BAB III

METODE PENELITIAN DAN PERANCANGAN APLIKASI

3.1 Metode Penelitian

Metode penelitian yang akan digunakan dalam penelitian ini dapat dijabarkan sebagai berikut.

a. Studi Literatur

Penelitian diawali dengan melakukan studi literatur menggunakan referensi buku, jurnal, dan artikel mengenai produktivitas, *data fetching*,, dan *sampling*.

b. Perancangan Aplikasi

Menentukan analisa kebutuhan, perumusan masalah, DFD, dan rancangan antarmuka aplikasi.

c. Observasi

Observasi dilakukan terhadap perusahaan untuk mendapatkan data-data mengenai tugas-tugas apa saja yang dilakukan karyawan dan bagaimana seharusnya menggunakan internet.

d. Pembangunan Aplikasi

Pembangunan aplikasi menggunakan bahasa pemrograman C# dengan Visual Studio 2010 dan *database* yang digunakan adalah MySQL dan SQLite.

e. Uji Coba dan Evaluasi

Uji coba aplikasi akan dilakukan pada kantor selama lima minggu, data akan dikumpulkan lalu dianalisis oleh aplikasi dan didapatkan hasilnya berupa data tabel dan grafik.

f. Penulisan laporan

Tahap penulisan dilakukan selama proses perancangan dan pengembangan aplikasi sebagai sumber dokumentasi untuk menjelaskan hasil pengujian, analisis data, simpulan, dan saran untuk penelitian selanjutnya

3.2 Timeline Penelitian

Tabel 3.1 *Timeline* Penelitian

No	Kegiatan	Weeks												
		1	2	3	4	5	6	7	8	9	10	11	12	13
1	Studi literatur													
2	Analisa kebutuhan													
3	Perancangan sistem													
4	Pembuatan sistem													
5	Pengujian sistem													
6	Penulisan laporan													

3.3 Perangkat Pengembangan

Untuk pengembangan aplikasi memerlukan:

a. Visual Studio 2010
b. Bahasa Pemrograman C#
c. SQLite *Database*
d. MySQL *Database*

Laptop yang akan digunakan dengan spesifikasi

a. Intel Core i3-2330M, quad core @2.2Ghz
b. RAM 4 GB
c. Sistem Operasi Windows 7 Home Basic 64-bit

Spesifikasi komputer yang digunakan di kantor

a. Intel Atom, quad core @1.9Ghz

b. RAM 2 GB

c. Sistem Operasi Windows 7 Home Basic 32-bit

Spesifikasi minimum yang dibutuhkan agar aplikasi dapat berjalan:

a. Net Framework 3.5

b. OS Windows 7

c. RAM 1 GB

3.4 Analisa Kebutuhan

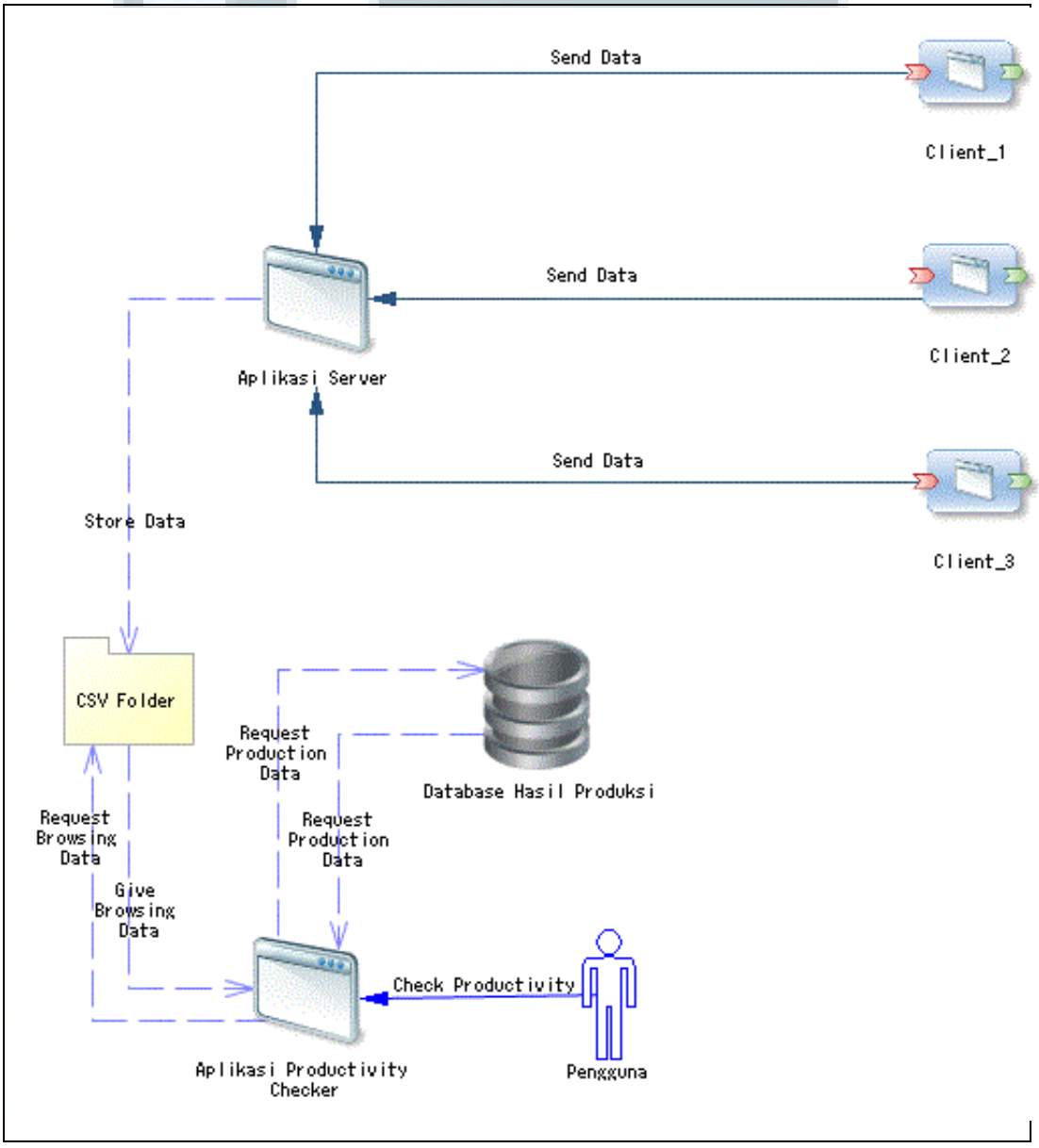
Dalam tahap ini, dianalisa hal-hal apa saja yang perlu dikembangkan dalam aplikasi. Hal-hal tersebut adalah.

1. Aplikasi *Client* untuk membaca *browsing history* dalam bentuk *database* SQLite dari aplikasi *browser* Mozilla Firefox dan Google Chrome lalu mengirimnya ke aplikasi *Server*.
2. Aplikasi *Server* yang berfungsi untuk menerima data dari aplikasi *Client* dan menyimpannya dalam format .csv yang bisa dibaca oleh aplikasi *Productivity Checker*.
3. Kedua aplikasi tersebut terhubung dalam hubungan *One to Many*, dengan satu *server* dan banyak *client*.
4. Aplikasi *Productivity Checker* berfungsi untuk membandingkan hasil kerja dengan tingkat *cyberloafing*.
5. Aplikasi *Client* harus berupa *Windows Service* agar tersembunyi atau tidak terlihat oleh pengguna.
6. Aplikasi *Client* dapat berjalan dengan sendirinya, tanpa perlu diaktifkan.

3.5 Proses Perancangan

Hal pertama yang dilakukan adalah membuat Diagram Umum, *Flowchart*, DFD, lalu dilanjutkan dengan desain UI (*User Interface*) dan diakhiri dengan implementasi.

