TUGAS AKHIR – SM141501

PENGEMBANGAN APLIKASI WEB BERBASIS *CLOUD* MENGGUNAKAN *FRAMEWORK* UNTUK PENGELOMPOKAN HASIL PENCARIAN TEKS

ZETA DHARMA PRAKASA
NRP 1212 100 097

Dosen Pembimbing
Dr. Budi Setiyono, S.Si, MT

JURUSAN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT – SM141501

CLOUD-BASED WEB APPLICATION DEVELOPMENT USING FRAMEWORK FOR TEXT-SEARCH RESULT GROUPING

ZETA DHARMA PRAKASA
NRP 1212 100 097

Supervisor
Dr. Budi Setiyono, S.Si, MT

DEPARTMENT OF MATHEMATICS
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2017

LEMBAR PENGESAHAN

PENGEMBANGAN APLIKASI WEB BERBASIS *CLOUD* MENGGUNAKAN *FRAMEWORK* UNTUK PENGELOMPOKAN HASIL PENCARIAN TEKS

CLOUD-BASED WEB APPLICATION DEVELOPMENT USING FRAMEWORK FOR TEXT-SEARCH RESULT GROUPING

Diajukan untuk memenuhi salah satu syarat
Memperoleh gelar Sarjana Sains
Pada bidang minat Matematika Ilmu Komputer
Program Studi S-1 Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember

Oleh :
ZETA DHARMA PRAKASA
NRP. 1212 100 097

Menyetujui,
Pembimbing

Dr. Budi Setiyono, S.Si, MT
NIP. 19720207 199702 1 001

Mengetahui,
Ketua Jurusan Matematika
FMIPA-ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Surabaya, 2017

PENGEMBANGAN APLIKASI WEB BERBASIS *CLOUD* MENGGUNAKAN *FRAMEWORK* UNTUK PENGELOMPOKAN HASIL PENCARIAN TEKS

Nama	**: Zeta Dharma Prakasa**
NRP	**: 1212 100 097**
Jurusan	**: Matematika**
Dosen Pembimbing	**: Dr. Budi Setiyono, S.Si, MT**

Abstrak

Cloud computing atau dikenal dengan komputasi awan merupakan gabungan pemanfaatan teknologi komputer dalam suatu jaringan berbasis internet. Salah satu pengembangan teknologi tersebut adalah mesin pencari informasi. Tugas akhir ini membuat aplikasi mesin pencari menggunakan *framework* CodeIgniter dan API untuk mengambil data yang ada di internet dan mengelompokan data tersebut menjadi beberapa kategori, yaitu buku, jurnal, gambar(JPG), gambar(GIF) dan video. Sistem yang dibuat dengan teknologi *cloud* ini memiliki keakuratan 100% kesesuaian isi dengan kategori buku dari 200 data, jurnal dari 111 data, gambar(JPG) dari 293 data dan video dari 23 data, serta 44% untuk kategori gambar(GIF) dari 30 data. Kemudian jenis *keyword* yang digunakan juga berpengaruh terhadap kesesuain pencarian dan kecepatan pencarian. *Keyword* dengan Bahasa Inggris memiliki rata-rata 96% informasi yang dihasilkan sesuai dengan *keyword* dan rata-rata kecepatan 45.276 detik dalam satu kali pencarian, sedangkan Bahasa Indonesia memiliki rata-rata 66% informasi yang dihasilkan sesuai dengan *keyword* dan rata-rata kecepatan 28.46 detik dalam satu kali pencarian.

Kata Kunci—*Cloud computing*, *framework* CodeIgniter, *API*, pengelompokan teks

CLOUD-BASED WEB APPLICATION DEVELOPMENT USING FRAMEWORK FOR TEXT-SEARCH RESULT GROUPING

Name	**:**	**Zeta Dharma Prakasa**
NRP	**:**	**1212 100 097**
Department	**:**	**Mathematics FMIPA-ITS**
Supervisor	**:**	**Dr. Budi Setiyono, S.Si, MT**

Abstract

Cloud computing is known by the combined utilization of computer technology within a network with cloud-based development. One of the technological development is the search engine information. The final project is to make the search engine application using CodeIgniter framework and API for retrieving data that is in the cloud and the data are grouped into several categories, namely books, journals, videos, GIF and JPG. Systems with cloud technology has 100% accuracy of the suitability of the contents by category of books with 200 data, journals 111 data, video with 23 data, JPG with 293 data and 44% for the category of GIF with 30 data. Then type the keyword that is used also affects the accuracy of search and search speed. Keyword with the English Language has an average of 96% of the information produced in accordance with keyword and an average speed of 45,276 seconds in a single search, while the Indonesian Language has an average of 66% of the information produced in accordance with keyword and an average speed of 28.46 seconds in a single search.

Keyword : *Cloud computing, framework CodeIgniter, API, text grouping*

KATA PENGANTAR

Puji syukur penulis panjatkan pada kehadirat Allah Swt. karena hanya dengan karunia rahmat, bimbingan, serta anugrah-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **Pengembangan Aplikasi Web Berbasis *Cloud* Menggunakan *Framework* untuk Pengelompokan Hasil Pencarian Teks.**

Dalam proses pembuatan Tugas Akhir ini, penulis mendapat bantuan dan dukungan dari berbagai pihak. Untuk itu penulis mengucapkan terima kasih kepada :

1. Ketua Jurusan Matematika FMIPA-ITS yang telah memberi dukungan dan kemudahan pengurusan persyaratan-persyaratan selama penulis menyelesaikan Tugas Akhir ini.
2. Bapak Dr. Budi Setiyono, S.Si, MT sebagai dosen pembimbing Tugas Akhir atas segala bimbingan dan motivasi yang telah diberikan pada penulis.
3. Ibu Dr. Dwi Ratna S, S.Si, MT, Bapak Drs. Daryono Budi Utomo, M.Si, Dr. M. Yunus, M.Si dan Kistosil Fahim, S.Si, M.Si selaku dosen penguji atas semua saran yang telah diberikan untuk perbaikan Tugas Akhir ini.
4. Bapak Dr. Didik Khusnul Arif, S.Si, M.Si selaku Kaprodi dan Bapak Drs. Iis Herisman, M.Sc selaku Sekretaris Kaprodi S1 Jurusan Matematika ITS yang telah memberi dukungan dan kemudahan pengurusan persyaratan-persyaratan selama penulis menyelesaikan Tugas Akhir ini.
5. Ibu Dian Winda Setyawati, S.Si, M.Si selaku dosen wali penulis yang telah memberikan motivasi dan arahan akademik.
6. Bapak dan Ibu Dosen serta seluruh staf Tata Usaha dan Laboratorium Jurusan Matematika FMIPA-ITS.
7. Keluarga tercinta terutama Bapak Darmalisan dan Ibu Kartini, penulis ucapkan banyak terima kasih atas doa serta dukungan yang telah diberikan baik moral maupun material, serta Kamelia Marchsenda dan Novika Darmela Santin yang

telah memberikan semangat dalam penyusunan Tugas Akhir ini.

8. Luffi Aditya Sandy yang telah membantu dalam pembuatan program Tugas Akhir dan Dinan Fakhrana Ramadhani yang telah banyak membantu selama penulis mengerjakan Tugas Akhir.
9. Teman – teman seperjuangan Matematika ITS 2011 dan 2012 khususnya MAT12IKS tercinta yang telah banyak membantu baik secara langsung maupun tidak.
10. Seluruh pihak yang tidak dapat penulis sebutkan satu persatu, yang turut membantu dalam penyusunan Tugas Akhir ini.

Penulis menyadari bahwa selama masa penelitian dan penyusunan laporan ini masih banyak kekurangan dan kesalahan. Oleh karena itu, penulis memohon saran dan kritik sebagai bahan perbaikan di masa yang akan datang. Semoga Tugas Akhir ini bermanfaat bagi semua pihak.

Surabaya, Januari 2017

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang semakin maju hingga saat ini, membuat kita umat manusia hidup di masa globalisasi atau bisa disebut dengan masa modernisasi. Menurut Kamus Besar Bahasa Indonesia (KBBI), modernisasi itu adalah proses pergeseran sikap dan mentalitas sebagai warga masyarakat untuk dapat hidup sesuai tuntutan masa kini. Salah satu contoh tuntutan pada masa sekarang adalah perkembangan ilmu pengetahuan dan teknologi, dimana hal ini membuat teknologi menjadi kebutuhan dasar bagi individu masing-masing. Hampir dari semua orang menggunakan teknologi dalam berbagai aspek kehidupannya. Kebutuhan manusia akan teknologi juga didukung dengan teknologi yang semakin maju dengan banyaknya penemuan terbaru yang memudahkan manusia dalam kehidupan sehari-hari. Salah satu teknologi tersebut adalah adanya internet.

Internet adalah rangkaian hubungan jaringan komputer yang dapat diakses secara umum diseluruh dunia, yang mengirimkan data dalam bentuk paket data berdasarkan standar *internet protocol* (IP) [16]. Untuk lebih jelasnya internet merupakan sebuah perpustakaan besar yang menyimpan informasi dimana semua orang bebas berkunjung dan bertukar informasi setiap saat. Sebuah perusahaan Amerika Serikat yaitu Google inc. adalah perusahaan yang bergerak pada jasa dan produk internet. Salah satu produknya adalah sebuah aplikasi pencarian yang juga disebut *google search*. *Google search* memanfaatkan teknologi internet untuk membuat sebuah layanan pencari informasi yang disajikan dalam bentuk web, dimana web ini merupakan bagian dari bentuk internet. Orang-orang dapat memanfaatkan keuntungan dari layanan tersebut untuk membantu mereka mencari dan mendapatkan informasi atau data yang mereka inginkan dalam hitungan detik. Namun, karena data dan

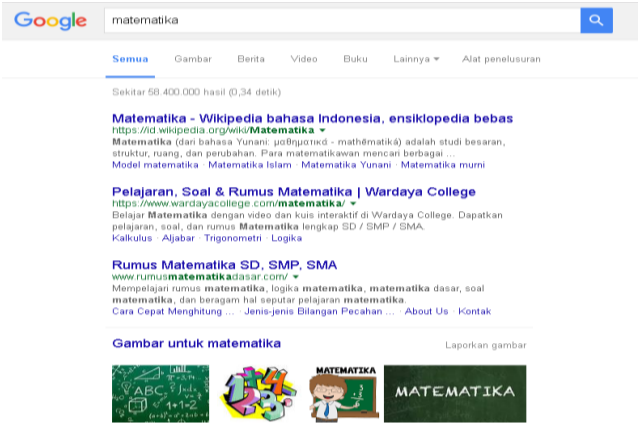
informasi yang disimpan di perpustakaan Google sangat banyak, layanan tersebut belum bisa memberikan informasi dengan sangat detail. Terkadang membutuhkan waktu yang lebih untuk mencari informasi yang sesuai dengan keinginan. Ini juga dikarenakan ambigunya permintaan dari *user* dan terbatasnya keakuratan dari sistem pencarian [6]. Contohnya ketika *user* atau pengguna menggunakan layanan *google* untuk mencari *file* bertipe PDF, namun jika pengguna tidak menambahkan kata PDF pada *keyword*, maka akan menghasilkan informasi yang sifatnya umum seperti pada Gambar 1.1. Berbeda ketika pengguna menambahkan kata PDF pada *keyword*, hasil yang diperoleh merupakan *link* bertipe PDF yang bisa langsung diunduh atau *download* seperti pada Gambar 1.2.

Internet merupakan bagian terpenting yang tidak dapat dipisahkan dari implementasi sistem pencarian informasi, karena tanpa adanya internet para pengguna tidak dapat menikmati keuntungan dari layanan tersebut. Oleh karena itu, dengan memanfaatkan teknologi komputer dan berkembangnya internet dapat menghasilkan suatu layanan baru dalam bidang teknologi informasi yang disebut dengan komputasi awan (*cloud computing*).

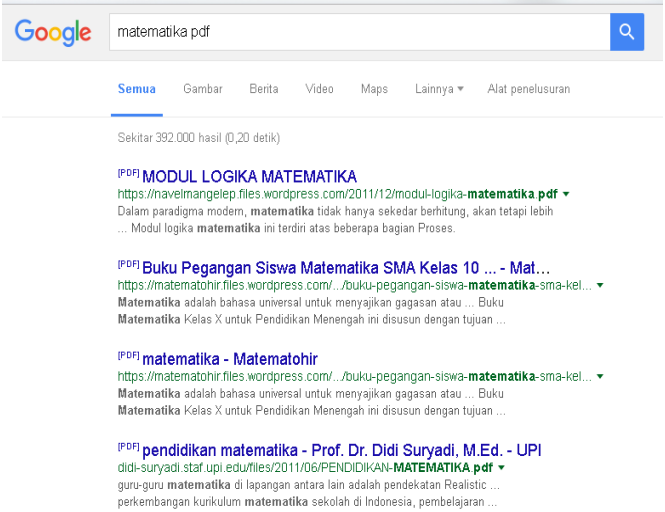
Cloud computing merupakan gabungan pemanfaatan teknologi komputer dalam suatu jaringan dengan pengembangan berbasis internet yang mempunyai fungsi untuk menjalankan program atau aplikasi melalui komputer. *Cloud computing* adalah sumber daya dan jasa yang ada di internet dengan semua data tersimpan di server secara terpusat. Jasa *cloud* dikirim dari data yang terletak di pusat ke seluruh dunia. *Cloud computing* memfasilitasi para pengguna dengan menyediakan sumber daya virtual melalui internet[1]. Penggunaan *cloud computing* mulai diterapkan karena komputasi berbasis *cloud* dinilai memberikan kemudahan dan keuntungan dimana pengguna bisa menjalankan aplikasi yang dibutuhkan tanpa menginstalnya terlebih dahulu dengan terhubung ke internet.

Salah satu penelitian yang telah dilakukan sebelumnya memanfaatkan teknologi *cloud computing*, yaitu membuat sebuah *prototype* dari implementasi *framework* berbasis *cloud* dengan sejumlah catatan medis dari *Respiratory Medicine Department.* Pada penelitian ini diusulkan *framework* berbasis *cloud* sebuah layanan peduli kesehatan yang dinamai *Home-diagnosis. Home-diagnosis* menyediakan layanan kepada pengguna untuk mendapatkan bantuan diagnosis berdasarkan catatan riwayat medis yang tersedia [17].

Terkait dengan penelitian yang pernah dilakukan sebelumnya, penulis akan mencoba mengembangkan metode sebelumnya dengan memanfaatkan *framework* untuk membuat sebuah mesin pencari *link website* berbasis *cloud* sederhana dengan menggunakan API *(Application Programming Interface)* untuk mendapatkan *metadata* atau informasi data dari dokumen yang ada di internet. *Metadata* atau informasi data dari API adalah berupa teks. *Metadata* ini akan dipisah sesuai dengan kategori yang telah ditentukan dan akan menghasilkan sebuah informasi *link website* yang telah dikelompokan berdasarkan kategori tersebut.



Gambar 1.1 Pencarian dengan kata kunci matematika



Gambar 1.2 Pencarian dengan kata kunci matematika pdf

1.2 Rumusan Masalah

Rumusan masalah dari Tugas Akhir ini adalah :

1. Bagaimana merancang aplikasi web berbasis *cloud* dengan menggunakan *framework* untuk pengelompokan hasil pencarian teks?
2. Bagaimana mengimplementasikan dan menggunakan API *(Application Programming Interface)* kedalam sebuah program untuk mendapatkan dan mengolah *metadata*?

1.3 Batasan Masalah

Batasan Masalah yang akan dibahas dalam penelitian tugas akhir ini adalah sebagai berikut :

1. Pencarian informasi data berdasarkan *metadata*.
2. Memberikan 5 kategori pada hasil pencarian, yaitu buku, jurnal, gambar(JPG), gambar(GIF) dan video.

3. Menggunakan *Framework* CodeIgniter.
4. Menggunakan 3 macam API yaitu Googlebook API, Flickr API dan ScienceDirect API.
5. Menggunakan DBMS (*Data Base Management System)* yaitu MYSQL untuk membantu pengolahan data.
6. Implementasi menggunakan *script* PHP dan SQL.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah :

1. Merancang aplikasi web berbasis *cloud* dengan menggunakan *framework* untuk pengelompokan hasil pencarian teks.
2. Mengimplementasikan dan menggunakan API *(Application Programming Interface)* kedalam sebuah program untuk mendapatkan dan mengolah *metadata*.

1.5 Manfaat

Manfaat dari tugas akhir ini adalah membantu pihak-pihak tertentu untuk pembelajaran atau penelitian dalam penerapan aplikasi web berbasis *cloud* untuk hasil pencarian informasi yang telah dikelompokan sesuai kategori yang bersesuaian dengan kata kunci (*keyword)* yang diinginkan *user* dengan memanfaatkan *metadata* dari dokumen yang tersimpan di internet. Kemudian untuk memberikan kemudahan untuk *user* yang ingin mencari informasi dengan kategori tertentu seperti buku atau jurnal.

1.6 Sistematika Penulisan Tugas Akhir

Sistematika penulisan didalam Tugas Akhir ini adalah sebagai berikut:

1. BAB I PENDAHULUAN
   Bab ini menjelaskan tentang latar belakang pembuatan tugas akhir, rumusan dan batasan permasalahan yang dihadapi dalam penelitian tugas akhir, tujuan dan manfaat pembuatan tugas akhir dan sistematika penulisan tugas akhir.

2 BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tentang penelitian sebelumnya dan penjelasan permasalahan dalam tugas akhir ini yang meliputi *cloud computing,* karakteristik dan model layanan *cloud*, struktur *framework* CodeIgniter, API (*Application Programming Interface*), DBMS (*Data Base Management System*), PHP dan SQL.

3 BAB III METODOLOGI PENELITIAN

Bab ini berisi metodologi atau urutan pengerjaan yang dilakukan dalam menyelesaikan tugas akhir, meliputi studi literatur, pengumpulan data, analisis dan desain sistem, pembuatan program, uji coba dan evaluasi, hingga penulisan tugas akhir.

4 BAB IV ANALISIS DAN PERANCANGAN

Bab ini menjelaskan tentang analisis sebuah sistem aplikasi web dan analisis kebutuhan sistem. Selain itu dijelaskan juga mengenai gambaran sistem, penjelasan umum sistem, perancangan alur sistem, perancangan penggunaan API dan perancangan halaman antarmuka. Pada perancangan alur sistem terdapat proses menentukan kategori untuk hasil aplikasi *search engine* yang diinginkan, pengecekan *history keyword,* pengambilan dan penyimpanan data serta skema *database*. Kemudian bab ini juga menjelaskan alur penggunaan 3 API, yaitu Google API, ScienceDirect API dan Flickr API.

5 BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini mengimplementasikan dari perancangan yang dilakukan pad bab sebelumnya dan menampilkan hasil uji coba aplikasi web pencarian yang telah dibuat meliputi perbandingan aplikasi dengan mesin pencari yang sudah ada

sebelumnya, keakuratan data yang dihasilkan dengan kata kunci yang diberikan *user*, kecocok data dengan kategori dan kecepatan aplikasi dalam mencari data. Hasil pengujian inilah yang akan digunakan dalam perumusan kesimpulan dan saran.

6 BAB VI PENUTUP

Bab ini merupakan penutup, berisi tentang kesimpulan yang dapat diambil berdasarkan data yang ada dan saran yang selayaknya dilakukan bila tugas akhir ini dilanjutkan.

BAB II
TINJAUAN PUSTAKA

Pada bab ini dijelaskan menjelaskan tentang kajian teori dari referensi penunjang serta penjelasan permasalahan yang dibahas dalam tugas akhir ini, meliputi penelitian sebelumnya terkait Tugas Akhir ini, terdiri dari *cloud computing*, karakteristik dan model layanan *cloud*, struktur *framework* CodeIgniter, API (*Application Programming Interface*), DBMS (*Data Base Management System*), PHP dan SQL.

Penelitian Sebelumnya

Penelitian terkait tugas akhir ini yang dilakukan oleh Liang-Chi Hsieh dkk yang berjudul *Online image search result grouping with MapReduce-based image clustering and graph construction for large-scale photos*. Penelitian ini menggunakan metode *cluster* gambar dan *Graph Construction* untuk gambar dengan skala besar yang hasilnya akan dikelompokan. Penelitian ini menggunakan dataset yang tersedia pada situs yang menyediakan data informasi gambar yaitu Flickr [6].

Penelitian terkait tugas akhir ini tentang *framework* berbasis *cloud* dilakukan oleh Wenmin Lin dkk yang berjudul *A cloud-based framework for home-diagnosis service over big medical data.* Dalam penelitian ini, *Hadoop framework* digunakan untuk mengimplementasikan sebuah *self-caring service* yang dinamakan *Home-diagnosis.* Berdasarkan hasil penelitian tersebut, *Home-diagnosis* menyediakan layanan kepada pengguna untuk mendapatkan bantuan diagnosis berdasarkan catatan riwayat medis yang tersedia. Seratus riwayat medis dari rumah sakit yang ada di Lianyungan pada bagian *Respiratory Medicine Department* digunakan sebagai dataset [17].

2.1 *Cloud Computing*

Istilah '*cloud computing'* mengacu pada jaringan sistem yang dapat memberikan pengguna dengan akses yang efisien dan cocok untuk sumber daya komputasi dengan jumlah besar yang mudah untuk dikelola dan tersedia dimana saja [2]. Model *cloud* mempunyai beberapa ciri khusus seperti *on-demand self-service, broad network access, resource pooling, rapid elasticity*, dan *measured service. Cloud computing* mengacu pada kedua layanan di internet, *hardware* dan *software* dibutuhkan untuk menyediakan layanan yang lainnya [3]. Suatu layanan dikatakan *cloud* apabila memiliki karakteristik yang menjadikan sebuah layanan tersebut sebagai layanan *cloud* dan layanan *cloud* itu sendiri terbagi menjadi 3 jenis, yaitu *Cloud Software as a Service (SaaS), Cloud Platform as a Service (PaaS), Cloud Infrastructure as a Service (IaaS).*

2.1.1 Karakteristik *Cloud Computing*

Lima karakteristik penting dari *cloud computing* [3], yaitu :

1. *On-demand self-service* yaitu konsumen dapat menentukan kemampuan komputasi secara sepihak, seperti *server time* dan *network storage*, secara otomatis sesuai kebutuhan tanpa memerlukan interaksi manusia dengan masing-masing penyedia layanan.

2. *Broad network access* yaitu kemampuan yang tersedia melalui jaringan dan diakses melalui mekanisme standar yang mengenalkan penggunaan berbagai *platform*

3. *Resource pooling* yaitu penyatuan sumber daya komputasi yang dimiliki penyedia untuk melayani beberapa konsumen virtual yang berbeda, ditetapkan secara dinamis dan ditugaskan sesuai dengan permintaan konsumen. Ada rasa kemandirian lokasi bahwa pelanggan pada umumnya tidak

memiliki kontrol atau pengetahuan atas keberadaan lokasi sumber daya yang disediakan, tetapi ada kemungkinan dapat menentukan lokasi di tingkat yang lebih tinggi (misalnya, negara, negara bagian, atau *data center*). Contoh sumber daya termasuk penyimpanan, pemrosesan, memori, *bandwidth* jaringan, dan mesin virtual.

