COOKIESLIVE: APLIKASI *STORAGE COOKIES* BERBASIS *WEB*

OLEH:

GANIS ZULFA SANTOSO

0404030393

SKRIPSI

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

COOKIESLIVE: APLIKASI *STORAGE COOKIES* BERBASIS *WEB*

SKRIPSI

OLEH:

GANIS ZULFA SANTOSO

0404030393

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

COOKIESLIVE*: APLIKASI *STORAGE COOKIES* BERBASIS *WEB

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada pendidikan Sarjana S1 Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan duplikasi dari skripsi yang telah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar keserjanaan di lingkungan universitas Indonesia maupun di perguruan tinggi atau instansi manapun, kecuali bagian yang sumber informasinya telah dicantumkan sebagaimana mestinya.

Depok, 24 Maret 2008

(Ganis Zulfa Santoso)

NPM 0404030393

PENGESAHAN

Skripsi dengan judul:

COOKIESLIVE*: APLIKASI *STORAGE COOKIES* BERBASIS *WEB

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan pada sidang skripsi.

Depok, 24 Maret 2008

Dosen Pembimbing

M. Suryanegara, S.T., M.Sc.

NIP. 040 705 0189

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada:

Muhammad Suryanegara, S.T., M.Sc.

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan pengarahan, diskusi, bimbingan dan saran-saran, serta kepada:

Prof. Dr. Ir. Dadang Gunawan MEng

selaku ketua Wireless dan Signal Processing Work Group yang telah membiayai skripsi ini sepenuhnya, sehingga skripsi ini dapat diselesaikan dengan baik.

Ganis Zulfa Santoso
NPM 04 04 03 0393

Dosen Pembimbing
M.Suryanegara, S.T., M.Sc.

Departemen Teknik Elektro

COOKIESLIVE*: APLIKASI *STORAGE COOKIES* BERBASIS *WEB

ABSTRAK

Skripsi ini membangun *CookiesLive*, sebuah aplikasi penyimpanan *cookies* yang berbasiskan *web* melalui internet. Aplikasi yang dibangun terbatas hanya untuk *browser Mozilla Firefox*.

Pada penggunaan *CookiesLive*, *user* pertama-tama mengambil *file cookies* dari *server CookiesLive*. Ketika sudah didapat hasil *file* dari database *server*, *user* bisa me-*load file* tersebut ke *Mozilla Firefox*. Setelah di-*load*, *user* akan merasakan pengalaman yang sama dengan *browser* yang berbeda.

Ujicoba dan analisa dilakukan untuk mengukur signifikansi dari penggunaan *CookiesLive* bagi *users*. Penghitungan signifikansi dilakukan dengan melakukan autentifikasi terhadap 20 *website*. Lalu dilakukan penghitungan *bandwidth* dan waktu yang terbuang. Didapat hasil selisih dari kedua metode dan dilihat peningkatan dari *CookiesLive* untuk masing-masing *cookies*. Tercatat peningkatan efisiensi mencapai 1178% untuk waktu dan 334.0% untuk bandwidth. Penghitungan yang lain adalah memvariasikan jumlah *account* yang digunakan. Dari hasil ujicoba didapat, jika user memiliki tujuh *account* atau lebih, maka *user* akan dapat merasakan manfaat dari program *CookiesLive* dari sisi *bandwidth* dan waktu.

Kata kunci : Cookies, Web, Bandwidth, Firefox, Storage, Database

Ganis Zulfa Santoso
NPM 04 04 03 0393
Counselor :
M.Suryanegara, S.T., M.Sc.
Electrical Engineering Departement

COOKIESLIVE: COOKIES STORAGE APPLICATION BASED ON WEB

ABSTRACT

This project is intended to build CookiesLive, an application for cookies storage based on web in internet network. This application is limited for Mozilla Firefox Browser.

In the usage of CookiesLive, users have to load cookies file from their cookies file ini CookiesLive server. When the users get the file from the database server, users can load the file to Mozilla Firefox. After it's loaded, user can feel the same experience with the different browser. This will enchance comfort, time efficiency as well as bandwidth efficency.

The trial and analysis is done to measure the significant of the usage of CookiesLive for users. The measurement is done by try to login to 20 websites. Then the measurement is done by counting the used time and bandwidth. From the test we get the efficiency improvement to 1178% for used time and 334.0% from bandwidth. Another measurement is by change the number of used account. From the test we get that if a user have at least seven account then user can feel the benefit of CookiesLive.

Keywords : Cookies, Web, Bandwidth, Firefox, Storage, Database

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR SINGKATAN

PHP	*Hypertext Preprocessor.*
HTTP	*Hypertext Transfer Protocol*
CSS	*Cascading Style Sheets*
HTML	*Hyprtext Markp Language*
GUI	*GraphicUser Interface*
Kb	*Kilobytes (1024 bytes)*

DAFTAR ISTILAH

Packaging	Proses pemaketan dengan *command* zip. Paket berbentuk jar maupun xpi.
Unpack	melakukan proses *reverse* dari *packaging.*
Client-side scripting	Bahasa yang diproses di komputer *user,* akibatnya user bisa melihat *source code*-nya.
Server-side scripting	Bahasa yang diproses di *server,* sehingga *user* hanya melihat hasil proses saja.

BAB I

PENDAHULUAN

1.1. LATAR BELAKANG MASALAH

Perkembangan internet dewasa ini menuntut semua data dapat diakses *online* dan aplikasi berbasiskan *web*. Akibatnya aplikasi berbasis *web* pun bermunculan, mulai dari yang menyediakan tempat untuk *networking, chatting, word processing, image processing,* dan lain sebagainya. Dengan semakin banyaknya *website* seperti ini membuat *users* harus memiliki *account* untuk setiap *website* untuk menikmati layanannya. Akibatnya *users* harus mengingat kombinasi *username*, *password* untuk masing-masing *website*.

Skripsi ini membahas rancang bangun dari *CookiesLive*, sebuah aplikasi berbasis web untuk menyimpan *cookies* secara *online* dan me-*load* cookies tersebut di *browser* milik *users*. Dengan adanya aplikasi ini, *users* dapat menggunakan komputer orang lain dan tetap merasakan pengalaman yang sama dengan komputernya sendiri. Karena *cookies* menyimpan preferensi dan autentifikasi dari sebuah *website.*

Dengan adanya program ini, maka *users* akan merasakan kenyamanan dalam melakukan aktifitas internetnya. Selain itu akan dicapai efesiensi dalam *waktu* dan *bandwidth*. Program ini dapat dibangun dengan JavaScripts sebagai sarana untuk menuliskan *cookies,* sementara PHP dan MySQL untuk melakukan *storage* secara *online*.

1.2. PERUMUSAN MASALAH

Memberikan *users* pengalaman yang sama di setiap *browser*. Sehingga kenyamanan dan efisiensi *waktu* dan *bandwidth* dapat tercapai.

1.3. TUJUAN

1. Membangun aplikasi *CookiesLive* yang berbasis *web*.

2. Mengukur dan menganalisa signifikansi dari penggunaan *CookiesLive*.

1.4. BATASAN MASALAH

Permasalahan dibatasi pada rancang bangun dan analisa kerja dari *CookiesLive System*. Rancang bangun ini meliputi pembuatan program yang terintegrasi pada *browser Mozilla Firefox* dan perancangan sistem penyimpanan di internet. Analisa dan ujicoba *CookiesLive* dilakukan secara *realtime* melalui media internet. Masalah keamanan jaringan *CookiesLive* tidaklah dibahas.

1.5. METODOLOGI PENELITIAN

Penelitian dilakukan dengan membangun *CookiesLive* secara keseluruhan. Kemudian dilakukan ujicoba dengan parameter jumlah *website* yang digunakan dan dilihat signifikansi *CookiesLive*.

1.6. SISTEMATIKA PENULISAN

Skripsi ini terdiri dari 5 bab dimana sistematika penulisan dalam skripsi ini menggunakan urutan sebagai berikut:

BAB I PENDAHULUAN

Membahas tentang latar belakang pemilihan tema, perumusan masalah, tujuan, batasan masalah dan sistematika penulisan.

BAB II LANDASAN TEORI

Membahas tentang teori *cookies*, aplikasi *web*, dan *Firefox Extension*. Dan juga dibahas mengenai perintah-perintah yang diperlukan.

BAB III PERANCANGAN COOKIESLIVE

Membahas arsitektur *CookiesLive*, dan perancangan *CookiesLive System*. Adapun *CookiesLive System* terdiri dari *CookiesLive Extension*, dan *CookiesLive Website*.

BAB IV ANALISIS DAN UJICOBA

Membahas mengenai hasil pengujian, evaluasi unjuk kerja, dan analisis dari *CookiesLive* yang dijalankan di internet.

BAB V KESIMPULAN

BAB II

COOKIES, *FIREFOX EXTENSION*, DAN APLIKASI BERBASIS *WEB*

2.1. COOKIES

HTTP cookies atau *cookies* untuk singkatnya, digunakan oleh *Web Server* untuk membedakan *users* dan untuk menyimpan data yang berhubungan dengan perilaku *users* dalam menjelajahi Internet. *Cookies* pertama kali digunakan sebagai keranjang belanja *virtual* dimana *users* bisa menyimpan barang ke dalamnya untuk dibeli, sehingga *users* dapat melihat-lihat lagi dan memasukkan benda yang ingin ditambahkan atau menukarnya[1]. Dan setelah selesai *users* bisa membayar semua barang belanjaannya.

Membolehkan *users* untuk *log in* ke sebuah *website* adalah kegunaan lain dari *cookies*. *Users* biasanya memasukkan *username* dan *password* ke dalam sebuah halaman *login*. *Cookies* membolehkan *server* untuk memberitahu *server* bahwa *users* ini telah diautentifikasi dan diperbolehkan untuk menggunakan layanan atau menjalankan fungsi yang hanya diperbolehkan untuk *users* yang telah login.

Cookies digunakan juga untuk mengikuti (*tracking*) *users* selama menjelajahi sebuah *website*. Ini dilakukan untuk menghasilkan statistik dari perilaku *users* sehingga perusahaan iklan dapat menargetkan iklan yang sesuai dengan profil *users*.

Jika terdapat lebih dari satu *browser* dalam sebuah komputer, masing-masing mempunyai tempat penyimpanan *cookies* yang berbeda. Selain itu *cookies* tidak mengenali orang tetapi kombinasi dari *users account* dan *web browser*. Karenanya siapapun yang menggunakan lebih dari satu *account* memiliki beberapa *cookies*. Kebanyakan *browser* akan memperbolehkan *users* untuk menerima *cookies* atau tidak, tapi jika *cookies* tidak diaktifkan maka beberapa fungsi dari *website* tidak akan berjalan.

Terkadang terjadi salah pengertian, banyak yang menggangap bahwa *cookies* adalah program komputer. *Cookies* bukanlah *spyware* maupun *virus*

walaupun *cookies* dari beberapa situs dianggap sebagai *spyware* oleh beberapa produk anti-*spyware*. Faktanya, *cookies* adalah seonggok data sederhana yang tidak dapat melakukan apapun dengan sendirinya.

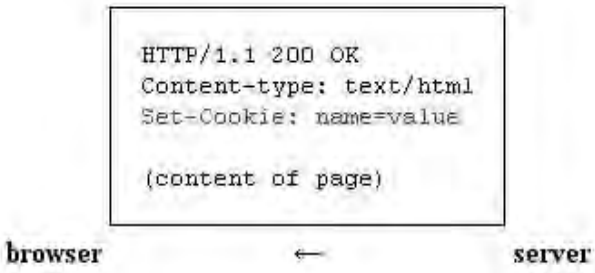
Selain itu, *cookies* pun memiliki *expiration date* yang telah ditentukan oleh *server* dimana *cookies* tersebut sudah tidak aktif lagi dikarenakan alasan tertentu.

