TUGAS AKHIR - KS 091336

APLIKASI *ADDONS INLINE TRANSLATOR* UNTUK MOZILLA FIREFOX MENGGUNAKAN ALGORITMA PORTER STEMMER

DEFIR ILMI FARIDHA
NRP 5206 100 010

Dosen Pembimbing
Dr.Eng. Febriliyan Samopa S.Kom, M.Kom

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2011

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT - KS 091336

INLINE TRANSLATOR ADDONS APPLICATION FOR MOZILLA FIREFOX USING PORTER STEMMER ALGORITHM

DEFIR ILMI FARIDHA
NRP 5206 100 010

Supervisor
Dr.Eng. Febriliyan Samopa S.Kom, M.Kom

INFORMATION SYSTEM DEPARTMENT
Information Technology Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2011

APLIKASI *ADDONS INLINE TRANSLATOR* UNTUK MOZILLA FIREFOX MENGGUNAKAN ALGORITMA PORTER STEMMER

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

DEFIR ILMI FARIDHA
5206 100 010

Disetujui Tim Penguji : Tanggal Ujian: 2 Februari 2011
Periode Wisuda : Maret 2011

Dr.Eng. Febriliyan Samopa, S.Kom, M.Kom **(Pembimbing I)**

Bambang Setiawan, S.Kom, MT **(Penguji 1)**

Faizal Johan Atletiko, S.Kom, M.Kom **(Penguji 2)**

APLIKASI ADDONS INLINE TRANSLATOR UNTUK MOZILLA FIREFOX MENGGUNAKAN ALGORITMA PORTER STEMMER

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

DEFIR ILMI FARIDHA
5206 100 010

Surabaya, Februari 2011

KETUA
JURUSAN SISTEM INFORMASI

Ir. A. HOLIL NOOR ALI M.KOM
NIP 19660602 199203 1 002

APLIKASI *ADDONS INLINE TRANSLATOR* UNTUK MOZILLA FIREFOX MENGGUNAKAN ALGORITMA PORTER STEMMER

Nama : **DEFIR ILMI FARIDHA**
NRP : **5206 100 010**
Jurusan : **SISTEM INFORMASI FTIF-ITS**
Dosen Pembimbing : **DR. ENG. FEBRILIYAN SAMOPA, S.KOM, M.KOM**

ABSTRAK

Perkembangan internet yang signifikan memunculkan ide untuk menjelajahi website dengan menggunakan browser. Sejak saat itu pertumbuhan pengguna internet semakin meroket, informasi menjadi sangat mudah untuk didapatkan bahkan dari belahan dunia lain. Akan tetapi, informasi yang di dapat sangatlah sulit untuk dipahami karena perbedaan bahasa antara pemberi informasi dan penerima informasi. akibatnya penyampaian informasi melalui internet sering kali terhambat.

Guna menjembatani perbedaan bahasa tersebut, dibutuhkan suatu aplikasi yang disediakan oleh browser untuk menerjemahkan informasi tersebut secara otomatis. Sehingga, pengguna internet diberi kemudahan untuk memperoleh informasi dan memahami informasi yang dibutuhkan dari website yang dikelola oleh pengguna internet dibelahan dunia lainnya yang berperan sebagai pemberi informasi.

Saat ini banyak penyedia aplikasi berbasis addons yang menyediakan fasilitas penerjemah otomatis dalam beberapa bahasa sehingga pengguna dapat menyesuaikan dengan keinginannya untuk menerjemahkan informasi tersebut dengan bahasa yang biasa mereka gunakan tanpa perlu meninggalkan halaman website yang sedang dibuka. Hanya saja aplikasi yang sudah ada itu memiliki kelemahan dalam menerjemahkan informasi yang dibutuhkan oleh pengguna. Hasil terjemahan dari

aplikasi-aplikasi tersebut sering kali tidak sesuai context. Oleh karena itu dibuatlah tugas akhir ini dengan harapan dapat membantu pengguna internet khususnya yang berasal di Indonesia untuk menerjemahkan informasi yang telah didapat dari website berbahasa asing dalam hal ini website berbahasa inggris sesuai dengan context bahasa indonesia, sehingga hasil terjemahan yang di dapat lebih akurat. Selain itu, pengguna dapat memperoleh beberapa arti kata dari suatu kata secara berbeda sesuai dengan kategori dari informasi yang di dapat. Untuk itu dibutuhkan penerapan algoritma yang dapat mengintegrasikan tujuan aplikasi dalam tugas akhir ini untuk memperoleh hasil terjemahan yang context sensitive sehingga pengembangan aplikasi penerjemah berbasis addons ini membutuhkan penerapan algoritma porter stemmer.

Kata kunci : Inline translator, Inlinetrans, addons Mozilla firefox

INLINE TRANSLATOR ADDONS APPLICATION FOR MOZILLA FIREFOX USING PORTER STEMMER ALGORITHM

Name : **DEFIR ILMI FARIDHA**
NRP : **5206 100 010**
Departement : **INFORMATION SYSTEM FTIF-ITS**
Supervisor : **DR. ENG. FEBRILIYAN SAMOPA, S.KOM, M.KOM**

ABSTRACT

Significant Internet developments led to the idea to browse website using browser. Since that time the growth of Internet users increased, information becomes very easy to get even from other parts of the world. However, information which obtain by users can be very difficult to understand because of language differences between users who give and receive of information. Consequently the delivery of information via Internet is often hampered.

In order to bridge differences in language, it takes an application provided by the browser to translate that information automatically. Thus, Internet users are provided with access to obtain information and understand the information needed from a website managed by internet users in other world that acts as a conduit of information.

Today many providers of applications based on addons that provide automatic translator in several languages so that users can customize with a desire to translate that information with the language that they use without ever leaving the web page being opened. It's just that the existing application that has a weakness in translating the information needed by the user. The translation of these applications are often inappropriate context. Therefore, this final project expected can help internet users especially those originating in Indonesia to translate the information has been

KATA PENGANTAR

Alhamdulillaahirobbil 'aalamiin. Puji Syukur penulis panjatkan kepada Allah SWT atas hidayah dan petunjuk-Nya, sehingga penulis dapat menyelesaikan laporan tugas akhir dengan judul **"APLIKASI *ADDONS INLINE TRANSLATOR* UNTUK MOZILLA FIREFOX MENGGUNAKAN ALGORITMA PORTER STEMMER"**, yang merupakan salah satu syarat kelulusan pada Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih kepada pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, bantuan baik materi maupun spiritual demi tercapainya tujuan pembuatan tugas akhir ini. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1) Ibunda tercinta Endang Anggorowati dan almarhum ayahanda tercinta Moch. Surip serta adik-adik tercinta Annisa Nurul Faridha,Muh. Miftah Farid, dan Moch. Rezza Pahlawan atas doa,dukungan dan kasih sayangnya selama ini dan sampai kapanpun, nasihat dan semangat yang selalu diberikan pada penulis.
2) Bapak Dr.Eng. Febriliyan Samopa, S.Kom, M.Kom selaku dosen pembimbing yang memberikan petunjuk, bantuan dan motivasi untuk kelancaran Tugas Akhir ini.
3) Bapak Rully Soelaiman,S.Kom, M.Kom dan Bapak Ir. Arif Djunaidy selaku dosen wali yang telah memberikan banyak bantuan selama penulis berkuliah.
4) Seluruh dosen serta staf di Jurusan Sistem Informasi FTIF ITS yang telah memberikan ilmu dan bantuan kepada penulis selama ini.

5) Sahabat-sahabat penulis selama berkuliah Diah Rahmah, Ika Wulandari, Luci Dwi Agustin, serta teman-teman seperjuangan Bella Anisa Puspitaloka, Novi Tri N, Ika VBS,dkk. Teman-teman seperjuangan di ebis, sahabat-sahabat masa sekolah, dan semua teman yang tidak dapat disebutkan satu per satu. Terima kasih atas semua doa, dukungan, dan bantuannya.

6) Seluruh kakak, teman, dan adik di Sistem Informasi atas semua bantuan ketika penulis berkuliah di Sistem Informasi.

7) Seseorang yang selama ini menemani penulis, memberikan nasihat,pengalaman dan motivasi dan doa yang tiada henti,memberikan semangat kepada penulis untuk segera menyelesaikan tugas akhir, memberikan warna dalam kehidupan penulis selama ini, Akhmad Syariful Anwar.

Terima kasih atas segala bantuan, dukungan, serta doanya. Semoga Allah SWT senantiasa melimpahkan rahmat hidayah serta membalas semua kebaikan yang telah diberikan kepada penulis.

Surabaya, Februari 2011

Penulis

DAFTAR ISI

ABSTRAK v
ABSTRACT vii
KATA PENGANTAR ix
DAFTAR ISI xi
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan permasalahan 3
1.3. Batasan Permasalahan 4
1.4. Tujuan 4
1.5. Metodologi Penulisan 4
1.6. Sistematika Penulisan 4
1.7. Relevansi dan Manfaat 5
BAB II TINJAUAN PUSTAKA 7
2.1. Add Ons 7
2.2. Foxbeans 8
2.3. XUL 9
2.4. Javascript 10
2.5. Algoritma Porter Stemmer 13
BAB III METODOLOGI PENELITIAN 27
3.1. Studi Literatur dan Pengumpulan Data 28
3.2. Analisa Kebutuhan 28
3.3. Perancangan Perangkat Lunak 29
3.4. Pembuatan Perangkat Lunak 32
3.5. Uji Coba dan Evaluasi 33
3.6. Pembuatan Buku Tugas Akhir 33
BAB IV ANALISA KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK 35
4.1. Deskripsi Umum Sistem 35
4.2. Studi Literatur dan Pengumpulan Data 36
4.3. Analisa Kebutuhan 36
4.3.1. Pengguna 36
4.3.2. Fungsi 36

4.3.3. Teknologi 37
4.4. Arsitektur Sistem 37
4.5. Use Case Diagram 38
4.6. Robustness Diagram 40
4.7. Sequence Diagram 40
4.8. Desain Antarmuka 41
4.9. Desain Test case 41
BAB V IMPLEMENTASI DAN UJI COBA SISTEM 43
5.1. Lingkungan Implementasi 43
5.2. Implementasi Teknologi Web Aplikasi Fungsionalitas Tambahan (*addons*) 44
5.2.1 Implementasi Antar Muka *Add-ons* 44
5.2.2. Implementasi Pengolahan Kata 49
5.2.3. Implementasi Lalu Lintas Data 57
5.3. Uji Coba Sistem Aplikasi 61
5.3.1. Lingkungan Uji Coba 61
5.3.2. Skenario Uji Coba 64
5.3.3. Uji Coba Fungsional 66
BAB VI PENUTUP 67
6.1. Kesimpulan 67
6.2. Saran 68
DAFTAR PUSTAKA 69
LAMPIRAN A NARRATIVE USE CASE A-1
LAMPIRAN B ROBUSTNESS DIAGRAM B-1
LAMPIRAN C SEQUENCE DIAGRAM C-1
LAMPIRAN D TEST CASE D-1
LAMPIRAN E HASIL UJI COBA E-1
LAMPIRAN F ANALISA PERFORMA APLIKASI *INLINE TRANSLATOR* F-1
LAMPIRAN G TAMPILAN APLIKASI *INLINE TRANSLATOR* G-1
LAMPIRAN H *REQUIREMENT* H-1

DAFTAR GAMBAR

Gambar 2.1 *Screen shoot add-ons mozilla firefox* 7
Gambar 2.2 *Screen shoot plug-in foxbeans* 8
Gambar 2.3 Contoh penggunaan kode *XUL* 10
Gambar 2.4 Kondisi pada *rules* 16
Gambar 2.5 *Control flow* algoritma *Porter Stemmer* 16
Gambar 3.1 Alur Metodologi Penelitian 27
Gambar 3.2 Alur Proses Metode Use Case Drive Object Modelling 30
Gambar 3.3 Model waterfall pada pengembangan aplikasi tugas akhir 31
Gambar 4.1 Arsitektur Sistem Aplikasi *inlinetrans* 38
Gambar 4.2 Diagram *use case* pada *client* 39
Gambar 4.3 Desain antarmuka 41
Gambar 5.1 Antar muka *instalasi Inlinetrans* 45
Gambar 5.2 Antar muka *Context Menu Inlinetrans* 46
Gambar 5.3 S*tatusbar* sebelum diklik kanan 47
Gambar 5.4 *Statusbar* setelah diklik kanan 48
Gambar 5.5 antar muka informasi aplikasi 49
Gambar 5.6 Proses mendapatkan kata 50
Gambar 5.7 Proses mendapatkan posisi kata 51
Gambar 5.8 Proses mendapatkan seluruh *content website* 51
Gambar 5.9 Proses menghilangkan *tag* dan *entity HTML* 52
Gambar 5.10 Menghilangkan *patern* 53
Gambar 5.11 Proses mendapatkan kata sebelum dan sesudah 54
Gambar 5.12 Proses mendapatkan 15 kata sebelum dan 15 sesudah 55
Gambar 5.13 Proses menghilangkan kata yang frekuensi kemunculannya sering 56
Gambar 5.14 Proses menghasilkan akar kata 56
Gambar 5.15 Proses mengirim *request* ke *server*-terjemahan dengan kategori 57
Gambar 5.16 Proses mengirim *request* ke *server*-terjemahan tanpa kategori 58
Gambar 5.17 Struktur XML menerjemahkan dengan kategori 59

Gambar 5.18 Struktur XML menerjemahkan tanpa kategori 59
Gambar 5.19 Proses membaca XML-terjemahan dengan kategori 60
Gambar 5.20 Proses membaca XML-terjemahan tanpa kategori 61
Gambar B.1 Robustness Diagram Memasukkan Kata.....B-1
Gambar B.2 Robustness Diagram Membaca Hasil TerjemahanB-2
Gambar C.1 Sequence Diagram Memasukkan Kata C-1
Gambar C.2 Sequence Diagram Membaca Hasil Terjemahan ..C-2
Gambar G.1 proses instalasi aplikasi *inline translator*..... G-1
Gambar G.2 tampilan aplikasi *inline translator* saat pertama di instal G-1
Gambar G.3 status bar aplikasi *inline translator* saat di klik kanan G-2
Gambar G.4 Informasi tentang aplikasi *inline translator*..... G-2
Gambar G.5 status bar aplikasi *inline translator* saat dalam kondisi aktif G-2
Gambar G.6 status bar aplikasi *inline translator* saat dalam kondisi tidak aktif G-2
Gambar G.7 *context menu* aplikasi *inline translator* saat dalam kondisi aktif G-3
Gambar G.8 *context menu* aplikasi *inline translator* saat dalam kondisi tidak aktif G-3
Gambar G.9 tampilan hasil terjemahan dengan kategori..... G-3
Gambar G.10 tampilan hasil terjemahan tanpa kategori..... G-4
Gambar G.11 tampilan peringatan jika tidak ada kata yang di blok G-4
Gambar H.1 *functional requirement* aplikasi *inline translator* H-1
Gambar H.2 *non functional requirement* aplikasi *inline translator* H-2

DAFTAR TABEL

Tabel 2.1 *Remove plural suffixation (step 1A)* 17
Tabel 2.2 *Remove verbal inflection(Step 1B)* 18
Tabel 2.3 *Continued for -ed and -ing rules* (Step 1b1).....18
Tabel 2.4 *y and i* (Step 1c).....19
Tabel 2.5 *Peel one suffix off for multiple suffixes (*Step 2).....20
Tabel 2.6 *Step 3* 22
Tabel 2.7 *Delete last suffix* 23
Tabel 2.8 Remove e 24
Tabel 2.9 Reduction.....25
Tabel 5.1 Spesifikasi Perangkat Keras dan Sistem Operasi untuk Implementasi Sistem.....43
Tabel 5.2 Teknologi yang Digunakan untuk Implementasi Sistem 44
Tabel 5.3 Lingkungan Uji Coba I Aplikasi Inline Translator.....62
Tabel 5.4 Lingkungan Uji Coba II Aplikasi Inline Translator 62
Tabel 5.5 Lingkungan Uji Coba III Aplikasi Inline Translator ...62
Tabel 5.6 Lingkungan Uji Coba IV Aplikasi Inline Translator...63
Tabel 5.7 Lingkungan Uji Coba V Aplikasi Inline Translator 63
Tabel A.0.1 Narrative Use Case Memasukkan Kata 1
Tabel A.0.2 Narrative Use Case Membaca Hasil Terjemahan 4
Tabel D.1 Use Case Scenarios Memasukkan Kata.....2
Tabel D.2 Partial Scenario Matrix Use Case Memasukkan Kata ..3
Tabel D.3 Use Case Scenarios Membaca Hasil Terjemahan.....4
Tabel D.4 Partial Scenario Matrix Use Case Membaca Hasil Terjemahan 4
Tabel D.5 Identify Test Case Memasukkan Kata 6
Tabel D.6 Identify Test Case Membaca Hasil Terjemahan.....8
Tabel E.1 Hasil Uji Test Case Memasukkan Kata.....2
Tabel E.2 Hasil Uji Test Case Memasukkan Kata.....5
Tabel F.1 Lingkungan Uji Coba I Aplikasi *Inline Translator*...F-1
Tabel F.2 Performa Aplikasi *Inline Translator* Pada Uji Coba I F-1
Tabel F.3 Lingkungan Uji Coba II Aplikasi *Inline Translator*..F-2

Tabel F.4 Performa Aplikasi *Inline Translator* Pada Uji Coba II F-3
Tabel F.5 Lingkungan Uji Coba III Aplikasi *Inline Translator* F-4
Tabel F.6 Performa Aplikasi *Inline Translator* Pada Uji Coba III F-4
Tabel F.7 Lingkungan Uji Coba IV Aplikasi *Inline Translator* F-5
Tabel F.8 Performa Aplikasi *Inline Translator* Pada Uji Coba IV F-6
Tabel F.9 Lingkungan Uji Coba V Aplikasi *Inline Translator* F-7
Tabel F.10 Performa Aplikasi *Inline Translator* Pada Uji Coba V F-7
Tabel F.11 Lingkungan Uji Coba Kompatibelitas Aplikasi *Inline Translator* F-9
Tabel F.12 Performa Kompatibelitas Aplikasi *Inline Translator* Pada Uji Coba F-9

BAB I
PENDAHULUAN

Pada bagian ini dikemukakan hal-hal yang melatarbelakangi pentingnya dilakukan penelitian, rumusan, dan batasan permasalahan yang dikerjakan dalam penelitian, tujuan dan relevansi atau manfaat penelitian terhadap perkembangan solusi dari permasalahan yang diangkat, serta sistematika penulisan dalam laporan tugas akhir ini.

1.1. Latar Belakang

Teknologi informasi dan komunikasi saat ini mengalami perkembangan pesat. Kedua hal inilah yang mempengaruhi individu maupun komunitas, segala aktivitas, kehidupan, gaya hidup dan cara berpikir secara signifikan. Adanya teknologi informasi dan komunikasi memberikan kemudahan, misalnya saja dalam melakukan komunikasi, penyebaran informasi, hiburan, dan sebagainya.

Internet merupakan salah satu bentuk pemanfaatan produk-produk teknologi informasi dan komunikasi yang saat ini begitu beragam. Internet bukan lagi menjadi barang baru bagi masyarakat, internet berasal dari kata *Interconnection Networking*, yang berarti hubungan dari banyak jaringan komputer dengan berbagai tipe dan jenis, dengan menggunakan tipe komunikasi seperti telepon, satelit, dan lainnya. Internet memungkinkan pengguna komputer di seluruh dunia untuk saling berkomunikasi dan berbagi informasi dengan cara saling mengirimkan email, menghubungkan komputer satu ke komputer yang lain, mengirim dan menerima file dalam bentuk text, audio, video, membahas topik tertentu pada *newsgroup*, *website social networking* dan lain-lain.

Aktivitas saling berkomunikasi dan berbagi informasi mendorong **Tim Berners-Lee** pada tahun 1990 untuk membangun suatu aplikasi *world wide web* (www). Aplikasi ini menjadi konten yang paling dinantikan oleh pengguna internet, sejak saat itu pertumbuhan internet semakin meroket. Perkembangan aplikasi *world wide web* (www), memunculkan ide untuk membangun suatu aplikasi yang berguna sebagai mesin penjelajah *web*. Aplikasi yang saat ini lebih dikenal sebagai *browser* merupakan suatu aplikasi yang digunakan untuk menampilkan dan melakukan interaksi dengan file-file yang disediakan oleh *web server*.

Pengguna internet dimungkinkan melakukan aktivitas dengan pengguna internet di seluruh dunia untuk saling berkomunikasi dan berbagi informasi, padahal aktivitas komunikasi seperti ini tentunya terkendala dengan perbedaan budaya dan bahasa. Sehingga, sangat dimungkinkan adanya kesalahpahaman dalam penyampaian informasi tersebut. Misalnya pengguna internet yang menggunakan bahasa Indonesia sebagai bahasa sehari-hari, tentunya penggguna tersebut akan berkecenderungan mencari informasi yang berbahasa Indonesia akan tetapi informasi yang didapat sangatlah terbatas, sehingga secara otomatis akan mendorong pengguna untuk mencari informasi dari website-website berbahasa asing. Padahal belum tentu semua pengguna internet tersebut memahami bahasa inggris dengan baik, sehingga pengguna tidak dapat secara maksimal memanfaatkan informasi tersebut. Cara yang umum dilakukan oleh para pengguna internet selama ini yaitu mencari arti kata yang tidak mereka pahami tersebut melalui aplikasi kamus atau *google translate*. Hal tersebut sangat tidak efisien untuk dilakukan oleh *user*, karena pengguna harus membuka dua beberapa aplikasi secara bersamaan yaitu *browser* dan aplikasi kamus atau membuka beberapa halaman website secara bersamaan.

