BAB 2
TINJAUAN PUSTAKA

2.1 Arisan

Arisan merupakan kegiatan pengumpulan uang oleh sekelumpok orang dalam periode tertentu kemudian diundi secara berkala untuk menentukan seseorang atau beberapa orang dari anggota yang berhak atas uang yang dikumpulkan [3]. Sebuah arisan selesai dilaksanakan setelah secara keseluruhan anggota telah mendapatkan kesempatan untuk ‘menang’. Arisan tidak lebih sebagai aktivitas menabung, hanya saja terdapat ikatan sosial antar sesama anggota untuk menyelesaikan semua rangkaian kegiatannya. Kegiatan arisan dilakukan pada umumnya secara manual yang merepotkan pengelola: ditulis dalam kertas, penagihan terhadap anggota melalui media komunikasi yang ada, kemudian pembayaran langsung secara cash atau transfer melalui bank.

2.2 Interaksi Sosial

Interaksi sosial merupakan gabungan dari dua kata yaitu interaksi dan sosial, arti interaksi menurut KBBI (Kamus Besar Bahasa Indonesia) adalah hal saling melakukan aksi, berhubungan, mem-pengaruhi, antarhubungan, sedangkan jika digabung dengan kata sosial maka interaksi sosial memiliki pengertian hubungan sosial yang dinamis antara orang perse-orangan dan orang perseorangan, antara perseorangan dan kelompok, dan antara kelompok dan kelompok [4].

2.3 Berbayar *Online*

Berbayar *online* merupakan kalimat yang terdiri dari dua kata yaitu berbayar dan *online*. Berbayar berasal dari kata bayar yang artinya memberikan uang (untuk pengganti harga barang yang diterima, melunasi utang, dan sebagainya), atau bisa juga dalam pengertian memenuhi; menunaikan (janji, nazar, hajat, dan sebagainya); di muka membayar sebelum menerima barang atau jasa; janji memenuhi yang telah dijanjikan; kaul membayar

nazar; kontan membayar tunai (tidak mengangsur atau bertangguh); membayar sekaligus (tidak meminjam atau mengangsur); nazar melaksanakan (memenuhi) yang telah dijanjikan; tunai membayar kontan (tidak mengangsur atau bertangguh) [5].

Online atau dalam bahasa Indonesia biasa disebut daring (dalam jaringan) adalah kegiatan atau servis yang bisa tersedia dengan menggunakan internet atau jaringan komputer [6].

Berbayar *online* secara pengertian berarti memberikan uang sebagai pengganti barang atau jasa dengan penggunaan media berupa internet atau jaringan komputer.

2.4 Payment Gateway

Payment Gateway adalah sebuah penyedia jasa aplikasi *e-commerce* yang mengotorisasi pembayaran untuk *e-business*, *online-retailer* dan lain-lain. Payment gateway melindungi data kartu kredit dengan melakukan enkripsi data nomor kartu kredit, dan memastikan informasi lewat dengan aman antara pembeli dan penjual serta antara penjual dengan payment processor.

Banyak payment gateway juga menyediakan alat-alat untuk menyaring pesanan secara otomatis untuk menghindari penipuan transaksi dan juga menghitung pajak secara *real time* sebelum permintaan otorisasi dikirim ke processor. Alat untuk mendeteksi penipuan termasuk geolocation, kecepatan analisis pola, verifikasi alamat pengiriman, teknologi sidik jari komputer, mengidentifikasi morphing, dan pengecekan AVS dasar [7].

Ketika pelanggan membeli sesuatu dari toko atau mall *online*, *payment gateway* akan memiliki fungsi sebagai berikut.

1. *Authorising* yaitu melakukan verifikasi terhadap kartu kredit atau debit pembeli.
2. *Clearing* yaitu melakukan transfer transaksi kepada rekening bank milik penjual.
3. *Reporting* yaitu mencatat semua transaksi.

Langkah umum dalam transaksi menggunakan *payment gateway*.

Langkah 1 - Konsumen mengunjungi situs belanja dan memilih barang atau jasa. Pesan dikirim ke website berisi keinginan konsumen untuk membeli dan melanjutkan ke tahap pembayaran.

Langkah 2 – Web server toko menerima pesan dari pembeli dan menambahkan sertifikat *digital* untuk mengidentifikasi mall. Pesan ini sekarang disebut "*Digital Order*" dan menyertakan alamat IP pembeli sekaligus jumlah transaksi. *Digital Order* lalu dikirim ke *payment gateway* melalui jaringan aman yang dijamin dengan enkripsi data.

Langkah 3 - Berdasarkan sertifikat *digital*, *payment gateway* mengotentikasi toko *online*.

Langkah 4 - *Payment gateway* menawarkan berbagai pilihan pembayaran pada layar komputer pembeli.

Langkah 5 - Pembeli memilih opsi pembayaran yang diinginkan, yang dikirimkan melalui tautan yang aman ke *payment gateway*.