Sebuah *browser* diharapkan dapat menyimpan setidaknya 300 *cookies* dengan ukuran 4kb masing-masingnya dan setidaknya 20 *cookies* setiap *server* atau *domain[2]*. Sementara itu untuk Firefox 1.5 maupun 2.0 dapat menyimpan hingga 50 setiap *server*-nya.

Saat kita mengetik www.ee.ui.ac.id di *browser*, maka *browser* akan meminta sebuah halaman dari *web server* dengan mengirimkan teks singkat yang disebut *HTTP request* yang terlihat seperti di Gambar 2.1.

Gambar 2.1. *Browser* melakukan HTTP *request.*

Server membalas dengan mengirimkan halaman yang diminta dan diikuti dengan sebuah paket teks (*HTTP header*) seperti yang ditunjukkan oleh gambar 2.2. Paket ini biasanya berisi permintaan kepada *browser* untuk menyimpan *cookies*.



Gambar 2.2. Server memberikan *cookies.*

Perintah *set-cookies* hanya akan dikirimkan jika *server* memiliki *cookies* untuk disimpan di *browser*. Ini adalah *request* untuk menyimpan *string "name=value"* dan mengirimkannya kembali dalam setiap *request* menuju *server*.

Jika *browser* memperbolehkan *cookies* maka setiap halaman yang diminta dari *server* yang sama akan mengandung *cookies* dengan string *"name=value"*, seperti pada gambar 2.3.

Gambar 2.3. *request* html dari *browser.*

Ini adalah *request* untuk meminta halaman spec.html dalam *server* yang sama, dan berbeda dengan sebelumnya karena ini mengandung *string* "*name=value*" yang telah dikirim. Dengan cara ini, *server* mengetahui bahwa *request* ini berhubungan dengan *request* sebelumnya. *Server* menjawab dengan mengirimkan halaman yang diminta dan mungkin menambah *cookies* yang lain juga.

Cookies diidentifikasi oleh *browser* dengan kombinasi nama, *domain* dan *path*. Dengan kata lain, nama yang sama tetapi *domain* atau *path* yang berbeda akan dikenali sebagai *cookies* yang berbeda. Sehingga nilai *cookies* akan berubah jika nilai yang baru diberikan untuk nama, *domain* dan *path* yang sama.

2.2. APLIKASI BERBASIS *WEB*

Dalam aplikasi berbasis *web* yang umumnya dipakai adalah bahasa *server-side* seperti PHP, ASP, dan lain sebagainya. Pada *CookiesLive*, sistem yang akan dibangun menggunakan PHP. Sedangkan untuk *database* akan digunakan MySQL.

2.2.1. PHP dan MySQL

Sampai saat ini PHP sudah mencapai versi 5, begitu juga halnya dengan MySQL. Ada banyak kelebihan *CookiesLive* dibanding bahasa *server-side scripting* yang lain. Diantaranya adalah:

1. Gratis,
2. Mudah,

3. Dapat ditanamkan HTML,
4. Kompabilitas lintas *platform*,
5. Stabil,
6. Cepat
7. *Open source*,
8. Populer, dan
9. Komunitas pengguna yang kuat.

2.2.2. Menciptakan *Database*

Kita akan menggunakan MySQL sebagai *database*. Berikut ini adalah perintah untuk menciptakan *database* di MySQL:

```
CREATE TABLE table(id serial, field varchar(30));
```

2.2.3. Perintah dalam PHP dan MySQL

CookiesLive akan memperbolehkan *users* untuk menyimpan, me-*load*, mengubah maupun menghapus data. Ada empat perintah yang akan digunakan untuk mendukung sistem dari *CookiesLive*.

- SELECT

  SELECT adalah perintah utama ketika kita ingin membaca data dari *database*. Adapun sintaks dari perintah *SELECT* adalah berikut:

  ```
  SELECT field1, field2, field3
  FROM table
  WHERE condition;
  ```

- INSERT

  INSERT adalah perintah yang digunakan jika kita ingin menulis data ke *database*. Adapun sintaks dari perintah *INSERT* adalah sebagai berikut:

  ```
  INSERT INTO table (col1, col2, col3)
  VALUES(val1, val2, val3);
  ```

- UPDATE

  UPDATE digunakan untuk merubah data di dalam *database*. Contoh sintaks dari perintah *UPDATE* adalah:

```
UPDATE table
SET field1='val1', field2='val2', field3='val3'
WHERE condition;
```

- DELETE

  DELETE digunakan untuk menghapus data dari *database*. Contoh dari sintaks ini adalah:

```
DELETE datapoint
FROM table
WHERE condition;
```

2.3. *FIREFOX EXTENSION*

Extension atau *add-on* menambah fungsionalitas baru untuk Mozilla Firefox. Mereka dapat menambah banyak hal mulai dari *toolbar* baru sampai sebuah fitur yang benar-benar baru. Dengan adanya *extension* ini, users dapat mengkostumisasi *browser Firefox* sesuai dengan kebutuhannya untuk menjaga aplikasi tetap ringan. Aplikasi bisa tetap ringan karena *users* tidak perlu menyimpan *extension* yang tidak dibutuhkannya.

Extension berbeda dengan *plugin*[3], dimana *plugin* membantu *browser* untuk menampilkan sebuah *content* yang spesifik, seperti memainkan *file multimedia*.

2.3.1. *Services*

CookiesLive menggunakan manipulasi *cookies*, *file*, *windows* dan *clipboard*. Untuk melakukan semua hal tersebut menggunakan *Firefox Extension*, kita memerlukan *service* khusus dari *Firefox*. Untuk mendapatkan *service* tersebut kita haruslah menggunakan *interface*. Berikut adalah *interface* yang digunakan di *CookiesLive*.

- nsICookieManager

  Ini merupakan *interface* untuk mengakses dan menghilangkan *cookies* yang berada di daftar *cookies*. Metode-metode yang terdapat di `nsICookieManager` adalah:

  - `remove (domain, name, path, blocked)`

Digunakan untuk menghilangkan *cookies* tertentu di Mozilla Firefox pengguna.

- `removeAll ( )`
  Digunakan untuk menghilangkan semua *cookies* yang terdapat di Mozilla Firefox pengguna.

➢ nsICookieService

Memberikan metode untuk menulis dan membaca *cookies*. *Interface* ini tidak memanipulasi *database cookies* secara langsung. *Interface* ini digunakan sebagai sebuah *service*. Metode-metode yang terdapat di *Interface* ini adalah:

- `getCookieString (URI, channel )`
  Digunakan untuk mendapatkan *cookies* dari *Mozilla Firefox*.
- `getCookieStringFromHttp (URI, firstURI, channel )`
  Digunakan untuk mendapatkan *cookies* dari *Mozilla Firefox* dari *HTTP* tertentu.
- `setCookieString (URI, prompt, cookie, channel )`
  Digunakan untuk menuliskan *cookies* ke Mozilla Firefox.
- `setCookieStringFromHttp (URI, firstURI, prompt, cookie, serverTime, channel)`
  Digunakan untuk mendapatkan *cookies* ke *Mozilla Firefox* dari *HTTP* tertentu.

➢ nsIURI

URI sangat penting untuk strukturisasi nama untuk banyak hal, terutama ketika kita menuliskan *cookies* ke *Mozilla Firefox*. *Interface* ini memberikan akses untuk menuliskan dan membaca komponen-komponen dasar dari sebuah URI. Adapun metode-metode yang terdapat di `nsIURI` adalah:

- `clone ( )`
  Menggandakan URI.
- `equals (other)`
  Test kesamaan dari URI.

- `resolve (relativePath)`
  Melakukan perubahan sebuah string ke dalam absolute URI.
- `schemeIs (scheme)`
  Optimisasi untuk melakukan cek skema tanpa proses GerScheme.

➢ nsIWindowMediator

Interface yang digunakan untuk manipulasi *windows* dan *browser* di *Firefox*. Berikut adalah beberapa metode yang terdapat di `interface` ini:

- `getMostRecentWindow (windowType )`
  Menjemput *window* yang berada di urutan pertama.
- `unregisterWindow (window )`
  Menghapus *window* dari daftar.

➢ nsIProperties

Interface yang digunakan untuk mendapatkan *property* dari sebuah file. Satu-satunya metode di *interface* ini adalah:

- `get (prop, iid, result)`
  digunakan untuk membaca *property*.

➢ nsIFile

Ini adalah satu-satunya cara yang benar untuk menspesifikasikan sebuah *file* untuk lintas *platform*. *Strings* yang umum digunakan tidaklah *valid*. Sebenarnya ada banyak metode untuk *interface* ini, tetapi yang umum dipakai yaitu:

- *`append (node)`*
  Digunakan untuk membangun kelanjutan dari nsIFile yang ada.

➢ nsIFileInputStream

Input *streaming* yang memperbolehkan untuk membaca dari sebuah *file*. Satu-satunya metode di `nsIFileInputStream`:

- `init (file, ioFlags, perm, behaviorFlags)`
  Membaca *file* dari yang ditunjukkan oleh nsIFile.

- nsIScriptableInputStream

nsIScriptableInputStream memberikan akses menuju `nsIInputStream`.

- `available ( )`
  Mengembalikan jumlah *byte* yang tersedia di *streaming*.
- `close ( )`
  Menutup *streaming.*
- `init (inputStream)`
  Membungkus nsIInputStream dengan `nsIScriptableInputStream`.
- `read (count)`
  Membaca data dari *streaming*.

- nsIClipboardHelper

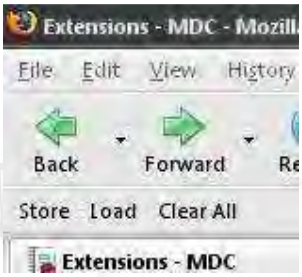
Service yang digunakan untuk penggunaan `nsIClipboard`. Dibawah ini adalah semua metode yang terdapat di *interface* ini:

- `copyString (string )`
  Melakukan *copy* dari *string* ke *clipboard.*
- `copyStringToClipboard (string, clipboardID )`
  Melakukan *copy* dari string ke *clipboard* dengan *ID* tertentu.

2.3.2. *Graphic User Interface*

GUI di *Firefox Extension* menggunakan bahasa XUL. XUL yang merupakan singkatan dari XML *User-interface Languange* diciptakan khusus untuk membuat produk-produk Mozilla lebih cepat, mudah dan bisa digunakan di berbagai *platform*. Sintaks dari XUL sama dengan XML, sedangkan beberapa fungsi yang terdapat di XUL tidak bisa digunakan di XML.

Gambar 2.4, 2.5, dan 2.6 adalah *output* GUI yang diharapkan. Pada gambar ini dapat dilihat *element-element* apa saja yang dibutuhkan untuk menciptakan GUI ini.



Gambar 2.4. *Toolbar* dari *CookiesLive*.

Gambar 2.5. *window load* dari *CookiesLive*.

Gambar 2.6. *windows Clear All* dari *CookiesLive*.

- *Toolbar*

Sebuah wadah yang biasanya mengandung beberapa tombol. Ini adalah tipe kotak yang orientasi defaultnya adalah horizontal. Contoh dari pemakaian *toolbar* adalah:

```
<toolbar id="toolbar-id">
    ..
</toolbar>
```

- *Window*

Menggambarkan struktur dasar dari sebuah tampilan. Ini adalah akar dari sebuah document XUL. Seperti halnya dengan *toolbar*, *window* juga memiliki orientasi secara horizontal. Contoh sintak dari pemakain *element* ini adalah:

```
<window
      id="id-window"

xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.
      only.xul">
 ...
</window>
```

Attribute `xmlns` memberitahu mozilla bahwa ini adalah *namespace* untuk XUL. URL ini tidak benar-benar diunduh, *Mozilla* akan mengenali URL ini secara internal.