Saat ini pertumbuhan *browser* modern dan halaman web biasanya menggunakan banyak fitur dan teknik yang tidak ada pada masa-masa awal *web*, dikarenakan adanya persaingan antar pengembang *browser*. Sehingga pengembang *browser* saling berlomba-lomba untuk mengembangkan fitur-fitur menarik pada browser mereka. *Mozilla* salah satu pengembang *browser firefox* memahami persaingan ini, *mozilla* merupakan pengembang *browser* yang berbasis *open source*, sehingga *mozilla* memperbolehkan semua orang untuk berpartisipasi dalam mengembangkan *browser Mozilla firefox*, misalnya mengembangkan *plug-in*, *add-ons*, dan sebagainya.

Tugas akhir ini bertujuan untuk memberikan solusi pada permasalahan tersebut, yaitu pengembangan suatu aplikasi yang diinstal ke dalam *browser Mozilla firefox* untuk menerjemahkan isi dari *website* secara otomatis, sehingga pengguna dapat memperoleh informasi yang diinginkan secara optimal.

Aplikasi yang dikembangkan pada tugas akhir ini memiliki kelebihan yang belum ditemui pada aplikasi sejenis. Kelebihannya terletak pada kemampuan menerjemahkan isi website secara *context sensitive*, sehingga hasil terjemahan yang didapatkan lebih mudah dipahami dan sesuai dengan konteks berbahasa yang baik, Sehingga pengembangan aplikasi penterjemah otomatis, *inline translator* ini membutuhkan suatu algoritma yang mampu untuk menjembatani keinginan penulis dalam mengembangkan aplikasi inline translator yang *context senstive* sebagai dasar pemikiran pengembangan aplikasi *inline translator*.

1.2. Rumusan permasalahan

Permasalahan yang akan diselesaikan dalam tugas akhir ini adalah:

1. Belum adanya otomatisasi dalam menterjemahkan *content* pada *website* asing yang *context sensitive* secara *inline*.

1.3. Batasan Permasalahan

Batasan permasalahan dalam tugas akhir ini yaitu:

1. Aplikasi yang dikembangkan berbasis *client.*
2. Aplikasi yang dikembangkan saat ini akan menampilkan hasil terjemahan sesuai yang *server* berikan.
3. Aplikasi yang dikembangkan saat ini diprioritaskan untuk menerjemahkan *content website* berbahasa inggris ke dalam bahasa indonesia.

1.4. Tujuan

Tujuan dari tugas akhir ini yaitu melakukan implementasi dalam pengembangan *browser Mozilla firefox* yang terkait dengan pengembangan fungsionalitas tambahan (*add-ons*) penerjemah otomatis.

1.5. Metodologi Penulisan

Metodologi penulisan yang digunakan dalam membuat laporan tugas akhir ini yaitu sesuai dengan pedoman penulisan tugas akhir pada jurusan Sistem Informasi dan Institut Teknologi Sepuluh Nopember Surabaya.

1.6. Sistematika Penulisan

Tugas akhir ini menggunakan sistematika penulisan laporan yang dibagi menjadi 6 bab, yaitu:

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan permasalahan, batasan masalah, tujuan, manfaat, metodologi penelitian, sistematika penulisan dari tugas akhir.

BAB II TINJAUAN PUSTAKA

Bab ini berisikan tentang dasar-dasar teori yang dapat menunjang tugas akhir dan memperkuat pembahasan yang ada.

Adapun bagian dari bab ini, yaitu definisi dari *add-ons*, *foxbeans*, XUL, *java script*, dan algoritma *porter stemmer*.

BAB III METODOLOGI PENELITIAN

Bab ini berisi langkah-langkah penelitian yang perlu ditempuh dalam menyelesaikan tugas akhir ini. Dimulai dari studi literatur dan pengumpulan data, perancangan perangkat lunak, pembuatan perangkat lunak, uji coba dan evaluasi, serta terakhir penyusunan buku tugas akhir.

BAB IV ANALISIS KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK

Bab ini berisi penjelasan mengenai tahapan perencanaan dalam pembuatan tugas akhir. Dimulai dari analisa kebutuhan hingga desain *test case*.

BAB V IMPLEMENTASI DAN UJI COBA PERANGKAT LUNAK

Pada bab ini akan dijelaskan mengenai tahapan pembuatan perangkat lunak serta uji coba dan evaluasi. Hal ini meliputi implementasi XUL, *javascript*, transfer data, dan penerapan algoritma *porter stemmer*.

BAB VI KESIMPULAN DAN SARAN

Bab terakhir ini menjelaskan tentang hasil kesimpulan dari penelitian yang telah dilakukan serta saran-saran terkait dengan pengembangan topik selanjutnya.

1.7. Relevansi dan Manfaat

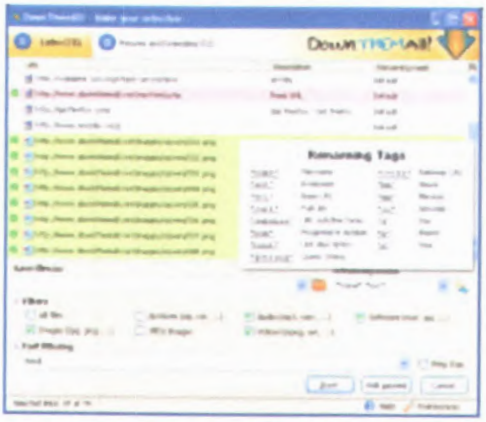
Tugas akhir ini diharapkan dapat membantu pengguna internet dalam menterjemahkan *content website* asing ke dalam bahasa Indonesia secara *inline*. Sehingga pengguna dapat memperoleh informasi yang diinginkan dengan cepat, efektif dan efisien.

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan diuraikan konsep dan teknologi yang akan digunakan dalam menyelesaikan tugas akhir ini. Teknologi yang digunakan antara lain adalah *add-ons*, *foxbeans*, *javascript*, XUL dan algoritma *porter stemmer*.

2.1. Add Ons

Add-ons merupakan suatu aplikasi yang dikembangkan untuk memberikan nilai tambah bagi *browser*. *Add-ons* dapat ditambahkan kedalam *browser* dengan cara menginstal, sehingga fungsionalitas tambahan yang diinginkan oleh pengguna dapat dipenuhi oleh *browser*. Pengguna dapat memanfaatkan fungsionalitas tambahan sesuai dengan kebutuhannya. *Add-ons* dapat ditemukan pada *browser mozilla firefox*, selain itu *add-ons* juga dapat ditemukan pada aplikasi *IDE editor Netbeans* serta CMS yang berbasis web.



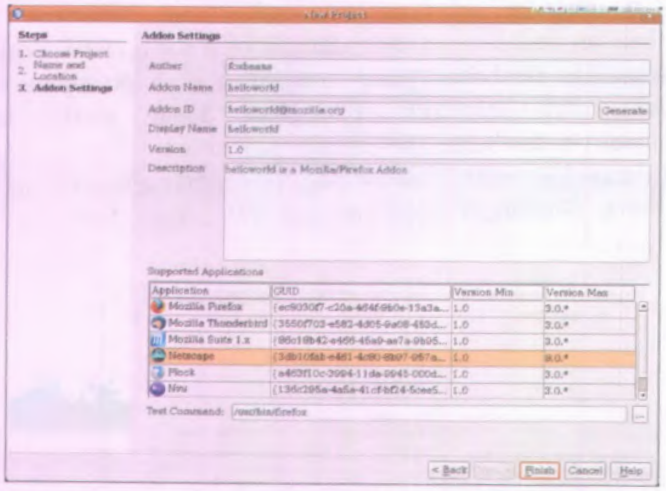
Sumber : https://addons.mozilla.org/en-US/firefox/addon/downthemall/

Gambar 2.1 ***Screen shoot add-ons mozilla firefox***

Aplikasi *addons* dapat di unduh dengan mudah sesuai dengan kebutuhan pengguna, seperti terlihat pada gambar 2.1 yang merupakan salah satu contoh *addons* yang dapat mengunduh seluruh thema yang disediakan oleh *mozilla.*

2.2. Foxbeans

Foxbeans merupakan suatu aplikasi yang dapat ditambahkan ke dalam *IDE editor*, biasanya aplikasi ini ditambahkan ke dalam IDE *editor netbeans*. *Foxbeans* dapat mendukung *netbeans* untuk mengembangkan *add-ons* pada *firefox*, *thunderbird*, *flock*, *netbeans*, dan sebagainya.



Sumber : http://www.teesoft.info/content/view/38/1/lang,en/

Gambar 2.2 *Screen shoot plug-in foxbeans*

Pada gambar 2.2, *Foxbeans* yang telah di instal ke dalam IDE *netbeans* akan memberikan langkah-langkah yang mudah untuk membuat aplikasi *addons*, seperti nama pembuat *addons*, nama *addons*, ID *addons*, dan sebagainya.

2.3. XUL

XUL - XML *User Interface Language*, dalam pemrograman komputer termasuk dalam bahasa *mark up* yang di kembangkan oleh *Mozilla.* XUL dapat dijalankan pada aplikasi *cross-platform* seperti *Mozilla firefox* dan *flock.* Mesin yang digunakan oleh Mozilla menyediakan implementasi XUL dalam *browser Mozilla firefox* contohnya dalam membangun aplikasi *addons*.

XUL bergantung pada beberapa standar *web* dan teknologi *web* yang sudah ada, termasuk CSS, *JavaScript*, dan DOM sehingga hal tersebut membuat XUL relatif mudah untuk dipelajari bagi setiap orang dengan latar belakang pemrograman web desain. Umumnya dalam mengembangkan atau membangun aplikasi dengan antarmuka berbasis XUL terdiri dari tiga komponen utama yaitu :

- *Content* : biasanya berisi berkas-berkas (*file*) yang memuat pengaturan dan tata letak antarmuka dalam bentuk XUL, dan memuat kontrol antar muka. Berkas-berkas yang merupakan isi dari komponen *content* umumnya berekstensi *.xul, *.js, dan sebagainya.
- *Skin* : biasanya berisi berkas-berkas yang digunakan untuk mengatur tampilan antarmuka suatu aplikasi, seperti gambar, pengaturan gambar, dan sebagainya. Berkas-berkas yang merupakan isi dari komponen *skin*umumnya berekstensi *.css, *.png, dan ekstensi gambar lainnya.
- *Locale* : biasanya berisi berkas-berkas yang menyimpan nilai berupa string yang nantinya akan ditampilkan pada antarmuka aplikasi. Berkas-berkas yang merupakan isi dari komponen *content* umumnya berekstensi *.dtd, *.properties, dan sebagainya

Ada beberapa unsur pada XUL jika ditinjau secara garis besar, yaitu :

- ***top-level elements***
  window, page, dialog, wizard, dan sebagainya.
- ***widgets***
  label, button, text box, list box, combo box, radio button, check box, tree, menu, toolbar, group box, tab box, colorpicker, spacer, splitter, dan sebagainya.
- ***box model***
  box, grid, stack, deck, dan sebagainya.
- ***events* dan *scripts***
  script, command, key, broadcaster, observer, dan sebagainya.
- ***data source***
  template, rule, dan sebagainya.
- **lainnya**

  overlay, iframe, browser, editor, dan sebagainya.

```
<? xml version = "1.0"?>
<? xml-stylesheet href = "chrome: / / global / skin /" type = "text / css"?>

<jendela id = "contoh vbox" title = "Teladan"
xmlns = "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <vbox>
    <button id = "yes" label = "Yes" />
    <button id = "no" label = "No" />
    <button id = "maybe" label = "Maybe" />
  </ vbox>
</ window>
```

Sumber : http://en.wikipedia.org/wiki/XUL

Gambar 2.3 Contoh penggunaan kode *XUL*

Contoh penggunaan XUL pada gambar 2.3 tersebut dimaksudkan untuk membuat suatu tampilan aplikasi berupa *window* yang memiliki beberapa *button.*

2.4. Javascript

Sejarah *javascript* di mulai ketika pertama kali diperkenalkan kepada public oleh *Netscape* pada tahun 1995.

Awal kemunculan *javascript* versi pertama tidak langsung menggunakan nama *javascript* akan tetapi menggunakan nama *LiveScript,* yaitu bahasa pemrograman sederhana yang dikembangkan oleh *Netscape* untuk membangun *browser Netscape Navigator 2* yang pada saat itu sangat popular. Pada waktu yang bersamaan *Netscape* sedang giat-giatnya melakukan kerjasama dengan *Sun* yang merupakan pengembang bahasa pemrograman *java,* sehingga membuat Netscape memberikan nama *javascript* untuk menggantikan nama *livescript* pada tangga; 4 Desember 1995. Pada saat yang bersamaan pabrik *software* raksasa *Microsoft* mencoba untuk mengadaptasi teknologi ini ke dalam browser milik mereka yaitu Internet Explorer 3, bahasa yang digunakan untuk mengembangkan IE pun diberi nama *JScript*.

Pengertian dari *javascript* yaitu suatu bahasa pemrograman sederhana berbasis prototype yang berjalan pada sisi klien atau secara sederhana dapat dikatakan bahwa *javascript* merupakan bahasa pemrograman yang hanya dapat dijalankan khusus untuk *browser* atau halaman website untuk memberikan kesan atraktif. Secara fungsional, *javascript* digunakan untuk menyediakan akses *script* pada objek yang dibenamkan (*embedded*). Contoh sederhana dari penggunaan *javascript* adalah membuka halaman *popup*, fungsi validasi pada form sebelum data dikirimkan ke server, dan sebagainya.

Hal terpenting dalam bahasa pemrograman *javascript* yaitu merupakan bahasa pemrograman yang *case sensitive*, sehingga dituntut ketelitian dalam melakukan *coding*. Penulisan perintah dalam *javascript* harus benar-benar diperhatikan besar-kecilnya huruf, Contohnya penulisan fungsi var tidak boleh ditulis Var dan juga tidak oleh ditulis VAR (huruf besar semua), yang benar adalah var (huruf kecil semua). Perintah lain adalah new Date tidak boleh ditulis new date (huruf kecil semua), dan sebagainya.

JavaScript bekerja pada *browser*, untuk menampilkan halaman *web*, *user* menuliskan alamat *web* pada *address bar url*,

kemudian *browser* mencari file html pada *server* sesuai dengan alamat yang diketikkan oleh *user* pada *url*. Selanjutnya file yang telah ditemukan tersebut ditampilkan pada *browser*. Kemudian file *JavaScript* yang terdapat pada *browser* melakukan tugasnya yaitu memberikan respon dengan cepat jika ada perintah dari *user* dan halaman *website* akan jauh lebih responsif. *Javascript* dapat melakukan sesuatu yang tidak dapat dilakukan oleh bahasa pemrograman web lainnya seperti HTML, PHP, dan CSS yaitu melakukan respon yang cepat seperti apa yang diinginkan oleh user.

Implementasi pemrograman web yang memanfaatkan pemrograman *JavaScript* terpopuler saat ini yaitu pemrograman web dengan menggunakan AJAX (*Asynchronous JavaScript and XMLHTTP*). AJAX biasanya dapat ditemukan dalam aplikasi yang berbasis *web* seperti Gmail, Google Reader, dan lain-lain. Kelebihan AJAX yaitu interaksi antara *user* dan aplikasi web yang sangat responsif saat terjadinya pertukaran data antara *server* dan *browser*.

Javascript sama halnya dengan bahasa pemrograman lainnya yang memiliki kelebihan dan kekurangan dalam pengimplementasiannya. Kelebihan dari bahasa pemrograman *javascript* yaitu *javascript* memiliki ukuran yang kecil sehingga ketika web tidak perlu lagi di olah oleh *server* ketika *browser* memanggil web dari sebuah *server*. Selain itu, kelebihan lain dari *javascript* yaitu mudah untuk dipelajari karena *javascript* merupakan turunan dari bahasa pemrograman *java* selain itu *javascript* tidak terikat oleh *hardware* maupun *software* tertentu bahkan sistem operasi tertentu seperti windows maupun unix. Karena *javascript* bersifat terbuka, maka *javascript* dapat dibuat maupun dibaca oleh semua jenis komputer. Kekurangan javascript bersifat client side, maka script yang kita buat di text editor dan telah dijadikan web di server, ketika user me-request web dari server tersebut maka sintak javascript mampu membuat bentuk web menjadi interaktif dan dinamis, namun javascript

tidak mampu membuat kelas-kelas yang bisa menampung objek-objek tambahan seperti java karena javascript teleh memiliki objek yang built-in pada sturktur bahasanya.

Selain kelebihan, *javascript* juga memiliki kekurangan yaitu *javascript* tidak terenkripsi karena *javascript* bersifat *client side*, maka *script* yang kita buat di text editor dan telah dijadikan web pada *server*, dan saat pengguna melakukan *request* web dari server tersebut, maka sintaks *javascript* akan langsung ditampilkan pada browser sehingga pengguna dapat melihat dan meniru dari source dari web tersebut. *Javascript* memiliki Kemampuan terbatas walaupun *javascript* mampu untuk membuat *website* menjadi interaktif dan dinamis, namun *javascript* tidak mampu membuat program aplikasi sendiri seperti yang bisa dilakukan oleh bahasa pemrograman *Java*. Selain itu, *javascript* memiliki Keterbatasan objek sehingga *javascript* tidak mampu membuat kelas-kelas yang bisa menampung objek-objek tambahan seperti halnya bahasa pemrograman *java*, karena *javascript* telah memiliki objek yang *built-in* pada struktur bahasanya.

2.5. Algoritma Porter Stemmer

Porter stemmer merupakan algoritma yang digunakan untuk menghilangkan akhiran *morphological* dan *infleksional* yang umum dari bahasa Inggris. Algoritma *Porter stemmer* ini terdiri dari himpunan kondisi atau *action rules*.

Sebuah konsonan merupakan suatu huruf selain huruf a,i,u,e,o, dan y (huruf y disini secara spesifik maksudnya adalah huruf y yang didahului oleh huruf-huruf konsonan sampai pada batasan tertentu), perbedaan perilaku huruf y jika dikatakan sebagai konsonan dan huruf y jika dikatakan sebagai vokal dapat dilihat pada contoh dibawah ini :

- **Y yang bersifat konsonan**
  Kata TOY yang merupakan huruf konsonan adalah huruf T dan Y, alasannya karena sebelum huruf Y bukan

merupakan huruf konsonan sehingga huruf Y disini merupakan **huruf konsonan dan bukan huruf vokal**.

- **Y yang bersifat vocal**
  Kata SYZYGY yang merupakan huruf konsonan adalah huruf S, Z dan G, alasannya karena sebelum huruf Y merupakan huruf konsonan sehingga huruf Y disini merupakan **huruf vokal dan bukan huruf konsonan**.

Jika huruf tersebut bukan termasuk huruf konsonan (selain huruf konsonan), maka huruf tersebut disebut *vowel*.

Huruf konsonan akan dinotasikan sebagai c sedangkan yang bukan konsonan akan dinotasikan sebagai v. rangkaian huruf konsonan ccc... yang panjangnya lebih besar dari 0 dinotasikan sebagai C, sedangkan rangkaian huruf *vowel* vvv... yang panjangnya lebih besar dari 0 dinotasikan sebagai V. tiap kata, bagian dari sebuah kata umumnya memiliki 4 pola sebagai berikut :

- **CVCV...C**
- **CVCV...V**
- **VCVC...C**
- **VCVC...V**

Dari keempat pola tersebut dapat disederhanakan menjadi 1 bentuk kata umum yang berpola

[C] VCVC...[V]

Dimana tanda kurung siku menunjukkan bahwa pola yang terdapat didalamnya dapat dilakukan penyesuaian, sehingga akan didapatkan pola umum dari suatu kata yaitu :

[C] (VC) {m} [V]

m menotasikan banyaknya perulangan VC yang terjadi pada pola kalimat. m merupakan representasi *measure* pada suatu

kata atau bagian dari suatu kata. Sehingga dapat disimpulkan pola kata umum berbentuk seperti :

$$[C] (VC)^m [V]$$

Himpunan kondisi dalam *Porter stemmer* dikelompokkan menjadi tiga kelas, yakni :

- **Kondisi pada stem**

  Ukuran (*measure*), dinotasikan dengan m, dari sebuah *stem* berdasarkan pada urutan vokal-konsonan.

