat lunak yang disebut *Database Management System.* DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol dan mengakses basis data dengan cara yang praktis dan efisien. DBMS dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda.

2.2.1 Komponen *Database* (Basis Data)

Apabila kita lihat komponen di dalam basis data, maka dapat disebutkan bahwa:

a. *Basis* data terdiri dari beberapa *file.*
b. *File* terdiri dari beberapa *record.*
c. *Record* terdiri dari *field.*
d. *Field* terdiri dari beberapa karakter.

Karakter merupakan bagian data yang terkecil dapat berupa karakter *numeric*, huruf maupun karakter khusus yang membentuk suatu data item (*field*). *Field* adalah kumpulan data-data *record* yang sejenis, yang merupakan kumpulan data untuk mewakili suatu *entity* data *record*. *Record* merupakan kumpulan dari *field* membentuk *record*.

Record menggambarkan unit dari data individu tertentu. *File* merupakan kumpulan dari *record-record* yang sejenis yang mempunyai panjang elemen yang sama, atribut yang sama tapi berbeda nilainya. *File* terdiri dari *record-record* yang menggambarkan suatu

kesatuan data yang sejenis. Dengan menggunakan sistem basis data masalah pada manajemen basis data dapat berkurang.

Dengan sistem basis data juga dapat mengurangi duplikasi data dengan tujuan untuk mengurangi biaya manajemen. Basis data dapat dibuat cukup fleksibel dalam arti mudah ditambah atau dikurangi bahkan dimodifikasi, dan sistem basis data dapat menghubungkan dengan data lainnya.

2.3 Teknik Normalisasi

Proses normalisasi merupakan proses pengelompokan elemen data menjadi table-tabel yang menunjukan entitas dan relasinya. Proses ini selalu diuji pada beberapa kondisi. Apakah ada kesulitan pada saat menambah (*insert*), menghapus (*delete*), mengubah (*update*), atau membaca (*retrieve*) pada suatu database. Bila ada kesulitan pada pengujian tersebut maka relasi maka relasi dapat dipecah dalam beberapa tabel (Tata Sutabri, 2012:138).

2.3.1 Tujuan Normalisasi :

a. Untuk menghilangkan kerangkapan data
b. Untuk mengurangi kompleksitas
c. Untuk mempermudah pemodifikasian data

2.3.2 Proses Normalisasi

a. Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu ke beberapa tingkat.

b. Apabila tabel yang diuji belum memenuhi persyaratan tertentu, maka tabel terebut perlu dipecah menjadi beberapa tabel yang lebih sederhanan sampai memenuhi bentuk yang optimal.

2.3.3 Bentuk Normalisasi

Bentuk-bentuk normalisasi yang ada adalah seperti berikut ini :

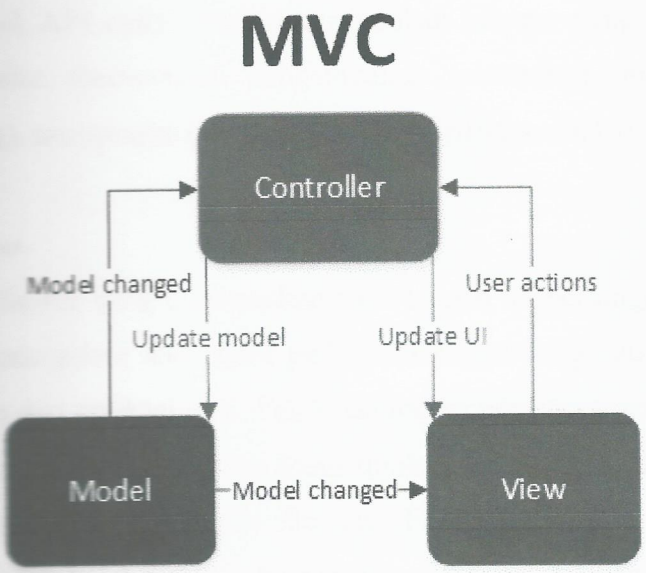
a. Bentuk Tidak Normal, menghilangkan perulangan *group*.
b. Bentuk Normal Pertama (1NF), menghilangkan ketergantungan sebagian.
c. Bentuk Normal Kedua (2NF), menghilangkan ketergantungan transitif.
d. Bentuk Normal Ketiga (3NF), menghilangkan anomali-anomali hasil dari ketergantungan fungsional.
e. Bentuk Normal Boyce-Codd (BCNF), menghilangkan Ketergantungan *Multivalue*.
f. Bentuk Normal Keempat (4NF), menghilangkan anomali-anomali yang tersisa.
g. Bentuk Normal Kelima (5NF).

2.4 *Model View Controller* (MVC)

Model View Controller (MVC) pertama kali diperkenalkan peneliti Xerox PARC yang bekerja pada bahasa pemrograman Smalltalk di akhir tahun 1970-an dan awal 1980-an. Smalltalk adalah bahasa pemrograman yang berorientasi objek, bertipe dinamis, dan reflektif. Smalltalk pertama kali digunakan dalam pembelajaran aplikasi, dan hal ini berbeda dari data mainframe dan struktur kontrol dalam program Smalltalk yang terlibat pada Windowed User

Interfaces, konsep pemrograman berorientasi objek, pengantar pesan antara komponen- komponen objek, dan kemampuan untuk memonitor dan memodifikasi struktur dan perilakunya sendiri. (Myer : 2008)

Singkatnya, Model-View-Controller (MVC), adalah pola desain pengembangan perangkat lunak. MVC adalah sebuah penilekatan untuk memisahkan aplikasi menjadi tiga segmen, yaitu Models, Views, dan Controller. MVC menstrukturisasi aplikasi dengan cara tersebut untuk mempromosikan penggunaan kembali dari kode program. (Griffiths : 2010)



Gambar 2.1. MVC

Berdasarkan gambar diatas dapat kita ketahui bahwa ketika datang sebuah permintaan dari user, maka permintaan tersebut akan ditangani oleh *Controller*, kemudian *Controller* akan memanggil

Model jika memang diperlukan operasi database. Hasil *query* oleh Model kemudian akan dikembalikan ke *Controller*. Selanjutnya *Controller* akan memanggil *View* yang tepat dan mengkombinasikannya dengan hasil *query* Model. Hasil akhir dari operasi ini akan ditampilkan ke *browser* yang selanjutnya bisa dilihat oleh *user*.

2.4.1 *Model*

Model merupakan jenis data yang dapat digunakan oleh aplikasi. Beberapa contoh data yang biasa digunakan adalah database, RSS *Feed*, API *calls*, dan setiap tindakan lainnya yang melibatkan pengambilan (retrieving), pengembalian (returning), memperbarui (updating), menghapus (removing) data. (Griffiths : 2010)

2.4.2 *View*

File-file yang ditempatkan pada bagian ini bertanggung jawab untuk menunjukan data kami pada para pengunjung situs kita, atau pengguna dari aplikasi kita. Tidak ada logika pemrograman, tidak ada query insert atau update yang harus dijalankan disini, meskipun akses data bisa terjadi pada file-file ini. File-file disini hanya untuk menunjukkan hasil dari dua bagian lainnya. Jadi kita mengambil data dalam model, dan menampilkan dalam view. (Blanco : 2009).

2.4.3 *Controller*

Controller adalah logika bisnis dari aplikasi. File-file yang ada di sini akan melayani sebagai perantara antara Model dan Views.

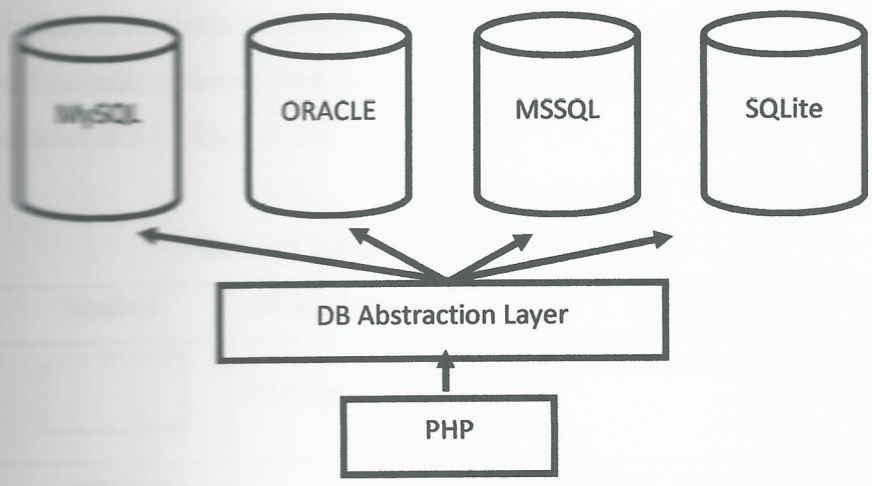
Controller akan merespon permintaan HTTP dan menghasilkan halaman web. Controller adalah inti dari aplikasi karena bagian menentukan bagaimana permintaan HTTP harus ditangani. (Griffiths : 2010)

2.5 PHP Data *Object*

PDO adalah extension baru untuk PHP 5.0 ke atas untuk melakukan manajemen database. PDO memberikan juga menyertakan sekumpulan driver untuk dapat bekerja pada berbagai perangkat lunak database yang berbeda. PDO dikembangkan agar dapat memberikan interface yang ringan untuk perangkat database yang berbeda. Keunggulan lainnya yaitu dapat memberikan penanganan error yang lebih baik, serta dapat mengeksekusi multiple query lebih cepat. Cara kerja PDO sama seperti Data Access Layer dimana dapat digunakan nama fungsi yang sama untuk semua perangkat database (Welling, 2008).

Setiap basis data mempunyai cara tersendiri didalam melakukan pengaksesan datanya, perbedaan ini menyebabkan terlalu sulit untuk berpindah dari suatu database ke database yang lainnya. Akan tetapi terkadang kita membutuhkan beberapa tipe database untuk keperluan tertentu pada suatu aplikasi yang sama. Misalnya perusahaan anda secara resmi menggunakan MySQL, kemudian terdapat beberapa aplikasi yang secara ekslusif harus dijalan di ORACLE. Maka apakah perusahaan anda akan beralih menggunakan ORACLE dengan mengeluarkan anggaran yang besar? Dan harus membangun kembali sistem yang sudah ada dengan database yang baru? Untuk mengatasi

masalah ini mulailah para developer mengembangkan abstraksi layer untuk melayani aplikasi didalam berkomunikasi dengan berbagai macam sistem basisdata.



Gambar 2.2. PDO

2.6 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software* (http://www.omg.org). Diagram *Unified Modelling Language* (UML) (Siti Fatima, 2015) antara lain sebagaui berikut:

2.6.1 *Use Case Diagram*

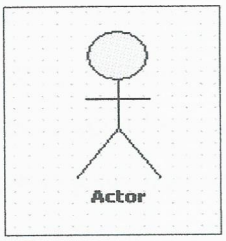
Use Case adalah rangkaian atau uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. *Use Case* digunakan untuk membentuk tingkah-laku benda atau *things* dalam sebuah model serta direalisasikan oleh sebuah *collaboration*. *Use Case* memiliki beberapa simbol seperti pada tabel 2.6 dibawah ini.

Tabel 2.1 Simbol *Use Case Diagram*

Simbol	Keterangan	Fungsi
	*Package*	Menambahkan paket baru dalam diagram.
	*Use Case*	Menambahkan *use case* dalam diagram.
	*Actors*	Menambahkan *actor* dalam diagram.
	Unidirectional Association	Menggambarkan relasi antara *actor* dengan *use case*.
	Dependencies or Instantiates	Menggambarkan kebergantungan *(dependencies)* antar *item* dalam diagram.
	*Generalization*	Mengggambarkan relasi lanjut antar *use case* atau menggambarkan struktur pewarisan antar *actor*.

a. *Actor*

Pada dasarnya *actor* bukanlah bagian dari *use case diagram*, namun untuk dapat terciptanya suatu *use case diagram* diperlukan beberapa *actor*. *Actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah *actor* mungkin hanya memberikan informasi masukan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima, dan memberi informasi pada sistem. A*ctor* hanya berinteraksi dengan *use case,* tetapi tidak memiliki kontrol atas *use case*. *Actor* digambarkan dengan *stick man* . *Actor* dapat digambarkan secara secara umum atau spesifik, dimana untuk membedakannya kita dapat menggunakan *relationship.*

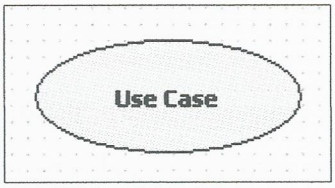


Gambar 2.3 Aktor

b. *Use Case*

Use case **adalah** gambaran fungsionalitas dari suatu sistem, sehingga *customer* atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun. *Use case diagram* adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian. Cara menentukan *Use Case* dalam suatu sistem:

1) Pola perilaku perangkat lunak aplikasi.
2) Gambaran tugas dari sebuah *actor*.
3) Sistem atau "benda" yang memberikan sesuatu yang bernilai kepada *actor*.
4) Apa yang dikerjakan oleh suatu perangkat lunak (bukan bagaimana cara mengerjakannya).



Gambar 2.4 *Use Case*

e. **Relasi dalam *Use Case***

Ada beberapa relasi yang terdapat pada *use case diagram*:

1) *Association*, menghubungkan *link* antar elemen.
2) *Generalization*, disebut juga *inheritance* (pewarisan), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.
3) *Dependency*, sebuah elemen bergantung dalam beberapa cara ke elemen lainnya.
4) *Aggregation*, bentuk *assosiation* dimana sebuah elemen berisi elemen lainnya.

Tipe relasi (*stereotype*) yang mungkin terjadi pada *use case diagram*:

1) <<include>> , yaitu kelakuan yang harus terpenuhi agar sebuah *event* dapat terjadi, dimana pada kondisi ini sebuah *use case* adalah bagian dari *use case* lainnya.

2) <<extends>>, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan alarm.

3) <<communicates>>, mungkin ditambahkan untuk asosiasi yang menunjukkan asosiasinya adalah *communicates association* . Ini merupakan pilihan selama asosiasi hanya tipe *relationship* yang dibolehkan antara *actor* dan *use case*.

2.6.2 *Class* Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu :

1. Nama, merupakan nama dari sebuah kelas.
2. Atribut, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada obyek dari *class.*
3. Operasi, adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class*.

2.6.3 *Statechart diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimulus yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*). Dalam UML, *state* digambarkan berbentuk segi empat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari event tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

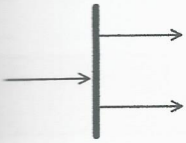
2.6.4 *Activity Diagram*

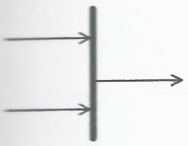
Activity Diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *Activity Diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Activity Diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision yang* mungkin terjadi, dan bagaimana mereka berakhir. *Activity Diagram* juga dapat menggambarkan proses paralel yang

mungkin terjadi pada beberapa eksekusi. *Activity Diagram* memiliki beberapa simbol seperti pada table 2.7 dibawah ini.

Tabel 2.2 Simbol *Activity Diagram*

Simbol	Keterangan	Fungsi
	*State*	Menambahkan *state* untuk suatu objek
	*Activity*	Menambahkan aktivitas baru pada diagram
	*Start Point*	Memperlihatkan dimana aliran kerja Berawal
	*End Point*	Memperlihatkan dimana aliran kerja berakhir
	State transition	Menambah transisi dari suatu aktivitas ke aktivitas yang lainnya
	*Fork* (Percabangan)	Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel

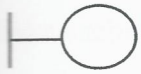
	*Join* (Penggabungan)	Digunakan untuk menunjukkan kegiatan yang digabungkan
	*Decision*	Menambahkan titik keputusan pada aliran kerja

2.6.5 *Sequence* Diagram

Sequence diagram (diagram urutan) adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut termasuk pengguna, *display,* dan sebagainya berupa pesan (*message*). *Sequence Diagram* digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai sebuah respon dari suatu kejadian (*event*) untuk menghasilkan *output* tertentu. *Sequence diagram* memiliki beberapa simbol seperti pada tabel 2.8 dibawah ini.

Tabel 2.3 Simbol *Sequence Diagram*

Simbol	Keterangan	Fungsi
	*Actors*	Menggambarkan orang yang sedang berinteraksi dengan sistem

	*Entity Class*	Menggambarkan hubungan kegiatan yang dilakukan
	*Boundary Class*	Menggambarkan sebuah dari *form*
	*Control Class*	Menggambarkan penghubung antara *boundary* dengan tabel
	*A focus of control and A life line*	Menggambarkan tempat mulai dan berakhirnya *messagge*
	*A messagge*	Mengggambarkan pengiriman pesan

2.6.6 *Collaboration* Diagram

Collaboration Diagram adalah perluasan dari objek dan diagram (objek diagram menunjukkan objek-objek dan hubungannya satu dengan yang lain). *Collaboration Diagram* menunjukkan *message-message* objek yang dikirim satu sama lain.