- *Button*

Element ini menciptakan sebuah tombol yang dapat ditekan oleh *users*. Pengendali *event* dapat digunakan untuk memberikan perintah. *Events* yang bisa ditangkap mencakup aktifitas *mouse, keyboard* maupun *events* yang lain. Tombol yang diciptakan umumnya adalah tombol abu-abu dengan bentuk persegi. Kita bisa memberikan label maupun gambar di tombol ini. Sintaks yang benar dari penggunaan *element* ini adalah:

```
<button
    id="button-id"
    label="Label"
    image="images/image.jpg"
    ..
    />
```

Jika kita ingin meletakkan gambar diatas tombol, maka kita bisa memasukkan atribut `image` ke dalam *tag*.

- *Label*

Element ini digunakan untuk menampilkan sebuah teks di tampilan. *Element* ini bisa diberikan sebuah perintah, sebagai contoh jika *users* menekan *label*, maka tampilan akan membuka halaman lain. Contoh sederhana dari penggunaan *element* ini adalah:

```
<label value="CookiesLive Text"/>
```

- *Toolbarbutton*

Sebuah tombol akan muncul di *toolbar* jika kita menggunakan *element* ini. Pada dasarnya *element* ini sama saja dengan tombol biasa, hanya saja *element* ini di-*render* secara berbeda. Sama juga seperti halnya tombol biasa, kita bisa meletakkan gambar diatasnya. Contoh dari penggunaan *element* ini adalah:

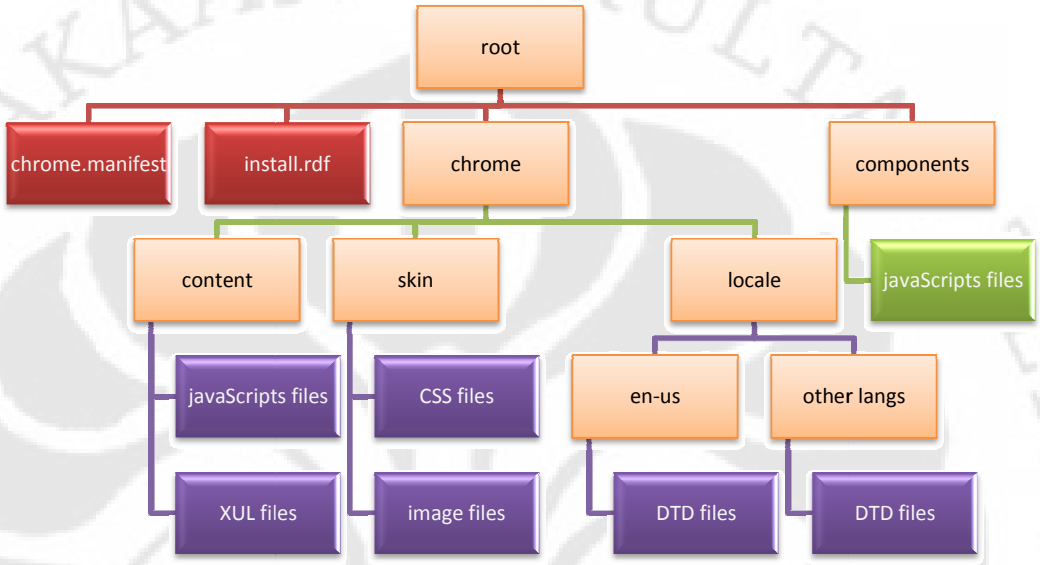
```
<toolbarbutton label="Forward"/>
```

2.3.3. *Packaging Firefox Extension*

Selanjutnya sebagai langkah awal dari pembuatan *Mozilla Firefox*, kita harus melakukan *Packaging* untuk menjadikannya *file* xpi. Dalam *Packaging*, hierarki sangatlah penting. Hierarki akan sebuah *Firefox Extension* digambarkan di *chrome.manifest*. Nantinya dalam melakukan *packaging*, ada beberapa sekuens yang harus dijalankan, yang akan merubah hierarki itu sendiri. Proses pembentukan xpi digambarkan oleh Gambar 2.7.

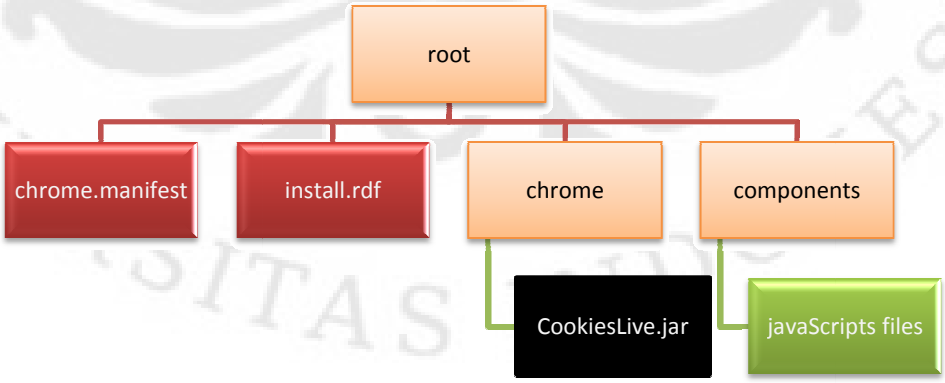
Gambar 2.7. Sekuens penciptaan xpi

Gambar 2.8 menunjukkan struktur yang ideal akan sebuah *Firefox Extension*. *Folder locale*, *components*, dan *skin* merupakan *optional*. Pada *CookiesLive*, ketiga *folder* ini tidak digunakan.



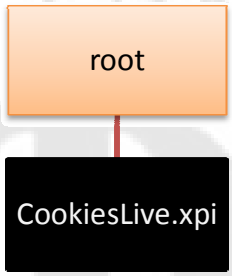
Gambar 2.8. Struktur awal dari sebuah *Extension*.

Selanjutnya kita akan melakukan penciptaan *file* jar. *File* .jar merupakan gabungan semua *file* maupun *folder* yang terdapat di *folder chrome*. Struktur baru dari *Firefox Extension* akan menjadi seperti pada Gambar 2.9.



Gambar 2.9. Tercipta *file* jar di dalam *folder chrome*.

Setelah kita dapatkan *file* jar, kita lakukan proses akhir, yaitu menciptakan xpi. *File* xpi merupakan gabungan dari keseluruhan file yang terdapat di bawah *folder root* termasuk *file* jar. Sehingga struktur yang baru akan menjadi seperti Gambar 2.10.



Gambar 2.10. Proses penciptaan *file* xpi.

Nantinya file *CookiesLive*.xpi ini dapat di-*install* di *Mozilla Firefox*. Yang perlu diingat adalah semua orang bisa melakukan *unpack file* xpi dan jar sehingga siapapun bisa melihat *source code* dari program ini.

BAB III

PERANCANGAN *COOKIESLIVE*

3.1. ARSITEKTUR *COOKIESLIVE*

Sistem *CookiesLive* yang akan dibangun berbasiskan *web* dan *Mozilla Firefox*. Arsitektur yang dibutuhkan untuk membangun sistem ini ditunjukkan oleh Gambar 3.1

Gambar 3.1 Arsitektur *CookiesLive System.*

3.1.1. *Browser*

Browser yang bisa digunakan adalah *Mozilla Firefox* mulai dari versi 1.5 sampai dengan versi 2.0. Alasan pemilihan dari *Mozilla Firefox* adalah jumlah pengguna dari browser ini meningkat pesat[4]. *Mozilla Firefox* menjadi *browser* terbesar kedua setelah *Microsoft™ Internet Explorer*. Dari sisi *development* sangat mudah bagi penulis untuk membuat *program* untuk *Mozilla Firefox* karena komunitas dari *developer Mozilla* sendiri sudah sangat besar. Selain itu *Mozilla Firefox* adalah *free* dan *open-source* [5].

3.1.2. *Server*

Server akan melayani permintaan user untuk *Storage* dari *Cookies*. Spesifikasi *server* yang dibutuhkan adalah minimal mendukung PHP versi 5 dan MySQL versi 5.

3.2. *COOKIESLIVE SYSTEM*

Sistem *CookiesLive* dibagi menjadi dua, yaitu *CookiesLive Extension* dan *CookiesLive Website* seperti yang digambarkan oleh Gambar 3.2. Keduanya dipisahkan agar menjamin keamanan bagi pengguna *CookiesLive*. *JavaScript* adalah program *Client-side* sehingga tidak aman untuk melakukan semua proses penyimpanan *database* di *Firefox* sehingga diperlukan bahasa *server-side*. Adapun bahasa *server* yang digunakan adalah PHP dan *database* yang digunakan adalah MySQL.

Gambar 3.2 *CookiesLive System.*

3.2.1 *CookiesLive Extension*

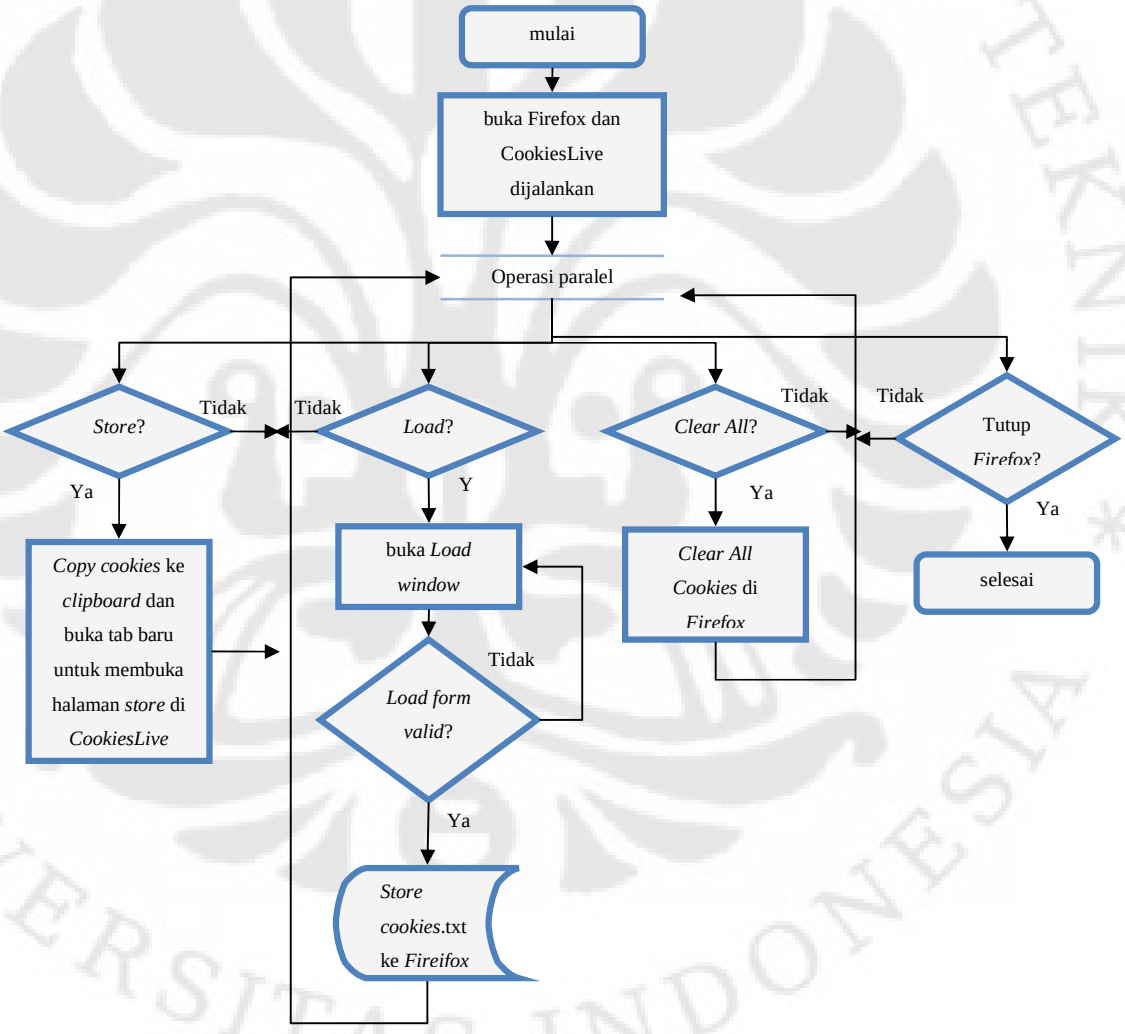
CookiesLive Extension merupakan sebuah *Firefox Extension* yang harus dipasang oleh *users* di *browser Firefox*. *CookiesLive Extension* dibutuhkan untuk dipasang di browser, karena dengan adanya program yang terpasang dengan *browser*, memudahkan *CookiesLive* untuk melakukan modifikasi terhadap *cookies*.