4. *Rapid Elasticity* yaitu kemampuan dapat ditetapkan dan dirilis secara elastis, dalam beberapa kasus dilakukan secara otomatis untuk menghitung keluar dan masuk dengan cepat sesuai dengan permintaan. Untuk konsumen, kemampuan yang tersedia yang sering kali tidak terbatas dan kuantitasnya dapat disesuaikan setiap saat.

5. *Measured Service* yaitu sistem *cloud computing* secara otomatis mengawasi dan mengoptimalkan penggunaan sumber daya dengan memanfaatkan kemampuan pengukuran (*metering*) pada beberapa tingkat yang sesuai dengan jenis layanan (misalnya, penyimpanan, pemrosesan, *bandwidth*, dan *account* pengguna aktif). Penggunaan sumber daya dapat dipantau, dikendalikan, dan dilaporkan sebagai upaya memberikan transparansi bagi penyedia dan konsumen dari layanan yang digunakan.

2.1.2 Model Layanan *Cloud Computing*

Tiga model layanan dari *cloud computing* [2], adalah sebagai berikut :

1. *Cloud Software as a Service (SaaS)* yaitu kemampuan yang diberikan kepada konsumen untuk menggunakan aplikasi penyedia dapat beroperasi pada infrasuktur *cloud*. Aplikasi dapat diakses dari berbagai perangkat klien melalui antarmuka seperti *web browser* (misalnya, *email* berbasis web). Konsumen tidak mengelola atau mengendalikan infrastruktur

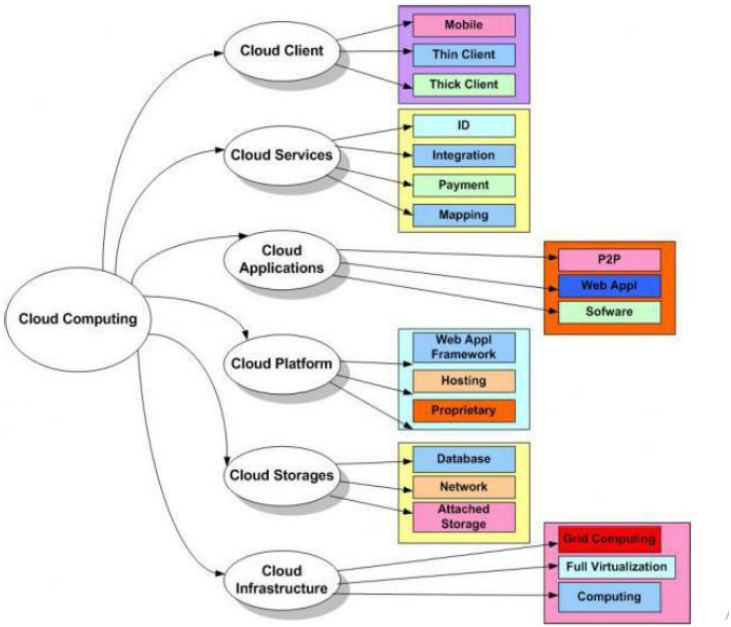
cloud yang mendasar termasuk jaringan, server, sistem operasi, penyimpanan, atau bahkan kemampuan aplikasi individu, dengan kemungkinan pengecualian terbatas terhadap pengaturan konfigurasi aplikasi pengguna tertentu. Contohnya adalah Google Apps, SalesForce.com dan aplikasi jejaring sosial seperti FaceBook.

2. *Cloud Platform as a Service (PaaS)* yaitu kemampuan yang diberikan kepada konsumen untuk menyebarkan aplikasi yang dibuat konsumen atau diperoleh ke infrastruktur *cloud* computing menggunakan bahasa pemrograman dan peralatan yang didukung oleh provider. Konsumen tidak mengelola atau mengendalikan infrastruktur *cloud* yang mendasar termasuk jaringan, server, sistem operasi, atau penyimpanan, namun memiliki kontrol atas aplikasi yang disebarkan dan memungkinkan aplikasi melakukan hosting konfigurasi. Contohnya yang sudah mengimplementasikan ini adalah Force.com dan Microsoft Azure investment.

3. *Cloud Infrastructure as a Service (IaaS)* yaitu kemampuan yang diberikan kepada konsumen untuk memproses, menyimpan, berjaringan, dan sumber komputasi penting yang lain, dimana konsumen dapat menyebarkan dan menjalankan perangkat lunak secara bebas, yang dapat mencakup sistem operasian aplikasi. Konsumen tidak mengelola atau mengendalikan infrastruktur *cloud* yang mendasar tetapi memiliki kontrol atas sistem operasi, penyimpanan, aplikasi yang disebarkan, dan mungkin kontrol terbatas komponen jaringan yang pilih (misalnya, *firewall host*). Contohnya seperti Amazon *Elastic Compute Cloud* dan Simple Storage Service.

2.1.3 Mekanisme *Cloud Computing*

Mekanisme akses ke *cloud computing* dapat dijalankan secara beraneka ragam mulai dari akses standar LAN maupun intranet dengan sedikit aplikasi agen atau klien, sampai kepada akses extranet dan internet melalui *browser* yang terhubung ke sebuah portal aplikasi dari penyedia layanan *cloud computing*. Protokol aplikasi yang digunakan pun dapat beragam, tetapi hal ini tidaklah terlalu signifikan bila dilihat dari sisi pengguna dimana pengguna akhir cukup mengetahui bagaimana cara mengakses dan mempergunakan jasa layanan yang terdapat pada *Cloud* computing [4].



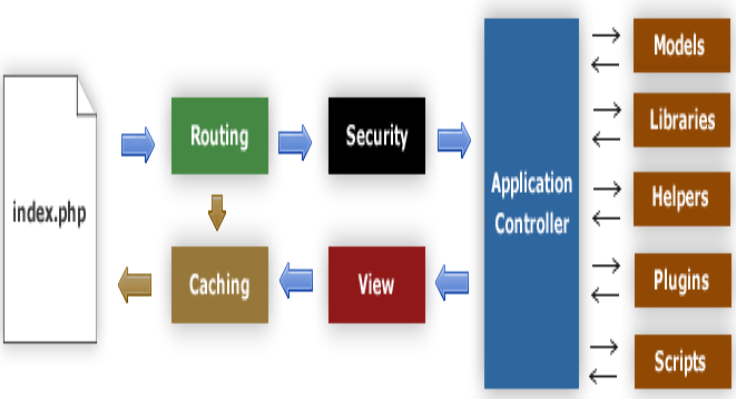
Gambar 2.1 Struktur *cloud computing* [4]

2.2 Framework

Framework adalah kumpulan perintah atau fungsi dasar yang membentuk aturan-aturan tertentu dan saling berinteraksi satu sama lain sehingga dalam pembuatan aplikasi *website*, harus mengikuti aturan dari *framework* tersebut [5]. *Framework* dapat diartikan sebagai komponen pemrograman yang siap pakai kapan saja, sehingga seorang *programmer* tidak perlu membuat *script* yang sama untuk permasalahan yang sama. *Framework* itu sendiri memiliki fungsi yang berbeda-beda serta cara pemakaiannya juga berbeda-beda. Untuk pemrograman aplikasi berbasis web ada beberapa *framework* yang menyediakan fungsi-fungsi yang mudah digunakan, salah satunya adalah CodeIgniter.

2.2.1 CodeIgniter

CodeIgniter adalah sebuah *framework* aplikasi web yang bersifat *open source* yang berfungsi untuk membangun aplikasi dengan bahasa pemrograman PHP secara dinamis. CodeIgniter memiliki banyak fitur yang membuat *framework* ini berbeda, salah satunya adalah dokumentasi yang lengkap yang mencakup seluruh aspek dalam *framework*. CodeIgniter dibangun menggunakan konsep MVC (*Model-View-Controller*) *development pattern*. CodeIgniter juga mampu berjalan pada lingkungan *shared hosting* karena memiliki ukuran yang sangat kecil, namun memiliki kinerja yang cepat [7]. Proses data mengalir pada sistem yang menggunakan menggunakan *framework* CodeIgniter dapat dilihat pada Gambar 2.2



Gambar 2.2 *Flow chart* data pada sistem [8]

Keterangan :

1. *Index.php* berfungsi sebagai *front controller,* meginisialisasi *base resource* untuk menjalankan CodeIgniter.
2. *Router* memeriksa HTTP *request* untuk menentukan apa yang harus dilakukan.
3. Jika *Cache* aktif, maka hasilnya akan langsung dikirimkan ke *browser* dengan mengabaikan aliran data normal.
4. *Security* berfungsi sebagai keamanan dimana sebelum *controller* dimuat, HTTP *request* dan data yang akan dikirim ke *user* akan di *filter*.
5. *Controller* memuat *model, core libraries, plugins, helpers,* dan semua *resource* yang diperlukan untuk proses *request*.
6. *View* yang dihasilkan akan dikirim ke *user*. Jika *chace* aktif, maka *view* akan disimpan sebagai *chace* sehingga pada *request* selanjutnya langsung ditampilkan.

2.2.2 Model-View-Controller (MVC)

MVC adalah *pattern* atau pola pemrograman yang memisahkan desain, data, dan proses. MVC menstrukturisasi aplikasi dengan cara tersebut untuk mempromosikan penggunaan kembali dari kode program[7]. Adapun komponen-komponen MVC antara lain:

1. *Model*

   Model berhubungan dengan data dan interaksi ke *database* atau *webservice*. Model merepresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentuk *file* teks, *file* XML maupun *webservice*. *Model* akan berisi *class* dan fungsi untuk mengambil, melakukan *update* dan menghapus data *website*. Sebuah aplikasi web menggunakan basis data dalam menyimpan data, maka pada bagian *Model* biasanya akan berhubungan dengan perintah-perintah *query* SQL.

2. *View*

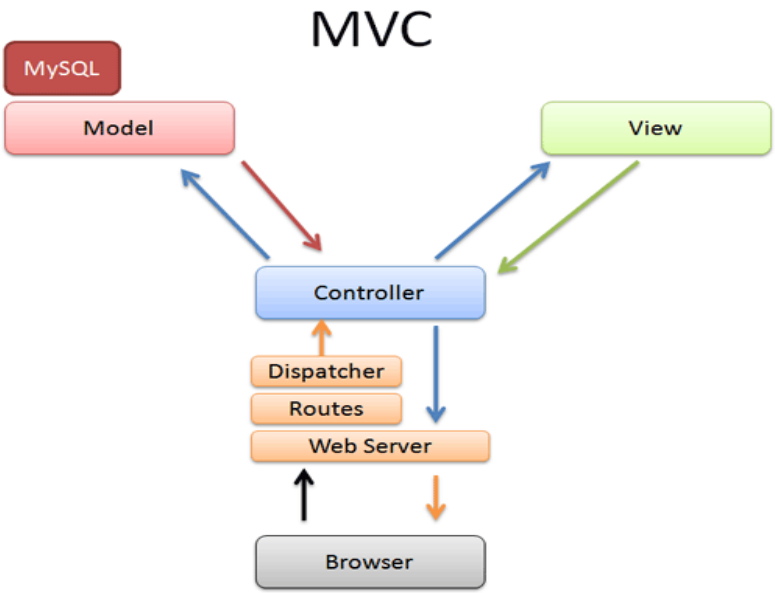
   View berhubungan dengan segala sesuatu yang akan ditampilkan ke *end-user* seperti halaman web, rss, javascript dan lain-lain. Di dalam *view* hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. *view* dapat dikatakan sebagai halaman antarmuka *website* yang dibuat dengan menggunakn HTML dan bantuan CSS atau JavaScript. Di dalam *view* jangan tidak terdapat kode perintah untuk melakukan koneksi ke *database*. *View* hanya dikhususkan untuk menampilkan data-data hasil dari *Model* dan *Controller*.

3. *Controller*

   Controller bertindak sebagai penghubung *model* dan *view*. *Controller* inilah didalamnya terdapat *class* dan fungsi yang memproses permintaan dari *view* ke dalam struktur data di dalam *model*. *Controller* juga berisi kode perintah untuk

mengakses *database* karena tugas mengakses data telah diserahkan kepada *model*. Tugas *Controller* adalah menyediakan berbagai variabel yang akan ditampilkan di *view*, memanggil *model* untuk melakukan akses ke *database*, menyediakan penanganan eror, mengerjakan proses logika dari aplikasi serta melakukan validasi terhadap input.

Alur diagram dari *framework* CodeIgniter dengan MVC dapat dilihat pada Gambar 2.3



Gambar 2.3 *Flow chart* MVC CodeIgniter [8]

2.3 *Database*

Database adalah kumpulan *item* data yang saling berhubungan satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di *hardware* dan *software* guna melakukan manipulasi untuk

kegunaan tertentu [13]. *Database* juga merupakan informasi yang disimpan didalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil *query* dari *database* disebut *Database Management System* (DBMS). Salah satu perangkat lunak yang sering digunakan adalah MySQL.

2.3.1 *Database Management System* (DBMS)

Database Management System adalah seperangkat program komputer yang didesain untuk mengatur sebuah basis data yang disimpan secara terstruktur, dan melakukan operasi-operasi data sesuai dengan permintaan penggunanya yang bertujuan untuk mengakses dan mengelola data tersebut [13].

2.3.2 MySQL

MySQL adalah sistem manajemen *database* yang bersifat *open source*. MySQL dibuat dan dikembangkan oleh MySQL AB yang berada di Swedia.

MySQL merupakan sistem manajemen *database* yang bersifa relasional. Artinya data-data yang dikelola dalam *database* akan diletakan pada beberapa tabel yang terpisah sehingga proses manipulasi data akan menjadi cepat [12]. MySQL dapa digunakan untuk mengelola *database* mulai dari yang kecil hingga yang sangat besar. MySQL juga dapat menjalankan perintah-perintah *Structure Query Language* (SQL) untuk mengelola *database* relasional yang ada didalamnya.

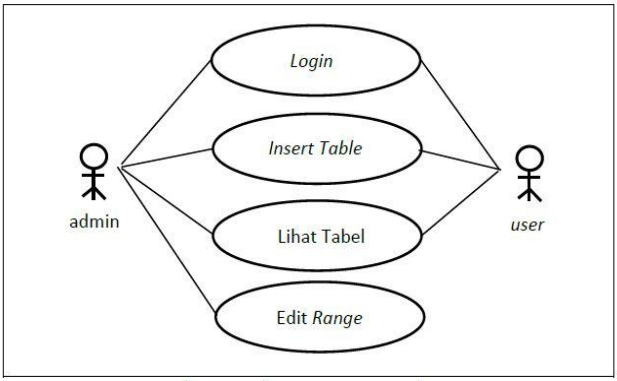
2.4 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak [9]. *Unified Modeling Language* (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta

aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut.

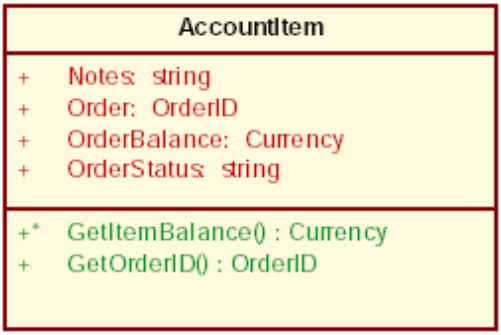
UML dapat memvisualisasikan, menetapkan, membuat dan mendokumentasikan aplikasi pada *software* yang akan dibangun. Saat sistem *software* menjadi lebih besar dan lebih kompleks, maka diperlukan penyederhanaan untuk lebih mudah dalam penggunaan [10]. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantic. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Ada beberapa diagram yang terdapat pada pemodelan UML namun hanya beberapa diagram yang sering digunakan yaitu sebagai berikut :

1. *Use Case Diagram*
   Use case diagram digunakan untuk memodelkan semua bisnis proses berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas diagram untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang akan berinteraksi dengan sistem aplikasi [9].



Gambar 2.4 Contoh *use case diagram* [10]

2. *Class Diagram*
   Class diagram menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara, yaitu *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (*group* bersama sebagai satu unit) [9]. *Class* memiliki tiga area pokok, yaitu nama, atribut dan metoda [10]. Contoh *class diagram* dapat dilihat pada Gambar 2.5.

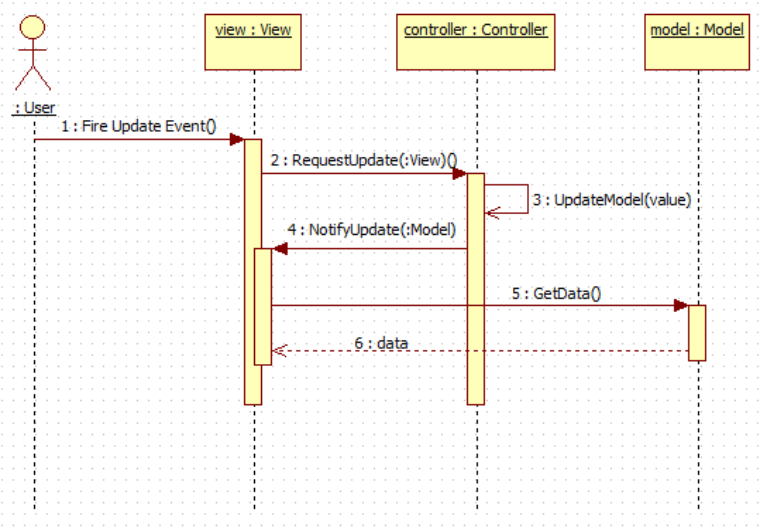


Gambar 2.5 Contoh *class diagram* berdasarkan nama, atribut dan metoda [10]

Pada Gambar 2.5 terdapat contoh nama *class* yang terdapat pada kolom pertama, atribut pada kolom kedua dan metoda atau operasi pada kolom ketiga.

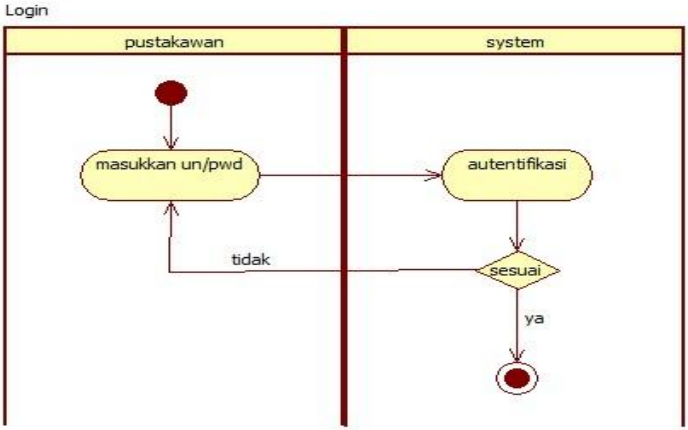
3. *Sequence Diagram*
   Diagram *sequence* merupakan salah satu diagram *Interaction* yang menjelaskan bagaimana suatu operasi

itu dilakukan, *message* atau pesan apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Contoh *sequence diagram* dapat dilihat pada Gambar 2.6



Gambar 2.6 Contoh *sequence diagram*

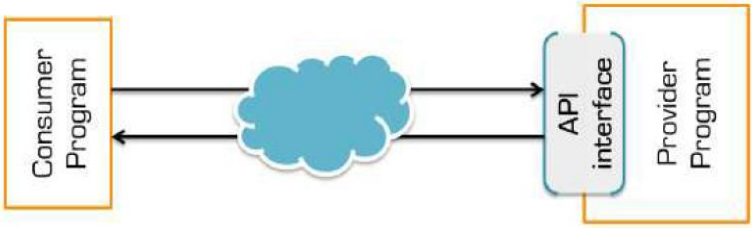
4. *Activity Diagram*
   Diagram *activity* digunakan untuk memodelkan perilaku *Use Cases* dan *objects* di dalam sistem. Pada diagram *activity* terdapat proses yang membuat sebuah keputusan dengan pilihan seperti pada Gambar 2.7



Gambar 2.7 Contoh *activity diagram* [10]

2.5 *Application Programming Interface* (API)

API merupakan *software interface* yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk *library* dan menjelaskan bagaimana agar suatu *software* dapat berinteraksi dengan *software* lain[13].



Gambar 2.8 Skema konektivitas API antar *software* [14]

Secara struktural, API merupakan spesifikasi dari suatu *data structure*, *objects*, *functions*, beserta parameter-parameter yang

diperlukan untuk mengakses *resource* dari aplikasi tersebut. Seluruh spesifikasi tersebut membentuk suatu *interface* yang dimiliki oleh aplikasi untuk berkomunikasi dengan aplikasi lain, dan API dapat digunakan dengan berbagai bahasa *programming*, ataupun hanya dengan menggunakan URL (*Uniform Resource Locator*) yang telah disediakan oleh suatu *website* [13].

2.6 XAMPP Server

XAMPP adalah program aplikasi pengembangan yang berguna untuk pengembangan *website* berbasis PHP dan MySQL. XAMPP berfungsi 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 [11]. Web server Apache berfungsi untuk simulasi pengembangan *website*. Melalui aplikasi ini, developer dapat menguji aplikasi secara langsung dari komputer tanpa perlu terkoneksi dengan internet. XAMPP juga dilengkapi fitur manajemen *database* PHPMyAdmin, sehingga pengembangan web berbasis *database* dapat dilakukan dengan mudah.

2.7 PHP

PHP adalah bahasa pemrograman *server-side* yang didesain untuk pengembangan web. Disebut bahasa pemrograman *server-side* karena PHP diproses pada komputer server. Hal ini berbeda dengan bahasa pemrograman *client-side* seperti JavaScript yang diproses pada web *browser* (*client*).

Untuk membuat halaman web, PHP bukanlah bahasa pemrograman yang wajib digunakan. Website dapat dibuat hanya dengan menggunakan HTML tetapi website yang dihasilkan merupakan website statis dimana konten dan halaman web bersifat tetap. Dengan menggunakan PHP, website yang dibuat dapat berupa website dinamis yang artinya tampilan konten tergantung situasi. Website dinamis juga dapat menyimpan data

ke dalam *database*, membuat halaman yang berubah-ubah sesuai *input user*, dan lain-lain.

2.8 JSON

JavaScript Object Notation (JSON) merupakan suatu *format* pertukaran data yang dirancang menjadi *format* yang mudah dibaca dan ditulis oleh manusia, serta mudah untuk dihasilkan dan diproses oleh komputer [15].

Format pertukaran data JSON tidak bergantung pada bahasa pemrograman yang dipakai, namun susunan data-nya menggunakan aturan yang mirip dengan bahasa pemrograman C. JSON diperkirakan dapat menguraikan data seratus kali lebih cepat dibandingkan dengan XML di *modern browser*.