2.6.7 *Component Diagram*

Component Diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) diantaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun

executable, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

2.6.8 *Package Diagram*

Adalah sebuah bentuk pengelompokkan yang memungkinkan untuk mengambil sebuah bentuk di UML dan mengelompokkan elemen-elemennya dalam tingkatan unit yang lebih tinggi. Kegunaan *package* yang paling umum adalah untuk mengelompokkan *class*.

2.6.9 *Deployment* Diagram

Deployment atau *physical diagram* menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah *server, workstation* atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya.

2.7 Perangkat Lunak Yang Digunakan

2.7.1 Bahasa Pemrograman *PHP*

PHP pertama kali dibuat oleh Rasmus Ledorf pada tahun 1995, pada waktu itu PHP bernama FI9 (*Form Interpreted*). PHP adalah sekumpulan script yang digunakan untuk mengelola Data form dari

web. perkembangan selanjutnya adalah adalah Rasmus melepaskan kode sumber tersebut kepanjangan dari PHP/FI adalah personal homepage/from interpreter. Dengan pelepasan kode sumber ini menjadi opensource, maka banyak programmer pada rilis ini interpreter sudah diimplementasikan dalam C.

Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan *PHP/FI* secara signifikan. Pada tahu 1997, sebuah perusahaan bernama Zend, menulis ulang *interpreter PHP* menjadi lebih bersih, lebih baik dan lebih cepat. Kemudian pada juni 1998 perusahaan tersebut merilis *interpreter* baru untuk *PHP* dan meresmikan nama rilis tersebut menjadi *PHP* 3.0. Pada pertengahan tahun 1999, Zend, merilis *interpreter PHP* baru dan rilis tersebut dikenal dengan *PHP* 4.0. adalah versi *PHP* yang paling banyak dipakai. Versi ini banyak dipakai sebab versi ini mampu dipakai untuk membangun aplikasi *web* kompleks tetapi memiliki kecepatan proses dan stabilitas yang tinggi. Pada juni 2004, Zend merilis *PHP* 5.0 Versi ini adalah versi mutakhir dari *PHP* dalam versi ini, dari *interpreter PHP* mengalami perubahan besar. Dalam versi ini juga di kenalkan model pemrograman berorientasi objek baru untuk menjawab perkembangan bahasa pemprograman berorientasi objek (Siti Fatima, 2013).

2.7.2 *MySQL*

MySQL adalah *Relational Database Management System* (RDBMS) yang di distribusikan secara gratis dibawah lisensi GPL (General Publik License). Dimana setiap orang bebas untuk

menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Keandalan suatu *system database* (DBMS) dapat diketahui dari kerja optimizer-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh *user* maupun program-program aplikasinya. Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibandingkan *Database server* lainnya dalam *Query* data. Hal ini terbukti untuk *Query* yang dilakukan *Single User*, kecepatan *Query* MySQL bisa sepuluh kali lebih cepat dari *Postgres SQL* dan lima kali lebih cepat dibandingkan *Interbase*.

PhpMyAdmin merupakan bagian untuk mengelola basis data *MySQL* yang ada dikomputer. *phpMyAdmin* juga merupakan *MySQL Client* berbasis web, atau program yang dapat digunakan untuk mengakses database *MySQL* melalui browser. Sebelum memulai halaman *MySQL* pastikan *webserver* (*XAMPP)* sudah dijalankan. Untuk membukanya, buka *browser* lalu ketikkan alamat *http://localhost/phpMyAdmin*, maka akan muncul halaman *phpMyAdmin*.

Bagian kiri halaman berisi daftar database bawaan server MySQL, Sedangkan pada bagian kanan berisi menu dan *tools* untuk melakukan operasional manajemen database. Lewat fasilitas yang disediakan *phpMyAdmin* inilah bisa membuat database baru, membuat

tabel baru, mengganti dan mengahpus tabel juga *database.* Selain itu semua operasi manajemen data juga bisa dilakukan, yaitu memasukan data, menampilkan dan menghapus data (Siti Fatima, 2013).

2.9.3 *XAMPP*

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache HTTP Server*, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan *Perl*. Nama *XAMPP* merupakan singkatan dari X (empat sistem operasi apapun), *Apache, MySQL*, *PHP* dan *Perl* (Siti Fatima, 2013).

XAMPP adalah singkatan dari masing-masing hurufnya yaitu:

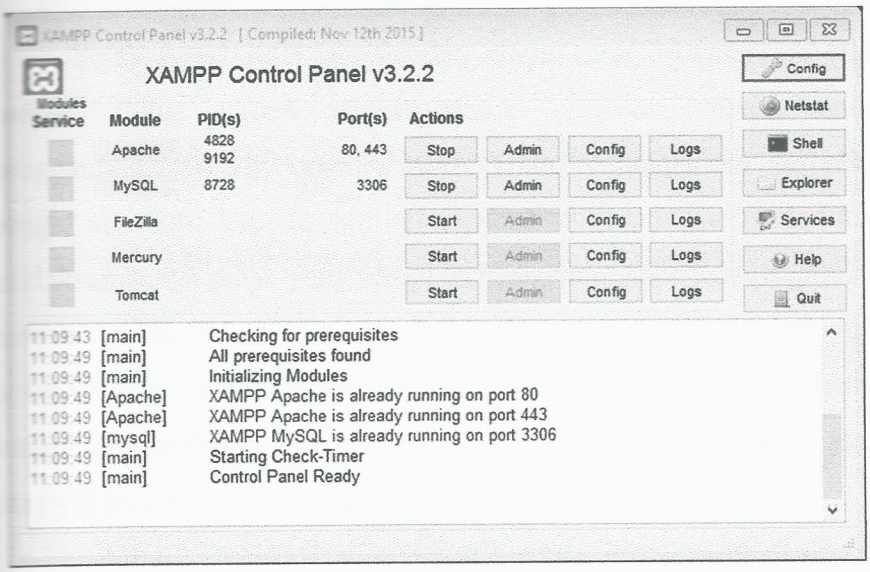
a. **X** adalah Program ini dapat dijalankan dibanyak sistem operasi, seperti *Windows*, *Linux*, MacOS, dan *Solaris*.
b. **A** adalah *Apache,* merupakan aplikasi *webserver*. Tugas utama *Apache* adalah menghasilkan halaman *web* yang benar kepada user berdasarkan kode *PHP* yang dituliskan oleh pembuat halaman *web*. Jika diperlukan juga berdasarkan kode *PHP* yang dituliskan, maka dapat saja suatu *database* diakses terlebih dahulu (misalnya dalam *MySQL*) untuk mendukung halaman *web* yang dihasilkan.
c. **M** adalah *MySQL*, merupakan aplikasi *database server*. Perkembangannya disebut *MySQL* yang merupakan kepanjangan dari *Structured Query Language*. *MySQL* merupakan bahasa terstruktur yang digunakan untuk mengolah database. *MySQL* dapat digunakan untuk membuat dan mengelola database beserta

isinya. Kita dapat memanfaatkan *MySQL* untuk menambahkan, mengubah, dan menghapus data yang berada dalam *database.*

d. **P** adalah *PHP*, bahasa pemrograman *web*. Bahasa pemrograman *PHP* merupakan bahasa pemrograman untuk membuat *web* yang bersifat *server-side scripting*. *PHP* memungkinkan kita untuk membuat halaman *web* yang bersifat dinamis. Sistem manajemen basis data yang sering digunakan bersama *PHP* adalah *MySQL*. Namun *PHP* juga mendukung sistem *manajement database Oracle, Microsoft Access, Interbase, d-base, PostgreSQL,* dan sebagainya.

e. **P** adalah *Perl,* adalah bahasa pemrograman.

Bagian penting *XAMPP* yang biasa digunakan pada umumnya *control panel* yang berfungsi untuk mengelola layanan *service XAMPP* Seperti menghentikan *stop* layanan, ataupun memulai *start*.



Gambar 2.5 *XAMPP*

2.7.4 *Web Browser*

Web Browser merupakan software yang berfungsi untuk menampilkan web di internet, web browser sangat penting karena tanpa web browser kita tidak dapat membuka halaman website internet. Web browser yang digunakan harus mendukung software yang diguanakan dalam mendesain web, yang tergolong kedalam web browser seperti internet explorer, mozila, firefox, opera, google chrome dan lain-lain.

Dalam hal ini penulis menggunakan *mozila firefox*. *Browser* ini mendukung berbagai fasilitas yang memuat halaman *web* menjadi lebih hidup. Dan *mozila firefox* menjadi *browser* yang paling banyak digunakan untuk membuka halaman *web*, dibandingkan *web browser* lainnya. *Mozila firefox* (aslinya bernama *Phonoenix* dan kemudian untuk sesaat dikenal sebagai *Mozila Firebird*) adalah peramban *web* lintas *platform* gratis yang dikembangkan oleh yayasan *Mozila* dan ratusan sukarelawan.

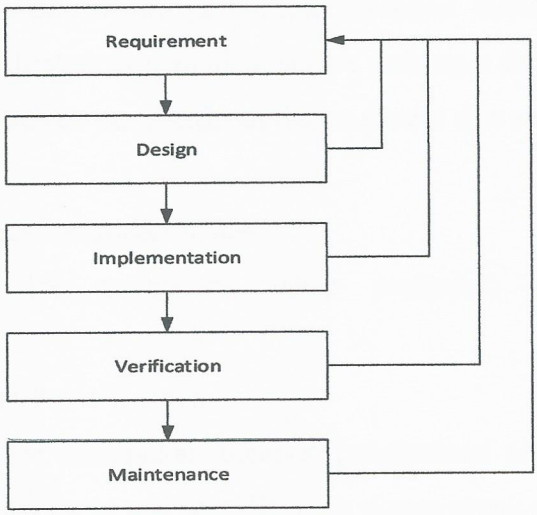
Sejak April 2003, *firefox* dan *client surel Thunderbird* telah menjadi fokus utama pengembangan yayasan *mozila* untuk menggantikan *mozila suite*. *Firefox* telah mendapatkan perhatian sebagai alternatif kepada *internet explorer* dikecam karena tuduhan ketidaksamaanya pihak yang setuju terhadap anggapan ini mengatakan *explorer* tidak mengikuti standar *web*, menggunakan komponen *Active*X yang sering membahayakan, dan kelemahannya terhadap pemasangan *software* dan *hardware* dan kurangnya *fitur-fitur* yang dianggap pemakai *firefox* penting (firefox.com)

BAB III
METODOLOGI PENELITIAN

Pada bab ini akan diuraikan metodologi penelitian dan kerangka kerja penelitian yang digunakan dalam penyelesaian penelitian ini. Kerangka kerja ini merupakan langkah-langkah yang akan di lakukan dalam rangka penyelesaian masalah yang akan dibahas.

3.1 Metode *Waterfall*

Dalam merancang sebuah aplikasi atau sistem, diperlukan metode-metode atau langkah-langkah dalam pembangunan atau pengembangan sistem. Dalam penelitian ini, penulis melakukan pengembangan sistem dengan metode *waterfall.* Metode *waterfall* merupakan metode pengembangan perangkat lunak yang secara umum dilakukan oleh para peneliti sistem.



Gambar 3.1 Metode *Waterfall*

Langkah-langkah *waterfall* (Dina Fitria Murad, 2013) adalah sebagai berikut:

1. Analisis Kebutuhan Sistem

Tahap ini merupakan tahap dalam mencari informasi sebanyak-banyaknya mengenai sistem yang diteliti dengan melakukan metode-metode pengumpulan data sehingga ditemukan kelebihan dan kekurangan sistem serta user requirement. Selain itu, tahap ini juga dilakukan untuk mencari pemecah masalah dan menganalisa bagaimana sistem akan dibangun untuk memecahkan masalah pada sistem sebelumnya.

2. Perancangan Sistem

Tahap ini merupakan tahapan perancangan sistem yang didalamnya dilakukan pemodelan sistem dengan UML, representasi interface dan coding Implementasi dan Pengujian Unit Tahap ini merupakan tahapan dalam pengimplementasian sistem yang sudah dirancang dan dilakukan pengujian secara unit, agar dapat mengetahui kesalahan-kesalahan yang terdapat dalam sistem dan segera dilakukan perbaikan.

3. Integrasi dan Pengujian Sistem

Tahap ini merupakan tahap pengujian sistem secara keseluruhan.

4. Operasi dan Pemeliharaan

Tahap ini merupakan tahapan penggunaan sistem oleh *user* yang didalamnya harus ada pemeliharaan sistem untuk menjaga proses

operasional sistem dan memungkinkan untuk dilakukan pengembangan sistem di kemudian hari

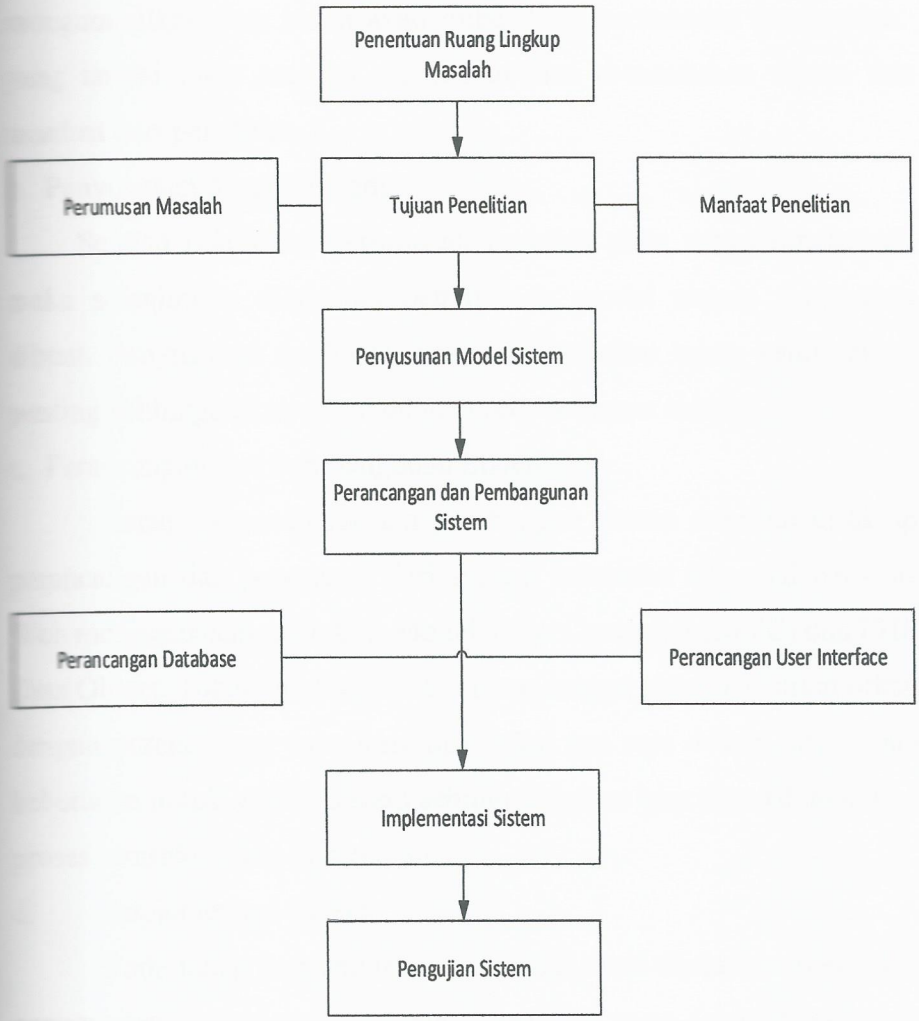
3.2 Kerangka Kerja

Dalam usaha mendapatkan hasil yang optimal dari penelitian ini maka dianggap penting untuk membuat suatu kerangka kerja dalam melakukan penelitian sebagai tolak ukur pencapaian tujuan penelitian.

Kerangka kerja dari penelitian ini dideskripsikan dengan langkah-langkah sebagai berikut :

1. Penentuan ruang lingkup permasalahan
    a. Perumusan masalah
    b. Tujuan penelitian
    c. Manfaat penelitian
2. Penyusunan model sistem
3. Perancangan dan Pembangunan Sistem
    a. Perancangan *Database*
    b. Perancangan *User Interface*
4. Implementasi sistem
5. Pengujian sistem

Deskripsi di atas digambarkan dengan diagram metode penelitian pada gambar 3.2.