Diagram alir pada Gambar 3.3 menunjukkan proses yang dilakukan oleh *user*. Semua proses ini dilakukan dengan bahasa JavaScripts, CSS dan XUL. Bahasa yang digunakan untuk *Firefox Extension*. Penjelasan dari diagram alir ini adalah:

1. Ketika *CookiesLive* sudah terpasang maka *Firefox* otomatis akan me-*Load CookiesLive*. Dan akan muncul *Toolbar* untuk menjalankan semua fungsi dari *CookiesLive*.
2. Jika *user* ingin melakukan *store* maka *CookiesLive* akan melakukan *copy* isi dari *cookies* ke *clipboard*. Dan akan dibuka

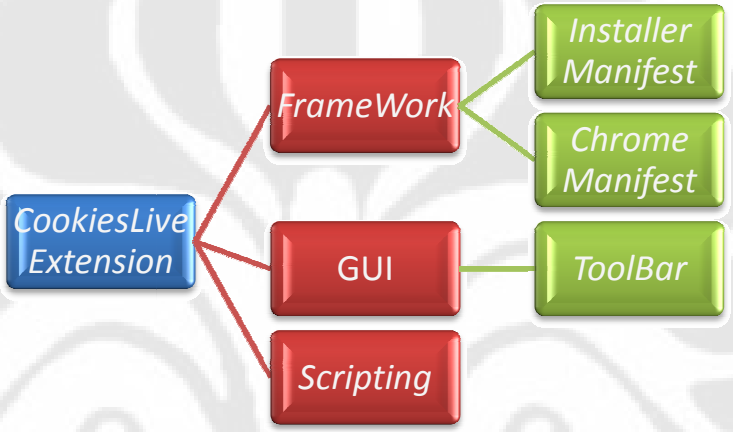
halaman *store* dari *CookiesLive* sehingga *user* dapat langsung mengisi *form*.

3. Jika *user* ingin melakukan *load* maka *CookiesLive* akan membuka *window load* yang dengan *form* pengisian untuk diisi *cookies* yang dapat diambil dari *CookiesLive*.
4. Jika *user* ingin melakukan *ClearAll* maka *CookiesLive* akan menghapus semua *cookies* yang terdapat di *Firefox*.



Gambar 3.3. Diagram Alir Algoritma *CookiesLive Firefox Extension.*

Gambar 3.4 menunjukkan rancang bangun dari *CookiesLive Extension*, yang terdiri dari beberapa *folder* dan *file*. *Framework* sebuah *extension* digunakan untuk memberitahu kepada *Firefox* tentang bagaimana *file*-nya terstruktur, siapa yang menciptakan, GUID-nya, dan lain sebagainya. Sejak *Firefox* 1.5, pengembangan pada sisi ini berubah pesat[6]. Mekanisme baru dalam menciptakan *framework* ini jauh lebih sederhana dibandingkan mekanisme yang digunakan untuk Firefox 1.0.x.



Gambar 3.4. Rancang Bangun *CookiesLive Extension.*

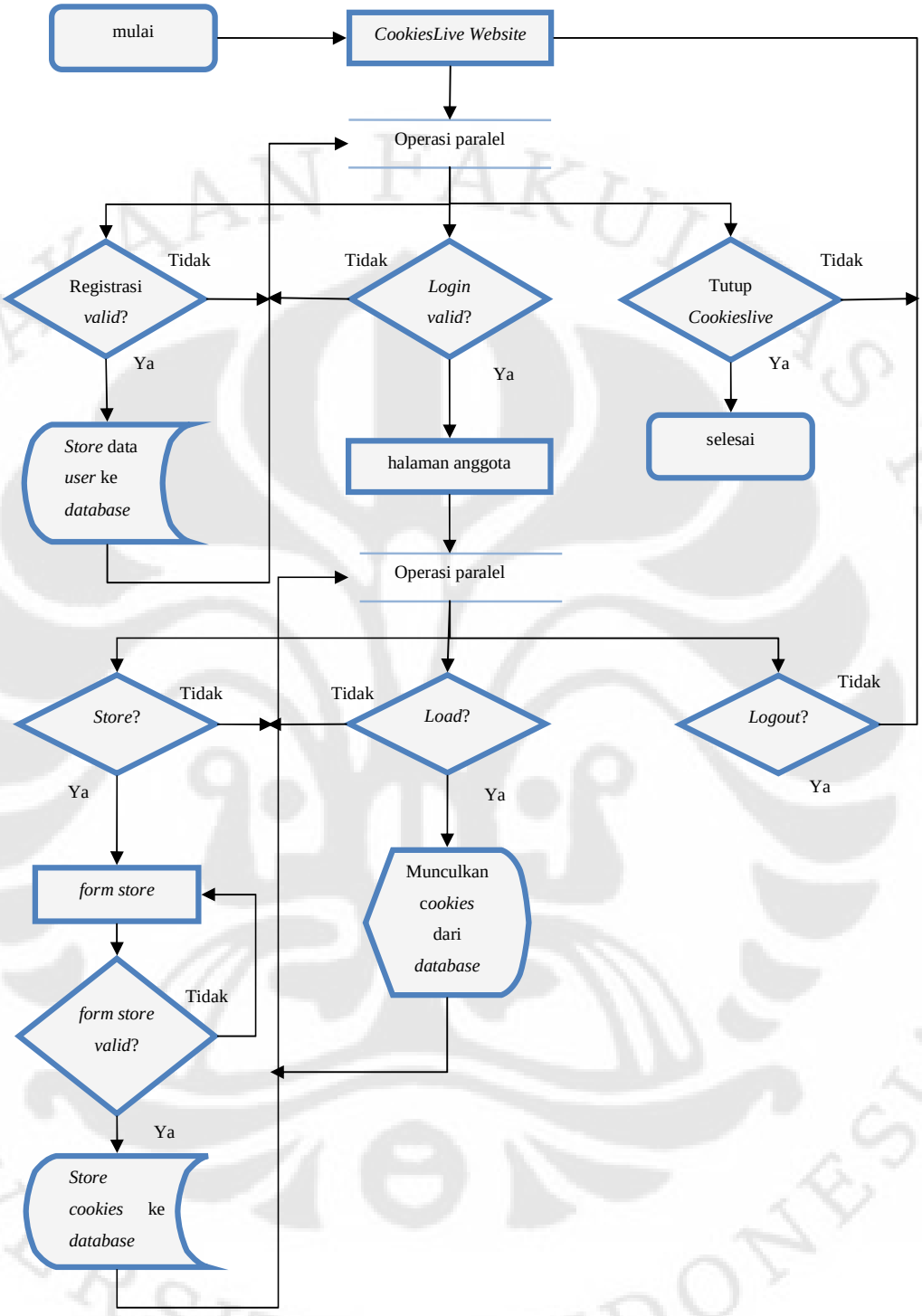
Installer manifest adalah bagaimana kita memberikan detail kepada extension untuk *Firefox*. Detail yang dimaksud adalah apa nama *Firefox Extension*, *author*, kompabilitas, dan lain-lain. *Chrome manifest* adalah bagaimana memberi tahu Firefox akan *content, locale, skin* dan *overlay* apa yang digunakan oleh CookiesLive. *Content* memberitahu Firefox hierarki dari CookiesLive. Locale untuk memberikan fitur alih bahasa, sementara ini di CookiesLive belum dipakai. Overlay untuk memberitahu *Firefox Extension* dimana letak GUI dari *CookiesLive*. *Skin* untuk mempercantik GUI dengan CSS. Untuk CookiesLive tidak digunakan CSS untuk mempercepat proses.

3.2.2. *CookiesLive Website*

CookiesLive Website kita gunakan untuk menyimpan *cookies* ke dalam *server* juga untuk mengambil *cookies* dari *server*. Suatu proses yang tidak bisa dilakukan dengan JavaScripts.

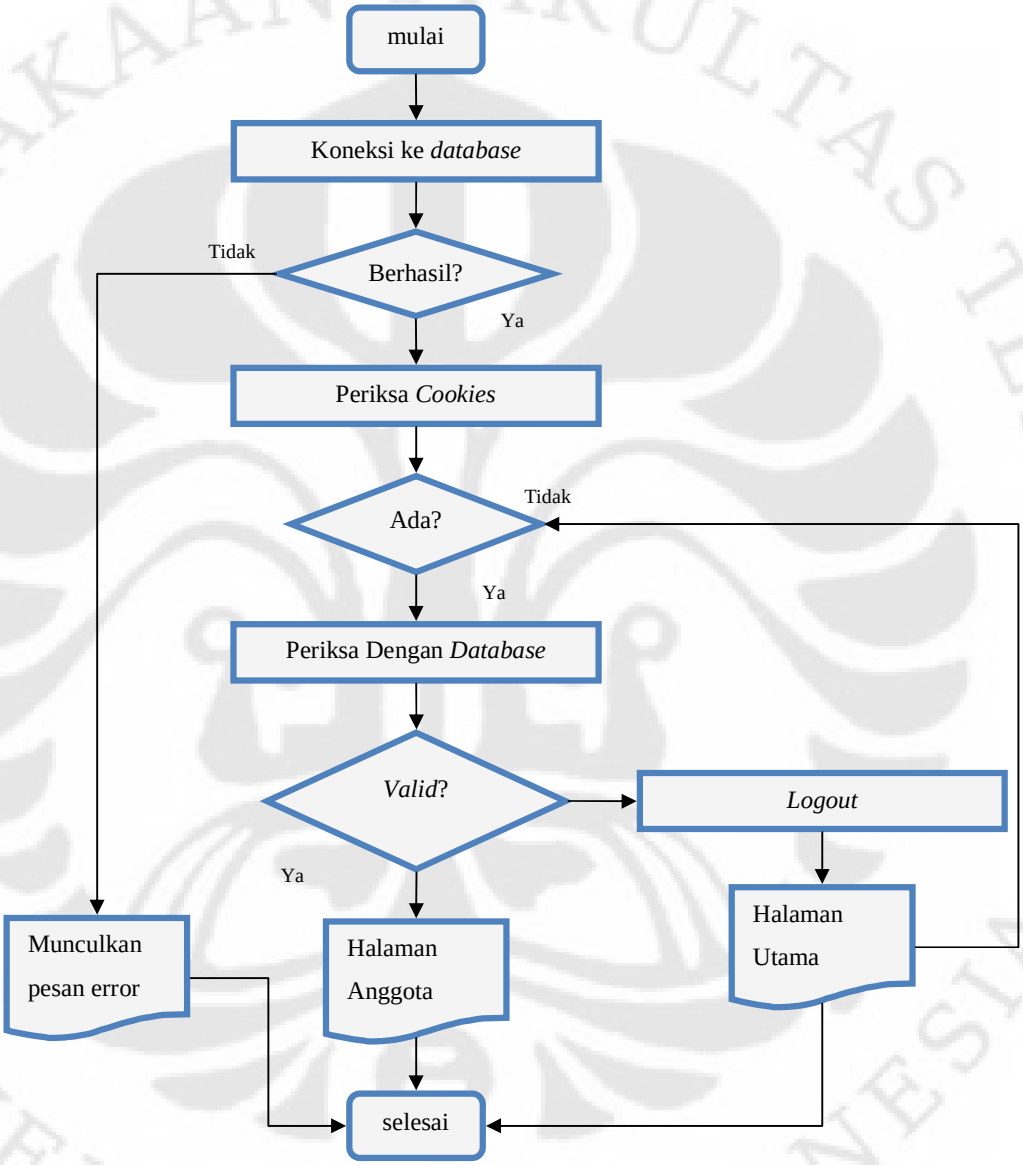
Gambar 3.5 adalah diagram alir untuk proses di *CookiesLive Website*. Digunakan bahasa HTML, CSS, PHP dan MySQL. Adapun penjelasan dari diagram alir ini adalah:

1. Ketika *browser* melakukan *request* menuju *website CookiesLive*, *server* akan mengirimkan halaman utama.
2. Di halaman utama user bisa melakukan registrasi. Jika registrasi *valid* maka *server* akan melakukan *storage* dari data *user*.
3. Selain itu jika *user* sudah melakukan registrasi maka user dapat melakukan *login*. Jika *login valid* maka *server* akan membawa *user* ke halaman anggota.
4. Di halaman anggota jika *user* sudah melakukan proses *login* dengan *valid* maka *user* dapat melakukan *store*. Proses ini akan menampilkan form penyimpanan sehingga user dapat melakukan *paste* dari *clipboard*-nya
5. Jika *user* ingin melakukan *load* maka *server* akan mengambil *cookies* dari *server* dan menampilkannya di *browser* untuk di *load* di *Firefox*.
6. Jika *user* melakukan *logout* maka akan keluar dari halaman anggota dan akan dibawa kembali kehalaman utama.



Gambar 3.5 Diagram Alir Algoritma *CookiesLive Website*

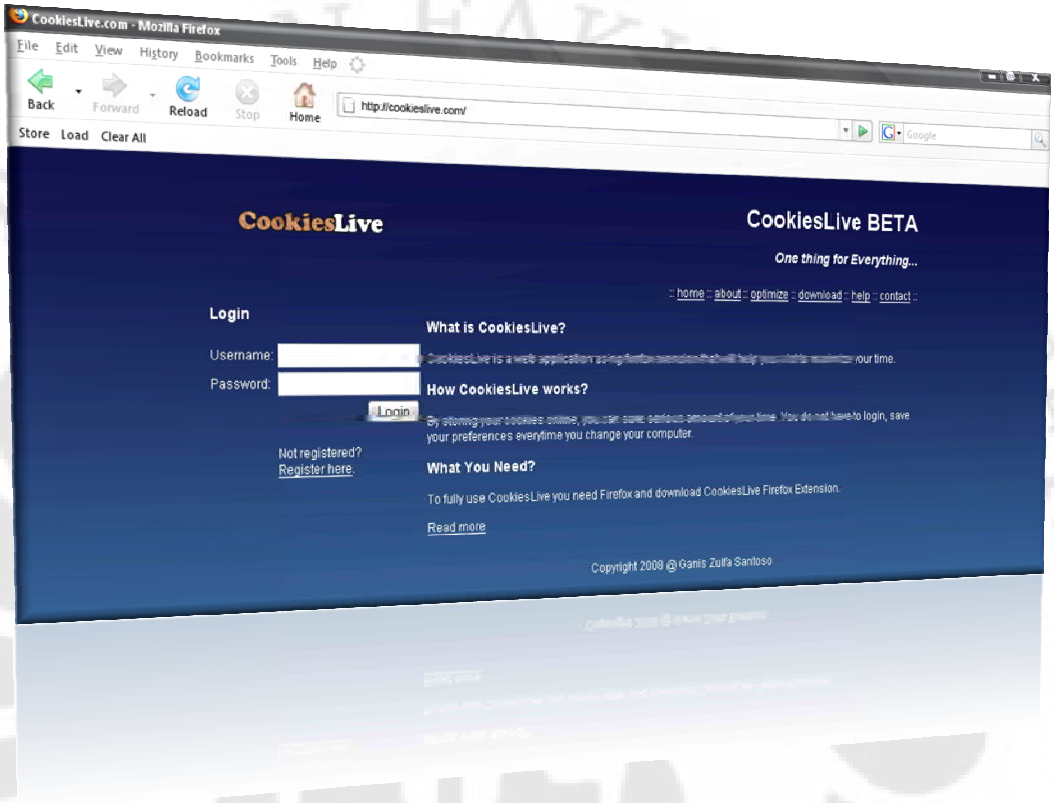
Semua halaman di *CookiesLive Website* mengikuti logika yang ditunjukkan oleh Gambar 3.6. Ini dilakukan agar menghemat waktu dari *users* selama membuka *CookiesLive*. Sehingga jika *users* sudah melakukan *login*, maka user tidak perlu lagi *login* jika *cookies* masih ada di *browser*.



Gambar 3.6 Diagram Alir Algoritma Halaman di *CookiesLive Website.*

Gambar 3.7 adalah desain akhir dari *CookiesLive Website*. Desain akan menggunakan HTML dan CSS. Pada desain ini diutamakan pada *bandwidth* yang

ringan sehingga teknologi yang umumnya membuat *browser* berkerja berat seperti *Flash* tidaklah dipakai.



Gambar 3.7 Desain akhir *CookiesLive Website*.

BAB IV

ANALISIS DAN UJI COBA

Ujicoba pada sistem dari *CookiesLive* ini akan melihat perbandingan signifikansi bagi *user* dalam menggunakan sistem ini baik dilihat dari waktu maupun *bandwidth* yang terpakai jika dibandingkan dengan tidak menggunakan sistem ini. Adapun parameter yang dipakai dalam pengujian ini adalah *bandwidth* dan detik. Pengambilan *bandwidth* dilihat dengan menggunakan *WireShark* dengan melihat berapa banyak *data* yang melintas. Pengambilan parameter detik dilihat dari waktu yang terpakai juga dengan menggunakan *Wireshark*.

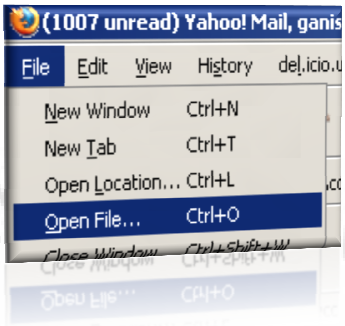
4.1. SKENARIO UJICOBA

Untuk menguji sistem *CookiesLive*, penulis menggunakan komputer dengan spesifikasi Intel® Core™ 2 Duo CPU E4500 @ 2.20GHz 1.18GHz, Memory RAM 0.99 GB menggunakan Microsoft Windows XP Professional Version 2002 Service Pack 2. *Browser* yang digunakan adalah *Mozilla Firefox* versi 2.0.0.12. Akses internet menggunakan *Telkomnet Instant.*

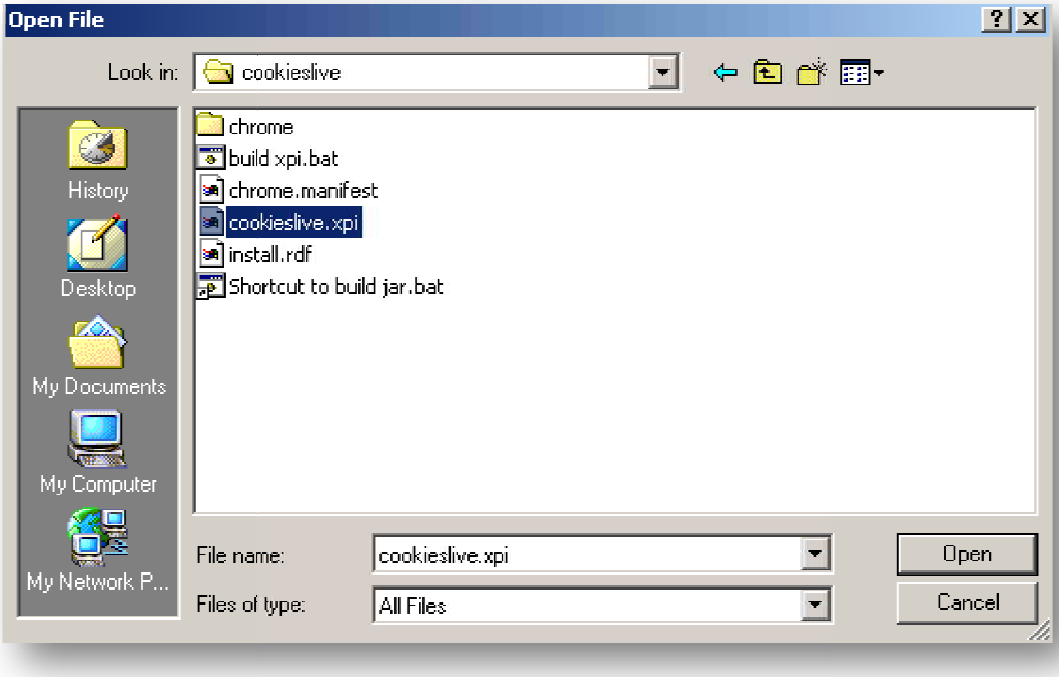
4.1.1. Instalasi CookiesLive

Proses instalasi *CookiesLive Firefox Extension* dapat dilakukan dengan menggunduh *file* instalasinya di http://cookieslive.com/cookieslive.xpi secara gratis. *File* instalasi ini sebesar 6kb. Langkah instalasi yang harus dilakukan setelah menyimpan *file* instalasi di hard disk lokal adalah dengan melakukan hal-hal berikut:

1. Membuka *file* tersebut dengan *Mozilla Firefox* seperti pada Gambar 4.1. dan Gambar 4.2.

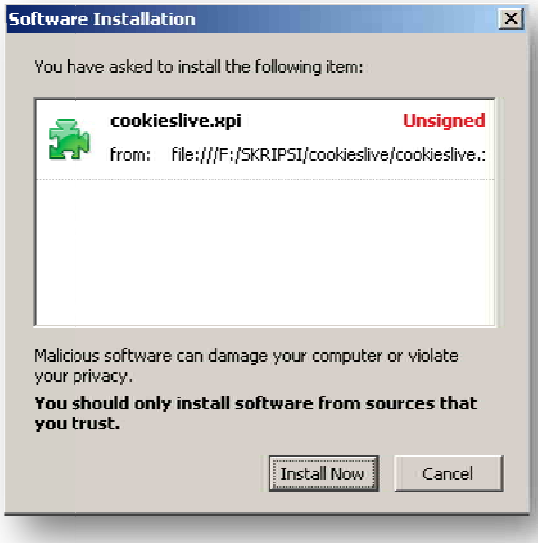


Gambar 4.1. *Open File* di *Browser Firefox*



Gambar 4.2 Seleksi *file cookieslive*.xpi

2. *Firefox* telah mengunduh *file cookieslive*.xpi secara sempurna. Klik *Install Now* seperti pada Gambar 4.3..



Gambar 4.3 Instalasi *software*.

3. *Restart Firefox* untuk menyelesaikan instalasi.

4. Apabila muncul *toolbar* baru seperti yang ditunjukkan oleh Gambar 4.4, maka *CookiesLive* telah terpasang secara sempurna.