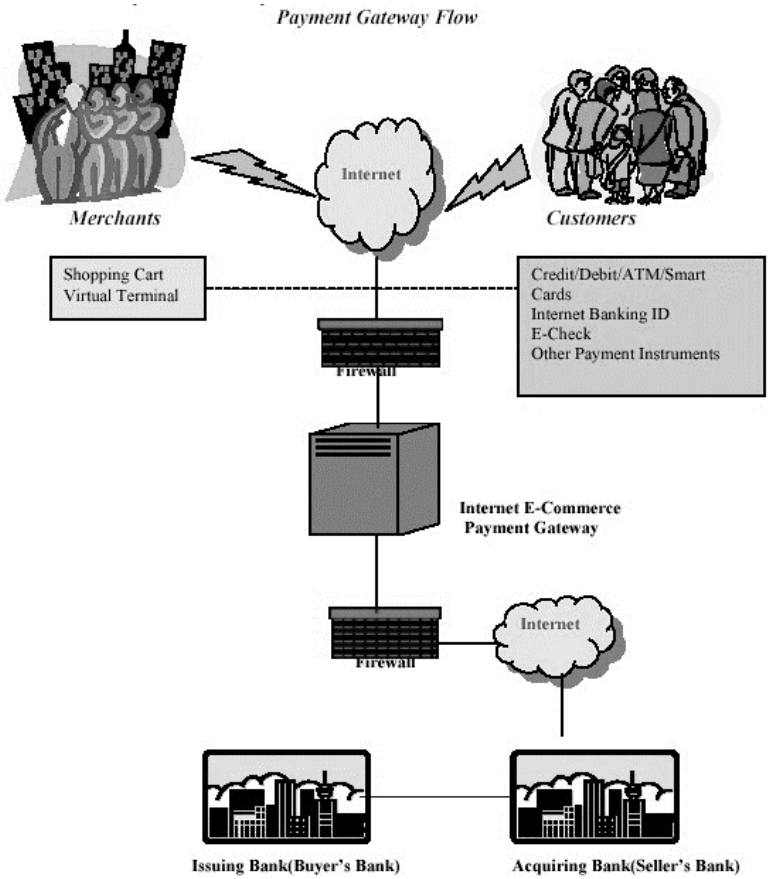
Langkah 6 - *Payment gateway* mengirimkan rincian pembayaran ke bank yang memperoleh pembayaran atau bank milik penjual (*seller's bank* dalam istilah untuk mempermudah penyebutan).

Langkah 7 – *Seller's bank* mengirimkan informasi ke bank milik pembeli (*buyer's bank* dalam istilah untuk mempermudah penyebutan).

Langkah 8 - Berdasarkan batas kredit dan validitas alat pembayaran, *buyer's bank* berhak menerima atau menolak transaksi dan pesan konfirmasi / penolakan ditransmisikan ke *payment gateway* melalui *seller's bank.*

Langkah 9 – *Payment gateway* kemudian mengirimkan tanda terima digital untuk situs belanja begitu juga pembeli.

Langkah 10 – Toko *online* dapat mengirim barang / jasa kepada pembeli.



Gambar 2.1 Alur *payment gateway*

Sebagai catatan tambahan, bahwa berbeda dengan proses secara *offline* yang memakan waktu lama, versi *online* sebagian besar hanya memerlukan waktu 30 hingga 40 detik.

2.5 Internet

Internet (*Interconnected Network)* adalah sebuah sistem komunikasi global yang menghubungkan komputer-komputer dan jaringan-jaringan diseluruh dunia tanpa mengenal batas wilayah, hukum dan budaya, sebagai sarana berkomunikasi dan menyebarkan informasi. Setiap komputer dan jaringan terhubung secara langsung maupun tidak langsung ke beberapa jalur utama yang disebut *internet backbone* dan dibedakan satu dengan yang lainnya menggunakan *unique name* yang biasa disebut dengan alamat IP.

Secara umum internet dipandang sebagai sumber daya informasi, sehingga isi internet adalah informasi yang dapat diibaratkan sebagai suatu basis data perpustakaan multimedia yang sangat besar dan lengkap. Bahkan internet dipandang sebagai dunia dalam bentuk maya, karena hampir seluruh aspek kehidupan di dunia nyata ada di internet, seperti bisnis, hiburan, olah raga, politik, dan lain-lain.

Internet mempunyai peranan yang sangat penting dalam dunia teknologi informasi. Ada dua peranan penting, yaitu:

1. Internet sebagai sumber data dan informasi.
2. Internet sebagai sarana pertukaran data dan informasi.

Internet telah banyak digunakan di berbagai bidang kehidupan dan digunakan oleh perusahaan, lembaga pendidikan, lembaga pemerintahan, lembaga militer, dan lain-lain.

Beberapa contoh manfaat penggunaan internet di berbagai bidang adalah:

1. Bidang Pendidikan

Untuk bidang pendidikan, internet memungkinkan kita untuk mendapatkan banyak referensi keilmuan dari perpustakaan maya (*online library*) dan sebagai media pembelajaran secara *online*, semisal belajar jarak jauh dengan menggunakan Teleconference Internet (*e-learning*).

2. Bidang Ekonomi dan Bisnis

Untuk bidang ekonomi dan bisnis, internet hadir dengan istilah e-commerce. Dengan adanya *e-commerce*, kegiatan perdagangan, jual beli, promosi, dan lain sebagainya dapat dilakukan lewat internet tanpa harus berpergian.

3. Bidang Pemerintahan

Untuk bidang pemerintahan, internet hadir dengan istilah e-government. Dengan adanya e-government, pemerintah dapat dengan mudah memberikan informasi dan layanan kepada masyarakat secara maksimal dan juga dapat dipergunakan untuk saling mempererat hubungan pemerintahan antar suatu negara.