Gambar 3.2 Kerangka Kerja

a. Ruang lingkup permasalahan

Langkah yang dilakukan untuk mengidentifikasi permasalahan adalah dengan melakukan pengumpulan data dari objek penelitian dan membandingkan dengan kebutuhan-kebutuhan transaksi kredit. Setelah

mengumpulkan data maka akan didapatkan perumusan permasalah yang terjadi pada saat ini, kemudian bisa menentukan tujuan dan manfaat dari penelitian.

b. Penyusunan Model Sistem

Setelah dilakukan perumusah masalah pada tahap sebelumya, maka selanjutnya dilakukan penyusunan model sistem yang akan dibuat. Penyusunan model sistem ini merupakan tahap yang sangat penting sehingga akan memudahkan proses desain sistem.

c. Perancangan dan Pembangunan Sistem

Tahap perancangan dan pembangun sistem merupakan tahap perancangan dan penerapan dari system informasi finansial berbasis Web menggunakan arsitektur Model View Controller (MVC) dan PHP Data Object. Tahap ini dimulai dari perancangan *database*, dilanjutkan dengan perancangan *user interface* yakni apa saja desain, menu dan kebutuhan untuk sebuah sistem sehingga system bias digunakan dalam proses transaksi yang diinginkan.

d. Implementasi sistem

Pada tahap implementasi sistem ini akan dijelaskan mengenai penggunaan dari sistem informasi finansial yang telah di rancang. Penjelasan ini mencakup tampilan sistem, fungsi dan kontrol aplikasi dan serta bagian-bagian yang dirancang.

e. Pengujian Sistem

Tahap terakhir pada penelitian ini adalah pengujian atau ujicoba sistem. Pengujian dilakukan menggunakan komputer dengan spesifikasi sebagai berikut :

1. *Hardware*

1) *PC Processor Intel Core 1,8 GHz*
2) *Harddisk 500 GB*
3) *RAM 4 GB*
4) *Intel Graphic Onboardb 1 GB*

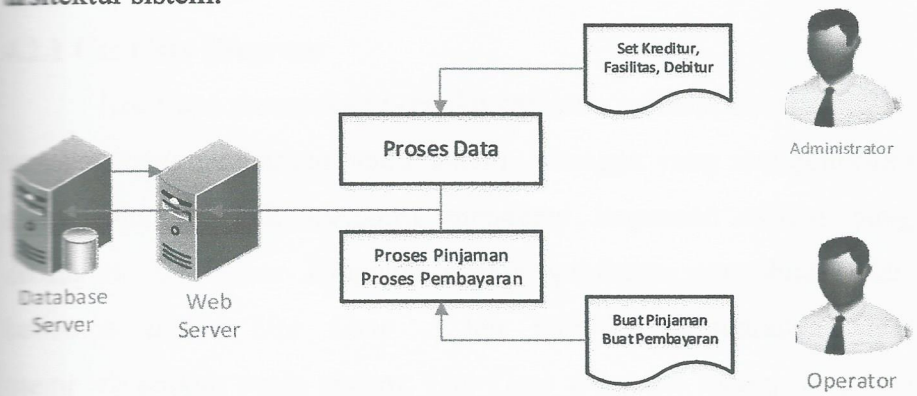
2. *Software*

1) *Windows* 10 *Professional*
2) *Mozilla Firefox 38.0*
3) *XAMPP 7.0.13*

BAB IV
PERANCANGAN SISTEM

4.1 Arsitektur Sistem

Untuk memahami konsep dari sistem informasi finansial yang akan dibangun, maka pada gambar 4.1 berikut ini digambarkan arsitektur sistem.



Gambar 4.1 Arsitektur Sistem

Pada gambar 4.1 Administrator bertugas melakukan pengisian Kreditur, Fasilitas dan Debitur. Sedangkan Operator bertugas menambahkan Pinjaman dan menerima Pembayaran. Masing-masing data tersebut diproses oleh *web server* dan disimpan dalama *database*.

4.2 Desain UML (*Unified Modelling Language)*

Unified Modelling Language (UML) adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. *Artifact* dapat berupa model,

deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya. Desain *Unified Modelling Language* (UML) dari sistem ini tampak seperti pada penjelasan berikut ini. UML mengizikan pengembang untuk mengembangkan berbagai tipe *visual diagram* yang mempersentasikan berbagai sudut pandang sistem.

4.2.1 *Use Case Diagram*

Use Case merupakan perilaku *software* aplikasi dimana proses tersebut menggambarkan suatu sistem, sehingga yang menggunakan sistem akan mudah mengerti mengenai kegunaan sistem yang dibangun. *Use Case Diagram* adalah gambaran (*graphical*) dari beberapa *actor*, *Use Case* , dan interaksi diantaranya yang memperkenalkan suatu sistem. *Use Case Diagram* menggambarkan siapa saja aktor yang melakukan prosedur dalam sistem serta fungsi-fungsi (proses) yang terlibat dalam transformasi pada sistem tersebut.

Adapun *use case diagram* yang diusulkan pada penelitian ini adalah sebagai berikut :

1. *Actor*

   Actor yang terdapat pada sistem yang diusulkan ini adalah :

   a. Operator, bisa melakukan pengolahan data produk, yaitu melakukan penambahan dan perubahan data produk. Operator juga bisa melakukan pengolahan data pelanggan termasuk melakukan pengolahan data order dan memeriksa konfirmasi order serta pembuatan laporan.

b. Admin, dalam sistem ini hanya bisa melakukan pengolahan data *user*. Jika ada data pengguna yang akan ditambahkan atau dirubah, maka admin berhak untuk melakukannya.

c. Pelanggan, bisa menambahkan pembelian ke keranjang belanja dan merubah pembelian. Pembeli juga bisa menambahkan dan merubah identitas alamat pengiriman serta mengisi konfirmasi pembayaran.

2. *Use Case Requirement*

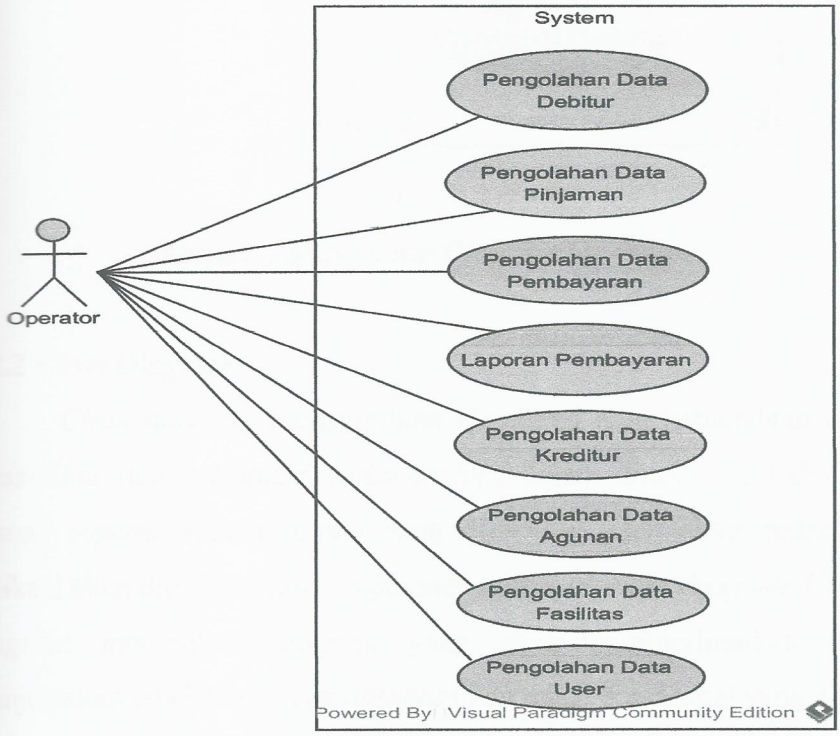
Setelah mengidentifikasi *actor*, tahap selanjutnya adalah menentukan kebutuhan fungsi *use case* yang dibutuhkan oleh sistem dalam interaksinya dengan *actor-actor* tersebut. Berikut adalah identifikasi kebutuhan *use case* pada sistem yang diusulkan ini adalah:

Tabel 4.1 ***Use Case Requirement***

No	*Requirement*	*Actor*	*Use Case*
1	Admin dapat menambahkan dan merubah dan Kreditur	Admin	Pengolahan data Kreditur
2	Admin dapat menambahkan dan merubah data Fasilitas	Admin	Pengolahan data Fasilitas
3	Operator bisa menambahkan dan merubah Debitur	Operator	Pengolahan data Debitur
4	Operator bisa menambahkan dan merubah Pembayaran	Operator	Pengolahan data Pembayaran

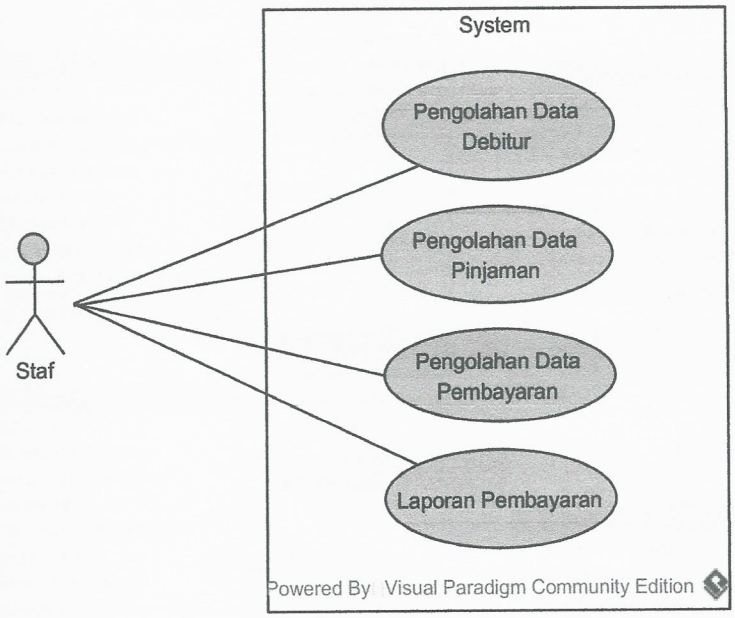
5	Operator dapat membuat laporan transaksi	Operator	Laporan transaksi
7	Admin dapat membuat dan merubah *user* (pengguna)	Admin	Pengolahan data *User*

Berikut ini merupakan *Use Case* diagram yang menggambakan proses yang dilakukan oleh Operator. Operator berhak untuk mengolah Debitur dan Data Pinjaman. Selain tersebut hanya administrator yang berhak untuk melakukan pengolahan data, baik menambahkan maupun memperbaiki data yang sudah ada.



Gambar 4.2 *Use Case Diagram* Administrator

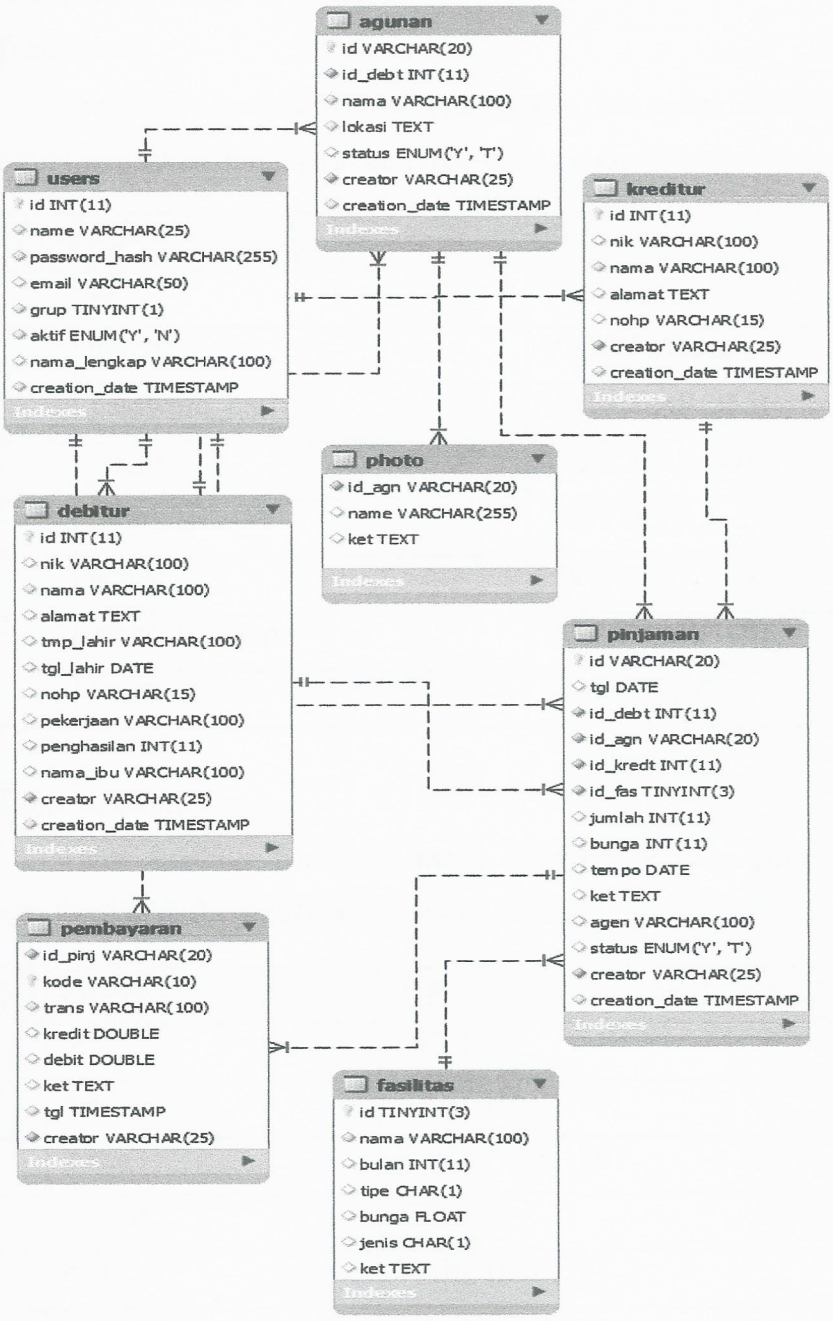
Kemudian berikut ini *use case* diagram yang menggambarkan proses yang dilakukan oleh staff atau operator



Gambar 4.3 *Use Case Diagram* Operator

4.2.2 *Class Diagram*

Class diagram menampilkan eksistensi atau keberadaan dari *class-class* dan hubungan (*relationship*) dalam desain logikal dari sebuah sistem. Semua proses yang dilakukan oleh aktor terhadap aplikasi akan didefinisikan dengan menggunakan *class diagram*. *Class diagram* merupakan diagram yang akan memperlihatkan dan menjelaskan tabel-tabel pada *database* dan relasi antar tabel yang akan digunakan didalam sistem ini.



Gambar 4.4 ***Class Diagram***

Keterangan dari *Class Diagram* :

1. Tabel *Users*

Tabel *users* berfungsi untuk menyimpan data pengguna sistem. Berikut merupakan daftar *field* dan *type* table *users*.

No	*Field*	*Type*
1	id	INT(11)
2	name	VARCHAR(25)
3	password_hash	VARCHAR(255)
4	email	VARCHAR(50)
5	grup	TINYINT(1)
6	active	ENUM('Y','N')
7	nama_lengkap	VARCHAR(100)
8	creation_date	TIMESTAMP

2. Tabel Kreditur

Tabel kreditur berfungsi untuk menyimpan data penerima kredit. Berikut daftar *field* dan *type* tabel kreditur.

No	*Field*	*Type*
1	id	INT(11)
2	nik	VARCHAR(100)
3	nama	VARCHAR(100)
4	alamat	TEXT
5	nohp	VARCHAR(15)
7	creator	VARCHAR(25)
8	creation_date	TIMESTAMP

3. Tabel Debitur

Tabel debitur berfungsi untuk menyimpan data yang memberikan dana. Berikut dafta *field* dan *type* tabel debitur.

No	*Field*	*Type*
1	id	INT(11)
2	nik	VARCHAR(100)
3	nama	VARCHAR(100)
4	alamat	TEXT
5	tmp_lahir	VARCHAR(100)
6	tgl_lahir	DATE
7	nohp	VARCHAR(15)
8	pekerjaan	VARCHAR(100)
9	penghasilan	INT(11)
10	nama_ibu	VARCHAR(100)
11	creator	VARCHAR(25)
12	creation_date	TIMESTAMP

4. Tabel Agunan

Tabel agunan berfungsi untuk menyimpan data agunan yang menjadi syarat peminjaman bagi kreditur. Berikut daftar tabel *field* dan *type* tabel agunan.

No	*Field*	*Type*
1	id	VARCHAR(10)
2	id_debt	INT(11)
3	nama	VARCHAR(100)

4	lokasi	TEXT
5	status	ENUM('Y','N')
6	creator	VARCHAR(25)
7	creation_date	TIMESTAMP

5. Tabel photo

Tabel photo berfungsi untuk menyimpan data photo agunan. Berikut daftar *field* dan *type* tabel photo.