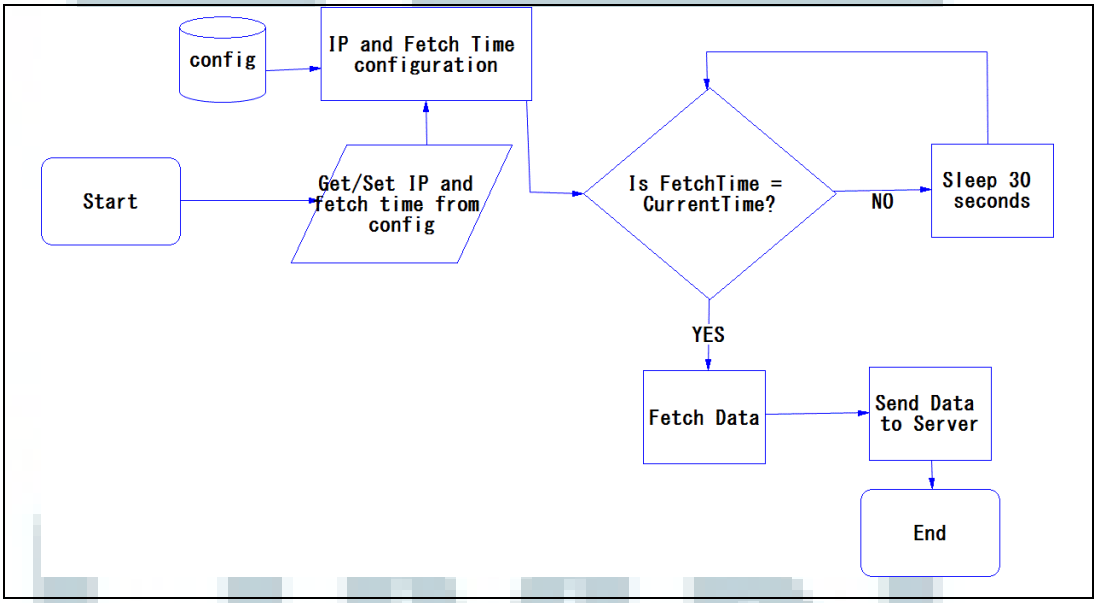
3.5.1 Diagram Umum



Gambar 3.1 Diagram Umum Aplikasi

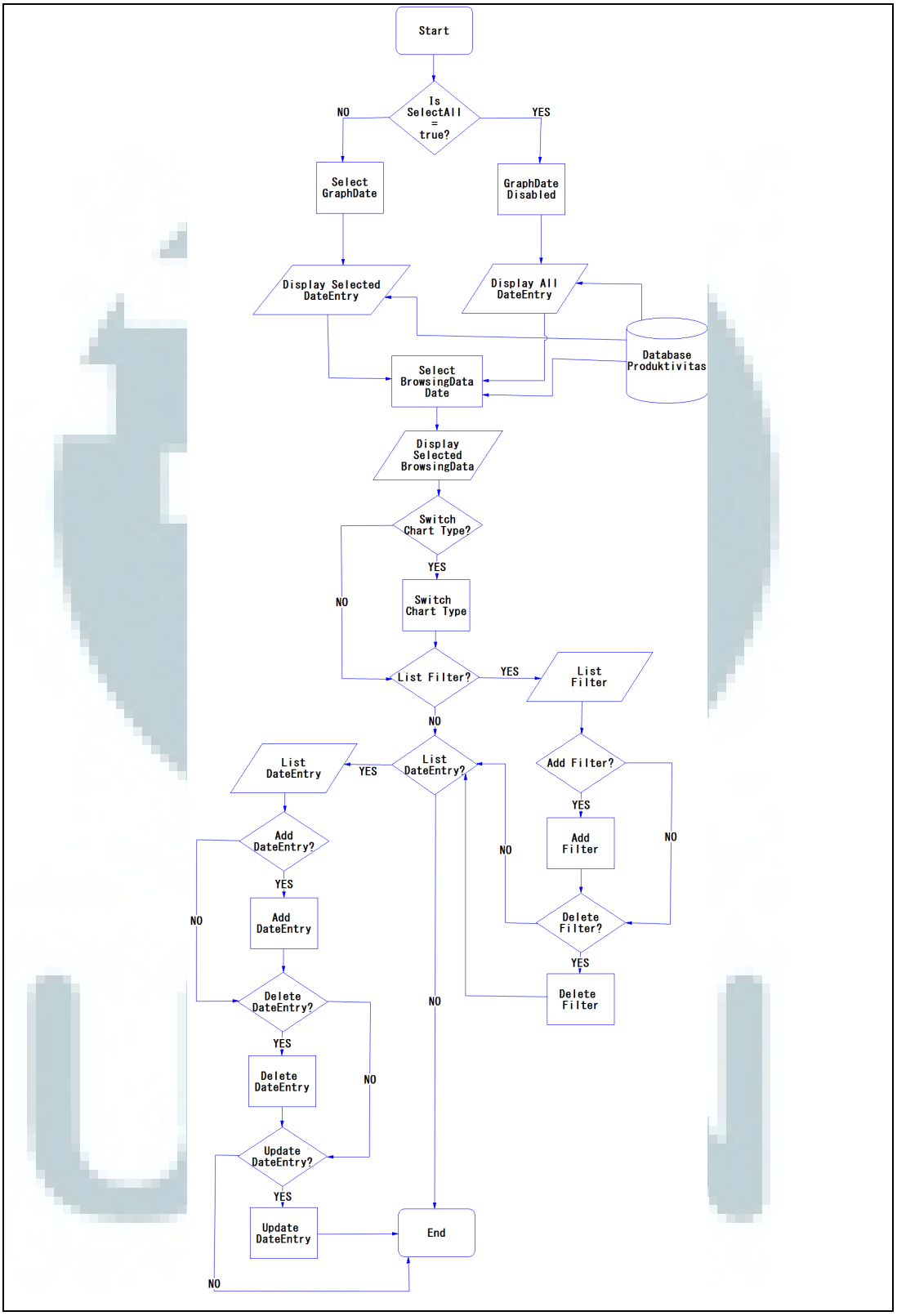
Dari diagram umum di atas, dapat dilihat cara kerja program secara garis besar. Pertama, *Service-Service Client* yang dipasang akan mengirim data *Browsing History* secara periodik kepada *Server* yang diterima oleh aplikasi *Server*. Selanjutnya, data-data *Browsing History* akan disimpan dalam sebuah *folder* dengan *path* "*C:_TestBackgroundProcess*" dalam bentuk file *csv*. Pengguna dapat menggunakan aplikasi *Productivity Checker* untuk membandingkan data *Browsing History* dengan tingkat Produktivitas pada suatu periode tertentu.

