APLIKASI *WEB GRABBER* UNTUK MENGAMBIL HALAMAN *WEB* SESUAI DENGAN *KEYWORD* YANG DIINPUTKAN

Gregorius S. Budhi, Djoni H. Setiabudi, dan Budi Raharjo
Fakultas Teknologi Industri, Jurusan Teknik Informatika, Universitas Kristen Petra
e-mail: greg@petra.ac.id, djonihs@petra.ac.id

ABSTRAK: Saat ini penggunaan *search engine* pada *internet* masih belum memperoleh hasil yang maksimal. Pengguna masih harus *browsing* halaman-halaman *web* hasil pencarian untuk mencari topik yang diinginkan satu-persatu. Untuk itu perlu dibuatkan aplikasi yang dapat mencari dan menyimpan halaman-halaman *web* dengan *link-link*-nya sesuai topik tanpa harus melalui proses *browsing*, sehingga hasilnya dapat dilihat secara *offline*. Aplikasi ini biasa disebut *Web Grabber*. Program aplikasi *Web Grabber* ini mendapatkan inputan berupa *keyword* dan kedalaman pencarian. Inputan ini akan diproses untuk memperoleh hasil berupa halaman – halaman *web* yang disimpan secara otomatis sesuai dengan *keyword* yang diberikan dan tingkat kedalaman pencariannya. Proses didahului dengan pencarian *keyword* pada *search engine* yang tersedia menjadi sebuah list yang dapat dipilih oleh pengguna. Hasil aplikasi ini berupa halaman – halaman *web* yang mengandung *keyword*. Aplikasi juga menyediakan fasilitas untuk *browsing* hasil yang didapat. Program yang dibuat berhasil melakukan penyimpanan halaman-halaman *web* sesuai dengan *keyword* yang diberikan dengan tingkat kedalaman tertentu, yang sebelumnya dicari melalui *search engine Altavista* atau *Google*. Halaman-halaman *web* yang mengandung *keyword* dapat diambil dan disimpan pada komputer sehingga dapat di-*browse* secara *offline*.

Kata kunci: *web grabber*, *keyword*, kedalaman, *search engine*, *internet.*

ABSTRACT: *At this time the use of search engine on the Internet still does not reach maximum result. Users need to browse every web page list as the result of the search that fit with the topic one by one. We need an application that has a capability to look and to save web pages with their links fit with the topic without browsing, so that the result can be accessed offline. This application is called Web Grabber. Web Grabber application uses inputs such as keyword and searching depth. The input will be processed to obtain the appropriate web pages with the keyword and the searching depth, which is automatically saved. Firstly a keyword is needed to be search in the available search engine to get the search lists, which can be picked by the user. Every pick list will be used as the first search page. This application results are web pages that contain the keyword. This application also has a facility to browse the results. This application was succeed to automatically save web pages that appropriate with the input keyword and the depth, that previously search in AltaVista or Google. Web pages that contain with the keyword saved in the computer so that it can be browsed offline.*

Keywords: *web grabber, keyword, depth, search engine, internet.*

PENDAHULUAN

Internet sebagai sarana media informasi banyak diminati oleh masyarakat. Dengan menggunakan *internet*, orang dengan mudah mengakses jutaan halaman *web* yang sarat informasi. Dengan memasuki satu situs saja, orang bisa *browsing* kemana saja.

Masalah timbul setelah mendapatkan halaman *web* yang dicari, *link-link* yang ada pada halaman *web* tersebut satu persatu harus di-*browsing* untuk mencari informasi yang diinginkan. Hal ini harus dilakukan secara *online*. Ditambah lagi proses penyimpanan harus dilakukan secara manual untuk tiap halaman *web*.

Dari masalah di atas, peneliti membuat aplikasi *offline browser* yang dapat melakukan *search* topik yang diinginkan. Kemudian dari masing-masing *list* dari *link* yang didapatkan, aplikasi melakukan pemilihan dan penyimpanan halaman-halaman *web* yang mengandung kata dari topik beserta semua halaman cabang *link*-nya, bila halaman *web* dari *link* cabang masih mengandung kata yang sama dengan *keyword* topik yang dimasukkan sehingga pembacaan halaman web diinginkan dapat dilakukan secara *offline*. Aplikasi seperti ini biasa disebut *Web Grabber*.