No	*Field*	*Type*
1	id_agn	VARCHAR(20)
2	nama	VARCHAR(255)
3	ket	TEXT

6. Tabel Fasilitas

Tabel fasilitas berfungsi untuk menyimpan data fasilitas peminjaman. Setiap peminjaman dibedakan berdasarkan fasilitas yang akan diterima oleh penerima kredit. Berikut dafta *field* dan *type* table fasilitas.

No	*Field*	*Type*
1	id	TINYINT(3)
2	nama	VARCHAR(255)
3	bulan	INT(11)
4	tipe	CHAR(1)
5	bunga	FLOAT
6	jenis	CHAR(1)

7. Tabel Pinjaman

Tabel pinjaman berfungsi untuk menyimpan data pinjaman yang dilakukan. Berikut daftar *field* dan *type* tabel pinjaman.

No	*Field*	*Type*
1	id	VARCHAR(20)
2	tgl	DATE
3	id_debt	INT(11)
4	id_agn	VARCHAR(20)
5	id_kredit	INT(11)
6	id_fasilitas	TINYINT(3)
7	jumlah	INT(11)
8	bunga	INT(11)
9	tempo	DATE
10	ket	TEXT
11	agen	VARCHAR(100)
12	status	ENUM('Y','N')
13	creator	VARCHAR(25)
14	creation_date	TIMESTAMP

8. Tabel Pembayaran

Tabel pembayaran berfungsi untuk menyimpan data pembayaran cicilan.

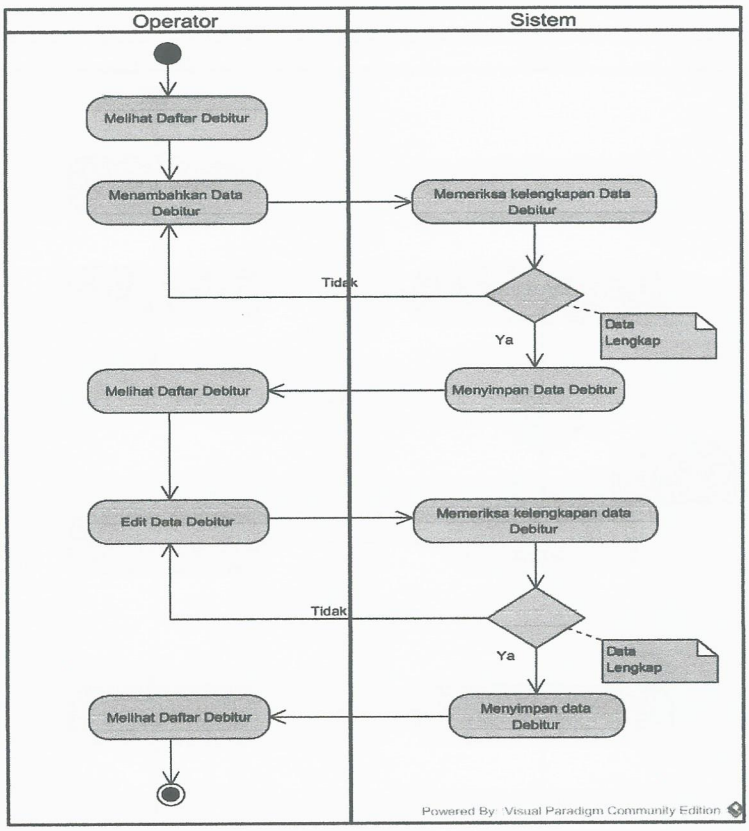
No	*Field*	*Type*
1	id_pinjaman	VARCHAR(20)
2	kode	VARCHAR(10)
3	transaksi	VARCHAR(100)
4	kredit	DOUBLE
5	debit	DOUBLE
6	ket	TEXT
7	tgl	TIMESTAMP
8	creator	VARCHAR(25)

4.2.3 *Activity Diagram*

Activity diagram merupakan diagram untuk menggambarkan logika prosedur, proses dan jalur kerja yang terjadi dalam sistem yang akan dirancang. Berikut *activity diagram* dari sistem informasi yang dirancang.

1. *Activity Diagram* pengolahan Debitur

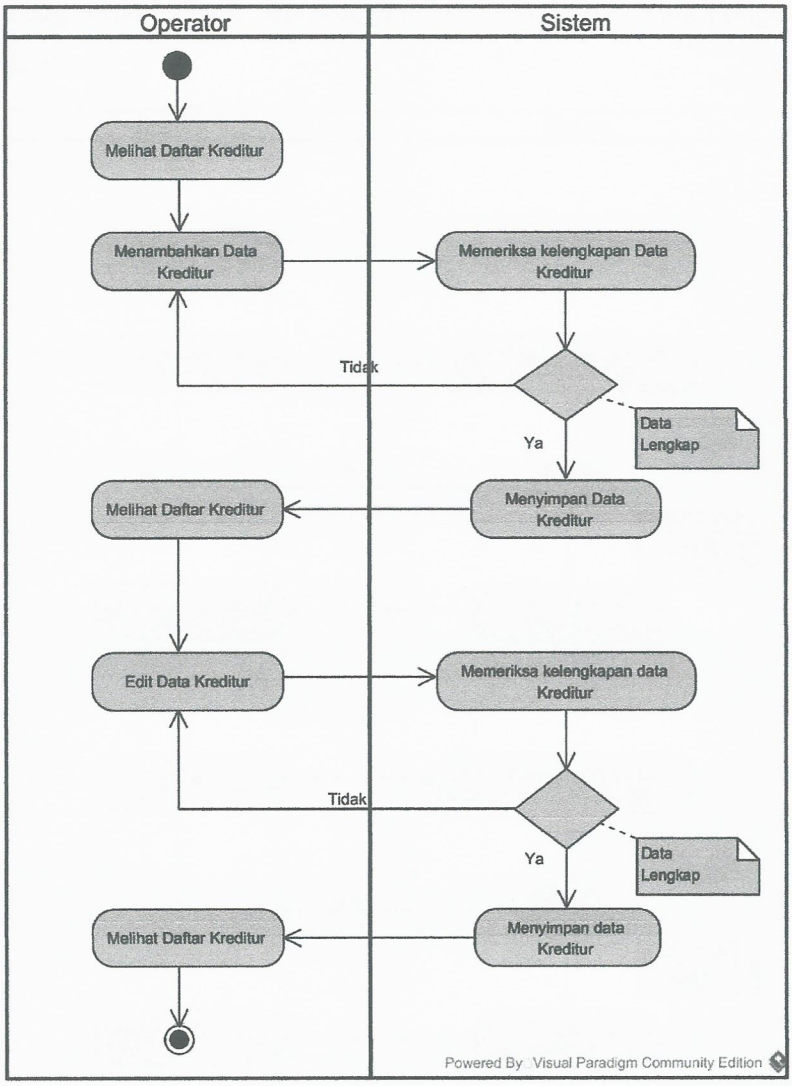
Pengolahan data Debitur dilakukan oleh administrator dan operator, dimana administrator dan operator bisa melakukan penambahan data debitur yang dilanjutkan dengan pemeriksaan kelengkapan data oleh sistem terhadap data yang ditambahkan. Berikut gambar *Activity Diagram* pengolahan data Debitur.



Gambar 4.5 *Activity Diagram* pengolahan data Debitur

2. *Activity Diagram* pengolahan Kreditur

Pengolahan data kreditur dilakukan oleh administrator dan operator, dimana administrator dan operator bisa melakukan penambahan data produk yang dilanjutkan dengan pemeriksaan kelengkapan data oleh sistem terhadap data yang ditambahkan. Berikut gambar *Activity Diagram* pengolahan data Kreditur.

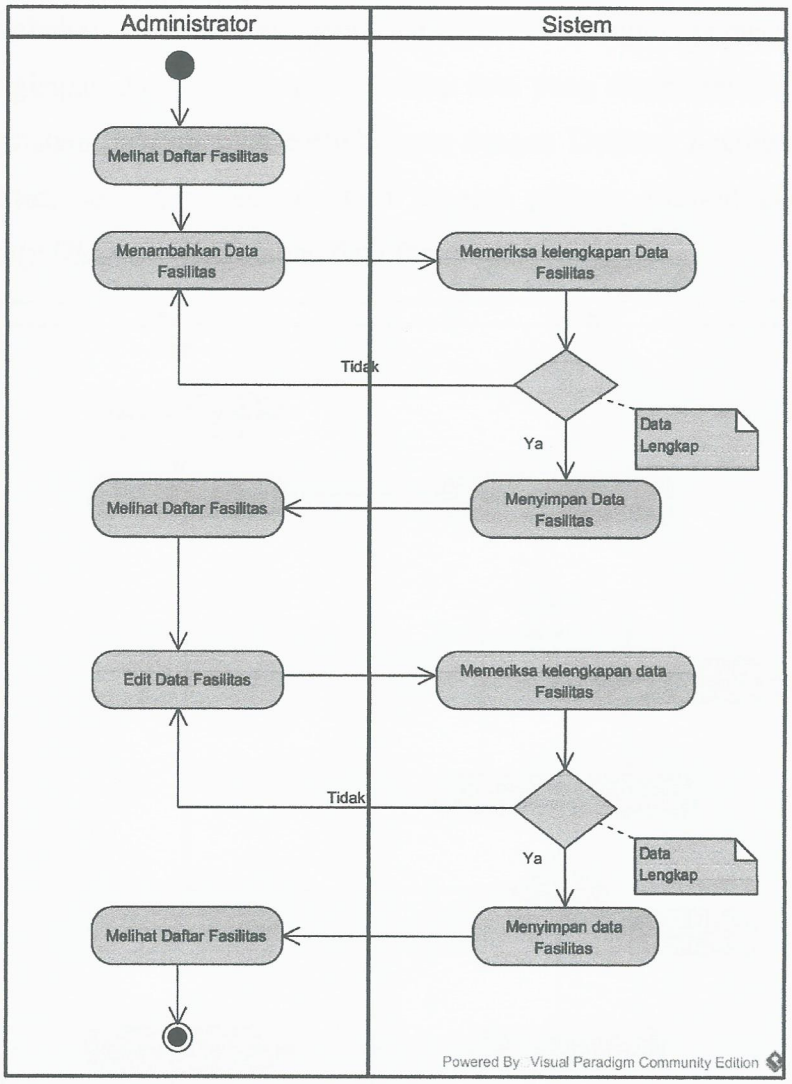


Gambar 4.6 *Activity Diagram* pengolahan data Kreditur

3. *Activity Diagram* pengolahan data Fasilitas

Pengolahan data order dilakukan oleh administrator, dimana administrator bisa melakukan penambahan data fasilitas yang

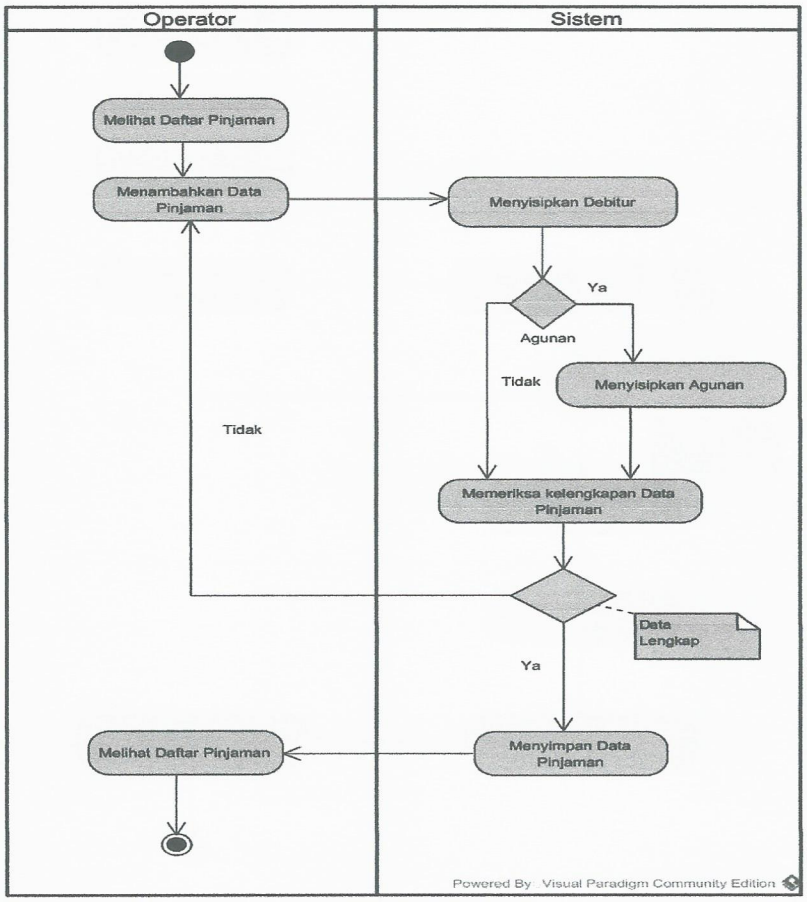
dilanjutkan dengan pemeriksaan kelengkapan data oleh sistem terhadap data yang ditambahkan. Berikut gambar *Activity Diagram* pengolahan data fasilitas.



Gambar 4.7 *Activity Diagram* pengolahan data Fasilitas

4. *Activity Diagram* pengolahan data Pinjaman

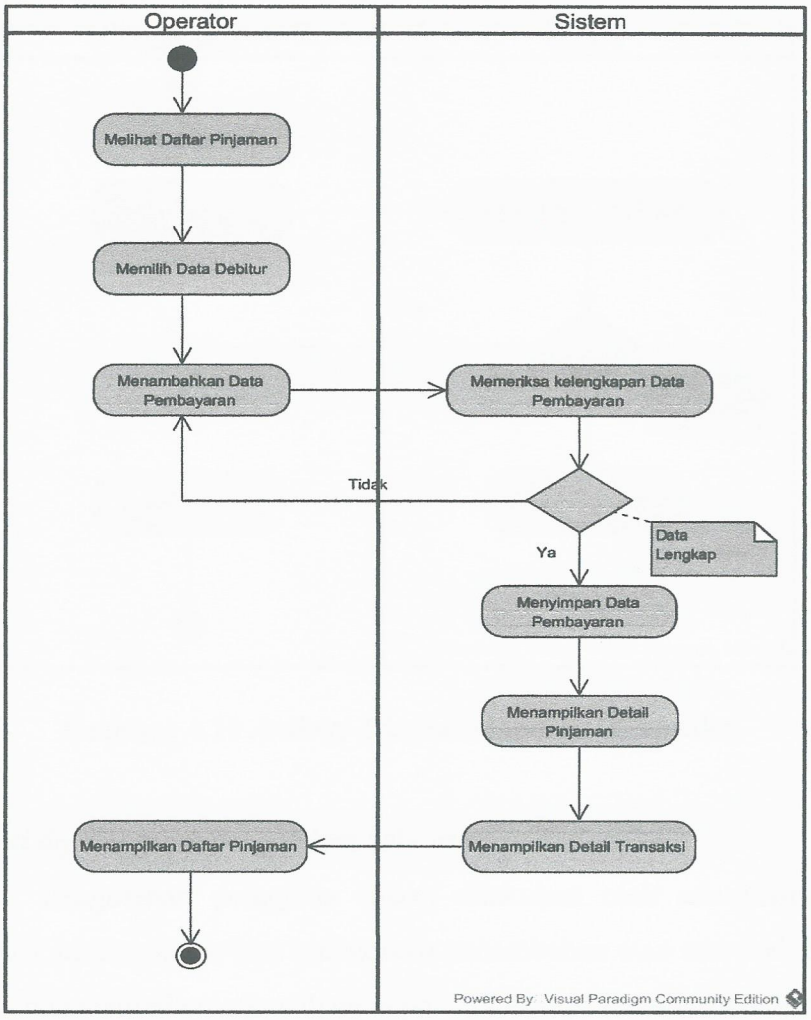
Pengolahan data Pinjaman dilakukan oleh administrator dan operator, dimana administrator dan operator bisa melakukan penambahan data dekan yang dilanjutkan dengan pemeriksaan kelengkapan data oleh sistem terhadap data yang ditambahkan. Pada pengolahan data ini juga berhubungan dengan Debitur, Kreditur dan Fasilitas, sedangkan agunan bisa sebagai pilihan. Berikut gambar *Activity Diagram* pengolahan data Pinjaman.