3.5.2 Flow Chart



Gambar 3.2 *Flow Chart Data Fetching*

Diagram di atas menggambarkan *System Flow* dari program *Fetching Data*. Pertama program *Client* akan mendapatkan IP *Server* dan memeriksa apakah *Current Time* sama dengan *Fetch Time* yang terdapat di *config.txt*. Jika tidak sama, maka program akan *sleep* selama 30 detik, jika sama, maka program akan *fetch data* lalu mengirimkannya ke *Server*.

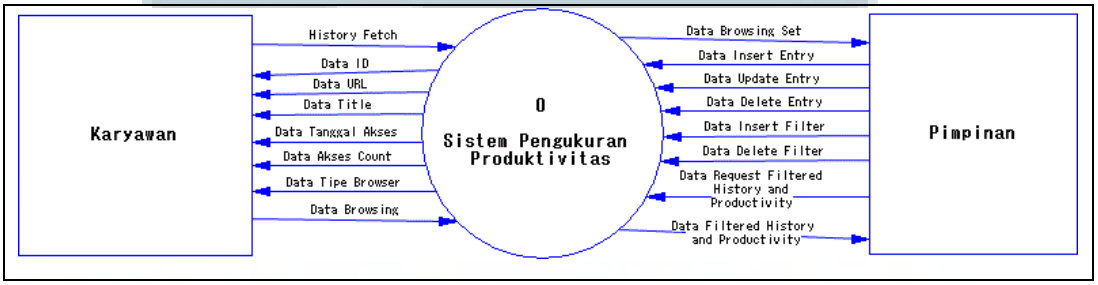


Gambar 3.3 *Flow Chart Productivity Checker*

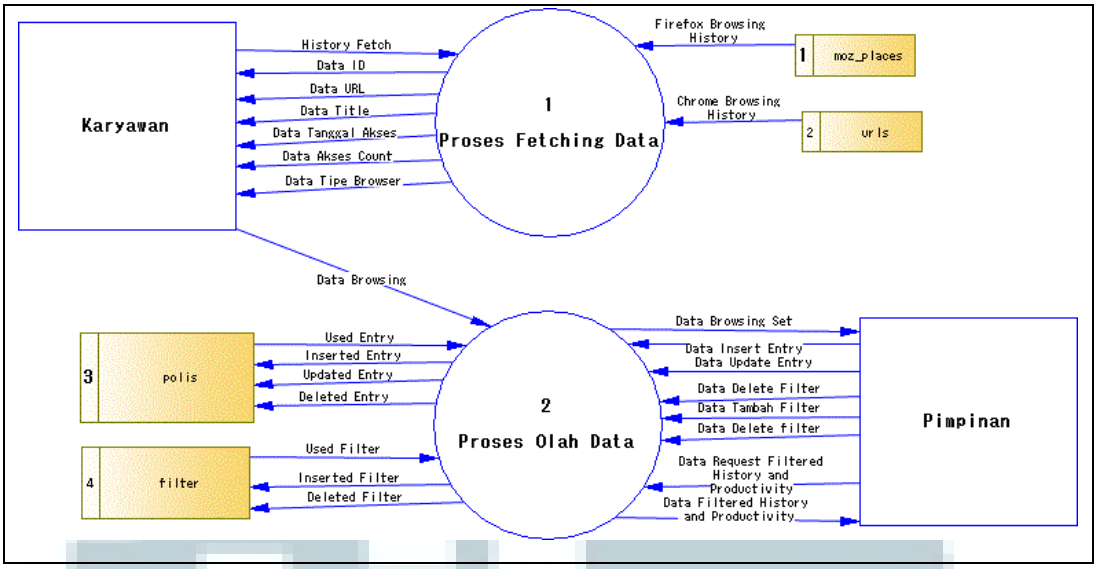
Diagram di atas menggambarkan *System Flow* dari program *Productivity Checker*. Program dimulai dengan menampilkan jendela *Main Menu*, dari titik ini, pengguna dapat melakukan berbagai macam aktivitas seperti menampilkan grafik data produktivitas dengan memilih tanggal atau mengaktifkan fungsi *Select All* untuk menampilkan semua data sekaligus; mengubah tipe *Chart* (Produksi atau Rasio); memilih tanggal untuk menampilkan data *Cyberloafing* yang berkaitan; *View*, *Insert*, atau *Delete* daftar *filter*; *View*, *Insert*, *Update*, atau *Delete Date Entry* dalam *database*.