TINJAUAN PUSTAKA

Web Browser, Offline Browser dan *Web Grabber*

Web Browser adalah sebuah program aplikasi yang memungkinkan *user* untuk melihat dan berinteraksi dengan teks, *image* dan informasi lain yang terletak pada *web page* yang berada pada *World Wide*

Web atau *Local Area Network*. Teks atau *image* pada sebuah *web page* dapat mengandung *hyperlink ke web page* lain pada website yang sama lainnya. *Web Browser* memungkinkan user untuk secara cepat dan mudah mengkases informasi dari banyak *web page* pada banyak *website* dengan mengikuti *link - link* yang ada [7].

Web Browser modern memiliki mode *offline browsing*, dimana sebuah *web page* disimpan secara lokal pada *folder* sementara dan akan dihapus secara otomatis pada waktu tertentu. Beberapa *web browser* bahkan menyediakan fasilitas untuk *user* guna memilih halaman-halaman mana yang dapat di-*browse* secara *offline* dan menggaransi bahwa halaman-halaman tersebut tidak dihapus secara otomatis. Aplikasi *offline browser* yang lain dapat mengikuti *link-link* pada sebuah *page* dan mnyimpan semua page yang dari *link-link* tersebut dalam *harddisk* lokal agar dapat dilihat *(browsed)* setelah *user disconnected* [8].

Web Grabber yang dibuat ini adalah sebuah aplikasi *Offline Browser* yang dapat mencari page - page pada *World Wide Web* sesuai dengan "*keyword*" yang ditulis, mengikuti *link-link* dari *page-page* tersebut sampai kedalaman tertentu secara rekusif dan selanjutnya menyimpan semua *page* yang ditemukan ke dalam *harddisk* lokal agar dapat dilihat lagi setelah *user disconnected* dari *World Wide Web*.

Teknologi Pembangun *Web*

HTML

Hypertext Markup Language (*HTML*) adalah format *hypertext* yang dipakai di *web*. *HTML* adalah standar yang didefinisikan oleh *W3C* (*World Wide Web Consortium*), yaitu badan yang mengontrol *internet*. Standar yang ada sekarang adalah *HTML 4* [4].

Kunci *HTML* adalah *tag* pembuka (misalnya <HTML>) dan *tag* penutup (</HTML>). Di dalam <HTML> dan </HTML> dapat ditambahkan pasangan *tag-tag* yang lain [4].

Link

Halaman *web* dapat terhubung ke halaman *web* lain. Untuk menambahkan *link*, digunakan pasangan *tag anchor*, yaitu <A>…</A>. *Tag* <A> mempunyai 3 atribut [4]:

- *HREF* untuk membuat *link* ke halaman lain.
- *NAME* untuk membuat *link* pada halaman yang sama.
- *TARGET* untuk menyatakan *file* yang berhubungan dengan atribut *HREF*.

Bentuk umum:

<A HREF = “URL”>Label</A>

URL menyatakan *URL* dari halaman, dan Label menyatakan label yang ditampilkan dan dapat dipilih.

URL

URL (*Uniform Resource Locator*) adalah deskripsi komplit dari lokasi sebuah *resource* yang ada pada jaringan. Berikut ini adalah contoh dari URL [4]:
http://www.Tsite.com/art/coba.dll/mamalia?hewan=kucing&warna=putih

URL adalah bagian dari *URI* (*Uniform Resource Identifier*) yang didefinisikan di dalam standar *HTTP* (*RFC1945*).

Backtrack

Backtrack adalah salah satu metode penyelesaian masalah dengan membagi masalah ke sejumlah *level* proses. Dalam setiap *level* terdapat sejumlah solusi. Dalam *backtrack* digunakan pemrograman secara rekursif, dimana program yang dikerjakan tiap *level*-nya adalah sama [7].