Gambar 4.8 *Activity Diagram* pengolahan data pinjaman

5. *Activity Diagram* pengolahan data Pembayaran

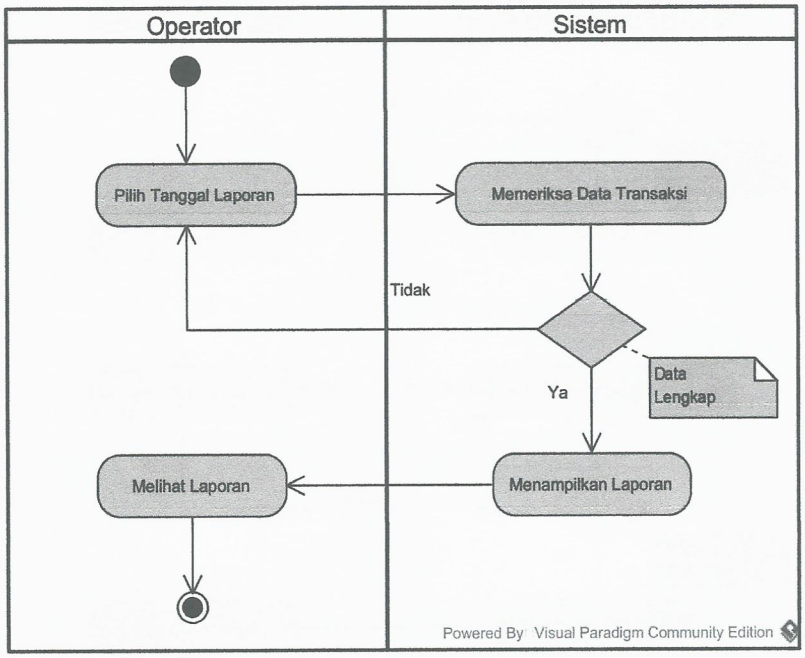
Pengolahan data pembayaran bisa dilakukan oleh administrator dan operator, dimana administrator dan operator bisa melakukan penambahan data pembayaran yang dilanjutkan dengan pemeriksaan kelengkapan data oleh sistem terhadap data yang ditambahkan. Berikut gambar *Activity Diagram* pengolahan data pembayaran.



Gambar 4.9 *Activity Diagram* pengolahan data pembayaran

5. *Activity Diagram* laporan Transaksi

Pembuatan laporan transaksi dapat dilakukan oleh administrator dan operator. Laporan transaksi diambil dari seluruh data transaksi yang dilakukan menyangkut peminjaman dan pembayaran cicilan oleh debitur yang telah diproses dalam jangka waktu tertentu. Berikut gambar *Activity Diagram* pembuatan laporan transaksi.

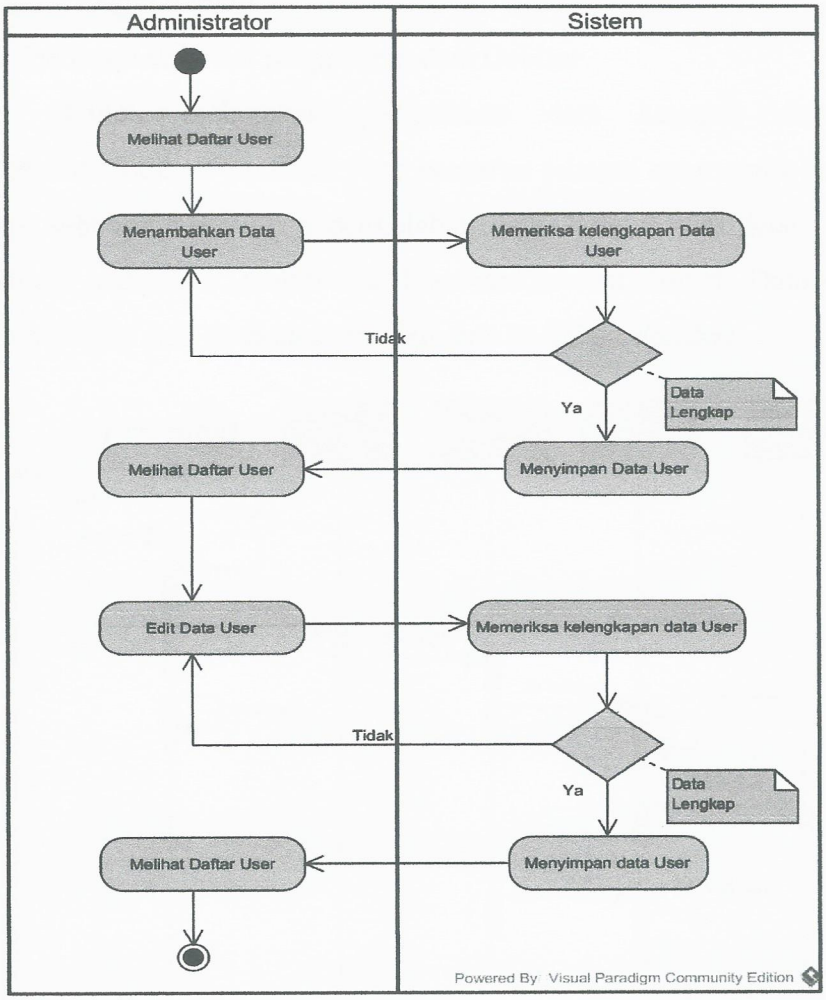


Gambar 4.10 *Activity Diagram* laporan Transaksi

6. *Activity Diagram* pengolahan data *user*

Pengolahan pengguna (*user*) dilakukan oleh administrator, dimana administrator bisa melakukan penambahan atau merubah data pengguna yang dilanjutkan dengan pemeriksaan kelengkapan data oleh

sistem terhadap data yang ditambahkan atau dirubah. Berikut gambar *Activity Diagram* pengolahan data pengguna.



Gambar 4.11 *Activity Diagram* pengolahan data user

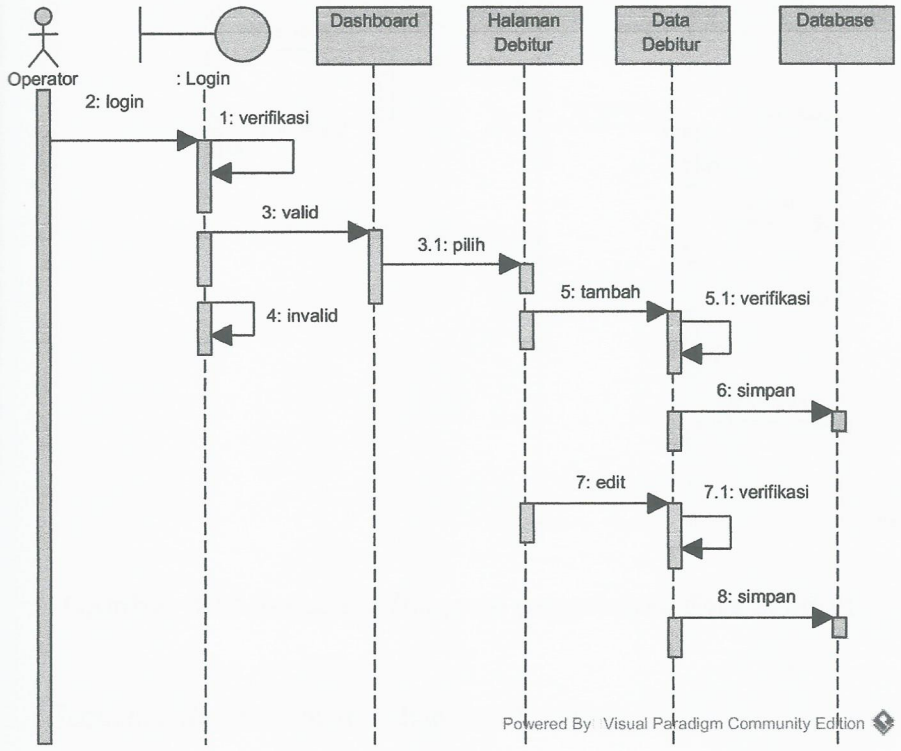
4.2.4 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek didalam dan disekitar sistem (termasuk pengguna, *display*, dan

sebagainya) berupa *message* yang digambarkan terhadap waktu. Adapun *sequence diagram* pada sistem informasi yang dirancang adalah sebagai berikut :

1. *Sequence diagram* pengolahan data Debitur

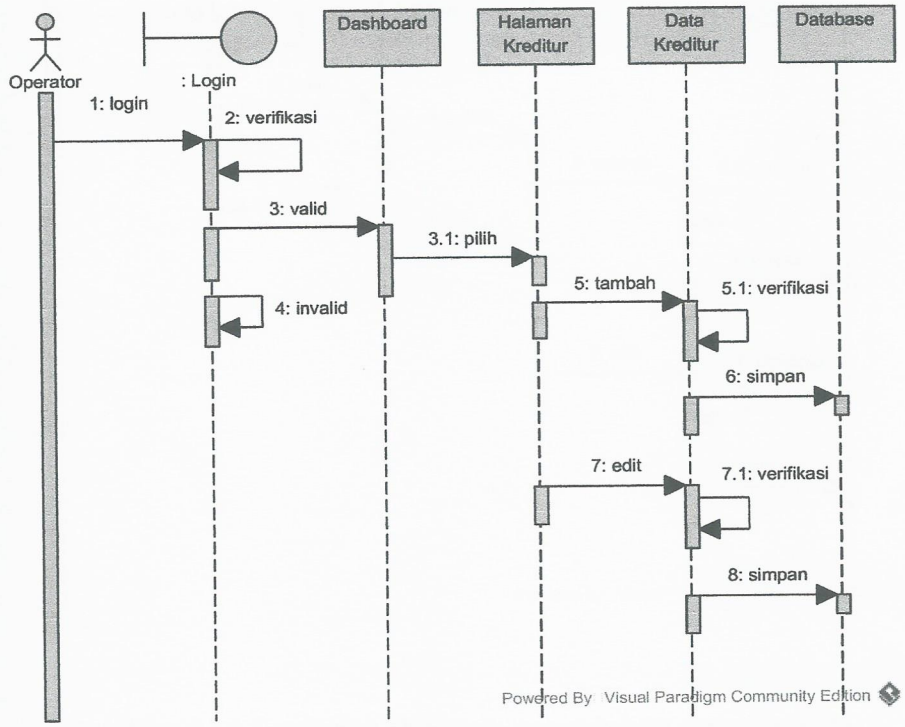
Untuk melakukan pengolahan data kategori debitur, Administartor atau Operator yang berperan sebagai *actor* wajib untuk *login* sebelum melakukan pengolahan data. Setelah verifikasi *login* berhasil, maka data Debitur bisa ditambahkan atau dirubah. Data yang ditambahkan atau dirubah akan disimpan ke dalam *database*.



Gambar 4.12 *Sequence Diagram* pengolahan data Debitur

2. *Sequence diagram* pengolahan data Kreditur

Untuk melakukan pengolahan data kreditur, Administrator dan Operator yang berperan sebagai *actor* wajib untuk *login* sebelum melakukan pengolahan data. Setelah verifikasi *login* berhasil, maka data kreditur bisa ditambahkan atau dirubah. Data yang ditambahkan atau dirubahkan akan disimpan ke dalam *database*.

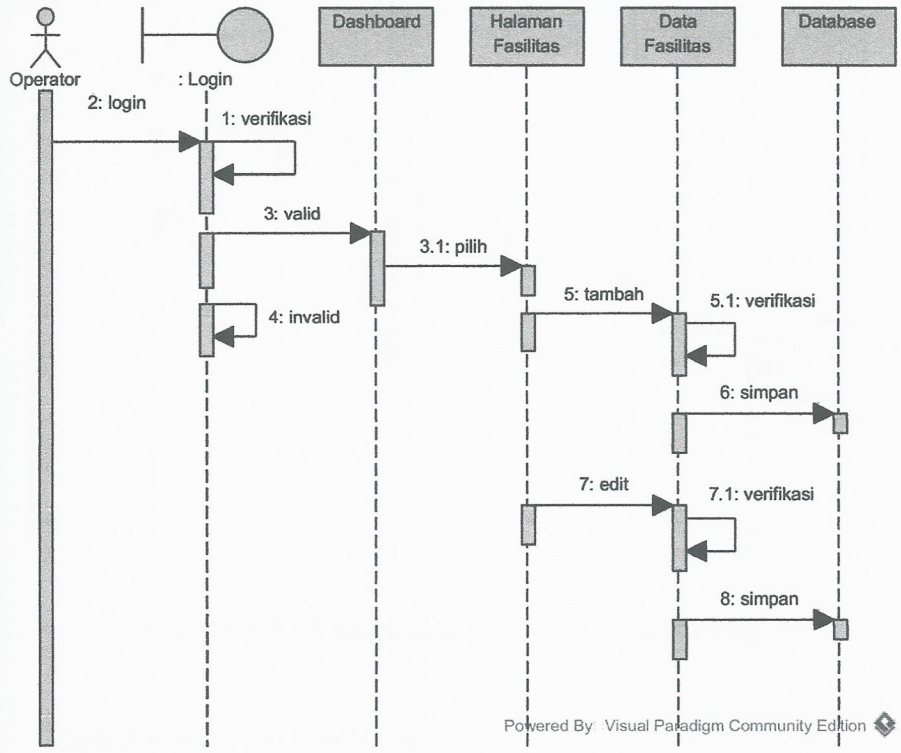


Gambar 4.13 *Sequence Diagram* pengolahan data produk

3. *Sequence diagram* pengolahan data Fasilitas

Untuk melakukan pengolahan data fasilitas, Administrator yang berperan sebagai *actor* wajib untuk login sebelum melakukan pengolahan data. Setelah verifikasi *login* berhasil, maka data fasilitas

bisa diolah. Data yang telah diolah akan disimpan ke dalam *database*. Sequence Diagra pengolahan data fasilitas bisa dilihat pada gambar berikut ini.

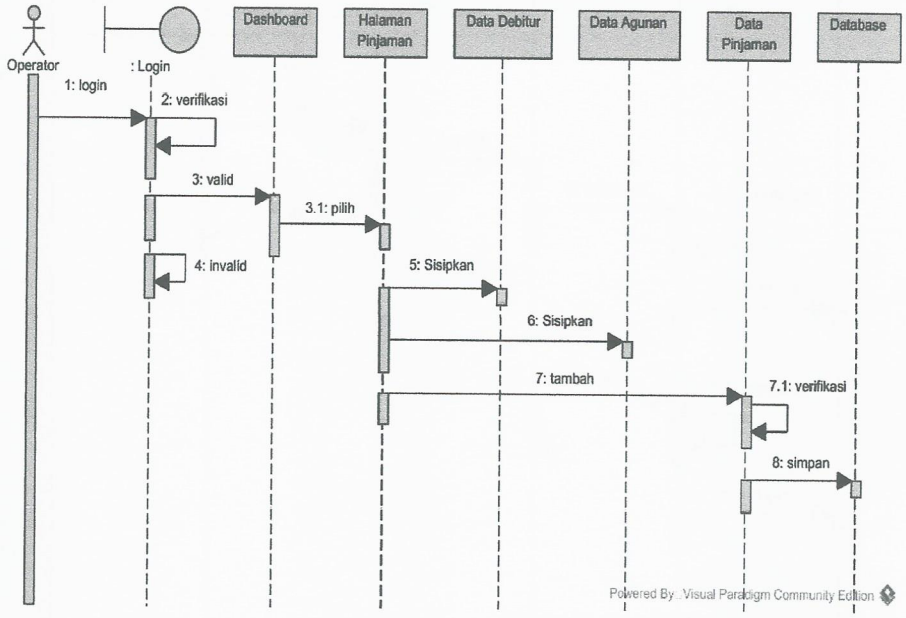


Gambar 4.14 *Sequence Diagram* pengolahan data Fasilitas

4. *Sequence diagram* Pinjaman

Untuk melakukan pengolahan data pinjaman, Administrator atau Operator yang berperan sebagai *actor* wajib untuk login sebelum melakukan pengolahan data. Setelah verifikasi *login* berhasil, maka data pinjaman bisa diolah. Data yang telah diolah akan disimpan ke