3.5.3 DFD

Proses utama adalah "Sistem Pengukuran Produktivitas" dengan dua entitas yaitu, Karyawan dan Pimpinan.

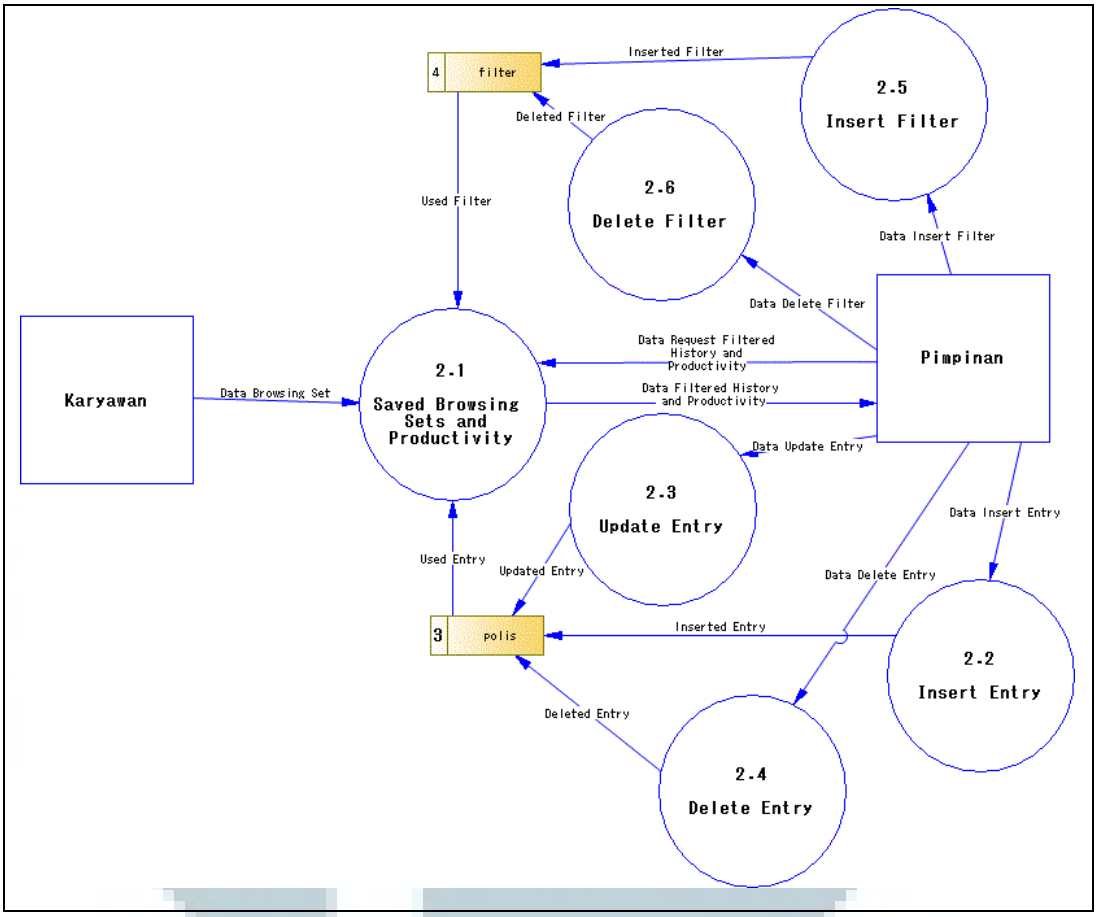


Gambar 3.4 DFD *Context*



Gambar 3.5 DFD Level 1

Dalam DFD Level 1, terdapat dua proses yaitu Proses *Fetching Data* dan Proses Olah Data. Proses *Fetching Data* berperan dalam mengambil data *browsing* yang disimpan oleh *browser* dalam tabel moz_places (*Firefox*) dan tabel urls (*Chrome*). Proses Olah Data merepresentasikan penggunaan program *Productivity Checker* oleh entitas Pimpinan. Proses ini juga menggunakan dua tabel sumber data, yaitu *polis* dan *filter*.



Gambar 3.6 DFD Level 2 Proses Olah Data

Pada DFD Level 2 dari Proses Olah Data, setelah menerima data *browsing set* dari Karyawan. Proses *Saved Browsing Sets and Productivity* menyimpannya dalam sebuah *folder* dan menjadi sumber data untuk program *Productivity Checker*. Pimpinan dapat melakukan *insert, update*, dan *delete entry* data ke tabel polis, juga *insert* dan *delete filter* ke tabel *filter*. Pimpinan dapat *request* program untuk menampilkan data *browsing* yang telah di*filter* oleh parameter-parameter yang ditentukan dalam tabel polis dan *filter*.