Algoritma *backtrack* secara umum adalah sebagai berikut [7]:

Prosedur *Backtrack* (level n)
1. Inisialisasi pemilihan prioritas pemanfaatan solusi
2. Sukses ← ***false***
3. **Repeat** thru step 4 until sukses **Or** solusi habis
4. Ambil solusi berikut sesuai prioritas
5. **If** solusi dapat dipakai **Then** Jalankan mengikuti solusi tersebut
6. **If** Solusi belum lengkap **Then** Sukses ← Backtrack (level n+1)
7. **If** Tidak sukses **Then** Batalkan tindakan untuk solusi tersebut
8. **Return** Sukses

Untuk implementasi *backtrack* pada aplikasi *web grabber* ini, “sukses” adalah ada atau tidak kata yang akan dicari pada halaman *web* tertentu yang di-*grab*. Sedangkan “solusi” didasarkan pada prioritas yang dipilih sesuai *link-link* pada halaman *web* secara berurutan dari awal sampai akhir. “*Level”* adalah jumlah kedalaman yang diinginkan pada suatu *site*.

NMHTTP

NMHTTP adalah sebuah komponen dari *Borland Delphi 5.0* yang digunakan untuk mengambil isi dari suatu *URL*. Hasil pengambilan dari *NMHTTP* adalah

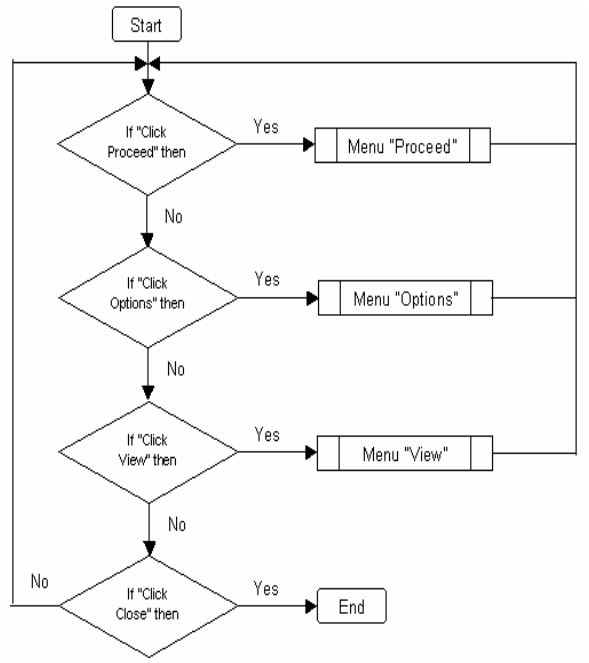
text base dan tidak dapat secara langsung dilihat hasilnya [4].

WebBrowser

Komponen *WebBrowser* dari Borland Delphi 5.0 digunakan untuk *browsing* halaman *web*. Metode yang digunakan untuk *download* pada komponen ini adalah *"Navigate"* [4].

DESAIN SISTEM

Program terdiri dari tiga bagian yaitu *proceed, view*, dan *options*



Gambar 1. *Flowchart* Menu Utama Program

Menu *Proceed*

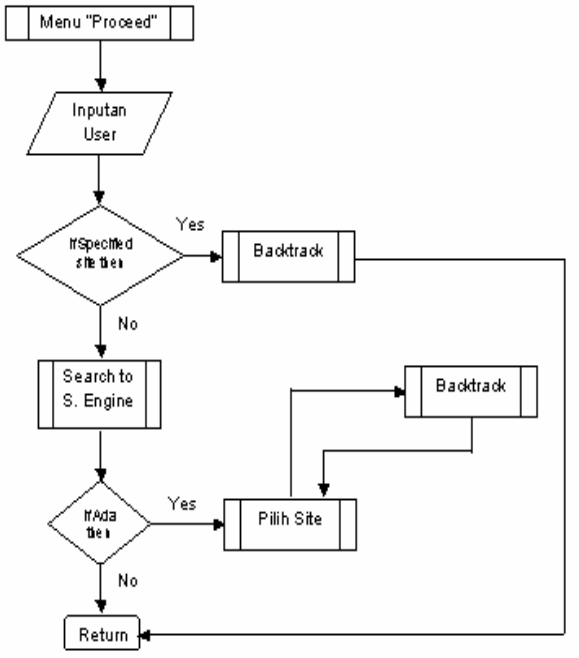
Pada bagian ini, program akan memproses inputan *user*, yang berupa kata yang dicari, kedalaman pencarian dan s*ite* spesifik jika pencarian tidak menggunakan *search engine*. Setelah itu, inputan dari *user* tersebut diolah untuk mendapatkan *web page* sesuai dengan keinginan.