4. Bidang Sosial

Bidang sosial, internet dapat dipergunakan untuk memberikan informasi mengenai berbagai macam kegiatan sosial yang telah, sedang atau akan dilaksanakan dan juga dapat digunakan untuk membantu penggalangan dana kegiatan sosial.

5. Bidang Keagamaan

Internet dapat digunakan untuk sarana diskusi, konsultasi, tanya jawab masalah agama, berbagi ilmu agama, dan lain sebagainya.

2.6 Basis Data

Basis data terdiri dari dua kata, yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang tempat berkumpul, sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek, seperti manusia, hewan, peristiwa, keadaan. Yang direkam kedalam bentuk angka, huruf, simbol, teks, gambar bunyi. Ada juga yang mengatakan bahwa basis data adalah sebagai kumpulan data rekaman, hasil proses pemasukan data dimana basis data harus disimpan dalam sistem basis data yang terintegrasi yang bisa dilakukan secara terpusat maupun secara terdistribusi. [8]

Basis data sendiri, dapat didefinisikan dalam sejumlah sudut pandang seperti :

1. Himpunan kelompok data (arsip) yang saling berhubungan kemudian diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. Kumpulan file / tabel / arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

Basis data dan lemari arsip sesungguhnya memiliki prinsip kerja dan tujuan yang sama. Prinsip utamanya adalah pengaturan data / arsip. Dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data. Perbedaannya hanya terletak pada media penyimpanan yang digunakan.

Suatu hal yang juga harus diperhatikan, bahwa basis data bukan hanya sekedar penyimpanan data secara elektronik (dengan bantuan komputer). Artinya tidak semua bentuk penyimpanan data secara elektronik bisa disebut basis data. Yang sangat ditonjolkan dalam basis data adalah pengaturan / pemilahan / pengelompokan / pengorganisasian data yang akan kita simpan sesuai fungsi / jenisnya. Oleh karena itu, perlu kita ketahui mengenai operasi-operasi dasar yang dapat kita lakukan berkenaan dengan basis data yang meliputi :

1. Pembuatan basis data baru (*create database*), yang identik dengan pembuatan lemari arsip yang baru.
2. Penghapusan basis data (*drop database*), yang identik dengan pengrusakan lemari arsip sekaligus beserta isinya jika ada.
3. Pembuatan file / tabel baru ke suatu basis data (*create table*), yang identik dengan penambahan map arsip baru ke sebuah lemari arsip yang telah ada.
4. Penghapusan file / tabel dari suatu basis data (*drop table*), yang identik dengan pengrusakan map arsip lama yang ada disebuah lemari arsip.
5. Penambahan / pengisisan data baru ke sebuah file / tabel di sebuah basis data (*insert*), yang identik dengan penambahan lembaran arsip ke sebuah map arsip.
6. Pengambilan data dari sebuah file / tabel (*retrieve* / *search*), yang identik dengan pencarian lembaran arsip dari sebuah map arsip.
7. Pengubahan data dari sebuah file / tabel (*update*), yang identik dengan perbaikan isi lembaran arsip yang ada di sebuah map arsip.
8. Penghapusan data dari sebuah file / tabel (*delete*), yang identik dengan penghapusan sebuah lembaran arsip yang ada disebuah map arsip.

Operasi yang berkenaan dengan pembuatan objek (basis data dan tabel) merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya. Sedang operasi yang berkaitan dengan isi tabel merupakan operasi rutin yang akan berlangsung berulang-ulang dan karena itu operasi inilah yang lebih tepat mewakili aktifitas pengelolaan dan pengolahan data dalam basis data.

Didalam basis data terdapat istilah yang menyatakan sebagai urutan data, yaitu :

1. *Characters*, merupakan bagian data yang terkecil yaitu berupa karakter numerik, huruf, maupun karakter spesial (simbol) yang membentuk *field*.
2. *Field*, merepresentasikan suatu atribut dari record yang menunjukan suatu item dari data.
3. *Record*, merupakan kumpulan dari *field*, menggambarkan suatu unit data individu tertentu. Kumpulan *record* membentuk suatu file.
4. File, terdiri dari record-record yang menggambarkan satu kesatuan data yang sejenis.

2.6.1 Objektif Basis Data

Tujuan awal pengelolaan data dalam sebuah basis data adalah agar kita dapat menemukan kembali data tersebut secara cepat dan mudah ketika dibutuhkan. Pemanfaatan basis data untuk pengelolaan data juga memiliki tujuan lain, yaitu:

1. Kecepatan dan kemudahan (*Speed*)

   Pemanfaatan basis data memungkinkan kita dalam menyimpan data, memanipulasi data dengan lebih cepat dan mudah dibanding melakukan penyimpanan secara manual maupun elektronis tetapi tanpa penerapan basis data.
2. Efisiensi ruang penyimpanan (*Space*)

   Basis data memungkinkan kita melakukan optimalisasi penggunaan ruang penyimpanan, karena kita dapat melakukan penekanan jumlah redudansi (pengulangan) data, baik dengan menerapkan pengodean atau membuat relasi-relasi antarkelompok data yang saling berhubungan.
3. Keakuratan (*Accuracy*)

   Penerapan aturan / batasan (*constraint*) tipe data, domain data pada sebuah basis data berguna dalam meningkatkan keakuratan penyimpanan data.
4. Ketersediaan (*Availability*)

   Pemanfaatan teknologi jaringan komputer memungkinkan data pada sebuah basis data dapat diakses dari tempat yang jauh dari pusat penyimpanan basis data sehingga ketersediaannya dapat dimanfaatkan bagi cabang lain.