3.5.4 Struktur Tabel

Database yang digunakan adalah *SQLite* dan *MySQL*. Berikut adalah struktur tabel-tabel yang digunakan.

Nama tabel : urls

Fungsi : Menyimpan data *browsing* dari *browser* Chrome.

Tabel 3.2 Struktur Tabel urls

Field Name	**Type**	**Length**	**Information**
id	Integer		*Primary key*
url	Varchar	100	
title	Varchar	100	
visit_count	Integer		
last_visit_time	Date/Time		

Nama tabel : moz_places

Fungsi : Menyimpan data *browsing* dari *browser* Firefox.

Tabel 3.3 Struktur Tabel moz_places

Field Name	**Type**	**Length**	**Information**
id	Integer		*Primary key*
url	Varchar	100	
title	Varchar	100	
visit_count	Integer		
favicon_id	Date/Time		*Foreign Key*

Nama tabel: moz_historyvisits

Fungsi : Menyimpan data *browsing* dari *browser* Firefox.

Tabel 3.4 Struktur Tabel moz_historyvisits

Field Name	**Type**	**Length**	**Information**
id	Integer		*Primary key; Foreign Key* ke moz_places.id
place_id	Integer		
visit_date	Date/Time		

Nama tabel: polis

Fungsi : Menyimpan data produktivitas

Tabel 3.5 Struktur Tabel polis

Field Name	Type	Length	Information
ID	Integer		*Primary key*
Jumlah_Polis	Integer		
Date_Awal	Date/Time		
Date_Akhir	Date/Time		
Jumlah_Pekerja	Integer		

Nama tabel: filter

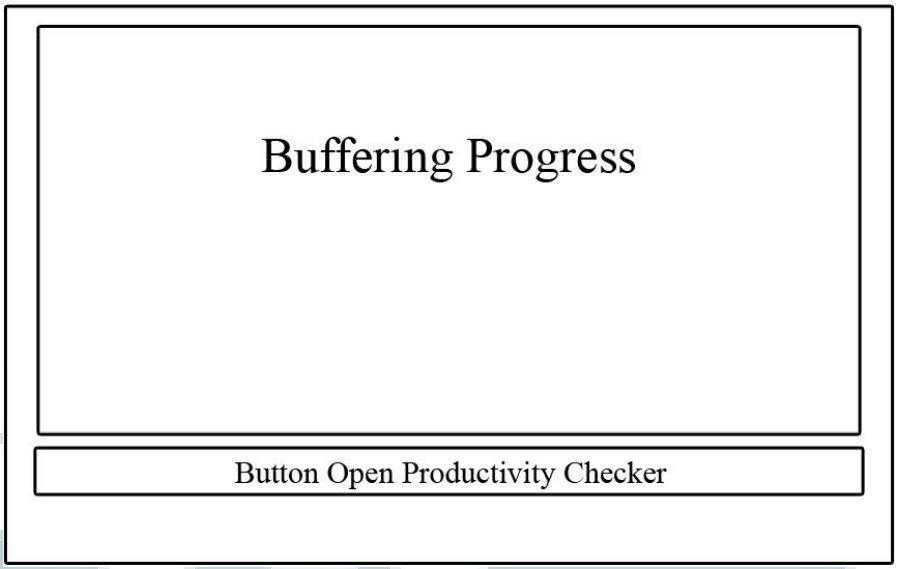
Fungsi : Menyimpan daftar *filter* situs *web*.