  + m= 0, contoh : *TR, EE, TREE, Y, BY*

  + m= 1, contoh : *TROUBLE, OATS, TREES, IVY*
  + m= 2, contoh : *TROUBLES, PRIVATE, OATEN*

  kondisi-kondisi yang terjadi pada proses *stemming* selain kondisi seperti yang dijabarkan sebelumnya dapat berupa kondisi sebagai berikut :

  S berarti *stem* berakhir dengan huruf S (dan huruf-huruf yang lain

  V berarti *stem* mengandung sebuah vowel

  ***d** berarti *stem* diakhiri dengan konsonan ganda (Misalnya : -TT, -SS).

  ***o** berarti *stem* diakhiri dengan **cvc** dimana c yang kedua bukan merupakan huruf W, X, atau Y (misalnya –WIL, -HOP)

- **Kondisi pada suffix**

  (current_suffix == pattern)

- **Kondisi pada rule**

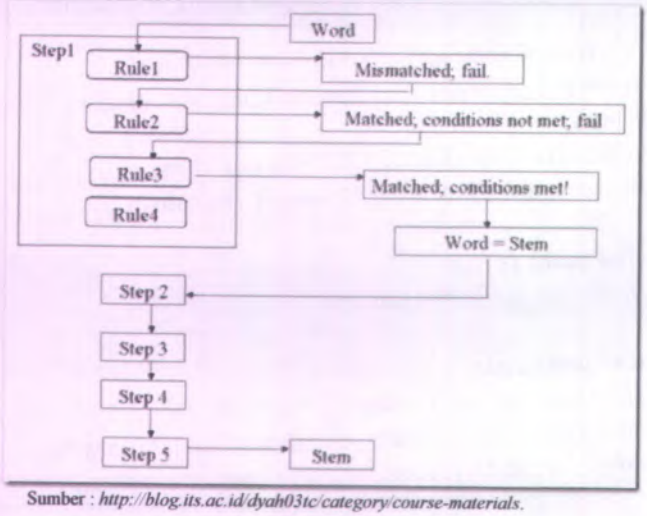
  Aturan-aturan (*rules*) dibagi menjadi beberapa langkah. Aturan-aturan (*rules*) dalam sebuah *step* diuji secara berurutan, dan hanya 1 rule dari suatu *step* saja yang diterapkan.

```
{
step1a(word);
step1b(stem);
if (the second or third rule of step 1b was used) step1b1(stem);
step1c(stem);
step2(stem);
step3(stem);
step4(stem);
step5a(stem);
step5b(stem);
}
```

Sumber : *http://blog.its.ac.id/dyah03tc/category/course-materials.*

Gambar 2.4 Kondisi pada *rules*

Gambar 2.4 menjelaskan tentang alur dari algoritma porter stemmer dalam bentuk logika sederhana, yang di kembangkan sesuai dengan rule-rule yang telah ditentukan.



Sumber : *http://blog.its.ac.id/dyah03tc/category/course-materials.*

Gambar 2.5 *Control flow* algoritma *Porter Stemmer*

Gambar 2.5 merupakan flowchart dari algoritma *porter stemmer*, yaitu alur yang membentuk pola pikir yang digunakan dalam menentukan langkah-langkah yang terjadi dan memenuhi aturan-aturan yang ditetapkan dalam algoritma tersebut. Untuk lebih memahami aturan-aturan yang berlaku dalam algoritma porter stemmer ini serta kondisi-kondisi yang harus dipenuhi, maka perhatikan langkah-langkah di bawah ini :

Tabel 2.1 ***Remove plural suffixation (step 1A)***

Conditions	Suffix	Replacement	Examples
Null	sses	Ss	caresses→ caress
Null	ied^{+}/ies*	I	ponies → poni ties → tie
Null	us^{+}/ss	Ss	caress→ caress
Null	s	Null	cats → cat

Cari akhiran terpanjang di antara akhiran-akhiran yang tercantum pada **tabel 2.1** di atas, maka sistem akan melakukan tindakan sesuai dengan kondisi yang ditunjukkan, jika akhiran yang ditemukan adalah **sses** maka akhiran tersebut akan digantikan menjadi **ss**, misalnya jika sistem menemukan kata **caresses** maka sistem akan menggantinya menjadi **caress**. Apabila sistem menemukan kata berakhiran **ied** atau **ies** maka akhiran tersebut akan digantikan dengan akhiran i apabila akhiran tersebut didahului lebih dari satu kata, seperti pada kata **ponies** maka akan menjadi **poni**. Apabila sistem menemukan akhiran **ss** maka sistem tetap mempertahankan akhiran tersebut, sedangkan jika ditemukan akhiran **s** maka sistem akan menghilangkan kata tersebut kecuali jika ada 1 huruf vocal pada sebelum akhiran s tersebut, maka sistem akan mempertahankan **s** sebagai akhiran.

Tabel 2.2 ***Remove verbal inflection(Step 1B)***

Conditions	Suffix	Replacement	Examples
(m > 0)	eed/eedly^{+}	Ee	feed → feed agreed → agree
(*v*)	ed/ edly^{+}	Null	plastered→ plaster
(*v*)	ing/ingly^{+}	Null	motoring→ motor sing → sing

Cari akhiran terpanjang di antara akhiran-akhiran pada **tabel 2.2** di atas, maka sistem akan melakukan tindakan sesuai dengan kondisi yang ditunjukkan, jika akhiran yang ditemukan adalah eed atau eedly maka akhiran tersebut akan digantikan menjadi ee jika dalam kondisi R1, yaitu kondisi dimana huruf vocal pertama diikuti huruf konsonan yang kemudian diikuti oleh huruf vocal atau jika akhir dari kata tersebut tidak ada huruf konsonan. Apabila sistem menemukan kata berakhiran ed/edly atau ing/ingly maka akhiran tersebut akan dihapus.

Tabel 2.3 ***Continued for -ed and -ing rules*** **(Step 1b1)**

Conditions	Suffix	Replacement	Examples
Null	at	Ate	conflat(ed)→conflate
Null	bl	Ble	troubl(ing)→trouble
Null	iz	Ize	siz(ed) → size

(*d and not (*<L>or *< S> or * <Z>))	null	single letter	hopp(ing)→hop tenn(ed)→tan fall(ing)→fall hiss(ing)→hiss fizz(ed)→fizz
(m=1 and *o)	null	E	fill(ing)→fill fil(ing)→file

Cari akhiran terpanjang di antara akhiran-akhiran pada **tabel 2.3** di atas Jika bagian kata sebelumnya berisi huruf vocal dan setelah diatas proses penghapusan jika terjadi kondisi dimana kata tersebut berakhiran **at, bl, iz** maka tambahkan huruf **s**, atau jika akhiran tersebut terdiri dari **dua huruf yang sama** maka hapus salah satu dari huruf tersebut. Atau jika akhiran tersebut terlalu pendek maka tambahkan huruf **e** sebagai akhiran.

Tabel 2.4 *y and i* (Step 1c)

Conditions	Suffix	Replacement	Examples
(*v*)	Y/y	I	happy→happi sky→sky

Jika akhiran kata adalah y seperti yang tertera pada **tabel 2.4** diatas maka ganti akhiran tersebut menjadi I, dengan syarat jika akhiran kata tersebut didahului oleh kata konsonan yang bukan huruf pertama dari sebuah kata.

Tabel 2.5 *Peel one suffix off for multiple suffixes (*Step 2)

Conditions	Suffix	Replacement	Examples
(m > 0)	ational	Ate	relational→ relate
(m > 0)	tional	Tion	conditional→ condition rational→ rational
(m > 0)	enci	Ence	valenci→ valence
(m > 0)	anci	Ance	hesitanci→ hesitance
(m > 0)	izer	Ize	digitizer→ digitize
(m > 0)	abli	Able	conformabli→ conformable

(m > 0)	alli	Al	radicalli→ radical
(m > 0)	ently	Ent	differently→ different
(m > 0)	eli	E	vileli→vile
(m > 0)	oush ously	Ous	analogoush→ analogous
(m > 0)	ization	Ize	vietnamization→vietnamize
(m > 0)	ation	Ate	predication→ predicate
(m > 0)	ator	Ate	operator→ operate
(m > 0)	alism	Al	feudalism→ feudal
(m > 0)	iveness	Ive	decisiveness→ decisive
(m > 0)	fullness fulli+	Ful	hopefullness→ hopefull
(m > 0)	ousness	Ous	callousness→ callous

(m > 0)	aliti	Al	formaliti→formal
(m > 0)	iviti	Ive	sensitivity→ sensitive
(m > 0)	bility bli+	Ble	sensibility→ sensible

Cari akhiran terpanjang di antara akhiran-akhiran seperti yang tertera pada **tabel 2.5** di atas, maka sistem akan melakukan tindakan sesuai dengan kondisi yang ditunjukkan dan akhiran-akhiran tersebut berada dalam kondisi **R1**, maka yaitu kondisi dimana huruf vocal pertama diikuti huruf konsonan yang kemudian diikuti oleh huruf vocal atau jika akhir dari kata tersebut tidak ada huruf konsonan. Maka akhiran-akhiran tersebut akan mengalami perubahan seperti yang tertera pada tabel 2.5

Tabel 2.6 *Step 3*

Conditions	Suffix	Replacement	Examples
(m > 0)	icate	Ic	triplicate→triplic
(m > 0)	ative*	Null	formative→form
(m > 0)	alize	Al	formalize→formal
(m > 0)	iciti	Ic	electriciti→electric
(m > 0)	ical	Ic	electrical→electric
(m > 0)	ful	Null	hopeful→hope
(m > 0)	ness	Null	goodness→good

Cari akhiran terpanjang di antara akhiran-akhiran yang tertera pada **tabel 2.6** di atas, maka sistem akan melakukan tindakan sesuai dengan kondisi yang ditunjukkan dan akhiran-akhiran tersebut berada dalam kondisi **R1**, maka yaitu kondisi dimana huruf vocal pertama diikuti huruf konsonan yang kemudian diikuti oleh huruf vocal atau jika akhir dari kata tersebut tidak ada huruf konsonan. Maka akhiran-akhiran tersebut akan mengalami perubahan seperti yang tertera pada **tabel 2.6**, untuk kata berakhiran **ative**, jika kata tersebut dalam kondisi **R2,** yaitu kondisi dimana huruf vocal pertama diikuti oleh huruf konsonan yang termasuk dalam kondisi **R1** atau akhir dari kata tersebut tidak ada huruf konsonan maka akhiran tersebut akan di hapus.

Tabel 2.7 ***Delete last suffix***

Conditions	Suffix	Replacement	Examples
(m > 0)	Al	Null	revival→reviv
(m > 0)	ance	Null	allowance→allow
(m > 0)	ence	Null	inference→infer
(m > 0)	Er	Null	airliner→airlin
(m > 0)	Ic	Null	gyroscopic→ gyroscop
(m > 0)	able	Null	adjustable→adjust
(m > 0)	ible	Null	defensible→ defense
(m > 0)	ant	Null	irritant→irrit
(m > 0)	ement	Null	replacement→replac
(m > 0)	ment	Null	adjustment→adjust
(m > 0)	ent	Null	dependent→ depend
(m > 0)	Ion	Null	adoption→adopt

(m > 0)	Ou	Null	homologou→homolog
(m > 0)	Ism	Null	communism→commun
(m > 0)	Ate	Null	activate→active
(m > 0)	Iti	Null	angularity→anggular
(m > 0)	ous	Null	homologous→homolog
(m > 0)	Ive	Null	effective→effect
(m > 0)	Ize	Null	bowdierize→bowdier

Cari akhiran terpanjang di antara akhiran-akhiran seperti yang tertera pada **tabel 2.7**di atas, maka sistem akan melakukan tindakan sesuai dengan kondisi yang ditunjukkan dan akhiran-akhiran tersebut berada dalam kondisi **R2**, maka yaitu kondisi dimana huruf vocal pertama diikuti huruf konsonan yang kemudian diikuti oleh huruf vocal atau jika akhir dari kata tersebut tidak ada huruf konsonan. Maka akhiran-akhiran tersebut akan mengalami perubahan seperti yang tertera pada **tabel 2.7**, untuk kata berakhiran **ion**, jika akhiran tersebut didahului oleh huruf **s** atau **t** maka akhiran tersebut tersebut akan di hapus.

Tabel 2.8 Remove e

Conditions	Suffix	Replacement	Examples
(m > 1)	E	Null	probate→probat rate→rate
(m = 0 and not *o)	E	Null	cease→ceas

Cari akhiran seperti kondisi di atas yang tertera pada **tabel 2.8** dan jika ditemukan, maka lakukan tindakan sesuai dengan kondisi akhiran tersebut. Jika kata tersebut diakhiri dengan huruf **e** dan dalam kondisi berada dalam kondisi **R2**, atau dalam kondisi **R1** dan tidak didahului oleh suku kata pendek maka hapus akhiran tersebut.

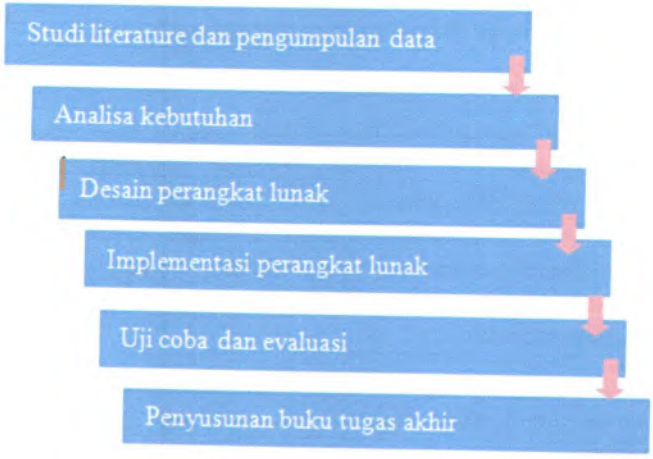
Tabel 2.9 Reduction

Conditions	Suffix	Replacement	Examples
(m = 1 and *d and *<L>)	Null	Single letter	controll→control roll→roll

Cari akhiran seperti kondisi di atas yang tertera pada **tabel 2.9** dan jika ditemukan, maka lakukan tindakan sesuai dengan kondisi akhiran tersebut. jika terdapat akhiran l pada suatu kata dan dalam kondisi R1 dan diawali oleh huruf l maka hapus akhiran tersebut.

BAB III
METODOLOGI PENELITIAN

Metodologi penelitian merupakan hal yang sangat penting dalam pengerjaan tugas akhir. Adanya metodologi penelitian, menyebabkan pengerjaan tugas akhir akan lebih terarah dan sistematis. Langkah-langkah yang perlu ditempuh dalam menyelesaikan tugas akhir ini dapat dilihat pada Gambar 3.1 di bawah ini :



Gambar 3.1 Alur Metodologi Penelitian

Alur metodologi penelitian seperti yang tertera pada gambar 3.1 akan dijelaskan lebih detal pada sub bab berikutnya.

3.1. Studi Literatur dan Pengumpulan Data

Studi literatur dilakukan untuk pemahaman konsep, teori, dan beberapa teknologi yang akan digunakan. Referensi yang akan digunakan dalam tugas akhir ini, antara lain:

a. *Text book* yang membahas tentang teori algoritma *porter stemmer.*

b. Dokumentasi yang membahas tentang *Javascript, XUL.*

c. Aplikasi sejenis yang lebih dulu ada.

Pengumpulan data dilakukan untuk mengetahui proses yang terjadi dalam sistem aplikasi, serta kebutuhan dari para pengguna, dalam hal ini pengguna aplikasi *add-ons*.

3.2. Analisa Kebutuhan

Analisa kebutuhan merupakan suatu fase awal yang dilakukan dalam melakukan metode penelitian. Analisa kebutuhan yang di maksud yaitu berupa analisa kebutuhan aplikasi. Fase analisa kebutuhan aplikasi dilakukan untuk mengidentifikasi kebutuhan yang relevan dengan permasalahan yang dihadapi. Analisa kebutuhan aplikasi akan dijabarkan detail pada bab tersendiri.

Pada tahapan ini akan dilakukan analisa kebutuhan berdasarkan hasil studi literatur dan pengumpulan data. Analisa kebutuhan yang dilakukan meliputi kebutuhan pengguna, kebutuhan fungsi, dan kebutuhan teknologi.

Analisa kebutuhan pengguna dilakukan untuk mendefinisikan siapa saja pengguna dari sistem yang akan dibuat. Kemudian setelah analisa kebutuhan pengguna dilakukan, langkah selanjutnya yaitu melakukan analisa kebutuhan fungsi.

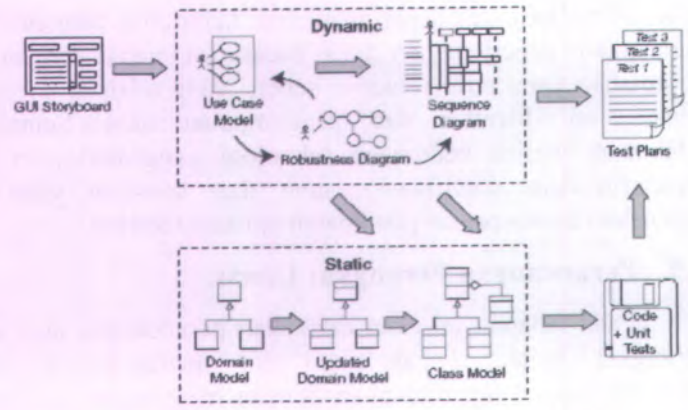
Analisa kebutuhan fungsi dilakukan untuk mendefinisikan fungsi-fungsi yang dapat diberikan oleh sistem kepada para pengguna tersebut. Fungsi-fungsi yang terdapat pada

sistem tersebut harus dapat memenuhi kebutuhan pengguna yaitu melakukan penerjemahan teks secara otomatis dengan hasil terjemahan yang *case sensitive* seperti yang telah di dapat pada tahap studi literatur dan pengumpulan data. Selanjutnya, dilakukan analisa kebutuhan teknologi yang bertujuan untuk mendefinisikan spesifikasi sistem dan *tool-tool* yang akan digunakan dalam proses pembuatan aplikasi tersebut.

3.3. Perancangan Perangkat Lunak

Pada tahapan ini akan dilakukan perancangan atau desain perangkat lunak. Tugas akhir ini menggunakan metode pengembangan yang dilakukan dengan metode *use case driven object modeling*. Istilah *use case driven* memiliki suatu definisi yang menggunakan *use case* sebagai pendorong terjadinya implementasi perangkat lunak, mulai dari pengumpulan informasi awal dan penentuan kebutuhan sampai proses implementasi aplikasi. Pemilihan metode ini dilakukan karena *use case* sangat sesuai untuk menangkap kebutuhan serta mendorong analisa, desain, dan implementasi. Gambar 3.2 menjelaskan alur proses dari metode *use case driven object modelling*.

Proses diawali dengan mendefinisikan kebutuhan-kebutuhan dari sistem. Terdapat berbagai jenis kebutuhan, namun pada level proses hanya akan dibagi berdasarkan kebutuhan fungsional dan kebutuhan non fungsional. Kebutuhan fungsional menjelaskan apa saja yang dapat dilakukan oleh sistem, sedangkan kebutuhan non fungsional menjelaskan bagaimana pengguna dan sistem akan berinteraksi. Bagi sebagian besar sistem perangkat lunak, interaksi antara pengguna dan sistem terjadi melalui *screens*, *windows*, atau *pages*. Karena itu dibuatlah *GUI storyboard*. *GUI storyboard* mendefinisikan bagaimana *user* dan sistem akan berinteraksi dalam bentuk GUI.



Sumber : Use Case Driven Object Modeling with UML

Gambar 3.2 Alur Proses Metode Use Case Drive Object Modelling

Di sisi lain, domain model juga dibuat. Domain model adalah suatu tugas membangun proyek *glossary*, istilah yang digunakan dalam suatu proyek. Domain model ini akan terus di-update sepanjang proyek, sehingga selalu mencerminkan pemahaman saat ini dari suatu permasalahan. Domain model tidak hanya di-*update* pada saat memodelkan *use case*, tetapi juga pada saat menggambarkan *robustness* diagram dan *sequence* diagram. Domain model akan berubah menjadi domain model yang di-*update*, yang pada akhirnya berubah menjadi *class* model yaitu model statis yang mendefinisikan *class-class* perangkat lunak.

Setelah *GUI storyboard*, selanjutnya akan dibuat model *use case*. *Use case* menggambarkan bagaimana pengguna akan berinteraksi dengan sistem dan bagaimana sistem tersebut akan merespon.

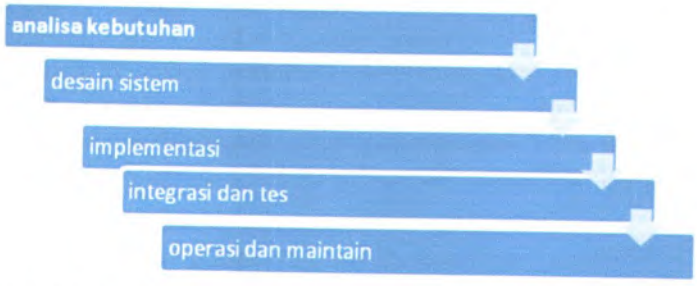
Untuk mengubah *use case* menjadi desain yang detail, yang kemudian dibuat menjadi program, *use case* perlu dihubungkan dengan objek. Hal inilah yang dilakukan pada saat *robustness analysis*. *Robustness analysis* digambarkan dalam bentuk

robustness diagram. *Robustness diagram* adalah gambar objek dari suatu *use case*, yang tujuannya untuk menyempurnakan teks use case dan model objek. Pastikan bahwa semua entitas dalam semua *robustness diagram* tampil pada domain model.