BAB III
METODE PENELITIAN

Bab ini membahas mengenai metodologi sistem yang digunakan untuk menyelesaikan tugas akhir. Metodologi penelitian yang digunakan berfungsi sebagai acuan sehingga penelitian dapat berjalan sistematis.

3.1 Studi Literatur

Pada tahap ini akan dilakukan pencarian jurnal-jurnal ilmiah, pencarian buku dan sumber referensi lainnya. Mempelajari dan memahami konsep CodeIgniter, konsep sebuah API, bagaimana mengimplementasikan API untuk mendapatkan *metadata* dari server yang menyediakannya, serta pengelompokan berdasarkan kategori dari *metadata* yang telah didapat dengan menggunakan *database* MYSQL.

3.2 Analisis dan Perancangan Sistem

Pada tahapan ini akan dilakukan analisis sistem, analisis kebutuhan sistem, perancangan algoritma, perancangan antarmuka sistem, dan perancangan *database*. Sistem ini memiliki *inputan* ke *database*, *inputan* tersebut adalah *metadata* berupa teks yang disediakan oleh server. *Metadata* ini diperoleh melalui *keyword* yang diberikan oleh *user* dengan menggunakan algoritma yang ada, sehingga *metadata* yang diinginkan sesuai dengan *keyword* dari *user* tersebut.

3.3 Implementasi Sistem

Algoritma yang sudah didapat diimplementasikan menggunakan *framework* CodeIgniter *script* PHP, serta menggunakan API pada algoritma yang telah dibuat pada *script* PHP.

3.4 Pengujian Sistem

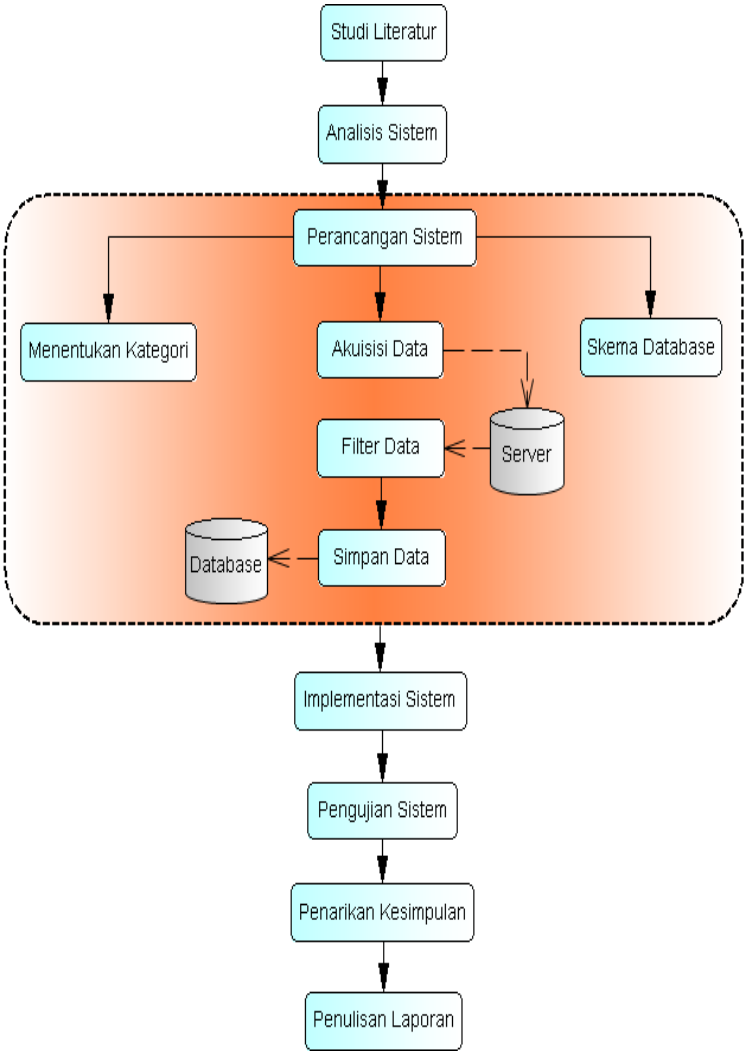
Pada tahap ini akan dilakukan pengujian terhadap aplikasi dengan mencoba beberapa *keyword* untuk mengetahui apakah *metadata* yang telah didapat mengelompok sesuai kategori yang diinginkan dan tampil pada interface aplikasi tersebut. Kemudian pengujian terhadap kesesuaian isi dengan *keyword* dan kecepatan aplikasi dalam mencari serta perbandingan aplikasi dengan mesin pencari yang telah ada sebelumnya.

3.5 Kesimpulan dan Saran

Tahap penarikan kesimpulan merupakan proses dimana dilakukan penarikan kesimpulan terhadap hasil yang telah dicapai, serta diberikan saran sebagai masukan dan perbaikan untuk penelitian selanjutnya

3.6 Diagram Alir

Tahap-tahap pengerjaan Tugas Akhir yang telah dijelaskan di atas digambarkan dalam diagram alir pada Gambar 3.1 sebagai langkah-langkah yang dilakukan untuk mencapai tujuan dari penelitian.



Gambar 3.1 Diagram alir metodologi penelitian

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai analisis dan perancangan sistem dimulai dari deskripsi sistem dan kebutuhan sistem, menentukan kategori yang akan ditampilkan, penggunaan API, penggunaan DBMS yaitu MYSQL untuk penyimpanan dan pengolahan data, serta penjelasan mengenai alur sistem pengelompokan teks yang sesuai dengan tujuan dari penelitian Tugas Akhir ini.

4.1 Analisis Sistem

Pada sub bab ini menjelaskan proses analisis dalam membangun aplikasi web pencarian. Analisis sitem terbagi menjadi 2, yaitu deskripsi sistem dan analisis kebutuhan sistem. Deskripsi sistem merupakan penjelasan umum sistem secara keseluruhan. Analisis kebutuhan sistem merupakan analisis terhadap kebutuhan sistem seperti perangkat keras dan perangkat lunak.

4.1.1 Deskripsi Sistem Secara Umum

Aplikasi yang akan dibangun dapat membantu pihak-pihak dalam pengembangan aplikasi web berbasis *cloud* untuk pencarian informasi yang telah dikelompokan berdasarkan kategori yang bersesuaian dengan *keyword* atau kata kunci. Menggunakan DBMS untuk pengolahan data yang telah didapat dan menggunakan *framework* CodeIgniter untuk membangun aplikasi berbasis web. Selain itu aplikasi ini juga menggunakan 3 API yaitu Google API, Flick API dan ScienceDirect API yang dapat membantu pihak-pihak yang ingin mengetahui bagaimana mengimplementasikan API pada sistem aplikasi berbasis web.

Inputan sistem berupa *keyword* atau kata kunci untuk mencari data yang berhubungan dengan kata kunci tersebut. Kemudian dengan teknologi *cloud computing*, sistem akan

mengambil data dari server yang telah terhubung, dan data akan disimpan ke *database* yang selanjutnya akan ditampilkan ke halaman antarmuka. *Cloud computing* pada sistem terdapat pada saat penggunaan API. API merupakan perintah yang digunakan untuk mengakuisisi data dari server yang menyediakan. Jadi sistem dikatakan berbasis *cloud* karena sistem memiliki komputasi yang menggunakan API dimana proses komputasi ini terjadi di *cloud* (internet) untuk mendapatkan data dari 3 server yang berbeda.

4.1.2 Analisis Kebutuhan Sistem

Aplikasi ini dibangun dengan menggunakan *software* XAMPP yang fungsinya adalah sebagai server yang berdiri sendiri (*localhost*), menggunakan *software* Sublime Text sebagai *editor script* PHP, aplikasi Google Chrome untuk menampilkan halaman *interface* dan DBMS (*Data Base Management System*) yang digunakan adalah MYSQL.

Tabel 4.1 Tabel kebutuhan sistem

Perangkat Keras	Prosesor	: AMD A10-5750M APU with Radeon™ HD Graphics (4 CPUs), ~2.5 GHz
	RAM Memory : 4 GB	
Perangkat Lunak	Sistem Operasi : Windows 8.1 Pro 64-bit	
	Tools	: XAMPP Sublime Text Google Chrome MYSQL

Ada beberapa aspek yang dibutuhkan sistem dalam membangun sebuah aplikasi pencarian yang menjadi dasar untuk perancangan sistem adalah sebagai berikut :

1. Sistem mampu melakukan query pengambilan data dari server yang berbeda, yaitu Google, ScienceDirect dan Flickr.
2. Sistem mampu mengelompokan data berdasarkan 5 kategori yaitu buku, jurnal, gambar(JPG), gambar(GIF) dan video.
3. Sistem mampu menyimpan data ke *database* berdasarkan kategori
4. Sistem mampu menyaring kembali data yang diberikan oleh masing-masing server sesuai dengan kebutuhan
5. Sistem mampu menampilkan kembali data yang telah dimasukkan ke *database* dalam bentuk informasi dari masing-masing kategori

4.2 Perancangan Sistem

Untuk memenuhi kebutuhan tersebut diperlukan perancangan sistem dimana perancangan sistem ini merupakan tujuan dari dari tugas akhir yaitu merancang aplikasi web berbasis *cloud* dengan menggunakan framework untuk pengelompokan hasil pencarian teks. Sistem yang akan dibangun menggunakan API dari 3 server yang berbeda, yaitu Google, ScienceDirect dan Flickr. Ketiga server tersebut digunakan karena memberikan layanan API secara gratis dimana syarat untuk penggunaan API adalah menggunakan API Key.

Ketiga server tersebut memiliki data yang berbeda-beda dan data yang diambil dari ketiga server tersebut akan dikelompokan menjadi 5 kategori, yaitu buku, jurnal, gambar(JPG), gambar(GIF) dan video. 5 kategori tersebut ditentukan dengan menyesuaikan data yang didapat dari masing-masing server. Seperti Google memberikan data buku, ScienceDirect memberikan data buku dan jurnal, serta Flickr memberikan data berupa gambar dan video. Data dari setiap server ini yang digunakan untuk menentukan kategori pada sistem.

Sistem menggunakan perangkat lunak manajemen basis data yaitu MySQL yang fungsinya untuk mengelola data yang telah diambil dari server. Tujuannya untuk menyatukan data dengan kategori yang sama dimana data yang akan dimasukkan kedalam *database* disaring terlebih dahulu. Hal ini dilakukan karena hasil output dari sistem nantinya hanya berupa informasi yang penting seperti judul, tahun pembuatan, pengarang dan link untuk kategori buku Setelah semua informasi masuk kedalam *database*, informasi yang ada pada *database* ditampilkan sesuai dengan kategori masing-masing. Kemudian selanjutnya perancangan sistem yang meliputi gambaran sistem, perancangan alur sistem, perancangan alur penggunaan API, perancangan *database* dan perancangan halaman antarmuka.

4.2.1 Menentukan Kategori

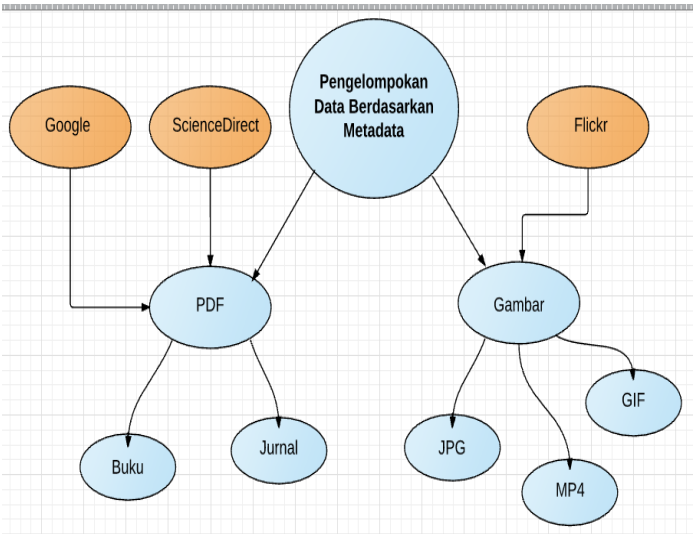
Pada perancangan sistem sebelumya dijelaskan bahwa terdapat 5 kategori yang nantinya menjadi output dari sistem dimana kategori tersebut ditentukan berdasarkan data-data yang diberikan oleh masing-masing server.

Pada Google terdapat data berupa buku dan majalah, namun Google API memberikan fungsi dimana semua data yang diberikan adalah buku jika tidak diberi parameter untuk mencari majalah. Pada ScienceDirect memiliki banyak jenis data, namun data yang jumlahnya lebih dominan adalah data buku dan jurnal, sehingga data ScienceDirect yang digunakan untuk menentukan kategori adalah buku dan jurnal. Kemudian Flickr memberikan data berupa gambar dan video. Data gambar dibagi menjadi kategori JPG dan GIF dimana kedua kategori tersebut dibedakan dengan menggunakan parameter yang ada pada Flickr API. Kemudian data video dimasukkan kedalam kategori video. Adapun kategori yang diberikan berdasarkan data dari server adalah sebagai berikut :

1. Buku diambil dari server Google dan ScienceDirect.
2. Jurnal diambil dari server ScienceDirect

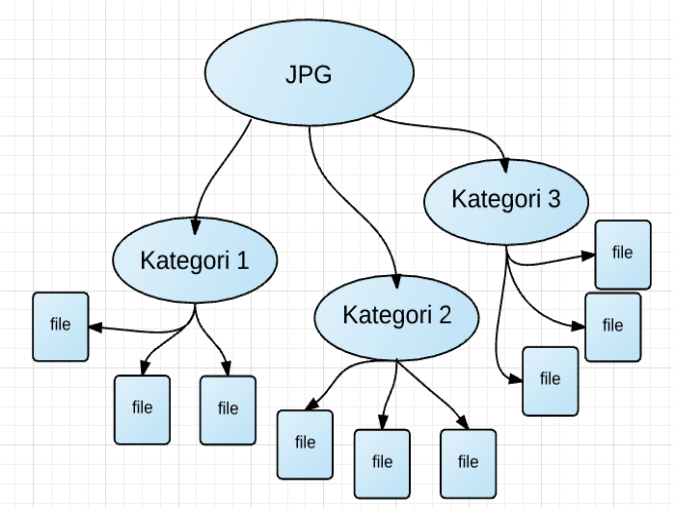
3. Gambar diambil dari server Flickr
4. Gambar Bergerak diambil dari server Flickr
5. Video diambil dari server Flickr

Buku dan jurnal mempunyai tipe data yang sama, yaitu PDF. Untuk menghasilkan informasi yang lebih spesifik, PDF dipisah menjadi buku dan jurnal, sehingga terjadi pengelompokan data berdasarkan *metadata* atau konten dari masing-masing data. Untuk Gambar, GIF dan Video berbeda dengan Buku dan Jurnal, karena ketiga kategori tersebut dikelompokan berdasarkan tipe data. Untuk gambar biasa bertipe JPG, gambar bergerak bertipe GIF dan Video bertipe MP4 dan semua data tersebut berada pada satu server. Pengelompokan data tersebut dapat dilihat pada Gambar 4.1.



Gambar 4.1 Pengelompokan data

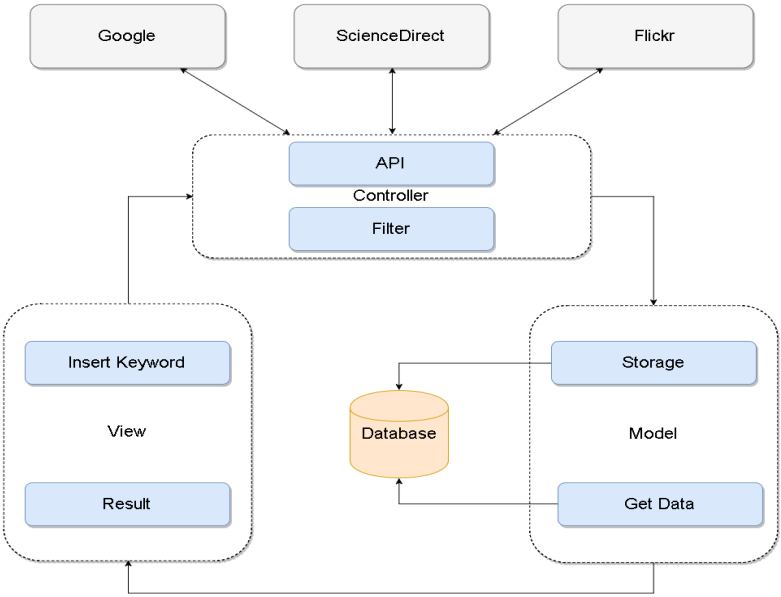
Untuk gambar(JPG) diberikan beberapa kategori lagi untuk memberikan kemudahan *user* memilih gambar(JPG). Pengelompokan dilakukan berdasarkan *tag* atau konten yang ada pada *metadata* gambar tersebut, sehingga *user* dapat memilih gambar apa yang diinginkan dengan diberi pilihan kategori. Gambaran pengelompokan kategori dapat dilihat pada gambar 4.2



Gambar 4.2 Pengelompokan gambar(JPG)

4.2.2 Gambaran Sistem

Berikut gambaran tentang arsitektur dari sistem aplikasi pencarian yang berupa proses berjalannya sistem secara umum yang disajikan pada Gambar 4.3. Proses berjalannya sistem terbagi menjadi 3 yaitu pemeriksaan kata kunci yang digunakan, pengambilan data ke server, dan penyimpanan data ke *database*.

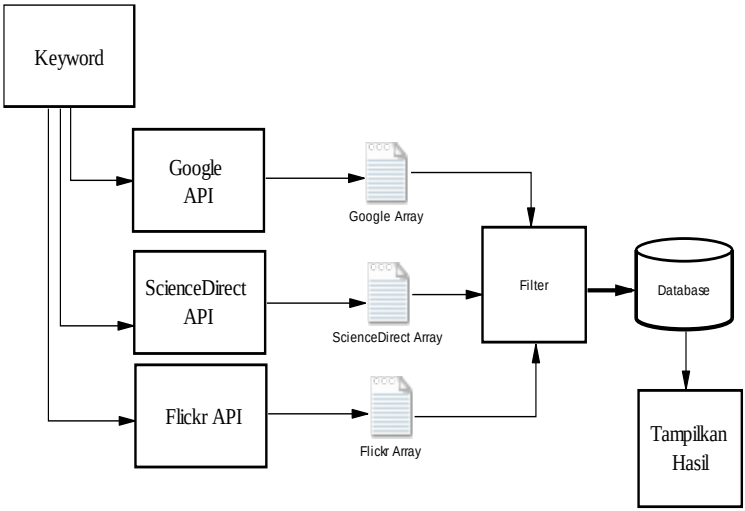


Gambar 4.3 Gambaran umum sistem

Sistem melakukan pemeriksaan terhadap *keyword* yang digunakan sebelum mengakuisisi data. Selanjutnya dengan menggunakan API data diambil berdasarkan kategori yang telah ditentukan. Setiap API memiliki algoritma yang berbeda dalam proses pengambilan data di server. Algoritma pada masing-masing API berfungsi untuk mengambil data sesuai dengan kategori yang sudah ditentukan. Data yang diambil berupa teks dengan format JSON dan kemudian disimpan ke dalam *database*. Data yang masuk ke *database* di tampilkan pada halaman antarmuka. Pada tampilan hasil terdapat 5 kategori pencarian yang telah didapat yaitu buku, jurnal, gambar(JPG), gambar bergerak (GIF) dan Video. Perancangan dibagai menjadi beberapa tahapan sebagai berikut :

a. Desain Pencarian Data

Saat melakukan pencarian *keyword* yang digunakan selalu menjadi parameter dalam menentukan apa yang akan dicari dan menentukan apa yang akan ditampilkan ketika data sudah masuk ke dalam *database*. Perancangan sistem dalam pencarian dengan menggunakan *keyword*



Gambar 4.4 Desain pencarian data

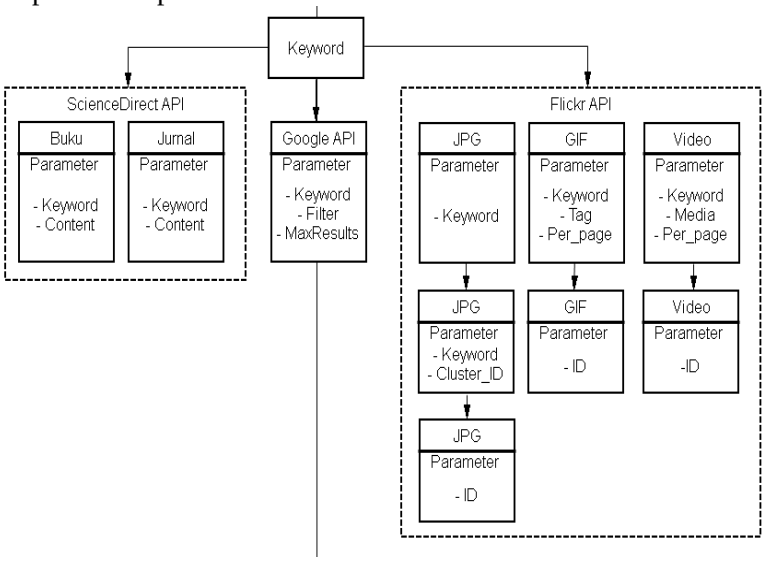
Pada Gambar 4.4 menjelaskan tentang proses bagaimana mengambil data yang ada di ketiga server berdasarkan *keyword* yang telah dimasukkan dengan menggunakan API. Pada saat query dijalankan, API dan *keyword* tersebut akan langsung terdistribusi ke server masing-masing untuk melakukan akuisisi data. API akan mencari semua data yang berhubungan dengan *keyword* yang di inputkan.

Kemudian dari masing-masing server memberikan *metadata* dengan format JSON dimana *metadata* ini akan di encode ke dalam bentuk *array*. Artinya data ini akan disimpan

sementara dalam bentuk *array file* untuk dilakukan proses *filter* sebelum dimasukkan ke dalam *database*. Proses *filter* ini diperlukan untuk menyaring informasi data apa saja yang dibutuhkan dan dapat memaksimalkan kinerja sistem. pada saat proses *filter* terjadi pengelompokan dimana data yang jenisnya sama akan di kelompokan dan dimasukkan ke dalam *database*. Terakhir informasi yang telah dimasukkan ke dalam *database* ini ditampilkan sesuai dengan kategori masing-masing.

b. Desain *Keyword* dan Parameter Pada API

Saat menggunakan API dibutuhkan parameter tertentu untuk mendapatkan data, berikut desain *keyword* dan parameter pada API.