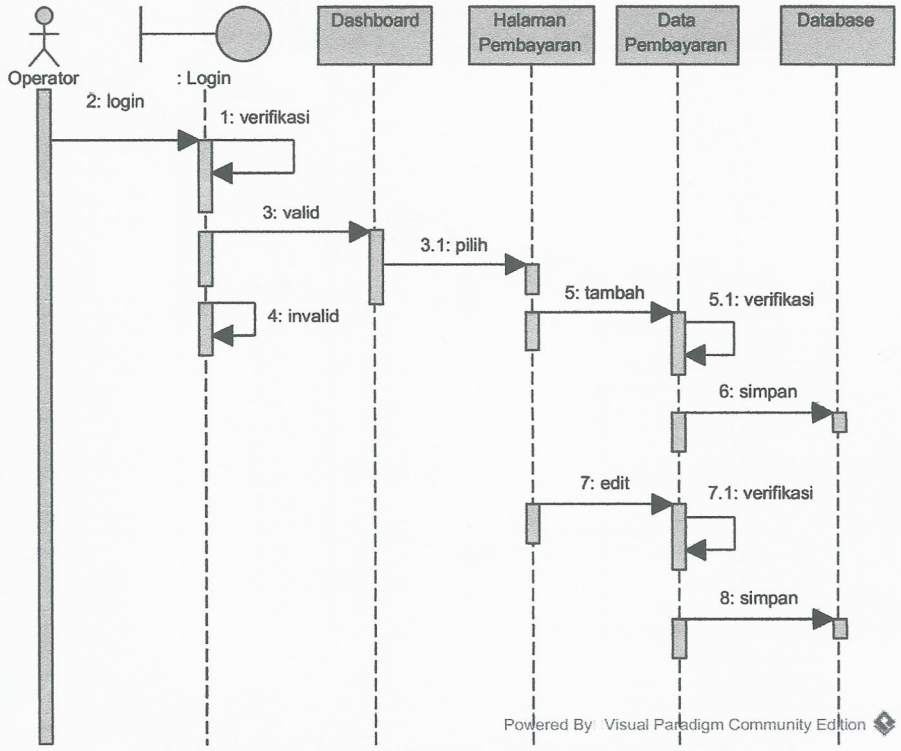
dalam *database*. *Sequence Diagram* pengolahan data pinjaman bisa dilihat pada gambar berikut ini



Gambar 4.15 *Sequence Diagram* data Pinjam

5. *Sequence diagram* Pembayaran

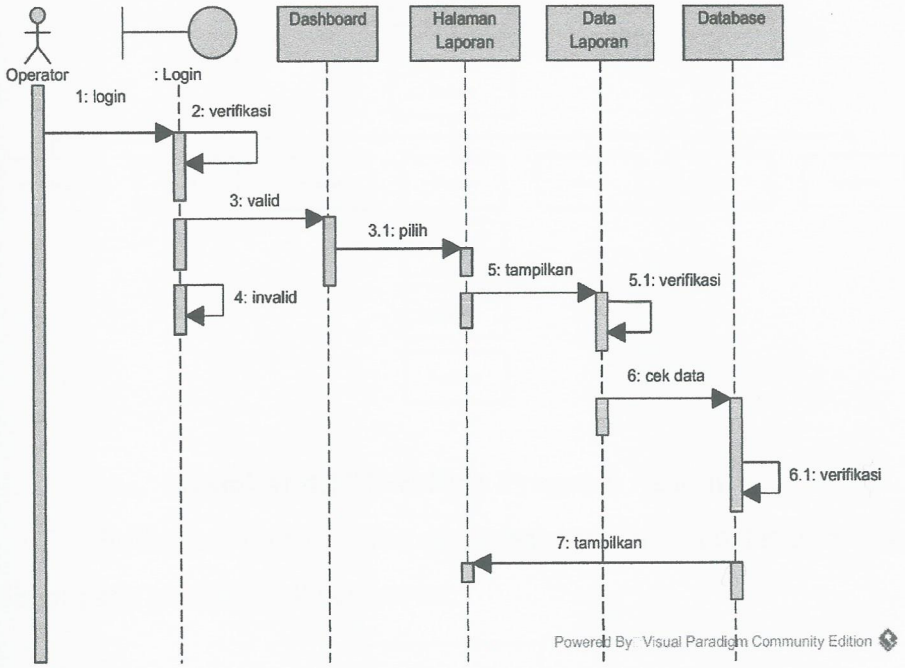
Untuk melakukan pengolahan data pembayaran, Administrator atau Operator yang berperan sebagai *actor* wajib untuk login sebelum melakukan pengolahan data. Setelah verifikasi *login* berhasil, maka data pembayaran bisa diolah. Data yang telah diolah akan disimpan ke dalam *database*. *Sequence Diagram* pengolahan data pembayaran bisa dilihat pada gambar berikut ini



Gambar 4.16 *Sequence Diagram* data Pinjam

6. *Sequence diagram* Laporan Transaksi

Untuk melakukan melihat laporan transaksi, Administrator atau Operator yang berperan sebagai *actor* wajib untuk login sebelum melakukan pengolahan data. Setelah verifikasi *login* berhasil, maka data laporan transaksi bisa ditampilkan sesuai dengan tanggal yang ditentukan. *Sequence Diagram* menampilkan data laporan bisa dilihat pada gambar berikut ini



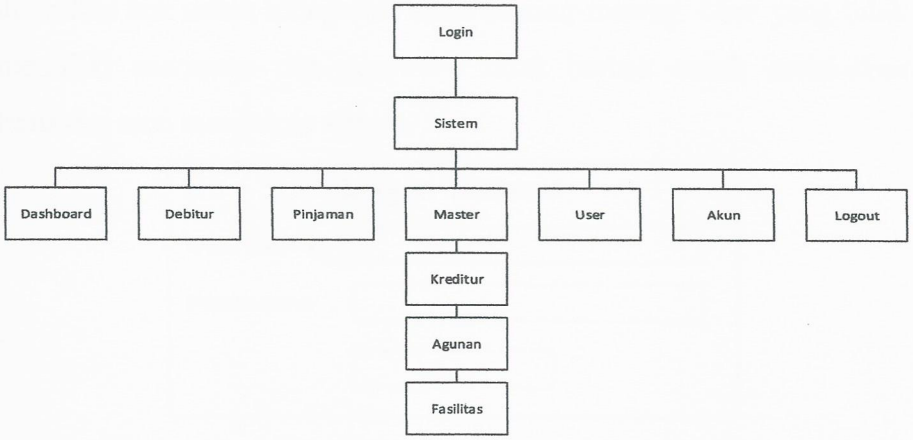
Gambar 4.17 *Sequence Diagram* Laporan

4.2.5 Perancangan Antar Muka

Perancangan antar muka bertujuan untuk memberikan *interface* tentang desain program yang akan dibuat. Di bawah ini terdapat desain *template* pada tampilan sistem yang dibuat oleh penulis.

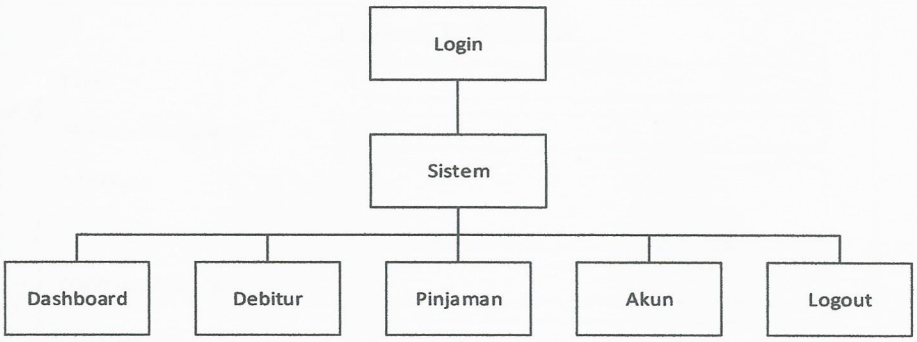
4.2.5.1 Struktur Program

Dalam perancangan sebuah sistem dibutuhkan struktur menu yang berisikan menu dan submenu yang berfungsi untuk memudahkan *user* dalam menggunakan sistem tersebut. Berikut ini digambarkan mengenai stuktur menu program bagian pengelola dalam sistem ini.



Gambar 4.18 Struktur Program Admin

Sedangkan untuk stuktur menu program bagian pelanggan bisa diihat pada gambar 4.18 berikut ini.



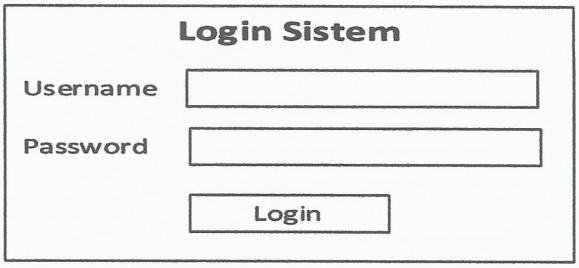
Gambar 4.19 Struktur Program Operator

4.2.5.2 Perancangan *Input*

1. *Form Login*

Form Login merupakan halaman untuk autentikasi *user*. Setiap *user* yang akan menggunakan melakukan transaksi ke dalam sistem wajib untuk memasukkan *username* dan *password* yang sah agar

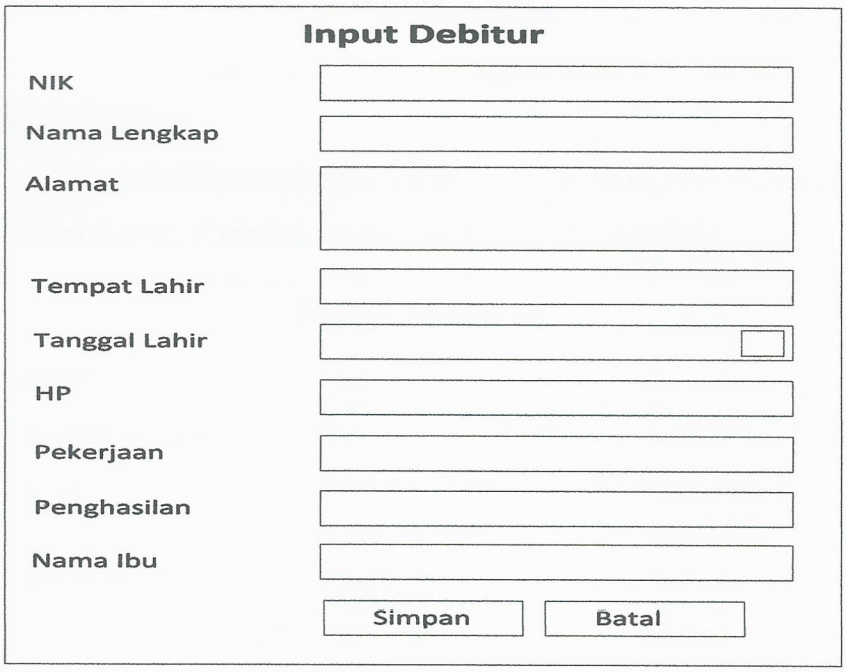
diberikan hak untuk mengelola akun masing-masing. *User* yang tidak memiliki *username* dan *password* tidak berhak untuk melakukan transaksi atau mengelola sistem.



Gambar 4.20 *Form Login*

2. *Form Input* Debitur

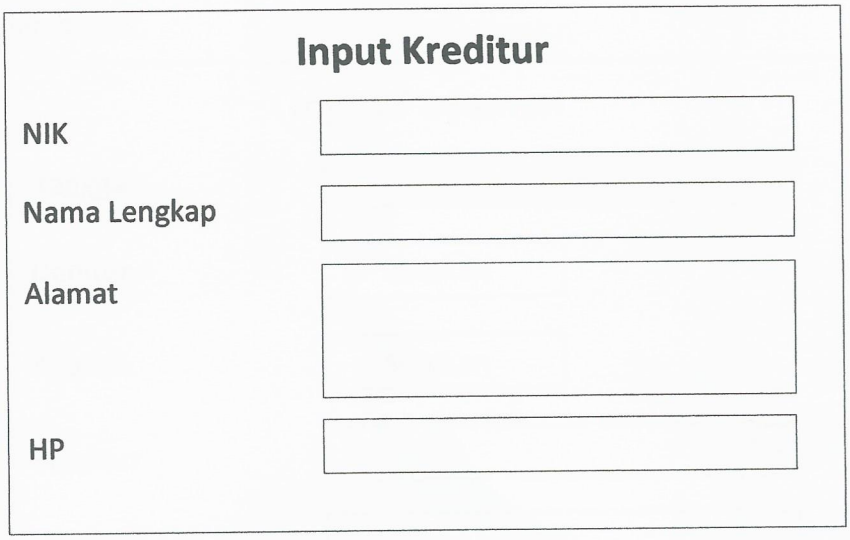
Form input debitur berfungsi untuk menambahkan data debitur yang diperlukan saat melakukan proses peminjaman.



Gambar 4.21 *Form Input* Konfirmasi

3. *Form* Kreditur

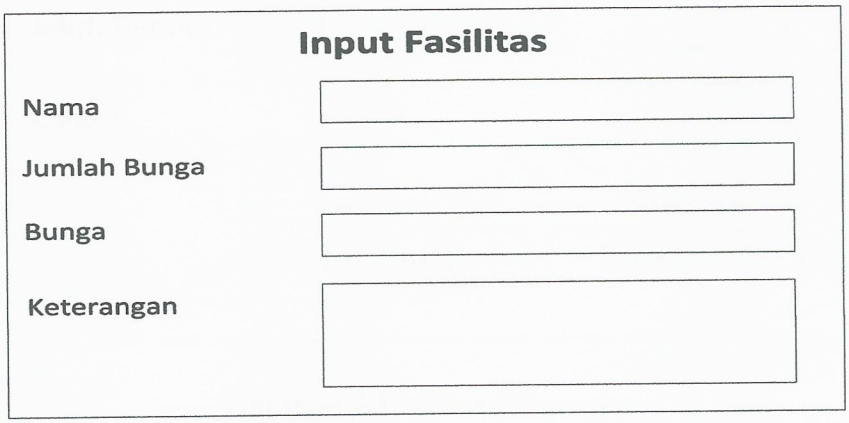
Form input kreditur berfungsi untuk menambahkan data kreditur yang diperlukan saat melakukan proses peminjaman.



Gambar 4.22 *Form Input* Kreditur

4. *Form* Fasilitas

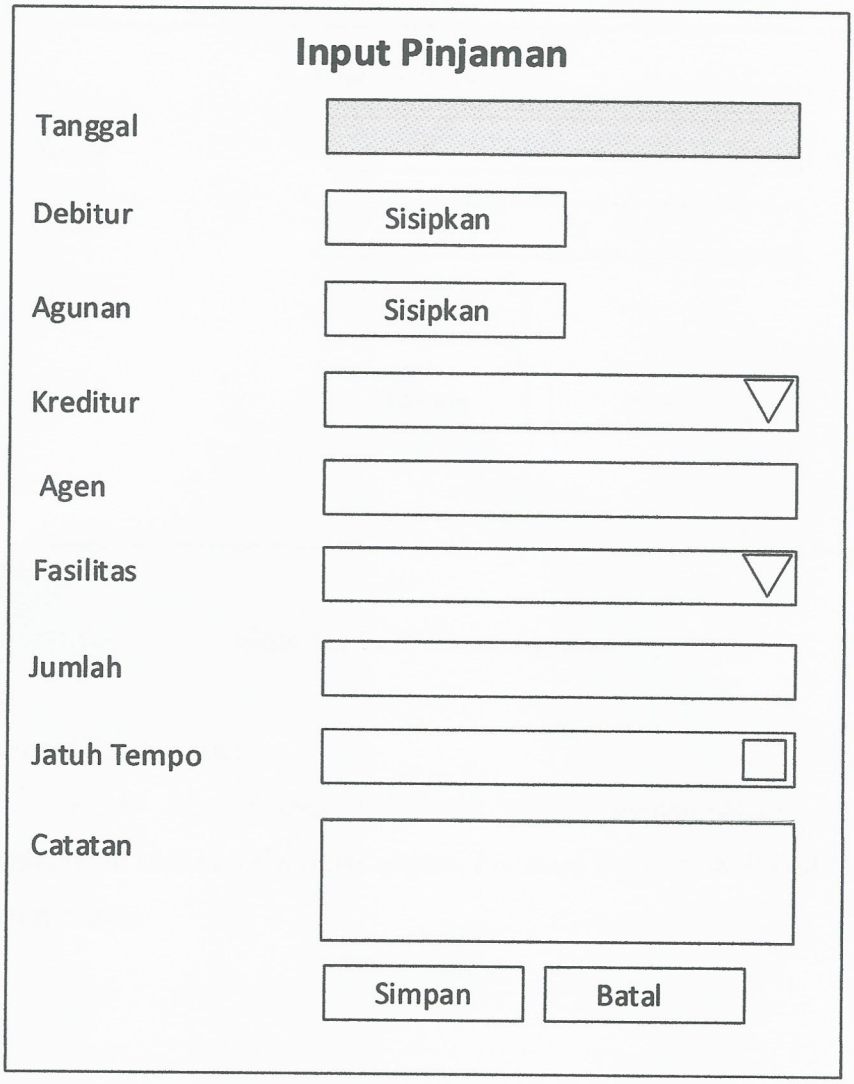
Form fasilitas berfungsi untuk menambahkan fasilitas kedalam *database* sistem, *Form* ini hanya bisa dilihat oleh administrator.