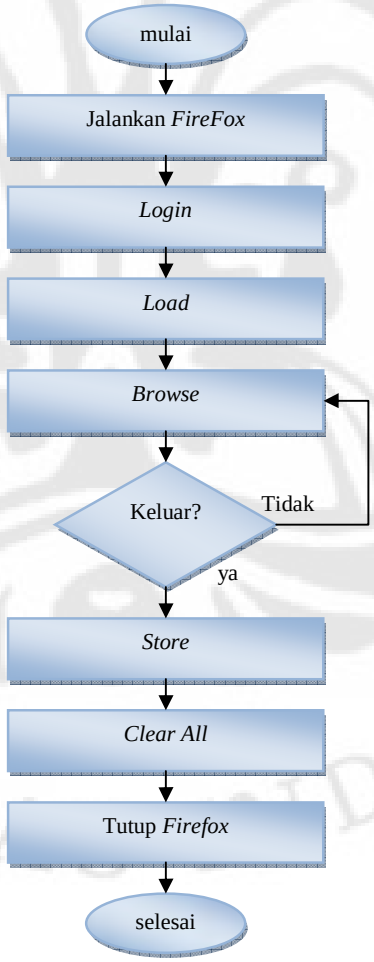
Gambar 4.4. *CookiesLive Extension*

4.1.2. Langkah Optimal dari Penggunaan *CookiesLive*

Gambar 4.5 menggambarkan langkah optimal yang dapat dilakukan dalam penggunaan *CookiesLive*. Berikut adalah penjelasan langkah yang harus dijalankan *user* agar pemakaian dari *CookiesLive* bisa efisien dan aman.

1. Pertama-tama ketika *user* membuka *Firefox* maka hal pertama yang harus dilakukan adalah membuka *CookiesLive Website* dan *login*.

2. Ketika sudah login maka hal yang harus dilakukan *users* adalah me-*load cookies* dari *CookiesLive website database* dan memasukkannya ke *Mozilla Firefox* menggunakan *CookiesLive Extension*.
3. Karena *cookies* sudah dimasukkan ke *Firefox*, maka *user* dapat melanjutkannya dengan melakukan *browsing*, *chatting* ataupun aktifitas internet lainnya.
4. Jika *users* berniat untuk menyelesaikan aktifitas internetnya maka sebelum mematikan *Mozilla Firefox*, *users* harus menyimpan *cookies* miliknya yang baru di *database CookiesLive*.
5. Sebelum menutup *Firefox*, *users* harus menghapus semua *cookies* yang ada di *Mozilla Firefox* dengan perintah *Clear All*. Ini diharuskan sehingga data *users* yang tertinggal di dalam komputer tidak akan digunakan oleh pihak lain.



Gambar 4.5. Diagram alir penggunaan *CookiesLive*.

4.2. UJICOBA SIGNIFIKANSI

Untuk mengetahui kelebihan dari menggunakan sistem ini jika dibandingkan dengan tidak memakai sistem ini, penulis akan melakukan ujicoba dengan menggunakan beberapa *account* yang sudah terdapat di internet. Ujicoba akan mengandaikan seorang *user* melakukan aktifitas di sebuah komputer yang bukan miliknya yang sudah terdapat *browser Mozilla Firefox* yang terdapat *CookiesLive*. Diasumsikan *user* melakukan proses autentifikasi kepada sejumlah *account.* 20 website yang diujicobakan termasuk 200 *website* yang paling sering dikunjungi di dunia[7]. Diasumsikan 20 *website* ini adalah situs yang akan dikunjungi oleh pengguna *CookiesLive.*

Penghitungan *bandwith* dan waktu dilakukan dengan *Wireshark Network Protocol Analyzer* versi 0.99.6a. *WireShark* merupakan *software opensource* dibawah naungan lisensi GNU.

4.2.1. Ujicoba Tanpa *CookiesLive*

Skenario ujicoba ini dilakukan *user* dengan melakukan *login* secara *manual* dengan tanpa kesalahan *password* maupun *username*. Diasumsikan *user* tidak ingat halaman khusus *login* dari sebuah *website* sehingga users melakukan *login* dengan me-*request* halaman utama *website* tersebut. Penghitungan dimulai ketika *browser* mengirim *request* halaman utama ke *server*. Dan penghitungan selesai ketika *users* sudah masuk ke halaman anggota.

Tabel 4.1. menunjukkan bahwa *bandwidth* yang terbuang tidak memiliki hubungan dengan jumlah halaman yang dibuka. Ini dimungkinkan karena *content* dari tiap-tiap *website* berbeda. Walaupun demikian, jumlah waktu yang terpakai mempunyai kecenderungan meningkat bersamaan dengan jumlah halaman yang terbuka. Ini dikarenakan *user* harus meng-*click link login* yang tentunya butuh waktu walaupun disaat itu penghitungan *bandwidth* berhenti.

Tabel 4.1 Proses *login* tanpa *CookiesLive.*

No	*website*	A1	B1 (detik)	C1(*bytes*)
1	www.yahoo.com/	2	19.690	44,632
2	www.youtube.com/	1	24.141	67,170
3	www.live.com/	2	26.021	19,421
4	www.wikipedia.org/	3	28.393	44,452
5	www.myspace.com/	1	45.077	164,160
6	www.facebook.com/	1	32.189	121,489
7	www.gamespot.com/	1	44.713	163,762
8	www.friendster.com/	1	15.095	101,572
9	www.geocities.com/	2	31.500	76,147
10	www.multiply.com/	1	28.842	100,928
11	www.blogger.com/	1	43.625	61,591
12	www.hi5.com/	1	16.719	105,589
13	www.ign.com/	2	54.251	431,391
14	www.megaflirt.com/	2	25.644	50,430
15	www.digg.com/	1	42.984	80,879
16	www.dailymotion.com/	1	46.587	137,865
17	www.deviantart.com/	1	19.373	185,585
18	www.nba.com/	2	55.314	218,313
19	www.badoo.com/	2	43.285	117,443
20	www.gamefaqs.com/	1	47.993	59,960

Ket:

A1 = Halaman yang dibuka sebelum masuk ke halaman anggota.

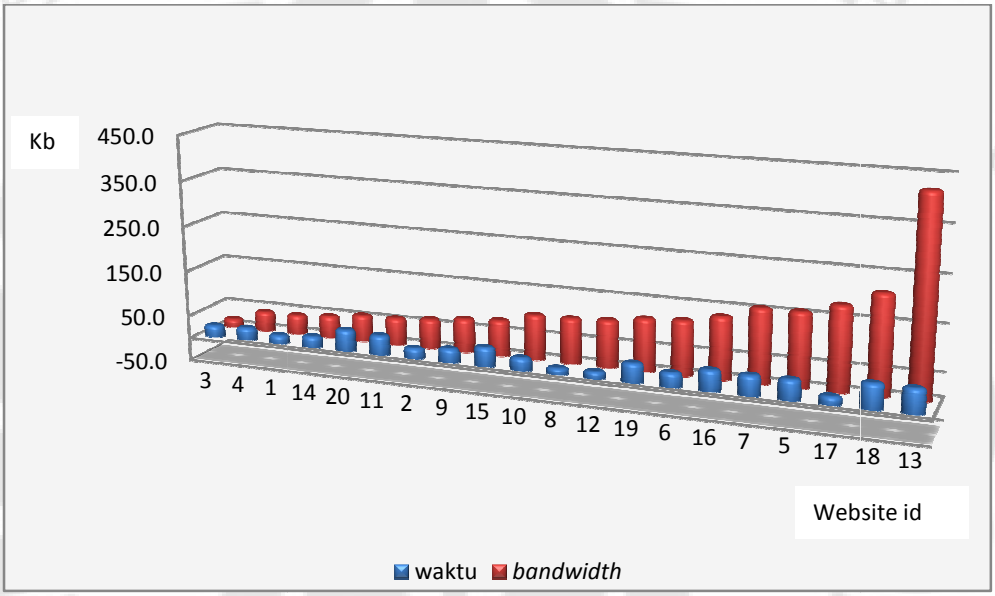
B1 = Waktu yang diperlukan untuk masuk ke halaman anggota.

C1 = *Bandwidth* yang diperlukan untuk masuk ke halaman anggota.

Dari Gambar 4.6 terlihat jelas walaupun *bandwidth* yang diperlukan semakin tinggi tetapi waktu yang terbuang tidak menunjukkan kecenderungan yang sama. Dapat dilihat bahwa tidak ada korelasi antara *bandwidth* yang

terbuang dengan berapa lama waktu yang dibutuhkan untuk masuk ke halaman anggota. Ini dimungkinkan karena beberapa faktor:

1. Kecepatan *server* dari website yang berbeda-beda.
2. Ketidakstabilan dari koneksi internet.
3. Teknologi yang dipakai oleh sebuah *website* membuat *browser* lebih lama melakukan *render*.



Gambar 4.6. Hubungan *bandwith* dan waktu untuk *login* tanpa *CookiesLive*.

4.2.2. Ujicoba Dengan *CookiesLive*

Skenario ujicoba dilakukan dengan terlebih dahulu user telah melakukan autentifikasi dengan melakukan proses load dengan sistem *CookiesLive*. Proses *load* dengan ukuran dari *cookies* ini disimulasikan dengan ukuran maksimal dari sebuah *cookies* untuk 20 *domain*. Ukuran *cookies* maksimal untuk satu *domain* adalah 4 Kb[2], sehingga untuk 20 *domain* kita ambil kemungkinan terburuk yaitu 80 Kb. Proses *load* dihitung sejak *browser* meminta *cookieslive*.com dari *server*. Proses *load* memakan waktu selama 16,474 detik dan *bandwidth* sebanyak 99.726 bytes.

Setelah dilakukan proses *load*, kita hitung waktu dan *bandwidth* yang dibutuhkan untuk masuk ke halaman anggota, dimulai dari halaman yang terdepan. Untuk mensimulasikan proses *login* dengan *browser* yang seolah-olah

baru, *cache* yang terdapat di *firefox* dihapus. Sehingga proses autentifikasi mengambil *file-file* pendukung yang baru dari *server* seperti halnya pada proses *login manual*. Seperti halnya dengan Tabel 4.1, Tabel 4.2 dan Gambar 4.7 menunjukkan bahwa *bandwidth* yang terbuang tidak berhubungan dengan jumlah waktu yang terbuang. Dan juga jumlah waktu yang terpakai mempunyai kecenderungan meningkat bersamaan dengan jumlah halaman yang terbuka.

Tabel 4.2. Proses login dengan *CookiesLive*.