Gambar 4.5 Desain penggunaan *keyword* pada API

Keyword yang diinputkan akan menjadi parameter bagi ketiga API. Setiap API mempunyai algoritma sendiri untuk

mendapat data yang telah disediakan. Berikut penjelasan mengenai proses ketiga API saat *keyword* diinputkan :

1. Google API

Data dari Google API ada dua jenis, buku dan majalah. Tetapi Google API secara default memberikan data buku. Untuk mendapatkan data dari Google diperlukan parameter yang fungsinya membantu pencarian dengan jenis tertentu. Pada google terdapat tiga parameter yang diperlukan yaitu *keyword*, *filter* dan MaxResults. Parameter *filter* digunakan untuk mencari data buku yang sifatnya viewable atau bisa dibaca, karena ada dari beberapa buku dari Google yang sifatnya berbayar. Kemudian parameter maxResults digunakan untuk menampilkan banyaknya data yang akan diambil, apabila tidak menggunakan parameter ini maka Google hanya akan memberikan 10 data.

2. ScienceDirect API

ScienceDirect API digunakan dua kali dalam satu kali pencarian, karena terdapat dua jenis data yang akan diambil. API ini menggunakan dua parameter, yaitu *keyword* dan *content*. Parameter content berfungsi untuk membedakan jenis data yang akan diambil.

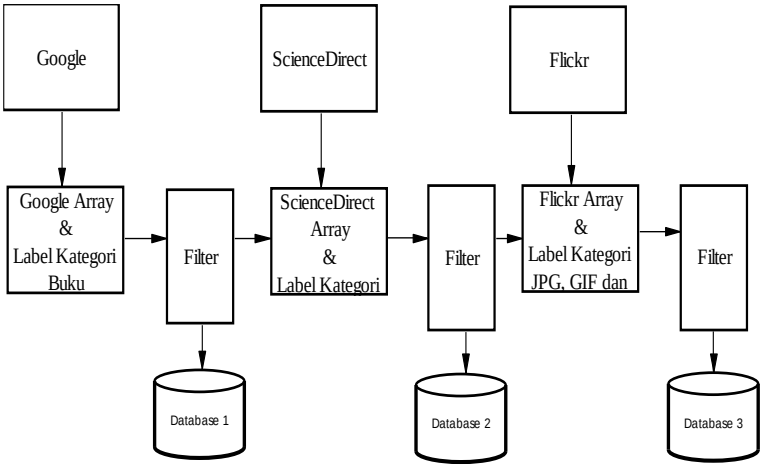
3. Flickr API

Pada Flickr API terdapat dua jenis data yang telah dibedakan oleh Flickr sendiri, yaitu gambar dan video. Namun Flickr API memberikan parameter tag yang membuat gambar tersebut bisa di bedakan berdasarkan tipe data nya, yaitu JPG dan GIF. Untuk data gambar bertipe JPG hanya menggunakan *keyword* untuk mendapatkan kategori, ini dikarenakan pada pencarian gambar hanya menghasilkan kategori dari gambar. Untuk data bertipe GIF dilakukan sebanyak dua kali pencarian. Pencarian pertama

menggunakan paramaeter *keyword* dan tag untuk mendapat ID dari gambar GIF tersebut. Pencarian kedua menggunakan ID yang diterima menjadi parameter untuk mendapatkan data gambar GIF. Selanjutnya pencarian data video juga dilakukan dua kali, dimana pencarian pertama adalah mencari ID dari video dengan parameter *keyword* dan media. Parameter media berfungsi untuk membedakan gambar dengan video. Lalu pada pencarian kedua digunakan ID yang diterima dijadikan parameter untuk mendapatkan data video.

c. Desain Akuisisi dan *Filter* Data

Setelah mencari dan mendapatkan data dengan menggunakan *keyword*, data yang didapat disimpan sementara didalam sistem dengan mengubah format data sebelumnya yaitu JSON menjadi *array*. Kemudian data *array* tersebut di *filter* kembali dan diambil informasi yang sesuai dengan kebutuhan. Lalu data yang telah di *filter* di masukkan ke *database* masing-masing. Berikut desain proses akuisisi dan *filter* data.



Gambar 4.6 Desain akuisisi dan *filter* data

Hasil akuisisi data disimpan ke dalam *array* dimana setiap data yang ada didalam *array* diberikan label untuk membedakan data yang masuk tersebut. Kemudian data *array* di *filter* agar yang dimasukkan kedalam *database* hanya informasi penting seperti link. Sebagai contoh jika memasukkan *keyword* "apple" maka akan diperoleh *metadata* seperti berikut:

1. *Metadata* dari Google

Metadata yang diterima sesuai dengan *keyword* yang dimasukkan, dan pencarian yang dilakukan dimulai dari judul. Ketika pada judul tidak ditemukan kata kunci "apple", pencarian dilakukan ke konten yang lain seperti deskripsi, penulis, dll. Konten judul adalah yang menjadi tujuan utama dalam pencarian. Berikut *metadata* yang diterima dari server


```
      "kind": "books#volume",
      "id": "R0UlAQAAIAAJ",
      "etag": "jiWOK0AVCJs",
      "selfLink":
"https://www.googleapis.com/books/v1/volumes/R0UlAQ
AAIAAJ",
      "volumeInfo": {
        "title": "One Green Apple",
        "authors": [
          "Eve Bunting"
        ],
        "publishedDate": "2006",
        "description": "While on a school field
trip to an orchard to make cider, a young immigrant
named Farah gains self-confidence when the green
apple she picks perfectly complements the other
students' red apples.",
        "industryIdentifiers": [
          {
            "type": "OTHER",
            "identifier": "STANFORD:36105128321820"
```


Gambar 4.7 Contoh *metadata* dari Google

2. *Metadata* dari SciencdeDirect

Pencarian *metadata* dimulai dengan mencari title, jika pada title terdapat kata kunci "apple" maka semua data tersebut akan diambil. Berikut contoh *metadata* yang diterima dengan menggunakan *keyword* "apple".


```
"dc:identifier":
"DOI:10.1016/j.foodchem.2016.10.029",
        "eid": "1-s2.0-S0308814616316466",
        "prism:url":
"http://api.elsevier.com/content/article/pii/S03088
14616316466",
        "dc:title": "Ursolic acid from apple pomace
and traditional plants: A valuable triterpenoid
with functional properties",
        "dc:creator": "Simone Tasca, Cargnin",
        "prism:publicationName": "Food Chemistry",
        "prism:issn": "03088146",
        "prism:volume": null,
        "prism:issueIdentifier": null,
        "prism:coverDate": [
          {
            "@_fa": "true",
            "$": "2017-04-01"
          }
        ],
        "prism:coverDisplayDate": "1 April 2017",
        "prism:startingPage": "477",
        "prism:endingPage": "489",
        "prism:doi":
"10.1016/j.foodchem.2016.10.029",
        "openaccess": "0",
        "openaccessArticle": false,
        "openArchiveArticle": false,
        "openaccessUserLicense": null,
        "pubType": "Review",
        "pii": "S0308-8146(16)31646-6",
        "authors": {
          "author": [
```


Gambar 4.8 Contoh *metadata* dari ScienceDirect

3. *Metadata* dari Flickr

Proses pengambilan *metadata* pada Flickr terbagi menjadi tiga tahapan untuk gambar(JPG) serta dua tahapan untuk gambar(GIF) dan video. Pada Gambar 4.9 dihasilkan konten dari gambar(JPG) dimana *metadata* yang diterima berupa konten-konten yang telah di cluster oleh Flickr. Konten pada *metadata* yang diterima digunakan untuk mencari gambar yang berkaitan dengan konten. Berikut *metadata* konten yang diterima dengan menggunakan kata kunci "apple".


```
"source": "apple",
"total": 4,
"cluster": [
  {
    "total": 5,
    "tag": [
      {
        "_content": "nyc"
      },
      {
        "_content": "newyork"
      },
      {
        "_content": "manhattan"
      },
      {
        "_content": "newyorkcity"
      },
      {
        "_content": "ny"
      }
    ]
  },
```


Gambar 4.9 Contoh *metadata* konten dari Flickr

```
{
  "photos": {
    "photo": [
      {
        "id": "4086348702",
        "secret": "843b46a99e",
        "server": "2527",
        "farm": 3,
        "owner": "22815110@N04",
        "username": "_me mude a
www.flickr.com/photos/xsb",
        "title": "_coraZón pixelado",
        "ispublic": 1,
        "isfriend": 0,
        "isfamily": 0
      },
```


Gambar 4.10 Contoh *metadata* ID dari Flickr

Sebelum mendapatkan *metadata* berupa url gambar, *metadata* dari gambar(JPG), gambar(GIF) dan video menghasilkan sebuah *id* yang berfungsi untuk mencari url gambar dengan berbagai ukuran. Contoh *metadata id* yang diterima dengan menggunakan kata kunci "apple" dapat dilihat pada Gambar 4.10.

Setelah itu ID digunakan untuk mendapatkan data url dengan berbagai ukuran serta link menuju gambar yang ada di *website* Flickr. Contoh *metadata* url dengan berbagai ukuran yang diterima dapat dilihat pada Gambar 4.11

```
        "label": "Square",
        "width": 75,
        "height": 75,
        "source":
"https://farm3.staticflickr.com/2527/4086348702_
843b46a99e_s.jpg",
        "url":
"https://www.flickr.com/photos/xsblack/408634870
2/sizes/sq/",
        "media": "photo"
      },
      {
        "label": "Large Square",
        "width": "150",
        "height": "150",
        "source":
"https://farm3.staticflickr.com/2527/4086348702_
843b46a99e_q.jpg",
        "url":
"https://www.flickr.com/photos/xsblack/408634870
2/sizes/q/",
        "media": "photo"
      },
```

Gambar 4.11 Contoh *metadata* url dari Flickr

Metadata yang telah diterima disimpan kedalam *array* untuk dilakukan proses *filter*. Dalam proses *filter*, *metadata* yang disimpan dalam *array* akan diambil satu per satu sesuai dengan kebutuhan. Kemudian data yang telah disaring akan dimasukkan ke dalam *database*. Pada saat akan memasukkan data ke *database* ini terjadi pengelompokan dari hasil *filter*, dimana setiap data akan diberi label yang membedakan data berdasarkan kategori yang telah dibuat. Algoritma pengkategorian data dan *filter* data untuk dimasukkan kedalam *database* terdapat pada Gambar 4.12.

Algoritma akuisisi dan *filter* data

1. Inisialisasi *metadata*
2. Ubah kedalam bentuk *array*
3. *Filter* data yang diperlukan seperti :
    a. Judul
    b. Penulis atau Penerbit
    c. Tahun Terbit
    d. *Link Website*
4. *Insert* data ke *database*

Gambar 4.12 Algoritma akuisisi dan *filter* data

Keterangan :

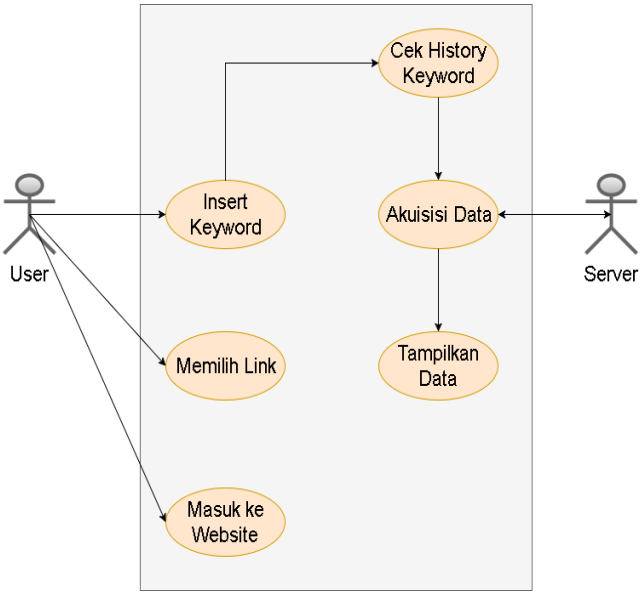
1. *Metadata* diinisialisasikan kedalam variabel dimana variabel tersebut diberikan sebuah label yang menandakan bahwa *metadata* tersebut adalah kategori tertentu
2. Variabel yang berisi *metadata* tersebut diubah kedalam bentuk *array* tujuannya untuk memudahkan proses *filter*
3. Proses penerimaan *metadata* menggunakan aturan antrian, artinya proses inisialisasi dan *filter* data dilakukan secara bergantian dimulai dari Google, lalu ScienceDirect dan Flickr
4. Data *array* yang telah disaring dan dikelompokan sesusai label dimasukkan kedalam *database*, dimana data tersebut berupa informasi yang diperlukan untuk ditampilkan pada hasil pencarian. Informasi tersebut berupa judul, penulis, tahun terbit dan link untuk buku dan jurnal, serta link untuk gambar(JPG), gambar(GIF) dan video.

4.2.3 Perancangan Alur Sistem

Perancangan alur sistem dibuat menggunakan UML (*Unified Modeling Language*) metodologi untuk membangun sistem OOP (*Oriented Object Programming*) yang diantaranya membuat *Use Case Diagram, Sequance Diagram, Activity Diagram* dan *Class Diagram*. Secara garis besar alur sistem di bagi dalam tiga tahapan, yaitu cek *history keyword*, pengambilan dan penyimpanan data, serta skema *database*. Berikut penjelasan mengenai metodologi untuk membangun sistem :

1. Perancangan *Use Case Diagram*

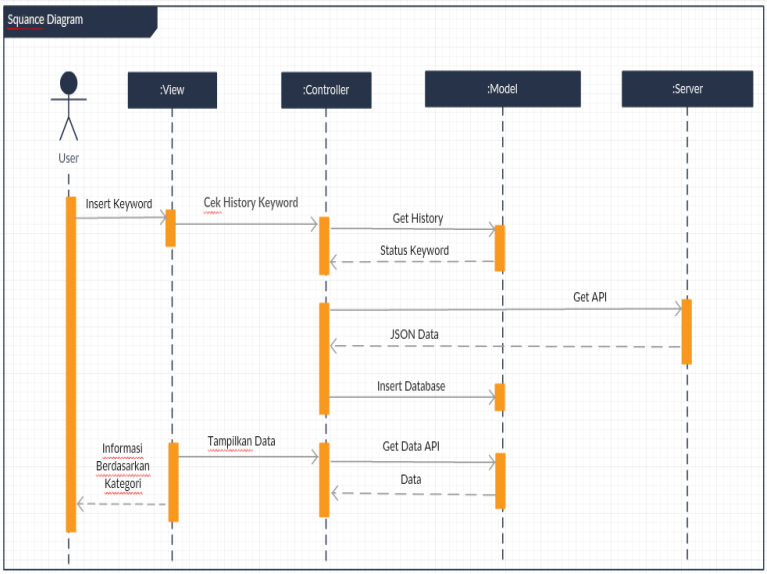
   Use case diagram digunakan untuk menjelaskan apa yang akan dilakukan sistem serta aktor yang berhubungan dengan sistem. Pada perancangan ini terdapat 2 aktor, yaitu *user* dan server. *Use case diagram* sistem dapat dilihat pada Gambar 4.13



Gambar 4.13 *Use case diagram sistem*

2. Perancangan *Sequance Diagram*

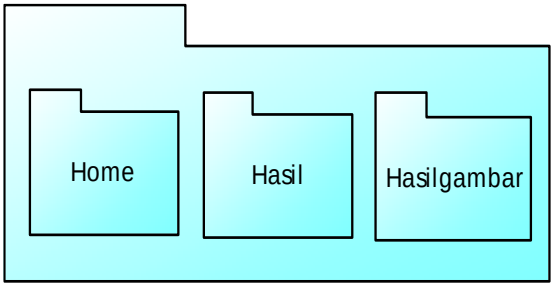
Sequence diagram sistem dapat dilihat pada Gambar 4.14. *Sequence diagram* menjelaskan proses pencarian informasi secara umum berdasarkan *keyword* atau kata kunci dengan menggunakan *framework* CodeIgniter melalui metode MVC. Proses dimulai dengan penginputan *keyword* dihalaman antarmuka. Kemudian sistem melakukan pengecekan terhadap *keyword* yang diteruskan ke *model* untuk proses pengecekan. Sistem memberikan respon untuk bahwa *keyword* tidak ada di dalam *database*. Selanjutnya sistem akan mengakses API untuk akuisisi data yang ada di server. Server memberikan respon berupa sejumlah data dengan format JSON yang akan dimasukkan ke *database*. Terakhir pada halaman antarmuka akan menampilkan data yang ada didalam *database*.



Gambar 4.14 *Squence Diagram* Alur Sistem

3. Perancangan *Class Diagram*

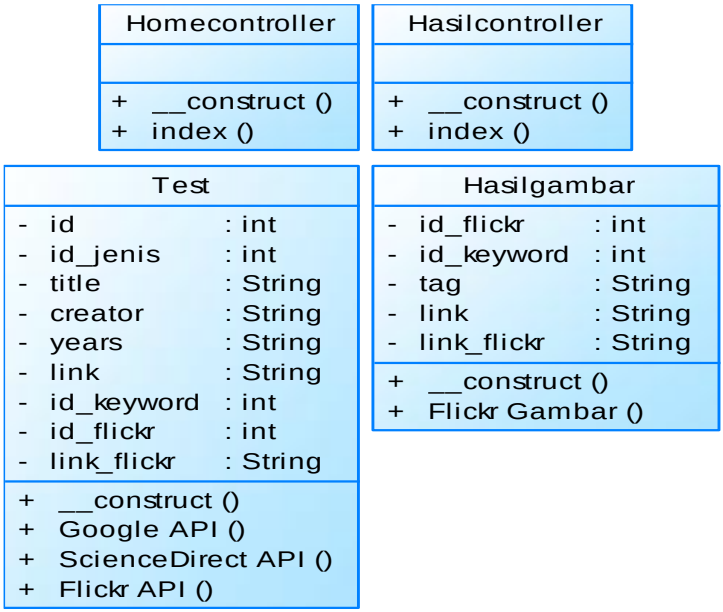
Class diagram ini digunakan untuk menggambarkan kumpulan dari class dan hubungannya. Class menggambarkan keadaan suatu sistem yang memiliki tiga area pokok, yaitu nama, atribut dan metode.



Gambar 4.15 *Class diagram view*

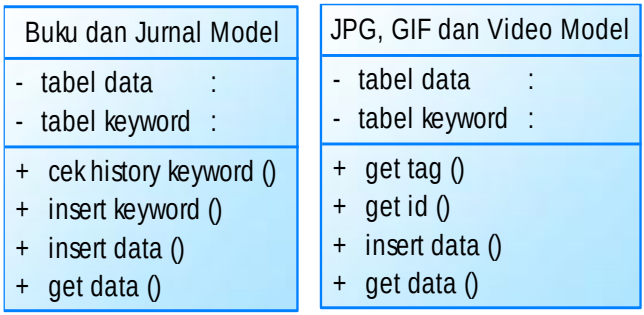
Pada Gambar 4.15 terdapat *class diagram view* dimana isi dari *class diagram* ini adalah tampilan antarmuka yang akan dilihat *user* diantaranya ada *home* adalah tampilan awal, hasil adalah tampilan hasil yang berupa informasi berdasarkan kategori, hasil gambar adalah tampilan dari hasil gambar yang dipilih. *Class diagram controller* terdapat proses yang dilakukan oleh sistem dan atribut apa saja yang digunakan dalam proses tersbut.

Pada Gambar 4.16 ada dua *controller* yang berperan saat melakukan pencarian data yaitu Test dan Hasilgambar. Untuk Test digunakan saat mencari semua data berdasarkan kategori sedangkan Hasilgambar digunakan untuk mencari gambar dengan kategori tertentu.



Gambar 4.16 *Class diagram controller*

Class diagram model adalah kelas yang berhubungan dengan data yang ada di dalam *database*. *Class diagram* dapat dilihat pada Gambar 4.17

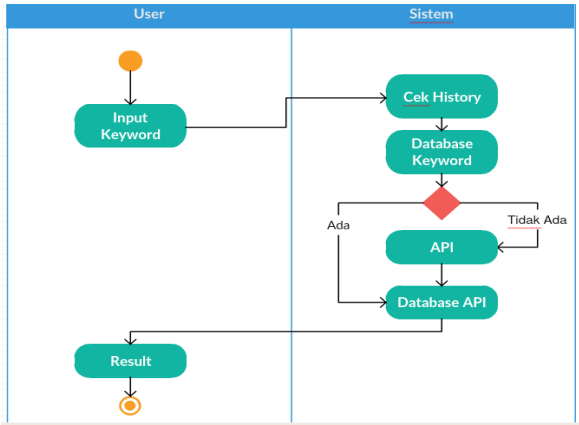


Gambar 4.17 *Class diagram model*

Pada Gambar 4.17 terdapat 2 *model* yang digunakan, dimana *model* yang pertama terdapat fungsi untuk mengecek *keyword* yang ada di *database*, kemudian fungsi untuk memasukkan *keyword* yang diinputkan, fungsi *insert* data ke dalam *database* dan fungsi mengambil data dari *database*. *Model* kedua digunakan saat mengambil data gambar yang pada *model* tersebut berisikan fungsi untuk mendapatkan kategori gambar dan juga data gambar, serta menampilkan gambar dari *database*.

a. Cek *History Keyword*

Selanjutnya sistem akan membuat sebuah aturan pengecekan *history* pencarian dengan *keyword* yang sama. Skenario untuk pengemeriksaan *keyword* pada *database* terdapat pada *activity diagram* yang dapat dilihat pada Gambar 4.18.



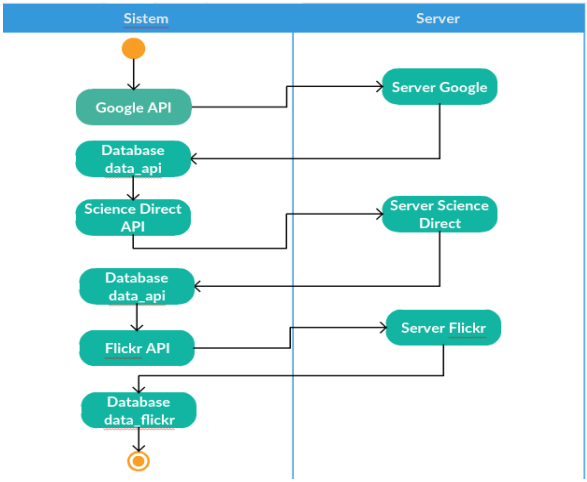
Gambar 4.18 *Activity diagram* cek *keyword*

Hal ini berguna untuk memaksimalkan kinerja sistem pada saat *user* menggunakan aplikasi pencarian ini. Saat *keyword* dimasukkan, sistem akan memeriksa terlebih dahulu

apakah *keyword* tersebut ada didalam *database*. Jika *keyword* tidak ada didalam *database keyword*, maka sistem akan mengambil data yang ada di server dengan menggunakan API. Jika *keyword* ada didalam *database keyword*, maka informasi dari *keyword* tersebut sudah ada didalam *database* sehingga sistem tidak perlu menggunakan API untuk mendapatkan informasi.

b. Akuisisi Data

Setelah itu, data dapat diambil sesuai dengan kategori yang telah ditentukan dengan menggunakan API dari masing-masing server. Data yang diambil dengan format JSON akan di *filter* kembali untuk diambil sesuai dengan kebutuhan. Kemudian data tersebut dimasukkan kedalam *database* masing masing. Data dari server Google dan ScienceDirect masuk ke dalam *database* data_api, sedangkan data dari server Flickr masuk ke dalam *database* data_flickr. Tahap ini digambarkan dengan *activity diagram* berikut.