5. Kelengkapan (*Completeness*)

   Lengkap maupun tidaknya data yang dikelola pada sebuah basis data bersifat relatif. Sebagai contoh, seorang pemakai mungkin menganggap data yang dikelola telah lengkap walaupun pemakai lain mungkin menganggapnya kurang lengkap untuk data yang sama. Untuk mengakomodasi kebutuhan kelengkapan data yang terus berkembang, basis data memungkinkan kita menambah *record* data, perubahan struktur data seperti penambahan tabel baru, *field* baru pada sebuah tabel, dll.

6. Keamanan (*Security*)

   Basis data mampu memberi keamanan data dengan dengan cara menentukan pemakai yang boleh melakukan akses, menentukan jenis operasi apa saja yang boleh dilakukan.

7. Kebersamaan Pemakaian (*Sharability*)

   Pemakai basis data tidak terbatas hanya satu pemakai saja, bahkan bisa diakses di lebih dari satu tempat, sehingga basis data yang dikelola oleh sistem yang mendukung *multi-user* harus dapat memenuhi kebutuhan ini dengan tetap menjaga konsistensi data atau menghindari terjadinya *dead-lock* karena banyak pemakai yang saling menunggu untuk menggunakan data.

2.6.2 Database Management System (DBMS)

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak (sistem) yang khusus / spesifik. Perangkat lunak inilah yang akan menentukan bagaimana data diorganisasi, disimpan, dirubah dan diambil kembali. Perangkat lunak ini disebut dengan *Database Management System* (DBMS). DBMS juga yang menerapkan mekanisme pengamanan data, pemakaian data secara bersama, pemaksaan keakuratan / konsistensi data, dan sebagainya. Pada dasarnya basis data hanyalah sebuah objek yang pasif / mati. Karena itu, secara umum sebuah sistem basis data merupakan sistem yang terdiri atas kumpulan file (tabel) yang saling berhubungan dan sekumpulan program (DBMS) yang memungkinkan untuk dapat diakses dan untuk memanipulasi file-file tersebut.

Salah satu tujuan dari DBMS adalah untuk menyediakan fasilitas / antarmuka (*interface*) yang dapat dilihat oleh pengguna (user) dan digunakan untuk menyederhanakan dan memberi fasilitas yang nyaman untuk mengakses data. Untuk hal itu, sistem akan menyembunyikan detail tentang bagaimana data disimpan dan dipelihara. Karena itu, seringkali data yang terlihat oleh pemakai sebelumnya berbeda dengan yang tersimpan secara fisik. Abstraksi data merupakan tingkatan / *level* dalam melihat data dalam sistem basis data, terdiri atas tiga tingkatan, yaitu :

1. Level fisik, merupakan level terendah yang ada dalam abstraksi data, dimana menunjukan bagaimana sesungguhnya suatu data disimpan. Pada level ini juga, pengguna melihat data sebagai gabungan dari struktur dan datanya sendiri.
2. Level lojik / konseptual, menggambarkan data apa yang sebenarnya disimpan dalam basis data dan hubungannya dengan data yang lain.
3. Level penampakan, merupakan level tertinggi dari abstraksi data yang hanya menunjukan sebagian dari basis data. Banyak pengguna dalam sistem yang tidak terlibat dengan semua data / informasi yang ada / tersimpan. Para pengguna umumnya hanya membutuhkan sebagian dari data / informasi yang terdapat didalam sebuah sistem basis data.

Implementasi dari DBMS memerlukan beberapa struktur data sebagai berikut, diantaranya :

1. File data, yaitu file yang digunakan untuk menyimpan data.
2. Kamus data, yaitu untuk menyimpan informasi struktur basis data dan otorisasi.
3. File indeks, yaitu untuk memberi akses cepat pada data.
4. Data statistik, yaitu untuk menyimpan informasi mengenai data di basis data.

2.6.3 Bahasa Basis Data

DBMS merupakan perantara antara user dengan database. Cara komunikasi diatur dalam suatu bahasa khusus yang telah ditetapkan oleh DBMS

(Contoh: SQL, dBase, QUEL, dsb.). Bahasa database, dibagi dalam 2 bentuk yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

1. *Data Definition Language* (DDL)

DDL memiliki beberapa fungsi, yaitu digunakan dalam membuat tabel baru, indeks, mengubah tabel, menentukan struktur tabel, dsb.

Hasil dari kompilasi perintah DDL berupa kumpulan tabel yang disimpan dalam file khusus: Kamus Data (*Data Dictionary*). *Data Dictionary* merupakan metadata (superdata), yaitu data yang mendeskripsikan data sesungguhnya. Data dictionary ini akan selalu diakses dalam suatu operasi database sebelum suatu file data yang sesungguhnya diakses.

2. *Data Manipulation Language* (DML)

DML memiliki beberapa cakupan, yaitu pemanggilan data yang tersimpan dalam database (*query*), penyisipan / penambahan data baru ke *database* (*Insert*), pengubahan data pada *database* (*Update*), penghapusan data dari *database* (*Delete*).