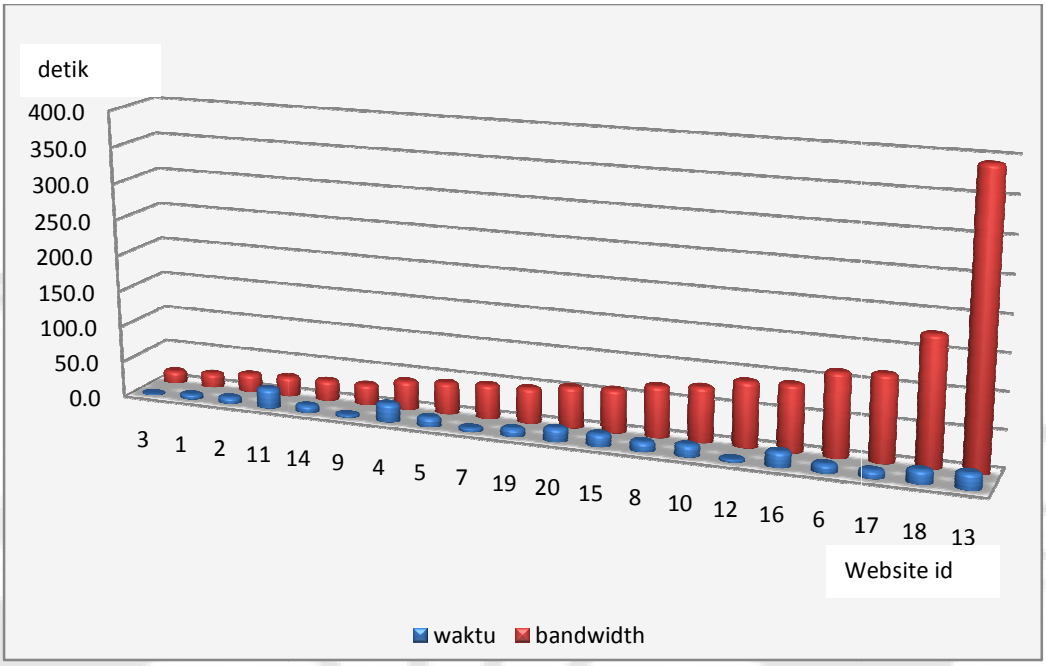
No	websites	A2	B2 (secs)	C2(bytes)
1	www.yahoo.com/	0	7.662	21,204
2	www.youtube.com/	0	9.765	26,523
3	www.live.com/	0	2.209	18,420
4	www.wikipedia.org/	2	26.500	41,561
5	www.myspace.com/	1	14.024	45,464
6	www.facebook.com/	0	12.804	117,523
7	www.gamespot.com/	0	5.933	49,036
8	www.friendster.com/	0	14.513	72,913
9	www.geocities.com/	0	4.775	30.159
10	www.multiply.com/	0	17.154	78,257
11	www.blogger.com/	0	28.187	28,623
12	www.hi5.com/	0	5.051	92,291
13	www.ign.com/	0	22.678	402,944
14	www.megaflirt.com/	0	11.138	29,791
15	www.digg.com/	0	18.103	61,741
16	www.dailymotion.com/	0	22.281	94,724
17	www.deviantart.com/	0	11.857	121,100
18	www.nba.com/	0	19.398	180,129
19	www.badoo.com/	0	12.027	50,837
20	www.gamefaqs.com/	0	19.041	58,793

Ket:

A2 = Halaman yang dibuka sebelum masuk ke halaman anggota.

B2 = Waktu yang diperlukan untuk masuk ke halaman anggota.

C2 = Bandwidth yang diperlukan untuk masuk ke halaman anggota.



Gambar 4.7. Hubungan *bandwidth* dan waktu untuk *login* dengan *CookiesLive*

Faktor-faktor yang sama seperti pada *login* tanpa *CookiesLive* juga mempengaruhi hubungan antara *bandwidth* dengan waktu. Sehingga dapat disimpulkan, penggunaan *CookiesLive* maupun tidak, tidak mempengaruhi hubungan antara *bandwith* dan waktu *login*.

Selanjutnya kita lakukan proses *store* ke *database*. Sesuai dengan proses *load*, kita ujicoba juga dengan ukuran *cookies* 80 Kb. Dari data didapat waktu yang terpakai 7,275 detik dan *bandwidth* sebanyak 113.434 *bytes*.

4.2.3. Parameter Waktu

Pada Tabel 4.3 dibawah ini kita akan membandingkan proses autentifikasi dengan *CookiesLive* maupun dengan tidak menggunakan *CookiesLive*. Tabel 4.3 akan melihat signifikansi dari parameter waktu. Akan dihitung selisih dari keduanya dan peningkatan jika menggunakan *CookiesLive*.

Disini terlihat bahwa dengan menggunakan *CookiesLive* akan menghemat waktu autentifikasi mulai dari 0,6 sampai dengan 35.9 detik hanya untuk proses autentifikasi saja. Dari segi peningkatan waktu, signifikansi dari penggunaan *CookiesLive* mencakup *range* 107.1% – 1178%. Sehingga waktu yang umumnya

dipakai untuk *login* tanpa *CookiesLive* hanya sekali maka dengan *CookiesLive* dapat dilakukan hingga sepuluh kali.

Tabel 4.3 Signifikansi *CookiesLive* dari parameter waktu

No	web	A1	A2	B1	B2	B1 - B2	B1/B2
1	www.yahoo.com/	2	0	19.69	7.662	12.028	257.0%
2	www.youtube.com/	1	0	24.141	9.765	14.376	247.2%
3	www.live.com/	2	0	26.021	2.209	23.812	1178.0%
4	www.wikipedia.org/	3	2	28.393	26.5	1.893	107.1%
5	www.myspace.com/	1	1	45.077	14.024	31.053	321.4%
6	www.facebook.com/	1	0	32.189	12.804	19.385	251.4%
7	www.gamespot.com/	1	0	44.713	5.933	38.78	753.6%
8	www.friendster.com/	1	0	15.095	14.513	0.582	104.0%
9	www.geocities.com/	2	0	31.5	4.775	26.725	659.7%
10	www.multiply.com/	1	0	28.842	17.154	11.688	168.1%
11	www.blogger.com/	1	0	43.625	28.187	15.438	154.8%
12	www.hi5.com/	1	0	16.719	5.051	11.668	331.0%
13	www.ign.com/	2	0	54.251	22.678	31.573	239.2%
14	www.megaflirt.com/	2	0	25.644	11.138	14.506	230.2%
15	www.digg.com/	1	0	42.984	18.103	24.881	237.4%
16	www.dailymotion.com/	1	0	46.587	22.281	24.306	209.1%
17	www.deviantart.com/	1	0	19.373	11.857	7.516	163.4%
18	www.nba.com/	2	0	55.314	19.398	35.916	285.2%
19	www.badoo.com/	2	0	43.285	12.027	31.258	359.9%
20	www.gamefaqs.com/	1	0	47.993	19.041	28.952	252.1%
				Total		406.336	6509.8%
				Rata-rata		20.3168	325.49%

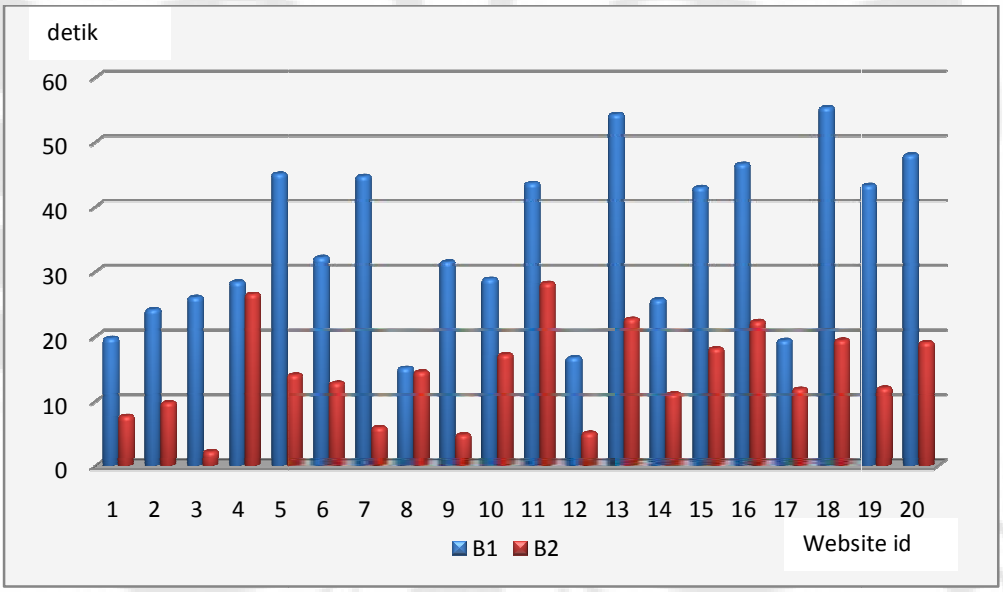
Ket:

B1 - B2 = Selisih waktu.

B1 / B2 = Peningkatan waktu.

Umumnya penghematan banyak terjadi pada *website* yang tidak menyimpan *form login* untuk autentifikasi *users* di halaman utama. Sehingga *users* harus berjalan beberapa halaman untuk bisa masuk ke halaman anggota. Ini membuat *bandwidth* dan waktu yang terbuang lebih banyak daripada *website* yang menyimpan *login form* di halaman utama. Tetapi dengan *CookiesLive* maka proses perjalanan *users* menuju *login form* sebelum menuju halaman anggota dapat diminimalisir karena umumnya *user* langsung dibawa ke halaman anggota.

Penghematan waktu bisa saja jauh lebih tinggi karena disini penulis melakukan autentifikasi dengan tanpa kesalahan *password* maupun *username*. Di keadaan nyata, kejadian-kejadian tersebut merupakan hal yang sering terjadi. Dengan *CookiesLive*, hal ini dapat diminimalisir. Gambar 4.8 menunjukkan perbandingan dari segi waktu. Terlihat dengan CookisLive akan dicapai pengurangan waktu yang tercapai unu semua website.



Gambar 4.8. Perbandingan pemakaian dari parameter waktu.

4.2.4. Parameter *Bandwidth*

Kita melakukan perhitungan dengan parameter bandwidth yang ditunjukkan oleh Tabel 4.4. Disini terlihat bahwa dengan menggunakan *CookiesLive* akan menghemat *bandwidth* mulai dari 1 Kb sampai dengan 118,6 Kb hanya untuk proses autentifikasi saja. Dari segi peningkatan *bandwidth*,

signifikansi dari penggunaan *CookiesLive* mencakup range 102.0% – 334.0%. Sehingga *bandwidth* dengan *CookiesLive* dapat dihemat hingga tiga kali.

Sama seperti parameter detik umumnya penghematan banyak terjadi pada *website* yang tidak menyimpan *form login* untuk autentifikasi *users* di halaman utama.

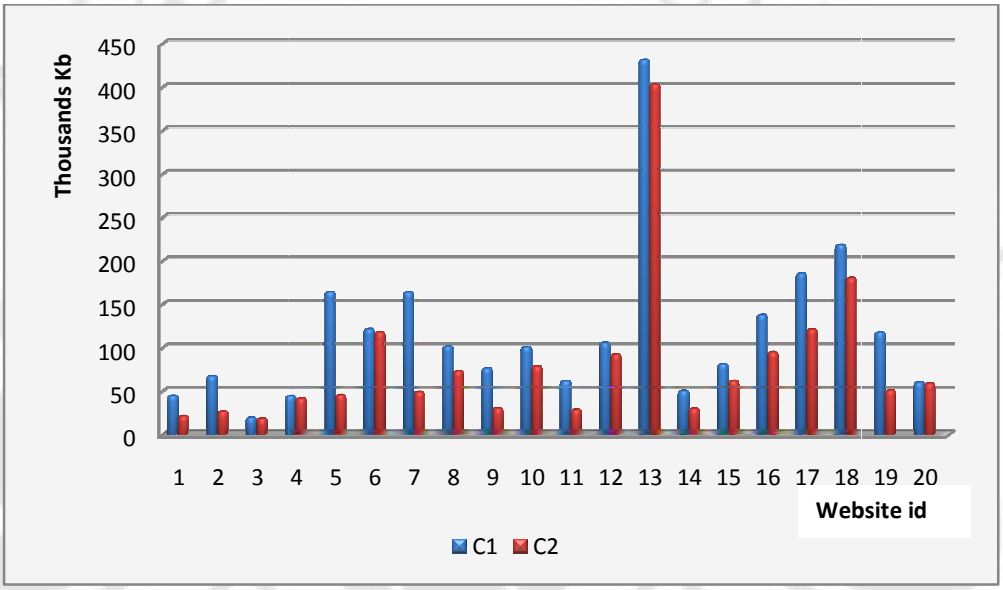
Tabel 4.4 Signifikansi *CookiesLive* dari parameter bandwidth

	web	A1	A2	C1	C2	C1 - C2	C1/C2
1	www.yahoo.com/	2	0	44,632	21,204	23,428	210.5%
2	www.youtube.com/	1	0	67,170	26,523	40,647	253.3%
3	www.live.com/	2	0	19,421	18,420	1,001	105.4%
4	www.wikipedia.org/	3	2	44,452	41,561	2,891	107.0%
5	www.myspace.com/	1	1	164,160	45,464	118,696	361.1%
6	www.facebook.com/	1	0	121,489	117,523	3,966	103.4%
7	www.gamespot.com/	1	0	163,762	49,036	114,726	334.0%
8	www.friendster.com/	1	0	101,572	72,913	28,659	139.3%
9	www.geocities.com/	2	0	76,147	30,159	45,988	252.5%
10	www.multiply.com/	1	0	100,928	78,257	22,671	129.0%
11	www.blogger.com/	1	0	61,591	28,623	32,968	215.2%
12	www.hi5.com/	1	0	105589	92,291	13,298	114.4%
13	www.ign.com/	2	0	431,391	402,944	28,447	107.1%
14	www.megaflirt.com/	2	0	50,430	29,791	20,639	169.3%
15	www.digg.com/	1	0	80,879	61,741	19,138	131.0%
16	www.dailymotion.com/	1	0	137,865	94,724	43,141	145.5%
17	www.deviantart.com/	1	0	185,585	121,100	64,485	153.2%
18	www.nba.com/	2	0	218,313	180,129	38,184	121.2%
19	www.badoo.com/	2	0	117,443	50,837	66,606	231.0%
20	www.gamefaqs.com/	1	0	59,960	58,793	1,167	102.0%
				Total		730,746	3485.2%
				Rata-rata			36537.3

C1 - C2 = Selisih pemakaian bandwidth.