Gambar 4.23 *Form Input* Fasilitas

5. *Form* Pinjaman

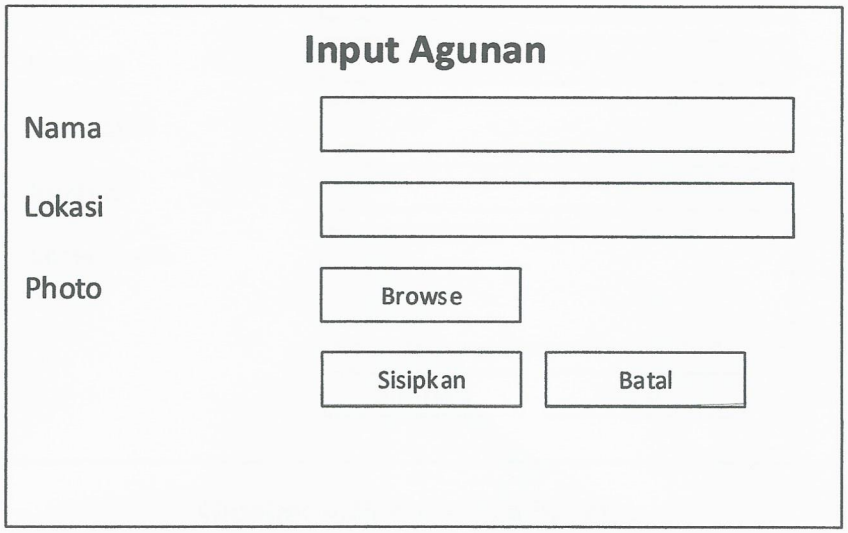
Form pinjaman berfungsi untuk menambahkan data pinjaman kedalam *database* sistem, *Form* ini hanya bisa dilihat oleh administrator.



Gambar 4.24 *Form* Pinjaman

6. *Form* Agunan

Form Agunan berfungsi untuk menambahkan data agunan kedalam *database* sistem, *Form* ini hanya bisa dilihat oleh administrator.



Gambar 4.25 *Form* Agunan

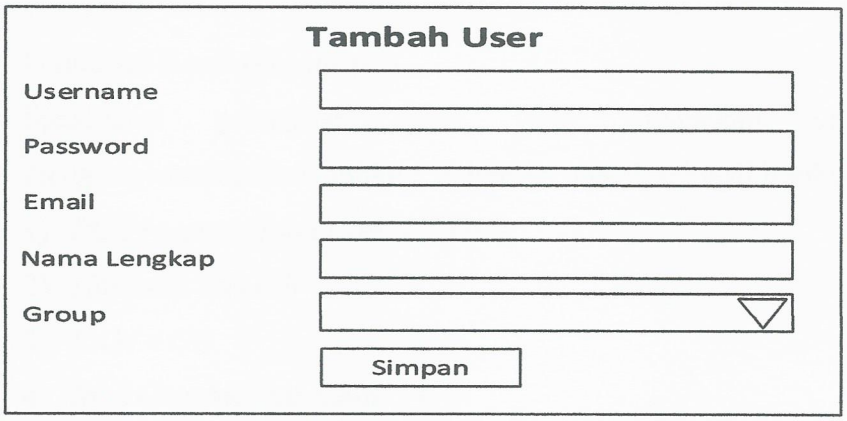
6. *Form* Pembayaran

Form Pembayaran berfungsi untuk menambahkan data pembayaran kedalam *database* sistem, *Form* ini hanya bisa dilihat oleh administrator.

Gambar 4.26 *Form* Pembayaran

6. *Form* User

Form User berfungsi untuk menambahkan data user kedalam *database* sistem, *Form* ini hanya bisa dilihat oleh administrator.



Gambar 4.27 *Form* User

BAB V
IMPLEMENTASI DAN HASIL

5.1 Implementasi Sistem

Setelah dilakukan perancangan sistem, maka tahap selanjutnya adalah pengimplementasian dan pengujian sistem. Tujuan dari implementasi yaitu untuk menerapkan sistem agar dapat dioperasikan secara optimal sesuai dengan ketentuan proses. Pengujian sistem merupakan tahap uji coba terhadap sistem yang telah dibuat apakah sistem sudah berjalan dengan benar serta uji coba langsung bagi pengguna bagaimana cara menjalankannya.

5.1.1 Komponen Utama dalam Implementasi Sistem

Komponen utama dalam implementasi sistem ini terbagi atas beberapa komponen yaitu perangkat keras (*Hardware*), perangkat lunak (*Software*) dan sumber daya manusia (*Brainware*).

1. Perangkat Keras (*Hardware*)

   Spesifikasi perangkat keras yang digunakan untuk mengimplementasikan aplikasi sistem ini adalah sebagai berikut:

   1) *PC Processor Intel Core 1,8 GHz*
   2) *Harddisk 500 GB*
   3) *RAM 4 GB*
   4) *Intel Graphic Onboardb 1 GB*

2. Perangkat Lunak (*Software*)

   Spesifikasi perangkat keras (*hardware*) yang digunakan untuk mengimplementasikan aplikasi sistem ini adalah sebagai berikut:

   1) *Windows* 10 *Professional*
   2) *Mozilla Firefox 38.0*
   3) *XAMPP 7.0.13*

3. Sumber Daya Manusia (*Brainware*)

   Pada implementasi ini dibutuhkan seorang *user* yang mampu menjalankan atau mengoperasikan komputer dan menjalankan sistem yang telah dibuat sehingga penerapan dan implementasi sistem dapat berjalan dengan lancar.

5.1.2 Kebutuhan *Server* dalam Implementasi Sistem

Agar sistem yang dirancang bisa berjalan dengan sempurna, maka dibutuh *server* yang mengakomodasi kebutuhan bahasa pemrograman. *Server* yang dimaksud sudah tersedia secara sempurna dalam satu buah *software* yaitu XAMPP 7.0.13. Dalam *software* tersebut terdapat:

1) *Apache2*
2) PHP7
3) MySQL5
4) *phpMyAdmin*

5.2 Pengujian Sistem

Pengujian sistem dilakukan pada modul-modul yang telah dirancang pada sistem informasi finansial. Adapun pengujian yang dilakukan adalah sebagai berikut:

1. Pengujian *Login* User

Pengujian yang dilakukan pada *form login* yaitu dengan melakukan *input* data sesuai *form* yang disediakan.

Untuk lebih jelasnya bisa dilihat pada tabel 5.1 berikut ini.

Tabel 5.1 Pengujian *Login User*

Data Masukan	**Proses Diharapkan**	**Pengamatan**	**Kesimpulan**
Data Normal			
Memasukkan data *username* dan *password* secara lengkap	Dapat masuk kedalam sistem melewati *form login*	Proses berhasil sesuai yang diharapkan	Berjalan
Data Kurang			
Tidak memasukkan salah satu data	Sistem menolak proses dan menampilkan peringatan kesalahan input	Proses berhasil sesuai yang diharapkan	Berjalan
Data Salah			

Memasukkan data *login* salah	Sistem menolak proses dan menampilkan peringatan kesalahan input	Proses berhasil sesuai yang diharapkan	Berjalan

2. Pengujian Data Kreditur

Pengujian yang dilakukan pada data kreditur yaitu dengan melakukan melakukan penambahan dan perbaikan data Kreditur pada form yang disediakan. Untuk lebih jelasnya bisa dilihat pada tabel 5.2 berikut ini.

Tabel 5.2 Pengujian Data Kreditur

Data Masukan	**Proses Diharapkan**	**Pengamatan**	**Kesimpulan**
Data Normal			
Memasukkan data secara lengkap	Sistem menyimpan data dan menampikan informasi data berhasil disimpan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Kurang			
Tidak memasukkan data wajib	Sistem menolak proses dan menampikan	Proses berhasil sesuai	Berjalan

	peringatan kesalahan	yang diharapkan	
Data Salah			
Tidak memasukkan data dan menekan tombol simpan	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan

3. Pengujian Data Debitur

Pengujian yang dilakukan padadengan melakukan penambahan dan perbaikan Debitur data sesuai *form* yang disediakan. Untuk lebih jelasnya bisa dilihat pada tabel 5.3 berikut ini.

Tabel 5.3 Pengujian Data Debitur

Data Masukan	Proses Diharapkan	Pengamatan	Kesimpulan
Data Normal			
Memasukkan data secara lengkap	Sistem menyimpan data dan menampikan informasi data berhasil disimpan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Kurang			

Tidak memasukkan data wajib	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Salah			
Tidak memasukkan data dan menekan tombol simpan	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan

4. Pengujian Data Fasilitas

Pengujian yang dilakukan pada data fasilitas yaitu dengan melakukan penambahan dan perbaikan data sesuai *form* yang disediakan.. Untuk lebih jelasnya bisa dilihat pada tabel 5.4 berikut ini.

Tabel 5.4 Pengujian Data Fasilitas

Data Masukan	Proses Diharapkan	Pengamatan	Kesimpulan
Data Normal			
Memasukkan data secara lengkap	Sistem menyimpan data dan menampikan informasi data	Proses berhasil sesuai yang diharapkan	Berjalan

	berhasil disimpan		
Data Kurang			
Tidak memasukkan data wajib	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Salah			
Tidak memasukkan data dan menekan tombol simpan	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan

5. Pengujian Data Pinjaman

Pengujian yang dilakukan pada data pinjaman yaitu dengan melakukan *input* data sesuai *form* yang disediakan. Data pinjaman berkaitan dengan Debitur, Agunan, Kreditur dan Fasilitas. Untuk lebih jelasnya bisa dilihat pada tabel 5.5 berikut ini.

Tabel 5.5 Pengujian Data Pinjaman

Data Masukan	**Proses Diharapkan**	**Pengamatan**	**Kesimpulan**
Data Normal			

Memasukkan data secara lengkap	Sistem menyimpan data dan menampikan informasi data berhasil disimpan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Kurang			
Tidak memasukkan data wajib	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Salah			
Tidak memasukkan data dan menekan tombol simpan	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan

6. Pengujian Data Pembayaran

Pengujian yang dilakukan pada data pembayaran yaitu dengan melakukan *input* data sesuai *form* yang disediakan. Untuk lebih jelasnya bisa dilihat pada tabel 5.6 berikut ini.

Tabel 5.6 Pengujian Data Pembayaran

Data Masukan	Proses Diharapkan	Pengamatan	Kesimpulan
Data Normal			
Memasukkan data secara lengkap	Sistem menyimpan data dan menampikan informasi data berhasil disimpan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Kurang			
Tidak memasukkan data wajib	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Salah			
Tidak memasukkan data dan menekan tombol simpan	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan

7. Pengujian Data Laporan

Pengujian yang dilakukan pada data laporan yaitu dengan melakukan proses pemilihan tanggal laporan yang diinginkan dan

menekan tombol tampilkan laporan. Untuk lebih jelasnya bisa dilihat pada tabel 5.7 berikut ini.

Tabel 5.7 Pengujian Data Laporan

Data Masukan	Proses Diharapkan	Pengamatan	Kesimpulan
Data Normal			
Memasukkan data secara lengkap	Sistem menyimpan data dan menampikan informasi data berhasil disimpan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Kurang			
Tidak memasukkan data wajib	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Salah			
Tidak memasukkan data dan menekan tombol simpan	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan

10. Pengujian Data *User*

Pengujian yang dilakukan pada data *user* yaitu dengan melakukan *input* data sesuai *form* yang disediakan. Untuk lebih jelasnya bisa dilihat pada tabel 5.10 berikut ini.

Tabel 5.8 Pengujian Data *User*

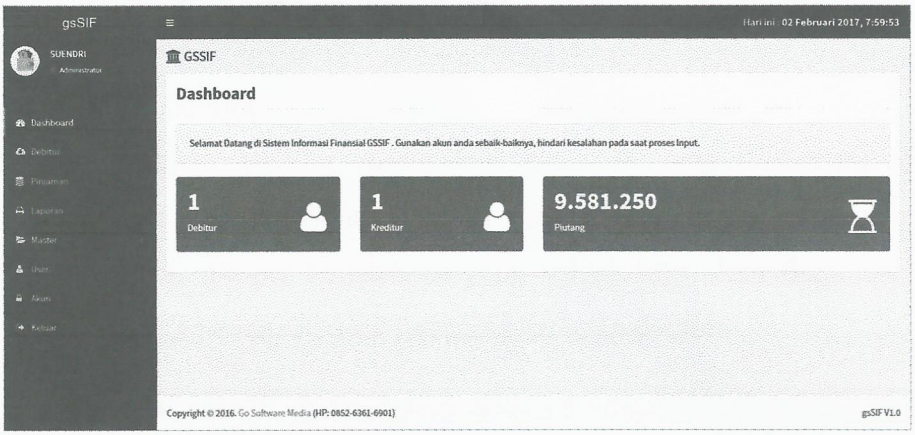
Data Masukan	**Proses Diharapkan**	**Pengamatan**	**Kesimpulan**
Data Normal			
Memasukkan data secara lengkap	Sistem menyimpan data dan menampikan informasi data berhasil disimpan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Kurang			
Tidak memasukkan data wajib	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan
Data Salah			
Tidak memasukkan data dan menekan tombol simpan	Sistem menolak proses dan menampikan peringatan kesalahan	Proses berhasil sesuai yang diharapkan	Berjalan

5.3 Hasil Pengujian

Hasil pengujian merupakan tampilan hasil akhir dari sistem informasi finansial yang dirancang.. Adapun hasil pengujian adalah sebagai berikut:

1. Halaman Utama

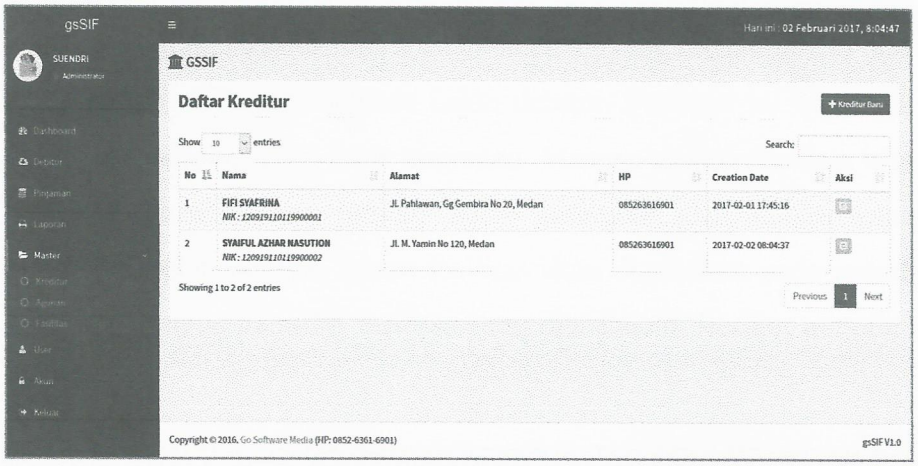
Halaman utama sistem merupakan halaman awal yang dijumpai dimana ketika user berhasil *login* kedalam sistem. Pada halaman utama atau biasa disebut dengan *Dashboard,* ditampilkan jumlah Debitur yang tersimpan dalam *database*, jumlah Kreditur dan total piutang.