Selanjutnya akan dibuat *sequence diagram* untuk tiap *use case* yang ada. *Sequence diagram* menggambarkan secara detail bagaimana *use case* akan diimplementasikan. Fungsi utama dari *sequence diagram* adalah mengalokasikan perilaku untuk tiap class. Pada saat menggambarkan *sequence diagram*, domain model di-*update* kembali. Selain itu, akan ditambahkan operasi-operasi untuk tiap objek domain. Pada tahap inilah, domain model telah berubah menjadi class diagram.

Selanjutnya perangkat lunak akan dibuat secara langsung dari desain yang telah ada. Dari *sequence diagram* dapat dibuat *test plan*, sedangkan dari class model dapat dibuat *coding* dan unit test.

Model pengembangan aplikasi tugas akhir ini dilakukan dengan menggunakan model *waterfall*. Model *waterfall* atau disebut juga *linier sequencial model* adalah suatu proses pengembangan perangkat lunak yang bersifat sekuensial, melalui beberapa tahap yaitu: *analysis, design, coding, testing*, dan *support*. Seperti yang dapat dilihat pada gambar 3.3 dibawah ini :



Gambar 3.3 Model waterfall pada pengembangan aplikasi tugas akhir

Model *waterfall* diawali dengan melakukan analisa kebutuhan, analisa kebutuhan dilakukan untuk mengetahui apa saja yang diperlukan dalam membangun aplikasi tersebut, misalnya saja analisa kebutuhan pengguna, kebutuhan teknologi dan sebagainya. Kemudian dilanjutkan dengan melakukan desain aplikasi, setelah seluruh analisa kebutuhan dilakukan. Desain aplikasi tersebut bertujuan untuk memudahkan dalam melakukan implementasi dalam hal ini melakukan kode aplikasi yang akan dibangun.setelah desain dan implementasi di lakukan maka langkah selanjutnya yaitu mengintegrasikan seluruh modul-modul yang telah dibuat dan melakukan uji coba pada aplikasi tersebut. Apakah aplikasi telah berjalan dengan baik dan telah memenuhi seluruh kebutuhan pengguna dan sesuai dengan tujuan dalam mengembangkan aplikasi tersebut. Tahap terakhir dalam model *waterfall* seperti yang tertera pada **gambar 3.3** yaitu operasi dan *maintain*, adalah proses yang dilakukan setelah seluruh aplikasi tersebut berjalan dengan baik, misalnya melakukan update aplikasi, dan sebagainya.

3.4. Pembuatan Perangkat Lunak

Setelah dilakukan perancangan atau desain aplikasi, maka tahap selanjutnya adalah melakukan implementasi aplikasi atau membangun sistem pada suatu aplikasi yang akan dibuat. Implementasi perangkat lunak dilakukan sesuai dengan metode *use case driven*. Apabila pada saat pembuatan program ditemukan kesalahan pada desain, maka model yang dihasilkan dari tahap desain dapat di-update kembali.

Aplikasi *addons inline translator* dibuat menggunakan *IDE Netbeans* yang telah di install *foxbeans*-suatu aplikasi yang digunakan untuk membangun aplikasi berbasis *addons*, untuk membangun interface digunakan XUL sedangkan untuk

membangun proses yang terjadi dalam aplikasi dan mengintegrasikannya dengan interface digunakan bahasa pemrograman *javascript.*

3.5. Uji Coba dan Evaluasi

Pada tahap uji coba dan evaluasi hasil dari pembuatan perangkat lunak diuji coba dan bila ada kesalahan dilakukan perbaikan sampai selesai. Setelah itu dibuat kesimpulan dan didokumentasikan. Uji coba yang dilakukan meliputi:

a. Uji coba yang digunakan untuk memastikan bahwa semua kebutuhan yang diharapkan sudah terpenuhi. Uji coba seperti ini dapat dilakukan dengan menggunakan suatu *modelling tool* yang mendukung *traceability* antara *requirement* dan *usecase.* Kemudian setiap kebutuhan setidaknya memiliki sebuah *test case*.

b. Uji coba yang selanjutnya yaitu uji coba yang digunakan untuk memastikan apakah semua fungsi yang terdapat dalam sistem telah berjalan dengan baik dan benar. Uji coba tersebut menggunakan *unit test* dan *application test* sesuai dengan *test plans* yang dihasilkan dari metode *use case driven.*

3.6. Pembuatan Buku Tugas Akhir

Setiap proses yang dilakukan dalam pengerjaan tugas akhir mulai dari awal pengerjaan hingga akhir pengerjaan didokumentasikan serta ditulis dalam suatu buku laporan yang sesuai dengan aturan yang berlaku di ITS untuk kemudian dijadikan sebagai laporan tugas akhir.

BAB IV
ANALISA KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK

Bagian ini berisi pembahasan mengenai tahapan perencanaan atau pembuatan desain aplikasi tugas akhir. Analisa kebutuhan diawali dengan analisa kebutuhan pengguna serta membuat spesifikasi kebutuhan sistem. Jika analisa kebutuhan telah terpenuhi maka desain aplikasi yang berupa diagram *use case* dari kebutuhan tersebut dibuat, selanjutnya desain ini akan digunakan dalam implementasi perangkat lunak. Tugas akhir ini menggunakan *use case diagram, robustness diagram, sequence diagram, prototype GUI* dan *test case*.

4.1. Deskripsi Umum Sistem

Sistem yang dibangun pada tugas akhir ini menitik beratkan pada sistem yang terdapat pada *client inline translator.* Sistem yang dibuat terbagi menjadi dua, yaitu sistem yang digunakan oleh pengguna untuk memberikan informasi kepada server serta sistem yang menampung informasi yang diberikan oleh server untuk diteruskan kepada pengguna sebagai hasil *feedback* dari sistem sebelumnya. Sistem aplikasi pada tugas akhir ini digunakan sebagai penambah fungsionalitas kinerja *browser Mozilla firefox*, sehingga *browser* yang telah diinstal dengan *add-ons* tersebut memiliki nilai tambah dibandingkan dengan sebelum dilakukan penginstalan *add-ons*. Pengguna dapat dengan mudah menerjemahkan isi *website* secara otomatis.

menggunakan aplikasi tersebut sesuai petunjuk penggunaan aplikasi.


```
<!-- status bar -->

  <statusbar id="status-bar">
    <statusbarpanel id="inlinetrans-status-bar"
    context="inlinetransContextMenu"
    label="inlinetrans"
    tooltiptext="addons dalam kondisi aktif"
    image="chrome://inlinetrans/skin/imagesOn.png"
    class="statusbarpanel-iconic-text"
    status="enabled"
    onclick="overlay.clickIcon(event);"
    >

    </statusbarpanel>
  </statusbar>
```

Gambar 5.3 *Statusbar* sebelum diklik kanan

Gambar 5.3 diatas merupakan potongan kode xul yang digunakan untuk membangun antar muka *status bar* sebelum pengguna melakukan klik kanan pada *status bar*. Selain itu potongaan kode tersebut juga digunakan untuk membangun antar muka *status bar* apabila pengguna menekan *icon* untuk menyalakan atau mematikan aplikasi *addons*.

5.2.1.3.2 *Statusbar* setelah diklik kanan.

Antar muka *statusbar* setelah diklik kanan dapat membantu pengguna untuk mengaktifkan maupun tidak mengaktifkan *addons* tanpa perlu membuka menu *tools* → *addons* yang terdapat pada *browser mozilla firefox.*

Gambar 5.4 *Statusbar* setelah diklik kanan

Gambar 5.4 diatas merupakan potongan kode xul yang digunakan untuk membangun antar muka *status bar* saat pengguna melakukan klik kanan pada *icon*. Ada dua antar muka yang dapat di lihat oleh pengguna yaitu status aplikasi *addons* apakah dalam kondisi aktif atau tidak aktif kemudian antar muka yang berisi informasi aplikasi.

5.2.1.4. Antar muka informasi aplikasi

Implementasi antar muka informasi aplikasi merupakan antar muka yang akan muncul saat pengguna melakukan klik menu *about.* antar muka ini menyediakan segala informasi yang berkaitan dengan aplikasi.

Gambar 5.5 antar muka informasi aplikasi

Gambar 5.5 diatas merupakan potongan kode xul yang digunakan untuk membangun antar muka saat pengguna memilih menu *about*. Pengguna dapat memperoleh informasi terkait cara menggunakan aplikasi, dan sebagainya.

5.2.1.5. Antar muka hasil terjemahan

Implementasi antar muka hasil terjemahan merupakan antar muka yang akan muncul saat aplikasi menampilkan hasil terjemahan yang diinginkan oleh pengguna, antar muka hasil terjemahan berupa *pop up*.

5.2.1.6. Antar muka peringatan

Implementasi antar muka peringatan merupakan antar muka yang akan muncul saat pengguna tidak melakukan pengeblokan pada halama website, antar muka peringatan berupa *pop up*.

5.2.2. Implementasi Pengolahan Kata

Bagian ini menjelaskan implementasi dalam sistem yang berhubungan dengan pengolahan kata dalam sistem aplikasi *inline translator*. Ada beberapa bagian penting dalam sistem yang berkaitan dengan pengelolaan kata, seperti proses pemilihan kata,

proses mendapatkan *content website,* proses menghilangkan kata-kata yang frekuensi kemunculannya tinggi, dan proses stemming.

5.2.2.1. Proses mendapatkan kata yang di pilih

Proses awal pengolahan kata di mulai saat pengguna melakukan pemilihan kata dengan cara memblok kata yang ingin diketahui terjemahannya dalam hal ini arti kata bahasa Indonesia dari kata tersebut. Berikut ini merupakan fungsi yang digunakan untuk mendapatkan kata yang ingin diterjemahkan.

```
/** untuk mendapatkan kata yang di blok **/
userSelection = document.commandDispatcher.focusedWindow.getSelection().toString();
```

Gambar 5.6 Proses mendapatkan kata

Gambar 5.6 diatas merupakan potongan kode javascript yang digunakan untuk mendapatkan nilai dari kata yang di pilih oleh pengguna sebagai informasi bagi sistem untuk melangkah ke proses selanjutnya.

5.2.2.2. Proses mendapatkan posisi kata yang di pilih

Proses mendapatkan posisi kata yang di pilih dilakukan saat *user* melakukan pengeblokan kata, proses ini dilakukan untuk mendapatkan posisi kata yang tepat sesuai dengan posisi kata yang di blok. Tujuan dari proses ini yaitu untuk mendapatkan kata-kata yang letaknya sebelum dan sesuadah kata yang di blok dengan tepat, apabila terdapat kata yang sama dengan kata yang diblok pada baris atau pun paragraph yang berbeda.

```
/** untuk mendapatkan range kata yang di blok serta mengubah content kata yang di blok**/
range = document.commandDispatcher.focusedWindow.getSelection().getRangeAt(0);
range.deleteContents();
textNode = document.createTextNode(word);
range.insertNode(textNode);


/** untuk mengembalikan content website yang dipilih ke bentuk awal **/
      range.selectNode(textNode);
      range.deleteContents();
      range.insertNode(document.createTextNode(userSelection));
      userSelection = " ";
```

Gambar 5.7 Proses mendapatkan posisi kata

Gambar 5.7 diatas merupakan potongan kode javascript yang digunakan untuk mendapatkan posisi dari kata yang di pilih oleh pengguna agar memudahkan sistem untuk mendapatkan kata-kata yang letaknya berada sebelum dan sesudah kata yang di pilih dengan tepat.

5.2.2.3. Proses mendapatkan seluruh *content* website

Proses mendapatkan *content* website dilakukan setelah *user* melakukan pengeblokan kata, proses ini dilakukan untuk mendapatkan seluruh *content* website agar mudah melakukan proses mendapatkan kata-kata yang letaknya sebelum dan


```
/** untuk menghilangkan beberapa pattern yang dianggap tidak penting **/
var str = appcontent.replace(/<\/?[^>]+(>|$)/g, "");
var emailPattern = /[_a-zA-Z0-9\.]+@[\.a-zA-Z0-9]+\.[a-zA-Z]+/gi;
var urlPattern = /[a-z]+:\/\/[^\s]+/gi;
var numberOrSymbolPattern = /[0-9\.,!@#$%\^\&*\(\)\-_=+\|\{\}\[\]\s:;<>\?\/]+/gi;
str  = str.replace(emailPattern, " ");
str  = str.replace(urlPattern, " ");
str  = str.replace(numberOrSymbolPattern, " ");
str  = str.replace(/▼/g," ");
str  = str.replace(/[\n\f\r\t]/g, " ");
str  = str.replace(/\s\s+/g, " ");


/** untuk menghilangkan tanda baca **/
var hilangtandabaca = str.replace(/[.,:;'",?]/g,"");
return hilangtandabaca;
```

Gambar 5.10 Menghilangkan *patern*

Gambar 5.10 diatas merupakan potongan kode javascript yang digunakan untuk menghilangkan seluruh *pattern* yang tidak

Gambar 5.8 diatas merupakan potongan kode javascript yang digunakan untuk mendapatkan seluruh isi dari website, yang nantinya akan digunakan oleh proses selanjutnya.

5.2.2.4. Proses menghilangkan *tag* dan *entity* HTML serta *patern* yang tidak penting dan tanda baca

Proses ini dilakukan agar *content* website yang didapatkan benar-benar *content* website tersebut, sehingga diharapkan nantinya tidak mengganggu proses pengklasifikasian, akibat banyaknya data-data yang tidak diperlukan dalam pengklasifikasian tersimpan.

- Proses menghilangkan *tag* dan *entity* HTML

```
** untuk menghilangkan tag-tag HTML pada content website **/
   appcontent = appcontent.replace(/&(lt|gt);/g, function (strMatch, p1) {
                return (p1 == "lt")? "<" : ">";});

* untuk menghilangkan entity HTML pada content website **/
   appcontent = replaceHtmlEntites(appcontent);
```

Gambar 5.9 Proses menghilangkan *tag* dan *entity HTML*

Gambar 5.9 diatas merupakan potongan kode javascript yang digunakan untuk menghilangkan seluruh *tag* dan *entity* dari html agar kata yang dikirimkan ke *server* hanya berupa kata-kata yang memiliki arti saja dan tidak merusak proses klasifikasi.

- Proses menghilangkan *patern* yang tidak penting dan tanda baca

```
    }

* untuk mendapatkan kata sesudah dari array awal **/
    for(var b = index+1; b <= contentArray.length-1; b++) {
        if(b == contentArray.length-1) {
            strAfter += contentArray[b];
        } else {
```

4.2. Studi Literatur dan Pengumpulan Data

Studi literatur dan pengumpulan data dilakukan untuk memahami proses yang terjadi dalam aplikasi, misalnya konsep algoritma *porter stemmer*, *script* yang digunakan untuk membuat *interface* aplikasi berbasis *addons* serta yang digunakan untuk membangun *logic* yang berjalan dalam sistem aplikasi tersebut. Studi literatur yang berkaitan dengan konsep algoritma *porter stemmer*, XUL, dan *Javascript* telah dijelaskan pada tinjauan pustaka.

4.3. Analisa Kebutuhan

Tahap perencanaan dapat dilakukan jika analisa terhadap kebutuhan sudah dipenuhi. Analisa kebutuhan yang dilakukan meliputi kebutuhan pengguna, kebutuhan fungsi, dan kebutuhan teknologi.

4.3.1. Pengguna

Aplikasi *inline translator* bertujuan untuk membantu pengguna internet khususnya pengguna internet yang menggunakan *browser* Mozilla firefox.

4.3.2. Fungsi

Dari studi literatur dan pengumpulan data yang dilakukan, dibuat suatu daftar kebutuhan para pengguna terkait dengan sistem aplikasi *add-ons inline translator* ini. Kebutuhan tersebut antara lain:

1. Pengguna dapat memperoleh terjemahan kata yang dipilih secara otomatis.
2. Pengguna dapat memperoleh hasil terjemahan kata tersebut berupa terjemahan yang *case sensitive*.

3. Pengguna dapat memperoleh hasil terjemahan kata tersebut berupa terjemahan dalam dua pilihan yaitu terjemahan dengan kategori dan terjemahan tanpa kategori.

Berdasarkan kebutuhan pengguna tersebut, maka selanjutnya akan dibuat daftar kebutuhan fungsi-fungsi dari sistem yang akan dibuat. Fungsi-fungsi yang dapat diberikan oleh sistem kepada pengguna antara lain:

1. Menampilkan hasil terjemahan secara otomatis dan *case sensitive*.
2. Menampilkan hasil terjemahan dengan kategori atau tanpa kategori.

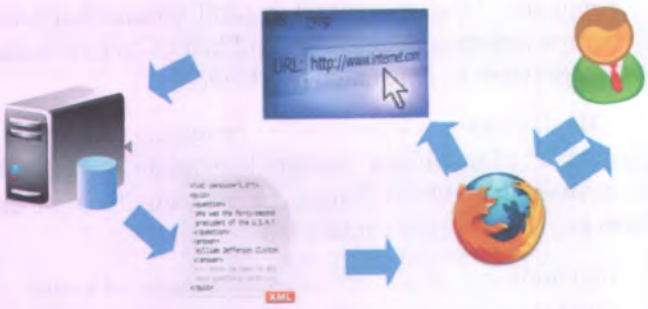
4.3.3. Teknologi

Spesifikasi sistem dan *tool* yang akan digunakan dalam pembuatan aplikasi *inline translator* ini adalah:

a. Sistem operasi: Windows Seven
b. IDE : *Netbeans* 6.8, dan *Notepad* ++
c. IDE *Plugin*: *foxbeans*
d. Bahasa Pemrograman : *Javascript*, XUL, XML
e. Desain : *Enterprise Architect*
f. *Browser* : *Mozilla firefox* 3.6.13

4.4. Arsitektur Sistem

Pada bagian ini dijelaskan tentang arsitektur sistem dari aplikasi *inline translator* yang akan dibuat. Aplikasi ini berhubungan dengan *webservice* yang menyediakan layanan penerjemah, dan menggunakan XML dalam lalu lintas data antara *client* dengan *server*.



Gambar 4.1 Arsitektur Sistem Aplikasi *inlinetrans*

Gambar 4.1 mengilustrasikan tentang arsitektur sistem dari aplikasi *inline translator* dari pengguna yang meminta informasi terjemahan dengan memanfaatkan aplikasi *addons inline translator,* sampai *client* mendapatkan informasi yang diberikan oleh *server* untuk ditampilkan kepada pengguna.

4.5. Use Case Diagram

Perencanaan use case diagram pada aplikasi *inline translator* merupakan gambaran fungsi dari fungsi-fungsi utama yang tersedia pada aplikasi tersebut sehingga *use case* diagram dapat dikatakan sebagai diagram yang berisi kumpulan *use case* dan aktor beserta *relationship*-nya. *Use case* diagram menggambarkan bagaimana *user* akan berinteraksi dengan sistem dan bagaimana sistem tersebut akan merespon aksi yang diminta oleh aktor.

Aktor adalah entitas yang berinteraksi secara langsung dengan sistem. aktor yang terlibat pada sistem aplikasi *inline translator* yaitu pengguna aplikasi (*end user*). Pengguna aplikasi *inline translator* ini dapat melakukan penerjemahan kata yang terdapat pada halawan *website* berbahasa asing (bahasa inggris), selain itu pengguna dapat melihat hasil dari terjemahan kata yang telah dihasilkan oleh sistem.

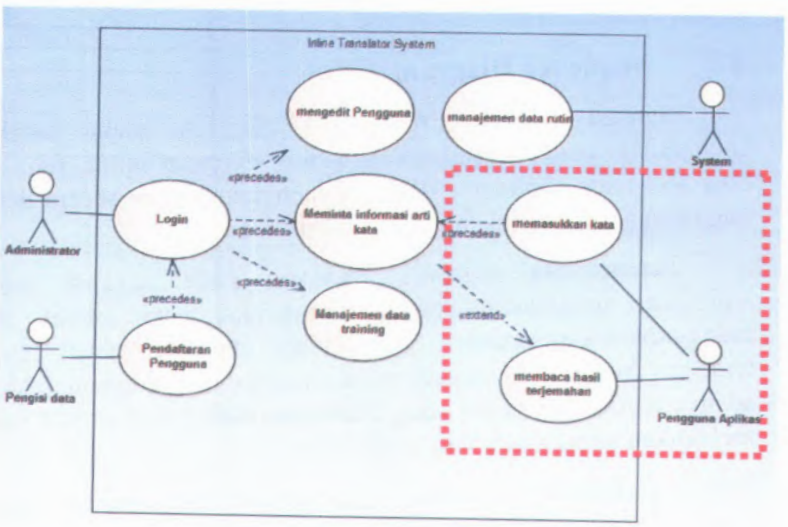
Use case dari sistem yang akan dibuat, merupakan *usecase* yang menampilkan keselurahan *usecase* yang ada pada sistem *inline translator* baik itu *server* maupun *client*, hanya saja pada tugas akhir ini akan dijelaskan mengenai *usecase* apa saja yang merupakan *usecase* yang terdapat pada *client*. Berikut ini merupakan *usecase* yang terdapat pada *client* :

1. Memasukkan kata

Usecase ini merupakan *usecase* yang berhubungan dengan pengelolaan kata yang diinputkan oleh pengguna aplikasi.

2. Membaca hasil terjemahan

Usecase ini merupakan *usecase* yang berhubungan dengan pengelolaan kata untuk menampilkan hasil terjemahan dari respon yang diminta.