Tabel 3.6 Struktur Tabel filter

Field Name	Type	Length	Information
filter_name	Varchar	100	

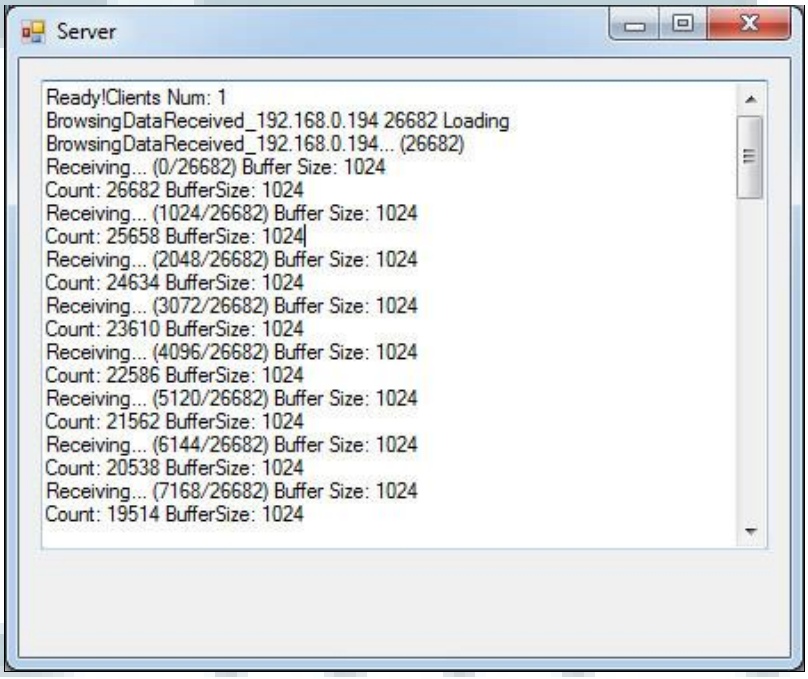
3.5.5 Desain Antarmuka

Pada tahap ini, didesain *user interface* dari program *Server* dan program *Productivity Checker*. Program *Client* tidak memiliki *user interface* karena berupa *Windows Service* yang tidak terlihat dan berjalan dengan otomatis. Gambar berikut menunjukkan bahwa *Server* siap menerima data dari *Client*. Ketika data sedang dalam proses pengiriman, maka *Server* akan menunjukkan data tersebut berasal dari *Client* nomor berapa dan berapa banyak data yang sudah dikirim dibandingkan ukuran total data.

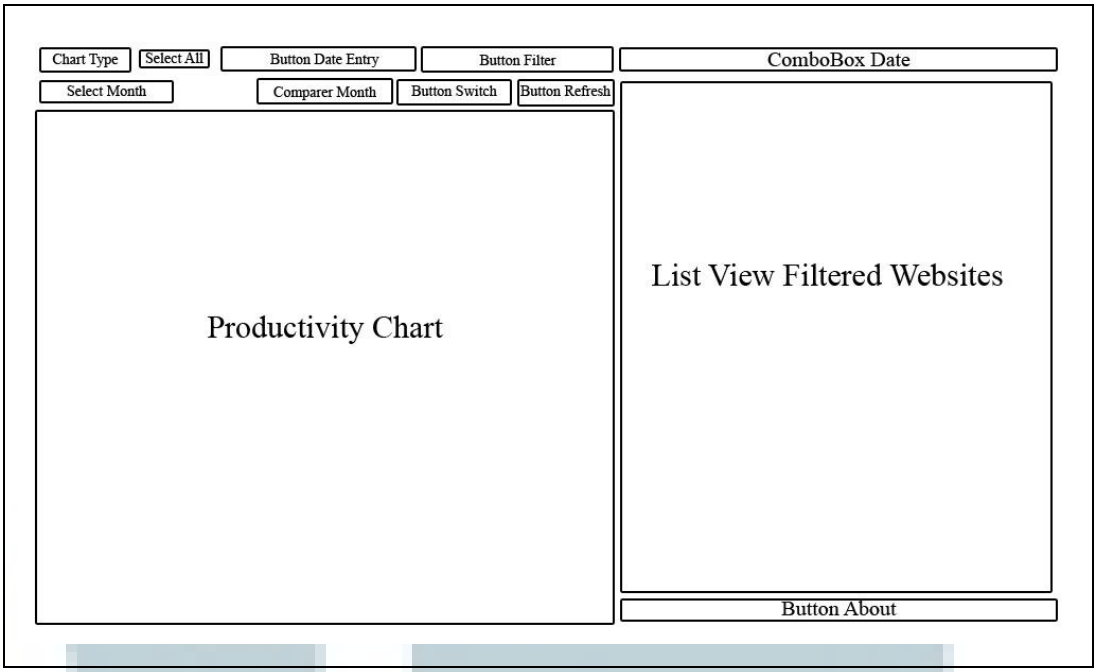


Gambar 3.7 Desain UI *Server*

Setelah tahap perancangan, berikutnya *UI* diimplementasikan dalam program dan hasilnya adalah sebagai berikut.



Gambar 3.8 UI *Server*



Gambar 3.9 Desain Antarmuka *Productivity Checker*

Pada jendela utama bagian kiri menampilkan grafik tingkat produktivitas sedangkan bagian kanan menampilkan data akses situs-situs yang telah di*filter*. Bagian atas mengandung tombol dan *combo box* untuk memanipulasi tampilan, menambahkan data *entry* dan *filter*, juga *refresh* tampilan.