Terdapat dua jenis DML, yaitu prosedural yang menghendaki pengguna untuk menspesifikasikan data apa yang diperlukan dan bagaimana cara mendapatkan data itu. (Contoh: bahasa C/C++, PL/SQL, dsb.). Nonprosedural yang menghendaki pengguna untuk menspesifikasikan data apa yang dibutuhkan, tanpa harus menspesifikasikan bagaimana cara mendapatkan data tersebut. (Contoh: SQL).

2.6.4 Entity Relationship Diagram (ERD)

Pemodelan sistem memainkan peranan yang penting dalam pengembangan sistem. Pemodelan data kadang-kadang disebut pemodelan *database* karena model data kadang-kadang diimplementasikan sebagai sebuah *database*. Pemodelan data dapat di gambarkan dengan *Entity Relationship Diagram* (ERD).

ERD merupakan suatu diagram yang digunakan untuk menghubungkan antar elemen (*relational condition*), dimana pada tahap selanjutnya dapat diimplementasikan kedalam bentuk tabel relasi.

ERD menggunakan sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar data, pada dasarnya ada 3 macam simbol yang digunakan, yaitu:

1. *Entity* adalah suatu objek yang dapat diidentifikasi dalam lingkaran pemakaian dan sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat.
2. Atribut merupakan elemen dari sebuah *entity* yang berfungsi mendeskripsikan karakter *entity*.
3. Hubungan, sebagaimana halnya entity maka dalam hubungan pun harus dibedakan antara hubungan atau bentuk hubungan antara *entity* dengan isi dari hubungan itu sendiri.

Beberapa macam hubungan antar relasi, antara lain:

1. Satu Ke Satu (*one to one*)

Bentuk relasi antara satu entitas dengan sejumlah satu ke entitas dengan jumlah yang sama.

2. Satu Ke Banyak (*one to many*)

Bentuk relasi dari entitas dengan jumlah satu ke entitas lain yang berjumlah lebih dari satu (entitas dengan banyak alternatif tujuan).

3. Banyak Ke Banyak (*many to many*)

Bentuk relasi yang mendeskripsikan permasalahan yang komplek yaitu hubungan antara entitas yang berjumlah lebih dari satu dengan entitas yang sama.

2.7 Structured Query Language (SQL)

SQL merupakan standar bahasa yang digunakan untuk memanipulasi atau *Data Manipulation Language* (DML) dan mendefinisikan atau *Data Definition Language* (DDL) objek-objek basis data sesuai dengan permintaan (*request*) dari pengguna sistem. Dimana DML adalah sekumpulan perintah-perintah *query* yang digunakan untuk melakukan manipulasi data seperti penyimpanan data ke suatu tabel, lalu kemudian mengubahnya atau menghapusnya atau hanya sekedar menampilkan saja, sedang DDL merujuk pada kumpulan perintah *query* yang

digunakan untuk mendefinisikan objek-objek basis data seperti pembuatan tabel basis data atau indeks primer atau sekunder dan lain-lain. [8]

2.8 Application Programming Interface (API)

API adalah kumpulan definisi subroutine, protokol dan alat untuk membangun perangkat lunak dan aplikasi. Sebuah API yang baik membuatnya lebih mudah untuk mengembangkan program dengan menyediakan semua blok pembangun yang kemudian disatukan oleh programmer. API bisa berbasis web, sistem operasi, sistem *database*, perangkat keras komputer, atau *library software* . Sebuah spesifikasi API dapat mengambil banyak bentuk, tetapi sering termasuk spesifikasi untuk rutinitas, struktur data, *object class*, variabel, atau panggilan jarak jauh. POSIX, Microsoft Windows API, C ++ Standard Template Library, dan Java API adalah contoh dari berbagai bentuk API. Dokumentasi untuk API biasanya disediakan untuk memfasilitasi cara penggunaan. Status API dalam hukum kekayaan intelektual adalah dalam status kontroversial.

Sama seperti antarmuka pengguna grafis, API membuat lebih mudah bagi orang untuk menggunakan program, API memudahkan pengembang untuk menggunakan teknologi tertentu dalam membangun aplikasi. Dengan abstraksi yang mendasari implementasi dan hanya mengekspos objek yang sesuai kebutuhan pengembang, API mengurangi beban kognitif pada *programmer*. Sementara antarmuka grafis untuk email pada *client side* mungkin menyediakan bagi pengguna berupa tombol yang melakukan semua langkah-langkah untuk mengambil dan menyoroti email baru, sebuah API untuk file *input* / *output* memberikan pengembang fungsi yang sama tanpa mengharuskan pengembang memahami operasi sistem file terjadi di belakang layar [9].

2.9 Framework

Framework adalah sebuah abstraksi di mana sebuah perangkat lunak yang melakukan fungsional umum mampu di modifikasi oleh sintaks yang digunakan pengguna untuk menghasilkan sebuah perangkat lunak yang spesifik [10].