Gambar 4.2 Diagram *use case* pada *client*

Gambar 4.2 memperlihatkan *use case* pada sistem aplikasi *inline translator*. *use case* yang diberi tanda merah merupakan *use case* yang ada pada *client*. Deskripsi *use case* dari aplikasi *inline translator* ini dapat dilihat pada Lampiran A.

4.6. Robustness Diagram

Usecase diubah menjadi desain yang detail, kemudian dari desain tersebut akan dibuat suatu program, selain itu *usecase* perlu dihubungkan dengan objek. Hal inilah yang dilakukan pada saat *robustness analysis*. *Robustness analysis* digambarkan dalam bentuk *robustness diagram*. *Robustness diagram* sebenarnya hampir sama dengan *activity diagram*, hanya saja *robustness* diagram memberikan penjelasan *use case* secara lebih detail yang tujuannya untuk menyempurnakan teks *use case* dan model objek.

Robustness diagram dari aplikasi *inline translator* ini dapat dilihat pada Lampiran B.

4.7. Sequence Diagram

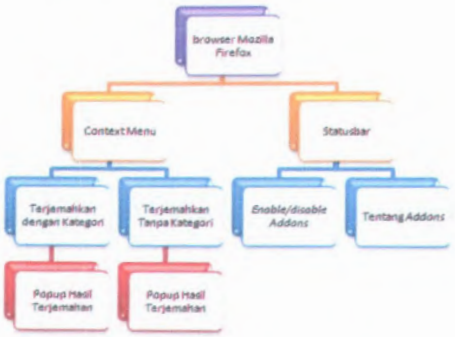
Setelah membuat *robustness diagram,* maka langkah selanjutnya adalah membuat *sequence diagram* untuk tiap *use case* yang ada. *Sequence diagram* menggambarkan secara detail bagaimana *use case* akan diimplementasikan.

Perencanaan *sequence diagram* pada aplikasi *inline translator* merupakan gambaran interaksi antar obyek dan mengindikasikan adanya komunikasi diantara obyek-obyek tersebut. Selain itu, *sequence diagram* ini dapat menunjukkan adanya pertukaran pesan yang dilakukan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu.

Semua pertukaran informasi yang terjadi pada aplikasi *inline translator* dapat dilihat pada *sequence diagram inline* yang terlampir pada lampiran C.

4.8. Desain Antarmuka

Desain antar muka aplikasi *inline translator* ini cukup sederhana, karena pengguna aplikasi hanya akan melihat *pop up* hasil terjemahan pada *browser*. Pengguna hanya diberi tawaran pada *preferences* (preferensi) untuk menampilkan hasil terjemahan dari semua kategori yang ada atau terjemahan tanpa kategori. Berikut ini merupakan sekilas gambaran desain antar muka aplikasi *client inline translator.*



Gambar 4.3 Desain antarmuka

Gambar 4.3 merupakan sekilas gambaran desain antar muka aplikasi *client inline translator* yang terdiri dari *context menu dan status bar,* untuk mengetahui detail antar muka pada aplikasi inline translator akan dijelaskan pada bab selanjutnya.

4.9. Desain Test case

Tahap terakhir dalam perancangan aplikasi *inline translator* adalah membuat desain *test case*. *Test case* adalah suatu langkah yang dilakukan untuk mengetahui apakah aplikasi dapat berjalan sesuai dengan rancangan yang diharapkan. *Test case* yang dilakukan untuk menguji aplikasi *inline translator* ini meliputi tes input, kondisi eksekusi, dan hasil yang diharapkan.

Tes ini untuk membuktikan jalannya sistem, seperti yang telah ditulis pada tiap *use case*, telah diimplementasikan secara benar. Desain *test case* dari aplikasi *inline translator* dapat dilihat di Lampiran D.

BAB V
IMPLEMENTASI DAN UJI COBA SISTEM

Bab ini akan menjelaskan tentang implementasi perangkat lunak yang berdasarkan racangan atau desain yang telah dibuat sebelumnya. Bab ini akan menjelaskan hal-hal yang berkaitan dengan implementasi yang meliputi setting atau konfigurasi yang perlu dilakukan dalam membangun aplikasi *inline translator,*serta penjelasan mengenai *source code* yang dibuat dalam aplikasi *inline translator* . Selain itu, pelaksanaan skenario uji coba juga dijelaskan dalam bab ini.

5.1. Lingkungan Implementasi

Dalam proses pembuatannya, sistem aplikasi *inline translator* ini dikembangkan dengan menggunakan piranti keras *Notebook*. Spesifikasi lingkungan implementasi perangkat keras yang dilakukan pada pengembangan sistem aplikasi dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras dan Sistem Operasi untuk Implementasi Sistem

Aplikasi	Perangkat	Spesifikasi
inline translator	Notebook	AMD Turion (tm) X2 Dual-core Mobile RM-77 2.30GHz
		Memori : 2 GB of RAM
		Sistem Operasi : Windows Seven

Selain itu untuk implementasi pada lingkungan piranti lunak, aplikasi *inline translator* ini dikembangkan dengan

menggunakan bahasa pemrograman *Javascript*. Selain itu, dalam melakukan *coding* menggunakan piranti lunak (editor) utama yang digunakan adalah editor *Notepad* ++ dan netbeans 6.8. Teknologi lain yang digunakan, selengkapnya tersaji dalam Tabel 5.2 berikut ini.

Tabel 5.2 Teknologi yang Digunakan untuk Implementasi Sistem

Aplikasi	Teknologi	Versi
inline translator	Bahasa Pemograman	*Javascript*, XML, XUL
	Editor	Notepad++, Netbeans 6.8
	Browser	Mozilla firefox 3.6.10

5.2. Implementasi Teknologi Web Aplikasi Fungsionalitas Tambahan (*addons*)

Sub bab ini akan menjelaskan mengenai proses implementasi teknologi web aplikasi fungsionalitas tambahan (*add-ons*). Penjelasan implementasi aplikasi *add-ons inline translator* dibagi berdasarkan fungsi dalam sistem, seperti implementasi antar muka aplikasi *add-ons*, implementasi pengolahan kata, dan implementasi lalu lintas data.

5.2.1 Implementasi Antar Muka *Add-ons*

Bagian ini menjelaskan implementasi antar muka *add-ons* yang sebagian besar menggunakan XUL sebagai dasar implementasi antar muka pada aplikasi *inline translator* ini.

5.2.1.1. Antar muka instalasi

Implementasi antarmuka instalasi merupakan antarmuka yang akan tampil saat aplikasi *addons* pertama kali diinstal oleh pengguna.

```
<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:em="http://www.mozilla.org/2004/em-rdf#">
  <Description about="urn:mozilla:install-manifest">
    <em:id>inlinetrans@teamdeveloper.org</em:id>
    <em:name>Indonesia Inline Translator</em:name>
    <em:version>1.0</em:version>
    <em:creator>inlinetrans team developer</em:creator>
    <em:description>pilih kata dan terjemahkan secara otomatis!</em:description>
    <em:aboutURL>chrome://inlinetrans/content/about.xul</em:aboutURL>
    <em:iconURL>chrome://inlinetrans/skin/imagesOuBesar.png</em:iconURL>
    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a1a9e97384}</em:id>
        <em:minVersion>1.0</em:minVersion>
        <em:maxVersion>3.6.*</em:maxVersion>
      </Description>
    </em:targetApplication>
    <em:localized>
        <Description>
                <em:locale>en-US</em:locale>
                <em:name>Indonesia Inline Translator</em:name>
                <em:description>tinggal klik dan terjemahankan secara otomatis!</em:description>
        </Description>
    </em:localized>
  </Description>
</RDF>
```

Gambar 5.1 Antar muka *instalasi Inlinetrans*

Gambar 5.1 diatas merupakan potongan kode xul yang digunakan untuk membangun antar muka saat proses instalasi dilakukan.

5.2.1.2. Antar muka context menu

Implementasi antar muka *context menu* merupakan menu yang ditampilkan saat pengguna selesai melakukan pemilihan kata dan melakukan aksi berupa klik kanan untuk menerjemahkan kata yang dipilih. Saat pengguna melakukan klik menu ini maka pada layar komputer akan muncul terjemahan dari kata yang dipilih sesuai dengan penyetingan aplikasi.

```
<popup id="contentAreaContextMenu">
    <menuseparator />
    <menuitem id="inlinetransContextMenuPageCat"
     label="Terjemahkan dengan Kategori"
     image="chrome://inlinetrans/skin/imagesOn.png"
     class="menuitem-iconic"
     hidden="false"
     oncommand="inlinetrans.processCat();"/>
     <menuitem id="inlinetransContextMenuPageNonCat"
     label="Terjemahkan Tanpa Kategori"
     image="chrome://inlinetrans/skin/imagesOn.png"
     class="menuitem-iconic"
     hidden="false"
     oncommand="inlinetrans.processNonCat();"/>
```

Gambar 5.2 Antar muka *Context Menu Inlinetrans*

Gambar 5.2 diatas merupakan potongan kode xul yang digunakan untuk membangun antar muka *context menu* saat pengguna melakukan klik kanan pada kata yang di blok. Ada dua antar muka yang dapat di lihat oleh pengguna yaitu terjemahkan dengan kategori dan terjemahkan tanpa kategori.

5.2.1.3. Antar muka statusbar

Implementasi antar muka *statusbar* merupakan antar muka yang dapat ditemukan pada sudut kanan bawah suatu *browser*. Antar muka ini fungsinya untuk mengetahui status aplikasi yang tertanam dalam *browser*, apakah aplikasi tersebut dalam keadaan aktif atau non aktif. Selain itu, melalui *statusbar* pengguna dalam melakukan penyetingan preferensi. Ada beberapa antar muka pada *statusbar* ini yaitu :

5.2.1.3.1 *Statusbar* sebelum diklik kanan.

Antar muka *statusbar* sebelum di klik kanan dapat digunakan sebagai informasi bagi pengguna jika aplikasi (*addons*) telah tertanam dalam *browser*, sehingga pengguna dapat

menggunakan aplikasi tersebut sesuai petunjuk penggunaan aplikasi.


```
<!-- status bar -->

  <statusbar id="status-bar">
    <statusbarpanel id="inlinetrans-status-bar"
    context="inlinetransContextMenu"
    label="inlinetrans"
    tooltiptext="addons dalam kondisi aktif"
    image="chrome://inlinetrans/skin/imagesOn.png"
    class="statusbarpanel-iconic-text"
    status="enabled"
    onclick="overlay.clickIcon(event);"
    >

    </statusbarpanel>
  </statusbar>
```

Gambar 5.3 *Statusbar* sebelum diklik kanan

Gambar 5.3 diatas merupakan potongan kode xul yang digunakan untuk membangun antar muka *status bar* sebelum pengguna melakukan klik kanan pada *status bar*. Selain itu potongaan kode tersebut juga digunakan untuk membangun antar muka *status bar* apabila pengguna menekan *icon* untuk menyalakan atau mematikan aplikasi *addons*.

5.2.1.3.2 *Statusbar* setelah diklik kanan.

Antar muka *statusbar* setelah diklik kanan dapat membantu pengguna untuk mengaktifkan maupun tidak mengaktifkan *addons* tanpa perlu membuka menu *tools* → *addons* yang terdapat pada *browser mozilla firefox.*

Gambar 5.4 *Statusbar* setelah diklik kanan

Gambar 5.4 diatas merupakan potongan kode xul yang digunakan untuk membangun antar muka *status bar* saat pengguna melakukan klik kanan pada *icon*. Ada dua antar muka yang dapat di lihat oleh pengguna yaitu status aplikasi *addons* apakah dalam kondisi aktif atau tidak aktif kemudian antar muka yang berisi informasi aplikasi.

5.2.1.4. Antar muka informasi aplikasi

Implementasi antar muka informasi aplikasi merupakan antar muka yang akan muncul saat pengguna melakukan klik menu *about.* antar muka ini menyediakan segala informasi yang berkaitan dengan aplikasi.

Gambar 5.5 antar muka informasi aplikasi

Gambar 5.5 diatas merupakan potongan kode xul yang digunakan untuk membangun antar muka saat pengguna memilih menu *about*. Pengguna dapat memperoleh informasi terkait cara menggunakan aplikasi, dan sebagainya.

5.2.1.5. Antar muka hasil terjemahan

Implementasi antar muka hasil terjemahan merupakan antar muka yang akan muncul saat aplikasi menampilkan hasil terjemahan yang diinginkan oleh pengguna, antar muka hasil terjemahan berupa *pop up*.

5.2.1.6. Antar muka peringatan

Implementasi antar muka peringatan merupakan antar muka yang akan muncul saat pengguna tidak melakukan pengeblokan pada halama website, antar muka peringatan berupa *pop up*.

5.2.2. Implementasi Pengolahan Kata

Bagian ini menjelaskan implementasi dalam sistem yang berhubungan dengan pengolahan kata dalam sistem aplikasi *inline translator*. Ada beberapa bagian penting dalam sistem yang berkaitan dengan pengelolaan kata, seperti proses pemilihan kata,

proses mendapatkan *content website,* proses menghilangkan kata-kata yang frekuensi kemunculannya tinggi, dan proses stemming.

5.2.2.1. Proses mendapatkan kata yang di pilih

Proses awal pengolahan kata di mulai saat pengguna melakukan pemilihan kata dengan cara memblok kata yang ingin diketahui terjemahannya dalam hal ini arti kata bahasa Indonesia dari kata tersebut. Berikut ini merupakan fungsi yang digunakan untuk mendapatkan kata yang ingin diterjemahkan.

```
/** untuk mendapatkan kata yang di blok **/
userSelection = document.commandDispatcher.focusedWindow.getSelection().toString();
```

Gambar 5.6 Proses mendapatkan kata

Gambar 5.6 diatas merupakan potongan kode javascript yang digunakan untuk mendapatkan nilai dari kata yang di pilih oleh pengguna sebagai informasi bagi sistem untuk melangkah ke proses selanjutnya.

5.2.2.2. Proses mendapatkan posisi kata yang di pilih

Proses mendapatkan posisi kata yang di pilih dilakukan saat *user* melakukan pengeblokan kata, proses ini dilakukan untuk mendapatkan posisi kata yang tepat sesuai dengan posisi kata yang di blok. Tujuan dari proses ini yaitu untuk mendapatkan kata-kata yang letaknya sebelum dan sesuadah kata yang di blok dengan tepat, apabila terdapat kata yang sama dengan kata yang diblok pada baris atau pun paragraph yang berbeda.

```
/** untuk mendapatkan range kata yang di blok serta mengubah content kata yang di blok**/
range = document.commandDispatcher.focusedWindow.getSelection().getRangeAt(0);
range.deleteContents();
textNode = document.createTextNode(word);
range.insertNode(textNode);


/** untuk mengembalikan content website yang dipilih ke bentuk awal **/
        range.selectNode(textNode);
        range.deleteContents();
        range.insertNode(document.createTextNode(userSelection));
        userSelection = " ";
```

Gambar 5.7 Proses mendapatkan posisi kata

Gambar 5.7 diatas merupakan potongan kode javascript yang digunakan untuk mendapatkan posisi dari kata yang di pilih oleh pengguna agar memudahkan sistem untuk mendapatkan kata-kata yang letaknya berada sebelum dan sesudah kata yang di pilih dengan tepat.

5.2.2.3. Proses mendapatkan seluruh *content* website

Proses mendapatkan *content* website dilakukan setelah *user* melakukan pengeblokan kata, proses ini dilakukan untuk mendapatkan seluruh *content* website agar mudah melakukan proses mendapatkan kata-kata yang letaknya sebelum dan sesudah kata yang di blok.

```
** untuk mendapatkan content website yang ada di dalam body **/
 var appcontent = content.document.getElementsByTagName("body");
 appcontent = appcontent[0].innerHTML;
```

Gambar 5.8 Proses mendapatkan seluruh *content website*

Gambar 5.8 diatas merupakan potongan kode javascript yang digunakan untuk mendapatkan seluruh isi dari website, yang nantinya akan digunakan oleh proses selanjutnya.

5.2.2.4. Proses menghilangkan *tag* dan *entity* HTML serta *patern* yang tidak penting dan tanda baca

Proses ini dilakukan agar *content* website yang didapatkan benar-benar *content* website tersebut, sehingga diharapkan nantinya tidak mengganggu proses pengklasifikasian, akibat banyaknya data-data yang tidak diperlukan dalam pengklasifikasian tersimpan.

- Proses menghilangkan *tag* dan *entity* HTML

```
** untuk menghilangkan tag-tag HTML pada content website **/
   appcontent = appcontent.replace(/&(lt|gt);/g, function (strMatch, p1) {
                    return (p1 == "lt")? "<" : ">";});

* untuk menghilangkan entity HTML pada content website **/
   appcontent = replaceHtmlEntites(appcontent);
```

Gambar 5.9 Proses menghilangkan *tag* dan *entity HTML*

Gambar 5.9 diatas merupakan potongan kode javascript yang digunakan untuk menghilangkan seluruh *tag* dan *entity* dari html agar kata yang dikirimkan ke *server* hanya berupa kata-kata yang memiliki arti saja dan tidak merusak proses klasifikasi.

- Proses menghilangkan *patern* yang tidak penting dan tanda baca

```
/** untuk menghilangkan beberapa patern yang dianggap tidak penting **/
 var str = appcontent.replace(/<[^>]+(>|$)/g, "");
 var emailPattern = /[_a-zA-Z0-9.]+@[.a-zA-Z0-9]+.[a-zA-Z]+/gi;
 var urlPattern = /[a-z]+://[^ s]+/gi;
 var numberOrSymbolPattern = /[0-9.,'##$%^&*()_~_-= +|\()[] s:;<>?/]+/gi;
 str  = str.replace(emailPattern, " ");
 str  = str.replace(urlPattern, " ");
 str  = str.replace(numberOrSymbolPattern, " ");
 str  = str.replace(/▼/g," ");
 str  = str.replace(/[ n f r t]/g, " ");
 str  = str.replace(/ s s+/g, " ");


/** untuk menghilangkan tanda baca **/
 var hilangtandabaca = str.replace(/[.(:;'",?]/g,"");
 return hilangtandabaca;
```

Gambar 5.10 Menghilangkan *patern*

Gambar 5.10 diatas merupakan potongan kode javascript yang digunakan untuk menghilangkan seluruh *pattern* yang tidak penting dan tanda baca dari seluruh isi website agar tidak ada pola-pola selain kata yang dikirimkan ke *server* agar tidak merusak proses klasifikasi.

5.2.2.5. Proses mendapatkan kata-kata yang mengikuti

Proses ini dilakukan untuk membantu proses klasifikasi yang dilakukan oleh *webservice* agar hasil terjemahan yang dihasilkan oleh aplikasi ini *context sensitive.*

- Proses mendapatkan kata sebelum dan sesudah dalam array

```
** untuk mendapatkan kata sebelum dari array awal **/
    for(var a = 0; a < index; a++) {
            if(a == index-1) {
                    strBefore += contentArray[a];
            } else {
                    strBefore += contentArray[a]+" ";
            }
    }

* untuk mendapatkan kata sesudah dari array awal **/
    for(var b = index+1; b <= contentArray.length-1; b++) {
            if(b == contentArray.length-1) {
                    strAfter += contentArray[b];
            } else {
                    strAfter += contentArray[b]+" ";
            }
    }
```

Gambar 5.11 Proses mendapatkan kata sebelum dan sesudah

Gambar 5.11 diatas merupakan potongan kode javascript yang digunakan untuk mendapatkan seluruh kata yang posisinya berada sebelum dan sesudah kata yang di pilih dari seluruh isi website.

- Proses mendapatkan kata sebelum dan sesudah sebanyak 15 kata sebelum dan 15 kata sesudah kata yang di pilih

```
/** untuk menyimpan kata-kata mengikuti dalam bentuk array **/
      strBefore = strBefore.split(" ");
      strAfter = strAfter.split(" ");

/** untuk mendapatkan kata sebelum sesuai dengan batas yang dikehendaki **/
       if (strBefore.length <= 15){
            strBefore = strBefore;
           }
           else{
            strBefore = strBefore.splice(strBefore.length-15,strBefore.length);
           }
          //  alert("kata sebelum:" +strBefore);

/** untuk mendapatkan kata sesudah sesuai dengan batas yang dikehendaki **/
       if (strAfter.length <= 15){
            strAfter = strAfter;
           }
           else{
            strAfter = strAfter.splice(0,15);
           }
```

Gambar 5.12 Proses mendapatkan 15 kata sebelum dan 15 sesudah

Gambar 5.12 diatas merupakan potongan kode javascript yang digunakan untuk mendapatkan kata yang posisinya berada sebelum dan sesudah kata yang di pilih dari seluruh isi website. tetapi hanya 15 kata yang posisinya berada sebelum dan sesudah kata yang di pilih untuk membantu proses klasifikasi. Proses ini di lakukan setelah proses mendapatkan seluruh kata yang posisinya berada sebelum dan sesudah kata yang di pilih.