Gambar 3.10 Desain Antarmuka *List Date Entry*

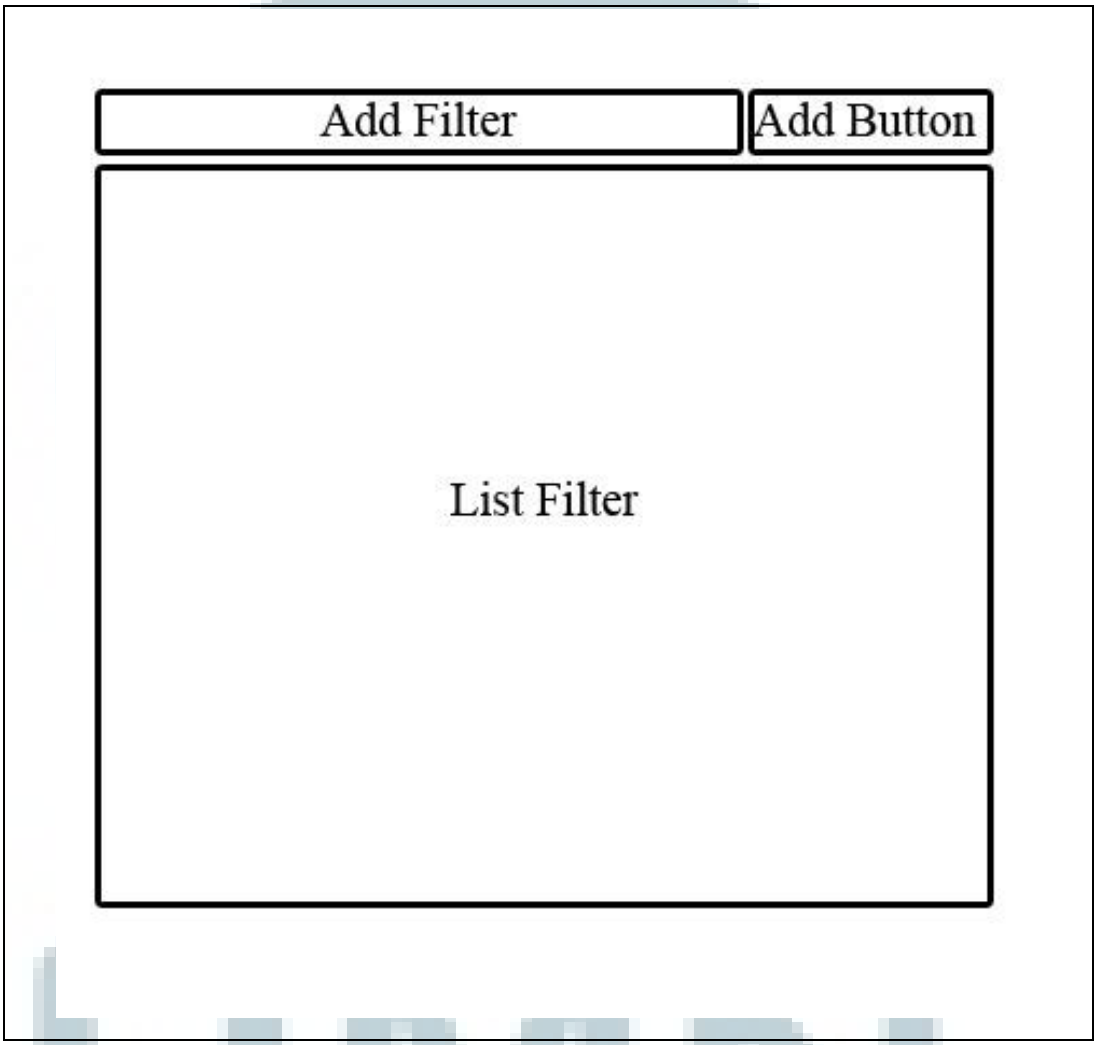
Pada desain antarmuka *List Date Entry*, terdapat *list view* untuk mendaftar semua *entry* dalam tabel polis, dan terdapat tombol di atas untuk membuka jendela penambahan *date entry*. Pengguna juga dapat membuka jendela perubahan *date entry* dengan menggunakan *context menu*.

Gambar 3.11 Desain Antarmuka *Add Date Entry*

Pada jendela *Add Date Entry*, terdapat tempat untuk mengisi jumlah polis yang dihasilkan, *range* tanggal awal, *range* tanggal akhir, dan berapa banyak karyawan yang bekerja selama *range* itu.

Gambar 3.12 Desain Antarmuka *Update Date Entry*

Pada jendela *Update Date Entry*, sebagian besar sama dengan jendela *Add Date Entry*, namun ditampilkan data ID dari *entry* yang akan diubah. ID ini otomatis dan tidak dapat diubah oleh pengguna.



Gambar 3.13 Desain Antarmuka *List Filter*

Pada jendela *List Filter*, terdapat *list view* untuk mendaftar semua *filter* dalam tabel *filter*, dan terdapat *text box* dan tombol di atas untuk menambahkan *filter* baru.