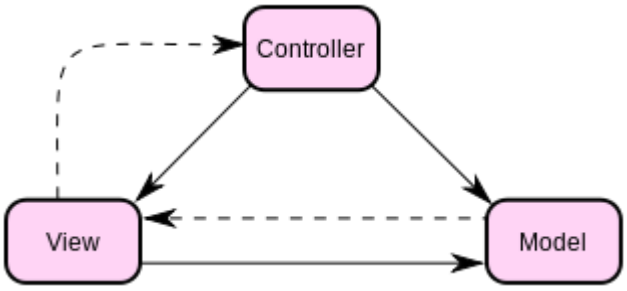
Framework bersifat universal dan mampu digunakan kembali. Berikut ini adalah perbedaan *framework* dengan *library* perangkat lunak pada umumnya:

1. *Framework* memiliki aturan atau *flow control* tersendiri.
2. *Framework* memiliki perilaku tersendiri.
3. *Framework* harus bisa di *extend* untuk digunakan oleh pengguna.
4. Sintaks *framework* tidak bisa dimodifikasi, oleh karena itu pengguna harus meng-*extend* frameiwork untuk menggunakannya.

2.10 MVC (Model-View-Controller)

MVC (Model-View-Controller) adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (Model) dari tampilan (View) dan cara bagaimana memprosesnya (Controller). Dalam implementasinya kebanyakan framework dalam aplikasi website adalah berbasis arsitektur MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna, dan bagian yang menjadi kontrol dalam sebuah aplikasi web, berikut penjelasan komponen pembentuk MVC:

1. Model mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain.
2. View adalah bagian yang mengatur tampilan ke pengguna berupa halaman web.
3. Controller merupakan bagian yang menjembatani model dan view. Controller berisi perintah-perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman web.



Gambar 2.2 Diagram Model-View-Controller

Dengan menggunakan metode MVC maka aplikasi akan lebih mudah untuk dirawat dan dikembangkan. Untuk memahami metode pengembangan aplikasi menggunakan MVC diperlukan pengetahuan tentang pemrograman berorientasi objek (*Object Oriented Programming*).

2.11 Flask

Flask adalah *microframework* untuk bahasa pemrograman Python yang berbasis Werkzeug dan Jinja 2, sekaligus memiliki lisensi BSD yang berarti siapapun bisa menggunakan flask untuk berbagai keperluan baik pribadi maupun komersial tanpa dipungut bayaran (*free software*) asalkan hak cipta flask tetap disertakan [11].

Keunggulan utama dari *framework* flask adalah ukurannya yang ringan dan sederhana, arti dari sederhana adalah bahwa flask memang bisa digunakan untuk menerapkan konsep MVC, namun tidak memaksa pengguna (*strict*) untuk menerapkannya, maka *developer* tetap bisa menggunakan paradigma pemrograman terstruktur.

Masalah utama dalam menggunakan flask sebagai framework adalah karena terlalu sederhana maka *developer* perlu menerapkan desain yang *scalable* dan *modular* (penulisan *script* berdasarkan modul-modul) ketika membuat program dengan flask, salah satu contoh terbaik adalah penerapan konsep MVC.

2.12 Flask-SocketIO

Flask-SocketIO merupakan sebuah *package* bagi aplikasi flask yang memberi akses ke *latency* rendah komunikasi dua arah antara *client* dan *server*. Aplikasi *client-side* dapat menggunakan salah satu *library* resmi SocketIO di Javascript, C++, Java dan Swift, atau *client* yang kompatibel untuk membuat sambungan permanen ke server [12].

2.13 Unified Model Language (UML)

UML adalah sebuah bahasa yang berdasarkan grafik / gambar untuk memvisualisasi, men-spesifikasikan, membangun, dan mendokumentasikan 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*. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat. Berikut ini merupakan beberapa bagian dari UML [13] :

1. *Use Case* Diagram

Menggambarkan sejumlah eksternal aktor dan hubungannya ke *use case* yang diberikan oleh sistem. Komponen utama pembentuk *use case* antara lain:

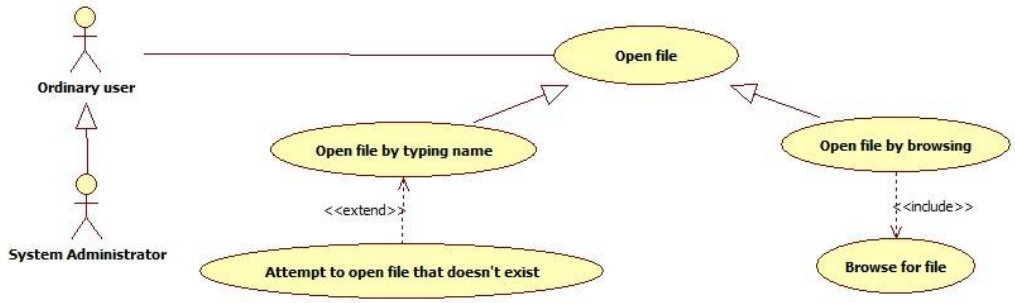
Aktor (actor), menggambarkan suatu hal diluar sistem yang berinteraksi dengan sistem. Use case , kegiatan/aktivitas yang disiapkan oleh sistem. Menekankan pada “apa” yang dikerjakan oleh sistem, bukan “bagaimana” sistem itu bekerja. Hubungan, penjelasan tentang hubungan suatu komponen *use case* diagram dengan komponen lainnya. Pada use case terdapat *use case relationship* yang menunjukkan sifat *reusable*, *optional*, and bahkan menspesialisasikan *use case behavior* antar tiap *use case* yang ada. Terdapat 4 jenis *relationship* pada *use case* yaitu Asosiasi, Generalisasi, <<*include*>> dan <<*extend*>> [14].