5.2.2.6. Proses menghilangkan kata-kata yang frekuensi kemunculannya sering

Proses ini tujuannya tidak jauh berbeda dengan proses yang dilakukan sebelumnya yaitu untuk membantu proses klasifikasi yang dilakukan oleh *webservice* agar hasil klasifikasi dapat tepat dilakukan.

```
/** untuk menghilangkan kata-kata yang frekuensi kemunculannya sering **/
    for(var j=0; j<kata.length; j++) {
            var regex = new RegExp(kata[j],"gi");
            kataptg = kataptg.replace(regex," ");
    }
```

Gambar 5.13 Proses menghilangkan kata yang frekuensi kemunculannya sering

Gambar 5.13 diatas merupakan potongan kode javascript yang digunakan untuk menghilangkan kata-kata yang frekuensi kemunculannya sering agar tidak merusak proses klasifikasi. Proses ini di lakukan setelah proses mendapatkan 15 kata yang posisinya berada sebelum dan sesudah kata yang di pilih.

5.2.2.7. Proses menghasilkan akar kata

Proses ini tujuannya untuk menghasilkan akar kata, proses ini dilakukan untuk mendapatkan akar kata dari kata yang dipilih serta akar kata dari kata-kata yang letaknya sebelum dan sesudah kata yang dipilih.

```
** untuk mendapatkan hasil stem dari kata yang dipilih dan kata yang mengikuti *
    hasilStemSel = stemmer(userSelection);
   // alert(hasilStemSel);

    hasilStem = stemmer(kataptg);
   // alert(hasilStem);
```

Gambar 5.14 Proses menghasilkan akar kata

Gambar 5.14 diatas merupakan potongan kode javascript yang digunakan untuk melakukan proses *stemming* dari kata-kata yang posisinya sebelum dan sesudah kata yang di pilih dan kata yang di pilih oleh pengguna.

5.2.3. Implementasi Lalu Lintas Data

Bagian ini menjelaskan implementasi dalam sistem yang berhubungan dengan lalu lintas data dalam sistem pada aplikasi *inline translator*. lalu lintas data yang terjadi dibagi menjadi dua proses yaitu proses mengirim *request* ke *server* dan proses membaca respon yang di dapat dari *server*.

5.2.3.1. Proses mengirim *request* ke *server*

Proses ini dilakukan saat *client* mengirimkan informasi berupa data-data. Ada dua kategori dalam melakukan pengiriman *request* ke *server* yaitu apabila pengguna memilih menerjemahkan dengan kategori dan apabila pengguna memilih menerjemahkan tanpa menggunakan kategori. Jika pengguna memilih menerjemahkan menggunakan kategori maka kata-kata yang akan di kirim ke *server* adalah kata yang dipilih oleh pengguna dan kata yang mengikuti setelah dilakukan proses *stemming*, dan kata yang di pilih pengguna sebelum dilakukan proses *stemming*. Sedangkan jika pengguna memilih menerjemahkan tanpa menggunakan kategori maka kata yang akan di kirim ke *server* adalah kata yang di pilih oleh pengguna tanpa melalui proses *stemming*.

- Proses mengirimkan *request* ke *server* jika pengguna memilih menerjemahkan menggunakan kategori.

```
var url = "http://sbit.ts.its.ac.id:8080/inlinetransproject/translate/"+userSelection+"/"+hasilStemSel+"/"+hasilStem;
// alert(url);
xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET",url,false);
xmlhttp.send();
```

Gambar 5.15 Proses mengirim *request* ke *server*-terjemahan dengan kategori

Gambar 5.15 diatas merupakan potongan kode javascript yang digunakan untuk melakukan proses pengiriman permintaan kepada server untuk mendapatkan informasi terjemahan. Pengiriman permintaan tersebut dengan cara menggunakan url dalam prosesnya sesuai dengan tipe terjemahan yang di minta oleh pengguna. Parameter yang dikirimkan ke server terdiri dari 3 kata, yaitu kata yang di pilih, kata yang di pilih setelah proses *stemming*, dan kata yang mengikuti setelah dilakukan proses *stemming*.

- Proses mengirimkan *request* ke *server* jika pengguna memilih menerjemahkan menggunakan kategori.

```
var url = "http://ebiz.is.its.ac.id:8080/inlinetransproject/translate/"+userSelection;
//alert(url);


xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET",url,false);
xmlhttp.send();
```

Gambar 5.16 Proses mengirim *request* ke *server*-terjemahan tanpa kategori

Gambar 5.16 diatas merupakan potongan kode javascript yang digunakan untuk melakukan proses pengiriman permintaan kepada server untuk mendapatkan informasi terjemahan. Pengiriman permintaan tersebut dengan cara menggunakan url dalam prosesnya sesuai dengan tipe terjemahan yang di minta oleh pengguna yaitu tanpa kategori, sehingga parameter yang di kirimkan cukup satu saja.

5.2.3.2. Proses membaca XML hasil *respond* dari *server*

Proses ini dilakukan saat *client* menerima informasi berupa data-data hasil terjemahan dari kata-kata yang dikirimkan sebelumnya ke *server*. Ada dua kategori dalam menerima XML yaitu apabila pengguna memilih menerjemahkan dengan kategori

dan apabila pengguna memilih menerjemahkan tanpa menggunakan kategori. Jika pengguna memilih menerjemahkan menggunakan kategori maka XML yang diterima oleh *client* strukturnya seperti pada gambar 5.17 dan apabila pengguna memilih menerjemahkan tanpa menggunakan kategori maka struktur XML yang didapatkan oleh *client* seperti yang tertera pada gambar 5.18.

- Struktur XML jika pengguna memilih menerjemahkan menggunakan kategori.

```
−<translate>
    <category>komputer</category>
    <result>perkembangan </result>
  </translate>
```

Gambar 5.17 Struktur XML menerjemahkan dengan kategori

Gambar 5.17 diatas merupakan potongan kode XML yang di dapat oleh sistem *client* yang digunakan sebagai sumber informasi yang akan ditampilkan oleh aplikasi sesuai dengan permintaan pengguna. Struktur XML yang di dapat oleh *client* tergantuk dari kata yang di pilih oleh pengguna. Jika pengguna memilih menggunakan kategori dalam proses penerjemahan maka server akan mengirim XML yang berisi *category* dan *result*.

- Struktur XML jika pengguna memilih menerjemahkan tanpa kategori.

```
−<translate>
    <result>pengembangan</result>
  </translate>
```

Gambar 5.18 Struktur XML menerjemahkan tanpa kategori

Gambar 5.18 diatas merupakan potongan kode XML yang di dapat oleh sistem *client* yang digunakan sebagai sumber informasi yang akan ditampilkan oleh aplikasi sesuai dengan permintaan pengguna. Struktur XML yang di dapat oleh *client* tergantuk dari kata yang di pilih oleh pengguna. Jika pengguna memilih tidak menggunakan kategori dalam proses penerjemahan maka server akan mengirim XML yang berisi *result*.

- Proses membaca XML jika pengguna memilih menerjemahkan menggunakan kategori.

```
xmlDoc = xmlhttp.responseXML;

var readXML = xmlDoc.getElementsByTagName("translate");
for (i=0;i<readXML.length;i++)
{
var kategori = readXML[i].getElementsByTagName("category")[0].childNodes[0].nodeValue;
var hasil = readXML[i].getElementsByTagName("result")[0].childNodes[0].nodeValue;
}
}
```

Gambar 5.19 Proses membaca XML-terjemahan dengan kategori

Gambar 5.19 diatas merupakan potongan kode javascript yang digunakan untuk melakukan proses membaca hasil permintaan dari server jika pengguna memilih menerjemahkan menggunakan kategori.

- Proses membaca XML jika pengguna memilih menerjemahkan tanpa kategori.

```
xmlDoc = xmlhttp.responseXML;

var readXML = xmlDoc.getElementsByTagName("translate");
for (i=0;i<readXML.length;i++)
{
var hasil = readXML[i].getElementsByTagName("result")[0].childNodes[0].nodeValue;
}
}
```

Gambar 5.20 Proses membaca XML-terjemahan tanpa kategori

Gambar 5.20 diatas merupakan potongan kode javascript yang digunakan untuk melakukan proses membaca hasil permintaan dari server jika pengguna tidak memilih menerjemahkan menggunakan kategori.

5.3. Uji Coba Sistem Aplikasi

Pada sub bab ini akan dijelaskan mengenai uji coba aplikasi sistem. Uji coba yang dilakukan diawali dengan menyusun skenario uji coba kemudian pelaksanaan dari skenario uji coba tersebut.

5.3.1. Lingkungan Uji Coba

Pada bagian ini akan dijelaskan mengenai lingkungan uji coba dari aplikasi *inline translator*, yang meliputi perangkat lunak dan perangkat keras. Spesifikasi dari masing-masing perangkat yang digunakan dalam uji coba ini tampak pada tabel-tabel berikut:

Tabel 5.3 Lingkungan Uji Coba I Aplikasi Inline Translator

Perangkat Keras	Prosesor : Intel Dual Core T4300 2,10GHz Memori : 2 GB of RAM Modem : smart ZTE AC2726 EV-DO Kec 153Kbps
Perangkat Lunak	Sistem Operasi : Windows XP SP 3. Web Browser: Mozilla Firefox Version 3.6.13

Tabel 5.3 diatas merupakan lingkungan uji coba aplikasi dengan menggunakan modem smart dengan kecepatan 153 Kbps untuk koneksi internet, dan akses server melalui *localhost*.

Tabel 5.4 Lingkungan Uji Coba II Aplikasi Inline Translator

Perangkat Keras	Prosesor : Intel Dual Core T4300 2,10GHz Memori : 2 GB of RAM Modem : Speedy Kec 100Mbps
Perangkat Lunak	Sistem Operasi : Windows XP SP 3. Web Browser: Mozilla Firefox Version 3.6.13

Tabel 5.4 diatas merupakan lingkungan uji coba aplikasi dengan menggunakan modem speedy dengan kecepatan 100 Mbps untuk koneksi internet, dan akses server melalui *localhost*.

Tabel 5.5 Lingkungan Uji Coba III Aplikasi Inline Translator

Perangkat Keras	Prosesor : AMD Turion (tm) X2 Dual-core Mobile RM-77 2.30GHz Memori : 2 GB of RAM Koneksi Internet : Wifi

Perangkat Lunak	Sistem Operasi : Windows 7 Web Browser: Mozilla Firefox Version 3.6.13

Tabel 5.5 diatas merupakan lingkungan uji coba aplikasi dengan menggunakan koneksi internet *wifi* untuk koneksi internet,dan akses server melalui http://ebiz.is.its.ac.id:8080/inlinetransproject/translate/.

Tabel 5.6 Lingkungan Uji Coba IV Aplikasi Inline Translator

Perangkat Keras	Prosesor : AMD Turion (tm) X2 Dual-core Mobile RM-77 2.30GHz Memori : 2 GB of RAM Koneksi Internet : smart ZTE AC2726 EV-DO Kec 153Kbps
Perangkat Lunak	Sistem Operasi : Windows 7 Web Browser: Mozilla Firefox Version 3.6.13

Tabel 5.6 diatas merupakan lingkungan uji coba aplikasi dengan menggunakan jaringan LAN untuk koneksi internet, dan akses server melalui http://ebiz.is.its.ac.id:8080/inlinetransproject/translate/.

Tabel 5.7 Lingkungan Uji Coba V Aplikasi Inline Translator

Perangkat Keras	Prosesor : AMD Turion (tm) X2 Dual-core Mobile RM-77 2.30GHz Memori : 2 GB of RAM Koneksi Internet : LAN (proxy ITS)
Perangkat Lunak	Sistem Operasi : Windows 7

	Web Browser: Mozilla Firefox Version 3.6.13

Tabel 5.7 diatas merupakan lingkungan uji coba aplikasi dengan menggunakan modem smart dengan kecepatan 153 Kbps untuk koneksi internet, dan akses server melalui http://ebiz.is.its.ac.id:8080/inlinetransproject/translate/.

5.3.2. Skenario Uji Coba

Uji coba dilakukan untuk menguji jalannya aplikasi *inline translator*, mulai dari proses input sampai dengan proses output. Selain itu pengujian dilakukan untuk mngetahui kesesuaian aplikasi dengan *browser* yang digunakan. Uji coba dilakukan berdasarkan desain test case yang telah dibuat sebelumnya yaitu pada tahap desain aplikasi.

Aplikasi *inline translator* dikatakan lulus uji coba apabila fitur yang dipilih untuk uji coba bisa berjalan dengan baik sesuai dengan yang tercantum dalam *narrative use case*.

- Uji coba I

  Uji coba dilakukan menggunakan laptop dengan sistem operasi *windows* 7, *browser mozilla firefox* versi 3.6.13 dan menggunakan modem *smart* EV-DO dengan kecepatan 153 Kbps untuk koneksi internet.

- Uji coba II

  Uji coba dilakukan menggunakan laptop dengan sistem operasi *windows* XP SP 3, *browser mozilla firefox* versi 3.6.13 dan menggunakan modem *speedy* dengan kecepatan 100 Mbps untuk koneksi internet.

- Uji coba III

  Uji coba dilakukan menggunakan laptop dengan sistem operasi *windows* 7, *browser mozilla firefox* versi 3.6.13 dan menggunakan *wifi* untuk koneksi internet.

- Uji coba IV

  Uji coba dilakukan menggunakan laptop dengan sistem operasi *windows* 7, *browser mozilla firefox* versi 3.6.13 dan menggunakan modem *smart* EV-DO dengan kecepatan 153 Kbps untuk koneksi internet.

- Uji coba V

  Uji coba dilakukan menggunakan laptop dengan sistem operasi *windows* 7, *browser mozilla firefox* versi 3.6.13 dan menggunakan *LAN* untuk koneksi internet.

Kelima uji coba tersebut dilakukan untuk mengetahui apakah aplikasi *inline translator* dapat berjalan dengan baik jika lingkungan uji coba berbeda-beda. Uji coba-uji coba tersebut juga digunakan untuk mengetahui waktu respon dari aplikasi *inline translator* saat pengguna meminta informasi terjemahan.

Selain ketiga uji coba diatas dilakukan pula uji coba untuk mengetahui kecocokan aplikasi dengan *browser mozilla firefox* yang telah di instal terlebih dahulu oleh pengguna.

- Uji coba VI

  Uji coba dilakukan menggunakan laptop dengan *browser mozilla firefox* versi 3.6.13

- Uji coba VII

  Uji coba dilakukan menggunakan laptop dengan *browser mozilla firefox* versi 3.0

- Uji coba VIII

  Uji coba dilakukan menggunakan laptop dengan *browser mozilla firefox* versi 3.7

5.3.3. Uji Coba Fungsional

Pelaksanaan uji coba terhadap *inline translator* dilakukan berdasarkan skenario yang telah disusun sebelumnya. Kemudian diberikan penjelasan tentang pelaksanaan uji coba yang akan disertai dengan gambar keadaan sistem yang sebenarnya. Seluruh hasil dari pelaksanaan uji coba untuk menguji jalannya aplikasi dapat dilihat pada lampiran E. Dan hasil uji coba kinerja SMS gateway terdapat pada lampiran F.

BAB VI
PENUTUP

Pada bab ini berisi kesimpulan dari seluruh proses pengerjaan tugas akhir beserta saran untuk proses pengembangan selanjutnya.

6.1. Kesimpulan

Berdasarkan hasil penelitian tugas akhir yang dilakukan, maka dapat disimpulkan beberapa hal sebagai berikut:

- Implementasi dalam pengembangan aplikasi *addons inline translator* sudah berhasil di lakukan. Aplikasi *addons inline translator* sudah berhasil dibuat dengan menggunakan *javascript* dan XUL.
- Dari hasil uji kinerja aplikasi *addons inline translator* dapat diketahui waktu respon untuk menerjemahkan dengan menggunakan kategori lebih kurang 40 detik, sedangkan jika menerjemahkan tanpa menggunakan kategori waktu respon yang dibutuhkan kurang lebih 2 detik jika mengakses server menggunakan *localhost* dan waktu respon untuk menerjemahkan dengan menggunakan kategori lebih kurang 8 detik, sedangkan jika menerjemahkan tanpa menggunakan kategori waktu respon yang dibutuhkan kurang lebih 2 detik jika mengakses server menggunakan http://ebiz.is.its.ac.id:8080/inlinetransproject/translate/.
- Kelemahan dari aplikasi *addons inline translator* ini dalam hal GUI yaitu tampilan hasil keluaran dari terjemahan masih menggunakan fungsi *alert.*
- Kelemahan dari aplikasi *addons inline translator* ini yaitu dalam menentukan posisi kata yang di blok masih dilakukan dengan cara mengganti isi dari kata yang di blok dengan sesuatu yang unik, sehingga pengguna tidak dapat melihat kata asli saat proses terjemahan dilakukan, sehingga untuk mengatasi hal tersebut sistem menampilkan kata asli tersebut

beserta kategori dan hasil terjemahan dari kata-kata yang di blok sebelumnya.

- Dari hasil uji kinerja aplikasi *addons inline translator* dapat diketahui bahwa kompatibelitas aplikasi ini telah sesuai dengan batasan minimal dan maksimal yang telah ditentukan dalam proses *code* yaitu versi minimal *browser Mozilla firefox* yaitu 3.0 dan versi maksimal *browser Mozilla firefox 3.6.**. Sehingga, jika pengguna ingin melakukan penginstalan aplikasi *inline translator* ini harus menggunakan *browser* yang memiliki versi *browser* tidak lebih kecil dari versi 3 dan tidak lebih besar dari versi 3.6.*.

6.2. Saran

Beberapa hal yang diharapkan dapat dikembangkan pada masa mendatang adalah sebagai berikut:

- Pada penelitian selanjutnya, dapat dilakukan implementasi untuk menerjemahkan isi website yang menggunakan bahasa Indonesia ke bahasa Inggris menggunakan algoritma Nazief dan Adriani.
- Pada penelitian selanjutnya, dapat dilakukan eksplorasi lebih lanjut dengan memanfatkan *engine babelfish, yahoo* dan sebagainya untuk mendapatkan database terjemahan kata dalam berbagai bahasa, misalnya bahasa Jerman ke bahasa Indonesia, bahasa Belanda ke Bahasa Indonesia, dan sebagainya.
- Pada penelitian selanjutnya, dapat dilakukan eksplorasi lebih lanjut dengan memanfaatkan algoritma porter stemmer untuk bahasa Denmark, Belanda, Inggris, Finlandia, Prancis, Jerman, Hungaria, Italia, Norwegia, Portugis, Rumania, Rusia, Spanyol, Swedia, dan Turki.

DAFTAR PUSTAKA

Anonym. 2008. Javascript. *URL:http://id.wikipedia.org/wiki/javascript*. 6 Desember 2009.

Anonym. 2008. XUL. *URL:http://en.wikipedia.org/wiki/xul.* 6 Desember 2009.

Anonym.2008.Javascript. *URL:https://developer.mozilla.org/en/About_JavaScript*. 6 Desember 2009.

Anonym.2008.Mozilla add-ons. *URL:http://en.wikipedia.org/wiki/Mozilla(add-ons).* 6 Desember 2009.

Anonym. 2009. Sejarah internet.*URL: http://www.sejarah-internet.com/perkembangan-internet*. 6 Desember 2009.

Anonym. 2008. Penjelajah_web. *URL:http://id.wikipedia.org/wiki/Penjelajah_web.* 6 Desember 2009.

Kusuma, Dyah wardhani. 2007. Stemming dengan Algoritma Porter Stemmer. *URL:http://blog.its.ac.id/dyah03tc/category/course-materials*. 20 Januari 2010.

Mozilla Developer Center. 2010. XUL. *https://developer.mozilla.org/En/XUL.* 20 Januari 2010.

Mozilla Developer Center. 2010. Javascript. *https://developer.mozilla.org/En/XUL.* 20 Januari 2010.

Rosenberg, Dough and Matt Stephens. 2007. *Use case driven object modeling with UML. Apress.* 20 Mei 2010

LAMPIRAN A
NARRATIVE USE CASE

Pada lampiran ini akan ditampilkan naratif use case sesuai dengan desain use case diagram sistem aplikasi *inline translator*.

Package: Memasukkan Kata

Tabel A.0.1 Narrative Use Case Memasukkan Kata

Use Case Name:	memasukkan kata	
Scenarios:		
Basic Path	basicPath	pengguna aplikasi membuka halaman website, kemudian melakukan setting preferensi dengan cara memilih opsi apakah menerjemahkan dengan kategori atau menerjemahkan tidak menggunakan kategori. setelah melakukan penyetingan preferensi kemudian pengguna aplikasi melakukan pemilihan kata dengan cara memblok kata yang ingin diterjemahkan, apabila pengguna aplikasi ingin mendapatkan hasil terjemahan dengan kategori maka sistem akan mengambil kata yang di

	pilih serta mendapatkan kata yang letaknya berada sebelum dan sesudah kata yang di pilih akan tetapi kata-kata yang berada sebelum dan sesudah kata yang di pilih tersebut akan dilakukan proses penghilangan kata-kata yang frekuensi kemunculannya sering, kemudian baik kata-kata yang letaknya sebelum dan sesudah kata yang di pilih serta kata yang di pilih akan dilakukan proses stemming lalu kata-kata tersebut akan di kirim oleh sistem ke webservice sedangkan jika pengguna ingin mendapatkan hasil terjemahan tanpa kategori maka sistem akan mengambil kata yang di pilih saja kemudian kata yang ingin diterjemahkan tersebut akan di kirim oleh sistem ke webservice.