Gambar 4.19 *Activity Diagram* Akuisisi Penyimpanan Data

4.2.4 Perancangan Alur Penggunaan API

API merupakan sebuah fungsi siap pakai yang disediakan oleh server. Syarat menggunakan API adalah memiliki API *key*. API *key* (*Application Programming Interface key*) adalah kode atau kunci yang digunakan untuk mengakses layanan yang ingin digunakan. Cara mendapatkan API *key* dari layanan yang ingin digunakan berbeda-beda dan juga penggunaan API dari setiap layanan berbeda. Penjelasan mengenai perancangan penggunaan masing-masing API adalah sebagai berikut :

a. ScienceDirect API

ScienceDirect API yang digunakan adalah ScienceDirect API *search* yang berfungsi untuk mencari *cluster* ScienceDirect yang berisi informasi berupa teks. Parameter yang diperlukan untuk mendapatkan data terdapat pada Tabel 4.2.

Tabel 4.2 Parameter (*required*)

No	Nama	Tipe	Pilihan	Keterangan
1	Accept	String	JSON, ATOM, XML	*Submit query* dengan parameter *string* "httpAccept" dan memberikan respon dalam bentuk *JSON, ATOM* atau *XML mark-up*
2	X-ELS-APIKey	String	-	Kode unik untuk para developer aplikasi mengakses ke *API resource* dengan *submit* menggunakan API key.
3	Query	String	-	Perintah eksekusi pencarian.

API *resource* ini memungkinkan penggunaan *Boolean query* ke dalam indeks ScienceDirect untuk mendapatkan hasil berupa *metadata* yang relevan dalam format teks.

Boolean query adalah *query* atau pernyataan yang menggunakan pemodelan sistem temu kembali. ScienceDirect API menggunakan metode GET *request* yang kegunaannya menyimpan data yang akan di diambil melalui URL. dan parameter tambahan pada Tabel 4.3

Tabel 4.3 Parameter tambahan

No	Nama	Tipe	Pilihan	Keterangan
1	Field	String	-	Mendapatkan informasi yang lebih spesifik seperti url, *identifier* dan *description*
2	Date	String	-	Rentang tanggal yang berkaitan dengan pencarian berdasarkan tahun, seperti Date=2007-2009
3	Count	String	-	Nilai maksimum dari pencarian data. Untuk default maksimal 25
4	Sort	String	coverDate, publicationName, relevancy, sort-order	Menyaring data yang diterima
5	Content	String	serial, non-serial, journals, allbooks	Kategori data yang akan dicari. Seperti buku dan jurnal
6	Facets	String	contenttype,	Navigator untuk

			srctitle, pubyear, topics	mencari informasi secara umum. Contohnya contenttype berguna untuk mengetahui ketegori.

Untuk menggunakan ScienceDirect API harus mendaftar sebagai member terlebih dahulu agar bisa mendapatkan API *key*. API *key* digunakan pada metode GET *request* dengan format JSON dan parameter tambahan yaitu *facets* untuk mengetahui kategori dari konten. Untuk pencarian data menggunakan *phrase* yang berguna untuk pencarian dengan menggunakan tanda baca atau karakter khusus. Setelah kategori diketahui gunakan metode GET *request* dengan parameter *content* untuk mendapatkan data dengan kategori buku dan jurnal. Data yang diterima selanjutnya dimasukkan kedalam *database* API.

b. Google API

API yang digunakan adalah Google Book API v1 yang fungsinya mencari informasi tentang buku. Pencarian isi buku dapat dilakukan dengan mengirimkan permintaan HTTP GET dan hanya memerlukan parameter q yang merupakan sebuah *keyword* untuk mencari informasi buku. Adapun parameter tambahan yang ditampilkan pada Tabel 4.4. Google Book API hanya menggunakan metode GET dengan parameter q dan semua informasi dari data dengan format JSON tersebut dimasukkan kedalam *database*.

Tabel 4.4 Parameter tambahan

No	Nama	Pilihan	Keterangan
1	Filter	Partial	Sedikit bagian dari teks buku bisa dilihat.
		Full	Semua bagian dari teks buku bisa dilihat.
		Free-ebooks	Hanya untuk ebooks gratis
		Paid-ebooks	Hanya untuk ebooks berbayar
		Ebooks	Menampilkan semua
2	MaxResults	-	Maksimum hasil pengambilan data.
3	Printtype	all	Semua data
		books	Hanya buku
		magazines	Hanya majalah
4	Sorting	relevance	Hasil relevan dengan *keyword*
		newest	Hasil terbaru dari publikasi

c. Flickr API

Proses penggunaan Flickr API hampir sama dengan ScienceDirect API. Diperlukan API *key* untuk bisa menggunakan Flickr API. Syarat untuk mendapatkan API *key* tersebut adalah mendaftar sebagai *user* dan masuk ke Flickr. Setelah itu API *key* tersebut digunakan ke dalam metode GET *request*. Untuk Flickr API terdapat 3 kategori data yang akan diambil, yaitu gambar(JPG), gambar(GIF) dan video. Untuk proses pengambilan gambar terdapat 3 tahapan sebagai berikut :

1. Kategori

Gambar yang dicari menggunakan *keyword* akan dibagi menjadi beberapa kategori. Pada tahap ini mengambil informasi mengenai kategori dengan menggunakan *keyword*.

2. ID Gambar

   Untuk mendapatkan gambar yang berkaitan dengan kategori dan *keyword* dibutuhkan sebuah ID dari gambar. Pada tahap ini mengambil ID gambar dengan menggunakan *keyword* dan kategori yang telah didapat.

3. Size Gambar

   Terakhir untuk mendapatkan gambar dengan berbagai ukuran digunakan *size* dari gambar yang isinya adalah ukuran dan url. Pada tahap ini mengambil url atau *link* dengan menggunakan ID yang telah didapat sebelumnya.

Untuk proses pengambilan GIF dan video menggunakan parameter yang berbeda, terdapat 2 tahapan sebagai berikut :

1. ID

   Untuk mendapatkan ID GIF yang diperlukan *keyword* dan parameter *tag* yang diisi dengan kata kunci GIF. Parameter *tag* berfungsi untuk membedakan setiap tag yang ada pada *metadata* sehingga bisa diketahui bahwa gambar tersebut termasuk kategori GIF. Kemudian untuk mendapatkan ID video diperlukan *keyword* dan parameter *media*, dimana *media* ini berisi kata kunci video yang nantinya akan membedakan antara gambar dan video

2. Size

   Selanjutnya untuk mendapatkan url dari video pada tahap ini menggunakan ID yang telah didapat sebelumnya, dan untuk mendapatkan ukuran dari GIF juga mengguanakan ID yang telah didapat sebelumnya.

Parameter untuk mendapatkan data JPG, GIF dan video terdapat pada Tabel 4.5, Tabel 4.6 dan Tabel 4.7

Tabel 4.5 Parameter JPG

No	Parameter	Nama	Pilihan	Keterangan
1	Kategori	api_key	-	Kode untuk menggunakan API
		tags	-	*Keyword*
		format		JSON
2	ID Gambar	api_key	-	Kode untuk menggunakan API
		tags	-	*Keyword*
		kategori_id	-	Nama dari kategori
		format		JSON
3	Size Gambar	api_key	-	Kode untuk menggunakan API
		gambar_id	-	Id gambar yang telah didapat
		format	JSON XML	Format metadata

Tabel 4.6 Parameter video

No	Parameter	Nama	Pilihan	Keterangan
1	ID Video	api_key	-	Kode untuk menggunakan API
		media	Photos, Videos	Jenis Video
		text	-	*keyword*
		per_page	-	Maksimum gambar yang dihasilkan
		format	JSON XML	Format metadata
2	Size Video	api_key	-	Kode untuk menggunakan API
		video_id	-	Id video yang telah didapat
		format	JSON XML	Format metadata

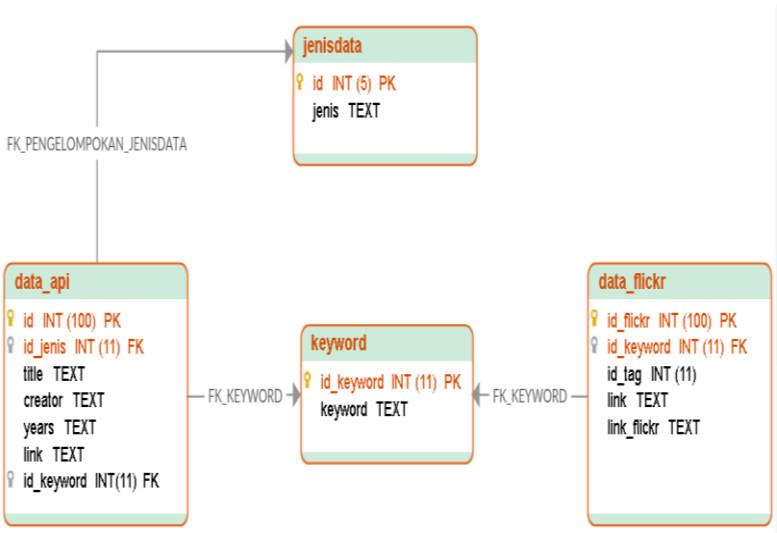
Tabel 4.7 Parameter GIF

No	Parameter	Nama	Pilihan	Keterangan
1	ID Gambar GIF	api_key	-	Kode untuk menggunakan API
		tags	-	GIF
		text	-	*keyword*
		per_page	-	Maksimum gambar yang dihasilkan
		format	JSON XML	Format metadata

2	Size Gambar GIF	api_key	-	Kode untuk menggunakan API
		gambar_id	-	Id gambar yang telah didapat
		format	JSON XML	Format metadata

4.2.5 Skema *Database*

Berdasarkan kebutuhan sistem aplikasi pencarian yang akan dibangun, maka rancangan skema *database* secara fisik dapat dilihat pada Gambar 4.20



Gambar 4.20 PDM dari *database*

Keterangan dari struktur masing-masing tabel akan dijelaskan sebagai berikut:

1. Tabel jenisdata

Untuk memudahkan saat menampilkan data pada setiap data yang masuk kedalam *database* diberi id dari tabel jenisdata.

Tabel 4.8 Struktur Tabel jenisdata

No	Atribut	Tipe Data	Keterangan
1	ID_JENIS	INT	*Primary Key*
2	JENIS_DATA	TEXT	Kategori Data

2. Tabel data_api

Data yang telah diambil dari server di masukkan kedalam tabel data_api dengan ditambahkan jenis data untuk membedakan data yang masuk. Isi dari tabel data_api adalah informasi-informasi yang akan ditampilkan. Data yang masuk ke dalam tabel data_api adalah buku dan jurnal. Kemudian tabel data_api memiliki kolom seperti ID_JENIS yang berfungsi membedakan data yang masuk, ID_KEYWORD berfungsi untuk mengetahui informasi yang masuk adalah hasil dari pencarian *keyword* tersebut, serta kolom JUDUL, PENULIS, TAHUN_TERBIT, LINK yang merupakan informasi dari data yang telah diambil.

Tabel 4.9 Struktur Tabel data_api

No	Atribut	Tipe Data	Keterangan
1	ID	INT	*Primary Key*
2	ID_JENIS	INT	Kategori Data
3	ID_KEYWORD	INT	*Keyword*
4	JUDUL	TEXT	Judul Buku/Jurnal
5	PENULIS	TEXT	Penulis/Penerbit
6	TAHUN_TERBIT	TEXT	Tahun
7	LINK	TEXT	URL

3. Tabel data_flickr

Pada tabel data_flickr tidak diberi ID_JENIS tetapi ID_TAG dimana TAG ini didapat saat akan mengambil data ke server. Fungsi dari TAG ini adalah untuk membedakan data JPG, GIF dan video serta membedakan data JPG setiap kategorinya. Kemudian terdapat kolom ID_KEYWORD yang juga digunakan untuk mengetahui *keyword* yang menghasilkan data tersebut, lalu LINK dan LINK_FLICKR. LINK berfungsi untuk menampilkan gambar(JPG), gambar(GIF) dan URL dari video, sedangkan LINK_FLICKR adalah url untuk menuju ke situs Flickr untuk melihat gambar dengan berbagai ukuran dan video.

Tabel 4.10 Struktur Tabel data_flickr

No	Atribut	Tipe Data	Keterangan
1	ID_FLICKR	INT	*Primary Key*
2	ID_KEYWORD	INT	*Keyword*
3	TAG	TEXT	Kategori
4	LINK	TEXT	URL Gambar/Video
5	LINK_FLICKR	TEXT	URL Situs

4. Tabel keyword

Kegunaan *keyword* adalah sebagai tanda dimana data tersebut merupakan data dari *keyword* yang telah di input *user* sehingga ketika *user* menggunakan *keyword* yang sama, data bisa langsung diambil di *database*.

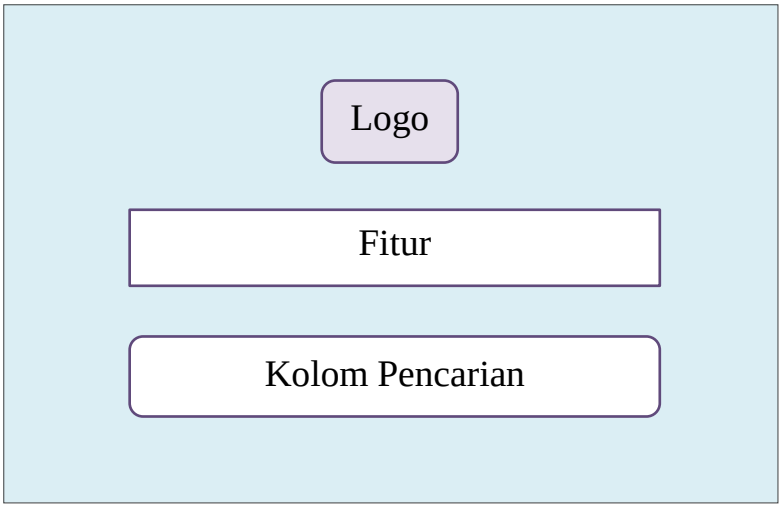
Tabel 4.11 Struktur Tabel keyword

No	Atribut	Tipe Data	Keterangan
1	ID_KEYWORD	INT	*Primary Key*
2	KEYWORD	TEXT	*Keyword*

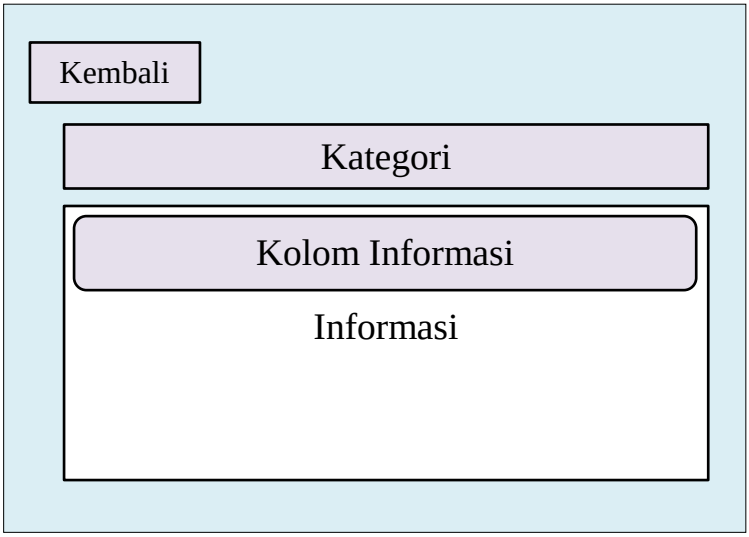
4.2.6 Perancangan Halaman Antarmuka

Halaman antarmuka aplikasi terbagi menjadi 2, yaitu halaman awal dan halaman hasil. Pada halaman awal yang merupakan tampilan pertama aplikasi berisi sebuah kolom pencarian dan fitur dimana kolom pencarian berfungsi untuk mencari data berdasarkan inputan *keyword* sedangkan fitur berisi tombol-tombol tambahan yang membuat aplikasi lebih menarik. Desain tampilan awal aplikasi terdapat pada Gambar 4.21

Selanjutnya halaman hasil merupakan halaman antarmuka yang menampilkan hasil dari pencarian yang dilakukan berdasarkan *keyword*. Pada halaman hasil terdapat tabel kategori dimana tabel-tabel tersebut berisi kolom informasi yang salah satunya adalah *link* yang berisi URI dari informasi yang dihasilkan. Lalu pada kiri atas tampilan terdapat tombol kembali ke halaman awal. Desain antarmuka halaman hasil dapat dilihat pada Gambar 4.22



Gambar 4.21 Halaman Awal



Gambar 4.22 Halaman Hasil

BAB V
IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dijelaskan mengenai tahapan implementasi dan pengujian aplikasi web pencarian atau *search engine*. Untuk implementasi dilakukan berdasarkan analisis dan perancangan sistem yang telah dibuat pada bab sebelumnya. Pada tahap implementasi dan pengujian yang akan dilakukan meliputi tahapan implementasi codeigniter untuk aplikasi web, implementasi *database*, implementasi API, implementasi antarmuka dan uji coba terhadap sistem aplikasi web pencarian.

5.1 Implementasi CodeIgniter

Codeigniter adalah sebuah *framework* aplikasi web yang bersifat *open source* yang berfungsi untuk membangun aplikasi dengan *script* PHP. Sebelum menggunakan Codeigniter, langkah awal yang dilakukan adalah melakukan konfigurasi dengan menggunakan fungsi *config*.

5.1.1 Konfigurasi A*utoload*

Autoload dalam codeigniter adalah konfirugasi untuk mengatur pemanggilan *class* apa saja yang akan panggil secara otomatis. Artinya dengan memasukkan nama *class* yang akan dipanggil, *class* tersebut tidak perlu dipanggil jika ingin digunakan dalam *controller*. Seperti Gambar 5.1. Secara otomatis akan memanggil *class database* dan *session*.

```
<?php
defined('BASEPATH') OR exit('No direct
script access allowed');
$autoload['libraries'] = array('database',
'session');
}
```

Gambar 5.1 Kode pada *autoload.php*

5.1.2 Konfigurasi *Config*

Konfigurasi ini berguna untuk mengarahkan ke alamat yang diakses. Setiap menggunakan base_url maka akan selalu mengarah ke http://localhost/SearchEngine

```
<?php
defined('BASEPATH') OR exit('No direct script
access allowed');
$config['base_url'] =
'http://localhost/SearchEngine';
}
```

Gambar 5.2 Kode pada *config.php*

5.1.3 Konfigurasi *Routes*

Konfigurasi ini berisi *controller* utama sebagai tampilan utama *website*. Setiap mengakses aplikasi, sistem memanggil *Homecontroller.php* untuk menampilkan halaman utama.

```
<?php
defined('BASEPATH') OR exit('No direct script
access allowed');
$route['default_controller'] =
'Homecontroller';
$route['404_override'] = '';
$route['translate_uri_dashes'] = FALSE;
}
```

Gambar 5.3 Kode pada *routes.php*

5.1.4 Konfigurasi *Session*

Konfigurasi ini bertujuan untuk membatasi waktu *update*. Pada aplikasi pencarian yang dibuat ini diatur untuk melakukan pencarian maksimal selama 60 detik atau 1 menit. Artinya, jika waktu pencarian melebihi 1 menit, maka pencarian akan dihentikan. Konfigurasi *session* dapat dilihat pada Gambar 5.4.

```
<?php
defined('BASEPATH') OR exit('No direct script
access allowed');
$config['sess_time_to_update'] = 60;
}
```

Gambar 5.4 Kode pada *config.php*

5.1.5 Konfigurasi *Database*

Konfigurasi ini digunakan untuk membuat koneksi ke *database*. *Database* yang digunakan adalah ta_db, dapat dilihat pada Gambar 5.5

```
<?php
defined('BASEPATH') OR exit('No direct script
access allowed');
$active_group = 'default';
$query_builder = TRUE;
$db['default'] = array(
      'dsn' => '',
      'hostname' => 'localhost',
      'username' => 'root',
      'password' => '',
      'database' => 'ta_db' );}
```

Gambar 5.5 Kode pada *database.php*

5.1.6 Konfigurasi *Error Reporting*

Konfigurasi ini berguna untuk mengatasi *error report* yang muncul setelah melakukan pencarian. Jika data yang dicari kosong maka akan terjadi *error* sebanyak perulangan dari data yang kosong dan akan merusak tampilan halaman antarmuka.

```
switch (ENVIRONMENT) {
      case 'development':
            error_reporting(-1);
            ini_set('display_errors', 0);
      break;
      case 'testing':
      case 'production':
            ini_set('display_errors', 0);
            if (version_compare(PHP_VERSION,
'5.3', '>=')) {
                  error_reporting(E_ALL &
~E_NOTICE & ~E_DEPRECATED & ~E_STRICT &
~E_USER_NOTICE & ~E_USER_DEPRECATED);
      }
            else        {
                  error_reporting(E_ALL &
~E_NOTICE & ~E_STRICT & ~E_USER_NOTICE);
      }
      break;
      default:
            header('HTTP/1.1 503 Service
Unavailable.', TRUE, 503);
            echo 'The application environment
is not set correctly.';
            exit(1); // EXIT_ERROR }}
```

Gambar 5.6 Kode pada *Index.php*

5.2 Implementasi *Database*

Implementasi *database* adalah proses pembuatan tabel-tabel yang diperlukan untuk menyimpan data yang akan diambil menggunakan API. Pembuatan tabel ini sesuai dengan perancangan *database* sebelumnya. Berikut ini *listing* program SQL yang digunakan untuk membuat tabel jenisdata, data_api, data_flickr dan keyword.