Asosiasi digunakan untuk menggambarkan interaksi antara aktor dengan setiap use case tertentu. Relasi ini digambarkan sebagai garis penghubung aktor terhadap *use case* yang berelasi dengan aktor tersebut.

Generalisasi yaitu suatu relasi antara *use case* umum (induk) dan *use case* yang lebih spesifik (anak). Relasi generalisasi memungkinkan *use case* yang lebih spesifik memiliki perilaku (*behaviour*) yang sama dengan *use case* yang lebih umum atau bisa disebut *use case* induk. Relasi generalisasi digambarkan dengan anak panah segitiga. *Use case* yang terletak di sisi anak panah adalah *use case* induk dan yang terletak di sisi lainnya adalah *use case* yang mewarisi perilaku *use case* induk.

<<*include*>> *relationship* memungkinkan terjadinya penambahan perilaku ke dalam *use case* awal yang pada dasarnya use case ini tidak dapat berdiri sendiri tanpa adanya penambahan *use case*, dan *use case* awal tidak akan lengkap tanpa adanya *use case* tambahan ini. *Use case* yang berada pada kepala anak panah adalah *use case* awal, dan pada sisi lain adalah *use case* penambah.

<<*extend*>> *relationship* memungkinkan terjadinya penambahan beberapa *behaviour* ke dalam *use case* awal yang pada dasarnya *use case* tersebut sudah dapat berdiri sendiri tanpa adanya penambahan. *Extend* digambarkan dengan anak panah yang mempunyai garis putus-putus. *Use case* yang berada pada kepala anak panah adalah *use case* awal dan yang berada di lain sisi adalah *use case* tambahan.



Gambar 2.3 Contoh diagram *use case* beserta relasinya [15]

2. *Activity* Diagram

Activity diagram digunakan untuk menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainya seperti *use case*.

3. *Class* Diagram

Menggambarkan struktur dan deskripsi *class, package* (paket) dan objek beserta hubungan satu sama lain seperti *containment* (penahanan), pewarisan, asosiasi dan lain-lain.

4. *Sequence* Diagram

Sequence diagram menggambarkan kolaborasi dinamis antara sejumlah objek dan untuk menunjukan rangkaian pesan yang dikirim antara objek juga interaksi. Sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

2.14 SQLAlchemy

SQLAlchemy adalah *toolkit* Python SQL dan *Object Relational Mapper* yang memberikan pengembang aplikasi kekuatan penuh dan fleksibilitas SQL.

Tool ini menyediakan rangkaian lengkap dari pola pengembangan setingkat *enterprise*, yang dirancang untuk akses *database* yang efisien dan berkinerja tinggi, diadaptasi ke dalam bahasa domain yang sederhana dan mendekati bahasa Python.

Database SQL berperilaku tidak seperti koleksi objek ketika ukuran besar dan kinerja mulai diperhatikan; koleksi objek berperilaku tidak seperti tabel dan baris ketika abstraksi mulai menjadi masalah. SQLAlchemy bertujuan untuk mengakomodasi kedua prinsip ini.

SQLAlchemy menganggap *database* menjadi mesin aljabar relasional, bukan hanya kumpulan tabel. Baris dapat dipilih dari tidak hanya tabel tetapi juga gabungan dan pernyataan lainnya; setiap unit ini dapat terdiri menjadi struktur yang lebih besar. bahasa ekspresi SQLAlchemy dibangun di atas konsep ini dari intinya.

SQLAlchemy dikenal karena ORM (*Object Relational Model*), komponen opsional yang menyediakan pola data *mapper*, di mana kelas dapat dipetakan ke *database* dalam berbagai kemungkinan memungkinkan model objek dan skema *database* untuk mengembangkan secara rapi dan terpisah sejak awal.

2.14.1 Deklarasi Mapping

Ketika menggunakan ORM, proses konfigurasi dimulai dengan menggambarkan tabel database yang akan diolah, dan kemudian mendefinisikan kelas kita sendiri yang kemudian akan dipetakan ke tabel tersebut. Pada SQLAlchemy modern, dua tugas ini biasanya dilakukan bersama-sama, menggunakan sistem yang dikenal sebagai *declarative*, yang memungkinkan kita untuk membuat kelas yang mencakup arahan untuk menggambarkan tabel *database* yang sebenarnya yang akan dipetakan.

Kelas yang dipetakan menggunakan sistem deklaratif lantas didefinisikan dalam hal kelas dasar yang mempertahankan katalog kelas dan tabel relatif

terhadap basis, hal ini dikenal sebagai kelas *declarative base*. Aplikasi yang dibangun biasanya hanya memiliki satu instansiasi dari dasar ini didalam modul yang diimpor. Cara menciptakan kelas dasar menggunakan fungsi declarative_base( ).

```
>>> from sqlalchemy.ext.declarative import declarative_base

>>> Base = declarative_base()
```