Search to Search Engine

Proses ini dilakukan bila *user* memberikan inputan dan memilih untuk menggunakan *search engine* dari *Altavista* atau *Google* sebagai alat bantu pencarian web page yang diinginkan.

Pada pencarian menggunakan *search engine Altavista,* kata yang ingin dicari *user* dikirim ke *Altavista* menggunakan *URL* dengan format seperti berikut ini:

http://www.altavista.com/web/results?q=....&kgs=0 &kls=1&avkw=xytx&stq=0



Gambar 2. *Flowchart* Menu *Proceed*

Sementara bila menggunakan *search engine Google,* URL yang digunakan memiliki format sebagai berikut:
http://www.google.com/search?q=....&hl=en&lr=&ie=UTF-8&start=0&sa=N

Untuk mengetahui apakah *search engine* menemukan *web page* yang diinginkan, dilakukan pengecekkan isi *html* hasil *URL* yang dimasukkan, atau dengan kata lain *html* yang diterima harus dibuka.

Cara pengecekan pada hasil html dari *Altavista* adalah dengan mencari kata berikut ini:
onMouseOver="status=

Kata ini akan dilanjutkan dengan alamat tiap *web page* yang dihasilkan oleh search engine *Altavista*. Pencarian dilakukan pada seluruh halaman *html* yang didapat *Altavista* itu sampai habis.

Sedangkan pada *Google*, indikator yang mengarahkan pada alamat-alamat *web page* yang didapat oleh *Google* adalah:
q=related:

Pencarian dilakukan pada halaman html yang didapat sampai indikator tersebut tidak lagi ditemukan pada halaman yang bersangkutan. Jika pencarian pada *search engine* membawa hasil, maka hasil yang berupa alamat-alamat *web* akan ditampilkan dan

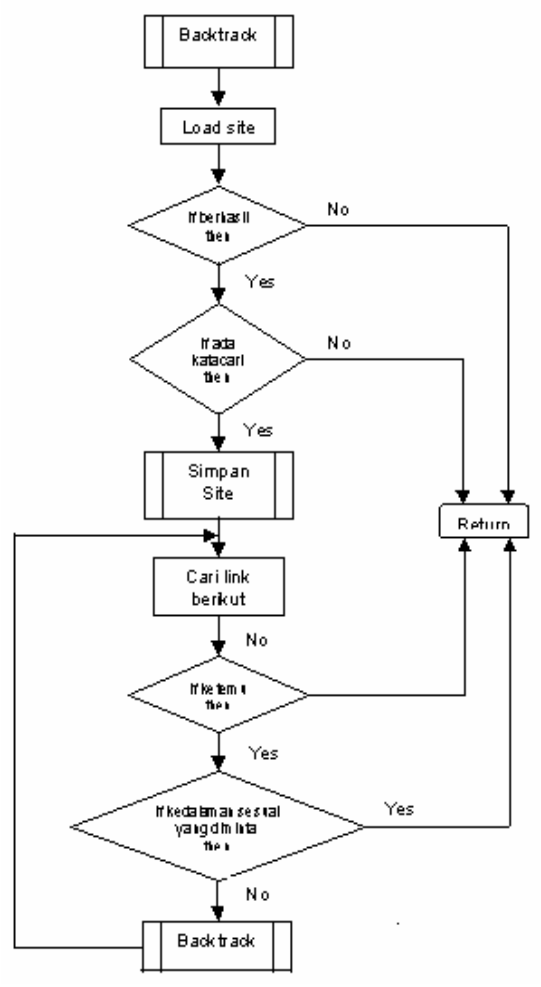
dapat dipilih oleh *user*. Proses ini terjadi pada proses Pilih *Site*.