Alternate	input kosong	apabila pengguna tidak memasukkan (memilih atau memblok) kata maka proses tidak akan dilanjutkan, sistem akan menampilkan peringatan, maaf anda harus terlebih dahulu memblok kata.
Alternate	input berupa angka	apabila pengguna memasukkan (memilih atau memblok) kata berupa angka maka sistem akan menampilkan terjemahan sesuai kata yang di pilih yaitu *output* berupa angka.
Alternate	input berupa simbol	apabila pengguna memasukkan (memilih atau memblok) kata berupa angka maka sistem akan menampilkan terjemahan sesuai kata yang di pilih yaitu *output* berupa simbol.
Alternate	input tidak jelas (emoticon,smiley,dsb)	apabila pengguna memasukkan (memilih atau memblok) kata berupa angka maka sistem akan menampilkan terjemahan sesuai kata yang di pilih yaitu *output* berupa emoticon, smiley,

	dan sebagainya.

Package: Membaca Hasil Terjemahan

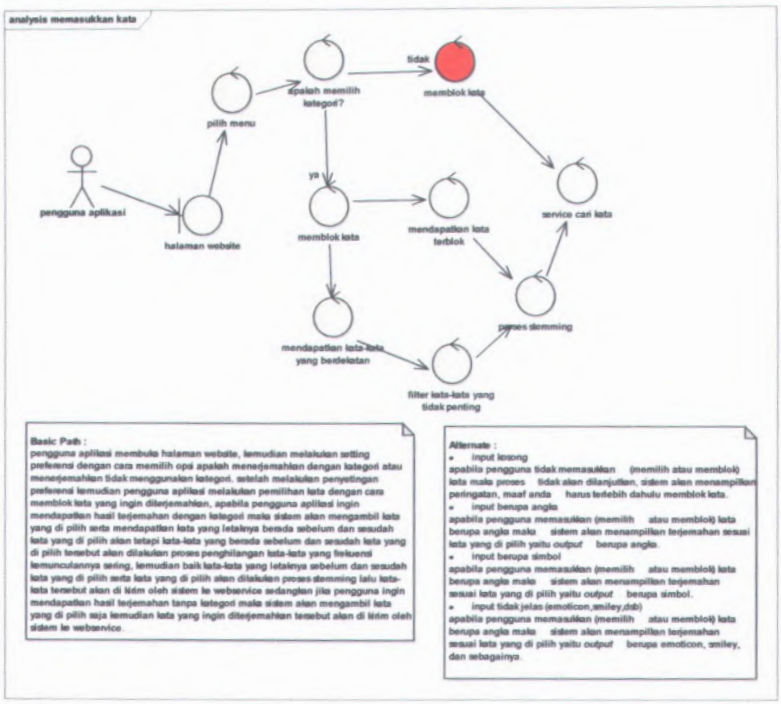
Tabel A.0.2 Narrative Use Case Membaca Hasil Terjemahan

Use Case Name:	membaca hasil terjemahan	
Scenarios:		
Basic Path	basic path	web service akan mengirimkan permintaan dari sistem dalam bentuk XML, kemudian sistem akan menerima hasil terjemahan tersebut dan membaca hasil terjemahan yang berupa xml untuk kemudian ditampilkan di halaman website sehingga pengguna aplikasi dapat mengetahui hasil terjemahan yang diinginkan.
Alternate	hasil terjemahan dengan kategori	sistem akan menampilkan terjemahan sesuai dengan opsi yang diminta oleh pengguna yaitu terjemahkan dengan opsi kategori

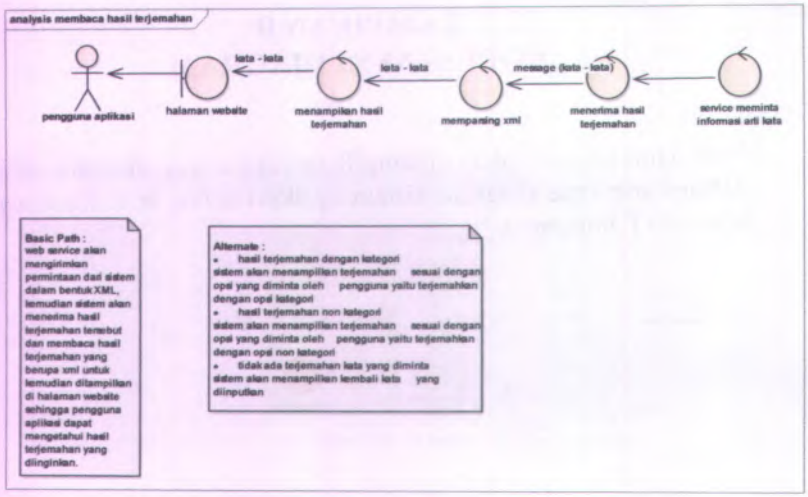
Alternate	hasil terjemahan non kategori	sistem akan menampilkan terjemahan sesuai dengan opsi yang diminta oleh pengguna yaitu terjemahkan dengan opsi non kategori
Alternate	tidak ada terjemahan kata yang diminta	sistem akan menampilkan kembali kata yang diinputkan

LAMPIRAN B
ROBUSTNESS DIAGRAM

Pada lampiran ini akan ditampilkan robustness diagram sesuai definisi use case diagram sistem aplikasi *inline translator* yang ada pada Lampiran A.



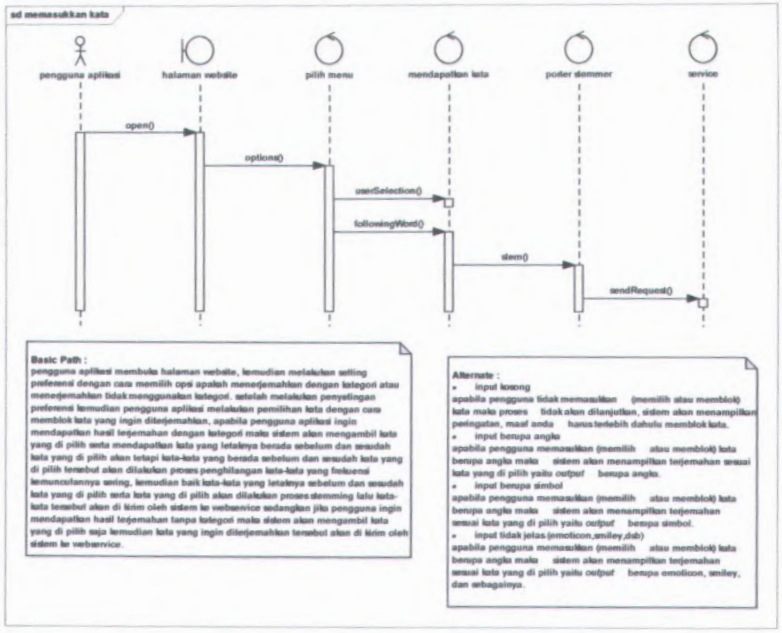
Gambar B.1 Robustness Diagram Memasukkan Kata



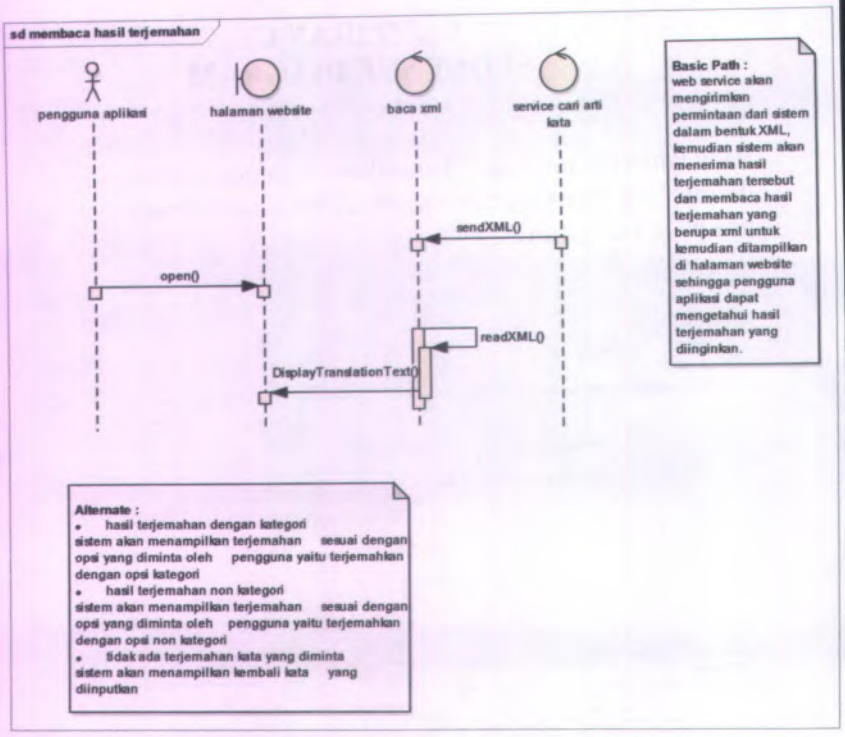
Gambar B.2 Robustness Diagram Membaca Hasil Terjemahan

LAMPIRAN C
SEQUENCE DIAGRAM

Pada lampiran ini akan ditampilkan sequence diagram sesuai definisi use case diagram sistem aplikasi *inline translator* yang ada pada Lampiran A.



Gambar C.1 Sequence Diagram Memasukkan Kata



Gambar C.2 Sequence Diagram Membaca Hasil Terjemahan

LAMPIRAN D
TEST CASE

Pada lampiran ini akan ditampilkan desain test case sesuai dengan definisi use case diagram dari sistem aplikasi *inline translator* yang terdapat pada Lampiran A.

Step 1 : Generate Scenarios

Tabel D.1 Use Case Scenarios Memasukkan Kata

Use Case Masukkan Kata		
Scenario Name	Starting Flow	Alternate
Scenario 1	Basic Flow	
Scenario 2	Basic Flow	A1
Scenario 3	Basic Flow	A2
Scenario 4	Basic Flow	A3
Scenario 5	Basic Flow	A4

Tabel D.2 Partial Scenario Matrix Use Case Memasukkan Kata

Use Case Masukkan Kata		
Scenario Name	Starting Flow	Alternate
Scenario 1 – Memasukkan kata berhasil	Basic Flow	
Scenario 2 - input/memilih kata berupa simbol	Basic Flow	Memblok simbol pada halaman web
Scenario 3 - input/memilih kata berupa angka	Basic Flow	Memblok angka pada halaman web
Scenario 4 - input/memilih kata yang tidak jelas (berupa emoticon, smiley, dan sebagainya)	Basic Flow	Memblok kata-kata tidak jelas (misal, emoticon, smiley,dan sebagainya) pada halaman web
Scenario 5 - input kosong	Basic Flow	Tidak ada kata yang diblok

Tabel D.3 Use Case Scenarios Membaca Hasil Terjemahan

Use Case Masukkan Kata		
Scenario Name	Starting Flow	Alternate
Scenario 1	Basic Flow	
Scenario 2	Basic Flow	A1
Scenario 3	Basic Flow	A2
Scenario 4	Basic Flow	A3

Tabel D.4 Partial Scenario Matrix Use Case Membaca Hasil Terjemahan

Use Case Masukkan Kata		
Scenario Name	Starting Flow	Alternate
Scenario 1 – berhasil menerjemahkan kata	Basic Flow	
Scenario 2 - hasil terjemahan dengan kategori	Basic Flow	Menerima hasil terjemahan dengan

		kategori dan menampilkannya pada halaman web
Scenario 3 - hasil terjemahan non kategori	Basic Flow	Menerima hasil terjemahan tanpa kategori dan menampilkannya pada halaman web
Scenario 4 - tidak ada terjemahan kata yang diminta	Basic Flow	Menerima hasil terjemahan berupa kata yang sama persis dengan kata yang di kirim ke server dan menampilkannya pada halaman website

Step 2 : Identify Test Case

Tabel D.5 Identify Test Case Memasukkan Kata

Use Case: Memasukkan Kata			
Test Case ID	Scenario/Condition	Input Kata	Expected Result
TC A1	Scenario 1 – Memasukkan kata berhasil	V	Sistem dapat melakukan proses pengolahan kata dan mengirimkan informasi berupa kata yang dimasukkan oleh pengguna ke server dan pengguna mendapatkan hasil terjemahan dari kata yang di blok
TC A2	Scenario 2 - input/memilih kata berupa simbol	I	Sistem tetap melakukan proses pengolahan kata dan mengirimkan informasi berupa simbol ke server hanya saja tidak ada hasil terjemahan yang di dapat, sistem akan

			menampilkan hasil terjemahan sama persis dengan simbol yang dimasukkan
TC A3	Scenario 3 - input/memilih kata berupa angka	I	Sistem tetap melakukan proses pengolahan kata dan mengirimkan informasi berupa simbol ke server hanya saja tidak ada hasil terjemahan yang di dapat, sistem akan menampilkan hasil terjemahan sama persis dengan angka yang dimasukkan
TC A4	Scenario 4 - input/memilih kata yang tidak jelas (berupa emoticon, smiley, dan sebagainya)	I	Sistem tetap melakukan proses pengolahan kata dan mengirimkan informasi berupa simbol ke server hanya saja tidak ada hasil terjemahan yang di dapat, sistem akan menampilkan hasil terjemahan sama persis dengan kata yang tidak jelas (berupa emoticon, smiley, dan

			sebagainya) yang dimasukkan
TC A5	Scenario 5 - input kosong	I	Sistem akan memberikan peringatan, **maaf anda harus terlebih dahulu memblok kata**

Tabel D.6 Identify Test Case Membaca Hasil Terjemahan

Use Case: Membaca Hasil Terjemahan			
Test Case ID	Scenario/Condition	Hasil Terjemahan Kata	Expected Result
TC B1	Scenario 1 – berhasil menterjemahkan kata	V	Sistem menerima informasi yang diberikan oleh server berupa terjemahan dari kata yang di minta oleh pengguna, kemudian server akan melakukan proses parsing xml, dan menampilkan hasil terjemahan kata tersebut dalam

			bentuk pop up kepada pengguna
TC B2	Scenario 2 - hasil terjemahan dengan kategori	I	Sistem menerima informasi yang diberikan oleh server berupa terjemahan dari kata dan opsi yang di minta oleh pengguna, kemudian server akan melakukan proses parsing xml, dan menampilkan hasil terjemahan kata dengan opsi kategori tersebut dalam bentuk pop up kepada pengguna
TC B3	Scenario 3 - hasil terjemahan non kategori	I	Sistem menerima informasi yang diberikan oleh server berupa terjemahan dari kata dan opsi yang di minta oleh pengguna, kemudian server akan melakukan proses parsing xml, dan menampilkan hasil terjemahan kata dengan opsi tanpa kategori tersebut dalam bentuk pop up kepada pengguna

TC B4	Scenario 4 - tidak ada terjemahan kata yang diminta	I	Sistem menerima informasi yang diberikan oleh server berupa kata yang sama persis dengan kata yang diberikan oleh pengguna kemudian sistem akan menampilkan informasi dari server tersebut dalam bentuk pop up kepada pengguna

LAMPIRAN E
HASIL UJI COBA

Pada lampiran ini akan ditampilkan hasil uji coba dari desain test case sistem aplikasi Iinline translator yang terdapat pada Lampiran D.

Tabel E.1 Hasil Uji Test Case Memasukkan Kata

Use Case: Memasukkan Kata				
Test Case ID	Scenario/Condition	Input Kata	Expected Result	Hasil
TC A1	Scenario 1 – Memasukkan kata berhasil	V	Sistem dapat melakukan proses pengolahan kata dan mengirimkan informasi berupa kata yang dimasukkan oleh pengguna ke server dan pengguna mendapatkan hasil terjemahan dari kata yang di blok	Sistem mendapatkan kata yang diblok oleh pengguna.
TC A2	Scenario 2 - input/memilih kata berupa simbol	I	Sistem tetap melakukan proses pengolahan kata dan mengirimkan	Hasil terjemahan nantinya sama seperti yang dimasukkan oleh

			informasi berupa simbol ke server hanya saja tidak ada hasil terjemahan yang di dapat, sistem akan menampilkan hasil terjemahan sama persis dengan simbol yang dimasukkan	pengguna yaitu simbol.
TC A3	Scenario 3 - input/memilih kata berupa angka	I	Sistem tetap melakukan proses pengolahan kata dan mengirimkan informasi berupa simbol ke server hanya saja tidak ada hasil terjemahan yang di dapat, sistem akan menampilkan hasil terjemahan sama persis dengan angka yang dimasukkan	Hasil terjemahan nantinya sama seperti yang dimasukkan oleh pengguna yaitu angka.

E-4

TC A4	Scenario 4 - input/memilih kata yang tidak jelas (berupa *emoticon, smiley*, dan sebagainya)	I	Sistem tetap melakukan proses pengolahan kata dan mengirimkan informasi berupa simbol ke server hanya saja tidak ada hasil terjemahan yang di dapat, sistem akan menampilkan hasil terjemahan sama persis dengan kata yang tidak jelas (berupa emoticon, smiley, dan sebagainya) yang dimasukkan	Hasil terjemahan nantinya sama seperti yang dimasukkan oleh pengguna yaitu kata-kata tidak jelas seperti *emoticon,smiley*, dan sebagainya.
TC A5	Scenario 5 - input kosong	I	Sistem akan memberikan peringatan, **maaf anda harus terlebih dahulu memblok kata**	Muncul peringatan berbunyi **maaf anda harus terlebih dahulu memblok kata**

Tabel E.2 Hasil Uji Test Case Memasukkan Kata

Use Case: Membaca Hasil Terjemahan				
Test Case ID	Scenario/Condition	Hasil Terjemahan Kata	Expected Result	Hasil
TC B1	Scenario 1 – berhasil menterjemahkan kata	V	Sistem menerima informasi yang diberikan oleh server berupa terjemahan dari kata yang di minta oleh pengguna, kemudian server akan melakukan proses parsing xml, dan menampilkan hasil terjemahan kata tersebut dalam bentuk pop up kepada pengguna	Sistem akan menampilkan hasil terjemahan sama persis dengan informasi yang diberikan oleh server.
TC B2	Scenario 2 - hasil terjemahan dengan kategori	I	Sistem menerima informasi yang diberikan oleh server berupa	Sistem akan menampilkan kategori dari kata

			terjemahan dari kata dan opsi yang di minta oleh pengguna, kemudian server akan melakukan proses parsing xml, dan menampilkan hasil terjemahan kata dengan opsi kategori tersebut dalam bentuk pop up kepada pengguna	yang dipilih dan hasil terjemahan dari kata yang di pilih sesuai dengan informasi yang diberikan oleh server.
TC B3	Scenario 3 - hasil terjemahan non kategori	I	Sistem menerima informasi yang diberikan oleh server berupa terjemahan dari kata dan opsi yang di minta oleh pengguna, kemudian server akan melakukan proses parsing xml, dan menampilkan hasil terjemahan kata dengan	Sistem akan menampilkan hasil terjemahan dari kata yang di pilih sesuai dengan informasi yang diberikan oleh server.

			opsi tanpa kategori tersebut dalam bentuk pop up kepada pengguna	
TC B4	Scenario 4 - tidak ada terjemahan kata yang diminta	I	Sistem menerima informasi yang diberikan oleh server berupa kata yang sama persis dengan kata yang diberikan oleh pengguna kemudian sistem akan menampilkan informasi dari server tersebut dalam bentuk pop up kepada pengguna	Sistem akan menampilkan kembali kata yang di pilih, sesuai dengan informasi yang di dapat dari server.

LAMPIRAN F
ANALISA PERFORMA APLIKASI *INLINE TRANSLATOR*

Pada lampiran ini akan ditampilkan hasil uji coba kinerja dari aplikasi *inline translator*.

Tabel F.1 Lingkungan Uji Coba I Aplikasi *Inline Translator*

Perangkat Keras	Prosesor : Intel Dual Core T4300 2,10GHz Memori : 2 GB of RAM Modem : Smart ZTE AC2726 EV-DO Kec 153Kbps
Perangkat Lunak	Sistem Operasi : Windows XP SP 3. Web Browser: Mozilla Firefox Version 3.6.13

Tabel F.2 Performa Aplikasi *Inline Translator* Pada Uji Coba I

Tipe Pilihan	Tipe Input	Waktu Respon
Terjemahkan Dengan Kategori	Kata	40 Detik
	Kalimat	40 Detik
	Paragraf	60 Detik
Terjemahkan Tanpa Kategori	Kata	2 Detik
	Kalimat	2 Detik
	Paragraf	6 Detik

Tabel F.2 menggambarkan performa aplikasi *inline translator* yang telah di uji sesuai dengan lingkungan uji coba yang tertera pada **tabel F.1**. hasil uji coba untuk melakukan penerjemahan menggunakan kategori didapatkan setelah

melakukan uji coba menggunakan *notebook, server* dijalankan pada *notebook* kemudian aplikasi *inline translator* dipasang (*install*) ke dalam Mozilla firefox. Kemudian untuk mengakses internet digunakan modem **Smart ZTE AC2726 EV-DO** dengan kecepatan 153 Kbps. Uji coba aplikasi dilakukan dengan melakukan *request* hasil terjemahan dengan cara mengakses *localhost* sebagai URL. Kemudian dari uji coba tersebut di dapatkan hasil waktu respon seperti yang tertera pada **tabel F.2**, rentang waktu yang lama terjadi karena aplikasi *inline translator* melakukan beberapa proses yang relative panjang, misalnya proses pengklasifikasian selain itu data-data yang digunakan dalam proses peengklasifikasian langsung didapatkan dari *database,* dikarenakan waktu yang diperlukan untuk mengakses *database* relative lama maka akan mempengaruhi waktu respon aplikasi. Sedangkan waktu respon untuk melakukan uji coba menerjemahkan tanpa menggunakan kategori waktu respon relative singkat karena dalam proses menerjemahkan, *server* tidak melakukan proses pengklasifikasian dan melakukan pengecekan ke dalam *database* sehingga respon yang di dapat dari *server* jauh lebih singkat.