Gambar 2.4 Mendefinisikan kelas *declarative base*

Setelah memiliki “dasar”, kita dapat mendefinisikan sejumlah kelas untuk dipetakan. Dimulai dengan hanya satu tabel yaitu *users*, yang akan menyimpan data pengguna yang menggunakan aplikasi. Sebuah kelas baru yaitu kelas *User* akan menjadi kelas yang akan memetakan tabel ini. Dalam kelas, kita mendefinisikan rincian tentang tabel yang kita akan petakan , terutama nama tabel dan tipe data dari kolom.

```
>>> from sqlalchemy import Column, Integer, String
>>> class User(Base):
...     __tablename__ = 'users'
...
...     id = Column(Integer, primary_key=True)
...     name = Column(String)
...     fullname = Column(String)
...     password = Column(String)
...
...     def __repr__(self):
...        return "<User(name='%s', fullname='%s', password='%s')>" % (
...                             self.name, self.fullname, self.password)
```

Gambar 2.5 Mendefinisikan *class*

2.15 iPaymu

iPaymu adalah suatu cara pembayaran *online* ataupun pengiriman uang yang berfungsi untuk memudahkan pengguna dalam bertransaksi dengan menggunakan layanan internet. iPaymu memungkinkan penggunanya melakukan pembayaran untuk produk, belanja *online*, donasi, produk berlangganan, pengiriman uang, dan penarikan uang dengan mudah, cepat dan aman.

Dalam segi keamanan iPaymu menggunakan 256 bit SSL *encryption* dan telah diverifikasi oleh Thawte sehingga pengguna bisa bertransaksi dengan aman.

iPaymu bekerja sama dengan 137 Bank di Indonesia, seperti ditampilkan di tabel berikut:

Tabel 2.1 Daftar Bank yang Bekerja Sama dengan iPaymu

• ANZ PANIN	• BANK TABUNGAN NEGARA
• BANGKOK	• BANK VICTORIA
• BANK AGRIS	• BANK WINDU KENTJANA
• BANK AGRONIAGA	• BANK WOORI
• BANK AKITA	• BANK YUDHA BHAKTI
• BANK ANGLOMAS	• BCA (BANK CENTRAL ASIA)
• BANK ANTAR DAERAH	• BII SYARIAH
• BANK ARTHA GRAHA	• BNI UUS
• BANK ARTOS	• BNP PARIBAS
• BANK BISNIS	• BOA
• BANK BNI 46	• BOT
• BANK BUKOPIN	• BPD ACEH
• BANK BUMI ARTA	• BPD ACEH UUS
• BANK CAPITAL	• BPD BALI
• BANK CHINATRUST	• BPD BANDA ACEH
• BANK COMMONWEALTH	• BPD BENGKULU
• BANK DANAMON INDONESIA	• BPD DIY
• BANK DANAMON SYARIAH	• BPD JABAR
• BANK DIPO	• BPD JAMBI
• BANK DKI	• BPD JATENG
• BANK EKONOMI	• BPD JATIM UUS
• BANK EKSEKUTIF	• BPD KALBAR
• BANK FAMA	• BPD KALBAR UUS
• BANK GANESHA	• BPD KALSEL
• BANK HANA	• BPD KALSEL UUS
• BANK HARDA	• BPD KALTENG
• BANK HARFA	• BPD KALTIM
• BANK HS 1906	• BPD KALTIM UUS
• BANK INA	• BPD LAMPUNG
• BANK INDEX SELINDO	• BPD MALUKU
• BANK INTERNATIONAL INDONESIA	• BPD NTB
• BANK JABAR BANTEN SYARIAH	• BPD NTT
• BANK JASA JAKARTA	• BPD PAPUA
• BANK JATIM	• BPD RIAU
• BANK KESAWAN	• BPD SULAWESI TENGAH

• BANK KESEJAHTERAAN EKONOMI • BANK LIMAN • BANK MANDIRI • BANK MASPION • BANK MAYAPADA INTERNATIONAL • BANK MAYORA • BANK MEGA • BANK MESTIKA • BANK METRO EKPRESS • BANK MITRANIAGA • BANK MIZUHO • BANK MUAMALAT • BANK MULTIARTA SENTOSA • BANK MUTIARA (D/H CENTURY) • BANK NAGARI • BANK NATIONALNOBU • BANK NUSANTARA PARAHYANGAN • BANK OCBC • BANK OF CHINA • BANK PANIN • BANK PERMATA • BANK PRIMA MASTER • BANK PURBA DANARTA • BANK RAKYAT INDONESIA (BRI) • BANK ROYAL • BANK SBI INDONESIA • BANK SINAR HARAPAN BALI • BANK SINARMAS • BANK SRI PARTHA • BANK SUMITOMO • BANK SWADESI • BANK SYARIAH MANDIRI	• BPD SULAWESI TENGGARA • BPD SULSEL • BPD SULUT • BPD SUMSEL • BPD SUMSEL UUS • BPD SUMUT • BPD SUMUT UUS • BTPN • BUMIPUTERA • CHASE • CITIBANK • CNB KANTOR PUSAT • DBS INDONESIA • DEUTSCHE • DKI UUS • HSBC • HSBC SYARIAH • ICBC • IFI • KEB INDONESIA • MAYBANK • NAGARI UUS • OCBC NISP • PERMATA SYARIAH • RABOBANK • RBS • RESONA PERDANIA • STANDARD CHARTERED BANK • SWAGUNA • SYARIAH BRI • SYARIAH BUKOPIN • SYARIAH MEGA • UOB BUANA • UOB INDONESIA