Pilih *Site*

Setelah hasil *search* dari *search engine* diambil hasilnya, maka masing-masing hasil yang berbentuk alamat *URL* ditampilkan pada suatu *list* yang biasa disebut *CheckListBox*. Disini, *user* dapat memilih *page-page* mana yang diinginkan, dengan cara memberikan *check* (√) pada *web page* yang dimaksud. Banyaknya alamat *web* yang ditampilkan pada *CheckListBox* ini sebanyak 10 buah.

Metode *Backtrack* untuk pencarian halaman–halaman web

Pada bagian ini semua *link web page* sampai kedalaman yang diinginkan akan di cari satu per satu menggunakan metode *backtrack programming*.

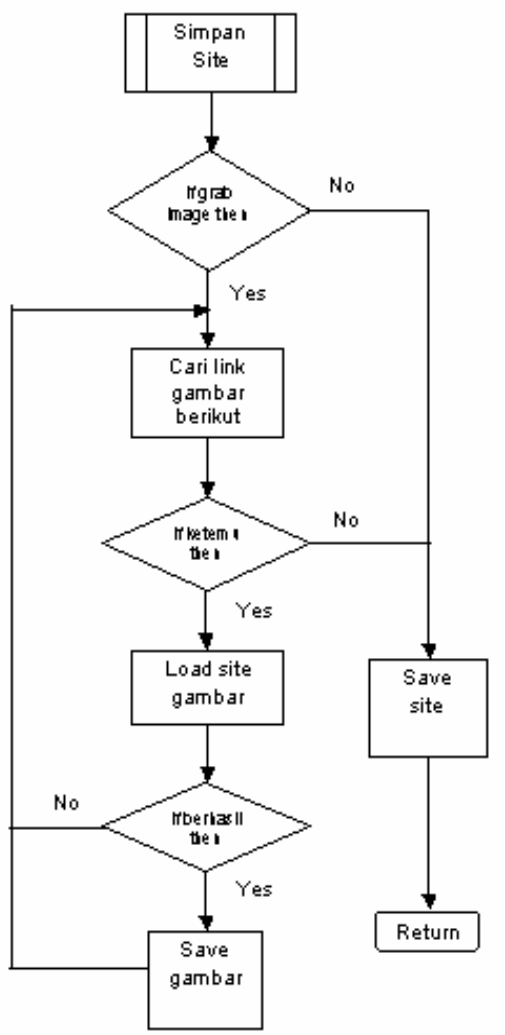


Gambar 3. *Flowchart* dari *Backtrack*

Setelah sebuah *web page* disimpan (akan dijelaskan pada sub bab berikutnya), *html* yang terambil tadi kembali diproses lagi untuk dicari *link-link*-nya. Caranya dengan mencari *tag* '<A HREF=…>' pada halaman *html* tersebut.

Proses penyimpanan *Site/Web Page*

Proses penyimpanan site/web page ini terbagi menjadi dua bagian, yaitu penyimpanan *site* tanpa menggunakan *image*, dan penyimpanan *site* dengan menggunakan *image*. Untuk pemilihan penyimpanan tanpa *image* atau dengan *image* dapat di-set terlebih dahulu.