C1 / C2 = Pengurangan pemakaian bandwidth.

Gambar 4.9 menunjukkan perbandingan antara pengunaan dengan *CookiesLive* maupun tidak. Terlihat di setiap *website*, *CookiesLive* dapat mengurani pemakaian *bandwidth* yang terbuang walaupun tidak sesignifikan jika dibandingkan dengan parameter waktu.



Gambar 4.9. Perbandingan pemakaian dari parameter *bandwidth.*

4.2.5. Variasi Jumlah *Account*

Selanjutnya kita hitung signifikansi bagi user dengan jumlah *account* mulai dari satu sampai dua puluh apabila menggunakan *CookiesLive* dengan tidak. Perhitungan ditunjukkan pada Tabel 4.5. Diasumsikan *user* akan membuka *website* sesuai dengan urutan yang disediakan kali ini. Khusus untuk penggunaan *CookiesLive* akan dihitung juga fase dimana *users* harus melakukan proses *store* dan *load.*

Tabel 4.5. Perhitungan dengan variasi jumlah *account*.

Jumlah *Account*	W	X	(W – X)	Y	Z	(Y – Z)
Inisialisasi	23.749			213,160		
1	31.411	19.69	-11.721	234,364	44,632	-189,732
2	41.176	43.831	-2.655	260,887	111,802	-149,085
3	43.385	69.852	26.467	279,307	131,223	-148,084
4	69.885	98.245	28.36	320,868	175,675	-145,193
5	83.909	143.322	59.413	366,332	339,835	-26,497
6	96.713	175.511	78.798	483,855	461,324	-22,531
7	102.646	220.224	117.578	532,891	625,086	92,195
8	117.159	235.319	118.16	605,804	726,658	120,854
9	121.934	266.819	144.885	605,834	802,805	196,971
10	139.088	295.661	156.573	684,091	903,733	219,642
11	167.275	339.286	172.011	712,714	965,324	252,610
12	172.326	356.005	183.679	805,005	1,070,913	265,908
13	195.004	410.256	215.252	1,207,949	1,502,304	294,355
14	206.142	435.9	229.758	1,237,740	1,552,734	314,994
15	224.245	478.884	254.639	1,299,481	1,633,613	334,132
16	246.526	525.471	278.945	1,394,205	1,771,478	377,273
17	258.383	544.844	286.461	1,515,305	1,957,063	441,758
18	277.781	600.158	322.377	1,695,434	2,175,376	479,942
19	289.808	643.443	353.635	1,746,271	2,292,819	546,548
20	308.849	691.436	382.587	1,805,064	2,352,779	547,715

Ket:

jumlah *account* = n.

Wn = Insialiasi(detik) + $\sum_{k=1}^{n}$ B2n

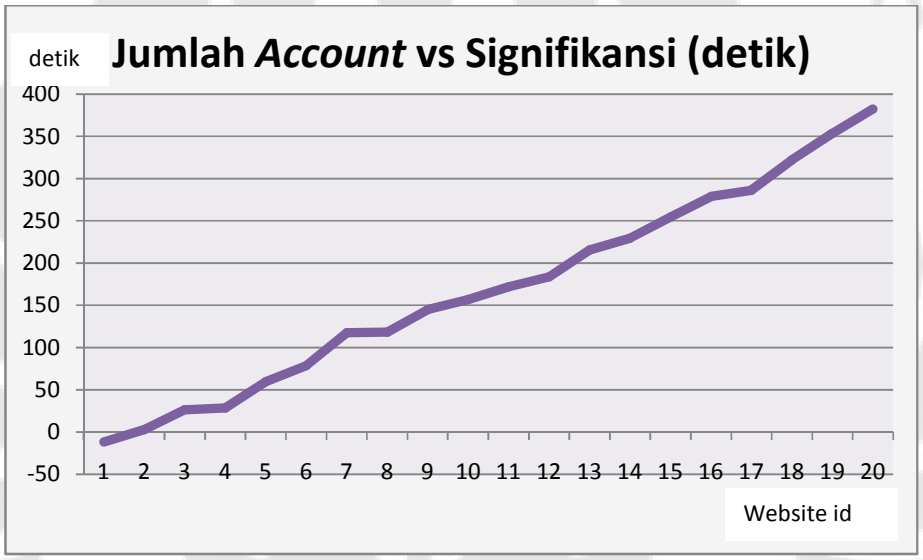
Xn = $\sum_{k=1}^{n}$ B1n

Yn = Insialiasi(*bytes*) + $\sum_{k=1}^{n}$ C2n

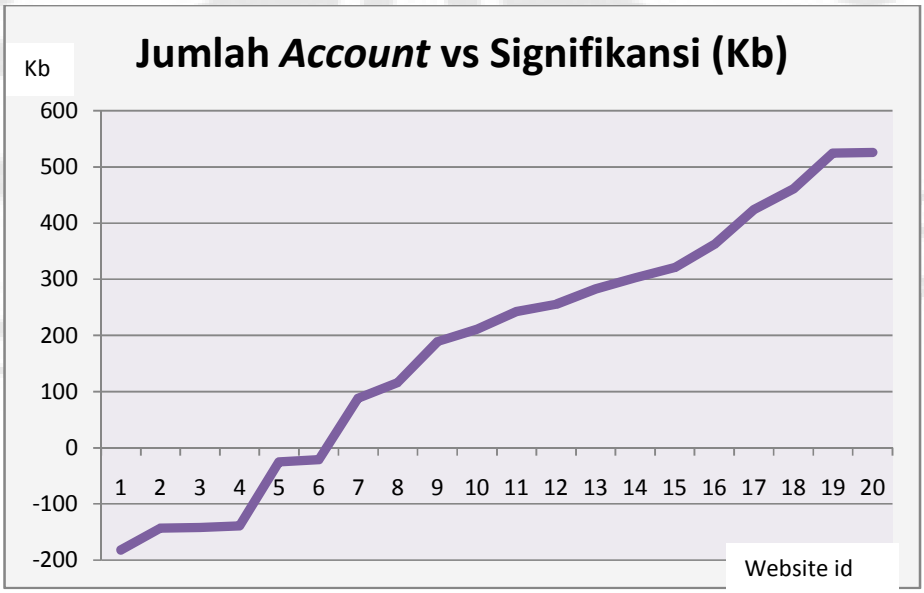
Zn = $\sum_{k=1}^{n}$ C1n

Dari Gambar 4.10 dan 4.11 dapat dilihat semakin banyak *account* yang dimiliki oleh seorang pengguna *CookiesLive* maka akan semakin besar keuntungan yang didapatnya.

Dari segi waktu yang terpakai, jika *users* hanya memiliki satu atau dua *account* maka *users* tidak akan merasakan keuntungan dari *CookiesLive* karena waktu yang dibutuhkan dari proses *login CookiesLive*, *store* dan *load cookies*. Tetapi jika *users* memiliki tiga atau lebih *account* maka users akan mendapatkan keuntungan dari pemakaian *CookiesLive*.



Gambar 4.10 Grafik Jumlah Account vs Signifikansi (detik).



Gambar 4.11 Grafik jumlah *account* vs signifikansi (Kb).

Sementara itu dari Gambar 4.11, untuk parameter *bandwidth* keuntungan dari kegunaan *CookiesLive* mulai dapat dirasakan saat memiliki *account* sejumlah tujuh atau lebih. Terlihat bahwa parameter detik lebih butuh sedikit *account* daripada parameter *bandwidth* untuk mendapatkan keuntungan. Ini dikarenakan karena waktu tidak berhenti saat *users* mengetikkan *username* dan *password*, sementara itu tidak berlaku untuk *bandwidth*. Ketika *browsers* sudah selesai memproses suatu halaman, maka ketika itu juga pemakaian *bandwidth* akan berhenti.

Hal ini menunjukkan untuk mendapatkan sepenuhnya keuntungan dari *CookiesLive* maka *users* setidaknya harus memiliki tujuh *account*. Dengan perkembangan *web* saat ini dimana akan banyak *web startups* bermunculan[8], *users* diharapkan akan memiliki setidaknya tujuh *account*.

BAB V

KESIMPULAN

1. *CookiesLive* dapat dijalankan dengan baik dan dapat menjalankan fungsinya untuk menyimpan *cookies* secara *online* dan melakukan *load* di *browser Firefox*.
2. Semakin banyak halaman yang harus dibuka memerlukan waktu yang semakin lama untuk *login*. Untuk itu *form login* sebaiknya disimpan di halaman utama.
3. Dari perhitungan untuk masuk ke masing-masing *website*, peningkatan efisiensi yang didapat untuk waktu memiliki *range* dari 107.1% – 1178%.
4. Dari perhitungan untuk masuk ke masing-masing website, peningkatan efisiensi yang didapat untuk *bandwidth* memiliki *range* dari 102.0% – 334.0%.
5. Dari peningkatan efisiensi waktu, *user* akan merasakan keuntungan dari *CookiesLive* ketika *user* memiliki minimal tiga *account*.
6. Dari peningkatan efisiensi *bandwidth*, *user* akan merasakan keuntungan jika *user* memiliki minimal tujuh *account*.

DAFTAR ACUAN

[1] David Kristol. HTTP Cookies: Standards, privacy, and politics. ACM Transactions on Internet Technology, 1(2), 151–198, 2001.

[2] RFC 2109 and RFC 2965. HTTP State Management Mechanism.

[3] Mozilla Developer Center. HTTP://developer.mozilla.org/ Februari 2008.

[4] Browser Version Market Share for October 2007. Net Applications. November 2007.

[5] Mozilla Foundation. Mozilla Code Licensing. 2007-09-17.

[6] Jonah Bishop. HTTP://www.borngeek.com/. Desember 2007.

[7] Alexa. Web Information Company. 21 Maret 2008.

[8] Paul Graham, “The Future of Web Startups”, FOWA, Oktober 2007.

DAFTAR PUSTAKA

Aaron Andersen, Neil Deakin, *XULPlanet* (http://www.xulplanet.com/, Januari 2008)

Elisabeth Freeman, Eric Freeman, *Head First HTML with CSS & XHTML* (O'Reilly, December 2005)

Danny Goodman, Michael Morrison, *JavaScript Bible Sixth Edition* (Indianapolis: Wiley Publishing, Inc., 2007)

Kenneth C. Feldt, *Programming Firefox* (Sebastopol: O'Reilly Media, Inc., April 2007)

Tim Converse, Joyce Park, Clark Morgan, *PHP5 and MySQL Bible* (Indianapolis: Wiley Publishing, Inc., 2004)