1. Tabel jenisdata

Tabel jenisdata terdiri dari kolom id dan jenis dapat dilihat pada Gambar 5.7. Kolom id sebagai *primary key* dan bertipe *integer* sedangkan kolom jenis merupakan dari jenis data yang bertipe *text*.

```
CREATE TABLE jenisdata(
    id            INT           NOT NULL,
    jenis         TEXT           NULL    ,
    PRIMARY KEY (id)
);
```

Gambar 5.7 *Listing* membuat tabel jenisdata

2. Tabel keyword

Tabel keyword terdiri dari id_keyword dan keyword dimana id_keyword merupakan *primary key* bertipe *integer* dan keyword adalah kata kunci bertipe *text.*

```
CREATE TABLE keyword(
    id_keyword    INT     NOT NULL        ,
    keyword       TEXT    NOT NULL        ,
    PRIMARY KEY (id_keyword)
);
```

Gambar 5.8 *Listing* membuat tabel keyword

3. Tabel data_api

Tabel data_api berfungsi untuk menyimpan data yang berasal dari Google dan ScienceDirect. Pada tabel ini tediri

dari id sebagai *primary key*, id_jenis dan id_keyword sebagai *foreign key*, serta title, creator, years dan link bertipe *text* dan boleh kosong apabila data tidak ada. Titel, creator, years dan link berfungsi untuk menyimpan informasi dari data yang dibedakan berdasarkan id_jenis. *Listing* untuk membuat tabel data_api dapat dilihat pada Gambar 5.9.

```
CREATE TABLE data_api(
    id           INT     NOT NULL        ,
    id_jenis     INT     NOT NULL        ,
    title        TEXT    NULL            ,
    creator      TEXT    NULL            ,
    years        TEXT    NULL            ,
    link         TEXT    NULL            ,
    id_keyword   INT     NOT NULL        ,
    PRIMARY KEY (id)
);
```

Gambar 5.9 *Listing* membuat tabel data_api

4. Tabel flickr

Tabel data_flickr ini berfungsi sebagai tempat penyimpanan data yang berasal dari flickr. Tabel data_flickr terdiri dari id_flickr sebagai *primary key*, id_keyword sebagai *foreign key*, id_tag berfungsi untuk membedakan data yang telah diambil, link dan link berisi informasi dari data yang bertipe *text*.

```
CREATE TABLE data_flickr(
    id_flickr    INT      NOT NULL        ,
    id_keyword   INT      NOT NULL        ,
    id_tag       TEXT     NOT NULL        ,
    link         TEXT     NULL            ,
    link_flickr  TEXT     NULL            ,
    PRIMARY KEY (id_flickr)
);
```

Gambar 5.10 *Listing* membuat tabel Flickr

Hasil dari pembuatan tabel dapat dilihat pada Gambar 5.11. terlihat pada gambar dihasilkan empat tabel yang akan digunakan untuk menyimpan data

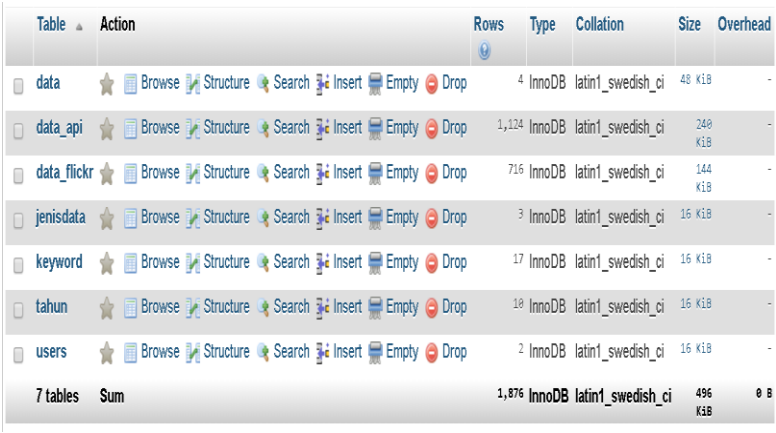


Table	Action	Rows	Type	Collation	Size	Overhead
data	Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	48 KiB	-
data_api	Browse Structure Search Insert Empty Drop	1,124	InnoDB	latin1_swedish_ci	240 KiB	-
data_flickr	Browse Structure Search Insert Empty Drop	716	InnoDB	latin1_swedish_ci	144 KiB	-
jenisdata	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
keyword	Browse Structure Search Insert Empty Drop	17	InnoDB	latin1_swedish_ci	16 KiB	-
tahun	Browse Structure Search Insert Empty Drop	10	InnoDB	latin1_swedish_ci	16 KiB	-
users	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	16 KiB	-
7 tables	Sum	1,876	InnoDB	latin1_swedish_ci	496 KiB	0 B

Gambar 5.11 Tabel di *database*

5.3 Implementasi Cek *History Keyword*

Cek *history* adalah kegiatan pertama yang dilakukan pada sistem aplikasi pencarian ini. Pada perancangan yang dilakukan sebelumnya dibuat sebuah algoritma untuk memeriksa setiap *keyword* yang diinputkan kedalam sistem. Dari algoritma yang telah dibuat diperoleh sebuah kode yang diterapkan dalam *script* PHP. Berikut potongan kode yang telah dibuat untuk memeriksa *keyword* yang diinputkan dengan *keyword* yang ada di *database*.

```
$parameter = $_POST["keyword"];
$dataHistory = $this->BookModel-
>gethistory($parameter);
$jumlahData = count($dataHistory);
if($jumlahData==0) {
$dataKeyword = array();
        $dataKeyword['keyword'] = $parameter;
        $idBaru = $this->BookModel-
>insertKeyword($dataKeyword);
        $idSekarang = $idBaru;
}
else{
        $idSekarang =
$dataHistory[0]['id_keyword'];
      }
```

Gambar 5.12 Potongan kode untuk pemeriksaan *keyword*

Pada *query* diatas menjelaskan proses pemeriksaan *keyword* dimulai dari menginisiasikan *keyword* menjadi variabel $parameter. Kemudian memanggil *class* model dengan fungsi gethistory yang fungsinya adalah memeriksa variabel $parameter ke *database keyword.* Fungsi gethistory dapat dilihat pada Gambar 5.13. Jika pemeriksaan pada *database keyword* kosong, maka dilakukan proses selanjutnya.

```
public function gethistory($keyword){
         $query = $this->db-
>get_where('keyword', array('keyword' =>
$keyword));
         return $query->result_array();
      }
```

Gambar 5.13 Kode pada *class model*

5.4 Implementasi API

Seperti pada proses cek *history keyword*, Implementasi API juga menggunakan algoritma yang telah dibuat dengan mengikuti aturan dari API masing-masing server. Implementasi API menggunakan fungsi json_decode dimana fungsi ini merupakan perintah untuk mengubah data dengan format JSON ke bentuk *array*. Fungsi json_decode ini disimpan didalam variabel $data. Kemudian dengan menggunakan fungsi file_get_content, API dapat diakses melalui *script* PHP. untuk kategori data yang akan diambil digunakan parameter *content*. Potongan kode untuk implementasi API dan hasil dari penggunaan API adalah sebagai berikut :

1. ScienceDirect API

   Penggunakan kode seperti pada Gambar 5.14 menghasilkan data dengan fomat JSON dan data yang diterima berasal dari *metadata* sebuah *file* yang ada di server ScienceDirect. Contoh data yang diterima dengan menggunakan ScienceDirect API terdapat pada Gambar 5.15.

```
<?php
public function find() {
            $parameter = $_POST["keyword"];
            $this-
      >Sciencedirect(1,$parameter,$idBaru);
}
public function
Sciencedirect($tipe,$parameter,$idKeyword) {
   if($tipe==1) {
    $data =
json_decode(file_get_contents("http://api.else
vier.com/content/search/scidir?httpAccept=appl
ication%2Fjson&apiKey=090ac729d3355c8f5ef3e4bd
a127b212&query=".rawurlencode($parameter)."&co
ntent=allbooks"));
 }  } }
```

Gambar 5.14 Potongan kode untuk ScienceDirect API


```
        "dc:identifier": "DOI:10.1016/j.jelechem.2017.01.015",
        "eid": "1-s2.0-S1572665717300152",
        "prism:url": "http://api.elsevier.com/content/article/pii/S1572665717300152",
        "dc:title": "Mapping the antioxidant activity of apple peels with soft probe
scanning electrochemical microscopy",
        "dc:creator": "Tzu-En, Lin",
        "prism:publicationName": "Journal of Electroanalytical Chemistry",
        "prism:issn": "15726657",
        "prism:volume": null,
        "prism:issueIdentifier": null,
        "prism:coverDate": [↔],
        "prism:coverDisplayDate": "1 February 2017",
        "prism:startingPage": "120",
        "prism:endingPage": "128",
```


Gambar 5.15 JSON data dari ScienceDirect

2. Google API

Untuk penggunaan Google API dan ScienceDirect API perbedaannya terdapat pada penentuan kategori. Google API secara *default* akan mengambil semua data dengan konten buku. Seperti pada perancangan yang telah dibuat, penggunaan Google API memiliki tiga parameter penting untuk memberikan hasil yang sesuai dengan keinginan dan contoh potongan kode untuk pengambilan data Google dapat dilihat pada Gambar 1.6. Contoh data yang diterima dari Google dengan menggunakan *keyword* "apple" dapat dilihat pada Gambar 5.17.

```
<?php
public function find()
      {
            $parameter = $_POST["keyword"];
        $data =
json_decode(file_get_contents("https://www.goo
gleapis.com/books/v1/volumes?q=".rawurlencode(
$parameter)."&filter=partial&maxResults=20"));
}
```

Gambar 5.16 Potongan kode untuk Google API


```
"volumeInfo": {
  "title": "Apple Cider Vinegar Miracle Health System",
  "authors": [
    "Patricia Bragg, N.D., Ph.D.",
    "Paul C Bragg, N.D., PH.D."
  ],
  "publisher": "Health Science Publications, Inc.",
  "publishedDate": "2003-04-01",
  "industryIdentifiers": [
    {
      "type": "ISBN_13",
```


Gambar 5.17 JSON data dari Google

3. Flickr API

Tahap penggunaan Flickr API berbeda dengan Google API dan SciecnDirecr API. Fungsi yang digunakan sama yaitu json_decode dan file_get_content. Namun untuk pengambilan data ada 2 tahapan proses untuk gambar (GIF) dan video (MP4) dan 3 tahapan proses untuk gambar (JPG). Untuk GIF dan MP4 tahap pertama mengambil id dari gambar yang ada diserver dan tahap kedua adalah mengambil URL dari server dengan menggunakan id gambar yang sudah didapat. Untuk tahap pengambilan video sama dengan pengambilan gambar(GIF). Pada kasus pengambilan gambar(JPG), data yang diambil pertama adalah kategori dari gambar. Kemudian dari kategori tersebut id gambar dapat diambil, dan selanjutnya mengambil URL gambar tersebut.

```
<?php
public function find() {
            $parameter = $_POST["keyword"];
            $this-
      >Flickr(1,$parameter,$idBaru);}
public function
Flickr($tipe,$parameter,$idKeyword){
if($tipe==1){
  $data =
json_decode(file_get_contents("https://api.fli
ckr.com/services/rest/?method=flickr.photos.se
arch&api_key=b4b1c5b6cb61c7d630abb4eeb9bc83a7&
tags=gif&text=".$parameter."&per_page=15&forma
t=json&nojsoncallback=1"));
  foreach ($data->photos->photo as $nowdata) {
              $dataId =
json_decode(file_get_contents("https://api.fli
```

Gambar 5.18 Potongan kode untuk Flickr API

Contoh data yang diterima dengan menggunakan Flickr API dapat dilihat pada Gambar 5.19, 5.20 dan Gambar 5.21


```
"source": "apple",
"total": 4,
"cluster": [
  {...},
  {
    "total": 22,
    "tag": [
      {
        "_content": "fruit"
      },
      {
        "_content": "red"
      },
      {
        "_content": "green"
      },
      {
        "_content": "food"
```


Gambar 5.19 JSON data berupa konten gambar dari Flickr


```
    "id": "4086348702",
    "secret": "843b46a99e",
    "server": "2527",
    "farm": 3,
    "owner": "22815110@N04",
    "username": "_me mude a www.flickr.com/photos/xsb",
    "title": "_coraZón pixelado",
    "ispublic": 1,
    "isfriend": 0,
    "isfamily": 0
  },
  {
    "id": "4080777472",
    "secret": "183bd22e20",
    "server": "2801",
    "farm": 3,
    "owner": "81333898@N00",
```


Gambar 5.20 JSON data berupa id gambar dari Flickr

```
    "label": "Square",
    "width": 75,
    "height": 75,
    "source": "https://farm3.staticflickr.com/2527/4086348702_843b46a99e_s.jpg",
    "url": "https://www.flickr.com/photos/xsblack/4086348702/sizes/sq/",
    "media": "photo"
  },
  {
    "label": "Large Square",
    "width": "150",
    "height": "150",
    "source": "https://farm3.staticflickr.com/2527/4086348702_843b46a99e_q.jpg",
    "url": "https://www.flickr.com/photos/xsblack/4086348702/sizes/q/",
    "media": "photo"
  },
  {
```


Gambar 5.21 JSON data berupa *size* dan URL gambar dari Flickr

5.5 Implementasi Penyimpanan Data

Penyimpanan data dimulai dengan menginisiasikan variabel $data sebagai data yang telah diambil. Semua data tersebut di *filter* yang disimpan kedalam bentuk *array* dengan hanya mengambil *title, publisher, publishDate* dan *webReaderLink* dari data tersebut. Kemudian data yang diterima disimpan kedalam *database* dengan memanggil fungsi insert_data_api($data). insert_data_api($data) ini merupakan *query* untuk menyimpan data ke *database* data_api. Contoh *query insert* dapat dilihat pada Gambar 5.23. Selanjutnya data akan tersimpan ke dalam *database* dengan perulangan sebanyak 20 kali. Contoh kode untuk menyimpan data menggunakan API ke dalam *database* ada pada Gambar 5.22.

```
public function find()
      {
            $parameter = $_POST["keyword"];
$data =
json_decode(file_get_contents("https://www.goo
gleapis.com/books/v1/volumes?q=".rawurlencode(
$parameter)."&filter=partial&maxResults=20"));
foreach ($data->items as $now) {
          $data = array();
          if($now->volumeInfo->title){
            $data['title'] = $now->volumeInfo-
>title}
          else{
            $data['title'] = "";          }
             $data['id_jenis'] = 1;
             if($this->BookModel-
>insert_data_api($data)){             }
}
         $idSekarang = $idBaru; }
      else{
        $idSekarang =
$dataHistory[0]['id_keyword'];}
```

Gambar 5.22 Potongan kode untuk proses akuisisi data

```
public function insert_data_api($data) {
         if ($this->db->insert("data_api",
$data)) {
            return true;
         }
      }
```

Gambar 5.23 Kode untuk memasukkan data ke *database*

Contoh hasil dari akuisisi data yang dimasukkan kedalam *database* dapat dilihat pada Gambar 5.24

+ Options

	id	id_jenis	title	creator	years	link	id_keyword
Edit Copy Delete	1	1	Acknowledgments	John R., Apel	1988	http://www.sciencedirect.com/science/article/pii/S...	1
Edit Copy Delete	2	1	Edited by	NULL	1988	http://www.sciencedirect.com/science/article/pii/S...	1
Edit Copy Delete	3	1	Copyright page	NULL	1988	http://www.sciencedirect.com/science/article/pii/S...	1
Edit Copy Delete	4	1	Discourse Ethics	W., Outhwaite	2012	http://www.sciencedirect.com/science/article/pii/B...	1
Edit Copy Delete	5	1	Dedication	NULL	1988	http://www.sciencedirect.com/science/article/pii/S...	1
Edit Copy Delete	6	1	Preface	J.R., A.	1988	http://www.sciencedirect.com/science/article/pii/S...	1
Edit Copy Delete	7	1	Publisher's Credits	NULL	1988	http://www.sciencedirect.com/science/article/pii/S...	1
Edit Copy Delete	8	1	Appendix One Fundamental Physical Constants	NULL	1988	http://www.sciencedirect.com/science/article/pii/S...	1
Edit Copy Delete	9	1	Appendix Two Astronomical and	NULL	1988	http://www.sciencedirect.com/science/article/pii/S...	1

Gambar 5.24 Hasil dari akuisisi data

5.6 Implementasi Antarmuka

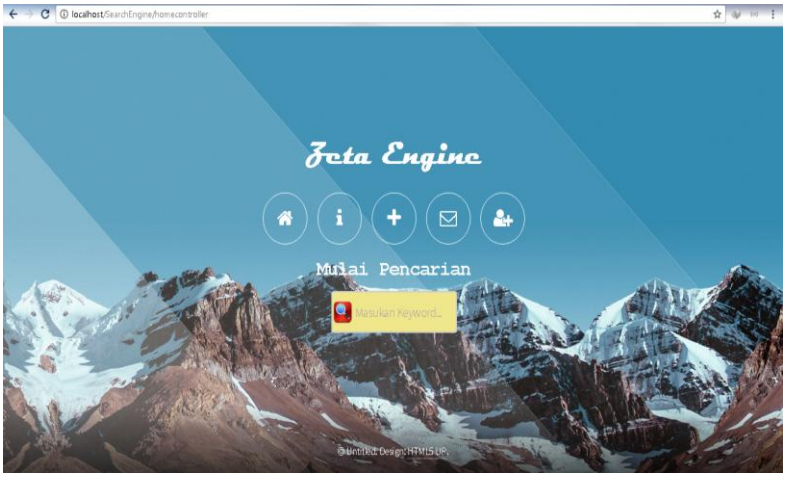
Halaman antarmuka pada aplikasi pencarian yang dibuat menggunakan fungsi *view* codeigniter. Pada dasarnya yang berperan pada untuk menampilkan fungsi *view* adalah *controller*, dimana setiap akan membuka halaman awal akan langsung terhubung dengan *controller* yang diteruskan ke fungsi *view*. Contoh kode untuk *controller* bisa dilihat pada Gambar 5.25. Terlihat pada Gambar *controller* langsung terhubung ke *home.php*. Halaman antarmuka aplikasi terbagi menjadi 2, yaitu halaman awal dan halaman hasil.

```
<?php
class Homecontroller extends CI_Controller{
      function  __construct() {
        parent::__construct();
        $this->load->helper('url');     }
     public function index()  {
            $data['title'] ='Home Search';
            $this->load->view( 'home',$data );
```

Gambar 5.25 Kode untuk menampilkan halaman awal pada *Homecontroller.php*

5.6.1 Halaman Awal

Pada halaman awal antarmuka berisi kolom pencarian yang berfungsi untuk mencari data berdasarkan inputan *keyword*. Pada halaman ini juga diberikan fitur berupa tombol yaitu, *home*, informasi, tambah data secara manual, kontak dan registrasi. Tampilan awal dari aplikasi ini akan tampak seperti pada Gambar 5.26.



Gambar 5.26 Tampilan Halaman Awal

5.6.2 Halaman Hasil

Pada halaman hasil menampilkan semua informasi berdasarkan kategori yang sesuai dengan *keyword*. Informasi tersebut dimuat kedalam tabel berdasarkan kategori. Setiap tabel diberikan *scroll* yang fungsinya membantu *user* mencari informasi dan juga membuat halaman tampilan tidak panjang kebawah, ini dilakukan karena data yang akan ditampilkan banyak. Pada bagian kiri atas terdapat tombol untuk kembali ke halaman awal atau halaman pecarian. Tampilan halaman hasil dari aplikasi ini akan tampak seperti pada Gambar 5.27.

Gambar 5.27 Tampilan Hasil

5.7 Pengujian Sistem

Uji coba terhadap aplikasi pencarian ini dilakukan dengan menganalisis tingkat kecocokan data berdasarkan kategori, keakuratan dari hasil pencarian, kecepatan pencarian dan membandingkan hasil pencarian pada aplikasi dalam penelitian dan mesin pencari google. Adapun batasan dalam melakukan uji coba pada aplikasi pencarian adalah sebagai berikut :

1. Menggunakan WIFI dengan kecepatan 65.0 Mbps yang stabil atau tidak adanya gangguan saat melakukan pencarian informasi di internet.
2. Maksimal data yang diperoleh dari kategori buku adalah 45 data
3. Maksimal data yang diperoleh dari kategori jurnal adalah 25 data
4. Maksimal data yang diperoleh dari kategori gambar(JPG) adalah 4 kategori, dan tiap kategori tidak dibatasi gambar yang akan diambil

5. Maksimal data yang diperoleh dari kategori gambar(GIF) adalah 15 data
6. Maksimal data yang diperoleh dari kategori video adalah 10 data
7. Waktu pencarian maksimal adalah 1 menit, apabila dalam 1 menit data belum ditemukan maka pencarian dihentikan.

5.7.1 Pengujian Kategori

Skenario yang dilakukan adalah dengan melakukan percobaan memasukkan *keyword* pada halaman antarmuka pencarian sebanyak 5 keyword dan melihat kecocokan antara informasi yang diterima dengan kategori. Misalnya pada kategori buku menghasilkan 10 data dan 10 data itu adalah informasi tentang buku. Kemudian dibuat persentase dari hasil pengujian dengan menggunakan persamaan sebagai berikut.

$$\text{Persentase Kategori} = \left(\frac{x}{n}\right) 100\% \quad \text{.....1)}$$

$$\text{Rata-rata Persentase Kategori} = \left(\frac{\sum_{j=1}^{j} p_i}{j}\right) 100\% \quad \text{.....2)}$$

$$\text{Rata-rata Persentase Keseluruhan} = \left(\frac{\sum_{k=1}^{k} q_i}{k}\right) 100\% \text{.....3)}$$

Dengan penjelasan,
$x =$ Data yang sesuai
$n =$ Jumlah data
$p_i =$ Persentase *keyword* ke-i
$j =$ Jumlah *keyword*
$q_i =$ Persentase kategori ke-i
$k =$ Jumlah kategori

Berikut hasil uji coba dengan menggunakan 5 *keyword* dengan menggunakan aplikasi pencarian yang ada pada Tabel 5.1

dan jumlah data yang diterima saat melakukan uji coba dapat dilihat pada Tabel 5.2

Tabel 5.1 Kecocokan informasi dengan kategori

No	Kategori	Keyword					Total
		Apple	Car	Apel	Kereta Api	Health Care?	
1	Buku	100%	100%	100%	100%	100%	100%
2	Jurnal	100%	100%	100%	100%	100%	100%
3	JPG	100%	100%	100%	100%	100%	100%
4	GIF	0%	87%	-	-	-	44%
5	MP4	100%	100%	100%	-	-	100%
Rata-rata							89%

Tabel 5.2 Jumlah data yang diterima

No	Kategori	Keyword					Total
		Apple	Car	Apel	Kereta Api	Health Care?	
1	Buku	45	45	45	20	45	200
2	Jurnal	25	25	25	11	25	111
3	JPG	89	80	41	21	62	293
4	GIF	15	15	-	-	-	30
5	MP4	10	9	4	-	-	23
Total Keseluruhan							657

Dari hasil Tabel 5.1 Dapat diberikan analisis bahwa dengan menggunakan *keyword* seperti apapun untuk kategori buku, jurnal, gambar(JPG) dan video akan menghasilkan data yang sesuai dengan kategori, namun tidak untuk kategori GIF. Kategori buku menghasilkan 100% data yang sesuai dengan kategori dari 200 data yang diperoleh, jurnal menghasilkan 100% data yang sesuai dengan kategori dari 111 data yang diperoleh, gambar(JPG) menghasilkan 100% data yang sesuai dengan

kategori dari 293 data yang diperoleh dan video menghasilkan 100% data yang sesuai dengan kategori dari 23 data yang diperoleh. GIF hanya menghasilkan 44% yang sesuai dengan kategori dari 30 data yang dihasilkan. Dari Tabel 5.1 dapat disimpulkan bahwa setiap *keyword* yang dimasukkan menghasilkan rata-rata tingkat kecocokan 89%.