Gambar 4. *Flowchart* dari Proses Penyimpanan *Site/Web Page*

Menu *Options*

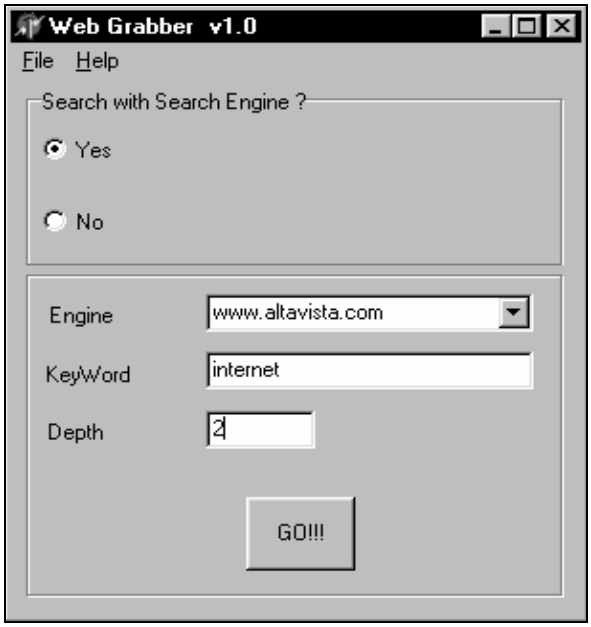
Menu *Options* digunakan untuk mengarahkan posisi *path directory* untuk penyimpanan *file web page* yang diambil dan untuk memberikan pilihan bagi *user* apakah ingin mengambil *image* dari *web page yang* disimpan atau tidak.

Viewing Grab Result (*Menu *View)

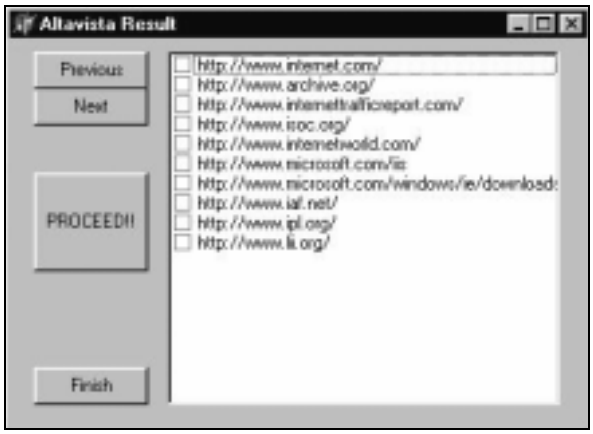
Untuk melihat isi *web* yang berhasil diambil, disediakan *browser* sederhana untuk melihat hasil *web* yang tersimpan satu persatu. Caranya dengan *browsing directory* terlebih dahulu sampai *list* yang berekstensi *html* keluar kemudian dipilih salah satu untuk melihat hasilnya.

EVALUASI *SOFTWARE*

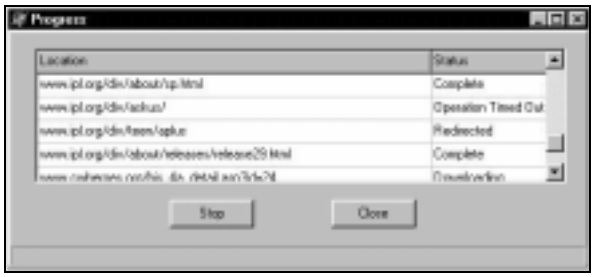
Dari hasil pengujian aplikasi ini dapat diketahui bahwa aplikasi telah berjalan dengan baik, seperti terlihat pada beberapa *screenshot* berikut ini.



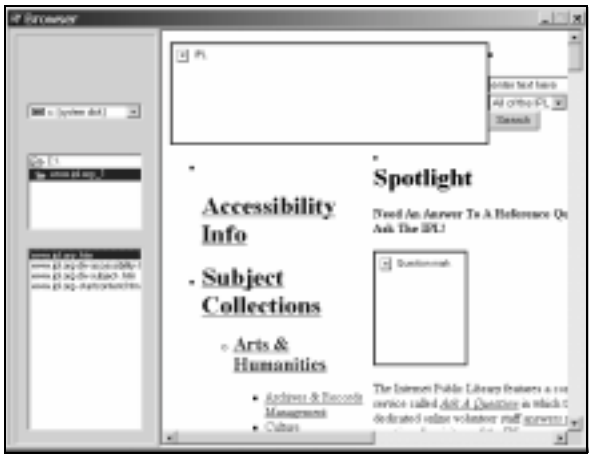
Gambar 5. Mencari *web page* yang mengandung kata "*internet*" dengan bantuan *search engine Altavista*, dengan kedalaman 2