Tabel F.3 Lingkungan Uji Coba II Aplikasi *Inline Translator*

Perangkat Keras	Prosesor : Intel Dual Core T4300 2,10GHz Memori : 2 GB of RAM Modem : Speedy Kec 100Mbps
Perangkat Lunak	Sistem Operasi : Windows XP SP 3. Web Browser: Mozilla Firefox Version 3.6.13

Tabel F.4 Performa Aplikasi *Inline Translator* Pada Uji Coba II

Tipe Pilihan	Tipe Input	Waktu Respon
Terjemahkan Dengan Kategori	Kata	40 Detik
	Kalimat	40 Detik
	Paragraf	60 Detik
Terjemahkan Tanpa Kategori	Kata	2 Detik
	Kalimat	2 Detik
	Paragraf	6 Detik

Tabel F.4 menggambarkan performa aplikasi *inline translator* yang telah di uji sesuai dengan lingkungan uji coba yang tertera pada **tabel F.3**. hasil uji coba untuk melakukan penerjemahan menggunakan kategori didapatkan setelah melakukan uji coba menggunakan *notebook, server* dijalankan pada *notebook* kemudian aplikasi *inline translator* dipasang (*install*) ke dalam Mozilla firefox. Kemudian untuk mengakses internet digunakan modem **Speedy Kec 100Mbps** dengan kecepatan 153 Kbps. Uji coba aplikasi dilakukan dengan melakukan *request* hasil terjemahan dengan cara mengakses *localhost* sebagai URL. Kemudian dari uji coba tersebut di dapatkan hasil waktu respon seperti yang tertera pada **tabel F.4**, rentang waktu yang lama terjadi karena aplikasi *inline translator* melakukan beberapa proses yang relative panjang, misalnya proses pengklasifikasian selain itu data-data yang digunakan dalam proses peengklasifikasian langsung didapatkan dari *database,* dikarenakan waktu yang diperlukan untuk mengakses *database* relative lama maka akan mempengaruhi waktu respon aplikasi. Sedangkan waktu respon untuk melakukan uji coba menerjemahkan tanpa menggunakan kategori waktu respon relative singkat karena dalam proses menerjemahkan, *server* tidak melakukan proses pengklasifikasian dan melakukan pengecekan ke dalam *database* sehingga respon yang di dapat dari *server* jauh lebih singkat.

Tabel F.5 Lingkungan Uji Coba III Aplikasi *Inline Translator*

Perangkat Keras	Prosesor : AMD Turion (tm) X2 Dual-core Mobile RM-77 2.30GHz Memori : 2 GB of RAM Koneksi Internet : Wifi
Perangkat Lunak	Sistem Operasi : Windows 7 Web Browser: Mozilla Firefox Version 3.6.13

Tabel F.6 Performa Aplikasi *Inline Translator* Pada Uji Coba III

Tipe Pilihan	Tipe Input	Waktu Respon
Terjemahkan Dengan Kategori	Kata	6 Detik
	Kalimat	6 Detik
	Paragraf	10 Detik
Terjemahkan Tanpa Kategori	Kata	2 Detik
	Kalimat	2 Detik
	Paragraf	5 Detik

Tabel F.6 menggambarkan performa aplikasi *inline translator* yang telah di uji sesuai dengan lingkungan uji coba yang tertera pada **tabel F.5**. hasil uji coba untuk melakukan penerjemahan menggunakan kategori didapatkan setelah melakukan uji coba menggunakan *notebook, server* dijalankan pada *PC (server ebis)* kemudian aplikasi *inline translator* dipasang (*install*) ke dalam Mozilla firefox. Kemudian untuk mengakses internet digunakan *wifi*. Uji coba aplikasi dilakukan dengan melakukan *request* hasil terjemahan dengan cara mengakses *ip server* sebagai URL. Kemudian dari uji coba tersebut di dapatkan hasil waktu respon seperti yang tertera pada

tabel F.6, rentang waktu yang diperlukan untuk mendapatkan hasil terjemahan **relative lebih singkat** jika dibandingkan dengan **dua uji coba sebelumnya** yang membutuhkan waktu relative lama untuk mendapatkan hasil terjemahan, hal itu terjadi karena *server* aplikasi *inline translator* pada **dua uji coba sebelumnya** melakukan beberapa proses yang relative panjang, misalnya proses pengklasifikasian selain itu data-data yang digunakan dalam proses peengklasifikasian langsung didapatkan dari *database,* dikarenakan waktu yang diperlukan untuk mengakses *database* relative lama maka akan mempengaruhi waktu respon aplikasi. Sedangkan uji coba yang ketiga ini lebih singkat waktu respon penerjemahannya karena *server* aplikasi *inline translator* melakukan proses-proses yang panjang tersebut di dalam *server* yang memang *resource*-nya biasa dikhususkan sebagai *server*, selain itu dalam proses penerjemahan tersebut, data-data yang diperlukan tidak langsung di ambil dari *database* melainkan dari *database* semu Sedangkan waktu respon untuk melakukan uji coba menerjemahkan tanpa menggunakan kategori waktu respon penerjemahannya sama dengan uji coba-uji coba sebelumnya, relative singkat karena dalam proses menerjemahkan, *server* tidak melakukan proses pengklasifikasian dan melakukan pengecekan ke dalam *database* sehingga respon yang di dapat dari *server* jauh lebih singkat.

Tabel F.7 Lingkungan Uji Coba IV Aplikasi *Inline Translator*

Perangkat Keras	Prosesor : AMD Turion (tm) X2 Dual-core Mobile RM-77 2.30GHz Memori : 2 GB of RAM Koneksi Internet : Smart ZTE AC2726 EV-DO Kec 153Kbps
Perangkat Lunak	Sistem Operasi : Windows 7 Web Browser: Mozilla Firefox Version 3.6.13

Tabel F.8 Performa Aplikasi *Inline Translator* Pada Uji Coba IV

Tipe Pilihan	Tipe Input	Waktu Respon
Terjemahkan Dengan Kategori	Kata	10 Detik
	Kalimat	10Detik
	Paragraf	12 Detik
Terjemahkan Tanpa Kategori	Kata	2 Detik
	Kalimat	2 Detik
	Paragraf	5 Detik

Tabel F.8 menggambarkan performa aplikasi *inline translator* yang telah di uji sesuai dengan lingkungan uji coba yang tertera pada **tabel F.7**. hasil uji coba untuk melakukan penerjemahan menggunakan kategori didapatkan setelah melakukan uji coba menggunakan *notebook, server* dijalankan pada *PC (server ebis)* kemudian aplikasi *inline translator* dipasang (*install*) ke dalam Mozilla firefox. Kemudian untuk mengakses internet digunakan Smart ZTE AC2726 EV-DO Kec 153Kbps. Uji coba aplikasi dilakukan dengan melakukan *request* hasil terjemahan dengan cara mengakses *server address*. Kemudian dari uji coba tersebut di dapatkan hasil waktu respon seperti yang tertera pada **tabel F.8**, rentang waktu yang diperlukan untuk mendapatkan hasil terjemahan **relative lebih singkat** jika dibandingkan dengan **dua uji coba sebelumnya** yang membutuhkan waktu relative lama untuk mendapatkan hasil terjemahan, hal itu terjadi karena *server* aplikasi *inline translator* pada **dua uji coba sebelumnya** melakukan beberapa proses yang relative panjang, misalnya proses pengklasifikasian selain itu data-data yang digunakan dalam proses peengklasifikasian langsung didapatkan dari *database,* dikarenakan waktu yang diperlukan untuk mengakses *database* relative lama maka akan mempengaruhi waktu respon aplikasi. Sedangkan uji coba yang ketiga ini lebih singkat waktu respon penerjemahannya karena *server* aplikasi *inline translator* melakukan proses-proses yang

panjang tersebut di dalam *server* yang memang *resource*-nya biasa dikhususkan sebagai *server*, selain itu dalam proses penerjemahan tersebut, data-data yang diperlukan tidak langsung di ambil dari *database* melainkan dari *database* semu Sedangkan waktu respon untuk melakukan uji coba menerjemahkan tanpa menggunakan kategori waktu respon penerjemahannya sama dengan uji coba-uji coba sebelumnya, relative singkat karena dalam proses menerjemahkan, *server* tidak melakukan proses pengklasifikasian dan melakukan pengecekan ke dalam *database* sehingga respon yang di dapat dari *server* jauh lebih singkat.

Tabel F.9 Lingkungan Uji Coba V Aplikasi *Inline Translator*

Perangkat Keras	Prosesor : AMD Turion (tm) X2 Dual-core Mobile RM-77 2.30GHz Memori : 2 GB of RAM Koneksi Internet : LAN
Perangkat Lunak	Sistem Operasi : Windows 7 Web Browser: Mozilla Firefox Version 3.6.13

Tabel F.10 Performa Aplikasi *Inline Translator* Pada Uji Coba V

Tipe Pilihan	Tipe Input	Waktu Respon
Terjemahkan Dengan Kategori	Kata	6 Detik
	Kalimat	6 Detik
	Paragraf	10 Detik
Terjemahkan Tanpa Kategori	Kata	2 Detik
	Kalimat	2 Detik
	Paragraf	5 Detik

Tabel F.10 menggambarkan performa aplikasi *inline translator* yang telah di uji sesuai dengan lingkungan uji coba yang tertera pada **tabel F.9**. hasil uji coba untuk melakukan penerjemahan menggunakan kategori didapatkan setelah melakukan uji coba menggunakan *notebook, server* dijalankan pada *PC (server ebis)* kemudian aplikasi *inline translator* dipasang (*install*) ke dalam Mozilla firefox. Kemudian untuk mengakses internet digunakan *wifi*. Uji coba aplikasi dilakukan dengan melakukan *request* hasil terjemahan dengan cara mengakses *server address*. Kemudian dari uji coba tersebut di dapatkan hasil waktu respon seperti yang tertera pada **tabel F.10**, rentang waktu yang diperlukan untuk mendapatkan hasil terjemahan **relative lebih singkat** jika dibandingkan dengan **dua uji coba sebelumnya** yang membutuhkan waktu relative lama untuk mendapatkan hasil terjemahan, hal itu terjadi karena *server* aplikasi *inline translator* pada **dua uji coba sebelumnya** melakukan beberapa proses yang relative panjang, misalnya proses pengklasifikasian selain itu data-data yang digunakan dalam proses peengklasifikasian langsung didapatkan dari *database,* dikarenakan waktu yang diperlukan untuk mengakses *database* relative lama maka akan mempengaruhi waktu respon aplikasi. Sedangkan uji coba yang ketiga ini lebih singkat waktu respon penerjemahannya karena *server* aplikasi *inline translator* melakukan proses-proses yang panjang tersebut di dalam *server* yang memang *resource*-nya biasa dikhususkan sebagai *server*, selain itu dalam proses penerjemahan tersebut, data-data yang diperlukan tidak langsung di ambil dari *database* melainkan dari *database* semu Sedangkan waktu respon untuk melakukan uji coba menerjemahkan tanpa menggunakan kategori waktu respon penerjemahannya sama dengan uji coba-uji coba sebelumnya, relative singkat karena dalam proses menerjemahkan, *server* tidak melakukan proses pengklasifikasian dan melakukan pengecekan ke dalam *database* sehingga respon yang di dapat dari *server* jauh lebih singkat.

Tabel F.11 Lingkungan Uji Coba Kompatibelitas Aplikasi *Inline Translator*

Perangkat Keras	Prosesor : AMD Turion (tm) X2 Dual-core Mobile RM-77 2.30GHz Memori : 2 GB of RAM Koneksi Internet : Wifi
Perangkat Lunak	Sistem Operasi : Windows 7 Web Browser: Mozilla Firefox Version 3.6.13, Mozilla Firefox Version 3.0, Mozilla Firefox Version 3.7

Tabel F.12 Performa Kompatibelitas Aplikasi *Inline Translator* Pada Uji Coba

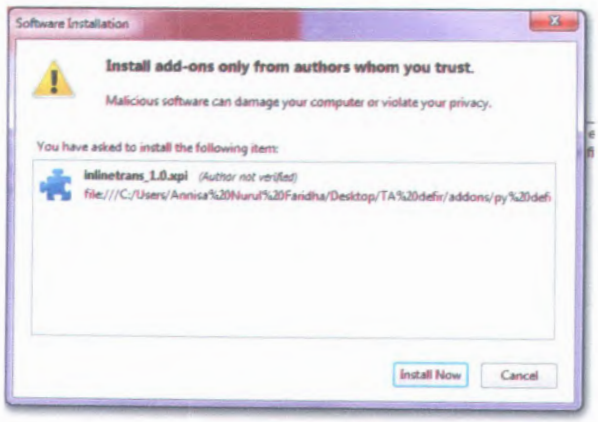
Uji Coba	Kompatibel	Tidak Kompatibel
Uji Coba VI (versi 3.6.13)	v	-
Uji Coba VII (versi 3.0)	v	-
Uji Coba VII (versi 3.7)	-	V

Tabel F.12 menunjukkan performa aplikasi *inline translator* saat dilakukan penginstalan ke dalam beberapa versi *browser Mozilla firefox,* yaitu pada versi 3.6.13, versi 3.0, dan versi 3.7. setelah dilakukan percobaan di atas maka didapatkan hasil seperti ini : aplikasi *inline translator* sukses di-*install* pada percobaan A dan B yang masing-masing menggunakan *browser Mozilla firefox* versi 3.6.7 atau dengan kata lain *browser* tersebut **kompatibel** dan versi 3.0, sedangkan pada percobaan C aplikai *inline translator* tidak dapat di-*instal* atau dengan kata lain *browser* tersebut **tidak kompatibel**. Sehingga pembatasan versi maksimal dan

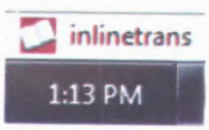
minimal yang dapat melakukan penginstalan aplikasi ini berhasil dilakukan, karena batasan kompatibelitas aplikasi minimal dapat di-*instal* jika pengguna menggunakan *browser Mozilla firefox* versi 3.0 sampai dengan versi 3.6.*.

LAMPIRAN G
TAMPILAN APLIKASI *INLINE TRANSLATOR*

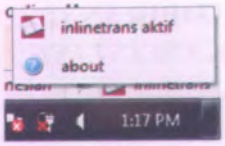
Pada lampiran ini akan ditunjukkan tampilan dari aplikasi *inline translator* yang dihasilkan pada tahap implementasi.



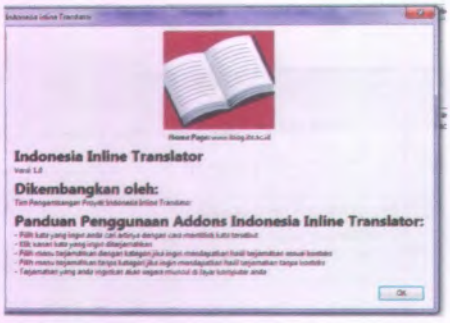
Gambar G.1 proses instalasi aplikasi *inline translator*



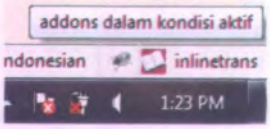
Gambar G.2 tampilan aplikasi *inline translator* saat pertama di instal



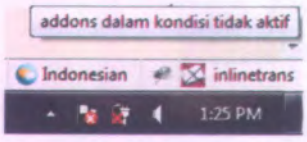
Gambar G.3 status bar aplikasi *inline translator* saat di klik kanan



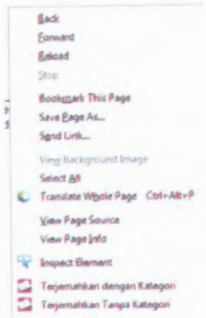
Gambar G.4 Informasi tentang aplikasi *inline translator*



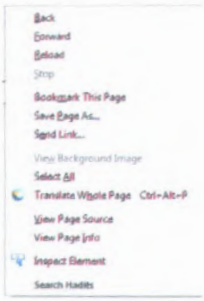
Gambar G.5 status bar aplikasi *inline translator* saat dalam kondisi aktif



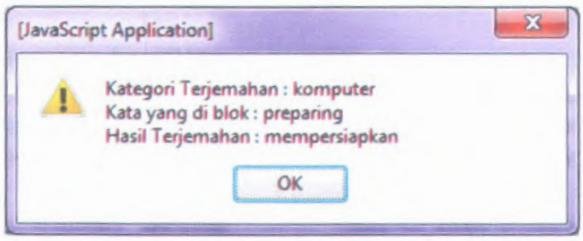
Gambar G.6 status bar aplikasi *inline translator* saat dalam kondisi tidak aktif



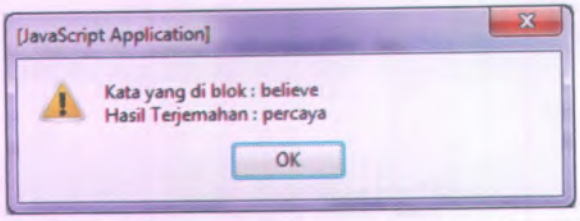
Gambar G.7 *context menu* aplikasi *inline translator* saat dalam kondisi aktif



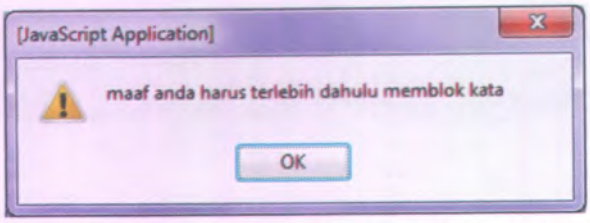
Gambar G.8 *context menu* aplikasi *inline translator* saat dalam kondisi tidak aktif



Gambar G.9 tampilan hasil terjemahan dengan kategori



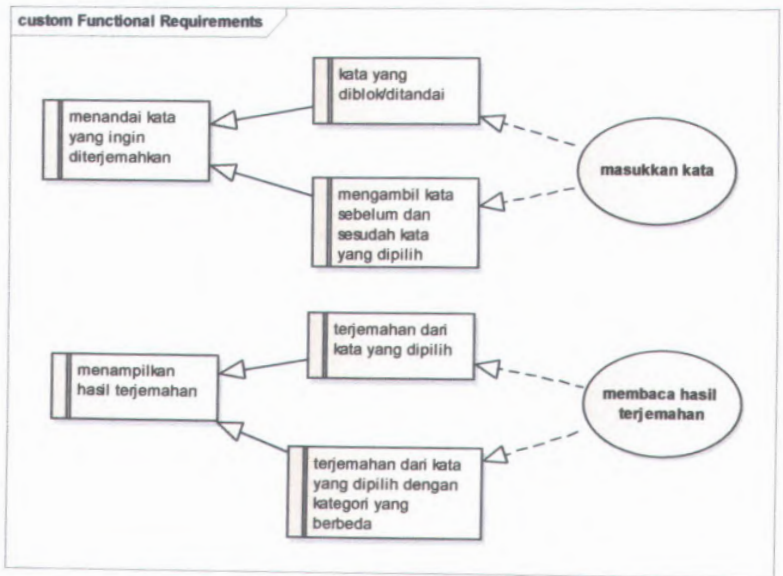
Gambar G.10 tampilan hasil terjemahan tanpa kategori



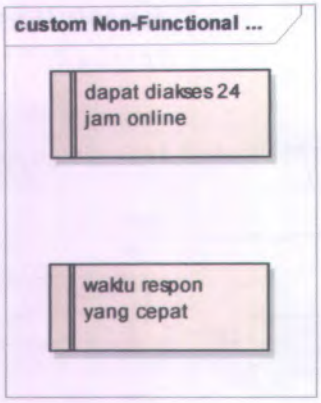
Gambar G.11 tampilan peringatan jika tidak ada kata yang di blok

LAMPIRAN H
REQUIREMENT

Pada lampiran ini akan ditampilkan *requirement* sistem aplikasi *inline translator*.



Gambar H.1 *functional requirement* aplikasi *inline translator*



Gambar H.2 *non functional requirement* aplikasi *inline translator*

RIWAYAT PENULIS

Penulis dilahirkan di Surabaya, 21 Oktober 1988, merupakan anak pertama dari empat bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK Tunas Mekar Surabaya dan TK Aisyah Bustanul Athfal 2 Surabaya pada tingkat taman kanak-kanak, SD Muhammadiyah 6 Surabaya pada tingkat sekolah dasar, SMPN 12 Surabaya pada tingkat menengah pertama, dan SMAN 16 Surabaya pada tingkat atas. Pada tahun 2006 Penulis mengikuti SPMB dan diterima di Jurusan Sistem Informasi FTIf-ITS dan terdaftar dengan NRP 5206100010. Di Jurusan Sistem Informasi ini penulis mengambil bidang minat E-Bisnis.