5.7.2 Pengujian Kesesuaian *Keyword*

Skenario yang dilakukan adalah dengan melakukan percobaan memasukkan *keyword* atau kata kunci pada halaman antarmuka pencarian sebanyak 5 *keyword* dengan menggunakan 2 jenis *keyword* yaitu bahasa Indonesia dan bahasa Inggris serta melihat kecocokan antara informasi yang diterima dengan *keyword* yang digunakan. Pada setiap kategori dan semua data yang telah diterima akan dilihat isi dari data tersebut sehingga bisa diketahui apakah data yang diterima sesuai dengan *keyword*. Kemudian dibuat persentase untuk hasil yang sesuai dengan menggunakan persamaan berikut.

$$\text{Persentase Keakuratan} = \left(\frac{x}{n}\right) 100\% \quad \text{.....1)}$$

$$\text{Rata-rata Persentase } \textit{Keyword} = \left(\frac{\sum_{j=1}^{j} p_i}{j}\right) 100\% \quad \text{.....2)}$$

$$\text{Rata-rata Persentase Keseluruhan} = \left(\frac{\sum_{k=1}^{k} q_i}{k}\right) 100\% \text{.....3)}$$

Dengan penjelasan,
$x =$ Data yang sesuai perkategori
$n =$ Jumlah data yang diterima perkategori
$p_i =$ Persentase kategori ke-i
$j =$ Jumlah kategori
$q_i =$ Persentase *keyword* ke-i
$k =$ Jumlah *keyword*

Berikut hasil uji coba dengan menggunakan 5 *keyword* bahasa Indonesia dan 5 *keyword* bahasa Inggris dapat dilihat pada Tabel 5.3. Untuk jumlah pencarian dari setiap *keyword* dapat dilihat pada Tabel 5.4

Tabel 5.3 Kesesuaian informasi dengan *keyword* berdasarkan jenis *keyword*

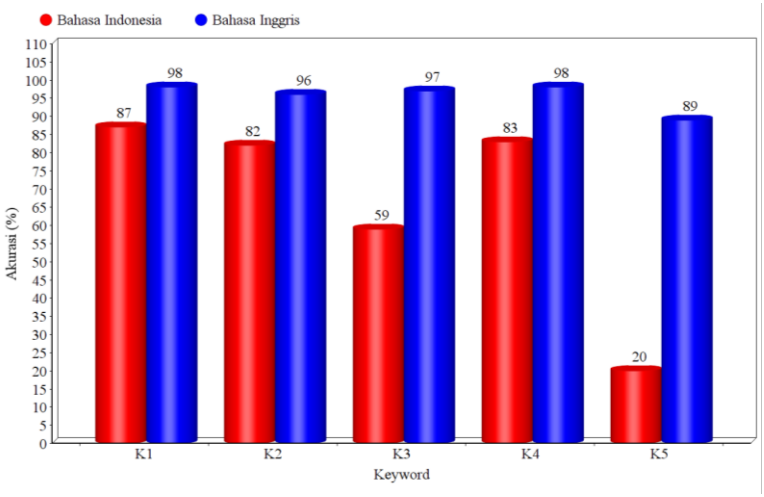
No	Jenis Keyword	Keyword	Persentase Keakuratan					
			Buku	Jurnal	JPG	GIF	MP4	**Total**
1	Bahasa Indonesia	apel	93%	96%	60%	-	100%	87%
		mobil	100%	88%	89%	-	50%	82%
		rumah	61%	60%	75%	100%	100%	59%
		oksigen	100%	50%	-	-	100%	83%
		kumis	47%	32%	-	0%	0%	20%
Total Keseluruhan								**66%**
2	Bahasa Inggris	apple	100%	100%	98%	93%	100%	98%
		car	100%	100%	99%	93%	89%	96%
		house	100%	100%	97%	87%	100%	97%
		oxygen	96%	100%	93%	100%	100%	98%
		mustache	60%	100%	100%	87%	100%	89%
Total Keseluruhan								**96%**

Dari hasil Tabel 5.3 dapat diberikan analisis bahwa *keyword* atau kata kunci dapat mempengaruhi kesesuaian data dengan kata kunci. Ada 2 jenis kata kunci yang digunakan, yaitu Bahasa Inggris dan Bahasa Indonesia. Pada Tabel 5.3 terlihat penggunaan Bahasa berpengaruh dengan keakuratan dengan kata kunci yang dimasukkan. Bahasa Inggris memiliki keakuratan rata-rata 96% dari 877 data dalam 5 kategori yang diuji, sedangkan Bahasa Indonesia memiliki keakuratan rata-rata 66% dari 465

data dalam 5 kategori yang diuji. Untuk jumlah data dapat dilihat pada Tabel 5.4. Dapat disimpulkan bahwa penggunaan Bahasa berpengaruh terhadap kesesuaian informasi dengan *keyword* yang digunakan. Berikut grafik yang menunjukkan perbedaan akurasi dari pencarian berdasarkan jenis *keyword* dapat dilihat pada Gambar 5.28.

Tabel 5.4 Jumlah data yang diterima

No	jenis keyword	keyword	Jumlah Item					Total
			Buku	Jurnal	JPG	GIF	MP4	
1	Bahasa Indonesia	apel	45	25	41	-	4	115
		mobil	45	25	69	-	6	145
		rumah	33	25	28	1	7	94
		oksigen	20	12	-	-	2	34
		kumis	38	25	-	7	7	77
Total Keseluruhan								465
2	Bahasa Inggris	apple	45	25	89	15	10	184
		car	45	25	80	15	9	174
		house	45	25	85	15	8	178
		oxygen	45	25	84	1	6	161
		mustache	45	25	87	15	8	180
Total Keseluruhan								877



Gambar 5.28 Grafik perbandingan akurasi pencarian berdasarkan jenis *keyword*

5.7.3 Pengujian Kecepatan

Skenario pengujian kecepatan adalah membandingkan kecepatan pencarian dengan menggunakan Bahasa yang berbeda, yaitu Bahasa Indonesia dan Bahasa Inggris. Pengujian dilakukan 5 *keyword* yang berbeda dengan masing-masing Bahasa. Kemudian dibuat persentase dari hasil pengujian dengan menggunakan persamaan sebagai berikut.

$$\text{Rata-rata Kecepatan} = \frac{\sum_{n=1}^{n} x_i}{n}$$

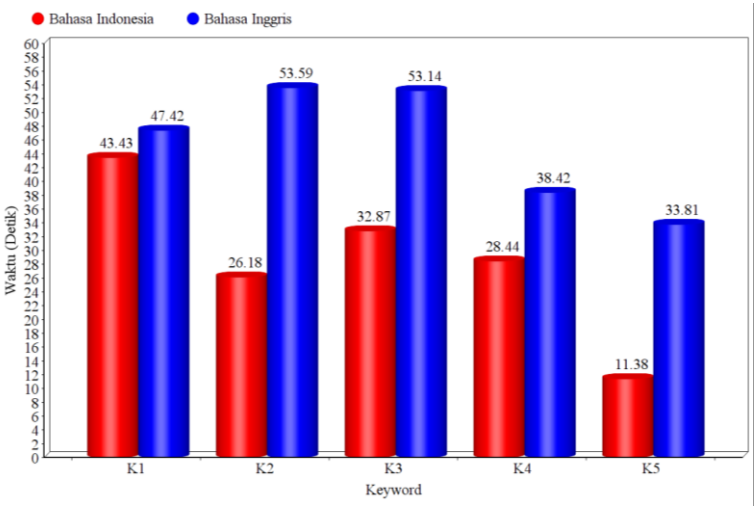
Dengan penjelasan,
x_i = Kecepatan waktu ke-i
n = Jumlah data kecepatan waktu

Berikut hasil uji coba kecepatan pencarian berdasarkan *keyword* dengan Bahasa Indonesia dan Bahasa Inggris pada Tabel 5.5.

Tabel 5.5 Perbandingan kecepatan waktu pencarian berdasarkan jenis *keyword*

No.	Jenis Keyword	Keyword	Jumlah Data Per Kategori						Waktu (detik)
			Buku	Jurnal	JPG	GIF	MP4	Total Item	
1	Bahasa Indonesia	Mobil	45	25	4	-	6	80	43.43
		Apel	45	24	3	-	4	76	26.18
		Kumis	38	25	-	7	7	77	32.87
		Rumah	33	25	2	1	7	68	28.44
		Oksigen	20	12	-	-	2	34	11.38
Rata-rata Waktu									**28.46**
2	Bahasa Inggris	Car	45	25	4	15	9	98	47.42
		Apple	45	25	4	15	10	99	53.59
		Mustache	45	25	4	15	8	97	53.14
		House	45	25	4	15	8	97	38.42
		Oxygen	45	25	4	1	6	81	33.81
Rata-rata Waktu									**45.276**

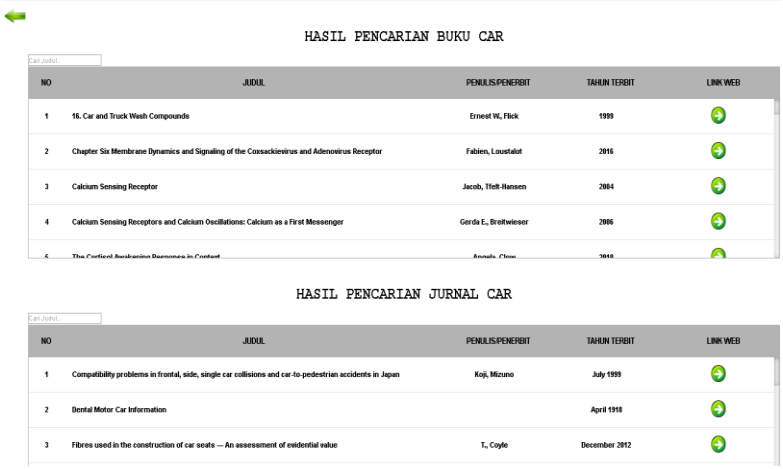
Hasil dari uji coba yang ada pada Tabel 5.3 menjelaskan bahwa penggunaan Bahasa pada *keyword* atau kata kunci yang digunakan berpengaruh pada kecepatan aplikasi dalam pencarian. Waktu yang dihasilkan dengan menggunakan Bahasa Indonesia adalah rata-rata 28.46 detik, sedangkan Bahasa Inggris adalah rata-rata 45.276 detik. Pada Gambar 5.29 terlihat dengan menggunakan 5 *keyword*, semua pencarian dengan menggunakan Bahasa Indonesia lebih cepat dibandingkan menggunakan Bahasa Inggris.



Gambar 5.29 Grafik perbandingan kecepatan waktu pencarian berdasarkan *keyword*

5.7.4 Perbandingan Aplikasi

Selanjutnya menguji perbandingan pencarian aplikasi *search engine* pada penelitian dengan mesin pencari Google. Uji coba ini membandingkan hasil pencarian pada aplikasi yang telah dibuat dengan hasil dari mesin pencari Google dengan memasukkan kata "*apple*" kedalam pencarian aplikasi dan menghasilkan informasi seperti berikut.



HASIL PENCARIAN BUKU CAR

NO	JUDUL	PENULIS:PENERBIT	TAHUN TERBIT	LINK WEB
1	16. Car and Truck Wash Compounds	Ernest W, Flick	1999	
2	Chapter Six Membrane Dynamics and Signaling of the Coxsackievirus and Adenovirus Receptor	Fabien, Loustalot	2016	
3	Calcium Sensing Receptor	Jacob, Tfelt-Hansen	2004	
4	Calcium Sensing Receptors and Calcium Oscillations: Calcium as a First Messenger	Gerda E., Breitwieser	2006	

HASIL PENCARIAN JURNAL CAR

NO	JUDUL	PENULIS:PENERBIT	TAHUN TERBIT	LINK WEB
1	Compatibility problems in frontal, side, single car collisions and car-to-pedestrian accidents in Japan	Koji, Mizuno	July 1999	
2	Dental Motor Car Information		April 1918	
3	Fibres used in the construction of car seats -- An assessment of evidential value	T., Coyle	December 2012	

Gambar 5.30 Hasil pencarian pada aplikasi

HASIL PENCARIAN GAMBAR (JPG) APPLE

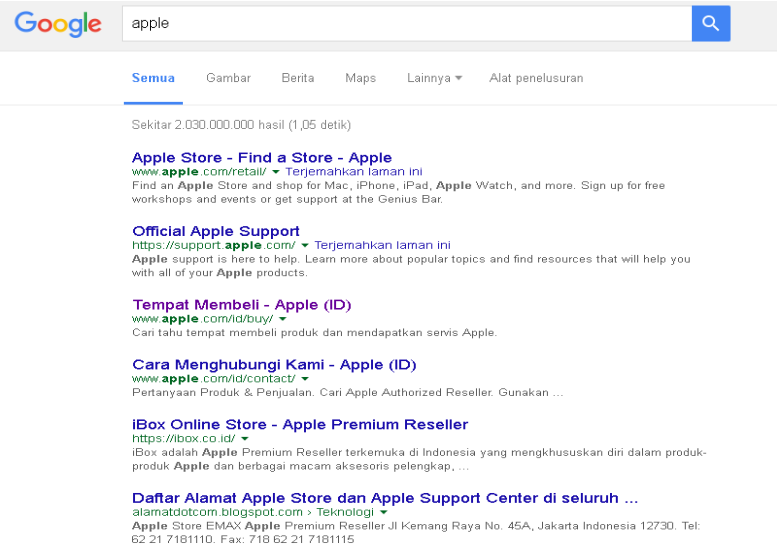
NO	KATEGORI	RELEVANSI	LIHAT GAMBAR
1	nyc	nyc | newyork | manhattan | newyorkcity | ny |	
2	fruit	fruit | red | green | food | tree | macro | canon | orange | blossom | apples | nature | flower | eos | yellow | banana | stilllife | closeup | dessert | spring | pink | color | pear |	
3	ipod	ipod | iphone | music | nano | touch | shuffle | mp3 | black | phone | ipodtouch | mobile |	
4	mac	mac | macbook | macintosh | computer | laptop | imac | keyboard | powerbook | osx | macbookpro | desktop | desk | pro | white | leopard | ibook | g4 | mouse | wallpaper | office | notebook | screenshot | nikon | work | os | light | screen | blue | g5 | tiw | itunes | book | logo | design | photobooth | pc |	

HASIL PENCARIAN GAMBAR (GIF) APPLE

NO	GAMBAR	LIHAT GAMBAR
1		
2		

Gambar 5.31 Hasil pencarian pada aplikasi

Sedangkan untuk hasil pencarian pada mesin pencari Google menghasilkan informasi seperti pada gambar berikut.



Gambar 5.32 Hasil pencarian pada mesin pencari

Gambar 5.33 Hasil pencarian pada mesin pencari

Dari hasil perbandingan antara aplikasi yang dibuat dengan mesin pencari terlihat bahwa mesin pencari Google menampilkan informasi yang umum pada halaman web. Berbeda dengan aplikasi pada penelitian, hasil yang ditampilkan berupa informasi yang lebih spesifik yaitu buku, jurnal, gambar(JPG), gambar(GIF) dan video. Pada mesin pencari Google juga memberikan kategori yang sama seperti aplikasi pada penelitian. Gambar yang dihasilkan mesin pencari Google terlihat lebih umum, berbeda dengan aplikasi pada penelitian yang menampilkan gambar dengan format JPG dan GIF. Namun pada mesin pencari Google bisa menelusuri pencarian dengan menggunakan *keyword* yang lebih spesifik untuk mendapatkan informasi yang lebih spesifik.

5.7.5 Hasil Uji Coba

Setelah melakukan beberapa uji coba terhadap aplikasi web berbasi *cloud* untuk pengelompokan hasil pencarian teks yang telah dibuat, dimulai dari uji coba kecocokan informasi dengan kategori, kesesuaian isi informasi dengan *keyword*, kecepatan pencarian dan perbandingan aplikasi yang dibuat dengan mesin pencari yang ada. Hasil dari beberapa uji coba tersebut dapat dilihat pada Tabel 5.4.

Tabel 5.6 Hasil uji coba

No	Proses Uji Coba	Hasil yang diharapkan	Keterangan
1	Pengujian Kategori	Informasi yang dihasilkan sesuai dengan kategori yang ada dengan rata-rata keberhasilan diatas 80%.	Berhasil
2	Perbandingan Kesesuaian *Keyword*	Mendapatkan hasil pengaruh perbedaan *keyword* untuk kesesuaian informasi dengan *keyword* yang digunakan	Berhasil
3	Perbandingan Kecepatan	Mendapatkan hasil dari pengaruh perbedaan *keyword* dalam menentukan kecepatan pencarian	Berhasil
4	Perbandingan Aplikasi	Aplikasi yang dibuat berbeda dengan mesin pencari yang sudah ada sebelumnya	Berhasil

BAB VI
PENUTUP

6.1 Kesimpulan

Dari hasil uji coba aplikasi web berbasis *cloud* menggunakan *framework* Codeigniter untuk pengelompokan hasil pencarian teks dapat disimpulkan sebagai berikut :

1. Aplikasi pencarian yang dibuat telah mampu mengimplementasikan 3 API, yaitu Google API, ScienceDirect API dan Flickr API untuk mendapatkan data.
2. Pada uji coba kesesuaian isi dengan kategori, rata-rata bahwa isi dari informasi tersebut adalah sesuai dengan kategori adalah 100% kecuali untuk gambar bergerak (GIF) hanya 44%, dimana percobaan yang dilakukn pada buku sebanyak 200 data, jurnal 111 data JPG 293 data dan video 23 data dan GIF 30 data. Artinya untuk pengelompokan data berdasarkan kategori buku, jurnal, JPG dan video bisa dikatakan berhasil, namun untuk kategori GIF masih belum.
3. Pada uji coba kesesuaian informasi yang diterima terhadap *keyword*, rata-rata kesesuaian Bahasa Inggris yaitu 96% lebih tinggi dari Bahasa Indonesia yaitu 66%. Percobaan yang dilakukan menggunakan 1342 data dengan bahasa Inggris menghasilkn 877 data dan bahasa Indonesia menghasilkan 465 data. Hal ini disebabkan karena server yang digunakan adalah server dengan basis Bahasa inggris seperti ScienceDirect dan Flickr, sedangkan untuk Bahasa Indonesia hanya disediakan oleh server Google
4. Pada uji coba kecepatan waktu pencarian berdasarkan *keyword*, rata-rata untuk Bahasa Inggris adalah 45.276 detik dan Bahasa Indonesia 28.46 detik. Pencarian dengan menggunakan *keyword* berbahasa inggris lebih

lama karena data yang ada di server lebih banyak sehingga membutuhkan waktu yang lebih banyak juga, hal ini dikarenakan ketiga server berbasis Bahasa Inggris.

5. Aplikasi ini dapat menggunakan API yang fungsinya mengambil data yang ada pada masing-masing server utama dalam satu kali pencarian dan data tersebut dimasukkan ke dalam *database*.
6. Aplikasi ini dapat mengelompokan data yang telah diambil menjadi 5 kategori yaitu, Buku, Jurnal, Gambar (JPG), Gambar (GIF) dan Video (MP4) berdasarka *metadata* yang ada dimana *metadata* tersebut berbentuk teks.
7. Aplikasi ini telah menerapkan *cloud computing* atau komputasi awan dimana pengelompokan teks tersebut terjadi ketika menggukana API yang terhubung ke internet serta mampu menampilkan semua hasil pencarian yang telah dilakukan dalam satu kali pencarian.

6.2 Saran

Berdasarkan hasil uji coba pada aplikasi yang telah dibuat masih terdapat banyak kekurangan. Untuk itu diberikan beberapa saran untuk pengembangan aplikasi berbasis web ini diantaranya adalah sebagai berikut :

1. Menambahkan kategori pada pencarian, semakin banyak kategori yang diberikan maka akan lebih memberikan kemudahan kepada *user*.
2. Mengatasi kecepatan pada saat melakukan pencarian. Untuk sebuah mesin pencari apabila cepat dan akurat maka akan semakin baik.
3. Membuat fungsi untuk *update* data secara berkala.

DAFTAR PUSTAKA

[1] Shaikh, F. B., dan Haider, S. (2011). *Security Threats in Cloud Computing*. *6th International Conference on Internet Technology and Secured Transactions, Abu Dhabi, United, IEEE*.

[2] P. Mell dan Grance T. (2011). *The NIST definition of cloud computing*. *National Institute of Standards and Technology (NIST)*.

[3] M. Armbrust dkk (2011). *Above the clouds*: *A Berkeley View of Cloud Computing. University of California Berkeley Reliable Adaptive Distributed Systems Laboratory, Berkeley*.

[4] Magang Industri. (2013). Definisi *Cloud Computing*. *Meruvian.org Cloud Computing*.

[5] Wardana, S.hut., m.si,. (2010). Menjadi Master PHP dengan *Framework* CodeIgniter. Jakarta : Elex Media.

[6] Hsieh, L.-C., G-L. W., Y.-M. H., dan W.H. (2014) *Online Image Result Grouping With MapReduce-based Image. J. Vis,. Commun. Image R. ScienceDirect.*

[7] Griffiths, Adam. (2010). *CodeIgniter 1.7 Professional Development. Brimingham* : *Packt Publishing.*

[8] https://www.codeigniter.com diakses pada tanggal 10 Desember 2016.

[9] https://id.wikipedia.org/wiki/Unified_Modeling_Language diakses pada tanggal 12 Desember 2016.

[10] Dharwiyanti, Sri. (2003). Pengantar *Unified Modeling Language (UML).* Ilmu Komputer.

[11] https://id.wikipedia.org/wiki/XAMPP diakses pada tanggal 12 Desember 2016.

[12] Andika, R.(2011). Penerapan CI (CodeIgniter) Dalam Pengembangan Sistem Informasi Manajemen Surat Dan Pengarsipan. Skripsi. Program Studi Teknik Informatika, Universitas Islam Negeri Jakarta.

[13] Dwi, K. U., Adi, M. Y., dan Ariel, M. D. P. (2014). Sinkronisasi Otomatis Status Dosen Dengan Raspberry Pi Melalui *Facebook Graph* API. Skripsi. Program Studi Teknik Informatika, Universitas BINUS.

[14] *3Scale Networks*. (2011). *What is an API?*

[15] Nurseitov, N. dkk. (2009). *Comparison of JSON and XML Data Interchange Formats: A Case Study. Department of Computer Science, Montana State University,USA.*

[16] https://id.wikipedia.org/wiki/Internet diakses pada tanggal 17 Januari 2017.