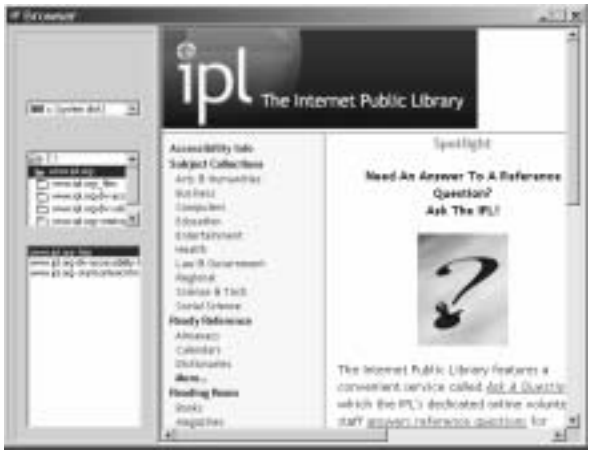
Gambar 6. Result Pencarian *Altavista*



Gambar 7. *Progress* Pencarian *p*ada *Www.Ipl. Org*



Gambar 8. *Browsing/View* pada *www.ipl.org* yang telah di-*grab*, dengan setting *grabbing* tanpa menyimpan *image* yang ada pada *web page*



Gambar 9. *Browsing/View* pada www.ipl.org yang telah di-*grab*, dengan setting *grabbing* dengan menyimpan *image* yang ada pada *web page*

KESIMPULAN

Berdasarkan penjelasan tentang perancangan dan pembuatan aplikasi *Web Grabber* ini dapat ditarik beberapa kesimpulan, yaitu antara lain:

- Program aplikasi *Web Grabber* yang dibuat berhasil melakukan pencarian halaman-halaman *web* sesuai dengan topik yang diinginkan tanpa harus membuka sendiri halaman-halaman *web* secara manual. Hasilnya yang berupa halaman-halaman *web* secara otomatis tersimpan pada komputer dan dapat dilihat secara *offline*.
- Timbul kesulitan bila ada *redirection* pada sebuah *URL*. Hal ini karena alamat *URL* baru sulit untuk ditelusuri, sehingga *link-link* cabang alamat baru tersebut tidak dapat dicari lebih lanjut.
- Ditemui kesulitan pada saat proses, bila pada *tag html* halaman *web* yang diambil ditemukan *tag script*. *Link-link* arahan dari *tag script* pada *html* ini masih belum dapat diambil.
- Adanya kesulitan saat grabbing, yang disebabkan oleh variasi bentuk penulisan *link*, baik itu *link* halaman maupun *link* untuk *image*.

DAFTAR PUSTAKA

1. Brin, Sergei, Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", http://www.knowlesys.com/research/paper/,1998.
2. Campbell, D.M., Y. Ding, D.W. Embley, K. Hewett, D.L. Jackman, S.S. Jeffries, Y.S. Jiang, D. Lewis, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, A.L. Peacock, D.J. Seer, R.D. Smith, S.H. Yau, M. Xu, L. Xu, "Demonstration: A Robust Web Data-Extraction Technique With High Recall and Precision", http://www. knowlesys.com/research/ paper/, 2000
3. Leander, Alberto H.F., Berthier A. Ribeiro Neto, Altigran S. da Silva, Juliana S. Teixeira, "A Brief Survey of Web Data Extraction Tools", http://www.knowlesys.com/research/paper/,2002.
4. Martina, Inge, *Pemrograman Internet dengan Delphi*. Jakarta: PT Elex Media Computindo, 2002.
5. Martina, Inge, *Delphi 5.0*. Jakarta: PT Elex Media Komputindo, 2000.
6. Pranata, Antony, *Tip dan Trik Pemrograman Delphi*. Yogyakarta: Andi Offset, 1998.
7. Weiss, Mark Allen**,** *Data Structures and Algorithm Analysis in C++*, Addison-Wesley, 2004
8. Wikipedia,http://en.wikipedia.org/wiki/Web_browser
9. Wikipedia,http://en.wikipedia.org/wiki/Offline_browsing