Gambar 5.1 Halaman Utama

2. Halaman Kreditur

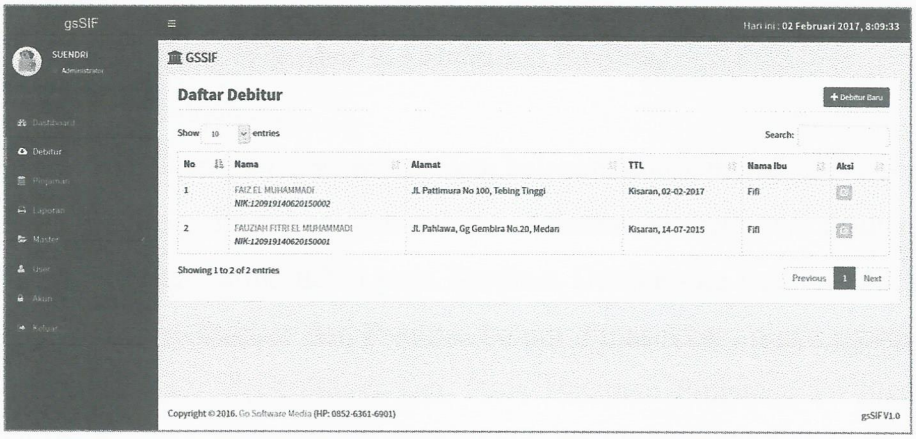
Halaman Kreditur merupakan halaman yang menampilkan data seluruh kreditur yang memberikan kredit kepada debitur. Pada sistem yang dirancang, kreditur tidak hanya seorang atau satu perusahaan saja, namun orang lain juga bisa berlaku sebagai kreditur. Berikut tampilan halaman kreditur



Gambar 5.2 Halaman Kreditur

3. Halaman Debitur

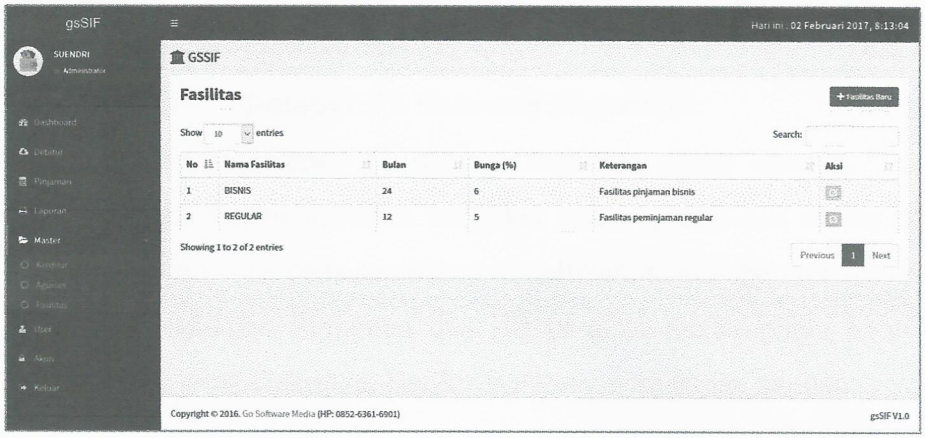
Halaman Debitur merupakan halaman yang menampilkan seluruh data debitur. Seluruh calon debitur yang akan melakukan peminajaman, diinput kedatabase yang selanjutnya akan digunakan pada menu Pinjaman. Berikut tampilan halaman debitur



Gambar 5.3 Halaman Debitur

4. Halaman Fasilitas

Halaman Fasilitas merupakan halaman yang menampilkan data fasilitas yang diberikan kepada calon debitur. Fasilitas ini berfungsi untuk mengelompokkan jenis pinjaman. Masing-masing fasilitas bisa dibedakan berdasarkan jumlah bulan pinjaman dan bunga yang ditanggung oleh debitur. Menu fasilitas akan digunakan pada proses pinjaman dengan memilih fasilitas yang disediakan. Berikut tampilan keranjang.

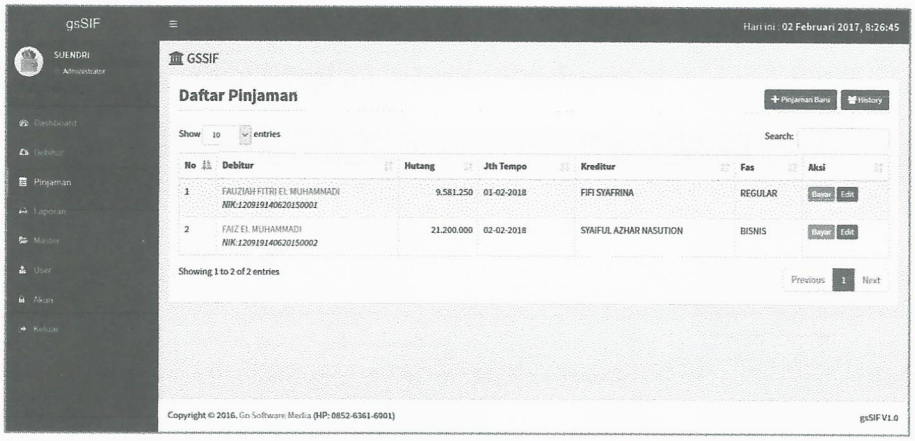


Gambar 5.4 Halaman Fasilitas

5. Halaman Pinjaman

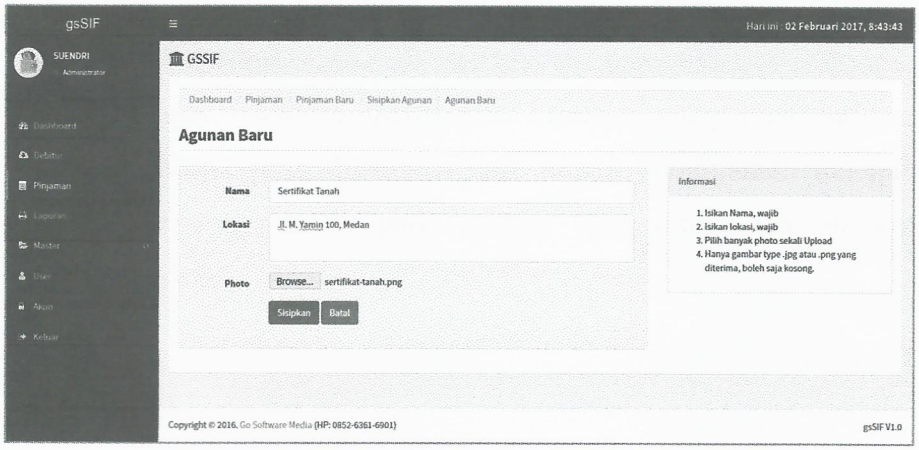
Halaman Pinjaman merupakan halaman untuk memproses data pinjaman yang dilakukan oleh debitur. Halaman ini berkaitan dengan data yang telah dimasukkan pada Kreditur, Debitur dan Fasilitas. Jika data Kreditur, Debitur dan Fasilitas belum dimasukkan, maka proses pinjaman tidak dapat dilanjutkan. Pada Halaman Pinjaman ini juga terdapat menu penyisipan Agunan yang diberikan pada saat proses peminjaman. Namun ini bersifat pilihan, Operator boleh mengabaikan

menu tersebut jika memang tidak diwajibkan menggunakan agunanan. Berikut halaman pinjaman.



Gambar 5.5 Halaman Pinjaman

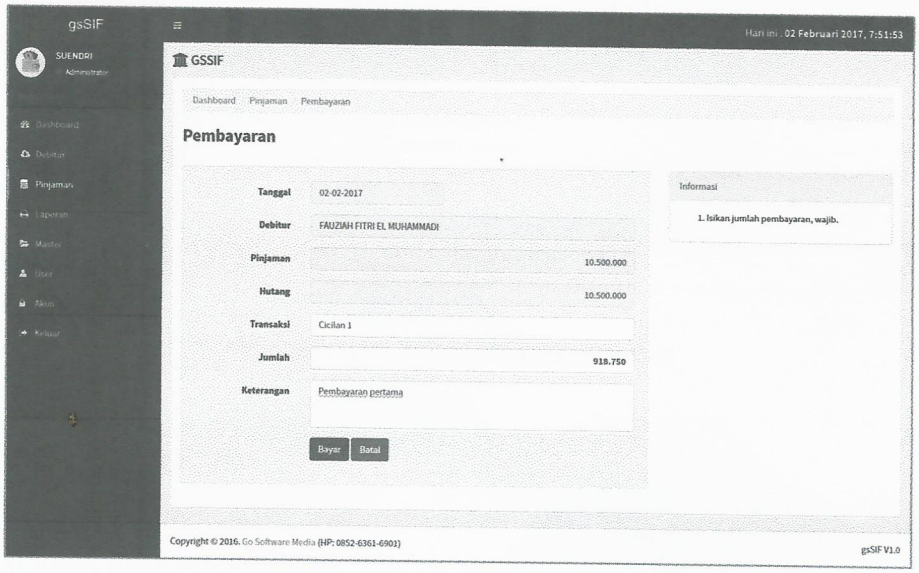
Proses penyisipan agunan melampirkan photo agunan, jika dalam bentuk sertifikat, maka sertifikat bisa diphoto dan disimpan kedalam sistem. Jika pada suatu saat diperlukan, maka operator cukup membukan halaman Pinjaman dan sistem akan menampikan photo agunan yang telah disimpan. Berikut halaman penyisipan agunan.



Gambar 5.6 Halaman penyisipan Agunan

7. Halaman Pembayaran

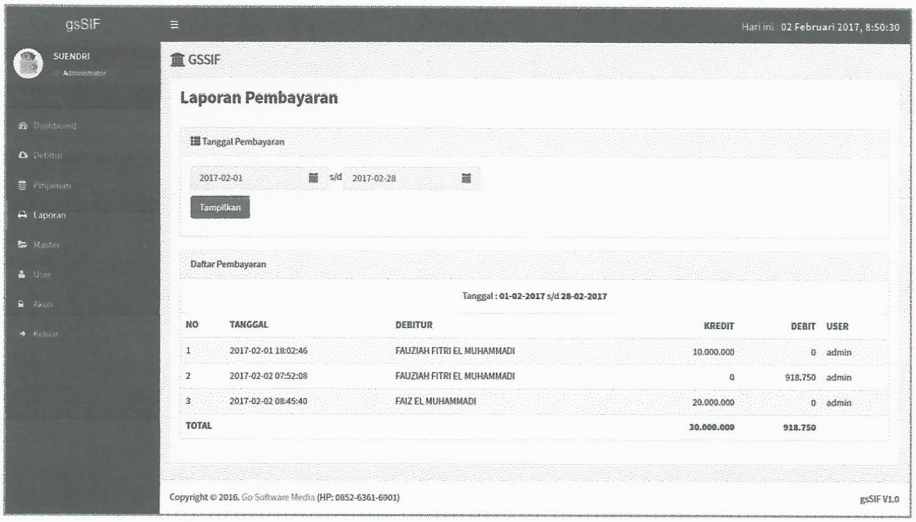
Halaman pembayaran merupakan halaman yang digunakan untuk melakukan proses pembayaran sesuai waktu yang telah ditentukan. Berikut tampilan halaman Pembayaran



Gambar 5.7 Halaman Pembayaran

8. Halaman Laporan

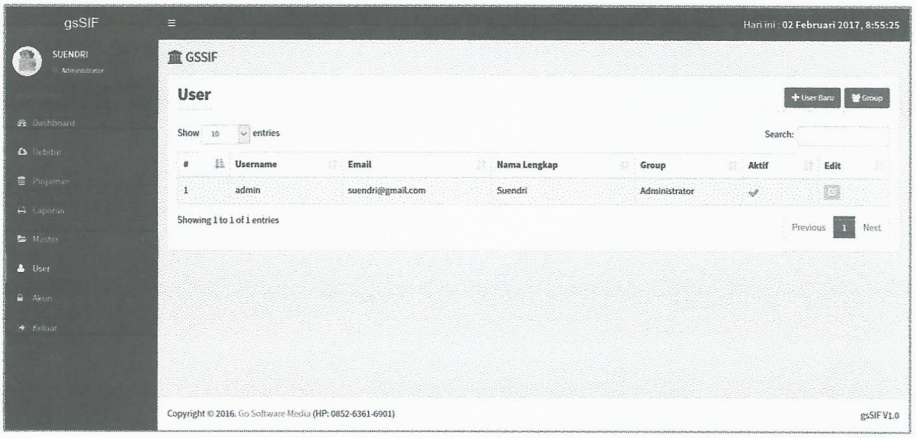
Halaman Laporan merupakan halaman yang menampilkan laporan transaksi yang dilakukan baik proses peminjaman maupun proses pembayaran kredit pada tanggal yang ditentukan. Berikut tampilan laporan



Gambar 5.8 Halaman Laporan

15. Halaman *User*

Halaman *user* merupakan halaman yang menampilkan data seluruh penggunan sistem. Berikut tampilan halaman *user*.



Gambar 5.9 Halaman *User*

5.4 Kesimpulan Pengujian

Dari hasil pengujian yang telah dilakukan, maka dapat ditarik kesimpulan, bahwa sistem yang dibangun telah berjalan dengan baik dan secara fungsional mengeluarkan fungsi sesuai dengan yang diharapkan.

BAB VI
PENUTUP

Berdasarkan uraian dan pembahasan pada bab-bab sebelumnya didapatkan hasil yang bermanfaat pada perancangan sistem informasi finansial menggunakan arsitektur *Model View Controller* (MVC) dan PHP Data *Object* yang merupakan salah satu API dari PHP yang disediakan untuk mengakses database dengan menggunakan teknik *objek-oriented* dan prosedural. Sedangkan untuk pengembangan penelitian berkelanjutkan dituliskan beberapa saran yang bisa diteruskan dan dijadikan kajian untuk bahan penelitian selanjutnya.

6.1 Kesimpulan

Mengacu kepada rumusan masalah yang telah dibahas maka dapat ditarik beberapa kesimpulan.

a. Perancangan sistem informasi finansial menggunakan arsitektur *Model View Controller* dan PHP Data *Object* menjadikan sistem lebih *reusable* untuk pengembangan berkelanjutan. Sistem juga lebih mudah karena pengelompokan antara *Controller*, *Model* yang mengatur database dan *View* yang bertugas untuk menampilkan data kepada *User*.
b. PHP Data *Object* merupakan salah satu solusi terbaik dalam penggunaan DBMS yang berbeda-beda, karena PHP Data Object, *User* dengan mudah melakukan migrasi *database* satu ke *database* yang lainnya. Tingkat kemudahan

keamanan menjadi faktor penting dalam pemilihan cara metode yang digunakan ini.

6.2 Saran

Sebagai bahan penelitian lanjutan, berikut ini adalah beberapa saran yang bisa dijadikan bahan penelitian selanjutnya.

a. Sistem belum dibuat dengan lengkap karena hanya fokus pada penerapan Arsitektur *Mode View Controller* dan PHP Data Objek dalam perancangan sistem.
b. Sistem perlu dilengkapi sesuai sistem yang sedang berjalan pada perusahaan tertentu termasuk print faktur cicilan yang belum dilengkapi.
c. Sistem hanya berjalan pada Server PHP5 dan PHP7 terbaru, tidak lagi support dengan Server dengan PHP4, jika perusahaan masih belum migrasi, maka sebaiknya dilakukan proses migrasi untuk menjalankan sistem.

DAFTAR PUSTAKA

Agung Gregorius. 2012. **Adobe Dreamweaver CS6**. Jakarta: Elex Media Komputindo.

Hartono Bambang. 2013. **Sistem Informasi Manajemen Berbasis komputer.** Bandung: Rineka Cipta

Basuki, Awan Pribadi. 2016. **Konsep dan Implementasi Pemrograman Laravel 5**. Yogyakarta: CV.Lokomedia.

Basuki, Adian Tri. 2011. **Perancangan Aplikasi Sistem Informasi Cuti Karyawan Berbasis Web Pada PT Integrasi Tri Tama Cendekia**. Jakarta: UIN Syarif Hidayatullah.

Blanco, Jose A. and Upton, David. 2009. **CodeIgniter 1.7**. Birmingham:Packt Publishing

Fatima, Siti. 2013. **Perancangan Sistem Informasi Penjualan Mebel Online pada UD. Melindo Jaya**. Kisaran: AMIK Royal Kisaran.

Griffiths, Adam. 2010. **CodeIgniter 1.7 Professional Development**. Birmingham : Packt Publishing

Hartono Bambang. 2013. **Sistem Informasi Manajemen Berbasis komputer**. Bandung: Rineka Cipta.

Jogiyanto. 2005**. Analisis dan desain sistem informasi**. Yogyakarta: Andi Offset.

Kadir, Abdul. 2014. **Pengenalan Sistem Informasi.**Yogyakarta: Andi Yogyakarta.

Kusbianto Deddy. 2010. **Analisis dan Perancangan Sistem Informasi**. Pasuruan: STMIK Yadika Bangil.

Manullang, Harlet Gilbert. 2014. **Sistem Pendukung Keputusan Penentuan Dosen Terbaik dengan Metode AHP pada Universitas Methodist Indonesia**. Medan:USU Press.

Murad, Dina Fitria. 2013. **Aplikasi Intelligence Website Untuk Penunjang Laporan Paud Pada Himpaudi Kota Tangerang**. Jurnal Vol.7 No.1 September 2013. STMIK Raharja Tangerang

Myer, Thomas. 2008. **Professional Codeigniter**. Indianapolis : Wiley Publishing.

Rosa AS, M. Shalahuddin. 2014. **Rekayasa Perangkat Lunak: Terstruktur dan Berorientasi Objek**. Bandung: Informatika.

Simarmata, Janner. 2007. **Perancangan Basis Data**. Yogyakarta: Andi Yogyakarta.

Sutabri Tata. 2014. **Analisis Sistem Informasi**. Yogyakarta: Andi Publisher.

Widodo Prabowo Pujo, Herlawati. 2011. **Menggunakan UML (Unified Modelling Language)**. Bandung: Informatika.

Welling, L., & Thomson, L. 2008. **PHP and MySQL Web Development (4th ed)**. Upperside River: Addison-Wesley Professional.

Yulianti. 2013. **Implementasi Arsitektur Client Server Dan MVC (Model View Controller) Untuk Membangun Aplikasi Administrasi Sekolah (Studi Kasus: SMK Averus Jakarta)**. Jakarta: Universitas Pemulang.