[17] Lin, W., W. D., Z. Z., dan C. L. (2015). *A cloud-based framework for Home-diagnosis service over big medical data*. *The Journal of Systems and Software, ScienceDirect*.

LAMPIRAN A

Hasil.php

```
<!DOCTYPE HTML>
<html >
<head>
<style>
h1{
font-size: 30px;
color: black;
text-transform: uppercase;
font-weight: 300;
text-align: center;
margin-bottom: 15px;
}
table{
width:100%;
table-layout: fixed;
}
.tbl-header{
background-color: rgba(0,0,0,0.3);
}
.tbl-content{
height:300px;
overflow-x:auto;
margin-top: 0px;
border: 1px solid rgba(0,0,0,0.3);
}
th{
padding: 20px 15px;
text-align: center;
font-size: 16px;
color: black;
text-transform: uppercase;
}
td{
padding: 15px;
vertical-align:middle;
font-size: 14px;
color: black;
border-bottom: solid 2px rgba(0,0,0,0.1);
```

```
}
.thick{
font-weight: bold;
}

body{
background: white;
font-family: 'Roboto', sans-serif;
}
section{
margin: 50px;
}
::-webkit-scrollbar {
width: 12px;
}
::-webkit-scrollbar-track {
-webkit-box-shadow: inset 0 0 12px rgba(0,0,0,0.3);
}
::-webkit-scrollbar-thumb {
-webkit-box-shadow: inset 0 0 12px rgba(0,0,0,0.3);
}
</style>
</head>
<body translate="no" >
<div style="position: fixed; top: 5px; left:
5px;"><a href="<?php echo
site_url('homecontroller/');?>"><img
src="../images/back1.png"></a></div>
<section>
<h1 style="font-family: courier"><b>Hasil Pencarian
Buku <?php echo $parameter ;?></b></h1>
<input style="" type="text" id="myInput"
onkeyup="myFunction()" placeholder="Cari Judul..">
<div class="tbl-header">
<table cellpadding="0" cellspacing="0" border="0">
<thead>
<tr>
<th width="5%" class="thick">No</th>
<th width="50%" class="thick">Judul</th>
<th width="15%" class="thick">Penulis/Penerbit</th>
<th width="15%" class="thick">Tahun Terbit</th>
<th width="15%" class="thick">Link Web</th>
```

```
</tr>
</thead>
</table>
</div>
<div class="tbl-content">
<table id="myTable" cellpadding="0" cellspacing="0"
border="0">
<tbody>
<?php $i=1; foreach ($resultDataApi as $row) {
if($row['id_jenis']==1 && $row['title']!=null){ ?>
<tr >
<td  width="5%" align="center" class="thick"><?php
echo $i++; ?></td>
<td width="50%" align="justify" class="thick"><?php
echo $row['title']; ?></td>
<td width="15%" align="center" class="thick"><?php
echo $row['creator']; ?></td>
<td width="15%" align="center" class="thick"><?php
echo $row['years']; ?></td>
<td width="15%" align="center" class="thick"><?php
echo "<a href='".$row['link']."'><img
src='../images/go.png'></a>"; ?></td>
</tr>
<?php }  }?>
</tbody>
</table>
</div>
</section>
<section>
<h1 style="font-family: courier"><b>Hasil Pencarian
Jurnal <?php echo $parameter ;?></b></h1>
<input type="text" id="Input" onkeyup="Function()"
placeholder="Cari Judul..">
<div class="tbl-header">
<table cellpadding="0" cellspacing="0" border="0">
<thead>
<tr>
<th width="5%">No</th>
<th width="50%">Judul</th>
<th width="15%">Penulis/Penerbit</th>
<th width="15%">Tahun Terbit</th>
<th width="15%">Link Web</th>
```

```
</tr>
</thead>
</table>
</div>
<div class="tbl-content">
<table id="Table" cellpadding="0" cellspacing="0"
border="0">
<tbody>
<?php $i=1; foreach ($resultDataApi as $row) {
if($row['id_jenis']==2 && $row['title']!=null){ ?>
<tr>
<td width="5%" align="center" class="thick"><?php
echo $i++; ?></td>
<td width="50%" align="justify" class="thick"><?php
echo $row['title']; ?></td>
<td width="15%" align="center" class="thick"><?php
echo $row['creator']; ?></td>
<td width="15%" align="center" class="thick"><?php
echo $row['years']; ?></td>
<td width="15%" align="center" class="thick"><?php
echo "<a href='".$row['link']."'><img
src='../images/go.png'></a>"; ?></td>
</tr>
<?php } }?>
</tbody>
</table>
</div>
</section>
<section>
<h1 style="font-family: courier"><b>Hasil Pencarian
Gambar (JPG) <?php echo $parameter ;?></b></h1>
<div class="tbl-header">
<table cellpadding="0" cellspacing="0" border="0">
<thead>
<tr>
<th width="5%">No</th>
<th width="20%">Kategori</th>
<th width="60%">Relevansi</th>
<th width="15%">Lihat Gambar</th>
</tr>
</thead>
</table>
```

```
</div>
<div class="tbl-content">
<table cellpadding="0" cellspacing="0" border="0">
<tbody>
<?php $i=1; foreach ($dataCluster->clusters-
>cluster as $now) { ?>
<tr>

<td width="5%" align="center" class="thick"><?php
echo $i++; ?></td>
<td width="20%" align="center" class="thick"><?php
echo $now->tag[0]->_content; ?></td>
<td width="60%" align="justify" class="thick">
<?php foreach ($now->tag as $nowContent){ echo
$nowContent->_content." | "; } ?></td>
<td width="15%" align="center">
<form action="<?php echo
site_url('hasilgambarcontroller/getApi/');?>"
method="post">

<input type="hidden" name="keywordId" value="<?php
echo $keyword;?>">
<input type="hidden" name="keyword" value="<?php
echo $parameter;?>">

<input type="hidden" name="flagKeyword"
value="<?php echo $flagKeyword;?>">
<input type="hidden" name="kategori" value="<?php
echo $now->tag[0]->_content ;?>">
<input type="image" src="<?php echo base_url();
?>images/go.png">
</form>
</td>
<?php } ?>
</tr>
</tbody>
</table>
</div>
</section>
<h1 style="font-family: courier"><b>Hasil Pencarian
Gambar (GIF) <?php echo $parameter ;?></b></h1>
<div class="tbl-header">
```

```
<table cellpadding="0" cellspacing="0" border="0">
<thead>
<tr>
<th width="30%">No</th>
<th width="40%">Gambar</th>
<th width="30%">Lihat Gambar</th>
</tr>
</thead>
</table>
</div>
<div class="tbl-content">
<table cellpadding="0" cellspacing="0" border="0">
<tbody>
<?php $i=1; foreach ($flickrgif as $row) { ?>
<tr>
<td width="30%" align="center" class="thick"><?php
echo $i++; ?></td>
<td width="40%" align="center"><?php echo "<img
src='".$row['link']."'>";?></td>
<td width="30%" align="center"><?php echo "<a
href='".$row['link_flickr']."'><img
src='../images/go.png'></a>";?></td>
</tr>
<?php } ?>
</tbody>
</table>
</div>
</section>
<section>
<h1 style="font-family: courier"><b>Hasil Pencarian
Video <?php echo $parameter ;?></b></h1>
<div class="tbl-header">
<table cellpadding="0" cellspacing="0" border="0">
<thead>
<tr>
<th width="30%">No</th>
<th width="40%">Link Video</th>
<th width="30%">Download</th>
</tr>
</thead>
</table>
</div>
```

```
<div class="tbl-content">
<table cellpadding="0" cellspacing="0" border="0">
<tbody>
<?php $i=1; foreach ($flickrvideo as $row) {?>
<tr>
<?php if($row['link']!=null) {?>
<td width="30%" align="center" class="thick"><?php
echo $i++; ?></td>
<td width="40%" align="center"><?php echo "<a
href='".$row['link_flickr']."'>".$row['link_flickr'
]."</a>"; ?></td>
<td width="30%" align="center"><?php echo "<a
href='".$row['link']."'><img
src='../images/download-icon.png'></a>";?></td>
<?php } ?>
</tr>
<?php } ?>
</tbody>
</table>
</div>
</section>
<script>
function myFunction() {
var input, filter, table, tr, td, i;
input = document.getElementById("myInput");
filter = input.value.toUpperCase();
table = document.getElementById("myTable");
tr = table.getElementsByTagName("tr");
for (i = 0; i < tr.length; i++) {
td = tr[i].getElementsByTagName("td")[1];
if (td) {
if (td.innerHTML.toUpperCase().indexOf(filter) > -
1) {
tr[i].style.display = "";
} else {
tr[i].style.display = "none";
}
}
}
}
```

```
function Function() {
var input, filter, table, tr, td, i;
input = document.getElementById("Input");
filter = input.value.toUpperCase();
table = document.getElementById("Table");
tr = table.getElementsByTagName("tr");
for (i = 0; i < tr.length; i++) {
td = tr[i].getElementsByTagName("td")[1];
if (td) {
if (td.innerHTML.toUpperCase().indexOf(filter) > -
1) {
tr[i].style.display = "";
} else {
tr[i].style.display = "none";
}
}
}
}
</script>
</body>
</html>
```

Home.php

```
<!DOCTYPE HTML>
<html>
<head>
<title><?php echo $title?></title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,
initial-scale=1" />
<script
src="assets/assets/js/ie/html5shiv.js"></script>
<link rel="stylesheet"
href="assets/assets/css/main.css" />
<link rel="stylesheet"
href="assets/assets/css/ie8.css" />
<link rel="stylesheet"
href="assets/assets/css/ie9.css" />
</head>
<style>
input[type=text] {
width: 220px;
box-sizing: border-box;
border: 2px solid #ccc;
border-radius: 4px;
font-size: 20px;
color: black;
background-color: khaki;
background-image: url('images/search-icon.png');
background-position: 2px 14px;
background-repeat: no-repeat;
padding: 12px 20px 12px 40px;
-webkit-transition: width 0.4s ease-in-out;
transition: width 0.4s ease-in-out;
}
input[type=text]:focus {
width: 50%;
}
</style>
<body class="loading">
<div id="wrapper">
<div id="bg"></div>
<div id="overlay"></div>
<div id="main">
```

```
<header id="header">
<p><img src="images/z.png"></p>
<font face="magneto" size="20">Zeta Engine</font>
<nav>
<ul>
<li><a href="<?php echo site_url(); ?>" class="icon
fa-home"><span class="label">Home</span></a></li>
<li><a href="<?php echo
site_url('welcomecontroller/'); ?>" class="icon fa-
info"><span class="label">About</span></a></li>
<li><a href="<?php echo site_url('addcontroller/');
?>" class="icon fa-plus"><span class="label">Add
Link</span></a></li>
<li><a href="<?php echo
site_url('welcomecontroller/'); ?>#contact"
class="icon fa-envelope-o"><span
class="label">Contact</span></a></li>
<li><a href="<?php echo
site_url('usercontroller/'); ?>#toregister"
class="icon fa-user-plus"><span
class="label">Register</span></a></li>
</ul>
<p><h1 style="font-size:180%; font-family: courier"
>Mulai Pencarian</h1></p>
<ul>
<form action="<?php echo site_url('test/find/');
?>" method="post">
<input type="text" name="keyword"
placeholder="Masukan Keyword..." ">
</form></ul></nav></header>
footer id="footer">
<span class="copyright">&copy; Untitled. Design: <a
href="http://html5up.net">HTML5 UP</a>.</span>
</footer>
<script>
window.onload = function() {
document.body.className = ''; }
window.ontouchmove = function() { return false; }
window.onorientationchange = function() {
document.body.scrollTop = 0; }
</script></body>
</html>
```

Test.php

```
<?php
defined('BASEPATH') OR exit('No direct script
access allowed');

class Test extends CI_Controller {
function __construct() {
parent::__construct();
$this->load->helper('url');
$this->load->model('BookModel');
$this->load->model('Flickrmodel');
}
public function find()
{
$parameter = $_POST["keyword"];
$dataHistory = $this->BookModel-
>gethistory($parameter);
$jumlahData = count($dataHistory);
if($jumlahData==0){

$data =
json_decode(file_get_contents("https://www.googleap
is.com/books/v1/volumes?q=".rawurlencode($parameter
)."&filter=partial&maxResults=20"));

//masukkan data keyword baru
$dataKeyword = array();
$dataKeyword['keyword'] = $parameter;
$idBaru = $this->BookModel-
>insertKeyword($dataKeyword);

$this->Sciencedirect(1,$parameter,$idBaru);
$this->Sciencedirect(2,$parameter,$idBaru);
$this->Flickr(1,$parameter,$idBaru);
$this->Flickr(2,$parameter,$idBaru);
foreach ($data->items as $now) {

$data = array();
if($now->volumeInfo->title){
$data['title'] = $now->volumeInfo->title;
}
```

```
else{
$data['title'] = "";
}
if($now->volumeInfo->publisher){
$data['creator'] = $now->volumeInfo->publisher;
}
else{
$data['creator'] = "";
}
if($now->volumeInfo->publishedDate){
$data['years'] = $now->volumeInfo->publishedDate;
}
else{
$data['years'] = "";
}
if($now->accessInfo->webReaderLink){
$data['link'] = $now->accessInfo->webReaderLink;
}
else{
$data['link'] = "";
}
$data['id_jenis'] = 1;
$data['id_keyword'] = $idBaru;
if($this->BookModel->insert_data_api($data)){
}
}
$idSekarang = $idBaru;
$flagKeyword = "baru";
}
else{
$idSekarang = $dataHistory[0]['id_keyword'];
$flagKeyword = "lama";
}
$resultDataApi = $this->BookModel-
>getDataApi($idSekarang);
$flickrgif = $this->Flickrmodel-
>getData($idSekarang,'gif');
$flickrvideo = $this->Flickrmodel-
>getData($idSekarang,'video');
```

```
$dataCluster =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.tags.getClusters&ap
i_key=b4b1c5b6cb61c7d630abb4eeb9bc83a7&tag=".rawurl
encode($parameter)."&format=json&nojsoncallback=1")
);
$resultData = $this->BookModel-
>get_data_manual($parameter);
$dataView = array();
$dataView['resultDataApi'] = $resultDataApi;
$dataView['resultData'] = $resultData;
$dataView['dataCluster'] = $dataCluster;
$dataView['keyword'] = $idSekarang;
$dataView['parameter']=$parameter;
$dataView['flagKeyword'] =  $flagKeyword;
$dataView['flickrgif'] =  $flickrgif;
$dataView['flickrvideo'] =  $flickrvideo;
$this->load->view( 'hasil',$dataView );
}
public function
Sciencedirect($tipe,$parameter,$idKeyword){

if($tipe==1){

$data =
json_decode(file_get_contents("http://api.elsevier.
com/content/search/scidir?httpAccept=application%2F
json&apiKey=090ac729d3355c8f5ef3e4bda127b212&query=
{".rawurlencode($parameter)."}&content=allbooks"));
}
else{
$data =
json_decode(file_get_contents("http://api.elsevier.
com/content/search/scidir?httpAccept=application%2F
json&apiKey=090ac729d3355c8f5ef3e4bda127b212&query=
{".rawurlencode($parameter)."}&content=journals"));
}
$dataApi = $data->{'search-results'}->entry;
foreach ($dataApi as $row ) {
$dataInput = array();
```

```
$dataInput["id"] = "";
$dataInput["id_jenis"] = $tipe;
$dataInput["link"] = $row->link[1]->{'@href'};
$dataInput["title"] = $row->{'dc:title'};
$dataInput["creator"] = $row->{'dc:creator'};
$dataInput["years"] = $row-
>{'prism:coverDisplayDate'};
$dataInput["id_keyword"] = $idKeyword;

$this->BookModel->insert_data_api($dataInput);
}
}
public function
Flickr($tipe,$parameter,$idKeyword){

if($tipe==1){
$data =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.photos.search&api_k
ey=b4b1c5b6cb61c7d630abb4eeb9bc83a7&tags=gif&text="
.$parameter."&per_page=15&format=json&nojsoncallbac
k=1"));


foreach ($data->photos->photo as $nowdata) {
$dataId =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.photos.getSizes&api
_key=b4b1c5b6cb61c7d630abb4eeb9bc83a7&photo_id=".$n
owdata->id."&format=json&nojsoncallback=1"));

$data = array();
$data['id_flickr']="";
$data['id_keyword']= $idKeyword ;
$data['id_tag']= "gif"  ;
$data['link']= $dataId->sizes->size[0]->source  ;
$data['link_flickr']= $dataId->sizes->size[0]->url
;

$this->Flickrmodel->insertData($data);
}
}
```

```
else{
$data =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.photos.search&api_k
ey=b4b1c5b6cb61c7d630abb4eeb9bc83a7&text=".$paramet
er."&media=videos&per_page=10&page=1&format=json&no
jsoncallback=1"));


foreach ($data->photos->photo as $nowdata) {
$dataId =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.photos.getSizes&api
_key=b4b1c5b6cb61c7d630abb4eeb9bc83a7&photo_id=".$n
owdata->id."&format=json&nojsoncallback=1"));

$data = array();
$data['id_flickr']="";
$data['id_keyword']= $idKeyword ;
$data['id_tag']= "video"  ;
$data['link']= $dataId->sizes->size[11]->source  ;
$data['link_flickr']= $dataId->sizes->size[11]->url
;

$this->Flickrmodel->insertData($data);
}
}
}
```

Hasilgambarcontroller.php

```
<?php
defined('BASEPATH') OR exit('No direct script
access allowed');
class Hasilgambarcontroller extends CI_Controller{

function  __construct() {
parent::__construct();
$this->load->model('Flickrmodel');
$this->load->model('BookModel');
}
public function getApi(){
$parameter = $_POST["keyword"];
$flagKeyword = $_POST["flagKeyword"];
$kategori = $_POST["kategori"];
$keywordId = $_POST["keywordId"];
if($flagKeyword=="baru"){

$data =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.tags.getClusterPhot
os&api_key=b4b1c5b6cb61c7d630abb4eeb9bc83a7&tag=".r
awurlencode($parameter)."&cluster_id=".$kategori."&
format=json&nojsoncallback=1"));

foreach ($data->photos->photo as $nowdata) {

$dataGambar =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.photos.getSizes&api
_key=b4b1c5b6cb61c7d630abb4eeb9bc83a7&photo_id=".$n
owdata->id."&format=json&nojsoncallback=1"));

$data = array();
$data['id_flickr']="";
$data['id_keyword']= $keywordId  ;
$data['id_tag']= $kategori ;
$data['link']= $dataGambar->sizes->size[4]->source
      ;
$data['link_flickr']= $dataGambar->sizes->size[4]-
>url ;
```

```
$this->Flickrmodel->insertData($data);
}
}
$allData = $this->Flickrmodel-
>getData($keywordId,$kategori);
$dataView = array();
$dataView['addData'] = $allData;
//$this->load->view("",$dataView);
$i = 1;
if (count($allData)>0) {
echo "data dari database kategori
<b>".$kategori."</b><br>";
foreach ($allData as $row) {
# code...
echo $i++;
echo "<a href='".$row['link_flickr']."'><img
src='".$row['link']."'>  </a>";
}}
else {
if($allData['id_tag']!=$kategori){

$data =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.tags.getClusterPhot
os&api_key=b4b1c5b6cb61c7d630abb4eeb9bc83a7&tag=".r
awurlencode($parameter)."&cluster_id=".$kategori."&
format=json&nojsoncallback=1"));

foreach ($data->photos->photo as $nowdata) {
$dataGambar =
json_decode(file_get_contents("https://api.flickr.c
om/services/rest/?method=flickr.photos.getSizes&api
_key=b4b1c5b6cb61c7d630abb4eeb9bc83a7&photo_id=".$n
owdata->id."&format=json&nojsoncallback=1"));
// echo $dataGambar->sizes->size[0]->source."id
keyword ".$keywordId." id tag".$kategori."<br>";

$data = array();
$data['id_flickr']="";
$data['id_keyword']= $keywordId ;
$data['id_tag']= $kategori  ;
$data['link']= $dataGambar->sizes->size[4]->source;
```

```
$data['link_flickr']= $dataGambar->sizes->size[4]-
>url ;

$this->Flickrmodel->insertData($data);
}
}
$allData = $this->Flickrmodel-
>getData($keywordId,$kategori);
// $dataView = array();
// $dataView['addData'] = $allData;
//$this->load->view("",$dataView);
$i = 1;
if (count($allData)>0) {
# code...

echo "data dari database kategori
<b>".$kategori."</b><br>";
foreach ($allData as $row) {
# code...
echo $i++;
echo "<a href='".$row['link_flickr']."'><img
src='".$row['link']."'>   </a>";
}
}
}
}
}
```

Homecontroller.php

```
<?php
class Homecontroller extends CI_Controller{

function  __construct() {
parent::__construct();
$this->load->helper('url');
}

public function index()
{
$data['title'] ='Home Search';
$this->load->view( 'home',$data );
}
}
```

Hasilcontroller.php

```
<?php
defined('BASEPATH') OR exit('No direct script
access allowed');

class Hasilcontroller extends CI_Controller{

public function index()
{
$this->load->helper('url');
$data['title'] ='Hasil';
$this->load->view( 'hasil',$data );
}
}
```

BookModel.php

```
<?php
class BookModel extends CI_Model {

function __construct() {
parent::__construct();
}

public function insert_data_api($data) {
if ($this->db->insert("data_api", $data)) {
return true;
}
}

public function gethistory($keyword){
$query = $this->db->get_where('keyword',
array('keyword' => $keyword));
return $query->result_array();
}

public function insertKeyword($data){
if($this->db->insert("keyword",$data)){
$idBaru = $this->db->insert_id();
return $idBaru;
}
return 0;
}

public function getDataApi($idKeyword){
$query = $this->db->get_where('data_api',
array('id_keyword' => $idKeyword));
return $query->result_array();
}

public function get_data_manual($key){

$cari=$this->db->query("select * from
data,tahun,jenisdata where data.id_jenis =
jenisdata.id and data.id_tahun = tahun.id_tahun and
(deskripsi like '%$key%' or judul like '%$key%')");
return $cari->result(); }}  ?>
```

FlickrModel.php

```
<?php

class Flickrmodel extends CI_Model{

function  __construct() {
parent::__construct();
$this->load->database();
}

public function insertData($data){
if ($this->db->insert("data_flickr", $data)) {
return true;
}
}

public function getData($keyword,$tag){
$query = $this->db->get_where('data_flickr',
array('id_keyword' => $keyword, 'id_tag' => $tag));
return $query->result_array();
}

public function getTagId(){
$query = $this->db->get('data_flickr');
if ($query->num_rows() > 0){
return $query->result();
} else {
return array ();
}
}
}
```

LAMPIRAN B

Flow Chart Sistem Aplikasi Pencarian